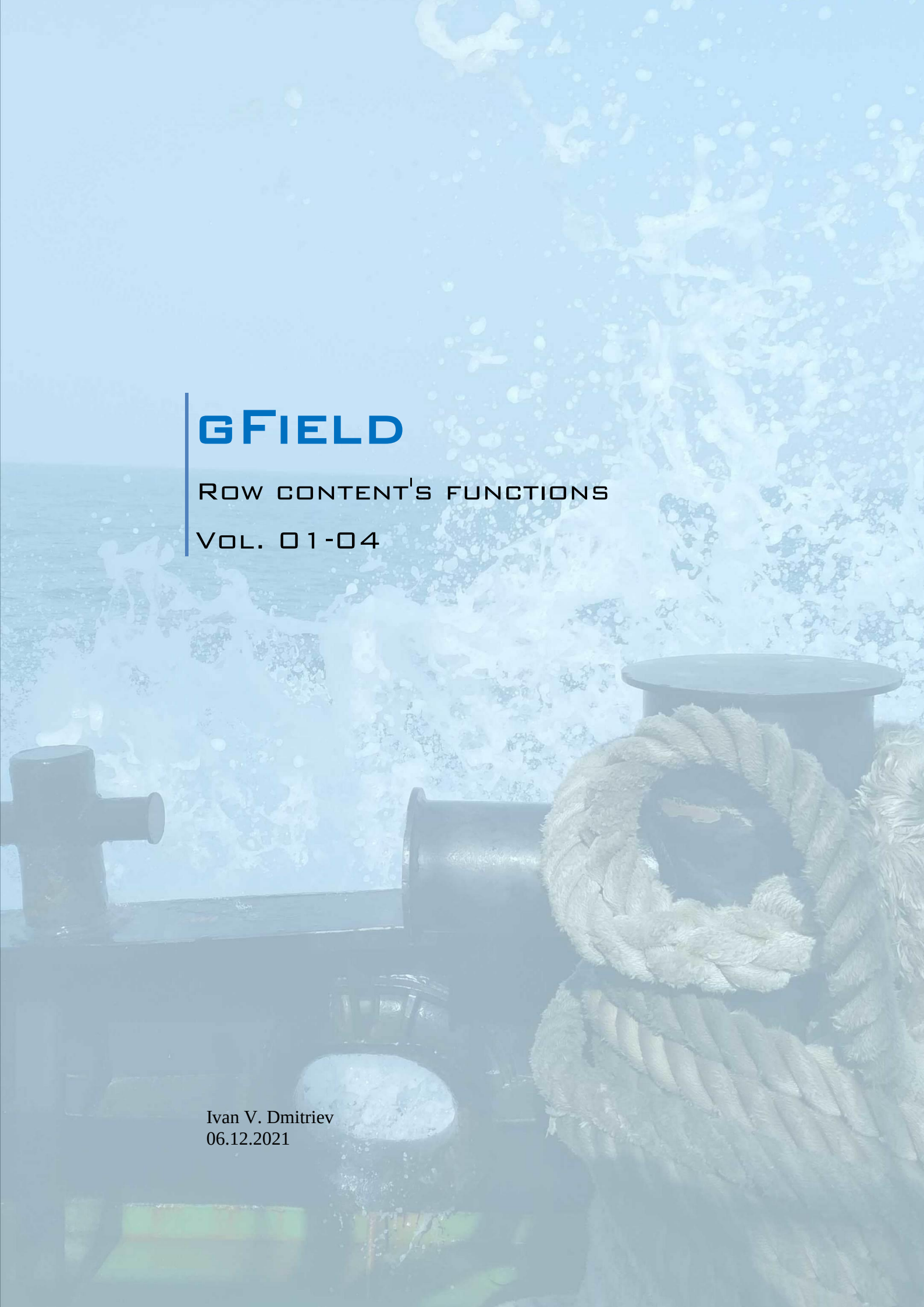

gField

Row content's functions

Vol. 01-04

Ivan V. Dmitriev
06.12.2021

Contents

Tables list

1. Introduction

MatLab functions set for general manipulations with Row-content and RowM-content (**Ошибка! Источник ссылки не найден.**): get and set values; append row-data; combine field names, etc. The row-content key is “target fields row length”; the fields with another length identify as “information content”.

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

The function used for manipulations with fields and fields’ data; there are: get, set, cut, delete, concatenate, combine. For example, a manipulation with Header fields (sgy, jsf, xtf Header and the same).

Table 1.1 gField functions

Function name	Function description
gFieldsRowGet	Get data from structure to same structure, using mask and key
gFieldsRowSet	Set data to structure, using mask and key
gFieldsRowAppend	Append Rows contents for two structures, using key
gFieldsRowInsert	Insert data to structure's Rows in defined position
gFieldsRowInterp	Interpolate data from fields to new time (the function will be changed)
gFieldsTakeFData	Take data from field-structure using string-name with sub-fields (example: 'info.data1.vels') or create cell with name’s parts
gFieldsCombine	Combine fields from two structures
gFields2PL	Create PL-structure from some structure fields’ data, using field-names
gFieldsXYZFileRead	Read XYZ file - a special kind of text file in which records starting with the word "Line" signify the start of groups of data

2. Functions

2.1 Get data from fields

function fi=gFieldsRowGet(fi,len,mask)

Get data from structure fi, using mask; key for structure fields choice is fields row length (len).

The output structure “fi” will create; it contained fields with length “len”, and applied “mask”.

Parameters:

fi – structure with fields (one or a number of row);

len – target fields row length (key for structure fields choice);

mask – mask for data get along row.

Mathematics: fi.zzz=fi.zzz(mask), where zzz – all fields with size(2)==len.

Example:

```
>> a=gFieldsRowGet (b,30000,10:100);
```

2.2 Write data to fields

function fi=gFieldsRowSet(fi,len,mask,dat)

Set data to structure fi, using mask; key for structure fields choice is fields row length (len).

Write “dat” to fields with length “len”, using “mask” position. The function usually used to delete some row numbers for row-content.

Parameters:

fi – structure with fields (one or a number of row);

len – target fields row length (key for structure fields choice);

mask – mask for data change along row;

dat – new data for fi.zzz(mask).

If fields are a number of rows, dat will apply to each row.

Mathematics: Tfi.zzz(n,mask)=dat, where zzz – all fields with size(2)==len.

Example:

```
>> a=gFieldsRowSet(b,30000,[10 100 1000],1.25);
>> a=gFieldsRowSet(b,30000,[10 100 1000],[]);
>> a=gFieldsRowSet(b,30000,[10 100 1000],[1 2 3]);
```

2.3 Append data for fields

function fi1=gFieldsRowAppend(fi1,fi2,len)

Append Rows contents for two structures; key for structure’s field choice is field row length (len).

Append “fi2” rows to the end of “fi1” rows, for “fi1” field with length “len”.

Parameters:

fi1 – structure1 with rows-fields;

fi2 – structure2 for fields-rows-contents appending;

len – rows-fields length in fi1 will be active in appending (key for structure's fields choice).

Example:

```
>> fi3=gFieldsRowAppend(fi1,fi2,3000);
```

2.4 Insert data to fields

function fi=gFieldsRowInsert(fi,dat,st_num,len)

Insert dat to structure's Rows in position st_num; key for structure's field choice is field row length (len).

Insert data "dat" into rows structure "fi", if fields (rows) length is "len"; data insert in position "st_num".

Parameters:

fi – structure with rows-fields (one or a number of row);

dat – data row for insert;

st_num – structure's Rows position for insert;

len – rows-fields length in fi will be active in appending (key for structure's fields choice).

If fields are a number of rows, dat will apply to each row.

Example:

```
>> fi1=gFieldsRowAppend(fi,[1 2 3 4],10,3000);
```

2.5 Interpolate data to new time

function gOu=gFieldsRowInterp(xIn,gIn,xOu,meth)

Interpolate gIn-fields from xIn-time to xOu-time (used interp1); the function will be changed.

Parameters:

xIn – time [Day Time] for GIn fields;

xOu – new time [Day Time] for GIn fields, the GOu will be created;

gIn – input data structure with fields;

gOu – output data structure with interpolated fields;

meth – interpolation method: 'linear','nearest','spline','pchip','cubic'.

Fields: CompDay, CompTime, GpsDay, GpsTime, CompTimeLocShift, CompTimeShift, CompTimeDelta;

Fields: GpsLat, GpsLon, GpsHgtGeoid, GpsAltSea, GpsEllipseParam, GpsProjName, GpsProjParam, GpsN, GpsE;

Fields: GpsFixQuality, GpsSatNum, GpsHorizDilution, GpsStdA, GpsStdB, GpsStdC, GpsStdD, GpsStdE, GpsStdF, GpsStdG;

Fields: GpsDgpsUpdate, GpsDgpsId;

Fields: GpsTrueTrack, GpsMagTrack, GpsGroundSpeed, GpsHeading, GpsCalcHead, GpsCalcSpeed, GPSLever;

Fields: MetaLever, MetaAltSea, TowPointLever;

Fields: MotionLAX, MotionLAY, MotionLAZ, MotionLAH, MotionARX, MotionARY, MotionARZ, MotionRoll, MotionPitch, MotionHeave, MotionRemoteHeave, MotionF, MotionHeading, MotionHeadingF, MotionLever; Fields: CompassHead, CompassPitch, CompassRoll, Altitude, Depth, MagyAbsT, MagyPrecSignal, MagyPrecLength, MagyF, CableLen, CableLever.

Example (interpolate Gps data to Sensor data):

```
>> GpsOut=gFieldsRowInterp([GpsIn.CompDay;GpsIn.CompTime],GIn,...
>> [Sens.CompDay;Sens.CompTime],'linear');Sens=gZFieldsCombine(Sens,GpsOut);
```

2.6 Get data from field with composite name

function DataFName=gFieldsTakeFData(Prof,FName)

Take data (one or several rows) from field-structure using string-name with sub-fields, without any numbers (example: 'info.data1.vels') or create cell with name's parts.

Parameters:

Prof – structure with sub-fields;

FName – string-name with sub-fields, without first point in name (example: 'info.data1.vels');

DataFName – data extracted from sub-field

DataFName – if Prof is empty, than DataFName is cells Nm for using with getfield(Prof,Nm{:}).

Example:

```
>> a=gFieldsTakeFData(Prof{10},'Mag.DepthRaw');b=gFieldsTakeFData([],'Mag.DepthRaw');
```

2.7 Combine fields

function fi1=gFieldsCombine(fi1,fi2)

Combine fields from two structures; if field name is equal, used fi2 structure's field.

Parameters:

fi1 – structure1 with fields;

fi2 – structure2 with fields for appending.

Example:

```
>> Sens=gFieldsCombine(Sens,GpsOut); %add fields from GpsOut struct to Sens struct.
```

2.8 Create PL-structure from some structure fields' data, using field-names

function PL=gFields2PL(Prof,KeyLineDraw,FieldName,FieldKP,FieldX,FieldY,FieldH)

Read fields from cell (Prof{1..n}) or fields (Prof(1..n)) structure to PL-structure, using sub-fields names (example: 'info.data1.vels')

Parameters:

Prof – cell (Prof{1..n}) or fields (Prof(1..n)) structure for sub-fields extraction;

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

FieldName – sub-field will get to 'PLName'; if empty, than used Prof-data number;

FieldKP – sub-field will get to 'GpsKP'; if empty, than used numbers 1..n;

FieldX – sub-field will get to 'GpsE';

FieldY – sub-field will get to 'GpsN';

FieldH – sub-field will get to 'GpsZ'; if empty, than not create field 'GpsZ';

PL – output structure: PL(n).PLName; PL(n).Type; PL(n).KeyLineDraw; PL(n).GpsE; PL(n).GpsN; PL(n).GpsKP.

Example:

```
>> PL=gFields2PL(Prof,'.-b','PrName','','Mag.GpsEL','Mag.GpsNL','');
```

2.9 Read XYZ file

function PR=gFieldsXYZFileRead(fName,stDelim,fiNames,datFormat)

Read XYZ file - a special kind of text file in which records starting with the word "Line" signify the start of groups of data. These groups typically are survey lines, but can be used to represent other groupings such as drill holes. The groups read into the cell vector.

Parameters:

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

stDelim – strings delimiter (usually char([13 10]));

fiNames – cell array with field names strings for columns;

datFormat – string with data format for columns reading;

PR – output cell vector; PR{nn}.LName contain string located after word "Line"; fields names defined in fiNames; fields data format defined in datFormat.

Example:

```
%PR=gFieldsXYZFileRead('e:\Dagi15.xyz',char([13
10]),{'YearUTC','MonthUTC','DayUTC','HourUTC','MinuteUTC','SecondUTC','E','N','ES','NS','EL','NL',
'CC','CCS','Depth','DepthS','Altitude','AltitudeS','MagSignal','MagAbsTRaw','MagAbsTS'},
'%f/%f/%f,%f:%f:%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f');
```

XYZ file example:

```
/ YearUTC/MonthUTC/DayUTC, HourUTC:MinuteUTC:SecondUTC, E, N, ES, NS, EL, NL, CC, CCS, Depth, DepthS,
Altitude, AltitudeS, MagSignal, MagAbsTRaw, MagAbsTS
Line 0101-D_L_AN_01
2019/06/02, 14:32:45.075, 681560.744, 5816158.483, 681560.925, 5816158.300, 681555.926, 5816168.085, 11.00, 11.00,
32.60, 32.61, 4.16, 4.16, 1824, 54709.978, 54709.978
2019/06/02, 14:32:45.199, 681560.845, 5816158.293, 681561.027, 5816158.100, 681556.028, 5816167.885, 11.00, 11.00,
32.64, 32.61, 4.13, 4.13, 1829, 54709.928, 54709.801
Line 0101-D_L_AN_02
2019/06/02, 14:32:48.011, 681563.668, 5816153.168, 681563.324, 5816153.563, 681558.339, 5816163.356, 11.00, 11.00,
32.67, 32.67, 3.75, 3.76, 1787, 54708.480, 54708.507
2019/06/02, 14:32:48.075, 681563.714, 5816153.104, 681563.375, 5816153.460, 681558.391, 5816163.254, 11.00, 11.00,
32.67, 32.67, 3.78, 3.76, 1782, 54708.703, 54708.669
```

Citation