Installing AASPI Software on Linux (3-25-2014)

Kurt J. Marfurt (kmarfurt@ou.edu)

Software Location

The current version of the AASPI software will be under a website called geology.ou.edu/aaspi/software . This site is password protected with a user name and password for each company. To avoid creation of confusion, this password is known to the person who is the company "champion" for the AASPI consortium and an identified IT contact who has root privileges on your network. People change assignments so please send me an email at kmarfurt@ou.edu if you are the new person or need to know the password. I will normally send out a note to each sponsor (including the IT contact) when a new release has been made.

As of January 2014, we now release two compiled versions of the software to support the Linux (Redhat 5 and above) and Windows (XP, 7, 8, 8.1) platforms. To simplify things, the "uploads" subdirectory has been eliminated. Instead, you will find two new directories under geology.ou.edu/aaspi/software : AASPI_for_Linux and AASPI_for_Windows. Note that the source code for the two implementations is *identical*. Most of the application code is written in FORTRAN2005, the graphical user interfaces (including displays) are written in C++ using the Fox Toolkit (www.fox-toolkit.org), while (when the porting is completed in early 2014) the shell scripts and a few utilities are written in Python. We use the Intel compiler to support the Message Passing Interface (MPI) parallelization libraries on both platforms. To avoid confusion, the source code will only be distributed with the Linux version.

Sponsor Software Modifications and Ownership

Each sponsor has full access to the software source code. While most of our 20+ sponsors use the software in the compiled form, at least five "cherry pick" applications and roll them into their own internal packages. It is perfectly permitted to use external 3rd party technical people to make such modifications for your internal and subsidiary use. While commercial software "sales" of the software in its current state requires a separate agreement, it would be naïve to think a software sponsor would expose themselves to such complicated maintenance problems. It is therefore both reasonable and permitted for a software vendor to recode, modify and improve upon our applications within their framework (say, using C# or Java and their internal data formats) using our software as a template. Furthermore, sponsors retain access and rights to the previously distributed source code if they cease sponsorship for either scientific or business reasons. Details of these issues are contained in the AASPI sponsorship agreement signed by your organization.

Downloading the AASPI software to the ${AASPIHOME} directory

Installing AASPI Software on Linux

Each company installation will have a different naming convention for disk drives. Let's assume that you have a directory called */external_software* . In this case go to this location by typing:

```
cd /external_software
```

First, you will need to *gunzip* and *untar* the AASPI software here by typing

```
tar zxvf AASPI_2014-01-15.tgz
```

(or whatever the current version is).

Alternatively, you may unpack the software with:

```
gunzip –c AASPI_2014-01-15.tgz | tar xvf –
```

This will create the directory *AASPI*. Within the *AASPI* directory there are several subdirectories:

Directory Name	Contents
bin64	Precompiled 64-bit AASPI executables
documentation	A local version of the documentation under http://geology.ou.edu/aaspi/documentation.html
ext_lib64	Precompiled 64-bit support libraries (fftw. Fox Toolkit, intel64, openmpi)
ext_rpm	Source RPMs (Red Hat Package Manager) that may be needed
ext_src	Source archives and fixes for support libraries (fftw, Fox Toolkit, openmpi, SEPlib, SU).
include	Windows include files (plus some AASPI 32-bit include files)
include64	AASPI 64-bit include and Fortran2005 module files.
lib64	AASPI precompiled 64-bit library files
lists	a suite of tables containing attribute names that controls the conversion of AASPI-format to SEGY-format files
maintenance	scripts and instructions to compile and release the software
par	AASPI default parameter files
pyscripts	AASPI Python (*.py) scripts
scripts	AASPI Bourne Shell (*.sh) LSF scripts.
sep_colors	Suite of color bars in SEP-format used by program aaspi_plot
src	The AASPI source code with a subdirectory for each application

in addition to a directory called *boonsville* which contains some public domain data from the UT BEG to test the software.

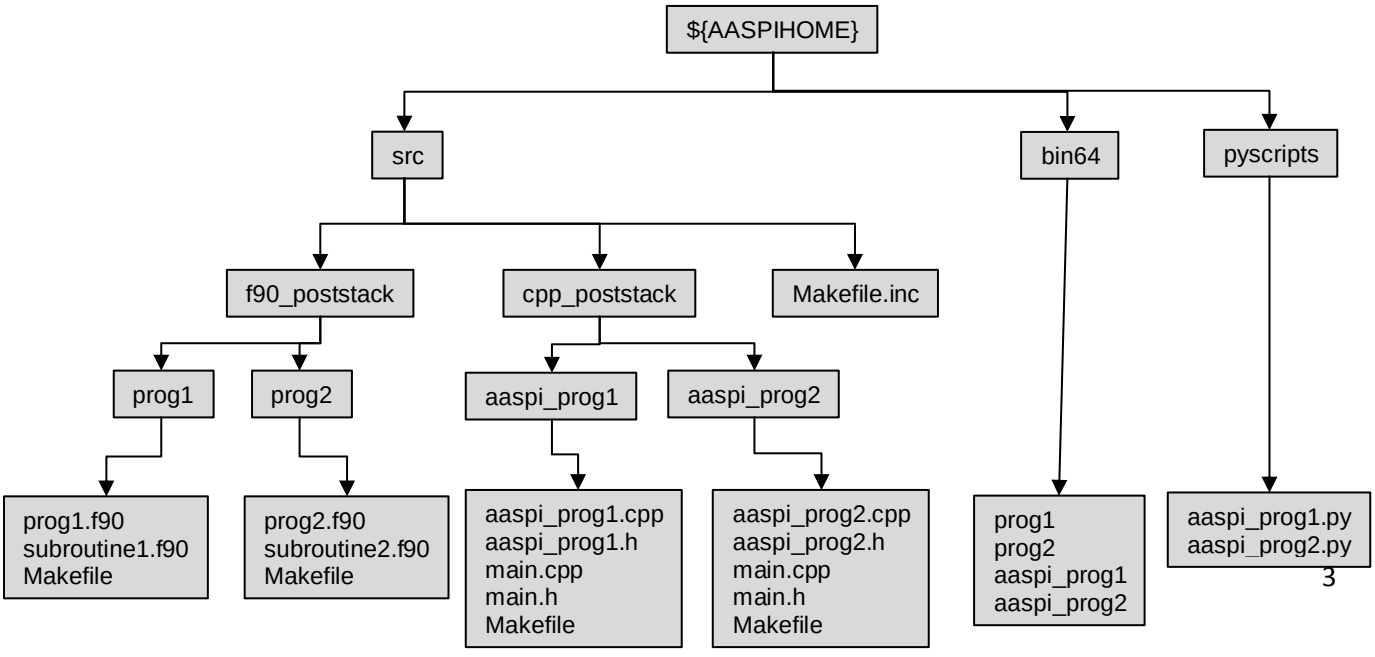
As of May 15th, 2014, the *src* folder has the following structure:

Directory Name	Contents
cpp_lib	AASPI C++ library source codes (aapsi_io, aaspi_gui_lib)
cpp_poststack	AASPI C++ post-stack GUI source codes
cpp_prestack	AASPI C++ pre-stack GUI source codes
f90_lib	AASPI Fortran90 library source codes (lib_new_reorg, or fortran_mod in Windows)
f90_poststack	AASPI Fortran90 post-stack application source codes
f90_prestack	AASPI Fortran90 pre-stack application source codes
linux_only	Other AASPI GUI and application source codes that have not been ported to Windows

Several 3rd party binary libraries are used by the AASPI code: OpenMPI – Message Passing Interface, and FFT-W – the Fastest Fourier Transform in the West.) As of 2014 we have removed all references to the SEP – Stanford Exploration Project, SU – Seismic Unix libraries, since neither of these, nor "Madagascar" will run under Windows. The Fortran90 compiler support libraries *intel64* are also distributed with this distribution. Currently we are using the Intel Fortran Compiler for Linux version 12.1.2 If you intend to use a different Fortran compiler, the source and most of the external libraries will need to be recompiled. We can provide compiled versions using former or future versions of Redhat on request.

The AASPI Software Tree

Although we try to be very explicit in our documentation about algorithm implementation and assumptions, algorithm developers and tech group staff may wish to examine details of how the software has been implemented. The figure below shows a subset of the tree described above.



Under ${AASPIHOME} there are src, bin64, and pyscripts directories. As of May 2014 we are using public domain *git* to control our software development and deployment (http://git-scm.com) giving rise to a the slightly modified tree structure under src as shown above. The Fortran90 source code is found under the f90_poststack and f90_prestack subdirectories. Within each of these are a suite of application programs (e.g. program *dip3d.f90*). These Fortran90 application programs are controlled by a graphical user interface or GUI which are found under the *cpp_poststack* and *cpp_prestack* directories. Our naming convention is simple, by adding the prefix *aaspi_* before each program name (e.g. *aaspi_dip3d.cpp*). Each program has a simple Makefile. These Makefiles include a file called ../../Makefile.inc where all the installation specific paths and external libraries are defined. If you recompile the software you will probably need to modify the *src* level *Makefile.inc* file but *not* any of the program level *Makefile* files. (See optional compilation box below).

Following common programming conventions, after compilation the binary codes are copied to the directory bin64 with names corresponding to the application (e.g. *dip3d* and *aaspi_dip3d*) but *without* the extensions of *.f90* or *.cpp*. Intermediate object and executable code (not shown in the tree above) are stored under an obj64 directory under each application or GUI. The GUIs residing in the bin64 directory are typically invoked from master GUIs (such as *aaspi_util* and *aaspi_util_prestack* but can also be directly invoked by typing the binary file name (e.g. *aaspi_dip3d*). In most cases, each GUI will invoke a python script with a corresponding name (e.g. *aaspi_dip3d.py*) and output a parameter file in the user directory (e.g. *dip3d.parms*). The python script reads in the parameter file, parses the arguments into command line arguments and then executes the corresponding program (e.g. dip3d).

Earlier versions of the AASPI software invoked the application code with command line arguments directly from the GUI. Windows does not support this construct (or Linux style pipes either). The introduction of the parameter file was key to the Linux port. An ancillary benefit is that it also facilitates debugging by decoupling the GUI from the application code.

Recompiling the AASPI software (Optional)

90% of our current sponsors use the AASPI software "as is". Some "cherry pick" key ideas by inserting or reworking AASPI source code into their own application development environment. In either of these two situations you will *not* need to recompile the AASPI software.

Reasons for recompiling may include adding additional capabilities or improvements to the current version of the software, better linkage to your commercial interpretation package, or in at least one environment, recompiling using a machine-specific set of optimized libraries. In our environment, the caretakers of the OU supercomputer center (Oscer) have installed tuned versions of *OpenMPI* , *fftw3*, and *lapack*. Linking to these optimized libraries requires recompiling.

In order to compile you will need to determine the name of your compiler (in our case at OU the Intel compiler is called *ifort64* for 64-bit architecture. The gnu gfortran compiler in general does not support MPI). You will also need to learn the absolute paths of each of the libraries that may be replaced. The only file that needs to be modified is *${AASPIHOME}/src/Makefile.inc* (see tree structure above). You will note that the variable F90 is set to ifort64 at OU for 64 bit compilation. Change it appropriately. More troublesome is determining where your external libraries reside. They will need to be explicitly defined using the *complete path name*, not some local short cut. You may find that the environment variables in your *xterm* window are quite different that when the *Makefile* is invoked. We will print out these variables in the June 2014 release of our software to allow you to check these paths.

Once these variables have been set, change to the compilation directory by typing

cd ${AASPIHOME}/maintenance/deploy_scripts

Then type

python make_aaspi_linux.py

The *make_aaspi_linux.py script will compile programs defined by two files: aaspi_gui_list* and *aaspi_application_list.* If you add new programs and wish to distribute to another site you may wish to augment these lists.

Copying and unzipping the Boonsville test data volume

Finally, we will wish to install some test code. We have chosen the Boonsville Survey. This is a public domain survey acquired by the Texas Bureau of Economic Geology in the Fort Worth Basin, TX, U.S.A. If you ever choose to publish any results from this survey, you will need to properly acknowledge them. Obviously, data volumes belong in a very different storage area than source code. In my case I will have some scratch space on a disk called /scr1/users/kmarfurt.

I will therefore *cd* to this spot, create a project directory called boonsville, and untar it, by
typing:

```
cd /scr1/users/kmarfurt

mkdir boonsville

cd boonsville

tar xvf boonsville_test_data.tar
```

While we are at it, let's create a directory to include the binary AASPI-format output files. (We explain this format and how to run the code in accompanying documentation). For now, let's create a directory to the side of the boonsville directory by typing

mkdir /scr1/users/kmarfurt/AASPI_Data

where obviously, you will need to type the correct location for your data. In my home directory, I will want to create a file called .datapath (note the dot "." in front of datapath!).
If you are either old or travel all over the world and cannot read a hangul or other user friendly editors (I qualify for both cases), you will use **vi** and type:

```
vi ~/.datapath
```

In this file there is only one line, which in my case will read:

```
datapath=/scr1/users/kmarfurt/AASPI_Data/
```

Note the extra "/" at the end. Save it and continue to the next step. In previous installations, you may have created directories called *SEP_Data*. Don't worry about it. Any name will do. We've changed the directory name to be "*AASPI_Data*" since SEP will not be supported on Windows. Adding some confusion, you may see a reference internal to the GUI software and python scripts to *PATH_SEP.* This is a variable that defines the directory PATH "SEParator" which is a forward slash "/" in Linux and a back slash "\" in Windows and has nothing to do with the Stanford Exploration Project.

Setting your ENVIRONMENT variables

One of the more commonly used routines is called aaspi_plot which displays 32 floating point data using a user-defined color bar. To see where the path of this routine is, you would type:

```
which aaspi_plot
```

You will almost certainly get a response that says something like:

no aaspi_plot in /usr/bin /usr/etc/bin followed by a long list of other directories where executable code may be found...

We therefore need to set our PATH. To determine what your current paths are you type:

```
echo ${PATH}
```

The braces { } are often optional, but it is good practice to always use them. The $ symbol indicates that we wish to echo the *value* of the variable PATH rather than the alphanumeric letters "PATH" themselves. Most installations will have a company specific .bashrc (for Linux) in your home directory. Your IT person will be very upset if

you modify these generic files. I have also found some installations that use a .cshrc file on Linux to make the IT folks life a little simpler (csh is the login shell for some UNIX systems). If you edit these files (note the dot "." in front of it!), it will probably have a statement that says something like "source my_cshrc" or "source .user_bashrc" or simply ". .user_bashrc" where the in this case the first dot is shorthand for "source".

Such a user_bashrc or equivalent construct allows a user to add his or her desired quirks to the login process. For instance you may wish to alias a commonly used Linux command such as *rm* to be another, with "*rm –i*" (such that you are prompted when you attempt to erase files) with the line

alias rm='rm –i' .

You may also wish to set the delete key to be a backspace. To make life easier, we have created shell scripts that set the PATH and other variables necessary for the AASPI software to work properly. These are named *set_aaspi_env.sh* and *set_aaspi_env.csh* for the Bash or C shell
respectively and are located in the scripts subdirectory of the AASPI directory. You will need to source one of these file in either your *.cshrc* (or *.my_cshrc*) or *.bashrc* (or *user_bashrc*) depending upon your shell. If you don't happen to know which shell you are using, you should be able to type

```
echo ${SHELL}
```

to see which shell you are using. You will need to add a line like

```
source /wherever/AASPI/scripts/set_aaspi_env.csh
```

or

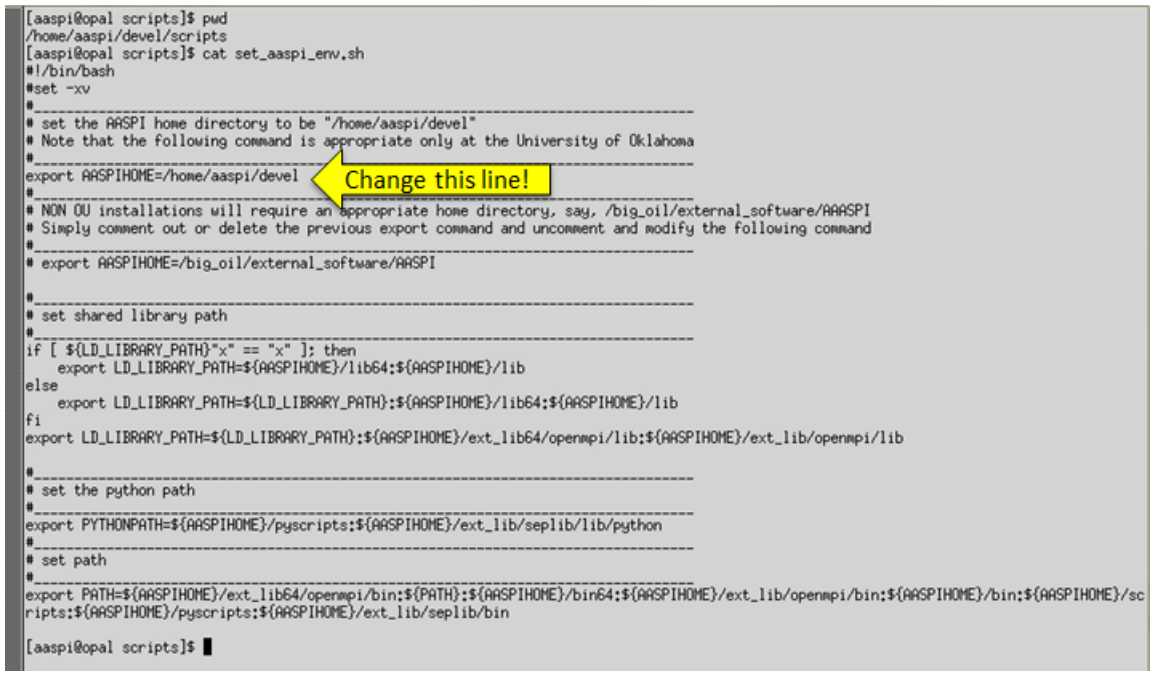
```
. /wherever/AASPI/scripts/set_aaspi_env.sh (a dot followed by a space)
```

to your *.cshrc* or *.bashrc* file. This should set up most everything for you to run. BE CAREFUL, IF YOU ARE RUNNING OTHER APPLICATIONS UNDER MPI YOU MAY NEED TO INVOKE YOUR NORMAL INSTALLED VERSION! This is why your IT support folks don't want you messing with the generic *.bashrc* or *.cshrc* files!

There is one variable that needs to be edited at the top of the *set_aaspi_env.sh* (or *set_aaspi_env.csh*) for AASPI to work properly. Look for the line that looks like:

```
export AASPIHOME=/home/aaspi/dist
```

and change it to point to your aaspi installation directory (something like */external_software/AASPI*). You will note that the variable *${AASPIHOME}* is used later in the script. Be sure you update the new file.




```
[aaspi@opal scripts]$ pwd
/home/aaspi/devel/scripts
[aaspi@opal scripts]$ cat set_aaspi_env.sh
#!/bin/bash
#set -xv
#-------------------------------------------------------------------------------
# set the AASPI home directory to be "/home/aaspi/devel"
# Note that the following command is appropriate only at the University of Oklahoma
#-------------------------------------------------------------------------------
export AASPIHOME=/home/aaspi/devel
#-------------------------------------------------------------------------------
# NON OU installations will require an appropriate home directory, say, /big_oil/external_software/AAASPI
# Simply comment out or delete the previous export command and uncomment and modify the following command
#-------------------------------------------------------------------------------
# export AASPIHOME=/big_oil/external_software/AASPI

#-------------------------------------------------------------------------------
# set shared library path
#-------------------------------------------------------------------------------
if [ ${LD_LIBRARY_PATH}"x" == "x" ]; then
    export LD_LIBRARY_PATH=${AASPIHOME}/lib64:${AASPIHOME}/lib
else
    export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${AASPIHOME}/lib64:${AASPIHOME}/lib
fi
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${AASPIHOME}/ext_lib64/openmpi/lib:${AASPIHOME}/ext_lib/openmpi/lib

#-------------------------------------------------------------------------------
# set the python path
#-------------------------------------------------------------------------------
export PYTHONPATH=${AASPIHOME}/pyscripts:${AASPIHOME}/ext_lib/seplib/lib/python
#-------------------------------------------------------------------------------
# set path
#-------------------------------------------------------------------------------
export PATH=${AASPIHOME}/ext_lib64/openmpi/bin:${PATH}:${AASPIHOME}/bin64:${AASPIHOME}/ext_lib/openmpi/bin:${AASPIHOME}/bin:${AASPIHOME}/sc
ripts:${AASPIHOME}/pyscripts:${AASPIHOME}/ext_lib/seplib/bin

[aaspi@opal scripts]$
```

Once the set_aaspi_env.sh has been modified for your system and set_aaspi_env.sh has been invoked in each user's .bashrc they will need to source it in order to initialize the new paths. Usually, you can do this by typing:

source ~/.bashrc

which of course will in turn source the recently edited file *.my_bashrc*

Now, if you type

which aaspi_plot

you should obtain a response like

/external_software_directory/AASPI/bin64/aaspi_plot

If this does not work, try invoking a new xterm to execute a clean run of the .bashrc file.

Testing the graphics display utilities

Now let's see if we can plot any data. Go to your boonsville directory and try to plot it using aaspi_plot. In my case, I might type:

cd /scr1/users/kmarfurt/boonsville (your directory will be something different!)

```
aaspi_plot d_fn=d_boonsville_orig.H gainpanel=every &
```

```
aaspi_plot d_fn=d_boonsville_orig.H gainpanel=every &
```

After a short time the data should start plotting up. If they don't, you may obtain two different errors. If you obtain a response that says "no such file or directory" it indicates one of the following four errors: (1) you did not spell *d_boonsville_orig.H* correctly, (2) you added a space between the d_fn= and d_boonsvillle_orig.H , (3) you are not in the directory with this file in it, or (4) you do not have read permission for this file.

A second common error is associated with **X11** (or equivalent display) when running on a remote compute server. In this case you need to echo your *DISPLAY* variable by typing

```
echo ${DISPLAY}
```

The response should come back to your local terminal. If not, you will need to set the DISPLAY variable by typing on the *remote* server window:

```
DISPLAY=local_computer_name:0.0
```
(in Linux)

and then on your *local* Linux window by typing

```
xhost +
```

If you are accessing your Linux server from a PC using **PuTTY**, you will need to invoke **Xming** or its equivalent on the PC side. (Details can be found in Section 2 of the documentation manual).

These commands are common to almost all computer applications (including interpretation workstation software) requires some variant of these processes. If you have a stand-alone computer, you may need to log in as root and type

```
DISPLAY=:0.0
```

```
xhost +
```

To determine if this works, you should try typing:

```
xterm –bg orange &
```

If an orange **xterm** window pops up you are in excellent shape.

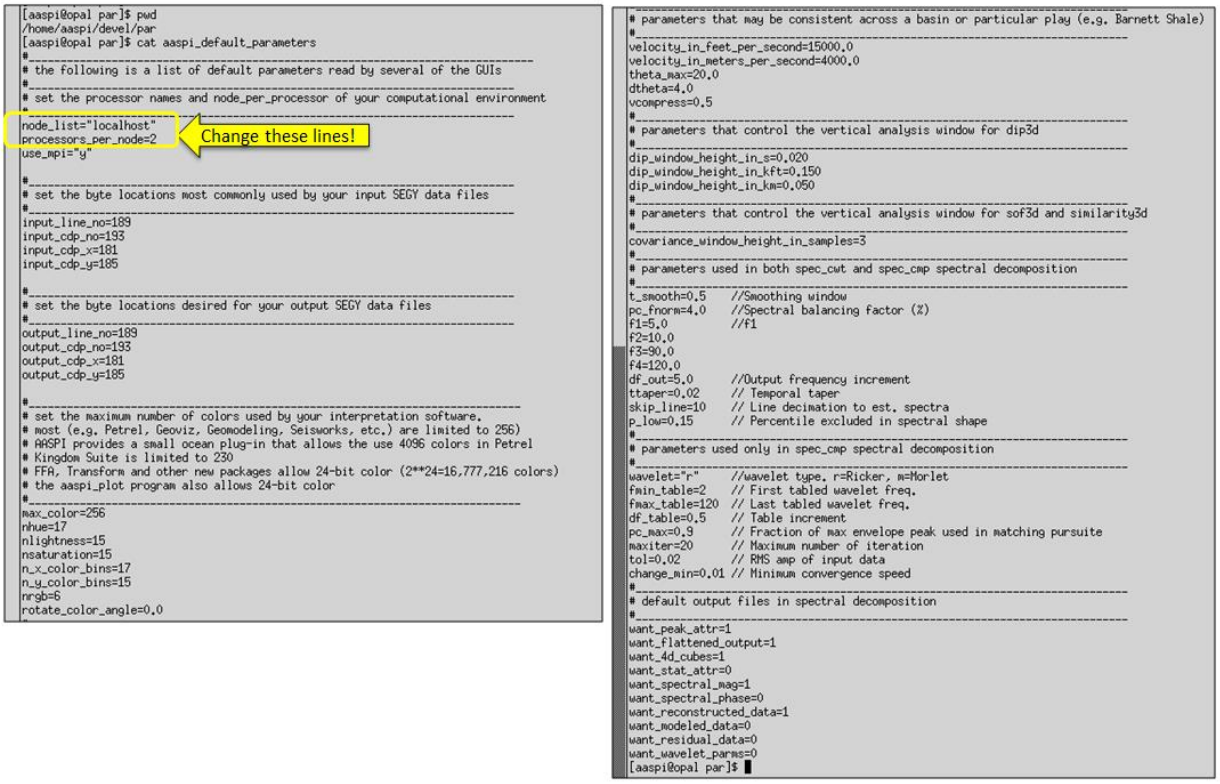
Setting AASPI Default Parameters

One of the major changes made in the September 21, 2012 release was the inclusion of an aaspi_default_parameters file containing, as the name implies, commonly used defaults. Some of the defaults will be installation specific, such as default node list and numbers of processors, and will be set by you, the person who installs the AASPI software. This version of the *aaspi_default_parameters* file resides in the *${AASPIHOME}/par* directory. Other parameters may be user specific, and will (optionally) be set up by each user who will create the *aaspi_default_parameters* in their home directory. Examples may include default velocity values which might be 10000 ft/s for someone working Tertiary targets and 18,000 ft/s for someone working Paleozoic targets. Finally, a user may (optionally) wish to set up a list of default parameters in their project directory. The search hierarchy will always be to search first for parameters in the user local directory, then the user home directory, then the *${AASPIHOME}/par* directory.

If I type in the following commands

cd *${AASPIHOME}/par*
cat aaspi_default_parameters

I will see the following file, which, if you wish, can be edited with vi or any other editor:



We have set many of these parameters to be quite conservative. For instance, in a classroom environment with dozens of students, we restrict our students to run on the

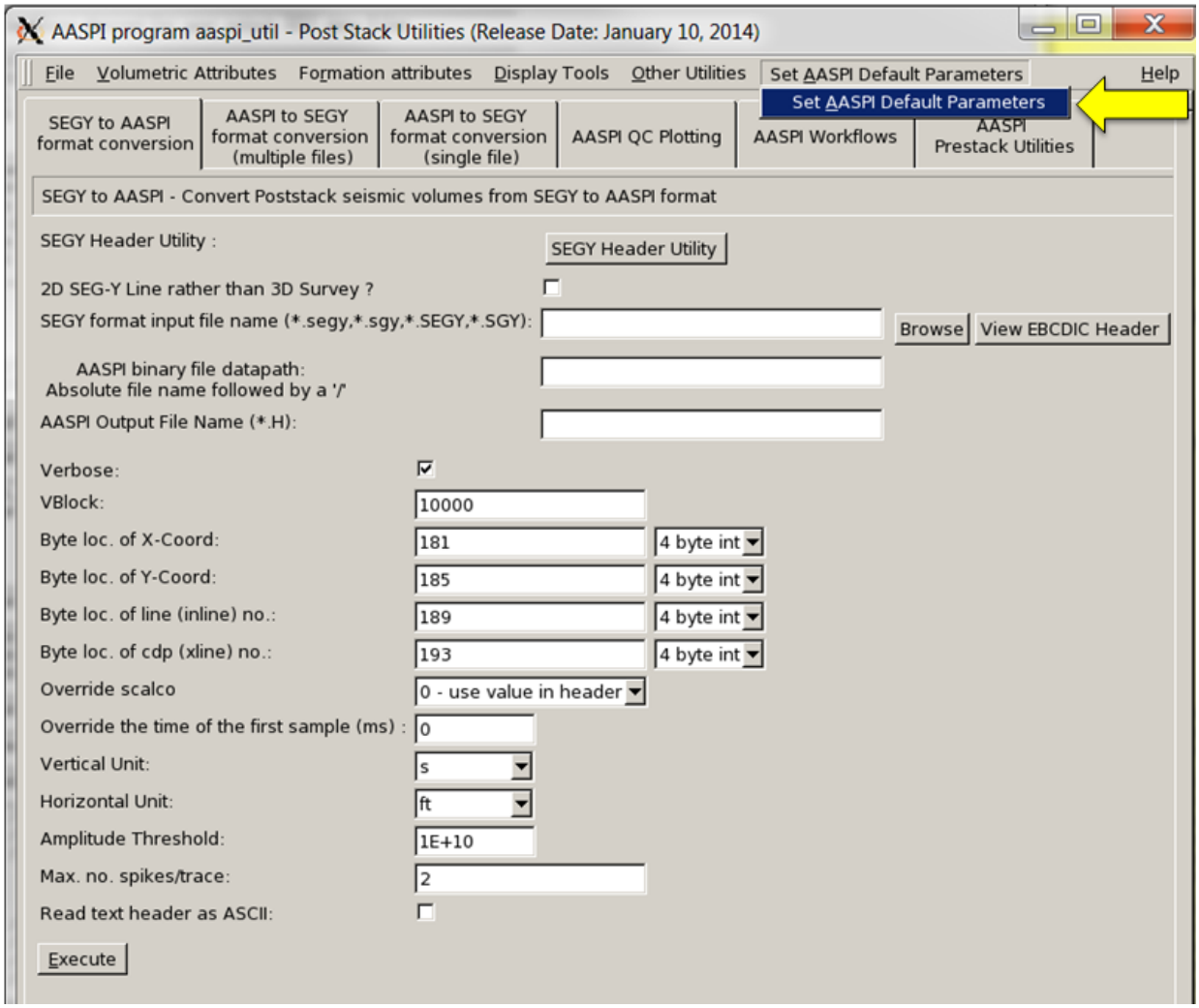
local node they have logged into (node_list=localhost) with just two processors (processors_per_node=2). Those students doing research will run on multiple nodes with more processors each (e.g. *node_list="d009 d010 d011 d012 d013 d014*" and *processors_per_node=8*) . Many companies have compute nodes with 64 processors. We recommend that you as the software installation person, choose some reasonable defaults for your environment.

Since the users will be able to modify their defaults, the following part of the documentation will be replicated in section 3 of the AASPI online manual. You can either create a new aaspi_default_parameters file in any directory of your choice and later replace the version that came with the aaspi installation, or if you log on with write permission to the ${AASPIHOME}/par directory and type

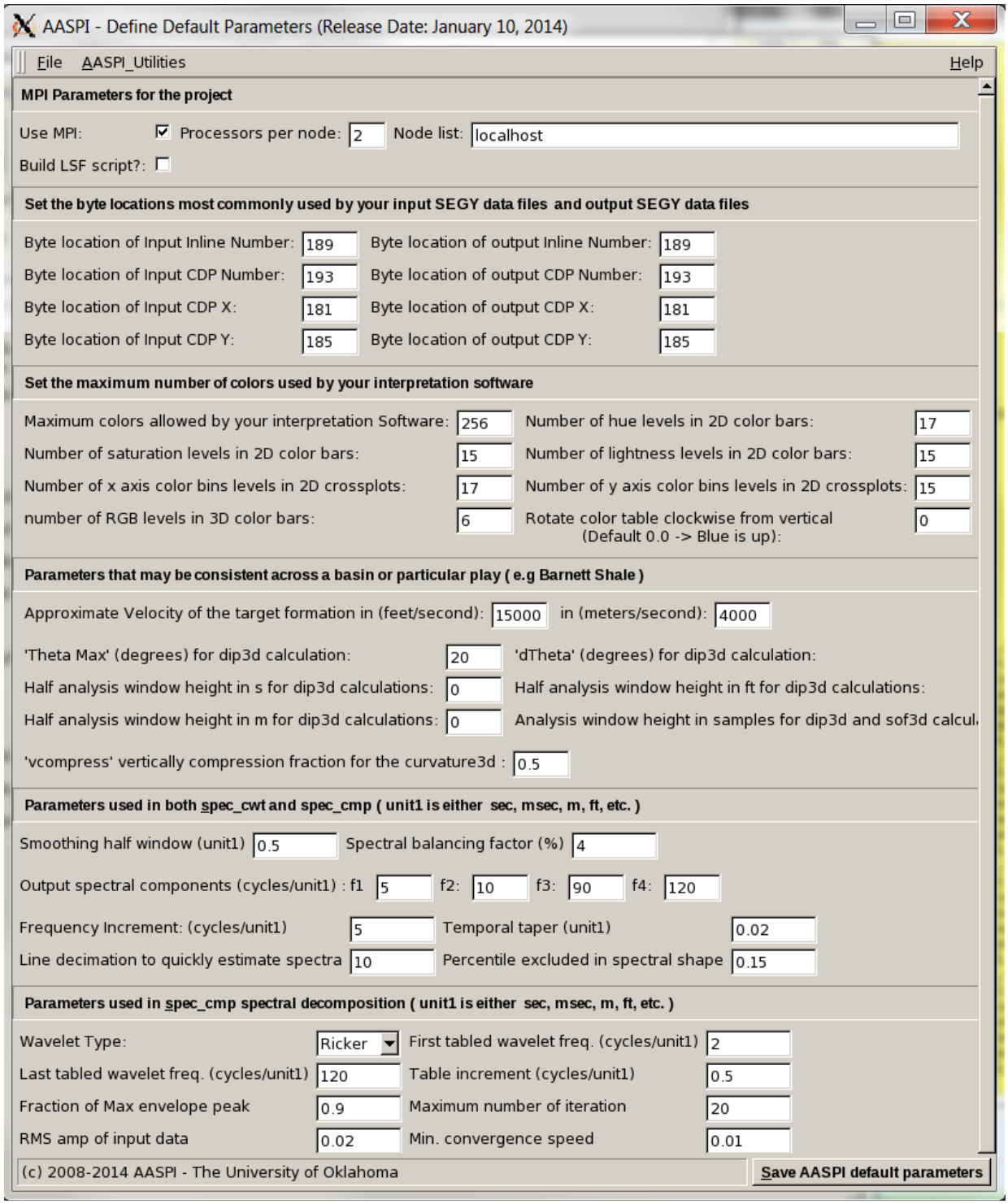
```
cd ${AASPIHOME}/par
```

```
aaspi_util &
```

The following GUI will appear:



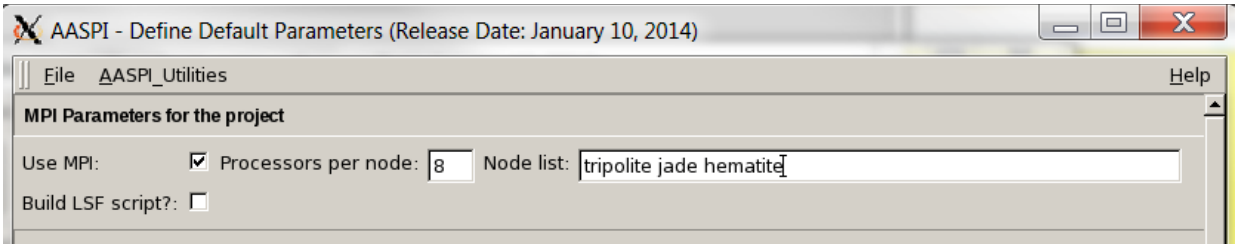
Click the tab “Set AASPI Default Parameters” to obtain another GUI:



Note that parameters in this GUI are identical to those in the *aaspi_default_parameters* file. By construction, the default parameters are read from an *aaspi_default_parameters* file in the following hierarchal order (1) in your local directory, (2) in your home directory, and (3) in the *${AASPIHOME}/par* directory. Any

changes you make will write out a file called *aaspi_default_parameters*. If you do this in your home directory, this will be the new user defaults. If you do so (as the installation person) in the *${AASPIHOME}/par* directory, these parameters will become the defaults for the entire system.

Several of these defaults are application and perhaps even data specific. Specifically, those used by spectral decomposition programs **spec_cmp** and **spec_cwt** will depend on data quality representative of deepwater or Paleozoic basins. However, the following should be first set by the AASPI installation person and then modified (if appropriate) by each user. The top of my GUI now looks like this:



I will always with to use MPI, so this box remains checked. I will run on across several nodes called tripolite, jade, and hematite. While tripolite has 12 processors, jade and hematite only have 8, so I set the Processors per node to be 8, giving me a total of 24 processors. The GUI will add the quotation marks to the output file. If you log into a large compute server with 64 nodes, but the data are on a remote disk farm, set Processors per node =63, allowing the 64th processor (named the 0th processor in the software) to serve as the master that does all the i/o asynchronously across the net. Similarly, if someone is running ProMAX on one of the processors (e.g. simply loading data for a subsequent run), reduce the number of processors requested to avoid conflict. The operating system is clever enough to distribute he workload to the unused processors.

While companies will obtain seismic data from a multitude of sources, many of companies have strategic alliances with different vendors such that data always come in with the same headers. Here, I have the byte locations for both the input SEGY data volumes and the output (AASPI attribute) SEGY volumes set to the SEG 2002 standard. However, if you intend to load all of your data into Geoframe, it will be easier if you set the output byte locations to be 9, 21, 73, and 77. Likewise, if most of your data are coming from CGG, you will want to use their common byte formats for input. The user can always change these values for any application. These are simply defaults.

I have also defaulted the number of colors that can be used. More modern systems, such as FFA, Transform, and Teraspark use 24-bit color (32-bit if you count transparency). However, many interpretation packages are built upon previously existing legacy software with most of them having a limitation of 256 colors (and Kingdom Suite only 240). In principal, 256 colors allow a 16x16 2D color bar. I've set the defaults to be 17x15, which uses 255 colors. AASPI display applications like **hlplot**

and **crossplot** will then set the remaining 256th color to white to display dead and padded traces.

At OU, we use Petrel for most of our seismic interpretation. While Petrel will not allow you to import more than 256 colors under their *Template* tab, internally, it can support 4096 colors. To achieve this we have written a simple Ocean utility called **aaspi_modulation_module_petrel_2011.1.64** that imports and assigns a 4096 color bar to a data volume that has been converted to 16-bit brick format. We will put this utility on our software directory for those of you who use Petrel. Note the 2D color bars used by programs hlplot, hsplot, and crossplot will be 65*63, defining 4095 colors. As before, the remaining (4096th) color will be set to white.

AASPI display program **rgbplot** works reasonably well with 16^3=4096 colors with the number of red, green, and blue values being set to *nrgb=16*. The results of **rgbplot** are almost worthless when using only 256 colors. Details on how to use these programs will be described in Section 8.

The next group of default parameters controls the behavior of programs dip3d and curvature3d. For smaller companies focused on resource plays or carbonates, a reasonable installation-wide default might be chosen. However, it is more likely that one business unit will work on Tertiary basin plays and other on older, more consolidated resource plays. In the absence of depth migrated or depth converted data, the dip estimates and curvature computations need to have a reference velocity. We suggest using a velocity appropriate to your target of interest. Thus, for interpreters working the Barnett Shale or Mississippi Lime plays, a relatively high velocity of 17000 ft/s may be appropriate.

One observation that many of the AASPI software users have noticed is how the curvature anomalies are vertically smeared more than they would like to see. Part of this smearing is due to an inappropriate velocity, and part of the smearing is due to our choices at OU in implementing volumetric curvature operators. If our data are highly complex, with dips ranging to 60 degrees, it makes sense to make the curvature operators as isotropic as possible. Specifically, if a “long wavelength” operator reaches out to 2000 ft (or about 18 traces for 110 ft spacing), then it should also reach up and down 2000 ft. If our conversion velocity is 10000 ft/s this value we will reach up and down 0.4 s in two-way travel time, or 100 samples for a 2 ms sample increment. In faster, Paleozoic basins, this operator would be reduced to 0.22 s or 56 samples vertically. Examining the derivative operators discussed in Section 10, we realize that these extreme values have only a small effect. Nevertheless, the amount of vertical mixing is directly controlled by the conversion velocity. The second parameter, *vcompress*, provides additional control. Most of the shale gas plays have dips that rarely exceed 5 degrees. Given this insight, there is no great motivation to use an isotropic derivative operator such that we can “compress” the vertical operator significantly, say to 50% or even 10% of its original value. Such increase in vertical resolution may help differentiate subtle changes in folding between adjacent stacked

lithologies. Currently, the default value of *vcompress=0.5* . We will discuss the impact of such parameters in Section 10.

Once you have chosen reasonable defaults for your environment, simply click the Save AASPI Default Parameters button in the lower right hand corner. If you are doing this for the system installation, be sure to copy your modified file to replace *${AASPIHOME/par}* .

Defining names used in the AASPI to SEGY conversion utilities

The AASPI system has the ability to rename the attribute files in manner more consistent with your environment. Many of our sponsors use Oracle and other data bases with a predefined naming convention. Some of the older interpretation software (such as Geoframe) may be relatively limited in the number of characters a file name can have. We therefore constructed a GUI that facilitates this naming strategy. We will repeat this part of the documentation in Section 3 of the AASPI User Manual, but wish to discuss it here so that you as the software installation person can predefine defaults that are well-aligned with your company's naming convention.

As with the default parameters defined above, and indeed with the interface between the GUIs and the shell scripts, everything is controlled by intermediate files. The use of files (rather than command line arguments) facilitates moving our software across the Linux/Windows7 divide. In this case, the files actually reside in the ${AASPIHOME}/lists directory and have the form *.list

```
[aaspi@opal lists]$ pwd
/home/aaspi/devel/lists
[aaspi@opal lists]$ ls -ltr
total 136
-rwxr-xr-x 1 aaspi aaspi 237 Apr 30  2012 aaspi_glcm3d_list
-rwxrw-r-- 1 mtha  aaspi 476 Apr 30  2012 aaspi_curvature3d_k_list
-rwxr-xr-x 1 aaspi aaspi  27 Apr 30  2012 aaspi_apparent_dip_list
-rwxr-xr-x 1 aaspi aaspi  37 Apr 30  2012 aaspi_apparent_gradient_list
-rwxr-xr-x 1 aaspi aaspi  17 Jun 27  2012 aaspi_euler_curvature_e_list
-rwxr-xr-x 1 aaspi aaspi  17 Jun 27  2012 aaspi_euler_curvature_k_list
-rwxrw-r-- 1 mtha  aaspi 214 Dec  2  2012 aaspi_sof3d_list
-rwxr-xr-x 1 aaspi aaspi  84 Apr 11  2013 aaspi_spectral_probe_list
-rwxr-xr-x 1 aaspi aaspi  52 Apr 11  2013 aaspi_stat3d_list
-rwxrw-r-- 1 mtha  aaspi 110 Jul 17 02:19 aaspi_dip3d_list
-rwxr-xr-x 1 aaspi aaspi 840 Jul 17 02:19 aaspi_image_filt3d_list
-rwxr-xr-x 1 aaspi aaspi 660 Jul 17 03:47 aaspi_footprint_suppression_list
-rwxrw-r-- 1 mtha  aaspi 455 Jul 17 04:26 aaspi_curvature3d_e_list
-rwxrw-r-- 1 mtha  aaspi 263 Sep  5 14:52 aaspi_similarity3d_list
-rw-rw-r-- 1 aaspi aaspi 268 Nov 13 04:29 aaspi_spec_cwt_list
-rwxrw-r-- 1 mtha  aaspi 443 Nov 13 04:30 aaspi_spec_cmp_list
-rwxrw-r-- 1 aaspi aaspi 495 Nov 13 04:49 aaspi_spec_clssa_list
[aaspi@opal lists]$
```

If I *cat* one of *the aaspi_dip3d_list* I note that it consists of two identical columns:

```
[aaspi@opal lists]$ cat aaspi_dip3d_list
inline_dip                  inline_dip
crossline_dip               crossline_dip
dip_magnitude               dip_magnitude
dip_azimuth                 dip_azimuth
[aaspi@opal lists]$
```

The column on the left will not be changed by the GUI and will form the root word if the AASPI format files, which typically have the form

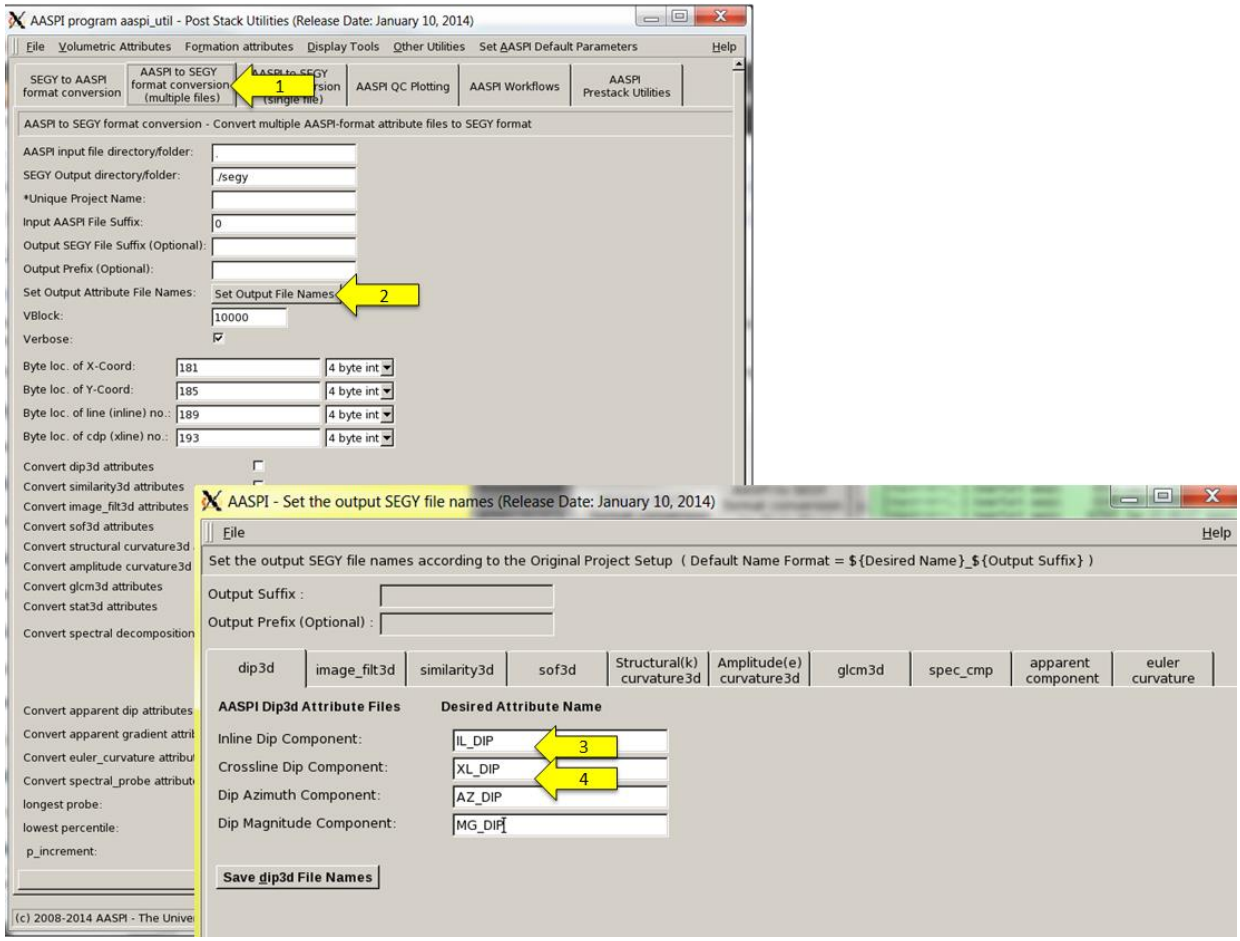
${root_left}_${unique_project_name}_${suffix}.H .

The column on the right *can* be changed. By default, the corresponding output file will have the form of previous AASPI releases:

${root_right}_${unique_project_name}_${suffix}.segy .

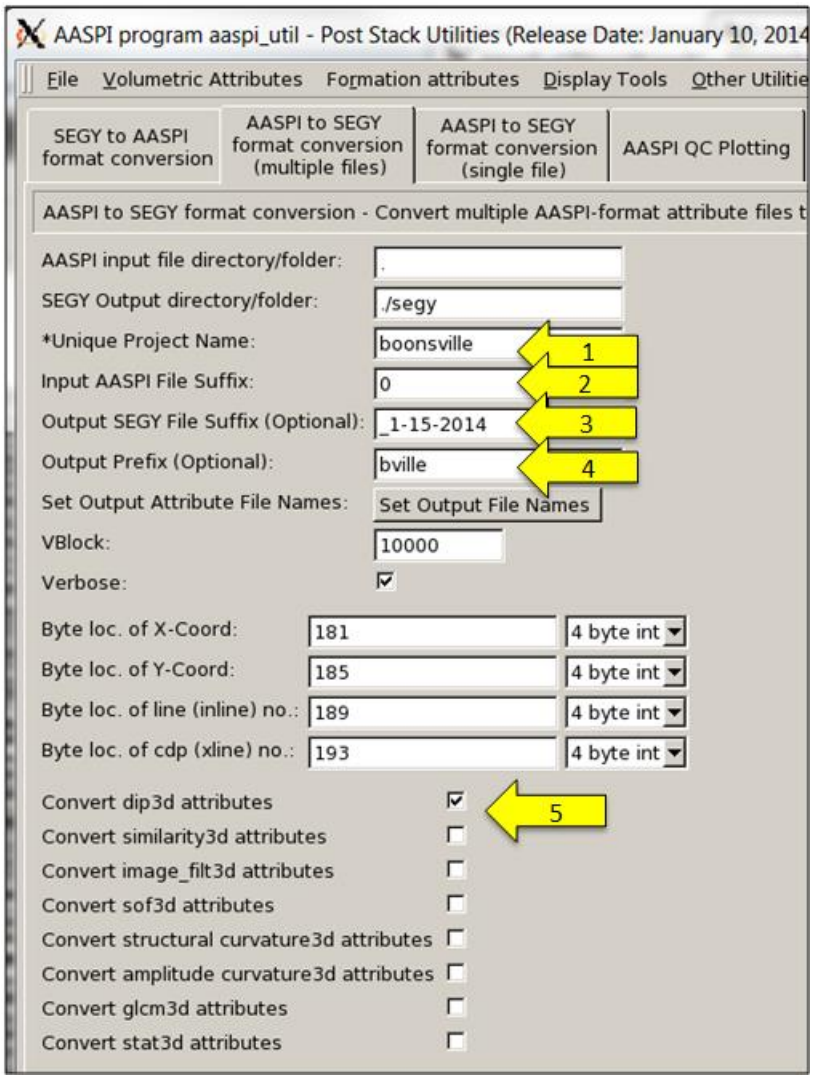
However, in the GUI, one can not only modify the right hand column, but also add a user-defined output prefix and output suffix (either of which may be blank). For instance, several Geoframe users require the jobname to be the leading characters in the file name. If there is a 16-character name limit, then the AASPI name needs to be shortened.

To set this up with GUI, (1) choose the *AASPI to SEGY Conversion (Multiple Files) tab*, and then (2) click *Set Output File Names* as shown below:



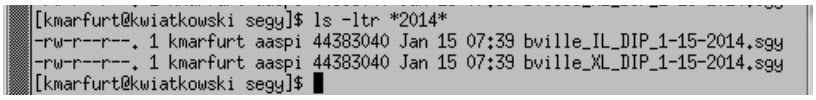
The GUI on the lower right appears. Click the tab of the programs for which you wish to change the names. In this case, I've chose the dip3d tab. Rather than have my output files begin with (4) *inline_dip* and (5) *crossline_dip* (the defaults), I've typed in shorter file names consistent with Geoframe called *IL_DIP* and *XL_DIP*. Ideally, you will only want to do this once and the person doing it will place it in the ${AASPIHOME}/lists directory so that everyone uses the same convention. For this exercise, I did it in my home directory and generated the new *aaspi_dip3d_list* file:

```
[kmarfurt@kwiatkowski boonsville]$ cat aaspi_dip3d_list
inline_dip                    IL_DIP
crossline_dip                 XL_DIP
dip_magnitude                 MG_DIP
dip_azimuth                   AZ_DIP
[kmarfurt@kwiatkowski boonsville]$
```



I had run this job before with (1) the *Unique Project Name* of *boonsville* and (2) a *Suffix* of *0*. I set (3) Output Suffix to be "*_1-15-2014"* (today's date) and (4) the Output *Prefix*

to be "*bville"*. I (5) put a checkmark in next to *Convert dip3d attributes* and then click *Execute*. The conversion completes and I obtain the following files in my segy subdirectory:



Testing ssh for use under mpi

See if you can ssh (secure shell) into one of the compute nodes (say big_oil_co_21) by typing

ssh big_oil_co_21 ls

If you are successful, you should see your home directory listing on big_oil_co_21.

If you are requested to provide a password, type it in. Then repeat the command. If you are unsuccessful, or alternatively, if the system did not remember that you had previously provided a valid password, you will need to modify your **ssh** configuration.

Steps for enabling Passwordless ssh
==========

ssh uses public key encryption to handle the initial user authentication. If you have not done so already, you will need to create a public key/private key pair. Assuming that you are using the openssh implantation of secure shell, you can use the handy utility ssh-keygen to create a public/private key pair:

ssh-keygen –t dsa

It will now prompt you for the location to save the generated key. Press *Enter* to accept the default. Then it will prompt you again for a passphrase. Press *Enter* again to set an empty passphrase. If you were to enter something here, then you would need to enter this passphrase each time you used ssh. It will now ask you to retype the passphrase – hit *Enter*. This will create two files named something like /myhome/.ssh/id_dsa and /myhome/.ssh/id_dsa.pub. These hold the private and public key respectively. We are not done yet. You still need to specify who can connect. Type

cd ~/.ssh

to change the directory to the hidden secure shell configuration directory in your home directory. You should see the *id_dsa* and *id_dsa.pub* files in this directory. Now you need to add your own public key to the *authorized_keys* file. If you don't have this file then you need to create it by copying your public key file to the *authorized_keys* file with

cp id_dsa.pub authorized_keys

chmod 400 authorized_keys

If you do have an *authorized_keys* file, you will need to do the following:

chmod 600 authorized_keys

cat id_dsa_pub >> authorized_keys

chmod 400 authorized_keys

Secure shell is very security conscious so the permissions on the authorized_keys file need to be set such that only the owner can read the file and no one can write to the file. **ssh** will ignore these files if the permissions are set incorrectly. If everything is correct, you should now be able to type:

ssh localhost ls

and see a listing of your home directory. Most cluster implementations mount the user's home directory across the different machines. If this is the case you are done. On the other hand, if you want to use a set of various machines you will need to add the contents of the *id_dsa.pub* file to each of the *authorized_keys* files on the various machines.

Installing some extra python libraries

In 2013 we began the process of replacing our Bourne shell scripts (*.sh) files with python scripts (*.py files), primarily because python runs in both Linux and Windows environments. Many professional computer programmers are able to prototype algorithmic applications as well as process control using python. Python comes "bare bones" with most Linux applications. At present, program "aaspi_graph" is the only application we have written that needs the more advanced python libraries, called "numpy" (www.numpy.org) . Future developments may use "scipy" (www.SciPy.org).

If you are using RedHat, ask your systems person to type

1) *yum install python-devel*
2) *python setup.py build*
3) *python setup.py install*

In principle, you can plot any *.H file. In practice, you will wish to plot simples operators and spectra from the curvature application programs that have the form d_dr_operator*.H and d_dr_spectrum*.H . If you have one of these try by typing

aaspi_graph.py d_dr_operator_boonsville_long_w_0.H &

if you have that file.

We also have a python SEGY header utility that called **PySEGY** that needs installation. Here are the steps:

1) sudo yum install python-devel (If you didn't do it above)

2) *Put PySEGY-1.0* (currently inside ${AASPIHOME}/pylib/) somewhere outside the main AASPI directory... for example /apps/aaspi/PySEGY-1.0/

Placing PySEGY-1.0 outside of the main AASPI directory (i.e. one level above) allows you not to have to "reinstall" PySEGY (build, install steps) everytime there is a new release of AASPI

3) *cd PySEGY-1.0* (wherever it now lives)

4) *python setup.py build*

5) *python setup.py install --home=/desired/path/* (such as /apps/aaspi/PySEGY-1.0/)

This will create a PySEGY.so file in /apps/aaspi/PySEGY-1.0/lib64/python/

6) edit PYTHONPATH in set_aaspi_env.sh/csh to have:

/apps/aaspi/PySEGY-1.0/lib64/python:/apps/aaspi/PySEGY-1.0 at the end ** This is probably the most crucial step as it is what allows python to find and use PySEGY!! at the end ** This is probably the most crucial step as it is what allows python to find and use PySEGY!!