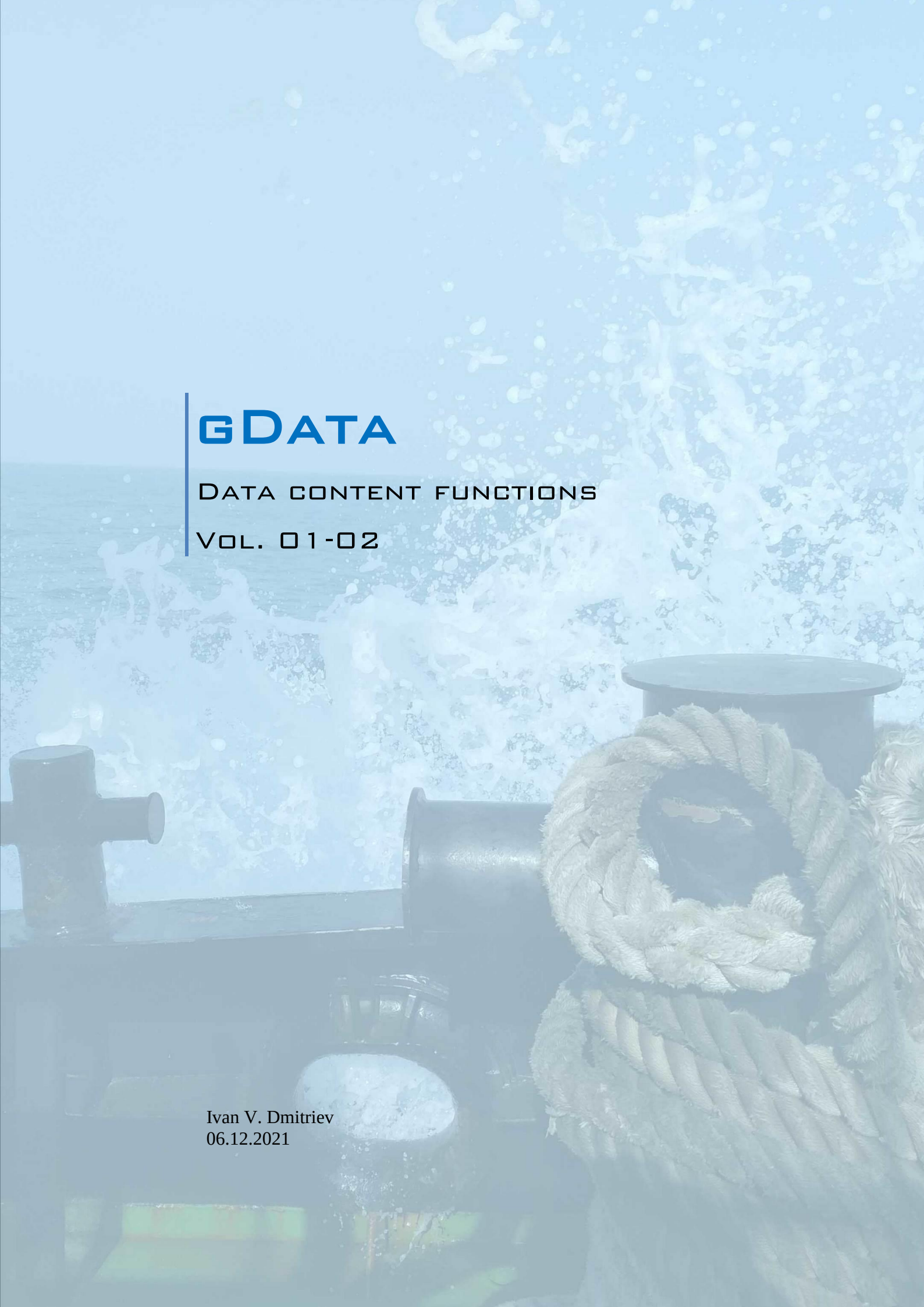

gData

Data content functions

Vol. 01-02

Ivan V. Dmitriev
06.12.2021

Contents

Tables list

Figures list

1. Introduction

MatLab functions set for manipulations with Matrix-content (**Ошибка! Источник ссылки не найден.**). The initially functions set was created for SBP section data processing: each matrix's column was considered as a seismic trace (below we will name "columns" as a "trace"). The functions were adapted to manipulations with Matrix-content data like to SBP-section, Side Scan Sonar waterfall and same.

The functions was created as universal function (not depend from structure's fields names) and any matrixes can used as input. The set's functions are shown in ***Table 1.1***.

Table 1.1 gData functions

Function name	Function description
gDataSave	Save matrix to tmp-file
gDataLoad	Load matrix from tmp-file
gDataTraceFilt	Traces/columns filtration with slice-window
gDataTraceWeight	Weighting traces/columns group (rows filtering) with slice-window
gData2DFilt	Matrix filtration with 2D-slice-window
gData2DFiltCreate	Create 2D-filter (central symmetric) for using with function gData2DFilt.
gDataNormPL	Normalize traces/columns between horizon1 and horizon2.
gDataGainPL	Traces/pings Gain.
gDataToPL	Shifted data-matrix (traces) from horizon1 to horizon2.
gDataFillPL	Fill data-matrix (traces) from horizon1 to horizon2 using number FillNum.
gDataCalcAttrib	Calculate "attributes" for data-matrix (along traces/columns).
gDataDrawSection	Draw image using Data matrix.
gDataPLPickHandle	Image/Matrix horizon handle-picking.
gDataPLPickAuto	Image/Matrix horizon auto-picking.
gDataTxtRead	Read data from text file with several Title strings and several formatted data columns.
gDataTxtWrite	Write data to text file with several Title strings and several formatted data columns.

The ***Horizon is 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.

This structure can be created by manual picking (set nodes) or auto-picking; it is like on-section seismic horizon. The structure's field's names and descriptions are shown in ***Table 1.2***. The Horizon structure used as input parameter for a number of functions.

Table 1.2 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.

2. Functions

2.1 Save Data

function gDataSave(fName,Data)

03/08/2020

Save matrix from file; file format: 1) text 'ge0mlib_Data'; 2) dimension numbers in uint64; 3) dimension values in uint64; 3) matrixes values in float64.

Parameters:

fName – name of file;

Data – saved matrix.

The function used for saving matrixes to tmp-file.

Function Example:

```
>> gDataSave('c:\temp\123.dat',Data);
```

2.2 Load Data

function Data=gDataLoad(fName)

29/10/2020

Load matrix from file; file format: 1) text 'ge0mlib_Data'; 2) dimension numbers in uint64; 3) dimension values in uint64; 3) matrixes values in float64.

Parameters:

fName – name of file;

Data – loaded matrix.

The function used for loading matrixes from tmp-file.

Function Example:

```
>> a=[1 2 3;4 5 6];
>> gDataSave('c:\temp\ggg.tmp',a); % Figure 2.1
>> d=gDataLoad('c:\temp\ggg.tmp')
```

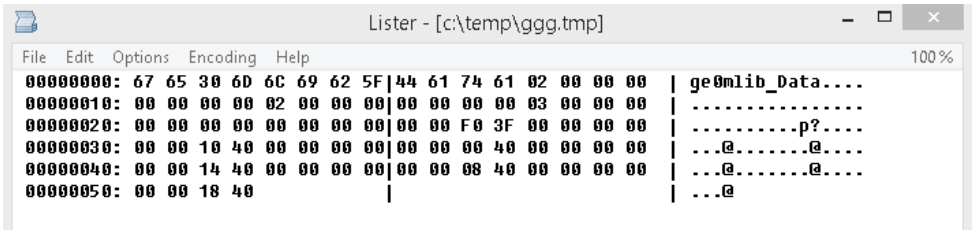

Figure 2.1 File, created by gDataSave

2.3 Traces (columns) filtering

function Data1=gDataTraceFilt(Data,wk,normFl)

18/08/2016

Traces/columns filtration with slice-window (includes filter's coefficients).

Parameters:

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

wk – filter's coefficients;

normFl – normalization flag; if normFl~=0, than wk=wk./sum(wk);

Data1 – output matrix with traces.

Example:

```
>> wk1=gausswin(N,Alpha);wk2=chebwin(L,r);wk3=blackman(N,SF);wk4=blackmanharris(N,SF);
>> wk1=gausswin(12,3);Data1=gDataTraceFilt(Data,wk1,normFl);
```

2.4 Weighting a number of traces/pings (rows filtering)

function Data1=gDataTraceWeight(Data,wk,normFl)

18/08/2016

Weighting traces/columns group (rows filtering) with slice-window (includes filter's coefficients).

Parameters:

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

wk – average filter coefficients;

normFl – normalization flag; if normFl~=0, than wk=wk./sum(wk);

Data1 – output matrix with traces.

Example:

```
>> wk1=gausswin(N,Alpha);wk2=chebwin(L,r);wk3=blackman(N,SF);wk4=blackmanharris(N,SF);
>> wk1=gausswin(12,3);Data1=gDataTraceWeight(Data,wk1,normFl);
```

2.5 2D filtering

function Data=gData2DFilt(Data,wk,normFl)

27/05/2018

Matrix filtration with 2D-slice-window (includes filter's coefficients).

Parameters:

Data – input matrix;

wk – 2D filter coefficients;

normFl – normalization flag; if normFl~=0, than wk=wk./sum(wk);

Data – output filtered matrix.

Example:

```
>> X=zeros(10);X(2,2)=1;h=zeros(7);h(4,4)=1;Data1=gData2DFilt(X,h,0);
```

function wk2=gData2DFiltCreate(wk,normFl)

03/12/2021

Create 2D-filter (central symmetric) for using with function gData2DFilt.

Parameters:

wk – uneven 1D vector with filter coefficients (basis for central symmetric 2D-filter, center is the first index);

normFl – normalization flag; if normFl~=0, than wk2=wk2./sum(wk2);

wk2 – output 2D-filter;

Example:

```
>> wk=gausswin(201,3);wk2=gData2DFiltCreate(wk(101:end),1);
>> wk=gausswin(1001,3);wk2=gData2DFiltCreate(wk(501:end),1);imagesc(wk2);
```

2.6 Normalization

function Data=gDataNormPL(Data,frL,toL,param)

17/10/2016

Normalize traces/columns between horizon1 and horizon2.

Parameters:

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)if empty, than =1; 5)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)if empty, than =1; 5)one rows polyline current_trace's_point_number for all traces;

param(1)- normalization type: 0)Data-mean(Data); 1)Data/mean(abs(Data)); 2)Data/std(Data);

param(2)- exception scalar (for example: Nan, 0, etc);

Data- output matrix with normalized traces;

Example:

```
>> Data=rand(9,5);
>> Data1=gDataNormPL(Data,[1 2 3 4 5; 4 7 1 2 2],[1 2 3 4 5; 7 7 7 8 8]);
>> Data1=gDataNormPL(Data,4,2,[0 nan]);
```

2.7 Gain

function [Data,tk]=gDataGainPL(Data,tp,k,frL)

18/10/2016

Traces/pings Gain.

Parameters:

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

tp – gain method ID: 'lg', 'exp', 'agc', 'pow';

k – gain method coefficients;

frL – gain from horizon: 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;

Data – output matrix with traces.

tk – gain coefficients.

Methods:

'nn' method: Data=Data*weight;weight=At+B; k=[A B].

'lg' method: Data=Data*weight;weight=At+20Blg(t)+C; k=[A B C dt].

'exp' method: Data=Data*weight;weight=At*exp(Bt)+C; k=[A B C dt]. If k(1)==0, than used out=exp(Bt)+C.

'agc' method: Data=Data/weight;weight=sum(abs(k*win))/nwin. k=[k1...kn] weight coefficients. Cur not used.

'agc_pow' method: Data=Data/weight;weight=sqrt(sum((k*win)^2))/nwin. k=[k1...kn] weight coefficients. Cur not used.

Example:

```
>> wk2=chebwin(L,r);wk3=blackman(N,SFLAG);wk4=blackmanharris(N,SFLAG);
>> wk1=gausswin(200,3); [Data1,~]=gDataGainPL(Data,'agc',wk1,[]);
>> [Data1,~]=gDataGainPL(Data,'exp',[0 0.001 0 1],30);
>> [Data1,tp]=gDataGainPL(Data,'lg',[0 0.2 10 1],30);
```

2.8 Shift to horizon

function varargout=gDataToPL(Data,frL,toL)

Shifted data-matrix (traces) from horizon1 to horizon2 (like to "rotation" in circle, where trace's start and end are connected).

Parameters:

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.

varargout{1}=Data – output matrix with shifred (rotated) traces;

varargout{2}=dL – shift value fromLine-toLine.

Example:

```
>> Data=rand(9,5);
>> Data1=gDataToPline(Data,[1 2 3 4 5; 4 7 0 2 2],[1 2 3 4 5; 7 7 7 8 8]);
>> Data1=gDataToPline(Data,4,2);
```

2.9 Fill between horizons

function Data=gDataFillPL(Data,frL,toL,FillNum)

Fill data-matrix (traces) from horizon1 to horizon2 using number FillNum.

Parameters:

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

frL – from horizon: 1)polyline struct; 2)two rows polyline [trace_number; current_trace's_point_number]; 3)scalar; 4)if empty, than =1; 5)one rows polyline current_trace's_point_number for all traces;

toL – to horizon: 1)polyline struct; 2)two rows polyline [trace_number; current_trace's_point_number]; 3)scalar; 4)if empty, than =1; 5)one rows polyline current_trace's_point_number for all traces;

FillNum – number for filling;

Data – output matrix.

Example:

```
>> Data=rand(9,5);Data1=gDataFillPL(Data,[1 2 3 4 5; 4 7 1 2 2],[1 2 3 4 5; 7 7 7 8 8],99);
```

2.10 Calculate Attributes

function Data=gDataCalcAttrib(Data,param)

Calculate "attributes" for data-matrix (along traces/columns).

Parameters:

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

param – attributes parameters (the elements indexes are used as scale):

instantaneous amplitude- param{1}=1; instantaneous phase- param{1}=2;

instantaneous phase cosine- param{1}=3; instantaneous frequency- param{1}=1;

envelope- param{1}=5, param{2..4}- use "help envelope".

Data – output matrix with normalized traces;

Example:

```
>> Data1=gDataCalcAttrib(Data,{3});Data1=gDataAttrib(Data,{1});
```

2.11 Draw image with Data

function gDataDrawSection(fig_num,ax,ay,Data,icaxis,icolor)

Draw image using Data matrix.

Parameters:

fig_num – figure number;

ax,ay – multiple for horizontal and vertical scale;

Data – matrix for bitmap (image);

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

icolor – colormap.

Example:

```
>> gDataDrawSection(7,1,Head.dt(1)*1e-3,Data34,[-32767 32767],[]);
```

2.12 Handle picking (create Horizon)

function outCur=gDataPLPickHandle(inCur,PLName,KeyLineDraw,extrapCur)

Handle Image/Matrix sub-horizontal horizon picking.

Parameters:

inCur – PickHandleImg input structure;

PLName – horizon name;

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

extrapCur – extrapolation flag to curve borders (create two [pX;pY] points for pX=1 and pX=end);

outCur – PickHandleImg output structure:

P.PLName; P.Type; P.KeyLineDraw; P.pX; P.pY; P.PickL

outCur.PLName – polyline name;

outCur.Type='Horizon';

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

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

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

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

Mouse&Keyboard:

LMK – set point;

RMK – delete point;

MMK – create and redraw "lines";

Space – pause mode;

q – picking end.

Example (***Figure 2.2***):

```
>> [SgyHead,Head,Data]=gSgyRead('c:\temp\2.sgy','',[]);imagesc(Data);
>> P=gDataPLPickHandle([],'123','r',1);P1=gDataPLPickHandle(P,[],[],1);
```

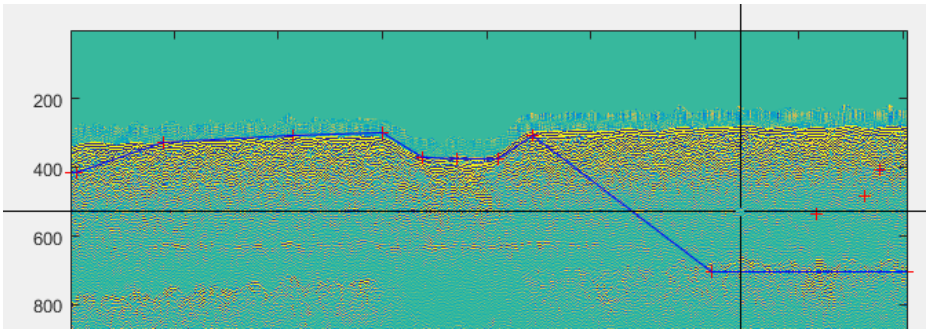

Figure 2.2 gDataPLPickHandle picking process

2.13 Auto picking (create Horizon)

function outCur=gDataPLPickAuto(Data,Pt,param,upBorder,dnBorder,PLName,KeyLineDraw)

Image/Matrix horizon auto-picking.

Parameters:

Data – 2D matrix with data for autopicking;

Pt – input polyline coordinates for autopicking: 1)polyline struct; 2) 2 rows [X;Y];

param – autopick parameters: 1)up border for "search window"; 2)down border for "search window"; 3)autopick condition 1-max, 2-min, 3-bigger than A, 4-smaller than A; 4)A for 3-4 conditions;

upBorder – upper autopicking border polyline coordinates; there are: 1)polyline struct; 2)one number; 3)row Y; 4)if empty, than =1; 5)one rows polyline current_trace's_point_number for all traces;

dnBorder – down autopicking border polyline coordinates; there are: 1)polyline struct; 2)one number; 3)row Y; 4)if empty, than =1; 5)one rows polyline current_trace's_point_number for all traces;

PLName – horizon name;

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

outCur – PickAutoImg output structure: P.PLName; P.Type; P.KeyLineDraw; P.pX; P.pY; P.PickL

outCur.PLName – polyline name;

outCur.Type='Horizon';

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

outCur.pX – autopick input polyline horizontal axis coordinates;

outCur.pY – autopick input polyline vertical axis coordinates;

outCur.PickL=[xL yL] – autopick coordinates for horizontal and vertical axis for each Image pixels.

Example (bottom picking *Figure 2.3*):

```
>> [SgyHead,Head,Data]=gSgyRead('c:\temp\2.sgy','',[]);
>> imagesc(Data,[-1000 1000]);colormap('gray');outCur=gDataPLPickHandle([],'123','r',0);
>> outCurA=gDataPLPickAuto(Data,outCur,[3 3 2],[],[],[],[]);hold on;plot(outCurA.PickL(2,:),'.b');
```

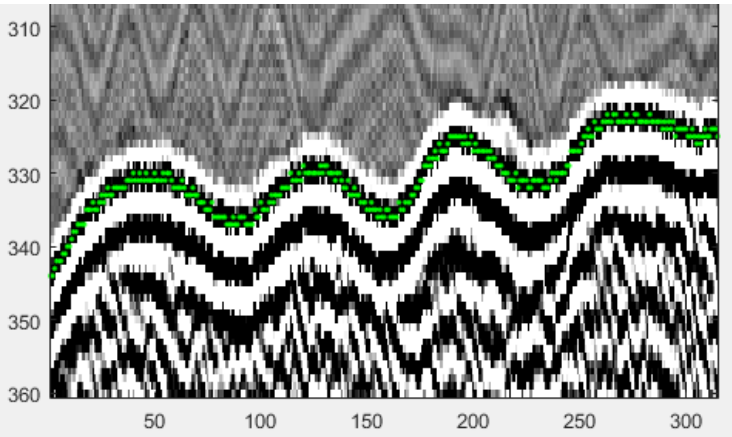

Figure 2.3 gDataPLPickAuto example

2.14 Read data from text file

function [Title,Data]=gDataTxtRead(fName,nTitle,nColumn,tForm,tDelim,tEOL)

Read data from text file with several Title strings and several formatted data columns.

Parameters:

fName – reading file name;

nTitle – number title strings (can be zero);

nColumn – number data columns;

tForm – data format (commonly '%f'); warning: not use '%s', only '%9c' or the same;

tDelim – data text Delimiters (for example: ' ' or ',' or '\t');

tEOL – data text End of Line (commonly '\r\n' or '\r' or '\n');

Title – output cell array with Title strings;

Data – output data matrix.

Example:

```
>> [Title,Data]=gDataTxtRead('c:\05_Prog\gSpi\Sample\PTS\123.txt','%f',9,3,'\t','\r\n');
>> [Title,Data]=gDataTxtRead('c:\05_Prog\gSpi\Sample\PTS\NEWGRID1.pts','%f',0,3,' ','\r\n');
```

2.15 Write data to text file

function gDataTxtWrite(Title,Data,fName,tForm,tDelim,tEOL)

Write data to text file with several Title strings and several formatted data columns.

Parameters:

fName – writing file name;

Title – cell array with Title strings;

Data – data matrix.

tForm – data format (commonly '%f'); warning: not use '%s', only '%9c' or the same;

tDelim – data text Delimiters (for example: ' ' or ',' or '\t');

tEOL – data text End of Line (commonly '\r\n' or '\r' or '\n');

Example:

```
>> gDataTxtWrite('c:\05_Prog\gSpi\Sample\PTS\123z.txt',Title,Data,'%d','\t','\r\n');
>> gDataTxtWrite('c:\05_Prog\gSpi\Sample\PTS\NEWGRID1z.pts',Title,Data,'%1.3f',' ','\r\n');
```

Citation