SegyMAT

Ed. version 1.5

COLLABORATORS

	TITLE : SegyMAT	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Thomas Mejer Hansen	November 1, 2011	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
1.5	October 28, 2011	ReadSegy.m : Added option 'traces' that allow fast reading of specific traces. When the 'minmax' option is used, the corresponding traces are first located thorugh header files, and then data are read using the 'traces' options. For larger files this casues the readomh tme to decrease significantly when using the 'minmax' option.	
1.4	April 05, 2011	ReadSegyHeader.m : Fixed 'SkipData' conflict with Robust Control Toolbox. Disabled Waitbar. wiggle.m : Allowed specification of line color. Allow overlaying wiggle plots. Allow NaN value in 'VA' style plotting. ReadSu : Fixed typo in line 221. MergeSegy.m : Added mfile to merge Segy Files.	
1.3	January 20, 2010	Added 'ReadSegyTraceHeaderValue' and 'WriteSegyTraceHeaderValue' that can be used to read and write the TraceHeaderValues one by one. Much faster that reading the whole dataset.	
1.2	January 7, 2009	Updated GUI to work for Matlab R2008a. Enabled loading of partial segyfile (using time and header ranges) from GUI (ctrl X). Enabled editing of the textual file header (both ASCII and EBCDIC)	
1.11	August 2008	Kristian Stormark contributed a change to GetSegyTraceHeader that reduce the number of discoperations causing a significant speed up.	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
1.08	March 2007	Urs Boeniger contributed a patch that allows arbitrary SegyTraceHeaders to be specified for WriteSegy.m.	
1.06		Fixed a bug taht casue a fixed length of 5011 samples in ReadSu.	
1.02		Cleaning up code to work with Octave 2.1.57.	
1.01		'jump' related fixes.	
1.00		Cleaning up some Matlab 7.0 specific bugs.	

Contents

List of Tables

Abstract

SegyMAT is a set of m-files that aims to ease importing and exporting SEG-Y files from Matlab. SegyMAT aims to: completely support SEG-Y revision 1; be easy to use in other projects; be compileable using the Matlab Compiler; be a Swiss Army knife dealing with the SEG-Y format in Matlab.

SegyMAT is not lightning fast. SegyMAT makes heavy use of 'structures'. Unfortunately structures are not very effective in terms of speed in Matlab. (Or they have not been implemented very effectively in SegyMAT). However structures make the implementation and maintainance easier, and the code (hopefully) easy to read. That said, some effort has been made to optimize SegyMAT for speed.

The latest version of SegyMAT is always available from Sourceforge : *http://segymat.sourceforge.net/*.

1 Requirements

has been developed and tested using Matlab 7.5 (R2007b) running on Linux and Windows XP. Any other Matlab supported platform should work.

As of version 1.02 Octave (version >2.1.64) is supported as well.

No Matlab toolboxes are required.

1.1 Local Installation

If you run Matlab with Java extensions (the default), you can run the pathtool, select the install directory (and subfolder GUI), and save the path and you are done ;

```
>> pathtool
```

To install without the Java gui (using 'matlab -nojvm') you can manually add the folder to Matlabs search path. If you unpack to /usr/share/matlab/SegyMAT simply do :

```
>> addpath /usr/share/matlab/SegyMAT -begin
>> addpath /usr/share/matlab/SegyMAT/GUI -begin
```

1.2 Global Installation

For a system wide installation add the following line (substituting the location of the directory)

```
>> addpath /usr/share/matlab/SegyMAT -begin
>> addpath /usr/share/matlab/SegyMAT/GUI -begin
```

to `pathdef.m`, usually located in `$MATLAB_INSTALL/toolbox/local/pathdef.m`

2 The Format

has been implemented using the *standard* as defined by SEG [1]

also has support for reading and writing the format used by CWP's *Seismic Unix* package (the SU format), which is merely a simplified version the format.

A short description of the formats follows here

2.1 Structure of a file

A file consists of a 3600 byte header; a number of extended textual headers; a number trace headers+data.

- A 3200 byte Textual File Header, ASCII or EBCDIC formated.
- A 400 byte Binary File Header
- A (optional) number of 'Extended Textual File Headers', 3200 bytes long, ASCII or EBCDIC formatted.
- A number of traces, separated into a 240 bytes long binary Trace Header, followed by the Trace Data, that can be formatted in a number of ways : IEEE, IBM Floating Point, 1,2 and 4 byte two's complement integers.

[1] The Society of Exploration Geophysicists

2.2 Structure of a SU file

A SU formatted file is just a simple version of a file, containing only trace information :

- No 3200 byte textual header and no extended textual headers.
- No binary header.
- The data must be formatted as IEEE.
- Data can be both little and big endian formatted.

2.3 What is supported in SegyMAT ?

As of version supports the following parts of the revision 0 and 1.

2.3.1 Textual file headers

The Textual 400 byte file header can be both ASCII and EBCDIC formatted, using revision 1.

Both ASCII and EBCDIC are supported since version 0.39.

2.3.2 Extended Textual Headers

In revision 1 a number of extended textual file headers are allowed.

As of version 0.9 extended textual headers are supported by .

2.3.3 Data Sample Format / Revision

The following data formats are supported :

Type	DataSampleFormat	Supported
1	4 Byte IBM Floating Point	Yes
2	4 Byte Fixed Point	No
3	2 Byte Fixed Point	No
4	4 Byte Fixed Point with Gain	No

Table 1: Supported revision 0 (1975)

Type	DataSampleFormat	Supported
1	4 Byte IBM Floating Point	Yes
2	4 Byte two's complement integer	Yes
3	2 Byte two's complement integer	Yes
4	4 Byte Fixed Point with Gain	No
5	4 Byte IEEE FLoating Pint	Yes
6	Not Specified	
7	Not Specified	
8	1 Byte Fixed Point with Gain	Yes

Table 2: Supported revision 1 (2002)

The type number is the number that should be used as '`dsf`' (Data Sample Format), for functions like ReadSegy.m, WriteSegy.m, WriteSegyStructure.m.

2.4 Segy Trace Header name definition

The definition of trace header names, and the location and preicision can be found by running

```
TraceHeaderDef(1);
```

which provides the folloing output:

```
 POS   PREC Traece Header Name
   0  int32 TraceSequenceLine
   4  int32 TraceSequenceFile
   8  int32 FieldRecord
  12  int32 TraceNumber
  16  int32 EnergySourcePoint
  20  int32 cdp
  24  int32 cdpTrace
  28  int16 TraceIdenitifactionCode
  30  int16 NSummedTraces
  32  int16 NStackedTraces
  34  int16 DataUse
  36  int32 offset
  40  int32 ReceiverGroupElevation
  44  int32 SourceSurfaceElevation
  48  int32 SourceDepth
  52  int32 ReceiverDatumElevation
  56  int32 SourceDatumElevation
  60  int32 SourceWaterDepth
  64  int32 GroupWaterDepth
  68  int16 ElevationScalar
  70  int16 SourceGroupScalar
  72  int32 SourceX
  76  int32 SourceY
  80  int32 GroupX
  84  int32 GroupY
  88  int16 CoordinateUnits
  90  int16 WeatheringVelocity
  92  int16 SubWeatheringVelocity
  94  int16 SourceUpholeTime
  96  int16 GroupUpholeTime
  98  int16 SourceStaticCorrection
 100  int16 GroupStaticCorrection
 102  int16 TotalStaticApplied
 104  int16 LagTimeA
 106  int16 LagTimeB
 108  int16 DelayRecordingTime
 110  int16 MuteTimeStart
 112  int16 MuteTimeEND
 114 uint16 ns
 116 uint16 dt
 118  int16 GainType
 120  int16 InstrumentGainConstant
 122  int16 InstrumentInitialGain
 124  int16 Correlated
 126  int16 SweepFrequenceStart
 128  int16 SweepFrequenceEnd
 130  int16 SweepLength
 132  int16 SweepType
 134  int16 SweepTraceTaperLengthStart
 136  int16 SweepTraceTaperLengthEnd
 138  int16 TaperType
 140  int16 AliasFilterFrequency
 142  int16 AliasFilterSlope
```

```
144  int16 NotchFilterFrequency
146  int16 NotchFilterSlope
148  int16 LowCutFrequency
150  int16 HighCutFrequency
152  int16 LowCutSlope
154  int16 HighCutSlope
156  int16 YearDataRecorded
158  int16 DayOfYear
160  int16 HourOfDay
162  int16 MinuteOfHour
164  int16 SecondOfMinute
166  int16 TimeBaseCode
168  int16 TraceWeightningFactor
170  int16 GeophoneGroupNumberRoll1
172  int16 GeophoneGroupNumberFirstTraceOrigField
174  int16 GeophoneGroupNumberLastTraceOrigField
176  int16 GapSize
178  int16 OverTravel
180  int32 cdpX
184  int32 cdpY
188  int32 Inline3D
192  int32 Crossline3D
196  int32 ShotPoint
200  int16 ShotPointScalar
202  int16 TraceValueMeasurementUnit
204  int32 TransductionConstantMantissa
208  int16 TransductionConstantPower
210  int16 TransductionUnit
212  int16 TraceIdentifier
214  int16 ScalarTraceHeader
216  int16 SourceType
218  int32 SourceEnergyDirectionMantissa
222  int16 SourceEnergyDirectionExponent
224   in32 SourceMeasurementMantissa
228  int16 SourceMeasurementExponent
230  int16 SourceMeasurementUnit
232  int32 UnassignedInt1
236  int32 UnassignedInt2
```

3 Reading SEG-Y files

3.1 ReadSegy

`ReadSegy.m` can be used to read files :

```
>> [Data,SegyTraceHeaders,SegyHeader]=ReadSegy('data.segy');
>> wiggle(Data,[],SegyHeader.time,[SegyTraceHeaders.cdp],'VA')
>> imagesc([SegyTraceHeaders.cdp],[SegyHeader.time],Data)
```

This will read `data.segy` using the revision and data sample format specified in the binary header, and plot the data.

`Data` is a 2D variable containing the seismic data. [Nsamples*Ntraces];

`SegyTraceHeaders`is a structure of size [1,Ntraces] structure containing all the header values from the traces. Type `SegyTraceHeaders` to see a list of header information. SegyTraceHeaders(9), list all header names and values of trace number 9.

To access an array of trace header values simply us square brackets as :

```
    cdp=[SegyTraceHeaders.cdp];
    offset=[SegyTraceHeaders.offset];
    ...
```

`SegyHeader` is a structure containing all the header values. Typing '>> SegyHeader' will list the names and values of all header values.

A number of arguments can be given to `ReadSegy`, controlling which part of the data set to read, and the format.

3.1.1 Read specific trace numbers

To read traces 100, 201 and 320 use e.g.

```
>> [Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'traces',[100 201 320]);
```

Use for example ReadSegyTraceHeadervalue.m and 'find' to find a list of trace ids (this is equivalent to using the 'minmax' option)

```
>> cdp=ReadSegyTraceHeaderValue(filename,'key','cdp');
>> traces = find(cdp>100 & cdp<200)
>> [Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'traces',[100 201 320]);
```

3.1.2 Read only every 5th trace

```
      >> [Data,SegyTraceHeaders,SegyHeader]=
      ReadSegy(filename,'jump',5);
```

3.1.3 To read time slice 0.5 < t < 5

```
      >> [Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'trange',.5,3);
```

3.1.4 Read only every 5th trace

```
      >> [Data,SegyTraceHeaders,SegyHeader]=
      ReadSegy(filename,'jump',5);
```

3.1.5 Read data in a CDP header range : 5000cdp5800

`cdp` can be changed to any other valid header name

```
>> [Data,SegyTraceHeaders,SegyHeader]=
      ReadSegy(filename,'minmax','cdp'5000,5800);
```

3.1.6 Read only header values

In some cases it can be desirable only to read the header files (file header and traceheaders). This will return an empty `Data` variable.

```
>> [Data,SegyTraceHeaders,SegyHeader]=
      ReadSegy(filename,'SkipData',1);
```

3.1.7 SEG-Y format revision

SEG-Y format revision number can be '0' (1975) or '1' (2002). By default the SEG-Y format revision number is read in the binary header, but this can be overruled using :

```
>> [Data,SegyTraceHeaders,SegyHeader]=
      ReadSegy(filename,'revision',0);
```

3.1.8 A specific Data Sample Format

One can overrule the Data Sample Format listed in the binary header, using the `dsf` argument. See Section 2.3.3 for a list of valid and supported values.

```
>> % Rev 0, IBM FLOATING POINT
>>  [Data,SegyTraceHeaders,SegyHeader]=
       ReadSegy(filename,'revision',0,'dsf',1);
>> % Rev 1, IEEE FLOATING POINT
>>  [Data,SegyTraceHeaders,SegyHeader]=
       ReadSegy(filename,'revision',1,'dsf',5);
```

If `dsf` is set to 5 and `revision` to 0, a warning message will occur, since data sample format 5 is only defined in revision 1. The revision is then automatically set to 1.

3.1.9 Force the use of a specific SegyHeader

```
>> [Data,SegyTraceHeaders,SegyHeader]=
      ReadSegy(filename,'SegyHeader',SegyHeader);
```

3.2 ReadSegyFast

`ReadSegyFast.m` is a faster implementation of ReadSegy.m since no trace header values are read. Thus this function will just return the seismic data and the -header. e.g. :

```
>> [Data,SegyHeader]=ReadSegy('data.segy');
>> imagesc(Data)
```

This will read `data.segy` using the revision and data sample format specified in the binary header, and plot the data.

3.2.1 ReadSegyFast options

As for ReadSegy.m the -revision a number of option can be used. ReadSegyFast.m uses the same convention as ReadSegy.m >.

The data sample format can be chosen using the 'revision' and 'dsf' tags. Also a 'SegyHeader' can be specified.

ReadSegyFast.m is currently optimized only for reading the whole -file, but the options 'jump' and 'trange' can be used (but will currently not result in faster read times).

Since the trace header values are not read, the 'minmax' option is not supported.

3.3 ReadSegyHeader

`ReadSegyHeader.m` reads the Binary Segy Header only. It can be called with the same options as ReadSegy.m

3.3.1 Force using little endian :

```
>> SegyHeader=ReadSegyHeader(filename,'endian','l');
```

3.4 ReadSegyTraceHeaderValue

`ReadSegyTraceHeaderValue.m`(Section 8.18) reads one trace header value into an array. This approach is much faster than to read the whole file

3.4.1 using keyword

To read a trace header value by its trace header key (see Section 2.4 for a list of defined keys) use

```
cdp=ReadSegyTraceHeaderValue(filename,'key','cdp');
SourceX=ReadSegyTraceHeaderValue(filename,'key','SourceX');
SourceY=ReadSegyTraceHeaderValue(filename,'key','SourceY');
plot(SourceX,SourceY)
```

3.4.2 using location+type

To read a trace header by its position in the trace header using a specific data sample format, use:

```
SourceX=ReadSegyTraceHeaderValue(filename,'pos',70,'precision','int32');
```

Take a look at Section 8.30, or check the manual at Section 2.4 to find the position of all trace header values.

3.5 ReadSegyConstantTraceLength

Assuming a constant trace length (which is much more common than not) allows much faster reading of parts of large file. For example to read trace number 2030, the whole segy data cube must be sequentially read, assuming variable trace length. Assuming constant trace length the trace can be directly (and fast) located in the data cube.

To read trace 2000 use

```
[Data,SegyTraceHeader,SegyHeader]=ReadSegyConstantTraceLength(filename,'trace',2000);
```

To read traces 1-2000 and 2020-2040 use

```
[Data,SegyTraceHeader,SegyHeader]=ReadSegyConstantTraceLength(filename,'trace ↩
    ',[1:2000,2020:2040]);
```

3.5.1 using keywords

Several keywords can be used to efficiently read parts of larger files.

To read only the part of a file with SourceX between 1000-2000 and SourceY between 4000-5000 use :

```
[Data,SegyTraceHeader,SegyHeader]=ReadSegyConstantTraceLength(filename,'minmax','SourceX ↩
    ',1000,2000,'minmax','SourceY',4000,5000]);
```

3.6 ReadSu

`ReadSu.m` works similar to ReadSegy.m and the same input parameters can be used. A `SuHeader` can optionally be returned, that simple is a mostly empty file header.

```
>> [Data,SuTraceHeaders,SuHeader]=ReadSu(filename);
```

4 WriteSegy

WriteSegy.m is available to write a Matrix to disk as a SEG-Y file.

WriteSegyStructure.m is available to write a SEG-Y file using a specified set of headers. This is of special use if one wants to load a seismic data set, work with the data in Matlab, and the write the data to disk using the same header values.

4.1 WriteSegy

`WriteSegy` can be used to save a matrix of data, in Matlab as a file.

4.1.1 Specify values for the SEG-Y Header

Here `dt` is a scalar and `Inline`, `Crossline`, `X` and `Y` are arrays of values of size size(data,2)

```
>> WriteSegy('datacube.segy',data,
    'dt',.004,'Inline3D',Inline,'Crossline3D',Crossline,
    'cdpX',X,'cdpY',Y);
```

4.1.2 Specify revision

```
>> WriteSegy('test.segy',seisdata,'revision',0); % SEG-Y Revision 0
>> WriteSegy('test.segy',seisdata,'revision',1); % SEG-Y Revision 1
```

4.1.3 Specify data sample format

See Section 2.3.3 for a list of valid and supported values for the datasample format `dsf`.

```
>> % Force Revision 1 and IEEE Floating point :
>> WriteSegy('test.segy',seisdata,'dsf',5,'revision',1);
>>
>> % Force Revision 0 and IBM Floating point :
>> WriteSegy('test.segy',seisdata,'dsf',1,'revision',0);
```

4.2 WriteSegyStructure

`WriteSegyStructure` can be used to write a seismic data to disk given that both `SegyHeader`, `SegyTraceheaders` and the data `Data` are known. They can be obtained using ReadSegy.m like ;

```
>> [Data,TraceHeaderInfo,SegyTraceHeaders,SegyHeader]=ReadSegy('data.segy');
```

To write the data using `WriteSegyStructure` simply do

```
>> WriteSegyStructure('datacube.segy',SegyHeader,SegyTraceHeaders,Data);
```

4.2.1 Force revision

```
>> % Revision 0
>> WriteSegyStructure('datacube.segy',SegyHeader,
                      SegyTraceHeaders,Data,'revision',0);
>> % Revision 1
>> WriteSegyStructure('datacube.segy',SegyHeader,
                      SegyTraceHeaders,Data,'revision',1);
```

4.2.2 Force Data Sample Format

See Section 2.3.3 for a list of valid and supported values for the datasample format `dsf`.

```
>> % To force the use of SEG Y revision 0 and data sampling format IEEE :
>> WriteSegyStructure('datacube.segy',SegyHeader,
                       SegyTraceHeaders,Data,'revision',1,'dsf',5);
```

4.3 WrityeSegyTraceHeaderValue

`WriteSegyTraceHeaderValue.m`(Section 8.33) writes one trace header from an array into the Trace Hader of a SEG-Y file.

4.3.1 using keyword

To read a read, edit and write the 'cdp' header values (see Section 2.4 for a list of defined keys) use for example:

```
cdp=ReadSegyTraceHeaderValue(file,'key','cdp');  % READ CDP
cdp=cdp+10;                                      % change CDP
WriteSegyTraceHeaderValue(file,cdp,'key','cdp'); % UPDATE CDP
```

4.3.2 using location+precision

To manually update a trace header at a specific location, using a specific data type (precision) use for for example:

```
% Update all trace header values starting at position 72, in integer32
% format, to the values in array 'data'
ntraces=311;
data=[1:1:311]*10;
WriteSegyTraceHeaderValue(filename,data,'pos',72,'precision','int32');
d_header=ReadSegyTraceHeaderValue(filename,'pos',72,'precision','int32');
```

Take a look at Section 8.30, or check the manual at Section 2.4 to find the position of all trace header values.

5 Misc

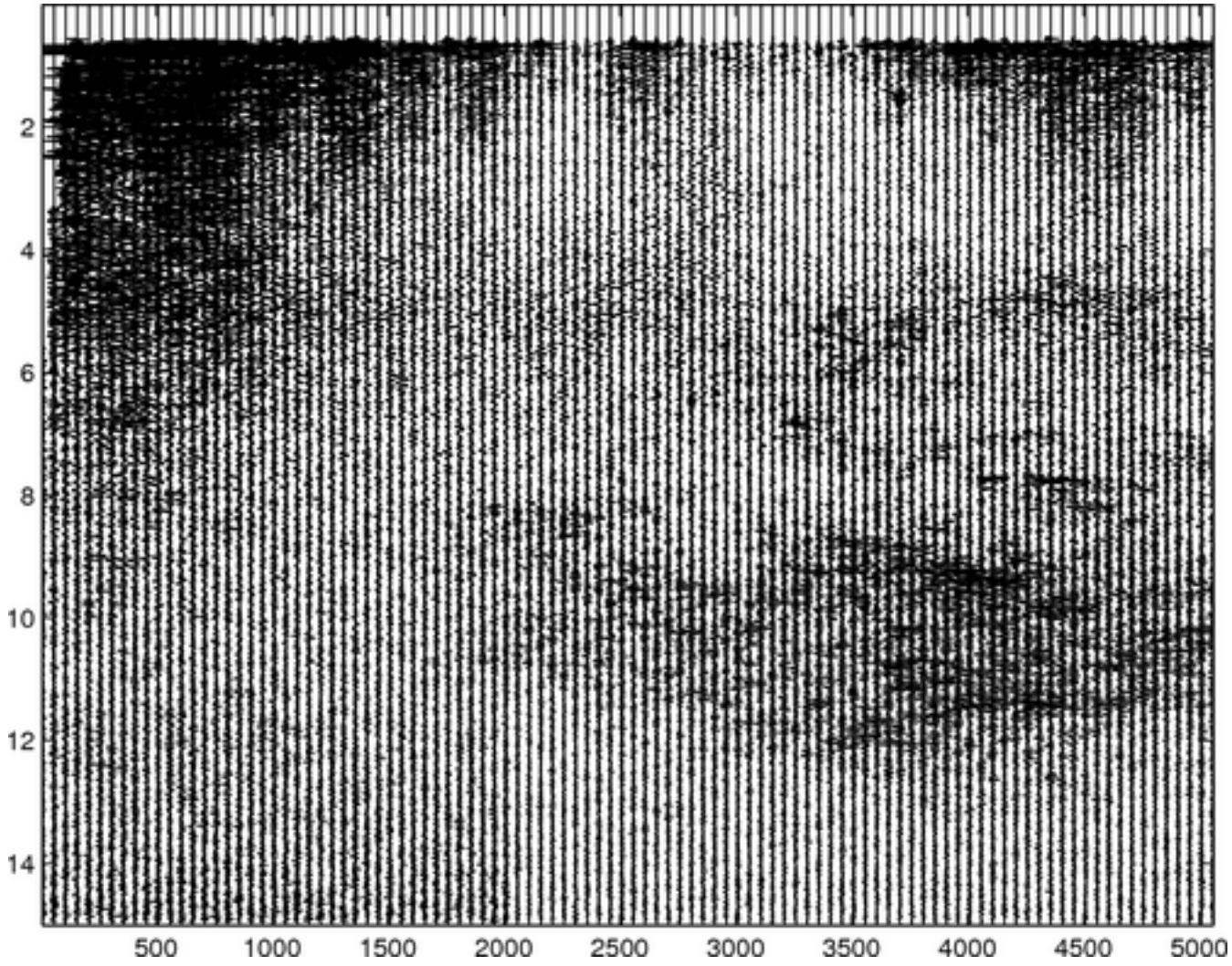
Here are some m-files developed as part of SegyMAT

5.1 Visualization

5.1.1 wiggle

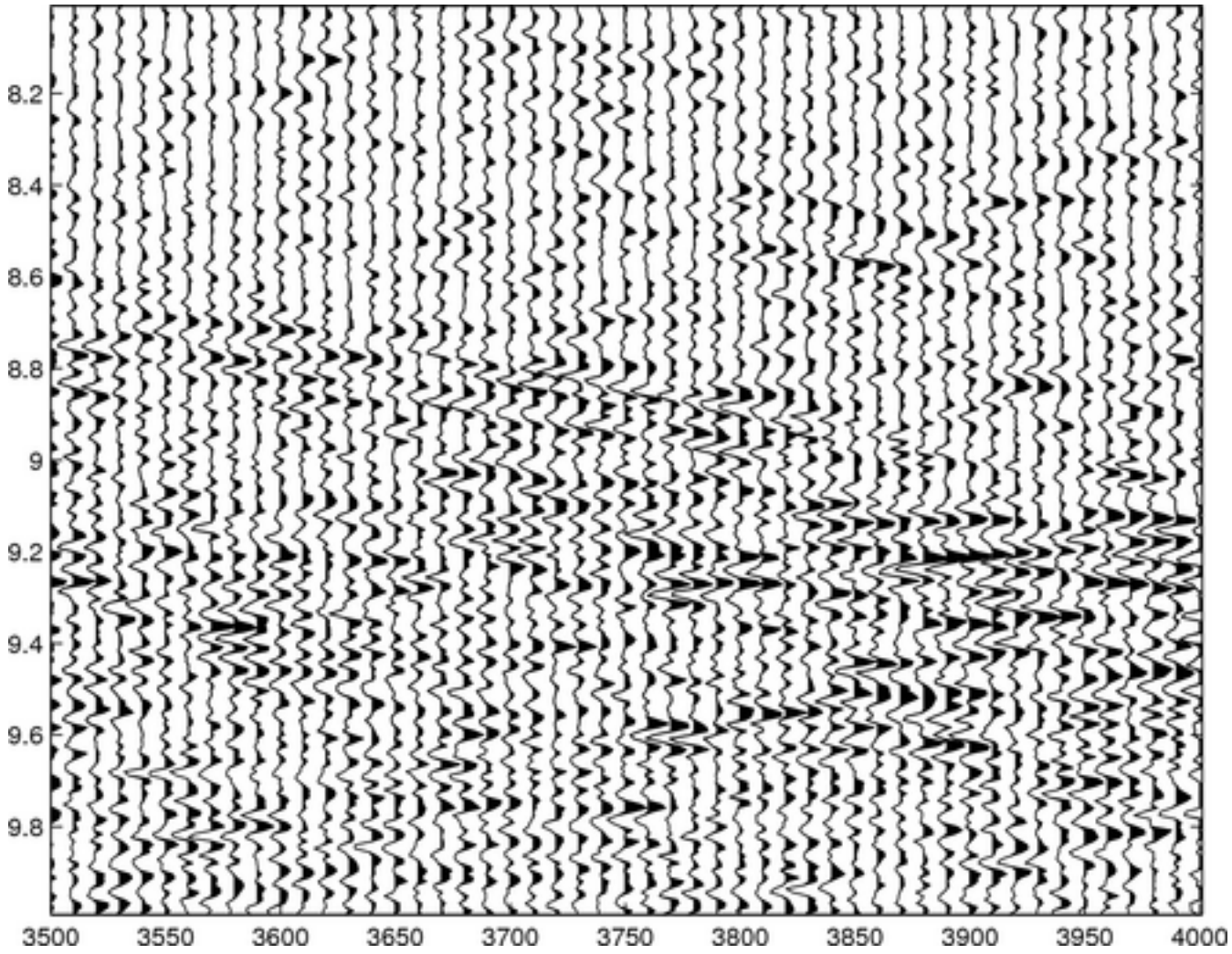
`wiggle.m` is used to plot seismic data using using wiggle or variable area type plotting, optionally on top of an image plot of the data wiggle type:

```
[Data,STH,SH]=ReadSegy('841_m.sgy','jump',10);
                      % get from
                      % http://gdr.nrcan.gc.ca/seismtlitho/archive/le/stacks_fgp_e.php
wiggle([STH.TraceNumber],SH.time,Data,'wiggle',700);
```



Variable area:

```
[Data,STH,SH]=ReadSegy('841_m.sgy','jump',10,'minmax','TraceNumber',3500,4000,'trange ↩
    ',8,10);
wiggle([STH.TraceNumber],SH.time,Data,'VA',700);
```

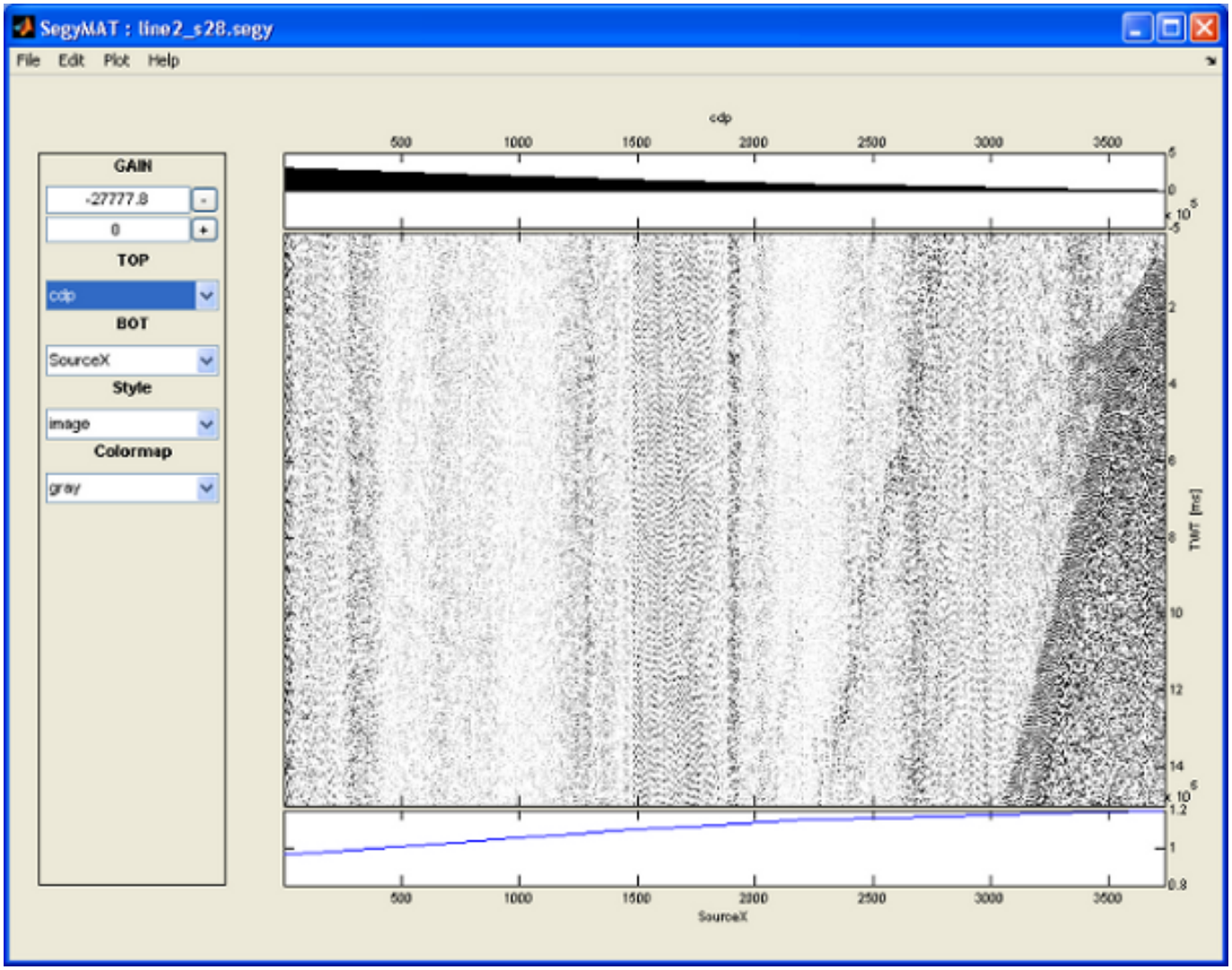


6 Graphical User Interface utilities

Note : this section (all graphical related features) applies only to Matlab, and is unsupported in Octave.

6.1 SegyMAT GUI

Calling `segymat` opens a graphical user interface for viewing and editing SEG-Y formatted files:

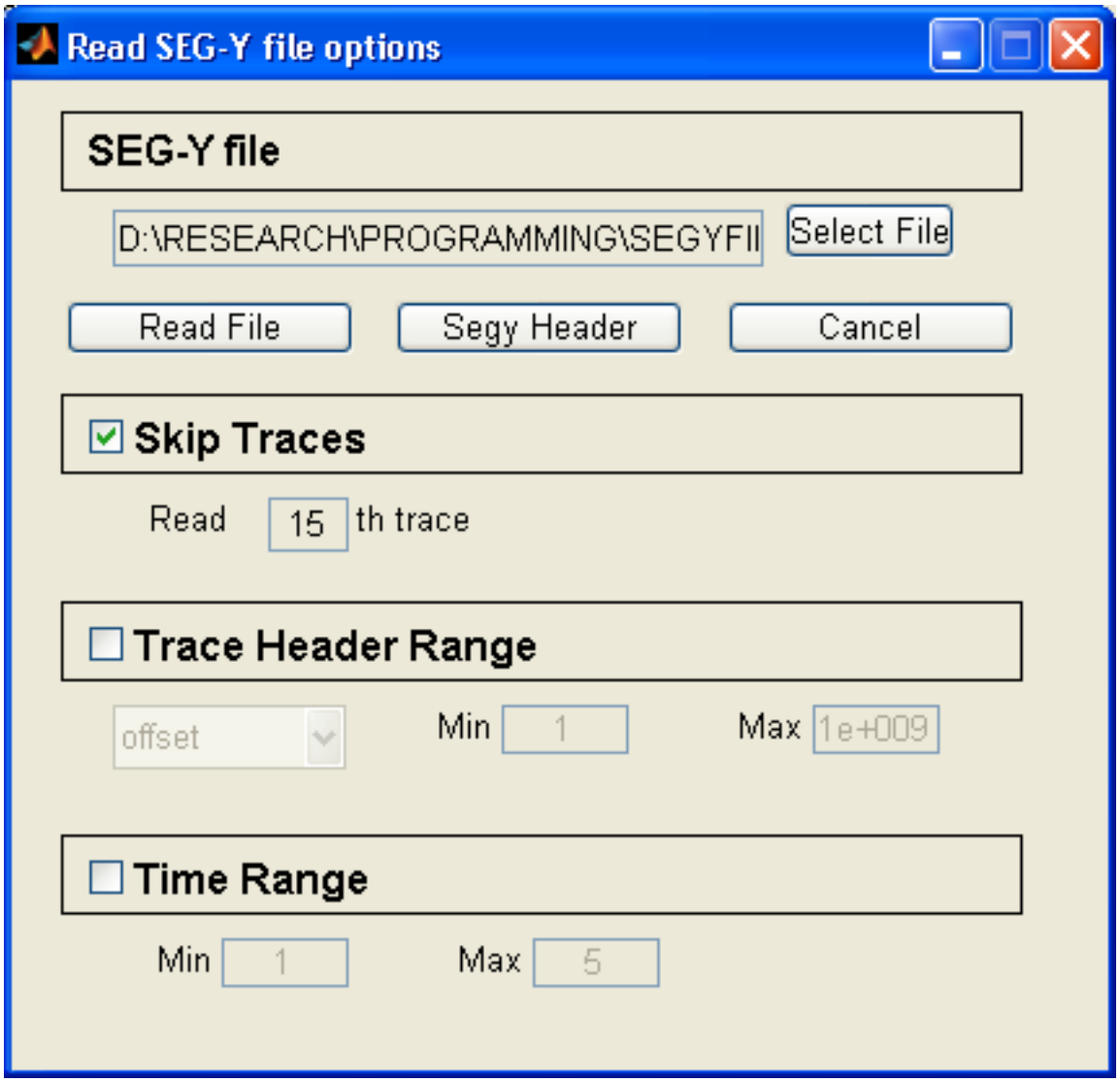


6.1.1 simple reading SEG-Y files

Select File->Open to select a SEG-Y file, which will be read using the original SEG-Y header information.

6.1.2 expert reading SEG-Y files

Select File->Open(expert) to handle SEG-Y header values prior to reading the file, and to read in only part of the SEG-Y file.



6.2 Keyboard shortcuts for moving/zooming/gain

zooming in: + zooming out: -

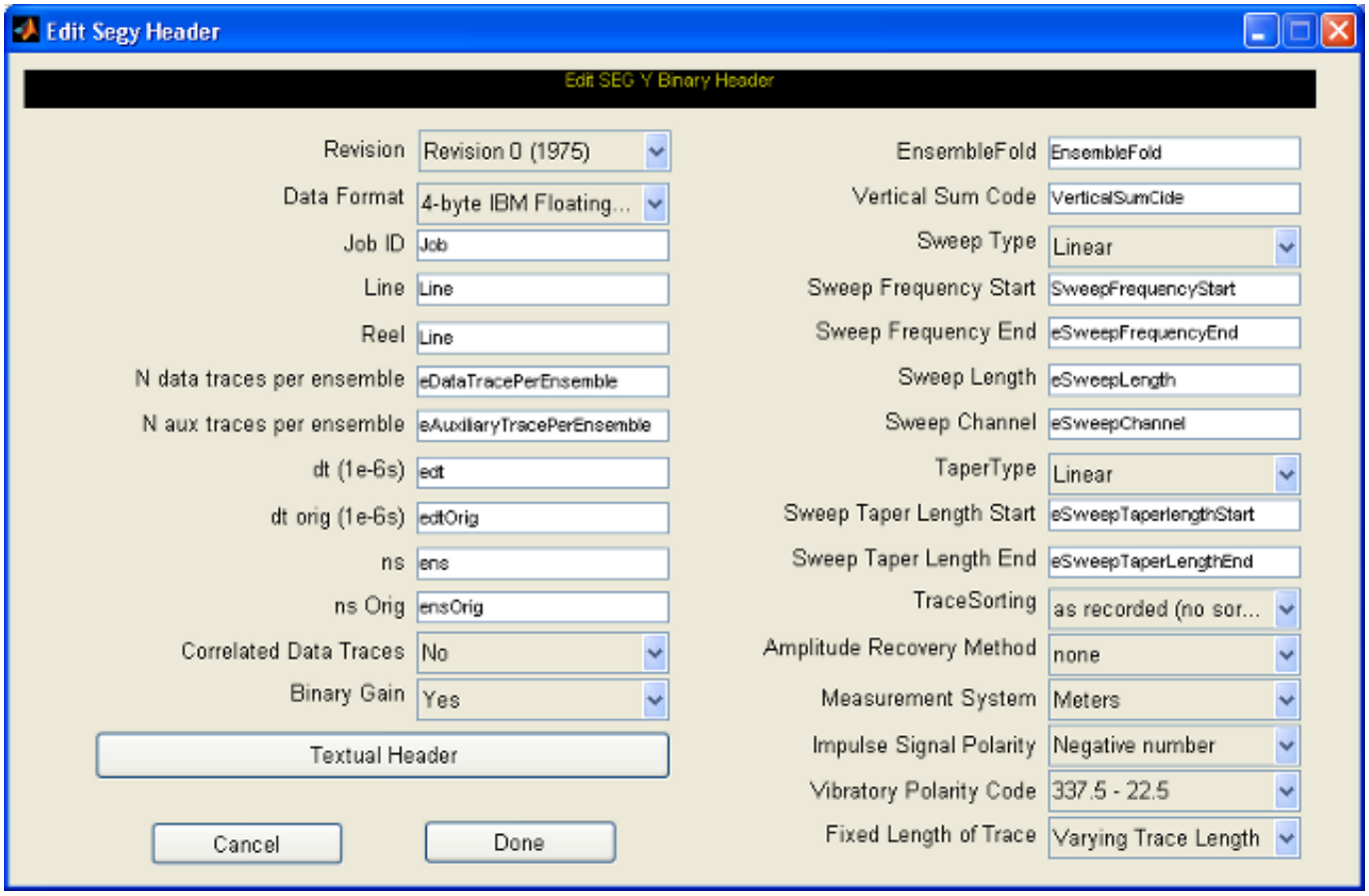
6.3 Editing the SEG-Y header

`GUIEditSegyHeader` is a GUI for editing the SEG-Y header.

```
[Data,STH,SH]=ReadSegy('841_m.sgy');
SH=GUIEditSegyHeader(SH);
```

Shortcut	Action
+	Increase gain
-	Decrease gain
4	Pan left
6	Pan right
2	Pan down
8	Pan up
1	Pan down/left
3	Pan down/right
7	Pan up/left
9	Pan up/right
5	Center
a / arrow left	Zoom in
z / arrow right3	Zoom out
h	toggle hiding plotting preferences

Table 3: Keyboard shortcuts

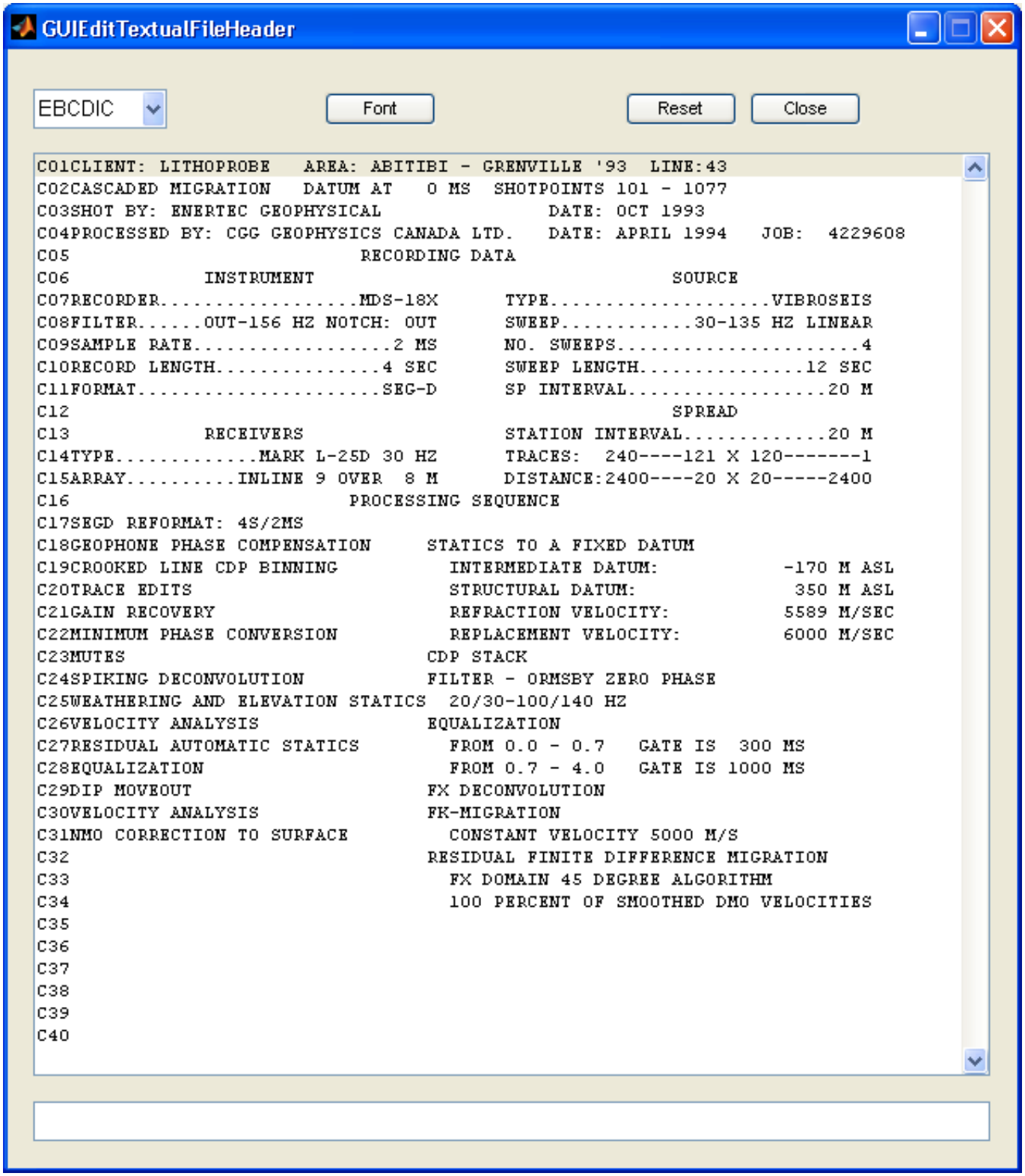


From this GUI it is possible to view and edit the Textual File Header (Section 6.3)

6.4 Viewing the textual file header

`GUIEditTextualFileHeader` is a GUI for viewing the textual file header (either in ASCII of EBCDIC format) [editing is not yet implemented].

```
[Data,STH,SH]=ReadSegy('841_m.sgy');
SH=GUIEditTextualFileHeader(SH);
```



This GUI is integrated into `GUIEditSegyHeader`(Section 6.3).

7 Acknowledgement

Thanks to Brian Farrelly, Norsk Hydro Research Centre, Bergen, Norway, for supplying functions to convert between IBM Floating Point format and doubles. (June, 2002. ver 0.35 ->)

Thanks to Urs Boeringer for adding a patch to WriteSegy, to enable use of an arbitrary set of TraceHeader values. (March 2007. ver 1.08 ->)

Thanks to *Sourceforge* for hosting the project.

8 M-file Reference

8.1 CheckSegyTraceHeader

```
SegyTraceHeader=CheckSegyTraceHeader(SegyTraceHeader);

Checks that all fields of the SegyTraceHeader is set.
If not, they are initialized.
```

8.2 Contents

```
SegyMAT : A toolbox to read, write and manipulating SEG Y formatted files
Version 1.00

New Features.
  README

Main
  ReadSegy          - Reads Segy File
  ReadSegyHeader    - Reads SegyHeader from Segy File
  ReadSegyFast      - Reads Segy File in fast mode. No header values will be read.
  WriteSegy         - Write Segy formatted data
  WriteSegyStructure- Write Segy formatted data using SegyMAT data structures

  ReadSu            - Reads a SU formatted file.
  ReadSuFast        - Reads a SU formatted file in fast mode. No header values will be  ↩
      read.
  WriteSu           - Write SU formatted data
  WriteSuStructure  - Write Su formatted data using SegyMAT data structures

Lower Level IO
  GetSegyHeader          - Reads the segyheader of a SEGY Y formatted file
  GetSegyHeaderBasics    - Default Segy header settings

  GetSegyTrace.m         - Read Segy Trace Header and Data from filehandle
  GetSegyTraceHeader     - Read Segy Trace Header from filehandle
  GetSegyTraceData       - Read Segy Trace Data from filehandle

  PutSegyHeader          - Write Segy Header to filehandle
  PutSegyTrace           - Write Segy Trace Header and Data to filehandle

  InitSegyTraceHeader    - Initalize all fields in the SegyTraceheader
  CheckSegyTraceHeader.m - Check a SegyTraceHeader for all required fields


SU<-> SEG-Y conversion
  Su2Segy - Convert SU formatted files to SEG Y
  Segy2Su - Convert SEG Y formatted files to SU

Plotting
  wiggle  - wiggle/variable area/image plotting of seismic data
```

```
Misc
  ibm2num - Convert IBM 32 bit floatto double
  num2ibm - Convert IEEE 754 doubles to IBM 32 bit floating point format
  ebcdic2ascii - convert ebcdic to ascii format
  SegymatVerbose - controls amount of info written to screen
  SegymatVersion - Return the current SegyMAT version


Seismic Processing :
  SegyMAT_GAIN : 'agc' and 'power' gain.

(C) 2001-2004 Thomas Mejer Hansen, tmh@gfy.ku.dk/thomas@cultpenguin.com


  Overloaded functions or methods (ones with the same name in other directories)
     help H5T/Contents.m
     help H5D/Contents.m
     help H5P/Contents.m
     help H5R/Contents.m
     help H5F/Contents.m
     help H5E/Contents.m
     help H5A/Contents.m
     help H5S/Contents.m
     help H5I/Contents.m
     help H5G/Contents.m
     help H5ML/Contents.m
     help H5/Contents.m
     help H5Z/Contents.m
     help timer/Contents.m
     help icinterface/Contents.m
     help instrument/Contents.m
     help serial/Contents.m
     help audiorecorder/Contents.m
     help audioplayer/Contents.m
     help strel/Contents.m
```

8.3 GetSegyHeader

```
GetSegyHeader : Reads the segyheader of a SEGY Y formatted file

Call :
 [SegyHeader]=GetSegyHeader(segyid);

 segyid can be a filehandle or a filename


(C) 2001-2004 Thomas Mejer Hansen, tmh@gfy.ku.dk/thomas@cultpenguin.com


   This program is free software; you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation; either version 2 of the License, or
   (at your option) any later version.

   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
```

```
   along with this program; if not, write to the Free Software
   Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
```

8.4 GetSegyHeaderBasics

```
GetSegyHeaderBasics : Default Segy Header Header settings

Call :
Rev=GetSegyHeaderBasics
```

8.5 GetSegyTrace

```
GetSegyTrace : Reads a seg y trace, data and header

[SegyTraceHeader,SegyData]=GetSegyTrace(segyid,TraceStart,DataFormat,ns);
```

8.6 GetSegyTraceData

```
GetSegyTraceData : Get Segy trace data if filehandle

Call :

  tracedata=GetSegyTraceData(segyid,ns,SegyHeader,SkipData
```

8.7 GetSegyTraceHeader

```
GetSegyTraceHeader : Reads a seg y trace, data and header

[SegyTraceHeader]=GetSegyTraceHeader(segyid,TraceStart,DataFormat,ns);


(C) 2001-2004 Thomas Mejer Hansen, thomas.mejer.hansen@cultpenguin.com

Revisions:
07/2008 Kristian Stormark (<kristian.stormark@gmail.com>) : Reduce the
        number of discoperations causing a significant speed up
```

8.8 GetSegyTraceHeaderInfo

```
GetSegyTraceHeaderInfo : Returns a array of a SEGY Y TraceHeader value

Call :
[value]=GetSegyHeaderInfo(SegyTraceHeaders,header)

header is a header value like 'cdp','dt','TraceNumber'
```

8.9 InitSegyTraceHeader

```
InitSegyTraceHeaders : returns an empty SegyTraceHeader structure

EX:
SegyTraceHeader=InitSegyTraceHeader(ns,dt);
```

8.10 MakeXmlRef

8.11 MergeSegy

```
MergeSegy : Merge multiple SEGY files

Example :
   MergeSegy('*.sgy','merge.sgy')

   f{1}='file1.sgy';
   f{2}='file2.sgy';
   f{3}='file3.sgy';
   MergeSegy(f,'merge.sgy')


Note: All imput segy files must have the same constant trace length
      The SEGY header of the merged SEGY file will be the SEGY header
      form the first input SEGY file.
```

8.12 PutSegyHeader

```
PutSegyHeader : Writes SEG-Y header to disk.
PutSegyHeader(segyid,SegyHeader)

(C) 2001-2004, Thomas Mejer Hansen, tmh@gfy.ku.dk/thomas@cultpenguin.com

   This program is free software; you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation; either version 2 of the License, or
   (at your option) any later version.

   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with this program; if not, write to the Free Software
   Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
```

8.13 PutSegyTrace

```
PutSegyTrace(segyid,tracedata,SegyTraceHeader,TraceStart);

Write a SegyTrace to a filehandle 'segyid'

(C) 2001-2004, Thomas Mejer Hansen, tmh@gfy.ku.dk/thomas@cultpenguin.com
```

8.14 ReadSegy

```
ReadSegy : Reads a SEG Y rev 1 formatted file

Call :
[Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename);

To read time slice 0.5<t<5 :
[Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'trange',.5,3);
Skip every 5th trace :
[Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'jump',5);
Read data in a CDP header range : 5000<cdp<5800 :
(change cdp to any other valid TraceHeader value)
[Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'minmax','cdp'5000,5800);
Read only the header values (Data will return empty)
[Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'SkipData',1);

SEG-Y format revision number can be '0' (1975) or
'100' (similar to '1') (2002).
By default the SEG-Y format revision number is read in the
binary header, but this can be overruled using :
[Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'revision',0);

Read using a specific Data Sample Format :
Rev 0, IBM FLOATING POINT
[Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'revision',0,'dsf',1);
Rev 1, IEEE FLOATING POINT
[Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'revision',1,'dsf',5);

A SegyHeader can be forced on the SEG-Y file using :
[Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'SegyHeader',SegyHeader);
The SegyHeader can be obtain by GetSegyHeader(segyfilename), and
then edited.

To read using little endian :
[Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'endian','l');

Combine any combination of the above
[Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'jump',1,'minmax','cdp',5300,5400);


Plot the data using e.g.
imagesc([SegyTraceHeaders.cdp],SegyHeader.time,Data);
wiggle([SegyTraceHeaders.TraceNumber],SegyHeader.time,Data);

(C) 2003-2004, Thomas Mejer Hansen, tmh@gfy.ku.dk
```

8.15 ReadSegyConstantTraceLength

```
ReadSegyConstantTraceLength : Reads a SEG Y rev 0/1 formatted file
                              Assumes CONSTANT TRACE LENGTH
                              which allows much faster code

Call :
[Data,SegyTraceHeaders,SegyHeader]=ReadSegyConstantTraceLength(filename);

To read time slice 0.5<t<5 :
[Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'trange',.5,3);
Skip every 5th trace :
[Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'jump',5);
Read data in a CDP header range : 5000<cdp<5800 :
(change cdp to any other valid TraceHeader value)
[Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'minmax','cdp',5000,5800);
Use several minmax entries
[Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'minmax','cdp',5000,5800,'minmax',' ↩
    SourceX',10,20);


Read from trace 13-18:
[Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'trace',13:18);
Read from trace 13-18 and 100-130:
[Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'trace',[13:18,100:130]);

SEG-Y format revision number can be '0' (1975) or
'100' (similar to '1') (2002).
By default the SEG-Y format revision number is read in the
binary header, but this can be overruled using :
[Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'revision',0);

Read using a specific Data Sample Format :
Rev 0, IBM FLOATING POINT
[Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'revision',0,'dsf',1);
Rev 1, IEEE FLOATING POINT
[Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'revision',1,'dsf',5);

A SegyHeader can be forced on the SEG-Y file using :
[Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'SegyHeader',SegyHeader);
The SegyHeader can be obtain by GetSegyHeader(segyfilename), and
then edited.

To read using little endian :
[Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'endian','l');

Combine any combination of the above
[Data,SegyTraceHeaders,SegyHeader]=ReadSegy(filename,'jump',1,'minmax','cdp',5300,5400);


Plot the data using e.g.
imagesc([SegyTraceHeaders.cdp],SegyHeader.time,Data);
wiggle([SegyTraceHeaders.TraceNumber],SegyHeader.time,Data);

(C) 2003-2004, Thomas Mejer Hansen, tmh@gfy.ku.dk
```

8.16 ReadSegyFast

```
ReadSegyFast : Reads a SEG Y rev 1 formatted file, without header values (faster than  ↩
    ReadSegy)
```

```
Call :
[Data]=ReadSegyFast(filename);
and equivalent to :
[Data]=ReadSegy(filename);


Read only the data of a SegFile - NOT Their headers.
Much faster than ReadSegy

'minmax', 'skip'
```

8.17 ReadSegyHeader

```
ReadSegyHeader : Reads a SEG Y Binary Header

Call :
[SegyHeader]=ReadSegyHeader(filename);

To read using little endian :
[SegyHeader]=ReadSegyHeader(filename,'endian','l');
```

8.18 ReadSegyTraceHeaderValue

```
ReadSegyTraceHeaderValue : Read a spedicifc trace header value

Call:
   % By Name
   cdp=ReadSegyTraceHeaderValue(filename,'key','cdp');
   SourceX=ReadSegyTraceHeaderValue(filename,'key','SourceX');
   SourceY=ReadSegyTraceHeaderValue(filename,'key','SourceY');

   % By location in Trace Header
   SourceX=ReadSegyTraceHeaderValue(filename,'pos',72,'precision','int32');

   % Call 'TraceHeaderDef(1)' to see a list of TraceHeader 'key' names

See also WriteSegyTraceHeaderValue, TraceHeaderDef
```

8.19 ReadSu

```
ReadSu : Reads a SU formatted file (Seismic Unix)

Call :
[Data,SuTraceHeaders,SuHeader]=ReadSu(filename);

To read in big endian format (default):
[Data,SuTraceHeaders,SuHeader]=ReadSu(filename,'endian','b');
To read in little endian format :
[Data,SuTraceHeaders,SuHeader]=ReadSu(filename,'endian','l');


To read in trace data as 'int32' :
[Data,SuTraceHeaders,SuHeader]=ReadSu(filename,'DataFormat','int32');
```

```
To read time slice 0.5<t<5 :
[Data,SuTraceHeaders,SuHeader]=ReadSu(filename,'trange',.5,3);
Skip every 5th trace :
[Data,SuTraceHeaders,SuHeader]=ReadSu(filename,'jump',5);
Read data in a CDP header range : 5000<cdp<5800
(change cdp to any other valid TraceHeader value)
[Data,SuTraceHeaders,SuHeader]=ReadSu(filename,'minmax','cdp'5000,5800);

Combine any combination of the above
[Data,SuTraceHeaders,SuHeader]=ReadSu(filename,'jump',1,'minmax','cdp',5300,5400);
```

8.20 ReadSuFast

```
ReadSuFast

PURPOSE : reads a SEISMIC section i  SU format in big endian format,
          strips the headers and returns the field in the matrix seis.
          If nx==0 and nt<>0, nx will be computed
          If nt==0 and nx<>0, nt will be computed

Call : function seis=ReadSuFast(fileid,nt,nx,'byteorder');
          byteorder : 'l' for little or 'b' for big endian (Default : Native )

BY : TMH 1/8 1997
Updated by Thomas Mejer Hansen : 22-03-1999
```

8.21 Sac2Segy

```
Sac2Segy : Reads SAC formatted data into a SegyMAT (SGY) structure

CALL :
  [Data,SegyTraceHeader,SegyHeader]=Sac2Segy(files_in,segyfile_out,varargin)

  files_in : Either a single filename or a strcture of filenames
           files_in='d1.SAC';
           or
           files_in{1}='d1.SAC';
           files_in{2}='d2.SAC';

Examples :
  [D,STH,SH]=Sac2Segy('','test.segy','FixedLengthTraceFlag',1);
             converts all SAC files into one SEGY file (test.segy), using
             a FixedLengthTraceFlag of 1. This is compatible with mosty
             any SEGY reader.

  [D,STH,SH]=Sac2Segy('','test.segy','FixedLengthTraceFlag',0);
             converts all SAC files into one SEGY file (test.segy), using
             a FixedLengthTraceFlag of 0, allowing varying trace length of SEGY files
             This is only compatible with revision 1 of the SEGY format.

  [D,STH,SH]=Sac2Segy('file.sac');
             convert file.sac to file.segy

  [D,STH,SH]=Sac2Segy('file.sac','another_file.segy');
             convert file.sac to another_file.segy
```

```
  Force little endian byte format for SAC file:
  Sac2Segy('file.sac','test.sgy','endian','l');

Relies on sac2mat.m

Download SAC files from : http://www.iris.edu/hq/ssn/events
```

8.22 Segy2Su

```
Segy2Su : Converts SEGY file to SU format

Call : Segy2Su(filename,ReadSegyOption)
   Replaces the filename suffix to '.su';
   'ReadSegyOptions' are the same as to 'ReadSegy'

 See also : ReadSegy
```

8.23 SegyMAT_GAIN

```
SegyMAT_GAIN : Gain plugin for SegyMAT

[Data,SegyTraceHeaders,SegyHeader]=SegyMAT_GAIN(Data,SegyTraceHeaders,SegyHeader,varargin ↩
    );

ex. AGC using AGC window of 100 ms :
[Data]=SegyMAT_GAIN(Data,SegyTraceHeaders,SegyHeader,'agc',.1);
ex. apply t^(pow), pow=2
[Data]=SegyMAT_GAIN(Data,SegyTraceHeaders,SegyHeader,'pow',2);


(C) Thomas Mejer Hansen (thomas@cultpenguin.com), 2002
```

8.24 SegyMATdemo1

```
SegyMATdemo1 : Creates, Reads and plots a Segy File;
```

8.25 SegymatHelp

8.26 SegymatRevision

```
SegymatRevision - Returns the revision history

Call : [Revision]=SegymatRevision
```

8.27 SegymatVerbose

```
SegymatVerbose : Writes out verbose information to the screen


Call :
  SegymatVerbose(text,verboselevel)
  prints out 'text' to screen if verboselevel is higher than threshold
  set in m-file.
```

8.28 SegymatVersion

```
SegymatVersion - Returns the version and release date

[ver,d]=SegymatVersion;
```

8.29 Su2Segy

```
SU2Segy : Converts SEGY file to SU format
```

8.30 TraceHeaderDef

```
TraceHeaderDef : Defines names, position, and precision for Trace Headers

% To get a Matlab structure with trace header definitions call:
STH==TraceHeaderDef;
% To get a list fo trace header definision listed on the screen call:
STH==TraceHeaderDef(1)

See also: ReadSegyTraceHeaderValue, WriteSegyTraceHeaderValue
```

8.31 WriteSegy

```
WriteSegy : writes data to disk using SEGY REV 1 standard.

EX
WriteSegy('datacube.segy',data,'dt',.004,'Inline3D',Inline,'Crossline3D',Crossline,'cdpX ↩
    ',X,'cdpY',Y);

to use a specific SEG revision use :
WriteSegy('test.segy',seisdata,'revision',0); % SEG-Y Revision 0
WriteSegy('test.segy',seisdata,'revision',1); % SEG-Y Revision 1

to use a specific Data Sampling Format use :
WriteSegy('test.segy',seisdata,'dsf',1); % IBM FLAOTING POINT

Forice Revision 1 and IEEE Floating point :
WriteSegy('test.segy',seisdata,'dsf',5,'revision',1);

See also : WriteSegyStructure, WriteSu, WriteSuStructure
```

8.32 WriteSegyStructure

```
WriteSegyStructure : writes data to disk using SEGY REV 0 and 1 standards.

EX
WriteSegyStructure('datacube.segy',SegyHeader,SegyTraceHeaders,Data);

To force the use of SEG Y revision 0
WriteSegyStructure('datacube.segy',SegyHeader,SegyTraceHeaders,Data,'revision',0);
To force the use of SEG Y revision 1
WriteSegyStructure('datacube.segy',SegyHeader,SegyTraceHeaders,Data,'revision',1);
To force the data sampling format to be IBM Floating Point
WriteSegyStructure('datacube.segy',SegyHeader,SegyTraceHeaders,Data,'dsf',1);

To force the use of SEG Y revision 0 and data sampling format IEEE :
WriteSegyStructure('datacube.segy',SegyHeader,SegyTraceHeaders,Data,'revision',1,'dsf',5) ↩
    ;

See the dokumentation for for proper values of 'dsf'
```

8.33 WriteSegyTraceHeaderValue

```
WriteSegyTraceHeaderValue : Write trace header valaue at specific location

Call:

   % Update all trace header values starting at position 72, in integer32
   % format, to the value 30
   data=30;
   WriteSegyTraceHeaderValue(filename,data,'pos',72,'precision','int32',);

   % Update all trace header values starting at position 72, in integer32
   % format, to the values in array 'data'
   ntraces=311;
   data=[1:1:311]*10;
   WriteSegyTraceHeaderValue(filename,data,'pos',72,'precision','int32');
   d_header=ReadSegyTraceHeaderValue(filename,'pos',72,'precision','int32');

   % Update the 'cdp' TraceHeader value:
   cdp=ReadSegyTraceHeaderValue(file,'key','cdp');  % READ CDP
   cdp=cdp+10;                                      % change CDP
   WriteSegyTraceHeaderValue(file,cdp,'key','cdp'); % UPDATE CDP

   Call 'TraceHeaderDef(1)' to see a list of TraceHeader 'key' names

See also ReadSegyTraceHeaderValue, PutSegyTraceHeader, TraceHeaderDef
```

8.34 WriteSu

```
WriteSu : writes data to disk using SEGY REV 2 standard.

EX
WriteSu('datacube.su',data,'dt',.004,'Inline3D',Inline,'Crossline3D',Crossline,'cdpX',X,' ↩
    cdpY',Y);
```

```
to use a specific SEG revision use :
WriteSu('test.su',seisdata,'revision',0); % SEG-Y Revision 0
WriteSu('test.su',seisdata,'revision',1); % SEG-Y Revision 1

to use a specific Data Sampling Format use :
WriteSu('test.su',seisdata,'dsf',1); % IBM FLAOTING POINT

Forice Revision 1 and IEEE Floating point :
WriteSu('test.su',seisdata,'dsf',5,'revision',1);
```

8.35 WriteSuStructure

```
WriteSuStructure : writes data to disk using SU-CWP format

EX
WriteSuStructure('datacube.segy',SegyHeader,SegyTraceHeaders,Data);
```

8.36 ascii2ebcdic

```
ascii2ebcdic : Converts ASCII formatted text to EBCDIC formatted text

CALL : ebcdic=ascii2ebcdic(ascii);

ascii  : Array on unsigned integers
ebcdic : Array on unsigned integers

(C) 2002-2009, Thomas Mejer Hansen, tmh@gfy.ku.dk/thomas.mejer.hansen@gmail.com
```

8.37 cmap_rwb

8.38 ebcdic2ascii

```
ebcdic2ascii : Converts EBCDIC formatted text to ASCII formatted text

CALL : ascii=ebcdic2ascii(ebcdic);

ebcdic : Array on unsigned integers
ascii  : Array on unsigned integers

(C) 2002-2004, Thomas Mejer Hansen, tmh@gfy.ku.dk/thomas@cultpenguin.com
```

8.39 gse2segy

8.40 ibm2num

```
ibm2num : convert IBM 32 bit floating point format to doubles
   x=num2ibm(b)
b is a matrix of uint32
x is a corresponding matrix of doubles


See also num2ibm
```

8.41 isoctave

```
isoctave : checks of octave
```

8.42 num2ibm

```
num2ibm : convert IEEE 754 doubles to IBM 32 bit floating point format
   b=num2ibm(x)
x is a matrix of doubles
b is a corresponding matrix of uint32

The representations for NaN and inf are arbitrary

See also ibm2num
```

8.43 pick_line

```
pick_line : pick a line from a figure;


Based on doc(ginput);
```

8.44 progress_txt

```
progress_txt : console based progress bar

Ex1 :
  for i=1:10000;
    progress_txt(i,10000,'Ciao');
  end

Ex1 :

  for i=1:10;
  for j=1:10;
  for k=1:10;
    progress_txt([i j k],[10 100 1000],'i','j','k');
  end
  end
  end

TMH/2005, thomas@cultpenguin.com
```

8.45 read_gse_int

8.46 sac2mat

```
[SACdata,SeisData,filenames] = SAC2MAT('file1','file2',..., 'filen',endian )

reads n SAC files file1, file2, filen
and converts them to matlab
format. The filenames can contain globbing characters (e.g. * and ?).
These are expanded and all matching files loaded.

files are assumed big endian formatted (e.g. SUN), little endian can be
forced using endian='l': sac2mat('file1.sac','l');


SACSUN2MAT( cellarray ) where cellarray={'file1','file2',...,'filen'}
is equivalent to the standard form.

SACdata is an n x 1 struct array containing the header variables
        in the same format as is obtained by using MAT function
        of SAC2000.
        SACdata(i).trcLen contains the number of samples.

SeisData is an m x n array (where m=max(npts1, npts2, ...) )
        containing the actual data.

filenames is a n x 1 string cell array with the filenames actually read.

Note that writing

 [SACdata,SeisData] = sac2mat('file1','file2',..., 'filen' ,endian)

is equivalent to the following sequence

sac2000
READ file1 file2 .. filen
MAT

(in fact the failure of above sequence to work properly on my
system motivated this script).


SAC2MAT was written by F Tilmann (tilmann@esc.cam.ac.uk)
based on sac_sun2pc_mat  by C. D. Saragiotis (I copied the
routines doing the actual work from this code but
used a different header structure and made the routine
flexible).
It was tested on MATLAB5 on a PC but
should work on newer versions, too.

(C) 2004

Update 10/2008 by Thomas Mejer Hansen: Merged sac2sun2mat and sacpc2mat
into sac2mat.m
```

8.47 sacpc2mat

```
[SACdata,SeisData,filenames] = SACPCMAT('file1','file2',..., 'filen' )

reads n SAC files file1, file2, filen (SAC files are assumed to have
PC byte order) and converts them to matlab
format. The filenames can contain globbing characters (e.g. * and ?).
These are expanded and all matching files loaded.

SACPCMAT( cellarray ) where cellarray={'file1','file2',...,'filen'}
is equivalent to the standard form.

SACdata is an n x 1 struct array containing the header variables
        in the same format as is obtained by using MAT function
        of SAC2000.
        SACdata(i).trcLen contains the number of samples.

SeisData is an m x n array (where m=max(npts1, npts2, ...) )
        containing the actual data.

filenames is a n x 1 string cell array with the filenames actually read.

Note that writing

 [SACdata,SeisData] = sacsun2mat('file1','file2',..., 'filen' )

is equivalent to the following sequence

sac2000
READ file1 file2 .. filen
MAT

(in fact the failure of above sequence to work properly on my
system motivated this script).


SACPC2MAT was written by F Tilmann (tilmann@esc.cam.ac.uk)
based on sac_sun2pc_mat  by C. D. Saragiotis (I copied the
routines doing the actual work from this code but
used a different header structure and made the routine
flexible).
It was tested on MATLAB5 on a PC but
should work on newer versions, too.

(C) 2004
```

8.48 sacsun2mat

```
[SACdata,SeisData,filenames] = SACSUN2MAT('file1','file2',..., 'filen' )

reads n SAC files file1, file2, filen (SAC files are assumed to have
SUN byte order) and converts them to matlab
format. The filenames can contain globbing characters (e.g. * and ?).
These are expanded and all matching files loaded.

SACSUN2MAT( cellarray ) where cellarray={'file1','file2',...,'filen'}
is equivalent to the standard form.

SACdata is an n x 1 struct array containing the header variables
        in the same format as is obtained by using MAT function
```

```
        of SAC2000.
        SACdata(i).trcLen contains the number of samples.

SeisData is an m x n array (where m=max(npts1, npts2, ...) )
        containing the actual data.

filenames is a n x 1 string cell array with the filenames actually read.

Note that writing

 [SACdata,SeisData] = sacsun2mat('file1','file2',..., 'filen' )

is equivalent to the following sequence

sac2000
READ file1 file2 .. filen
MAT

(in fact the failure of above sequence to work properly on my
system motivated this script).


SACSUN2MAT was written by F Tilmann (tilmann@esc.cam.ac.uk)
based on sac_sun2pc_mat  by C. D. Saragiotis (I copied the
routines doing the actual work from this code but
used a different header structure and made the routine
flexible).
It was tested on MATLAB5 on a PC but
should work on newer versions, too.

(C) 2004
```

8.49 testWriteSegy

```
testWriteSegy : Script to test WriteSegy and WriteSegyStructure
```

8.50 wiggle

```
wiggle : plot wiggle/VA/image plot

Call
   wiggle(Data); % wiggle plot
   wiggle(Data,scale); % scaled wiggle plot
   wiggle(x,t,Data); % wiggle plt
   wiggle(x,t,Data,'VA') % variable Area (pos->black;neg->transp)
   wiggle(x,t,Data,'VA2') % variable Area (pos->black;neg->red)
   wiggle(x,t,Data,'wiggle',scale); % Scaled wiggle
   wiggle(x,t,Data,'wiggle',scale,showmax); % Scaled wiggle and max
                                            showmax traces.
   wiggle(x,t,Data,'wiggle',scale,showmax,plimage); % wiggle + image
   wiggle(x,t,Data,'wiggle',scale,showmax,plimage,caxis); % wiggle +
                                                          scaled image

Data : [nt,ntraces]
x : [1:ntraces] X axis (ex [SegyTraceheaders.offset])
t : [1:nt] Y axis
```

```
style : ['VA'] : Variable Area
        ['wiggle'] : Wiggle plot
scale : scaling factor, can be left empty as []
showmax [scalar] : max number of traces to show on display [def=100]
plimage [0/1] : Show image beneath wiggles [def=0];
caxis [min max]/[scalar] : amplitude range for colorscale


MAKE IT WORK FOR ANY X-AXIS !!!
```