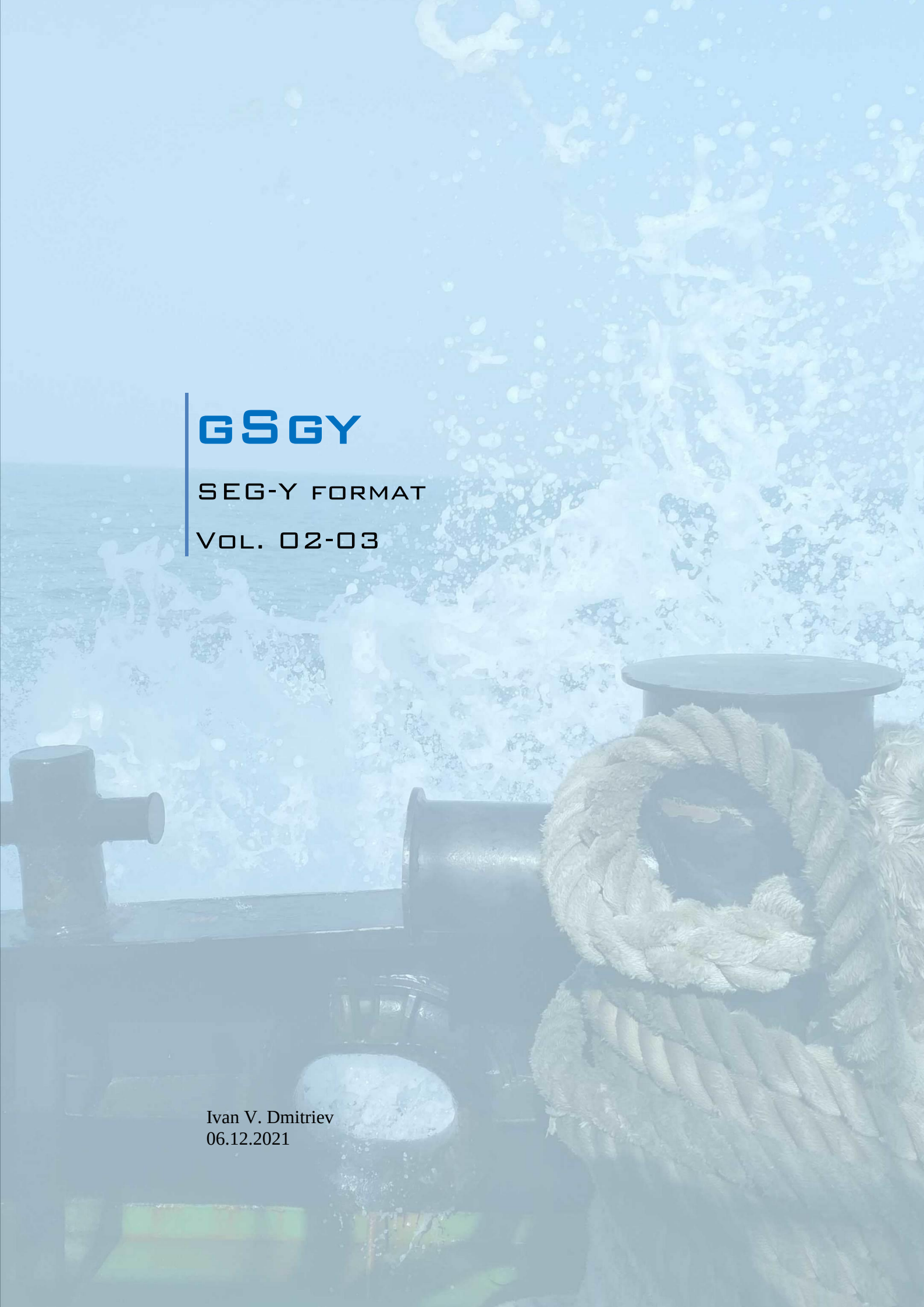


gSgy

SEG-Y format

Vol. 02-03

Ivan V. Dmitriev
06.12.2021

Contents

Tables list

Figures list

1. gSgy functions set description

MatLab functions set for reading, writing and manipulations with Sgy-files. The functions are shown in ***Table 1.1***. The functions are designed for marine one-channel seismic (SBP) processing, but some of them can be used for headers applying to multi-channels data (for example, geometry attaching).

Sgy-file read in MatLab to three variables, named below SgyHead (binary, textural and extended textural headers), Head (trace headers) and Data (Matrix with trace's data). Several files can be read to Dataset: SgyHead(1..n), Head(1..n), Data{1..n}. It can be useful for quick access to all surveyed files Headers.

The "trace time/coordinates" can be converted to fields GpsKP, GpsDay, GpsTime, GpsE, GpsN, GpsH (see gNav manual) using function gSgyDTEN; those fields can used for export to AutoCAD, P1/90 files creation, etc.

gData set's functions can be used for Data matrix processing: gain, filtering, shifting, etc; some functions are included in ***Table 1.1***. Own gSgy set's functions deal with specific Sgy-information contained in SgyHead and Head variables.

The SBP-horizon structure, used for time-to-depth conversion is the same to Horizon-structure in gData, but includes additional fields (for example, Vbelow – velocity below horizon).

Table 1.1 gSgy and the same functions

Function name	Function description
Read/Write/Manipulations for files	
gSgyRead	Read Sgy variables [SgyHead,Head,Data] from file.
gSgyWrite	Write Sgy variables [SgyHead,Head,Data] to file.
gSgyAppendFiles	Append Sgy-files to variables [SgyHead,Head,Data].
gSgyDatasetImport	Add Sgy-files from file-list or folder to Dataset.
gSgyDatasetExport	Export Sgy-variables from Dataset to files
gSgyDatasetPack	Save data-matrixes from Data to temporary files
gSgyDir2PL	Read coordinates form directory with *.sgy to PL-structure
gSgyJpMatRead	Read [SgyHead,Head,Data] from files *.mat and *.jp2 (includes Data)
gSgyJpMatWrite	Write [SgyHead,Head,Data] to files *.mat and *.jp2 (includes Data)
Text Header	
gSgyTextAscii2Ebcdic	Convert ASCII to EBCDIC (Sgy text header)
gSgyTextEbcdic2Ascii	Convert EBCDIC to ASCII (Sgy text header)
gSgyTextCorrect	Replace symbols in Text Header
gSgyTextCorrectScript	Script; Create text-headers for Sgy-files and apply it to files in folder
DTEN-fields	
gSgyDTEN	Create Nav's fields GpsKP,GpsDay,GpsTime,GpsE,GpsN,GpsH
gSgyDTENinv	Re-calculate Sgy-fields using GpsDay,GpsTime,GpsE,GpsN,GpsH
gFields2PL	Create PL-structure using GpsKP,GpsE,GpsN,GpsH fields (see gFields manual)
Manipulations with Data matrix	
gDataTraceFilt	Traces/columns filtration with slice-window (see gData manual)
gDataTraceWeight	Weighting traces/columns group (rows filtering) with slice-window (see gData manual)

Function name	Function description
gData2DFilt	Matrix filtration with 2D-slice-window (see gData manual)
gDataNormPL	Normalize traces/columns between horizon1 and horizon2 (see gData manual)
gDataGainPL	Traces/pings Gain (see gData manual)
gDataToPL	Shifted data-matrix (traces) from horizon1 to horizon2 (see gData manual)
gDataCalcAttrib	Calculate "attributes" for data-matrix (along traces/columns) (see gData manual)
gSgyDatasetAppend	Append Sgy variables from Dataset
gSgyDeleteTraces	Delete traces from Sgy variables
gSgySetEndRec	Change Sgy Traces Length
gSgySetDelayRecTime	Change Sgy DelayRecordingTime for each trace; add or delete trace's parts for Data
gSgyDataToPL	Shifted Sgy Data matrix from polyline1 to polyline2; apply shift mod(1ms) to DelayRecTime
gSgyResample	Interpolate Sgy-section form current SampleInterval to new SampleInterval (data repeat or delete)
gSgyUnificate	Create Unificate Sgy-section: Head.SampleNumber= =const (SgyHead.FixedLengthTraceFlag=1); Head.DelayRecordingTime= =const; sample interval= =const (interp Head and insert Nan to Data)
Velocity	
gDataPLPickHandle	Image/Matrix horizon handle-picking (see gData manual)
gDataPLPickAuto	Image/Matrix horizon auto-picking (see gData manual)
gSgyHorizCreate	Create horizon for time-to-depth conversion, using 1- single value, 2-vector, 3- Horizon structure
gSgyHoriz2VelMatrix	Create Velocity matrix for time-to-depth conversion, using horizons structure (see gData)
gSgyHorizTime2Depth	Convert Data-matrix from time to depth, using Velosity matrix; the horizons structure is converted too
gSgyHorizTime2DepthScript	Convert folder with Sgy-files from time to depth as 2-layer section (water and rock); used pre-picked depth from Head-structure field 'UnassignedInt1'.
Draw	
gDataDrawSection	Draw image using Data matrix (see gData manual)
gSgyDraw3Section	Draw Sgy Data matrix as 3D image
gSgyDrawSectionPipe	Draw Sgy Data matrix converted to depth and create tips with hyperbola for pipe // in progress

1.1 Sgy read and write – variables and fields

The functions are based on the paper "SEG Y rev 1 Data Exchange format. SEG Technical Standards Committee. Release 1.0, May 2002".

Sgy-file includes the 3200 byte Textural File Header, 400 byte Binary File Header, optional 3200 byte Extended Textural File Headers, and Data Blocks, consists of 240 byte Trace Head and Trace Data (*Figure 1.1*). Sgy-file read in MatLab to three variables, named below SgyHead (binary, textural and extended textural headers), Head (trace headers) and Data (Matrix with trace's data).

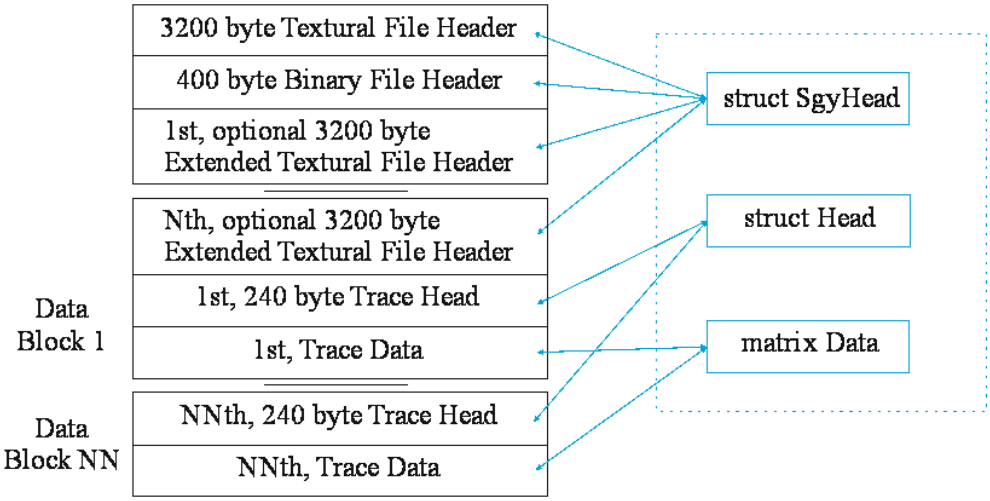


Figure 1.1 Sgy records structure

SgyHead and Head structures

SgyHead include 3200 byte Textural File Header, 400 byte Binary File Header, optional 3200 byte Extended Textural File Headers from sgy-file. Head include 240 byte 240 byte Trace Heads from Data Blocks.

SgyHead and Head can include the additional fields or specific interpretation for field values; there is described in *Table 1.2*.

Table 1.2 Sgy Header additional fields

Segy_rev1 Byte	Ge0MLib Field Name and Segy_rev1 Description	Ge0MLib Description
	SgyHead.Descript	Sgy fields description. Created by gSgyRead.
	SgyHead.fName	The file name used for file reading. Created by gSgyRead.
	SgyHead(n).fNameTmp	Temporary file name and path (which contained Data-matrix). Created for Dataset by gSgyDatasetImport
	SgyHead.Endian	Set Big-endian ordering or Little-endian ordering for sgy-file. Created by gSgyRead.
	SgyHead.FDataSampleFormat	Forced Sample Format (1, 2, 3, 4, 5 or 8); if empty, then set from SgyHead.DataSampleFormat field. Used as Sample Format for read/write operations. Created by gSgyRead.

Segy_rev1 Byte	Ge0MLib Field Name and Segy_rev1 Description	Ge0MLib Description
35-36	Head.DataUse 1 = Production 2 = Test	Bit flags Bit0 – set if “Production” Bit1 – set if “Test” Bit2 – set if “Seismic Trace were generated” Bit3 – set if “Original Coordinates and Time were killed as spikes”
91-92	Head.WeatheringVelocity Weathering velocity (ft/s or m/s as specified in Binary File Header bytes 3255-3256).	Water sound velocity (effective value) for marine survey.
93-94	Head.SubWeatheringVelocity Subweathering velocity (ft/s or m/s as specified in Binary File Header bytes 3255-3256).	Under-bottom “conventional” seismic waves velocity for marine survey.
233-236	Head.UnassignedInt1 Unassigned — For optional information; Uint32	Processed Sea bottom by SBP (using shifting / heave compensation / swell filter / reduction to MBES bottom, etc); Values in trace’s samples from sea surface.
237-240	Head.UnassignedInt2 Unassigned — For optional information; Uint32	Original SBP-bottom picking result by SBP (without swell filter or heave compensation) Values in trace’s samples from sea surface.
	Head.GpsDate	Created from Year-Day using gSgyDTXY_On.
	Head.GpsTime	Created from HH-MM-SS using gSgyDTXY_On.
	Head.GpsE	Created from Head.(FieldX), Head.CoordinateUnits and SgyHead.Nav using gSgyDTXY_On.
	Head.GpsN	Created from Head.(FieldY), Head.CoordinateUnits and SgyHead.Nav using gSgyDTXY_On.
	Head.GpsH	Created when coordinate transformation using gSgyDTXY_On.
	Head.SeaDepth	The Sea Depth calculated using MBES data for “trace” coordinates

Field Head.UnassignedInt1 must be manual corrected, if gDataToPL-function was used. Example:

```
>> [Data1,dL]=gDataToPL(Data,frL,toL);Head.UnassignedInt2=Head.UnassignedInt2+(toL-frL);
```

Compare with SegyMAT Library

The gSgy field names the same SegyMat Library structure names, there are several differences in Head structure names:

1) field MuteTimeEND renamed to MuteTimeEnd;

2) field GeophoneGroupNumberRoll1 (last symbol “one”) renamed to GeophoneGroupNumberRoll;

3) removed “additional” fields SegyMAT_TraceStart, TimeBaseCodeText, SegyMAT_TraceDataStart, TraceValueMeasurementUnitText.

As opposed to SegyMAT the trace-headers not grouped by trace-to-trace-structures, but each field contain vector (row) with length equal trace number.

The gSgy and SegyMAT reading comparison is show in ***Figure 1.2***.

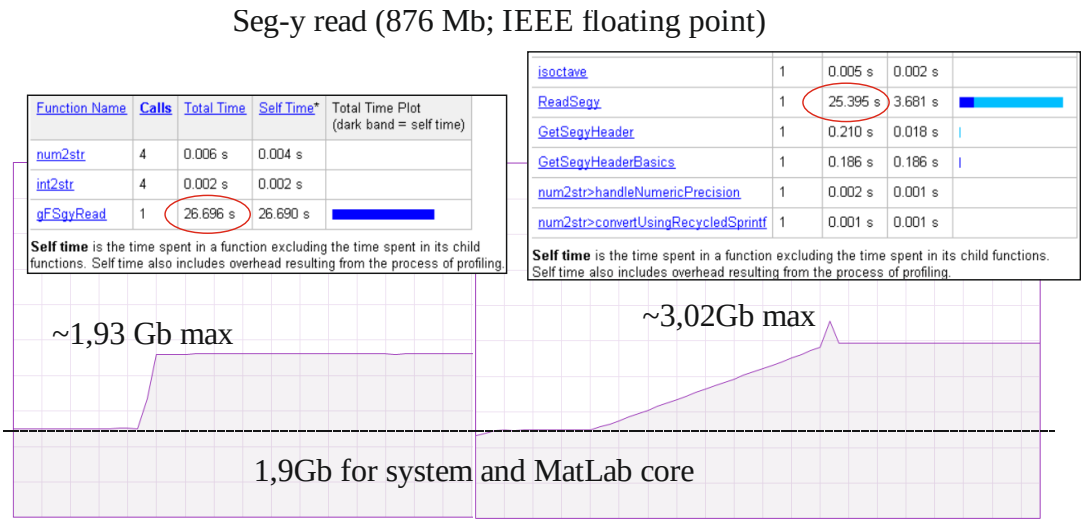


Figure 1.2 gSgy and SegyMAT reading comparison

1.2 SgyDataset Structures

Several files can be read to Dataset: SgyHead(1..n), Head(1..n), Data{1..n}, using gSgyDatasetImport function. Dataset can be writes using gSgyDatasetExport function.

The Data{1..n} cells can includes:

-- Data matrix with traces;

-- Tmp-file name, which contained Data matrix. Tmp-file is read/write using functions gDataSave and gDataLoad. Functions gSgy controls Data’s entry; if there is file-name, the data from file will be loaded using SgyHead(n).fNameTmp field.

1.3 DTEN-fields

GpsKP, GpsDay, GpsTime, GpsE, GpsN, GpsH are named the DTEN-fields, it is define measurement time and coordinates in planar projection. The fields are used for “universalized” time/coordinates fields for different structures, for example sgy-structure, xtf-structure, jsf-structure (the functions were used gSgyDTEN, gSgyDTENinv, gXtfDTEN, gJsfDTEN).

1.4 Horizon structure

The Sgy-Horizon structure is the enchanted “Data-Horizon on-matrix (image) Poly-Line structure”, there are follow features:

-- it can contain one point for one matrix column;

-- it is uninterrupted in the matrix “segment”;

-- usually, it continuous from fist column to last column, but fist and last points can defined to any column numbers;

-- the Sgy-Horizon contained field Vbelow – velocity below horizon.

This structure can be created by manual picking (function gDataPLPickHandle) or auto-picking (function gDataPLPickAuto). The additional field Vbelow can be add manually or using function gSgyHorizCreate. The structure’s field’s names and descriptions are shown in ***Table 1.3***. The SBP-Horizon structure used as input parameter for time-to-depth conversion functions or for surfaces creation.

Table 1.3 SBP-Horizon structure fields’ names

Field name	Field description
PLName	Horizon (polyline) name. String; Information-content.
Type	Horizon (polyline) type: ‘Horizon’ String; Information-content.
KeyLineDraw	String key for Horizon (polyline) drawing in MatLab’s figure (for example: '-r','xb'). String; Information-content.
pX	Matrix’s column number (polyline’s X-axis coordinates) for each node was peak/set. Vector; Row-content.
pY	Matrix’s row number (polyline’s Y-axis coordinates) for each node was peak/set. Vector; Row-content.
PickL	[matrix’s column number; matrix’s row number] – it is contained interpolated coordinates for each trace (column). This field is interpolation result between handle picking nodes or auto-picking result between base nodes. Vector [xL(1...n); yL(1...n)]; RowM-content.
Vbelow	Velocity below horizon (m/s).
Output fields for horizon was converted to depth:	
fName	horizon's source file name;
Digit	scalar, one digit "length" or "step" for Data matrix
GpsE	horizon’s points X or Easting coordinates.
GpsN	horizon’s points Y or Northing coordinates.

When Dataset is used, the pick results can be saved in Head(n).PL(1..m) field. The follow picking numbers are usually used:

PL(1) – up-limiter for original bottom track auto-picking;

PL(2) – down-limiter for original bottom track auto-picking;

PL(3) – original bottom auto-picked track (contained in Head(n).UnassignedInt2);

PL(4) – processed bottom track (contained in Head(n).UnassignedInt1);

PL(4).Vbelow – under-bottom “conventional” seismic waves velocity for marine survey (contained in Head(n).SubWeatheringVelocity);

PL(5..m) – other horizons are used for drawing, mapping and bottom-to-depth conversion.

2. Seg-y files read and write

Seg-y files read and write functions are based on the paper “SEG Y rev 1 Data Exchange format. SEG Technical Standards Committee. Release 1.0, May 2002”.

2.1 Read Sgy-file

function [SgyHead,Head,Data]=gSgyRead(fName,Endian,DataSampleFormat)

Read Sgy variables [SgyHead,Head,Data] from file.

Parameters:

fName – the target file name;

Endian – forced Endian; 'b' - big-endian; 'l' - Little-endian; '' - Endian auto detection (used DataSampleFormat bytes);

DataSampleFormat – forced Sample Format (1, 2, 3, 4, 5 or 8); empty matrix, if no forced (used DataSampleFormat from file);

SgyHead – Header structure, included Textual File Header, Binary File Header, Extended Textual File Header;

Head – Header structure, included Trace Headers;

Data – matrix with Traces Data.

Function Example: [SgyHead,Head,Data]=gSgyRead('c:\temp\1.sgy','',[]);

2.2 Write Sgy-file

function gSgyWrite(SgyHead,Head,Data,fNameNew)

Write Sgy variables [SgyHead,Head,Data] to file.

Parameters:

SgyHead – Header structure, included Textual File Header, Binary File Header, Extended Textual File Header;

Head – Header structure, included Trace Headers;

Data – matrix with Traces Data.

fNameNew – string, the target file for writing.

Function Example: gSgyWrite(SgyHead,Head,Data,'c:\temp\1new.sgy');

Example 1

```
%Read Seg-y File, Little-endian ordering, Data Sample Format from Binary Header
>> [SgyHead,Head,Data]=gSgyRead('c:\05_Prog\2014_HTML\sample.sgy','l',[]);
%Set Big-endian ordering
>> SgyHead.Endian='b';
% Set Forced Data Sample Format (applied to data) and Data Sample Format (recording to Header)
% to 5 (IEEE floating-point)
```

SgyHead.FDataSampleFormat=5;SgyHead.DataSampleFormat=5;

% Write Data to a new file

>> gSgyWrite(SgyHead,Head,Data,'c:\05_Prog\2014_HTML\sampleZ.sgy');

% Read Sgy File

[SgyHead,Head,Data]=gSgyRead('c:\05_Prog\2014_HTML\sampleZ.sgy','',[]);

% Draw trace segment (***Figure 2.1, a***)

>> plot(Data(11600:12000,1));

Example 2

%Read Sgy File

>> [SgyHead,Head,Data]=gSgyRead('c:\05_Prog\gSpi\Sample\011_ET3300HM.sgy','',[]);

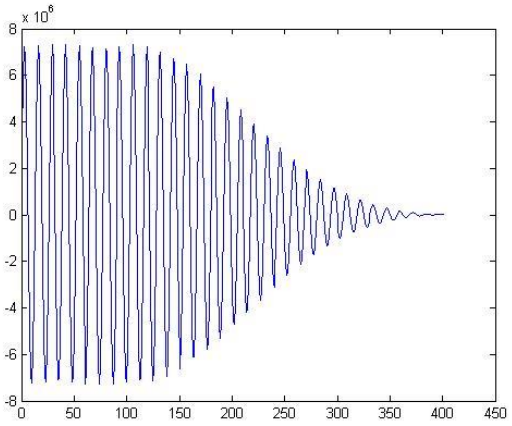
%Create Data image (***Figure 2.1, b***)

>> image(Data./10);

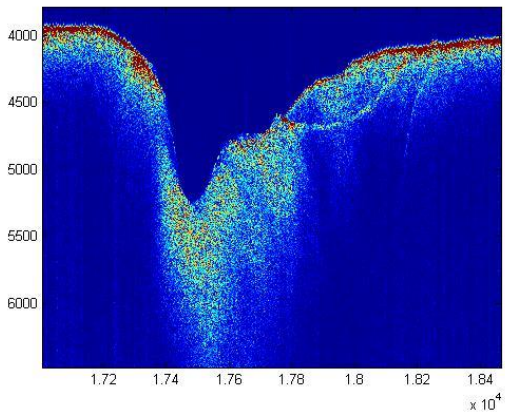
% Draw track plot (***Figure 2.1, c***)

>> plot(Head.SourceX./nnn,Head.SourceY./nnn);

(a)



(b)



(c)

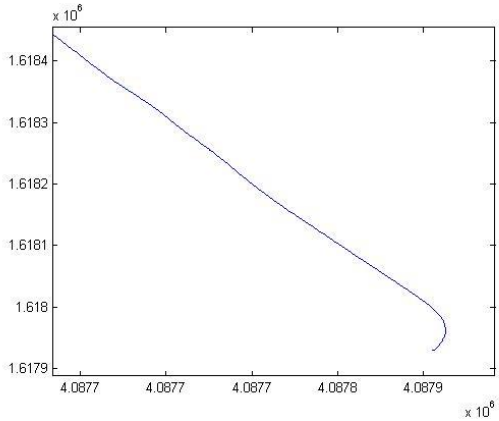


Figure 2.1 gSgy using (example)

2.3 Append Sgy-files

function [SgyHeadOut,HeadOut,DataOut]=gSgyAppendFiles(varargin)

Append Sgy files to [SgyHead,Head,Data].

Parameters:

varargin – one or several files, file's lists, folders;

[SgyHeadOut,HeadOut,DataOut] – appended files data;

The follow fields are checked as equal for both data sets: SgyHead.dt.

Function Example:

```
>> [SgyHead,Head,Data]=gSgyAppendFiles('c:\temp\1.sgy','c:\temp\2.sgy','c:\temp\3.sgy');
>> [SgyHead,Head,Data]=gSgyAppendFiles(['c:\temp\1.sgy';'c:\temp\2.sgy';'c:\temp\3.sgy']);
>> [SgyHead,Head,Data]=gSgyAppendFiles('c:\temp1\','c:\temp2\');
```

2.4 Read Dataset with a number of Sgy files

function [SgyHead,Head,Data]= gSgyDatasetImport(fName,tmpName,SgyHead,Head,Data, FieldKP,FieldX,FieldY,NavS,NavP,CoordinateUnits,SourceGroupScalar)

Add Sgy files from file-list or folder to Dataset.

Parameters:

fName – list with file's names or folder name with sgy will be loaded;

tmpName – folder name for temporary files saving; if isempty, than Data will be empty;

SgyHead – input SgyHead(1..n) structure (can be empty);

Head – input Head(1..n) structure (can be empty);

Data – input cells with Data-matrix or temporary file names;

FieldKP – the name of Kp-field; there are TraceSequenceLine,TraceSequenceFile,FieldRecord,EnergySourcePoint;

FieldX,FieldY – the names of X and Y fields; there are SourceX, SourceY, GroupX, GroupY, cdpX, cdpY;

NavS – sensor's navigation structure (see gNavCoord2Coord) for coordinates transformation;

NavP – project's navigation structure (see gNavCoord2Coord) for coordinates transformation;

CoordinateUnits – forced value for Head.CoordinateUnits field: 1-cartesian meters;2-geographic seconds;3-geographic degree;4-geographic DMS; if empty, than Head.CoordinateUnits(:)=Head.CoordinateUnits(1);

SourceGroupScalar – forced value for Head.SourceGroupScalar (-100,-10,-1,1,10,100,etc); if empty, than no forced changes; [SgyHead,Head,Data] – output variables with added data from files;

Additional fields:

SgyHead(n).fNameTmp – name of temporary file with Data matrix;

GpsDay,GpsTime,GpsE,GpsN,GpsH – fields created by function gSgyDTEN;

Function Example:

```
>> NavS=struct('TargCode',2);NavP=struct('EllipParam',[6378137 0.081819190842],'ProjParam',[0 142 0.9996 500000 0],'ProjForvFunc','gNavGeog2ProjUtm','ProjRevFunc','gNavProjUtm2Geog','TargCode', 6);
>> [SgyHead,Head,Data]=gSgyDatasetImport('c:\sgyin\','c:\sgyin\tmp\',SgyHead,Head,Data, 'FieldRecord','SourceX','SourceY',NavS,NavP,[],-100);
>> [SgyHead,Head,Data]=gSgyDatasetImport('c:\sgyin\','c:\sgyin\tmp\',[],[],[],'TraceSequenceFile', 'GroupX','GroupY',[],[],1,-100);
```

2.5 Write Dataset to a number of Sgy-files

function gSgyDatasetExport(DirName,SgyHead,Head,Data,FieldKP,FieldX,FieldY,NavS,NavP, CoordinateUnits,SourceGroupScalar)

Export Sgy-variables from Dataset to files (GpsDay,GpsTime,GpsE,GpsN,GpsH fields are used).

Parameters:

DirName – folder for export;

SgyHead – exported SgyHead(1..n) structure;

Head – exported Head(1..n) structure;

Data – exported Data-cells with matrix or temporary file names;

FieldKP – the name of Kp-field; there are MessageNum,TraceSequenceLine,TraceSequenceFile,FieldRecord,EnergySourcePoint;

FieldX,FieldY – the names of X and Y fields; there are SourceX, SourceY, GroupX, GroupY, cdpX, cdpY;

NavS – sensor's navigation structure (see gNavCoord2Coord) for coordinates transformation;

NavP – project's navigation structure (see gNavCoord2Coord) for coordinates transformation;

CoordinateUnits – scalar, units code for coordinates (see sgy description); if empty, than value form initial Sgy-file will used;

SourceGroupScalar – scalar, coded multiple for coordinates (see seg-y description); if empty, than value form initial Sgy-file will used;

Additional fields:

SgyHead(n).fName – name of original file;

SgyHead(n).fNameTmp – name of temporary file;

GpsDay,GpsTime,GpsE,GpsN,GpsH – fields created by function gSgyDTEN.

Function Example:

```
>> NavS=struct('TargCode',2);NavP=struct('EllipParam',[6378137 0.081819190842],'ProjParam',[0 142 0.9996 500000 0],'ProjForvFunc','gNavGeog2ProjUtm','ProjRevFunc','gNavProjUtm2Geog','TargCode', 6);
>> gSgyDatasetExport('c:\sgyout\',SgyHead,Head,Data,'TraceSequenceFile','SourceX','SourceY', NavS,NavP,1,-100);
```

>> gSgyDatasetExport('c:\sgyout\',SgyHead,Head,Data, 'TraceSequenceFile',
'GroupX','GroupY',[],[],[],[]);

2.6 Save Dataset's sections to temporary files

function Data=gSgyDatasetPack(SgyHead,Data)

Save Data-matrixes from Dataset to temporary files (temporary files names are put in Data cells).

Parameters:

SgyHead – input SgyHead(1..n) structure which include name of temporary file (SgyHead(n).fNameTmp);

Data – input cells with Data-matrix or temporary file names;

Data – output cells with temporary file names.

Function Example:

>> Data=gSgyDatasetPack(SgyHead,Data);

2.7 Read coordinates form Sgy-files to PL-structure

function PL=gSgyDir2PL(fName,KeyLineDraw,FieldKP,FieldX,FieldY,NavS,NavP,ChNum)

Read coordinates form Directory with *.sgy to PL-structure; coordinates transformation is applied.

Parameters:

fName – reading file name or files name or folder name with files (last name's symbol must be '\');

keyLineDraw – string key for line drawing: '-r','xb', etc;

FieldKP – field from Head structure (gFSgyRead function) copied to PL(n).GpsKP; if empty, than PL(n).GpsKP=1:length(PL(n).GpsE).

FieldX,FieldY – the names of X and Y fields; there are SourceX, SourceY, GroupX, GroupY, cdpX, cdpY.

NavS – (see gNavCoord2Coord) navigation datum for Sensor, fields: EllipParam, ProjParam, ProjForvFunc, ProjRevFunc, EllipTransParam, EllipForvTransFunc, EllipRevTransFunc, TargCode.

if ~isfield(NavS.EllipTransParam), then transformation Sensor's_Ellipsoid-to-Project's ellipsoid not calculate (fields EllipTransParam, EllipForvTransFunc, EllipRevTransFunc not used).

NavP – (see gNavCoord2Coord) navigation datum for Project, fields: EllipParam, ProjParam, ProjForvFunc, ProjRevFunc, TargCode.

NavP.TargCodes=output_datum_code (see gNavCoord2Coord); there are: 1)sensor planar; 2)sensor geographic; 3)sensor geocentric; 4)project geocentric; 5)project geographic; 6)project planar.

if isempty(NavS)&&isempty(NavP), than there is no any coordinate transformation;

ChNum – channel number; if ~isempty, then used field Head.TraceNumber to select points from multi-channel streamer;

PL – output structure: PL(n).PLName; PL(n).Type; PL(n).KeyLineDraw; PL(n).GpsE; PL(n).GpsN; PL(n).GpsKP (to KP write shot number in file)

Function Example:

```
>> NavS=struct('TargCode',2);NavP=struct('EllipParam',[6378137 0.081819190842],'ProjParam',[0 142 0.9996 500000 0],'ProjForvFunc','gNavGeog2ProjUtm','ProjRevFunc','gNavProjUtm2Geog','TargCode',6);
>> PLSgy=gSgyDirLines2PL('c:\temp\SSS\3\','-b',[],NavS,NavP,[]);
>> gMapPLDraw(100,PLSgy,1);axis equal;
```

2.8 Write [SgyHead,Head,Data] to files: *.mat and *.jp2

function gSgyJpMatWrite (SgyHead,Head,Data,fName,CompressionRatio)

Write [SgyHead,Head,Data] in two files: *.mat includes [SgyHead,Head]; *.jp2 includes Data, converted to 16-bits and compressed to Jpeg2000 picture.

Parameters:

[SgyHead,Head,Data] – sgy-file components;

fName – output files names without extension;

CompressionRatio – the ratio of the input image size to the output compressed size (for Jp2000); greater than or equal to 1.

Function Example (see ***Figure 2.2***):

```
>> gSgyJpMatWrite(SgyHead,Head,Data,'c:\temp\JpExample10',10);
```

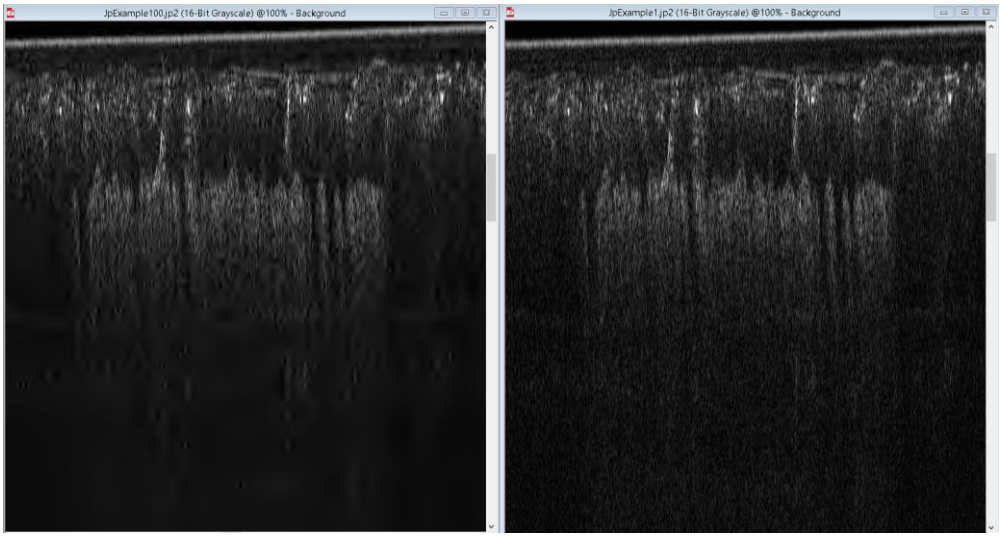


Figure 2.2 The Jpeg2000 ratio 100 (left; file size 2.2Mb) and 1 (right; file size 155.8Mb); original sgy-file size 220.2Mb, mat-file size 0.3Mb

2.9 Read [SgyHead,Head,Data] from files *.mat and *.jp2

function [SgyHead,Head,Data]=gSgyJpMatRead(fName)

Read [SgyHead,Head,Data] from two files: *.mat includes [SgyHead,Head]; *.jp2 includes Data, converted to 16-bits and compressed to Jpeg2000 picture.

Parameters:

fName – output files names without extension;

[SgyHead,Head,Data] – Sgy-file components.

Function Example:

```
>> [SgyHead,Head,Data]=gSgyJpMatRead('c:\temp\JpExample10');
```

3. Text Header

3.1 Convert ASCII to EBCDIC (Sgy text header)

function [E,L]=gSgyTextAscii2Ebcdic(A)

Convert ASCII to EBCDIC (Sgy text header).

Parameters:

A – uint8 ASCII code;

E – chars in EBCDIC;

L – position for letters recorded incorrectly.

Function Example:

```
>> [E,L]=gSgyTextAscii2Ebcdic('QWERTYUIOPASDFGHJKL')
```

3.2 Convert EBCDIC to ASCII (Sgy text header)

function [A,L]=gSgyTextEbcdic2Ascii(E)

Convert EBCDIC to ASCII (Sgy text header).

Parameters:

E – chars in EBCDIC;

A – uint8 ASCII code;

L – position for letters recorded incorrectly.

Function Example:

```
>> [A,L]=gSgyTextEbcdic2Ascii('QWERTYUIOPASDFGHJKL')
```

3.3 Replace symbols in Text Header

function THeader=gSgyTextCorrect(THeader,varargin)

Replace symbols in the Text Header.

Parameters:

THeader – input textural header contains 3200 symbols (if length~=3200 then try to use THeader as a txt-file name);

varargin – data for replace,

1) there can be struct with fields:

varargin{1}(n).Pos- text position; varargin{1}(n).Rec- text for writing; varargin{1}(n).Num- max number of symbols for text;

2) there can be vars: text_for_replace1, position_for_replace1,...text_for_replaceN, position_for_replaceN;

THeader – output textural header.

Function Example:

```
>> SgyHead.TextualFileHeader=gSgyTextCorrect('c:\temp\THead.txt',StHead);
```

```
>> SgyHead.TextualFileHeader=gSgyTextCorrect(SgyHead.TextualFileHeader,'NewText1',15,
'NewText2',55);
```

3.4 Create text-headers for Sgy-files and apply it to files in folder

script gSgyTextCorrectScript

Create text-headers for Sgy-files and apply it to files in folder; there is EdgeTech 3200SX/512I equipment.

The textural headers template is formed in accordance with the requirements of the State Bank of Digital Geological Information (Russia) for 2DHR: http://www.rfgf.ru/instrukziy/seismika.pdf (page 11-13). The 16th mandatory records marked in comments as: !!!!MANDATORY!!!!. C39 is defined as mandatory in SEGY standard.

Used Ge0MLib functions: gSgyRead, gSgyWrite, gSgyTextAscii2Ebcdic, gSgyTexturalCorrect.

StHead is the data for change, there are a number of fields:

StHead(n).Pos – text position;

StHead(n).Rec – text for writing;

StHead(n).Num – max number of symbols for text.

StHead(n).Rec need to change for own survey.

The folder rootD\conv will be clear and rewrite.

Start script with command

```
>> {'d:\3200SX\'};gSgyTextCorrectScript;
```

or the same.

There is parameter: Root Folder.

4. Sgy structure manipulations

MatLab functions set for Sgy-variables manipulations: append, delete traces, set Delay Record Time and Traces End Time, etc. Current functions are change SgyHead and/or Head structures. The functions which change the Data-variable only are realized in gData functions set.

4.1 Create DTEN-fields for Sgy-header

function Head=gSgyDTEN(Head,FieldKP,FieldX,FieldY,NavS,NavP,CoordinateUnits, SourceGroupScalar,varargin)

Create Nav's fields GpsKP,GpsDay,GpsTime,GpsE,GpsN,GpsH from YearDataRecorded,DayOfYear,HourOfDay,MinuteOfHour,SecondOfMinute,SourceGroupScalar,CoordinateUnits,Head.SourceX,Head.SourceY.

Parameters:

Head – input sgy's Header; need fields: YearDataRecorded,DayOfYear,HourOfDay, MinuteOfHour,SecondOfMinute,SourceGroupScalar,CoordinateUnits,FieldX,FieldY;

FieldX,FieldY – the names of X and Y fields; there are SourceX, SourceY, GroupX, GroupY, cdpX, cdpY;

NavS – sensor's Nav-structure (for segy-file);

NavP – project's Nav-structure;

CoordinateUnits – forced value for Head.CoordinateUnits field: 1-cartesian meters;2-geographic seconds;3-geographic degree;4-geographic DMS; if empty, than Head.CoordinateUnits(:)=Head.CoordinateUnits(1);

SourceGroupScalar – forced value for Head.SourceGroupScalar (-100,-10,-1,1,10,100,etc); if empty, than no forced changes;

varargin=H – height in sensor's Nav-structure datum.

Head – output sgy's Header; create fields: GpsDay,GpsTime,GpsE,GpsN,GpsH.

Function Example:

```
>> Head=gSgyDTEN(Head,'TraceSequenceLine','SourceX','SourceY',NavS,NavP,[],[],H);
```

4.2 Convert DTEN-fields data to Sgy-header

function [Head,varargout]=gSgyDTENinv(Head,FieldKP,FieldX,FieldY,NavS,NavP, CoordinateUnits,SourceGroupScalar)

Re-calculate Nav's fields YearDataRecorded,DayOfYear,HourOfDay,MinuteOfHour, SecondOfMinute,SourceGroupScalar,CoordinateUnits,Head.SourceX,Head.SourceY from GpsDay,GpsTime,GpsE,GpsN,GpsH.

Parameters:

Head – input sgy's Header; need fields: GpsDay,GpsTime,GpsE,GpsN,GpsH;

FieldKP – the name of Kp-field; there are
MessageNum,TraceSequenceLine,TraceSequenceFile,FieldRecord,EnergySourcePoint;
FieldX,FieldY – the names of X and Y fields; there are SourceX, SourceY, GroupX, GroupY, cdpX, cdpY.
NavS – sensor's Nav-structure (for sgy-file);
NavP – project's Nav-structure;
CoordinateUnits – forced value for Head.CoordinateUnits field: 1-cartesian meters;2-geographic seconds;3-geographic degree;4-geographic DMS; if empty, than Head.CoordinateUnits(:)=Head.CoordinateUnits(1);
SourceGroupScalar – forced value for Head.SourceGroupScalar (-100,-10,-1,1,10,100,etc); if empty, than no forced changes;
Head – output sgy's Header; create fields:
YearDataRecorded,DayOfYear,HourOfDay,MinuteOfHour,SecondOfMinute,SourceGroupScalar, CoordinateUnits,Head.SourceX,Head.SourceY.
varargout=H – height in sensor's Nav-structure datum.

Function Example:

```
>> Head=gSgyDTENinv(Head,'TraceSequenceFile','SourceX','SourceY',NavS,NavP,2,-100);
```

4.3 Append Sgy-variables from Dataset

function [SgyHeadOut,HeadOut,DataOut]=gSgyAppend(SgyHead,Head,Data)

Append Sgy-variables from Dataset.

Parameters:

SgyHead – input SgyHead(1..n) structure;
Head – input Head(1..n) structure;
Data – input cells with Data-matrix or temporary file names;
[SgyHeadOut,HeadOut,DataOut] – output data appended.

The follow fields are checked as equal for both data sets: SgyHead.dt.

Warning!!! SgyHeadOut.fNameTmp==SgyHead(1).fNameTmp

Function Example:

```
>> [SgyHead(10),Head(10),Data{10}]=gSgyDatasetAppend(SgyHead(1:4),Head(1:4),Data(1:4));
```

4.4 Delete traces from Sgy variables

function [Head,Data]=gSgyDeleteTraces(Head,Data,mask)

Delete traces from Sgy variables.

Parameters:

[Head,Data] – Sgy input variables;
mask – mask for traces delete (logical or trace_numbers);

[Head,Data] – Sgy output variables;

Function Example:

```
>> [SgyHead,Head,Data]=gSgyRead(['c:\temp\1.sgy'],'',[]);
>> [Head1,Data1]=gSgyDeleteTraces(Head,Data,1:100);
```

4.5 Change Sgy Traces Length

function [SgyHead,Head,Data1]=gSgySetEndRec(SgyHead,Head,Data,EndT)

Change Sgy Traces Length.

Parameters:

[SgyHead,Head,Data] – Sgy input variables;

EndT – new trace's EndTime in milliseconds scalar (apply to all traces) or row (apply to each own trace) or SamplesNumber (scalar only);;

[SgyHead,Head,Data1] – Sgy output variables;

Sgy Head fields changed: Head.DelayRecordingTime; Head.ns; SgyHead.FixedLengthTraceFlag.

if TraceLength less than one, than trace_data_samples number (Head.ns(n)) set to 1; Data sample set to 0.

Function Example:

```
>> [SgyHead,Head,Data]=gSgyRead(['c:\temp\1.sgy'],'',[]);
>> [SgyHead1,Head1,Data1]=gSgySetEndRec(SgyHead,Head,Data,20,1);
```

4.6 Change Sgy's DelayRecordingTime

function [SgyHead,Head,Data1]=gSgySetDelayRecTime(SgyHead,Head,Data,DeT)

Change Sgy DelayRecordingTime for each trace; add or delete trace's parts for Data.

Parameters:

[SgyHead,Head,Data] – Sgy input variables;

DeT – scalar (apply to all traces) or row (apply to each own trace) with new DelayRecordingTime in milliseconds;

[SgyHead,Head,Data1] – Sgy output variables;

Segy Head field changed: Head.DelayRecordingTime; Head.ns; Head.UnassignedInt1; Head.UnassignedInt2; SgyHead.FixedLengthTraceFlag.

if DelayRecordingTime bigger than trace length, than trace_data_samples number (Head.ns(n)) set to 1; Data sample set to 0.

Function Example:

```
>> [SgyHead,Head,Data]=gSgyRead(['c:\temp\1.sgy'],'',[]);
>> [SgyHead1,Head1,Data1]=gSgySetDelayRecTime(SgyHead,Head,Data,0);
```

4.7 Shift Sgy Data matrix from polyline1 to polyline2 using DelayRecTime

function [Head,Data]=gSgyDataToPL(Head,Data,frL,toL)

Shifted Sgy Data matrix from polyline1 to polyline2; apply shift mod(1ms) to DelayRecTime

Parameters:

Head – Sgy structure;

Data – input matrix with traces; Data(trace_length,trace_num);

frL – from polyline: 1)polyline struct; 2)two rows polyline [trace_number; current_trace's_point_number]; 3)scalar; 4)one rows polyline current_trace's_point_number for all traces;

toL – to polyline: 1)polyline struct; 2)two rows polyline [trace_number; current_trace's_point_number]; 3)scalar; 4)one rows polyline current_trace's_point_number for all traces;

The traces will be shift from frL_current_trace's_point_number to toL_current_trace's_point_number; frL or toL can be scalar, with trace's_point_number for all traces.

Segy Head field changed: Head.DelayRecordingTime.

Function Example:

```
>> [SgyHead,Head,Data]=gSgyRead(['c:\t\1.sgy'],'',[]);[Head1,Data1]=gSgyDataToPL(Head,Data,1,84);
```

4.8 Change sample interval

function [SgyHead,Head,Data1]=gSgyReSample(SgyHead,Head,Data,siNew)

Interpolate Sgy-section form current SampleInterval to new SampleInterval (data repeat or delete).

Parameters:

[SgyHead,Head,Data] – Sgy input variables;

siNew – new SampleInterval in microseconds for all traces;

[SgyHead,Head,Data1] – Sgy output variables.

Function Example:

```
>> [SgyHead,Head,Data]=gSgyRead(['c:\temp\1.sgy'],'',[]);
>> [SgyHead1,Head1,Data1]=gSgyReSample(SgyHead,Head,Data,6);
```

4.9 Sgy unification

function [SgyHead,Head,Data]=gSgyUnificate(SgyHead,Head,Data,k)

Unification for Sgy: Head.SampleNumber=const (SgyHead.FixedLengthTraceFlag=1); Head.DelayRecordingTime=const; check day changes at 00:00.

Parameters:

[SgyHead,Head,Data] – Sgy input variables;

[SgyHead,Head,Data1] – Sgy output variables;

Function Example:

```
>> [SgyHead1,Head1,Data1]=gSgyUnificate(SgyHead,Head,Data);
```

5. Time-to-depth conversion

5.1 SBP-horizon create

function Hrz=gSgyHorizCreate(PLdat,sizeDat,V,PLName,KeyLineDraw)

Create SBP-horizon for time-to-depth conversion, using 1-single value, 2-vector, 3-Horizon structure (see gData).

Parameters:

sizeDat – size of Data matrix;

V – velocity below horizon in m/s (will write to Hrz.Vbelow);

PLdat – horizons values, in trace number (horizontal) and digit number (vertical); it can be defined: single value; vector for all traces; two-rows vector with trace number and horizon's depth in digits; Horizon structure;

Horizon structure contained fields: Cur.PLName; Cur.KeyLineDraw; Cur.pX; Cur.pY; Cur.PickL

Cur.PLName – polyline name;

Cur.KeyLineDraw – string key for line drawing: '-r','xb', etc;

Cur.pX – polyline point's horizontal axis coordinates;

Cur.pY – polyline point's vertical axis coordinates;

Cur.PickL=[xL yL] – interpolated picked points coordinates for horizontal and vertical axis for each Image pixels;

PLName – horizon's name (will write to Hrz.PLName);

KeyLineDraw – string key for Horizon (polyline) drawing in MatLab's figure (for example: '-r','xb'); will write to Hrz.KeyLineDraw;

Hrz – SBP-horizon structure for time-to-depth conversion, contained fields:

Hrz.PLName – horizon's name;

Hrz.KeyLineDraw – string key for line drawing: '-r','xb', etc;

Hrz.pX, Hrz.pY – base-points for picking (applied for compatibility with picking functions);

Hrz.PickL=[xL yL] – two-rows vector with trace number and horizon's depth in digits (for each Image pixel); if horizon is not exist, than yL(n1..n2)= =nan;

Hrz.Vbelow – velocity below horizon in m/s;

Function Example:

```
>> Hrz(1)=gSgyHorizCreate(1,size(Data),1500,[],'first','.-b');
>> Hrz(2)=gSgyHorizCreate(Head.UnassignedInt1,size(Data),2000,[],'bottom','.-b');
```

5.2 Velocity matrix create

function VelMat=gSgyHoriz2VelMatrix(Hrz,sizeDat,fl)

Create Velocity matrix for time-to-depth conversion, using horizons structure.

Parameters:

Hrz(1..m) – horizon for time-to-depth conversion, contained fields:

Hrz(n).PLName – horizon's name;

Hrz(n).KeyLineDraw – string key for line drawing: '-r','xb', etc;

Hrz(n).pX, Hrz.pY – base-points for picking (applied for compatibility with picking functions);

Hrz(n).PickL=[xL yL] – two-rows vector with trace number and horizon's depth in digits (for each Image pixel); if horizon is not exist, than yL(n1..n2)= =nan;

Hrz(n).Vbelow – velocity below horizons in m/s;

Hrz(n).Digit – scalar, one digit "length" (step for Data matrix);

sizeDat – size of Data matrix;

fl – the method of Velocity matrix filling;

VelMat – Velocity Matrix same size with Data matrix filled using horizons (based on Velocities below horizons).

Function Example:

```
>> Hrz(1)=gSgyHorizCreate(1,size(Data),1500,[],'time','first','.-b');
>> Hrz(2)=gSgyHorizCreate(Head.UnassignedInt1,size(Data),2000,[],'time','bottom','.-b');
>> VelMat=gSgyHoriz2VelMatrix(Hrz,size(Data),1);
```

5.3 Convert Data-matrix from time to depth

function [SgyHeadD,HeadD,DataD,HrzD]=gSgyHorizTime2Depth(SgyHead,Head,Data,VelMat,min_dd,Hrz,FieldX,FieldY)

Convert Data-matrix from time to depth, using Velocity matrix; the horizons structure is converted too.

Parameters:

SgyHead,Head,Data – Sgy variables in time; if SgyHead is empty, than convert SBP-horizon only (no conversion to depth for seismic section);

VelMat – Velocity Matrix same size with Data matrix filled using horizons (based on Velocities below horizons).

Hrz(1..m) – SBP-horizons; can be empty (no conversion to depth for horizons); there are follow fields:

Hrz(n).PLName – horizon's name and identifier for mapping;

Hrz(n).KeyLineDraw – string key for line drawing: '-r','xb', etc;

Hrz(n).pX, Hrz(n).pY – base-points for picking (applied for compatibility with picking functions);

Hrz(n).PickL=[xL yL] – two-rows vector with trace number and horizon's depth in digits (for each Image pixel); if horizon is not exist, than yL(n1..n2)==nan;

Hrz(n).Vbelow – velocity below horizon in m/s;

min_dd – size for depth-step, integer number in millimeters; if empty, than will used a minimal step for converted Data;

SgyHeadD,HeadD,DataD – sgy structure converted to depth;

HrzD – horizons converted to depth; includes additional fields:

HrzD(n).fName – horizon's source file name;

HrzD(n).Digit – scalar, one digit "length" for Data matrix;

HrzD(n).GpsE – horizon’s points X or Easting coordinates;

HrzD(n).GpsN – horizon’s points Y or Northing coordinates.

Function Example:

```
>> Hrz(1)=gSgyHorizCreate(1,size(Data),1500,[],'first','.-b');
>> Hrz(2)=gSgyHorizCreate(Head.UnassignedInt1,size(Data),2000,[],'bottom','.-b');
>> VelMat=gSgyHoriz2VelMatrix(Hrz,size(Data),1);
>> [SgyHeadD,HeadD,DataD,HrzD]=gSgyHorizTime2Depth(SgyHead,Head,Data,VelMat,Hrz,[]);
```

5.4 SBP-horizon create

script gSgyHorizTime2DepthScript

Convert folder with Sgy-files from time to depth as 2-thickness section (water and rock); used pre-picked bottom surface from Head-structure field ('UnassignedInt1' default).

Start script with command >>> {'d:\002_Sgy\',1480,1640,'UnassignedInt1',5};gSgyHorizTime2DepthScript; <<< or the same.

There are follow parameters: Root Folder, Speed of Water, Speed of Sediments, Pre-picked Bottom Head-field, DataSampleFormat.

6. Matlab Graphics

function gSgyDraw3Section(fig_num,StepXY,StepZ,KZ,Head,Data,icaxis,icolor)

Draw Sgy Data matrix as 3D image (vertical axis is the milliseconds).

Parameters:

fig_num – figure number;

StepXY – step for traces drawing;

StepZ – step for trace's discrete drawing;

KZ – ratio for Z-axis drawing;

Head – head structure;

Data – matrix for texturemap;

icaxis – min and max data for colormap (will find from Data, if isempty);

icolor – colormap.

Function Example:

```
>> gSgyDraw3Section(7,2,2,0.01,Head,Data,[0 30000],[]);
```

7. Swell picking and filter examples

in progress

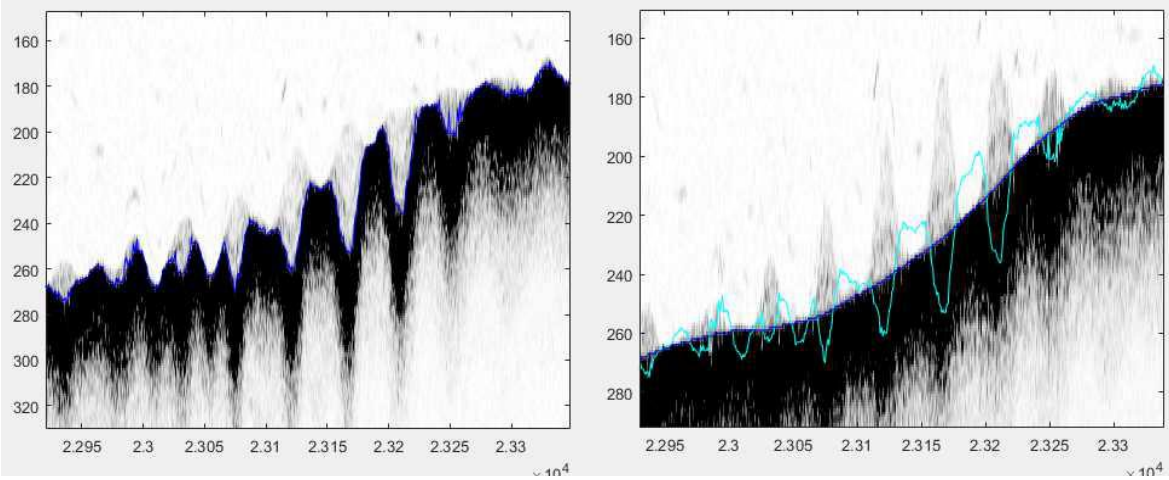


Figure 7.1 "Needle" effect (Innomar SES2000Compact; primary frequency 100kHz)

Right side – swell filter was applied

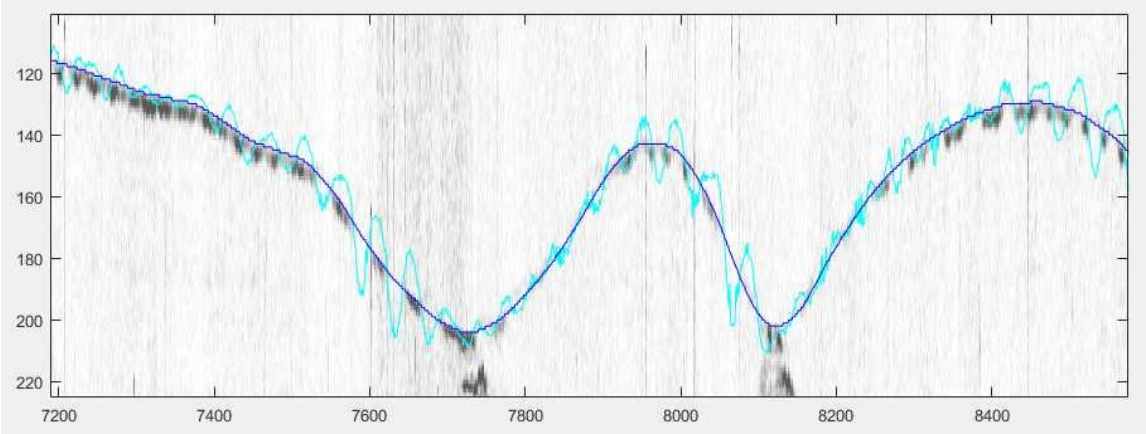


Figure 7.2 "Swell's influence (Innomar SES2000Compact; difference frequency 8kHz; swell filter was applied)

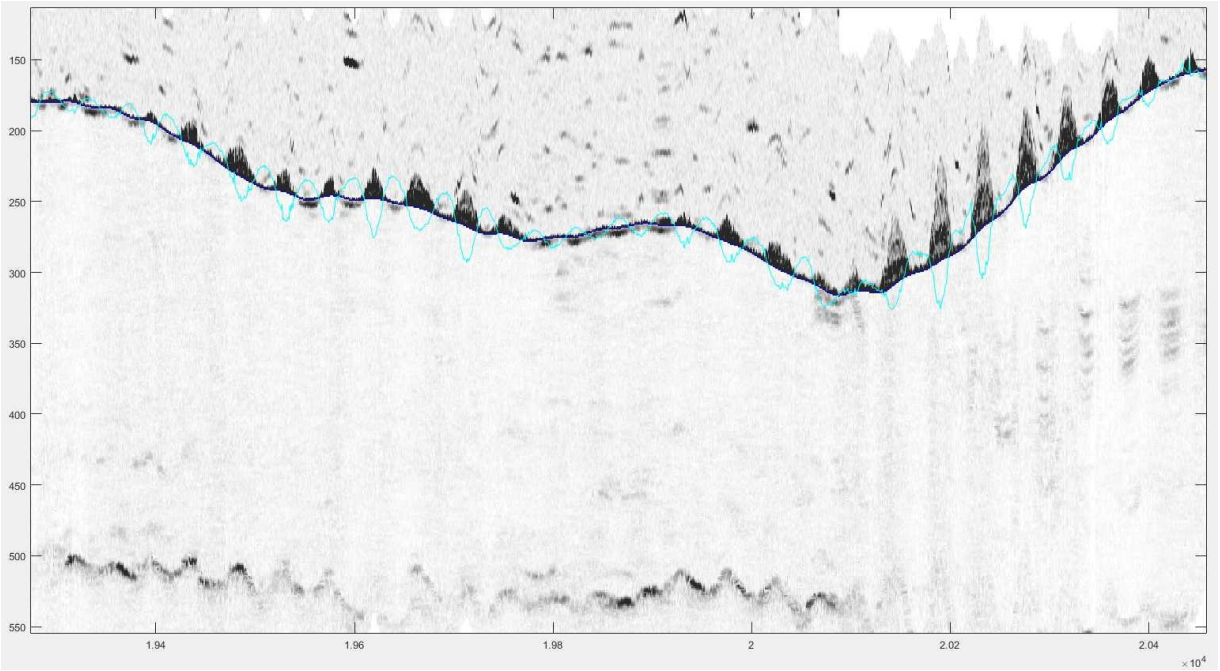


Figure 7.3 "Swell's influence (Innomar SES2000Compact; combined section –difference frequency 8kHz below bottom and primary frequency 100kHz under bottom; swell filter was applied)

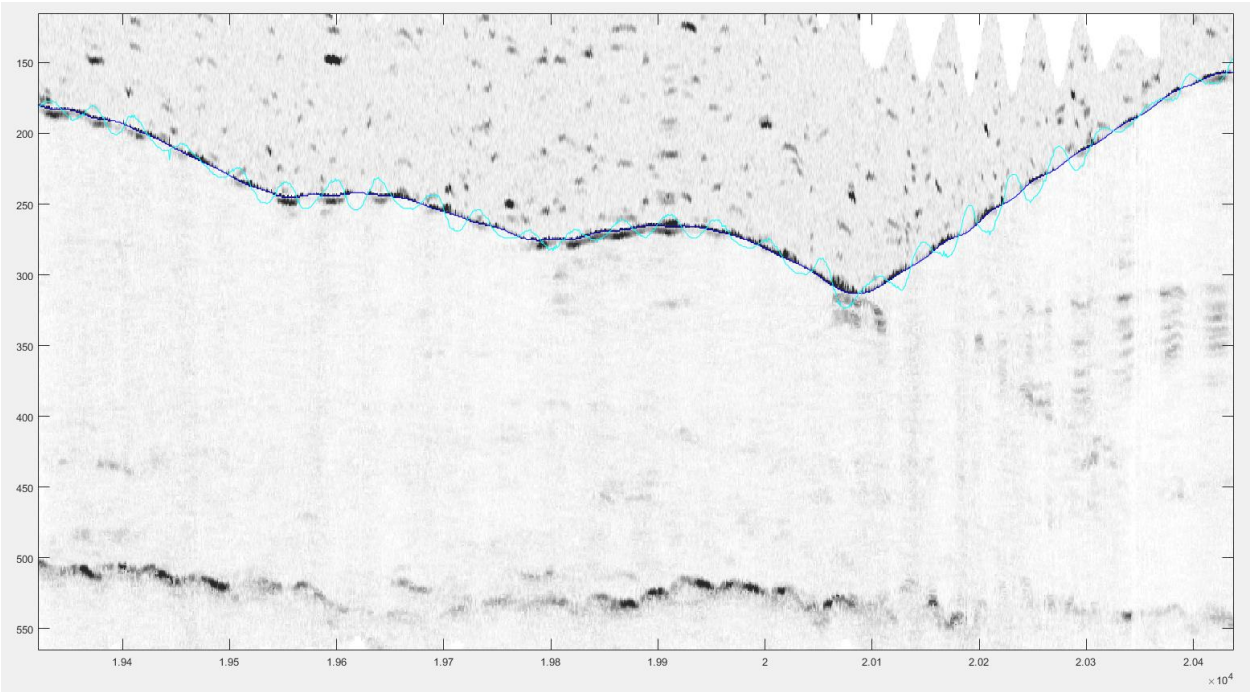


Figure 7.4 "Swell's influence (Innomar SES2000Compact; combined section –difference frequency 8kHz below bottom and primary frequency 100kHz under bottom; swell filter was applied)

8. Hyperbola calculation procedure

in progress

1) Normal point search

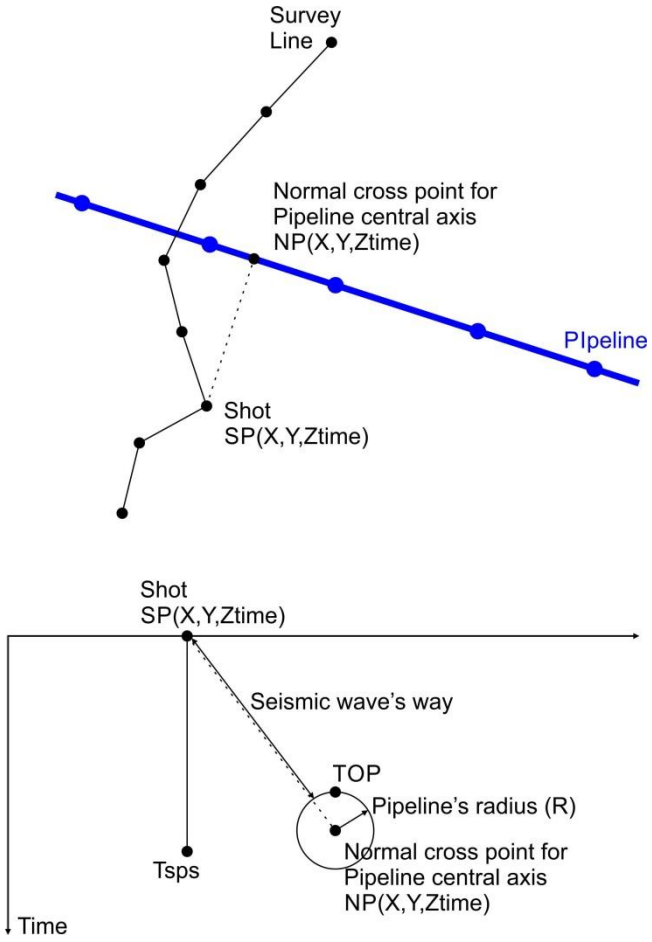


Figure 8.1 Hyperbola calculation

Using pipeline direction in survey line area we find the normal cross point (NP) coordinates for each shot point (SP) coordinates. The algorithm is on the Vector cross product basis.

2) Hyperbola's times calculation

The distance from NP to SP, in milliseconds, can calculate as

$$T_{SP} = \left(\sqrt{(X_{NP} - X_{SP})^2 + (Y_{NP} - Y_{SP})^2 + (Z_{NP} - Z_{SP})^2} - R\right) \cdot \frac{2}{V};$$

where

(X_{SP}, Y_{SP}, Z_{SP}) is the shot point coordinates in meters,

(X_{NP}, Y_{NP}, Z_{NP}) is the normal point (pipeline central axis) coordinates in meters,

R is the pipeline radius in meters,

V is the "equivalent" speed of seismic wave (it is depend from speed in water, speed in sediments, etc.; the "equivalent" speed is estimated using measurements for open pipeline and for confident-detected pipeline),

T_{SP} is the time of hyperbola for SP-point.

Apply the next substitution:

$$Z_{TOP} = Z_{NP} - R;$$

$$Z_{SP} = 0;$$
$$Z_{TOP} = \frac{T_{TOP} \cdot V}{2}.$$

The result is

$$T_{SP} = \left(\sqrt{(X_{NP} - X_{SP})^2 + (Y_{NP} - Y_{SP})^2 + \left(\frac{T_{TOP} \cdot V}{2} + R \right)^2} - R \right) \cdot \frac{2}{V};$$

where

T_{TOP} is the time for top-of-pipe.

Our "SBP-bottom" was shifted to "MBES-bottom" to Δ milliseconds by previous processing. The "MBES-bottom" is in "absolute" coordinate system (LAT or the same). We need to shift time to original survey datum point for correct hyperbola's point calculation:

$$T_{SP} = T_{SPS} - \Delta;$$
$$T_{TOP} = T_{TOPS} - \Delta.$$

The final formula is

$$T_{SPS} = \left(\sqrt{(X_{NP} - X_{SP})^2 + (Y_{NP} - Y_{SP})^2 + \left(\frac{(T_{TOPS} - \Delta) \cdot V}{2} + R \right)^2} - R \right) \cdot \frac{2}{V} + \Delta;$$

where

T_{SPS}, T_{TOPS} is the times shifted to "absolute" coordinate system (LAT or the same).

Δ is the shift value in milliseconds.

Citation