New MATLAB® functions for reading, writing, and modifying SEG-Y files

Heather J.E. Lloyd, Kevin W. Hall and Gary F. Margrave

ABSTRACT

The SEGY_Toolbox for MATLAB® is a new set of tools that allow SEG_Y files to be read, modified, and written. This toolbox adheres to the SEG-Y Revision 1 standard set out in the SEG Y rev 1 Data Exchange format publication. Unlike previous SEG-Y MATLAB® tools, this toolbox tries to be as flexible as possible when reading SEG-Y files but very stringent when writing SEG-Y files. Currently the following tools are available: SEGY_EditTextHeader, SEGY_StandardizeHeader, SEGY_endianSwap, SEGY_getData, SEGY_getHeader, SEGY_getTraces, SEGY_read, SEGY_readHeader, SEGY_readMulti, SEGY_setHeader, SEGY_write, SEGY_writeHeaders, and SEGY_writeTraces. Please contact the authors if any further functionality is desired.

INTRODUCTION

In the past few years CREWES has developed several tools for reading and writing SEG-Y files. Most of these tools have been based strictly on the SEG-Y Revision 1 standard, making it difficult to read in files that do not conform to this standard. There have also been some tools that have been developed that will read in non-conforming files, but were fragmented and difficult to use. It has also, in the past, been difficult to use trace header values effectively. A new set of tools has been created that unites these fragmented tools into a cohesive and user friendly tool box. The SEGY_Toolbox allows most SEG-Y files to be read even if they do not conform to the SEG-Y standards. The new tools are distributed as part of the CREWES Matlab toolbox, and are in the segy/Segy_Toolbox directory. In order to use these tools, you will need to download the toolbox, install it, and update your Matlab path.

THE OBJECTS

This toolbox has been written using an object oriented approach. This means that each of the parts of a SEG-Y file have been made into an object. These objects are difficult to interrogate by themselves, so a series of tools have been made to help the users get the information that they need. There are five main objects that are used in the tool box: the BinaryHeader object, TextHeader object, and the Trace object, which consists of two other objects: the TraceHeader object and the TraceData object.

Trace

Main Contents:

- *TraceHeader Object*
- *TraceData Object*

A Trace object contains two other object, a TraceHeader and a TraceData object. The Trace object is primarily used to keep these two objects together. To get the trace header

information out of a Trace object use SEGY_getHeader. To get the trace data out of a Trace object use SEGY_getData.

TraceHeader

Main Contents:

- *Header Information*
- *Definition Information*

A TraceHeader object stores the trace header information. This object can exist by itself or as part of a Trace object. It uses a Definition file to format the header information into useful data. The information contained in each trace header is specific to the trace it belongs with. To get a piece of header information use the SEGY_getHeader tool. To change a header value, use the SEGY_setHeader tool. These tools will return a value for every trace that was read into Matlab.

TraceData

Main Contents:

- *Trace Data*

A TraceData object contains the traces of the SEG-Y file. It is found in a Traces object. To get the data into a usable Matlab matrix use the tool SEGY_getData.

BinaryHeader

Main Contents:

- *Header Information*
- *Definition Information*

A BinaryHeader object stores the binary header information. The information contained in this object is usually about the whole SEG-Y file. It uses a Definition file to format the header information into useful data. To get a piece of header information use the SEGY_getHeader tool. To change a header value, use the SEGY_setHeader tool.

TextHeader

Main Contents:

- *Text header Information*

A TextHeader object stores the text header information. This is ASCII or EBCDIC formatted text, which is used to communicate information to the User and not the computer. To get the text use the SEGY_getHeader tool. To save changes made to the header use the SEGY_setHeader tool.

THE TOOLS

SEGY_read

Syntax: *[traces,texthead,binaryhead,extendedhead]=SEGY_read(filein)*

- Inputs:
  filein = the name of the SEG-Y file that is to be read in (optional).
- Outputs:

traces = a TraceHeader object. To get the trace header values use SEGY_getHeader.
texthead = a TextHeader object. To get the character array use SEGY_getHeader.
binaryhead = a BinaryHeader object. To get the binary header values use SEGY_getHeader.
extendedhead = a cell array of TextHeader Objects. To get the character array use SEGY_getHeader.

SEGY_read is a tool that reads an entire SEG-Y file into Matlab objects. This tool automatically determines whether the text header is ASCII or EBCDIC format, uses comma delimited files to decode the binary header and the trace header, uses a graphical user interface that allows the user to verify the format of trace data (Figure 1) and uses a dialog box to identify any extended headers. For trace headers and binary headers that do not follow the SEG-Y revision 1 standards some additional steps are required, please see the non-standard SEG-Y files section in this paper.

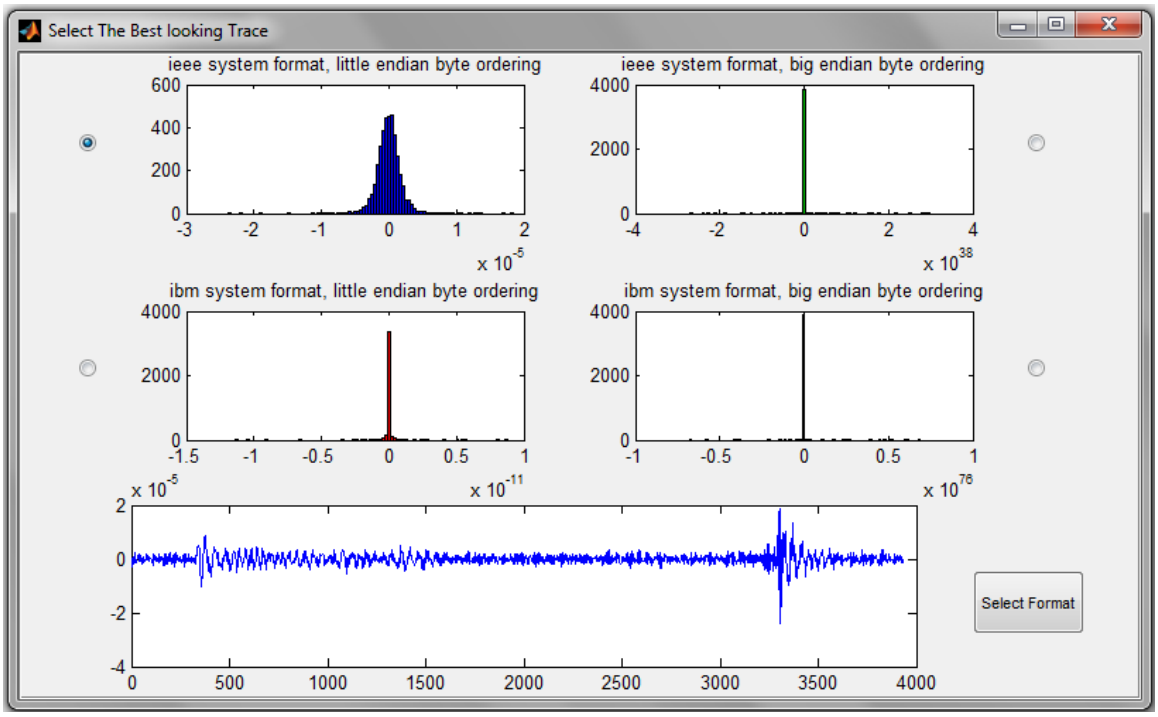

FIG. 1. This figure shows the trace format graphical user interface. The four top plots are histograms of the data when decoded using different file formats. The lower window is a plot of the trace using the selected format indicated by the radio button (in this case the upper left option).

SEGY_readHeader

Syntax: *[tracehead,texthead,binaryhead,extendedhead]=SEGY_read(filein)*

- Inputs:
  filein = the name of the SEG-Y file that is to be read in.

- Outputs:
  texthead = a TextHeader object. To get the character array use SEGY_getHeader.
  binaryhead = a BinaryHeader object. To get the binary header values use SEGY_getHeader.
  tracehead = a TraceHeader object. To get the trace header values use SEGY_getHeader.
  extendedhead = a cell array of TextHeader Objects. To get the character array use SEGY_getHeader.

SEGY_readHeader is a tool that will only read in the text header, binary header, extended headers and the trace headers. SEGY_getTraces will load the trace data of selected traces.

SEGY_readMulti

Syntax: *[traces, texthead,binaryhead,extendedhead]=SEGY_readMulti({})*
[traces, texthead,binaryhead,extendedhead]=SEGY_readMulti({file1,file2,file3})
[traces, texthead,binaryhead,extendedhead]=SEGY_readMulti({path})

- Inputs:
  file = a cell array containing the names of the SEG-Y files that are to be read in. file can also be the path to a directory containing *.sgy files. If file is set to {} then a prompt will ask the user to select files.
- Outputs:
  texthead = a TextHeader object. To get the character array use SEGY_getHeader.
  binaryhead = a BinaryHeader object. To get the binary header values use SEGY_getHeader.
  traces = a Trace object. To get the trace header values use SEGY_getHeader. To get the trace data values use SEGY_getData.
  extendedhead = a cell array of TextHeader Objects. To get the character array use SEGY_getHeader.

SEGY_readMulti is a tool that will read in several files, or a directory of files and combine them. For this to work, the job id, a number that indicates that the files belong to the same survey, must be the same for all files. If the job id is the same then this program assumes that the file contains the same amount of extended headers in the first file as the subsequent files. This program also assumes that if the trace format code, a value stored in the binary header, is the same for two files then they contain the same format of data. For example, both files would have IBM floating point data, rather than one having IBM type data and the other has IEEE type data.

SEGY_getTraces

Syntax: *traces = SEGY_getTraces(tracehead)*
traces = SEGY_getTraces(tracehead,word,value);
traces = SEGY_getTraces(tracehead,word1,value1,word2,value2,word2,value3,...);

- Input:
  tracehead = can be either a TraceHeader object which can be created by SEGY_readHeaders or a filename.
  word = a name defined in the definitions file, These values are given in Table 1.
  value = can be either a scalar value or a numerical array

- Output:
  traces = a Trace object. To get the traceheader values use SEGY_getHeader. To get the trace data values use SEGY_getData.

SEGY_getTraces is a function that allows the user to selectively read some traces from the file. This is done by comparing *value* to the traceheader data. The input parameter *word* must be one of the names defined in the definitions file. To follow the guidelines set out in the SEG-Y revision 1 standard, the word must any of the ones listed in Table 1. If a word is not among the ones listed in Table 1, an error will occur. It is possible to search using multiple words. An example would be:

traces = SEGY_getTraces(tracehead,'gx',1:50,'cdpx',60);

which would search for traces that have a group (receiver) *x* coordinate in the range 1:50 and a common depth point *x* coordinate of 60.

Table 1. This table contains the name of the variables used in the definitions file for the trace headers along with a description of what each variable is. These variables are based on the SEG-Y revision 1 Standard.

Name	Description
tracl	Trace sequence number within line
tracr	Trace sequence number within SEGY file
fldr	Original field record number
tracf	Trace number within the original field record
ep	Energy source point number
cdp	Ensemble number
cdpt	Trace number within the ensemble
trid	Trace identification code
nvs	Number of vertically summed traces yielding this trace
nhs	Number of horizontally stacked traces yielding this trace
duse	Data use
offset	Distance from center of the source point to the center of the receiver group
gelev	Receiver group elevation
selev	Surface elevation at source
sdepth	Source depth below surface
gdel	Datum elevation at receiver group
sdel	Datum elevation at source

swdep	Water depth at source
gwdep	Water depth at group
scalel	Scalar to be applied to all elevations and depths
scalco	Scalar to be applied to all coordinates
sx	Source coordinate X
sy	Source coordinate Y
gx	Group coordinate X
gy	Group coordinate Y
counit	Coordinate units
wevel	Weathering velocity
swevel	Subweathering velocity
sut	Uphole time at source in milliseconds
gut	Uphole time at group in milliseconds
sstat	Source static correction in milliseconds
gstat	Group static correction in milliseconds
tstat	Total static applied in milliseconds
laga	Lag time A
lagb	Lag Time B
delrt	Delay recording time
muts	Mute start time in milliseconds
mute	Mute end time in milliseconds
ns	Number of samples in this trace
dt	Sample interval in microseconds for this trace
gain	Gain type of field instruments
igc	Instrument gain constant (dB)
igi	Instrument early or initial gain (dB)
corr	Correlated
sfs	Sweep frequency at start (Hz)
sfe	Sweep frequency at end (Hz)
slen	Sweep length in milliseconds
styp	Sweep type
stas	Sweep trace taper length at start in milliseconds
stae	Sweep trace taper length at end in milliseconds
tatype	Taper type
afilf	Alias filter frequency (Hz)
afils	Alias filter slope (dB/octave)
nofilf	Notch filter frequency (Hz)
nofils	Notch filter slope (dB/octave)
lcf	Low-cut frequency (Hz)
hcf	High-cut frequency (Hz)
lcs	Low-cut slope (dB/octave)
hcs	High-cut slope (dB/octave)

year	Year data recorded
day	Day of year
hour	Hour of day
minute	Minute of hour
sec	Second of minute
timbas	Time basis code
trwf	Trace weighting factor
grnors	Geophone group number of roll switch position one
grnofr	Geophone group number of trace number one within original field record
grnlof	Geophone group number of last trace within original field record
gaps	Gap size
otrav	Over travel associated with taper at beginning or end of line
cdpx	X coordinate of ensemble (CDP) position of this trace (scalar in Trace Header bytes 71-72 (scalco) applies). The coordinate reference system should be identified through an extended header Location Data stanza (see section D-1).
cdpy	Y coordinate of ensemble (CDP) position of this trace (scalar in bytes Trace Header 71-72 (scalco) applies). The coordinate reference system should be identified through an extended header Location Data stanza (see section D-1).
iline	For 3-D poststack data, this field should be used for the in-line number. If one in-line per SEG Y file is being recorded, this value should be the same for all traces in the file and the same value will be recorded in bytes 3205-3208 (lino) of the Binary File Header.
xline	For 3-D poststack data, this field should be used for the cross-line number. This will typically be the same value as the ensemble (CDP) number in Trace Header bytes 21-24, but this does not have to be the case.
sp	Shotpoint number — This is probably only applicable to 2-D poststack data. Note that it is assumed that the shotpoint number refers to the source location nearest to the ensemble (CDP) location for a particular trace. If this is not the case, there should be a comment in the Textual File Header explaining what the shotpoint number actually refers to.
scalsp	Scalar to be applied to the shotpoint number in Trace Header bytes 197-200 to give the real value. If positive, scalar is used as a multiplier; if negative as a divisor; if zero the shotpoint number is not scaled (i.e. it is an integer. A typical value will be -10, allowing shotpoint numbers with one decimal digit to the right of the decimal point).
tval	Trace value measurement unit
tconstm	Transduction Constant mantissa
tconste	Transduction Constant exponent
tunit	Transduction Units
devtr	Device/Trace Identifier
scalt	Scalar to be applied to times
stypeo	Source Type/Orientation

sedm	Source Energy Direction with respect to the source orientation mantissa?
sede	Source Energy Direction with respect to the source orientation exponent?
smmtm	Source Measurement mantissa
smmte	Source Measurement exponent
smmtu	Source Measurement Unit
fbpicks	First Break Picks (This is not part of the SEG-Y Revision 1 Standard but has been added as a CREWES Standard)
scalfb	Scaling Factor for first breaks (This is not part of the SEG-Y Revision 1 Standard but has been added as a CREWES Standard)

SEGY_getHeader

Syntax: *hdr = SEGY_getHeader(obj,word,scalevals)*
Examples: *hdr = SEGY_getHeader(Texthead,'header')*
val = SEGY_getHeader(Binaryhead,word)
val = SEGY_getHeader(Tracehead,word)
val = SEGY_getHeader(Trace,word)

- If *obj* is a TextHeader object, a 40 x 80 char array will be returned;
- If *obj* is a BinaryHeader object, word must be a name from the definitions file. SEG-Y revision 1 standard binary header words can be found in Table2. The value returned would be the value that was associated with the variable name.
- If *obj* is a TraceHeader object or a Trace object, word it must be a name from the trace header definitions file. SEG-Y revision 1 trace header standard words can be found in Table 1. This function will return one value associated with this word per trace.

For Trace, TraceHeader, or BinaryHeader objects, the values will be automatically scaled by the scale factor in the header, if it is desired to not have these values scaled, then *scalevals* should be set to zero. An example of this is seen below:

val=SEGY_getHeader(tracehead,'fbpicks',0);

To allow the values to be scaled enter either of the following:

val=SEGY_getHeader(tracehead,'fbpicks',1);
or
val=SEGY_getHeader(tracehead,'fbpicks');

SEGY_getHeader is a tool that will return the header or header values. For a text header object this tool will return the 40 x 80 character text header. For a binary header, trace header, or trace, object this tool requires the header variable name that is desired, such as ‘cdp’ or ‘gx’ for trace headers. This tool will then return the values associated with that variable name, for a binary header object there will only be one value returned whereas for a trace or trace header object there will be an array of values where one value belongs to each trace.

TABLE 2. This table contains the name of the variables used in the Binary Header definitions file along with a description of what each variable is. These variables are based on the SEG-Y revision 1 Standard.

Name	Description
jobid	Job identification number
lino	Line number
reno	Reel number
ntrpr	Number of data traces per record
nart	Number of auxiliary traces per record
hdt	Sample interval in microseconds
dto	Sample interval of field data in microseconds
hns	Number of samples per data trace
nso	Number of samples recorded in field data per data trace
format	Data sample format code
fold	Ensemble fold
tsort	Trace sorting code
vscode	Vertical sum code
hsfs	Sweep frequency at start (Hz)
hsfe	Sweep frequency at end (Hz)
hslen	Sweep length (ms)
hstyp	Sweep type code
schn	Trace number of sweep channel
hstas	Sweep trace taper length (ms)
hstae	Sweep trace taper length (ms)
htatyp	Taper type
hcorr	Correlated data traces
bgrcv	Binary gain recovered
rcvm	Amplitude recovery method
mfeet	Measurement system
polyt	Impulse signal polarity
vpol	Vibratory polarity code
rev	SEGY Format Revision Number
flen	Fixed length trace flag
netfh	Number Extended Textual File Headers
unasn1-84	Unassigned values 1 through 84

SEGY_setHeader

Syntax: *obj=SEGY_setHeader(obj,word,value,scalevals)*
Examples: *Texthead=SEGY_setHeader(Texthead,'header',header)*
Binaryhead=SEGY_setHeader(Binaryhead,word,value)
Tracehead=SEGY_setHeader(Tracehead,word,value)

Trace=SEGY_setHeader(Trace,word,value)

- For a TextHeader object, word must be 'header' and value must be a 40 x 80 character array.
- For a BinaryHeader object, word must be a name from the definitions file. The binary header words based on the SEG-Y revision 1 standard can be found in Table 2. Value must be a single value that belonging to the word.
- For a TraceHeader object or a Trace object, word must be a name from the definitions file. SEG-Y revision 1 standard trace header words can be found in Table 1. Value must be a numerical array where the values are described by the word, and the length of value must be the same as the number of traces.

For Trace, TraceHeader, or BinaryHeader objects, the values will be automatically scaled by the scale factor in the header, if it is desired to not have these values scaled then *scalevals* should be zero. An example of this is seen below.

tracehead=SEGY_setHeader(tracehead,'fbpicks',values,0);

To allow the values to be scaled enter either of the following:

tracehead=SEGY_setHeader(tracehead,'fbpicks',values,1);
or
tracehead=SEGY_setHeader(tracehead,'fbpicks',values,);

SEGY_setHeader is a tool that will set the header or header values. For a text header object this tool will change the 40 x 80 character text header to the user specified header. This header must be of the same dimensions. For a binary header, trace header or trace object, this tool will set the header variable name specified by the user to the values that was specified. It is essential that for trace or trace header objects there is one value for each trace.

SEGY_getData

Syntax: *data = SEGY_getData(trace);*

- Input
  trace = a Trace object containing trace data
- Output
  data = the trace data in the object. This data is organized as follows [number of samples x number of traces]

SEGY_getData will return the trace data that is stored in a trace object. This is different from SEGY_getTraces as this data is already in Matlab format and is stored in an object. The data can then be manipulated, but if traces are added or subtracted then the trace header values must also be changed to accommodate the change in traces.

SEGY_EditTextHeader

Syntax: *texthdr = SEGY_EditTextHeader(texthead);*

texthdr = SEGY_EditTextHeader();

- Inputs:
  texthead can either be left blank or be a TextHeader object
- Outputs:
  texthdr is a TextHeader object that contains the updated header if the input was a TextHeader object, else texthdr is a 40 x 80 character array of the updated information.

SEGY_EditTextHeader is a tool that allows the user to modify the text header. If the text header is in an un-editable format, this tool allows the user to flip between ASCII and EBCDIC formatting.

SEGY_write

Syntax: *SEGY_write(fileout,traces,texthead,binaryhead,extendedhead)*
SEGY_write(fileout,traces,texthead,binaryhead)

- Inputs:
  traces = can be a Trace object or a cell array where:
  {1,1}= is a matrix of traceheader information. This is a (240 x number of traces) of type uint8.
  {1,2}= is a matrix of traces. This is (tracelength x number of traces) of 4-byte floats.
  fileout = the name of the new sgy file should end in .sgy
  texthead = can be a 40 x 80 char array or a TextHeader object.
  binaryhead = can be a numerical array that satisfies SEG-Y Revision 1 or a BinaryHeader object.
  extendedhead = is optional. This should be either a 40 x 80 char array or a TextHeader object. If multiple extended headers are required extendedhead can be a cell array containing multiple char arrays or multiple TextHeader objects.

SEGY_write is a tool for writing a new SEG-Y file. It strictly adheres to the SEG-Y Revision 1 standards. The text header is written in ASCII format, and trace data are written in big-endian IEEE floating-point format. The binary header and trace headers are written using the definitions in the Revision 1 standard, however, CREWES has set the last two unassigned variables to first break picks (bytes 233-236) and a scale factor for the first break picks (bytes 237-240). If using nonstandard definitions it is essential that the definitions file contains a SEGY_Rev1_EQ column in the spreadsheet, with a list of the Revision 1 standard equivalents. Please see the non-standard SEG-Y files section for more information.

SEGY_writeHeaders

Syntax:*SEGY_writeHeaders(fileout,texthead,binaryhead,extendedhead,permission)*
SEGY_writeHeaders(fileout,texthead,binaryhead)

- Inputs:
  fileout = the name of the new sgy file should end in .sgy

texthead = can be a 40 x 80 char array or a TextHeader object.
binaryhead = can be a numerical array that satisfies SEG-Y Revision 1 or a BinaryHeader object.
extendedhead = is optional. This should be either a 40 x 80 char array or a TextHeader object. If multiple extended headers are required then extendedhead can be a cell array containing multiple char arrays or multiple TextHeader objects.
permission = if you want to rewrite the headers (not recommended) then use permission='r+' If you want to overwrite an old file or create a new one use permission='w+'. 'w+' is the default.

SEGY_writeHeaders is a tool that will only write the text header, binary header, and any extended headers. This is to allow traces to be written in a loop in a subsequent step. To write the traces use SEGY_writeTraces. Like SEGY_write, this tool also adheres to the SEG-Y Revision 1 standards.

SEGY_writeTraces

Syntax: *SEGY_writeTraces(fileout,traces,numcurtraces,numcurshots)*
SEGY_writeTraces(fileout,{tracehead, tracedata},numcurtraces,numcurshots)

- Inputs:
  fileout = the name of the new sgy file should end in .sgy
  traces = can be a Trace object or a cell array where:
    {1,1}= is a matrix of traceheader information. This is a (240 x number of traces) of type uint8.
    {1,2}= is a matrix of traces. This is a (tracelength x number of traces) of 4-byte floats.
  numcurtraces = this numeric value states the number of traces that have been written to the file already. This is required for SEG-Y Revision 1 Standards.
  numcurshots = this is a numeric value that depicts the number of shots that are currently in the file. This is optional but can be helpful when retrieving data.

SEGY_writeTraces is a tool that will write trace headers and traces to a file. This tool appends new traces to the end of the file. This function requires the number of traces that are already in the file to be specified. The headers must be written to the file first and then the traces may be appended. To write the headers to the file use SEGY_writeHeaders first.

NON-STANDARD SEG-Y FILES

When attempting to read SEG-Y files it is rare that they will conform to the SEG-Y Revision 1 Standard. This toolbox can read in these files, however, a few more steps are required. If the trace headers or binary header do not conform to the SEG-Y Revision 1 standard, then a new comma delimited file with definitions needs to be created. This file can either be generated using SEGY_StandardizeHeader, or any spreadsheet program. If a spreadsheet program is being used there must be the column headings: *Name, startByte,*

endByte, Type, SEGY_Rev1_EQ, and *Description*. *Name* is a variable name used to describe the value, *startByte* is where the value begins and the *endByte* is where the value ends, *Type* is the number format such as int16, and *Description* is a description of the variable. *SEGY_Rev1_EQ* is the SEG-Y Revision 1 equivalent; this should be one of the variable names defined in the Table 1 or Table 2. An example of the SEGY_StandardizeHeader interface can be seen in Figure 2. Once this file is created, the non standard file can be read in by adding 'bindefinitions', 'bindefinitionfilename.csv' for binary headers or 'trcdefinitions', 'tracedefinitionfilename.csv'. For binary header definitions using the tool SEGY_read it would look like this:

[txth, binh, trc, exth]=SEGY_read(filein, 'bindefinitions','nameofdefinitionsfile.csv')

This tool box uses bytes 3224 and 3225 in the file to determine whether or not the file is big endian or little endian, these bytes, in the binary header, are reserved to identify the format of the data traces. If these bytes are not reserved for the format of the data traces the values in the file will not be decoded correctly. If this occurs, use SEGY_endianSwap to correct for this. If a text header is read in incorrectly use SEGY_EditTextHeader to decode appropriately. Extended Headers are assumed to be 40 x 80 character arrays in this initial release.

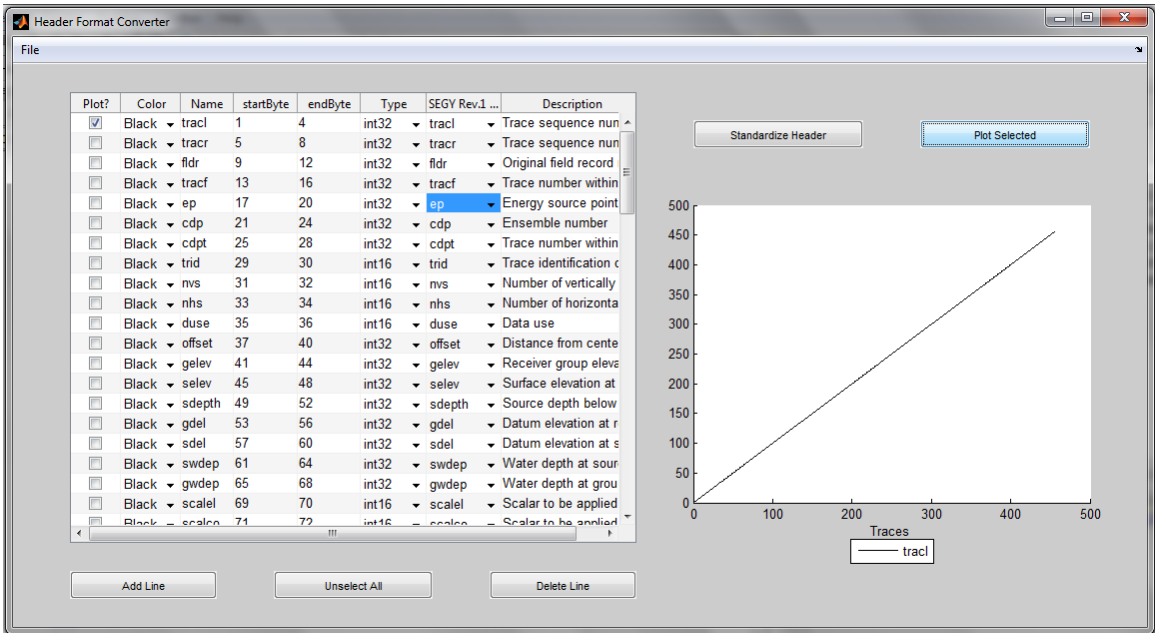

FIG. 2. This figure shows the SEGY_StandardizeHeader interface. The table contains the essential definition information along with some plotting controls for the plot on the right. Several different variables can be plotted at the same time.

CONCLUSIONS

The SEGY_Toolbox is a new set of Matlab tools for the purpose of reading, modifying and writing SEG-Y files. As this toolbox is new, suggestions and comments on the functionality of this toolbox will be appreciated.

EXAMPLES

Reading a SEG-Y file into Matlab.

Some analyses are more easily done in a Matlab environment rather than a processing environment. Using commercial processing software, processed data was saved as a SEG-Y file and then opened using the SEGY_Toolbox in Matlab. The goal is to find the common midpoint gather at 550 m and plot against offset.

1. Read in the SEG-Y file called prodata.sgy.
   ```
   [traces,texthead,binaryhead,extendedhead]=SEGY_read(prodata.sgy);
   ```
2. Get the cmp values and offset values for all traces
   ```
   cmp=SEGY_getHeader(traces,'cdp');
   offset=SEGY_getHeader(traces,'offset');
   ```
3. Search the cmp to find the value at 550m.
   ```
   ind=find(cmp==550);
   ```
4. Find the traces and offsets that belong to those values
   ```
   offset550=offset(ind);
   data=SEGY_getData(traces);
   data550=data(ind);
   ```
5. Plot an image of the data against offset.
   ```
   [traces,texthead,binaryhead,extendedhead]=SEGY_read(prodata.sgy);
   dt= SEGY_getHeader(binaryhead,'hdt'); or dt=SEGY_getHeader(traces,'dt');
   time=0:dt(1):size(data550,1)*dt(1);
   plotimage(data550,time,offset550);
   ```

Writing a SEG-Y file from Shot Records Created in Matlab.

In the CREWES toolbox there are tools that use finite differencing algorithms to create shot records. It would be useful to make several shot records and write them to a SEG-Y file.

1. Create the variables for the shot records.
   ```
   velocity=1500*ones(101,151);velocity(51:end,:)=4500;
   dx=10;
   dtstep=(dx*(1/sqrt(2)./(max(velocity(:))))-.0001); % For stability
   dt=0.002;
   tmax=3;
   xrec=0:dx:1500;
   zrec=0*xrec;
   [wavelet,twave]=ricker(dt,40);
   ```
2. Plan the seismic experiment. There will be 3 shots at 300m, 900m and 1200m. Each shot will have a receiver every 10 meters along the line for a total of 151 receivers per shot.

```
shotpos=[300,900,1200];
ns=length(0:dt:tmax);
numtraces=length(xrec)*3;% number of receivers * number of shots
```

3. Create the binaryheader and textheader using AFD_makeSEGYheaders, a utility that will return values that can be used with the SEGY_Toolbox.

```
[texthead,binaryhead]=AFD_makeSEGYheaders(dt,ns,numtraces,1,3);
```

4. Write the binaryheader and textheader to a SEG-Y file using SEGY_writeHeaders.

```
fileout='seismicexp1.sgy';
SEGY_writeHeaders(fileout,texthead,binaryhead);
```

5. Loop over the number of shots to create the synthetic finite differenced shot records, create traceheaders using AFD_makeSEGYtraceheaders and write the traces and the traceheaders to the SEG-Y file;

```
for k=1:length(shotpos)
snap1=zeros(101,151);snap1(1,(shotpos(k)/dx)+1)=1; snap2=snap1;
[seismogram,seis,t]=afd_shotrec(dx,dtstep,dt,tmax,velocity,snap1,snap2,...
xrec,zrec,wavelet,twave,1);
tracehead=AFD_makeSEGYtraceheaders(xrec,ones(size(xrec)),dt,...
shotpos(k),1,k);
SEGY_writeTraces(fileout,{tracehead, seismogram},length(xrec)*(k-1),k-1);
end
```

6. The SEG-Y file has now been created.

ACKNOWLEDGEMENTS

The authors would like to thank everyone currently or formerly in CREWES, who have previously worked on the Matlab SEG-Y tools. A large part of the code used in the current project was developed by Chad Hogan. Other contributors include Henry Bland and Carla Osborne.

REFERENCES

Norris, M. W., Faichney, A. K., 2002, SEG Y rev 1 Data Exchange format, SEG Technical Standards Committee. SEG.

Hogan, C., 2004, SEGY Matlab Tool Documentation, CREWES Matlab Toolbox