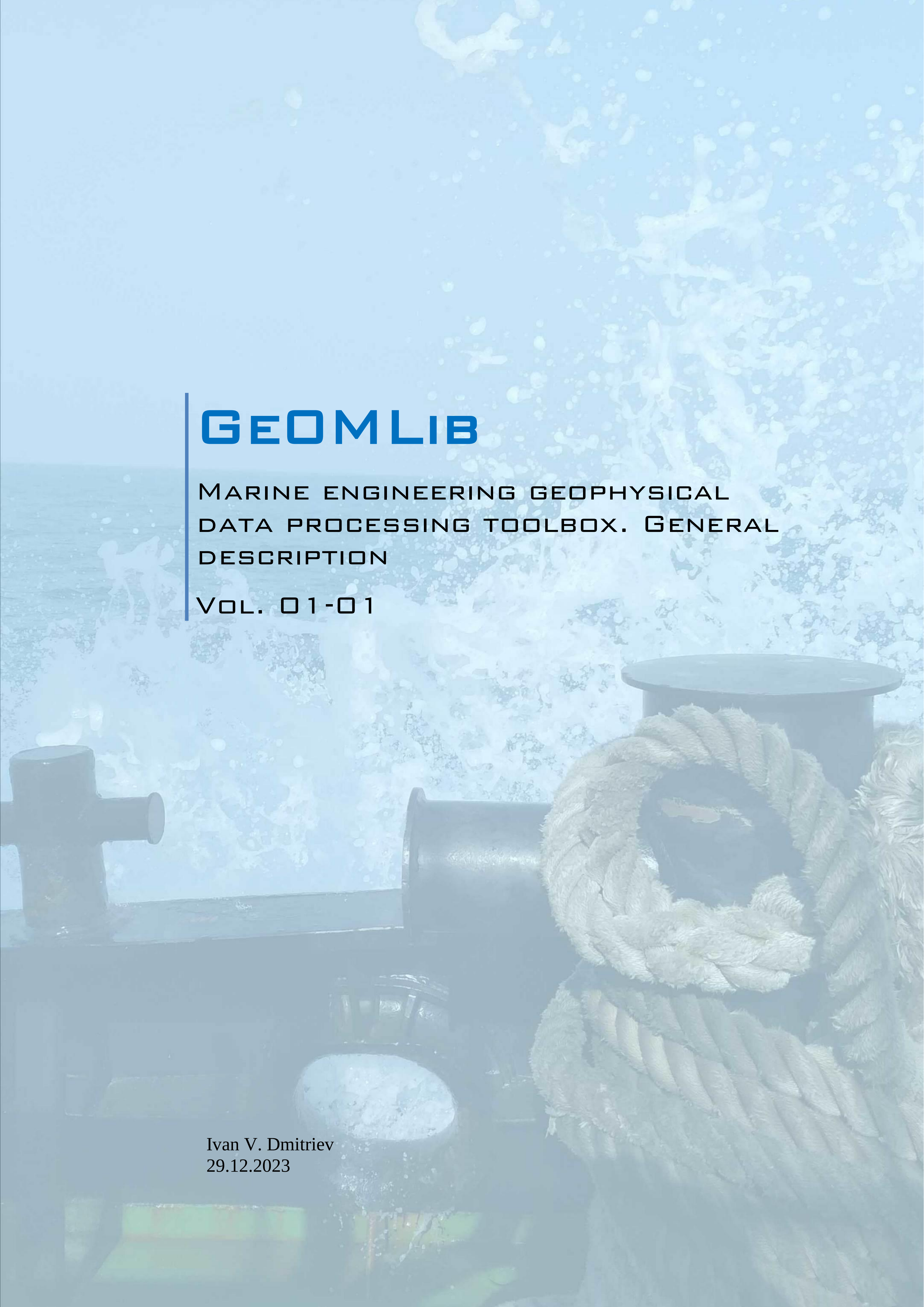


GeOMLib

Marine engineering geophysical data processing toolbox. General description

Vol. 01-01

Ivan V. Dmitriev
29.12.2023

Contents

1. **Introduction** **4**

2. **Geophysical data "components"** **7**

3. **Geophysical data acquisition and processing stages** **8**

4. **Data Abstraction Levels** **11**

5. **Data "Content types"** **12**

6. **Content structuring** **14**
   - 6.1 Content structuring review 14
   - 6.2 Structure with strictly-predefined field names, postfixes and prefixes 15
   - 6.3 Structure-vector 17
   - 6.4 General variables with fixed names 17
     - 6.4.1 SU (setup information) 18
     - 6.4.2 ML ("map layers" data) 19
     - 6.4.3 RS (raw sensors data) 20
     - 6.4.4 PR (survey lines sensors' data) 21
     - 6.4.5 GL ("geophysical processing" data) 23

7. **Processing-flow script** **24**
   - 7.1 Script's commands 24
   - 7.2 MatLab session example with Script's commands 26
   - 7.3 Survey Line processing log 28
   - 7.4 Script's command functions 28

Citation **29**

Tables list

Table 1.1 Ge0MLib's functions sets (year 2021)..4
Table 6.1 "Navigation" fields' names (example)..15

Figures list

Figure 3.1 Data acquisition and processing stage's chart..8
Figure 5.1 Measurements' data content types..13
Figure 6.1 Functions and scripts input/output variable types..14
Figure 6.2 SU-variable fields example (seismic streamer geometry) 19
Figure 6.3 ML{1} fields example (Line Planning coordinates)..20
Figure 6.4 PR{1} fields example (seismic streamer geometry) 22
Figure 6.5 PR{1}.LLog fields example (seismic streamer geometry) 22
Figure 7.1 PR{n}.ProcLog example (seismic streamer geometry) 28

1. Introduction

Ge0mlib toolbox includes algorithms for Marine Engineering Geophysical Survey data processing (Navigation, SBP, SSS, Magnetic survey, HR/UHR streamer geometry). Functions are developed in MatLab 2018b (mathworks.com) and tested for compatibility with GNU Octave 8.4.0 (www.octave.org). Serial data loggers are developed in Free Pascal (www.freepascal.org).

The ge0mlib presents a convenient space (predefined fields, variables structures, basic functions) for manipulating geophysical and navigation data. The specific ge0mlib features are:

-- fast scripting and compact code writing;

-- easy access to every byte of navigation and geophysical data;

-- convenient variable structures for geophysical tasks solving;

-- all sensors can be processed as a single dataset (e.g., corrections in MRU data can be easily applied to coordinates in Seg-y, Jsf, Xtf, and P1/90 files);

-- full access to function and script code, algorithms and math.

The Ge0MLib's functions sets are shown in the ***Table 1.1***.

Table 1.1 Ge0MLib's functions sets (year 2021)

Name	Code	Description
	01	**Basic conceptions, data types and functions**
	01-01	Ge0MLib basic
gData	01-02	Matrix content's functions
gFields	01-03	Row and RowM content's functions
	02	**Data formats and messages**
gJsf	02-02	Jsf files read/write/manipulations
gXtf	02-03	Xtf files read/write/manipulations
gSgy	02-04	SEG-Y files read/write/manipulations
gAcad	02-05	AutoCAD scripts creation
gP190	02-05	P1/90 files read/write/manipulations
gKml	02-07	Kml-files creation for Google Earth
	03	**Data logging**
gLog	03-01	Serial data logging and read
	04	**Navigation and Map**
gNav	04-01	Time and angle transformations, coordinates datum's and transformations, Nod's and Layback calculation
gMap	04-02	Simple geometric tasks decision (for example, find cross-point for pipeline and survey line) and track-plots drawing
gWfr	04-03	Images with coordinate world-files and XYZ-grid-files manipulations
	05	**Bathymetry**
	06	**Subbottom profiling**
gSgy	02-04	SEG-Y files read/write/manipulations
	07	**Multi-channel seismic**
gUhr	07-01	Functions for Logs reading (GeoEel, MultiTrace, GunLink2000) and streamer geometry calculation
	09	**Magnetic survey**
gMagy	09-01	G882 and SeaSpy marine magnetometer/gradiometer files reading, simple modeling and charting

Lyrical introduction

The first functions were written around 2013 to "play" with the seg-y format. At that time there already existed a SegyMat library, but in it the traces' information (coordinates, time, etc.) was located in a structure with an index for each trace. This was inconvenient, because putting coordinates over seg-y traces required either a for cycle or a conversion through an array of cells. Therefore, the read-write seg-y functions were rewritten so that each seismic trace header corresponded to a vector-row, which was placed in a field of the structure.

Six months later, it was necessary to "correct" coordinates in seg-y files and "move" seismic traces to the seafloor for a large volume of SBP data. I don't remember the details of the execution, but I do remember that it was urgent, three people were involved, and the process had been going on for several days; several programs were used, among which were "heavy" ones, including ProMax SeeSpace. A MatLab script written that evening did the job overnight. Probably it was very lucky, that script ran all 15 hours without fail and made a strong impression with its "ultra-fast processing". The impression was so strong that for some time I preferred the MatLab even over specialized SBP processing software.

Why MatLab?

If the language for the library were chosen today, it would most likely be Pyton because of its "popularity" and use in GIS systems, and it would most likely be a mistake. But Pyton was not yet widespread at that time, and Fortran was no longer widespread. The attractions of MatLab were:

(1) The language was mathematical, that is, you didn't have to look for a special library to compute the determinant of a matrix – the language syntax itself already included basic math. It was simply enough to write det(a);

(2) Any variable could be a matrix or a vector, so there was no need of loops for calculations, instead of for n=1 to 100, a(n)=a(n)+5;end, it was enough to write a=a+5;

(3) Items 1-2 resulted in very compact code. What in other languages would take 10-20 lines, in MatLab required writing a single expression. Compactness led to the fact that the code itself shows the "formalized solution steps", and the eye does not get bogged down in auxiliary constructs;

(4) The language is not compilable, but interpretable like the same Pyton or earlier Basic. That is, executable/distributable files are not conditional *.exe, but the program code itself. The command window of the language looked something like an "advanced calculator", and simple programs on it (from an engineer's point of view) are close to a "sequence of actions on a calculator", which is simply written in a separate file.

Most often, (4) is perceived as a disadvantage. However, there are areas where it is a plus. For example, "passable tasks" with simple calculations and unstable input and output data formats. Example: let an attentive on-line surveyor found 10 "flaws" in the format of logs with coordinates when performing geophysical surveying; each time he carefully corrected another flaw... after that, with a sense of accomplishment he dumped 50 files with perfectly recorded logs and another 100500 files with just logs in different formats before processing. For a compiled program, in such cases you need to reformat the input data manually (for example, in a text editor) or have a "formatting block" for the input data in the program itself (i.e. a menu with a choice of delimiters or data positions in a line, a choice of which "column" contains which type of data and other nuances). Let's not dive into the abyss of horror with "formatting blocks" or with 2 million mouse clicks in Excel, or into the slightly more complicated task of programming a "formatting block" for output data. Let's just say that the easiest way to do it for "passable task" is to correct a line like "%f:%f:%f:%f,%0.3f;%0.6f;%s/n/r" in the code itself, which is more natural for an interpreted rather than compiled language (and even easier – just write a script in 4 minutes and bring the data to a single format).

There are four types of application areas for "programs of engineer"

Each of them has its own specificity:

1) Trivial simple cases. To automate simple actions that can be performed with the help of "universal programs", but such performance would take a large amount of time. An example is given above. Another example - recalculation of seg-y coordinates on another ellipsoid; execution without programming: unloading coordinates from seg-y using SeiSee to a text file, reformatting the text file in a text editor, recalculating coordinates in a geodetic calculator and writing them to a text file, reformatting the text file in a text editor, loading coordinates into seg-y using SeiSee.
2) Non-trivial simple cases. When the data format is rare enough to be read by "universal programs", but the problem itself does not require advanced mathematics and processing methods. Example – entering a shift for time stamps in a Jsf-file (link to example).
3) Complex tasks. Require relatively complex mathematical calculations that can be handled only by specialized libraries. As a rule, "general-purpose software" do not solve such tasks... until these tasks become common (as, for example, happens with GPT chat). If you are dealing with such tasks, it means that you are not far from the leading edge of progress and there is no general-purpose software for this area yet... well, or just your task is too non-mainstream. An example of a MatLab library for-complex-calculations – Nav-lab; the library is designed to calculate the path of an AUV over inertial systems when the USBL signal is lost.
4) New knowledge. When you need to create something fundamentally new, that's what an algorithmic language is for. If you have such "new", you are somewhere at the forefront of engineering or scientific thought. An example of a library is n-vector. As a rule, such libraries are compact in code, but contain the very "new", which did not exist in the world before (or existed in a veiled form).

The ge0mlib library is oriented for application in areas 1-2. The library files are written for non-professional programmers in "procedural" style, without OOP elements and with the simplest possible data structures and syntax. Using the library will make it easier to implement code for cases 3-4, but the library does not contain any mega-algorithms or breakthrough functions in the field of data processing.

The functions are currently being adapted to GNU Octave. It is compatible with MatLab at the syntax level and does not require a license purchase. Just put-and-work... like Pyton))

2. Geophysical data “components”

The measured “geophysical field” is characterized the follow components:

-- time reference (synchronization) for physical field measurement;

-- coordinates reference for physical field measurement (coordinates/attitude for all geophysical system’s components);

-- physical field measurements’ value (includes active source/transmitter data and receiver data).

The components (in the wide sense) are characterized the **physical field distribution in time and space**. Each component is calculated/formed using a number of sensors and algorithms. There are:

1) The data from “**time sensors**”. Usually, this is navigation equipment data or computer’s clock.

2) The data from “**attitude/coordinates sensors**”. This is navigation equipment data.

3) The data from “**physical field sensors**”. This is geophysical equipment data.

So, the navigation equipment’s measurements are integral part of geophysical field data (its mean “physical field distribution in time and space”).

The time is the basis for different sensors and survey systems synchronizations for further combined processing. There are two related “**time sensors**”, were realized in ge0mlib:

-- the “global” source is GPS data flow with/without GPS PPS (the variables fields GpsTime, GpsDay and the same);

-- the “local” source is the computer clock data flow with 0.001 or 0.000001 second discrecity (the variables fields CompTime, CompDay and the same; data logging is realized using gLog tools).

On the one hand, the computer clock time is used for interpolation inside GPS’s 1-second intervals; on the other hand the GPS clock applies long-time-drift correction to computer’s clock data. The gLog software is used to link any messages from serial port to computer’s clock. The one GPS data flow can be used for several computer clocks’ time reference.

The “**attitude/coordinates sensors**” are used to solve “physical field sensors” positioning task; the typical tasks are:

-- nod’s position calculation in a vessel (used GPS, MRU/IMU, Gyrocompass, Draft-sensor, etc.);

-- towing equipment’s position calculation by Layback (used nod position, cable length, depth, etc);

-- off board equipment’s position calculation using USBL (used GPS, MRU/IMU, Gyrocompass, USBL data, etc.).

The positioning result is the sensor’s coordinates/attitude (the variables fields GpsLat, GpsLon, GpsE, GpsN, GpsH, Head, Pinch, Roll, CableLen, Depth, Altitude, WaterDepth and the same).

3. Geophysical data acquisition and processing stages

The Geophysical Survey data acquisition and processing stages, in accordance with Ge0MLib view point, is shown in ***Figure 3.1***. The stages are described below.

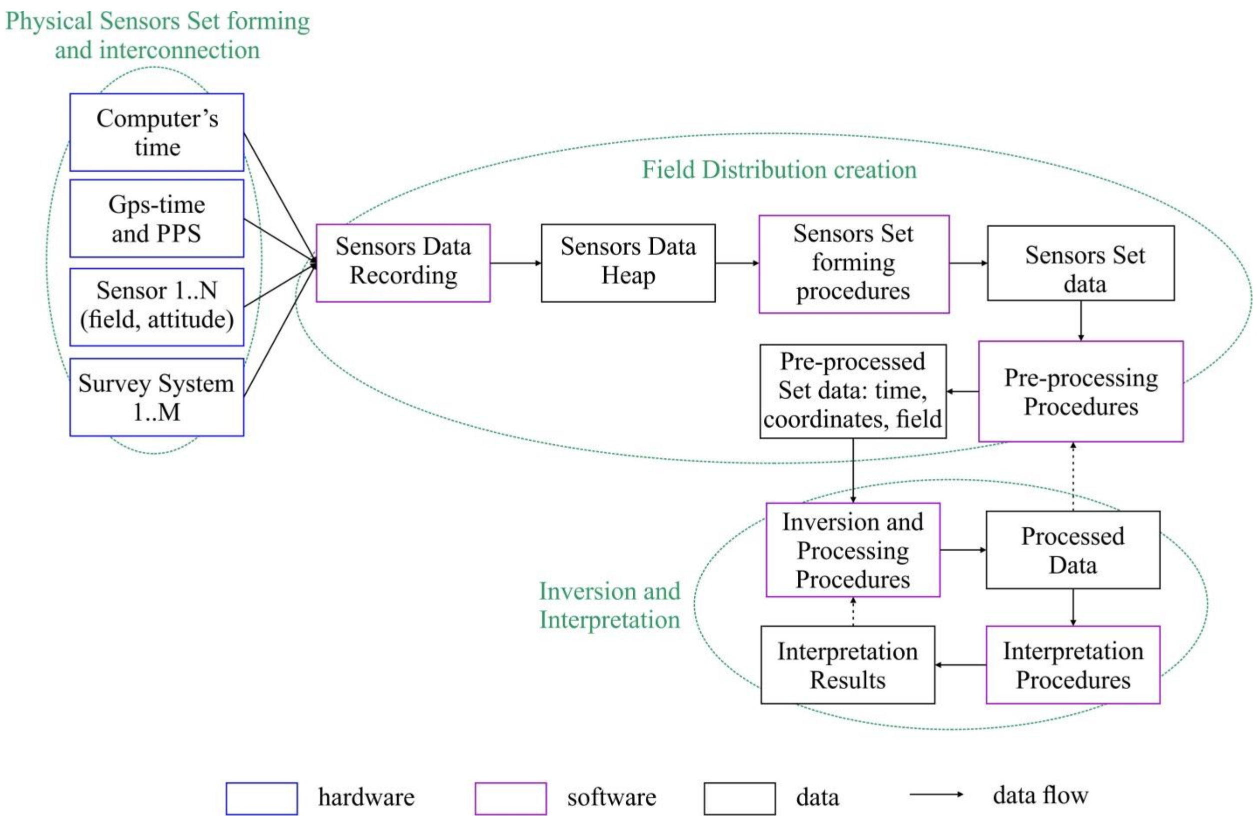

Physical Sensors Set forming and interconnection
Computer's time
Gps-time and PPS
Sensor 1..N (field, attitude)
Survey System 1..M
Field Distribution creation
Sensors Data Recording
Sensors Data Heap
Sensors Set forming procedures
Sensors Set data
Pre-processed Set data: time, coordinates, field
Pre-processing Procedures
Inversion and Interpretation
Inversion and Processing Procedures
Processed Data
Interpretation Results
Interpretation Procedures
hardware
software
data
data flow

Figure 3.1 Data acquisition and processing stage's chart

(0) The Sensors/Hardware set forming and interconnecting (there are physical fields' sensors, timing sensors, navigation sensors, etc.) in accordance with assigned task decision (magnetic UXO survey; geological SBP survey).

(1) Data acquisition/recording

The data flows from sensors come to data recording software; the data flows linkage is required. The main source for linkage is time-stamps (synchronization – sensors data link to time-of-measurements presented by "time sensors"). This stage result is **Sensors Data Heap (files)**, which contained linked records for sensors we have in digital format; there is the "on-line survey result" or "Raw Data".

(2) The Sensors Set data forming

The Sensors Set is formed from a number of sensors. There are follow procedures for each sensor:

(a) Data import and data search (into Sensors Data Heap) using defined time (the RS or PR variable created, – see below);

(b) The sensors data are bind to common time (computer clock or/and GPS-clock) and cut to “units”. The Sensors Set “unit” start-and-end time is the Survey Line record time (used Survey Log-file and PR variable created – see below).

(c) The “bad” measurements delete from Raw Data. This measurements natures is not defined by measured value physics, it is define by extraneous (third-party) forces. There are two “bad” measurements types:

-- Spikes and other “features”, which cannot defined by measured value physics;

-- Values with bad quality flag, included to data flow from sensor.

(d) Measurements smooth/filtering/de-noise procedures for normal-distributed noise decrease, sensors’ output resolution influence decrease, sensors characteristics applying (for example: noise’s frequency decrease using amplitude-frequency characteristic for seismic source and seismic receiver).

The Sensors Set forming procedures are referring to applying “specific” sensor’s characteristics; the Sensors Set “processed data” look like to “ideal sensor data” without own sensor features.

The Sensors Set “parts” examples, for different geophysical equipment:

-- sensors for SBP survey, if a transducer and receiver are “single point”;

-- sensors for SBP survey, if a transducer (source) and receiver are “different points”;

-- sensors for Seismic survey with single source and multi-channel receiver;

-- sensors for Magnetometer/gradiometer survey (several magnetic sensors can used);

-- sensors for SSS survey;

-- sensors for Single-beam survey;

-- sensors for MBES survey.

The two variables are used this stage: RS – raw sensors data were read from files; PR – raw sensors data cut to Survey Lines. Usually, we need not to process (c) and (d) procedures to total time sensor’s record and they are applied to part recorded in Survey-line time. This case PR{n} (for survey line number “n”) contained a number of sensors were cut from RS and processed using (c) and (d).

It is very important, that sensor’s “one and the same processing” result will use for follow processing steps for different equipment. So, if we find some positioning error for SSS sonograms, this correction will be applied and for magnetometer’s position was towed in piggy-back. The best way is the collect all vessel’s sensors to Survey Line’s own PR{n} cell.

(3) The Pre-processing Procedures and Pre-processed Set data forming

The Pre-processing procedures mathematical model deal with “sensors data combination” to create “geophysical field distribution in time and space”, usually this data are attached to **Survey Lines (PR-variable)**. The sensors “specific” characteristics was applied in previous stage; for current stage the mathematical model deal with abstract “measurements results”, which characterized measurement’s error. The model’s task example: SBP-nod’s position calculation in a vessel (used sensors are GPS, MRU/IMU, Gyrocompass, Draft-sensor, etc.).

There are follow steps for Pre-processed Set creation:

(a) Geophysical sensors coordinate (attitude) calculation on the basis of navigation sensors data;

(b) Sensors (navigation and geophysical) data interpolation to target-sensor time;

(c) Some calculations in accordance with pre-processing geophysical processes models (for example: Earth’s magnetic variations or deviations applying);

(d) Disambiguation for Sensors Set data (measurements or model correction);

(e) Using Sensors Set, create the Pre-processed Set (PR variable fields – see below), which include: physical field measurements, time reference for physical field measurements, space/coordinates reference for physical field measurements.

This stage used for positioning tasks decision. For PR{n} cell calculate the fields with “all-in-one” data for different types of geophysical single-equipment. Those fields includes attitude for each single-equipment (or a number of parts for single-equipment). Data-in-field usually are synchronized to single-equipment measurement time. One PR{n} cell contained own field with pre-processed data for SBP, SSS, Magy and others single-equipment types.

(4) The Processing Procedures and Processed Data forming

This procedures deal with physical field distribution in time and space (Pre-processed Set). The procedures’ mathematical model deals with physical field equations. The procedures’ tasks are inverse problem decision or the same. The Processed Data is the physical field’s sources distribution/characteristics or another inverse geophysical tasks’ decision.

The processing results are located in GL-variable.

(5) The Interpretation Procedures and Interpretation Results forming

The Interpretation Procedures is the Processed Data “interpretation” in the terms of physical characteristics and objects for geological/engineering knowledge area. The Interpretation Results are the some information about geological/engineering “features” for survey area.

The steps (3)-to-(4) and (4)-to-(5) can be recursive for results best conformity to selected mathematical models.

The Ge0MLib’s main area is the stages (1), (2) and (3). There is the border land between navigation and geophysical areas/specializations. The many types of sensors can be used; the many sensors interconnections and combining can realized; the many variants of sensors data processing and cross checking can be applied to raw data; the many barriers between navigational processing and geophysical processing software are presents.

The Ge0MLib is present “programming open space” to create data flow (for on-line data logging and post-survey processing) in opposite to “static software” with multi-purpose based on “deep nested menu and checkbox”. The finally, “programming open space” approach to a sensors’ data processing is more controlled and reliable than manual “deep checkbox”.

4. Data Abstraction Levels

On the “Geophysical data acquisition and processing stages” basis, the follow **Data Abstraction Levels** can be described:

1) The Sensors Set forming procedures deal with **“specific” sensor’s and data’s features**. All specific information is applied to data this stage.
2) The Pre-processing Procedures deal with **ideal sensors data**; ideally – without specific features and without irrelevant noise.
3) The “ideal sensors data” used for Pre-processed Set creation – the calculation of **physical field distribution** (there are physical field values, time and space coordinates).
4) The Processing Procedures deal with physical field equations, which described physical field’s distribution. The result is inverse geophysical tasks’ decision, named Processed Data. To wide extent, the Processed Data is the information about **physical field’s sources distribution and properties**. There can be: magnetic masses location for UXO; seismic reflection or refraction horizons location and velocities. There can be several levels of deep processing and detailing for Processed Data (for example, we can apply to reflection seismic different migration types, anisotropy, kinematic analysis, etc.). The processing level depends from final survey tasks and geological/engineering objects properties are researched.
5) The Interpretation Procedures deal with transfer “information” from physical field’s knowledge space (physical field’s sources distribution and properties) to geological/engineering knowledge objects space (**physical substance distribution and properties**). This transfer results named as geological/engineering Interpretation Results.

5. Data "Content types"

The data, Ge0Mlib's functions used, can be considered as a "specific content" by the structure of data. The content's types are described below, there are: Information content; Row-content; RowM-content; Matrix-content; MatrixM-content. The general features for data content types (the "containers" for different types of content) are shown in ***Figure 5.1***.

Information-content

This content contained "constant" information about sensor's characteristics, measurements mode, etc. This information is applied to data block unit: Data Heaps' sensor, all Data Set or one survey lines' data. The information-content examples: Ellipsoid characteristics, coefficients for depth sensor or altimeter.

Row-content

This content is contained measurements results along time axis (binding/linking source axis). All rows are designed as variables' fields by usability. The row-content fields for data block unit (for example, one survey lines' data) have equal length. Row-content field examples: Easting coordinate, Northing coordinate, Water temperature, Pitch angle, Total magnetic field. There are several predefined Row-content field names for general data types, which used by Ge0MLib functions.

The row-content is the equipment measurements result with "scalar output", when one measurement (along binding/linking source axis) is one number by one sensor. There are any time-series "scalar" measurements: GPS data, IMU/MRU data, magnetic sensor data, etc.

RowM-content

If we have a number of the same and synchronized sensors with row-content, it is reasonable to create matrix with a number of row. This matrix is alternate to several Row-content fields. The RowM-content example is magnetic measurements with gradiometer; there are a number (from 2 to 7) synchronized and the same magnetic sensors.

Matrix-content

This content is contained measurements results along time axis (binding/linking source axis); each measurement is time-series inserted to matrix column. The matrix-content is the equipment measurements result with "time-series-output", when one measurement (along binding/linking source axis) is the series numbers along time axis (by one receiver).

The matrix column examples are SBP or SSS traces. Other measurements (column/traces) characteristics (coordinates, time stamp, etc.) are contained in associated Row-content structure. The matrix column number and Row-content field's length is equal.

MatrixM-content

If we have a number of receivers with matrix-content, it is reasonable to create additional axis with a sensor number; we will have 3-axis dimension. A "number of receivers" example is digital-seismic streamer. There are dimension axes: 1) short time or short numbers; 2) record-time for hydrophones or recorded digits number; 3) receiver numbers.

Other measurements characteristics (coordinates, time stamp, etc.) are contained in associated Row-content structure (for example, seismic source coordinates) or RowM-content structure (for example, streamers' hydrophones coordinates).

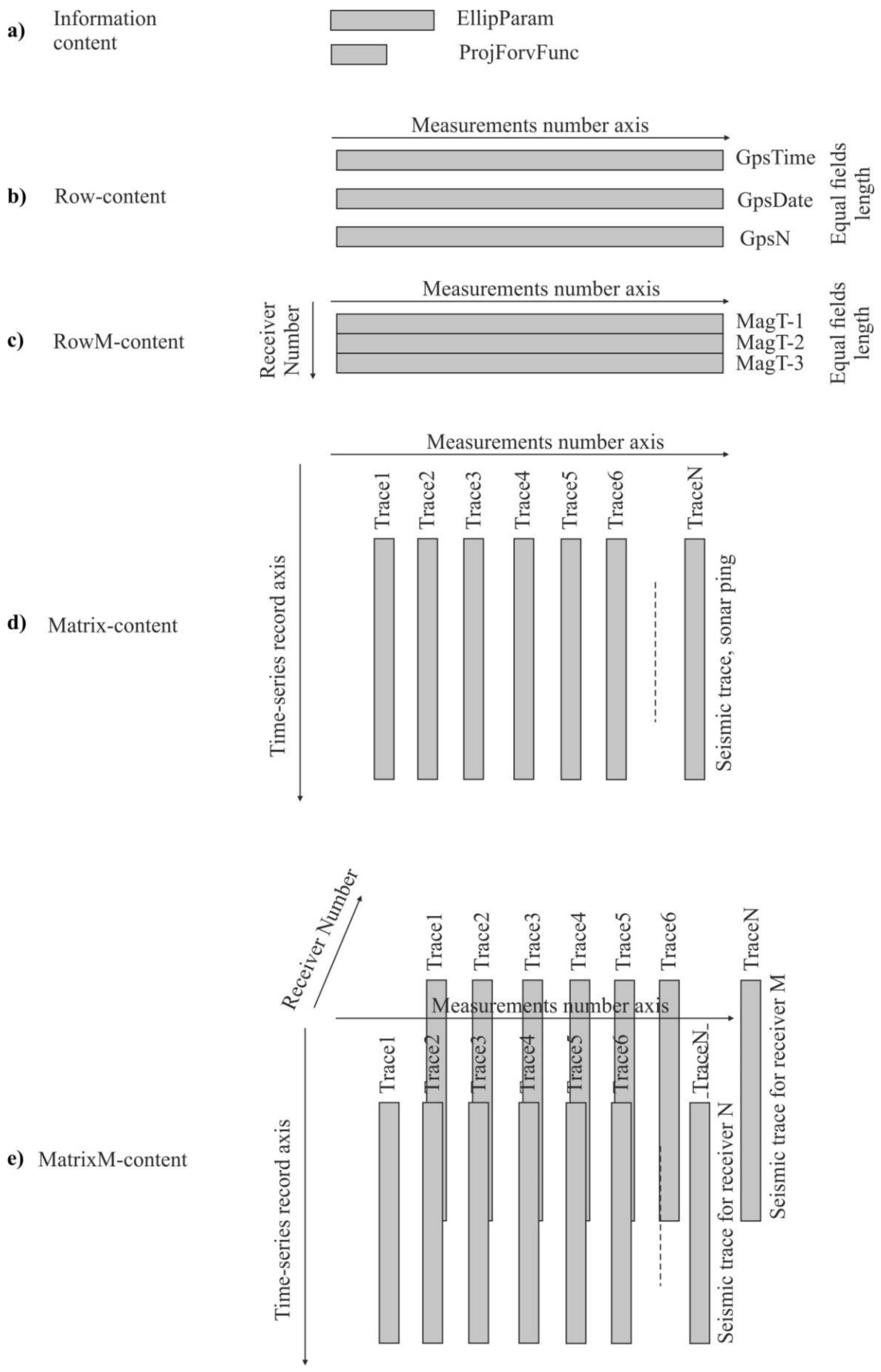


Figure 5.1 Measurements' data content types

6. Content structuring

6.1 Content structuring review

The data (a heap of numbers) are order to five “content types”; there are Information content; Row-content; RowM-content; Matrix-content; MatrixM-content (see ***Data “Content*** types). The content types are used to form compound structures; those structures include four rank levels. Accordingly, the function and scripts input/output can characterize with the four levels of formalization (***Figure 6.1***):

(1) Not specified content. The first level contained the “universal” functions with input/output parameters defined as a vectors, numbers, strings, etc. We can understand this level as functions/content without “sense load”.

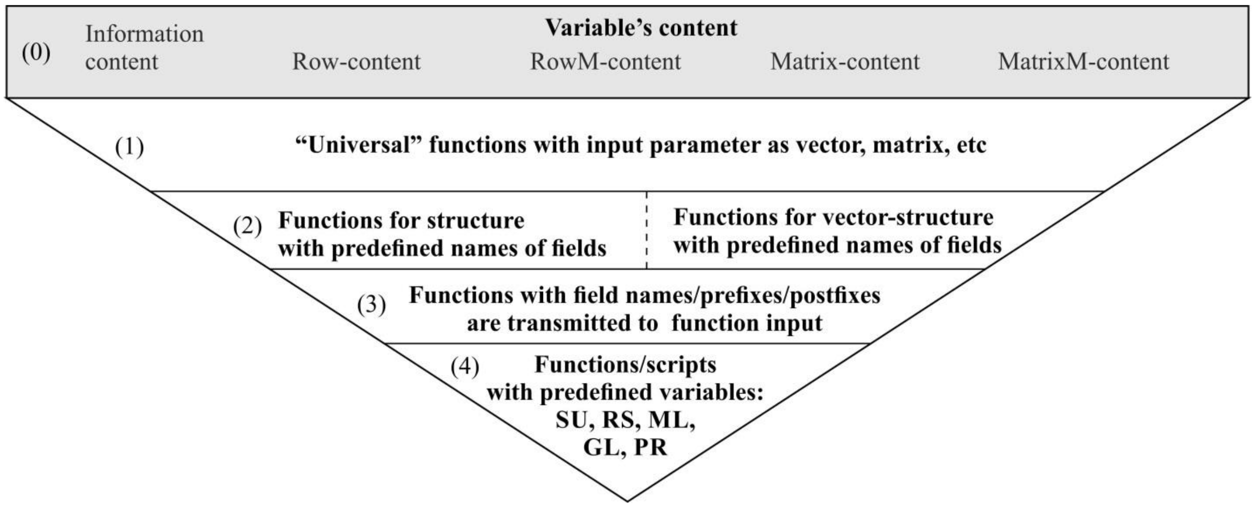

Variable’s content
(0)
Information content
Row-content
RowM-content
Matrix-content
MatrixM-content
(1)
“Universal” functions with input parameter as vector, matrix, etc
(2)
Functions for structure with predefined names of fields
Functions for vector-structure with predefined names of fields
(3)
Functions with field names/prefixes/postfixes are transmitted to function input
(4)
Functions/scripts with predefined variables: SU, RS, ML, GL, PR

Figure 6.1 Functions and scripts input/output variable types

(2.1) Strictly-predefined “sensor” (read file). The second level is a single “structure with strictly-predefined field names”. The field-names are define “sense load” of data contained (coordinates, heading, temperature, etc.). The functions have “knowledge” about field names, make some manipulations with data in fields and create new “known” fields. Each structure includes **only one level of field-names**. The one structure content can be associated with a “single sensor data” (magnetic fish is the “single sensor” which includes follow fields: depth by pressure, altitude by altimeter, magnetic field by magnetic sensor, signal strength by magnetic sensor and other fields for other sub-systems data). Usually same structure is used for sensors data-files loading.

(2.2) Vector of Strictly-predefined “sensor” (read files). A number of same “structures with predefined field names” can formed a structure-vector. Each vector’s element is contained the same field-names and data types than other. For example, a number of sgy-files (SBP survey) can be load to vector-structure. Each vector’s structure element includes only one level of field-names.

(3) Pre-processing (data interpolation to target-sensor time and “combining”). The third level is the same (2), but contained functions used field-names as additional input parameter or field-names prefixes/postfixes (it will add to strictly-predefined field-names). The special rules for prefixes/postfixes using and forming take place. This mechanism let to use a number of fields with the same “sense load”. For example: [GpsE, GpsN] is the strictly-predefined field-names for coordinates; [TpGpsE, TpGpsN] is the coordinates of tow-point; [UsblGpsE, UsblGpsN] is the coordinates of USBL beacon’s attached to tow-cable; [SssGpsE, SssGpsN] is the coordinates of SSS located in 5 meters back from USBL beacon. The processor-man can create a number of fields with the same “sense load” and use these fields into processing data flow.

(4) Survey Project pack (includes sensors were read at RS-variable and sensors Pre-processing at PR-variable). The fourth level of formalization is characterized the cell-vectors with fixed names. There are follow variables:

-- SU – the “strictly-predefined sensor” with general project’s setup information

-- ML – the cell vector with “map layer” variable,

-- RS – the cell vector with raw sensors data for all survey time (one cell is one sensor),

-- PR – the cell vector with Survey Lines (one cell is a number of sensors for one Survey Line),

-- GL – the cells vector for Processed Data (“geophysical processing data” variable).

The follow rules take place:

-- Each cell contained its own object: sensor’s measurements, survey line data, map-object, etc.; the objects are located to variables in accordance with object’s type.

-- SU is the structure with general project’s setup information, type 2.1.

-- Each ML-cell includes one object (sensor) with finally processed “extraneous data”, level (2).

-- Each RS-cell includes one sensor, level (2).

-- Each PR-cell is the “Survey Line”; it can include a number of sensors with level (3). That way, we can have two levels of field-names: the first is the “sensor’s name” (magnetic fish) and the second is the “sensor’s sub-systems name” (depth, altimeter, magnetic field, etc.).

-- Each GL-cell is a single “Processing Method data”; it includes a number of field-names levels in accordance with processing method’s notion for own survey type.

6.2 Structure with strictly-predefined field names, postfixes and prefixes

A sensors data, imported to MatLab includes the predefined field names for each type of data. The predefined coordinates and other “navigations” fields for some point are shown in ***Table 6.1***. These fields’ names can be located to RS variable’s cell when sensor’s data will read using “data-reading functions”. These fields’ names are use and modify (with postfixes and prefixes) when sensors data processed and concatenate.

Table 6.1 “Navigation” fields’ names (example)

Field name	Field Description
GpsLat	Latitude (WGS84 usually); DD.DDDDDD.
GpsLon	Longitud (WGS84 usually); DDD.DDDDDD.

Field name	Field Description
GpsAltSea	Height by Gps, GpsAltSea data
GpsHgtGeoid	Height by Gps, GpsHgtGeoid data.
GpsHead	True heading by differential_GPS/Gyrocompass/MRU.
GpsE	Rectangular projection Easting.
GpsN	Rectangular projection Northing.
GpsH	Height created when Ellipsoid-to-Ellipsoid coordinates transformation take place; field used for coordinates transformation's stability
GpsKp	Point's ID, it can be measurement number, kilometer point's number or any other.
GpsZ	Points Z-coordinate for vertical geodetic datum (pipeline or towing equipment position).
Altitude	Point's Altitude (usually by altimeter for towing equipment).
Depth	Point's Depth (usually by depth sensor for towing equipment).
Head	Heading in Rectangular projection.
WaterDepth	Water Depth for current points coordinates (can be "actual" or referenced to some system MLS, LAT, Baltic, etc).
XXXXRaw	Raw data postfix (no smoothing, de-spiking, etc), where XXXX is any field name.

To keep raw sensors data, the variable field-names can include the special **postfix "Raw"**. Those fields automatically created for original data keeping (which was loaded form sensor's files) by functions which transform raw-data. For example, the raw-coordinates from GpsE and GpsN can be kept as the GpsERaw and GpsNRaw, after that, the coordinates in GpsE and GpsN fields can be de-spiked and smoothed. If we restart "de-spiking and smoothing", the GpsERaw and GpsNRaw fields were used for as data-source.

Usually, the postfix "Raw" used for "original data" only; the sensors data was interpolated to target-sensor time and "calculated" data cannot have postfix "Raw". But in general case, any function can create own specific postfix for own "pre-processed data" to keep original data for repeated processing for better "processing parameters" search. The postfix Raw is a good decision because all raw-data need to be de-spiked and smooth and there are no previous data for raw-data.

The different data which includes fields with similar names (with the same "sense load") are assign the own "field-names prefixes" defined by user. For example, tow-point coordinates (with fields, named as GpsE, GpsN and GpsH for data was loaded) interpolated to magnetic field measurement time and copied to magnetic sensor can be assigned the prefix Tp (TpGpsE, TpGpsN and TpGpsH); for calculated towed body coordinates can be assigned prefix Tb (TbGpsE, TbGpsN and TbGpsH); for calculated magnetic sensors (located on the towed body) coordinates can be assigned prefix Ts (TsGpsE, TsGpsN and TsGpsH). So, one sensor's data fields include a number of "sense load" data: the coordinates of tow-point, towed body and sensors located on the towed body. In other hand, all field-names are contained strictly-predefined name's part.

6.3 Structure-vector

Some sensor data, loaded from a number of files to variable-structure with strictly-predefined field names, can be offer as a Structure-vector. It is mean that each file loaded to own vector-structure element.

The example for vector-structure A:

-- A(1..n).GpsLat;

-- A(1..n).GpsLon;

-- A(1..n).GpsAltSea.

The Structure-vector used for several files loading and simple output; for example, we can load a number of P1/90 files to Structure-vector and create track-plot map for survey lines.

The one more example: we can load a number of Sgy-files to Structure-vector. Sgy-file read in MatLab to three variables, usually named SgyHead (binary, textural and extended textural headers), Head (trace headers) and Data (Matrix with trace's data). The three Structure-vectors are formed as:

-- SgyHead(1..n);

-- Head(1..n);

-- Data{1..n}.

SgyHead(m) and Head(m) are the vector-structure element with predefined fields. The Data{m} is the cell-vector (not Structure-vector) with elements which contained seismic traces data or tmp-file name with seismic traces data (see gSgy manual).

If we have the uninterrupted SBP survey data (not stop survey between survey lines, not change survey parameters), we can load Sgy-files to Structure-vector and cut it to "profiles" (form the Sgy-files for target-time when vessel have a straight moving or run-in).

Any case, a Structure-vector is a good decision for raw sensors data loading-and-cut in a "Sensors Set data forming stage". A Cell-vector, which contained Survey Lines sensors data, is a good decision for "Pre-processed Set data forming" stage.

6.4 General variables with fixed names

There are follow cell-vectors and variables with fixed names:

SU – structure with general project's setup information;

ML – the cell vector with "map layers" (extraneous "static" data were final processed); examples: infrastructure objects' coordinates, coastline, wells head coordinates, pipelines or cables coordinates and other characteristics, bathymetry data for different years;

RS – the cell vector with raw sensors data for all survey time;

PR – the cell vector with sensors for Survey Lines (includes sensor's data interpolated to target-sensor time for each survey line);

GL – the cell vector with “geophysical processing data” (the data created in the PR-processing process); examples: SBP-sections picking results (seismic horizons), magnetic targets picking and position/mass estimation result.

6.4.1 SU (setup information)

The “setup information” variable includes information which concerned global project’s settings; it is mean general settings same for all Survey lines. The SU-variable used as a method for settings input too (it is simple correct the MatLab script code, than each time create menu or function for ini-file reading).

There are two ways for data to SU variable was input. The “strongly constant settings” are kept in SU variable. The settings which potentially can be changed are copy to “profile settings” in PR{n}.LLog field.

SU “strongly constant” fields’ define code example:

```
SU.RootDir='d:\2019\009_Grad'; % the name of project’s root folder;
SU.LLog=1; % RawSensor's cell number with Survey Log-file;
SU.LPlan=1; % map layer’s cell number with Line Planning;
SU.Bott=2; % map layer’s cell number with actual sea bottom;
SU.NavS=struct('EllipParam',[6378137 sqrt(2.* 1/298.257-(1/298.257).^2)],'EllipTransParam',[6.102269
        -10.161 -5.186 -0.050 0.0701 0.0583 2.0293e-
        6],'EllipForvTransFunc','gNavGeoc2Geoc1032', 'EllipRevTransFunc',
        'gNavGeoc2Geoc1032inv','TargCode',2); % Navigation Sensors coordinates datum;
SU.NavP=struct('EllipParam',[6378137 sqrt(2.* 1/298.257-(1/298.257).^2)],'ProjParam',[0 121 0.9999
        250000 0],'ProjForvFunc', 'gNavGeog2ProjUtm','ProjRevFunc','gNavProjUtm2Geog',
        'TargCode',6); % project’s coordinates datum;
SU.CompTimeLocShift=8*3600; % local time shift in seconds;
SU.MagNorm=[44000 -3.60 35.50]; % magnetic locally-normal field [T(nT) D(deg) I(deg)];
SU.SgyFormatForced=1; % forced sgy format. If empty, than no forced;
SU.SgyReelPos=1:4; % prefix char position in LineName (sequence or Reel for Sgy, contained as a
        prefix in LineName);
SU.Job=25; % Job identification number (JIN).
```

SU “potentially changed” fields’ (for seismic streamer’s geometry) define code example:

```
SU.WaterVelocity=1470; % water velocity in m/s;
SU.NameP190GunHead='P190_GunHeader.txt';
SU.NameP190Cmp1Head='P190_Cmp1Header.txt';
SU.NameSgyTxt='Sgy_TextHeader.txt';
SU.SpInterval=6.25; % shotpoint interval;
SU.GunTpNod=[]; % gun towpoint position relative vessel's CRP;
SU.GunDist=54; % length of gun towcable (from towpoint to "acoustic center");
```

SU.StTpNod=[]; % streamer's towpoint position relative vessel's CRP;

SU.StBuoyDist=1418.4; SU.StBuoyNum=numel(SU.StBuoyDist); % Streamer length from towpoint to tail buoy // number of tail for streamer;

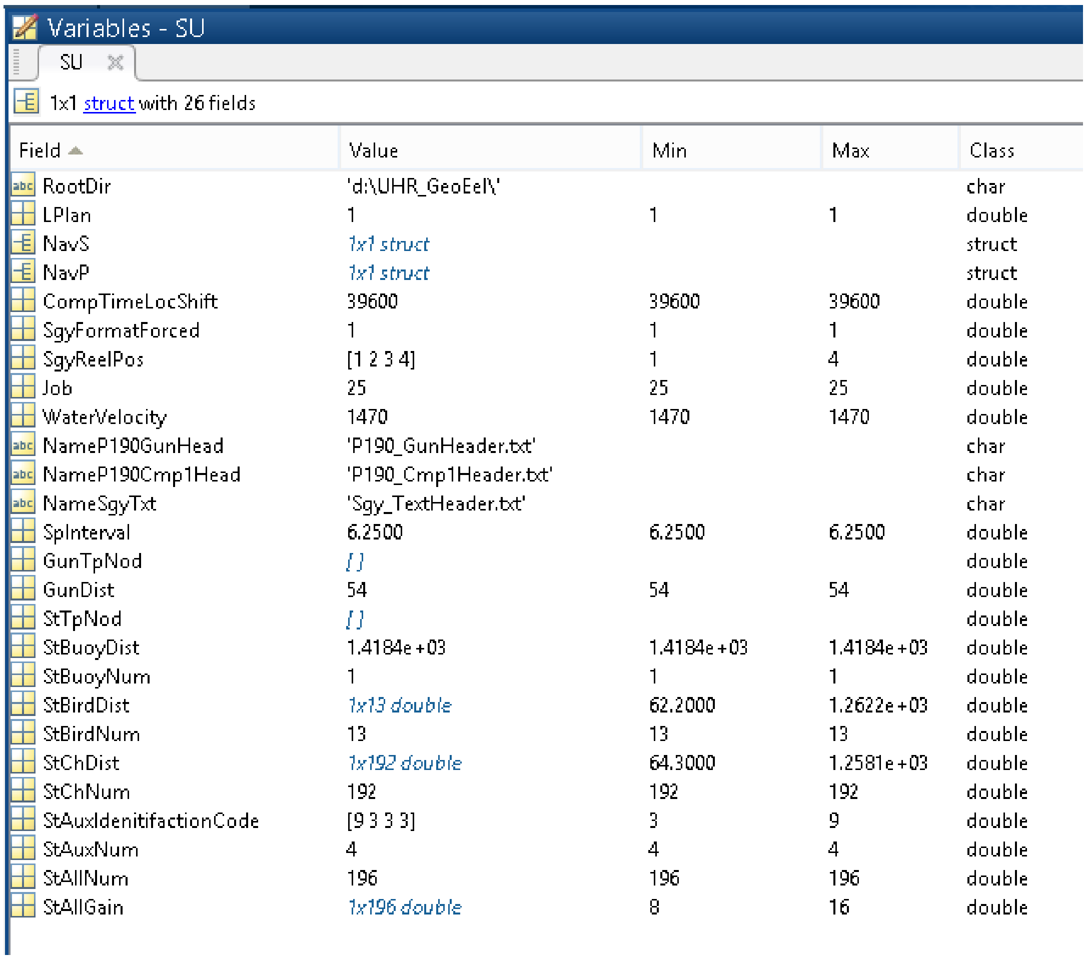
SU.StBirdDist=[62.2 162.2 262.2 362.2 462.2 562.2 662.2 762.2 862.2 962.2 1062.2 1162.2 1262.2]; SU.StBirdNum=numel(SU.StBirdDist); % distance to birds from towpoint, calculated by section length // number of birds for streamer;

SU.StChDist=(0:192-1).*6.25+64.3;SU.StChNum=numel(SU.StChDist); % distance to Channels from towpoint, calculated by section length // number of data channels for streamer;

SU.StAuxIdenitifactionCode=[9 3 3 3];SU.StAuxNum=numel(SU.StAuxIdenitifactionCode); %Sgy Bytes29-30; TraceIdenitifactionCode;

SU.StAllNum=SU.StChNum+SU.StAuxNum; % number of all channels;

SU.StAllGain=[zeros(size(SU.StChDist))+8 16 16 16 16]; % gain for data channels and aux channels.


Variables - SU

SU

1x1 struct with 26 fields

Field	Value	Min	Max	Class
RootDir	'd:\UHR_GeoEel\'			char
LPlan	1	1	1	double
NavS	1x1 struct			struct
NavP	1x1 struct			struct
CompTimeLocShift	39600	39600	39600	double
SgyFormatForced	1	1	1	double
SgyReelPos	[1 2 3 4]	1	4	double
Job	25	25	25	double
WaterVelocity	1470	1470	1470	double
NameP190GunHead	'P190_GunHeader.txt'			char
NameP190Cmp1Head	'P190_Cmp1Header.txt'			char
NameSgyTxt	'Sgy_TextHeader.txt'			char
SpInterval	6.2500	6.2500	6.2500	double
GunTpNod	[]			double
GunDist	54	54	54	double
StTpNod	[]			double
StBuoyDist	1.4184e+03	1.4184e+03	1.4184e+03	double
StBuoyNum	1	1	1	double
StBirdDist	1x13 double	62.2000	1.2622e+03	double
StBirdNum	13	13	13	double
StChDist	1x192 double	64.3000	1.2581e+03	double
StChNum	192	192	192	double
StAuxIdenitifactionCode	[9 3 3 3]	3	9	double
StAuxNum	4	4	4	double
StAllNum	196	196	196	double
StAllGain	1x196 double	8	16	double

Figure 6.2 SU-variable fields example (seismic streamer geometry)

6.4.2 ML ("map layers" data)

The ML is the cell-vector. Each cell includes one object with finally processed "extraneous data" (the data are designed as "for using", but not for "processing"; the data is not change), level (2); the examples: infrastructure objects' coordinates, coastline, wells head coordinates, pipelines or cables coordinates and other characteristics, bathymetry data for different years (each year in own cell).

ML-cells can include the follow objects kinds:

-- vector-data, (like PolyLine structure) – survey lines planning, infrastructure objects position,

-- geo-referenced images (bathymetry image, SSS mosaic, Backscatter image);

-- grids (for example – bathymetry data will used for seismic bottom leveling).

The usually data are used for objects visualization only or for export; for example, the Line Planning object (*Figure 6.3*) is draw in MatLab figures and can be exported to AutoCad's script. Another data are used for calculations; for example – pipeline position, depth and radius will used for calculation crossing with survey line or seismic hyperbola modeling for real track.

ML-fields define code example for Line Planning loading:

ML{SU.LPlan}.fRec='LinePlan20190916.txt'; % the name of file with Line Planning (located in the project's root folder);

ML{SU.LPlan}.fRecFormat='LinePlan'; % format identification code;

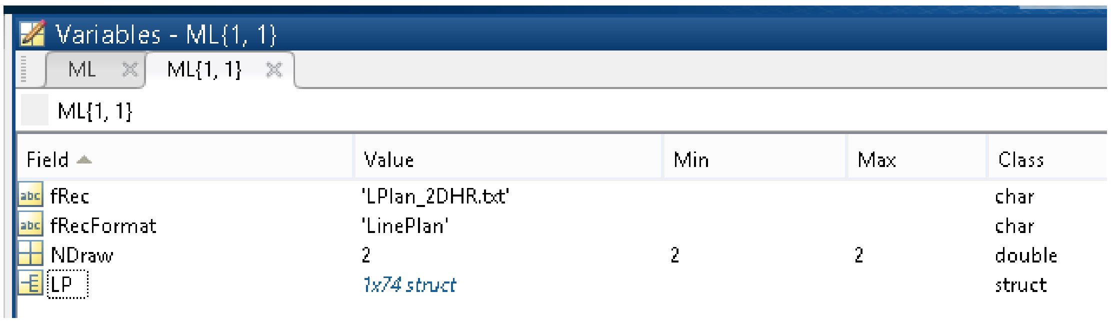
ML{SU.LPlan}.NDraw=2; % draw order (NaN if not drawing).

ML-fields define code example for Bathymetry data loading

ML{SU.Bott}.fRec='3961_EGS#2.pts'; % the name of file with Bathymetry data (located in the project's root folder);

ML{SU.Bott}.fRecFormat='Xyz2Triang'; % format identification code;

ML{SU.Bott}.NDraw=nan; % draw order (NaN if not drawing).



Field	Value	Min	Max	Class
fRec	'LPlan_2DHR.txt'			char
fRecFormat	'LinePlan'			char
NDraw	2	2	2	double
LP	1x74 struct			struct

Figure 6.3 ML{1} fields example (Line Planning coordinates)

6.4.3 RS (raw sensors data)

The cell vector with raw sensors data for all survey time (one cell is one sensor). It is includes information which necessary for sensor's files loading and the results of loading. The information is the different for different sensor's types. The fields for sensors are strictly-predefined.

RS-fields for "Magnetic survey Log-file" loading (sensor's number SU.LLog=1)

RS{SU.LLog}.fRec='Log_mag.txt'; % file with Log (located in the project's root folder);

RS{SU.LLog}.fRecFormat='gLLog'; % format identification code;

RS{SU.LLog}.fgLogDelim=[]; % un-applicable.

RS-fields for "Towed body position" loading (sensor's number 2)

RS{2}.fRec='000_TB\'; % the name of folder contained files with towed body position;

RS{2}.fRecFormat='gLogGPGGA'; % position's format identification code;

RS{2}.fgLogDelim=['~' ',']; % gLog delimiters for Computer Time label (data was recorded using gLog);

RS{2}.Period=1; % sample period (NaN if not applicable).

RS-fields for "SeaSpy2 magnetic gradiometer data" loading (sensor's number 7)

RS{7}.fRec='000_Mag3\'; % folder contained files with magnetic gradiometer data;

RS{7}.fRecFormat='gLogSeaSpyGrad'; % magnetic gradiometer identification code (examples: MagLogG88Magy, MagLogG88Grad, gLogSeaSpyMagy, gLogSeaSpyGrad);

RS{7}.fgLogDelim=['~' ',']; % gLog delimiters for Computer Time label (the data flow was recorded using gLog);

RS{7}.Period=0.5; % sample period;

RS{7}.ID=[999 999 999;999 nan 999]; % serial numbers [magnetic_sensor1_SN, altimeter1_SN, depth_sensor1_SN; ....; magnetic_sensorN_SN, altimeterN_SN, depth_sensorN_SN];

RS{7}.Coef=[1 0 1 0;1 0 1 0]; % scale and bias coefficients for Altimeter and Depth sensor [Altimeter1Scale Alttimeter1Bias Depth_sensor1Scale Depth_sensor1Bias; ....; AltimeterNScale AltimeterNBias Depth_sensorNScale Depth_sensorNBias].

6.4.4 PR (survey lines sensors' data)

The "Survey Lines variable" variable is the cell vector; each cell is include sensor's data from RS (raw sensors data) which cut for Survey Line time borders and (for next steps) can be interpolated to target-sensor time.

Each PR-cell can include a number of sensors (***Figure 6.4***). That way, we can have two levels of field-names: the first is the "sensor's name" (magnetic fish) and the second is the "sensor's sub-systems name" (depth, altimeter, magnetic field, etc.).

The PR-cell contained "LLog" data field with information about current Survey Line (***Figure 6.5***). There are:

(1) own Lines information fields:

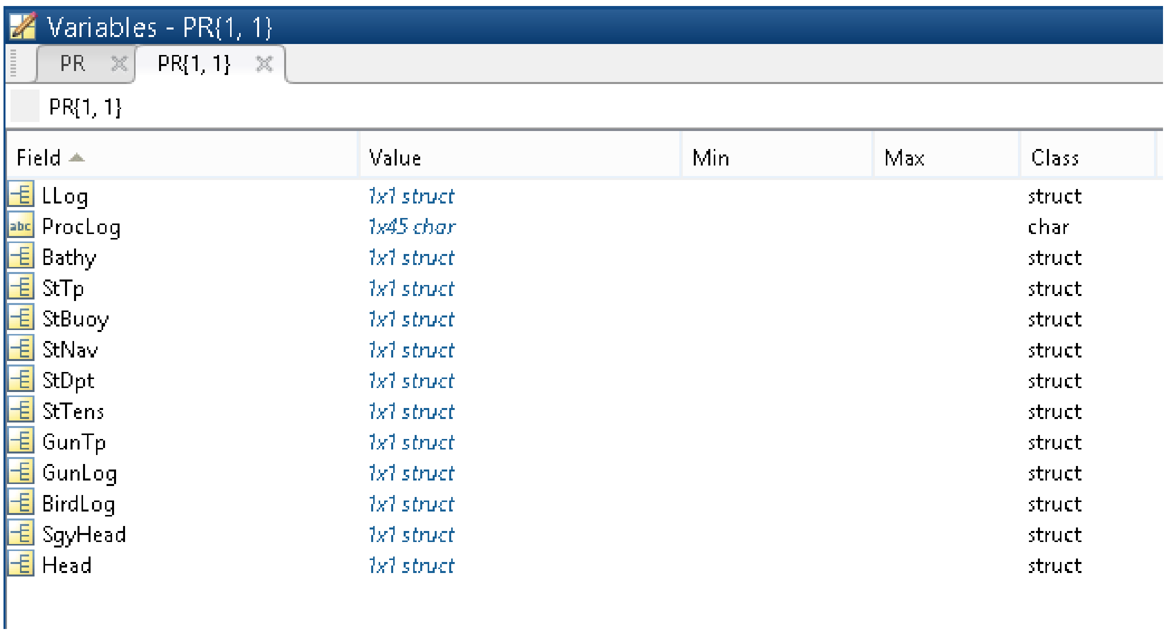
-- LLog.LName – survey line name,

-- LHead – survey line direction,

-- CompDay, CompTime – time marks for Computer's time (usually taken from gLog computer),

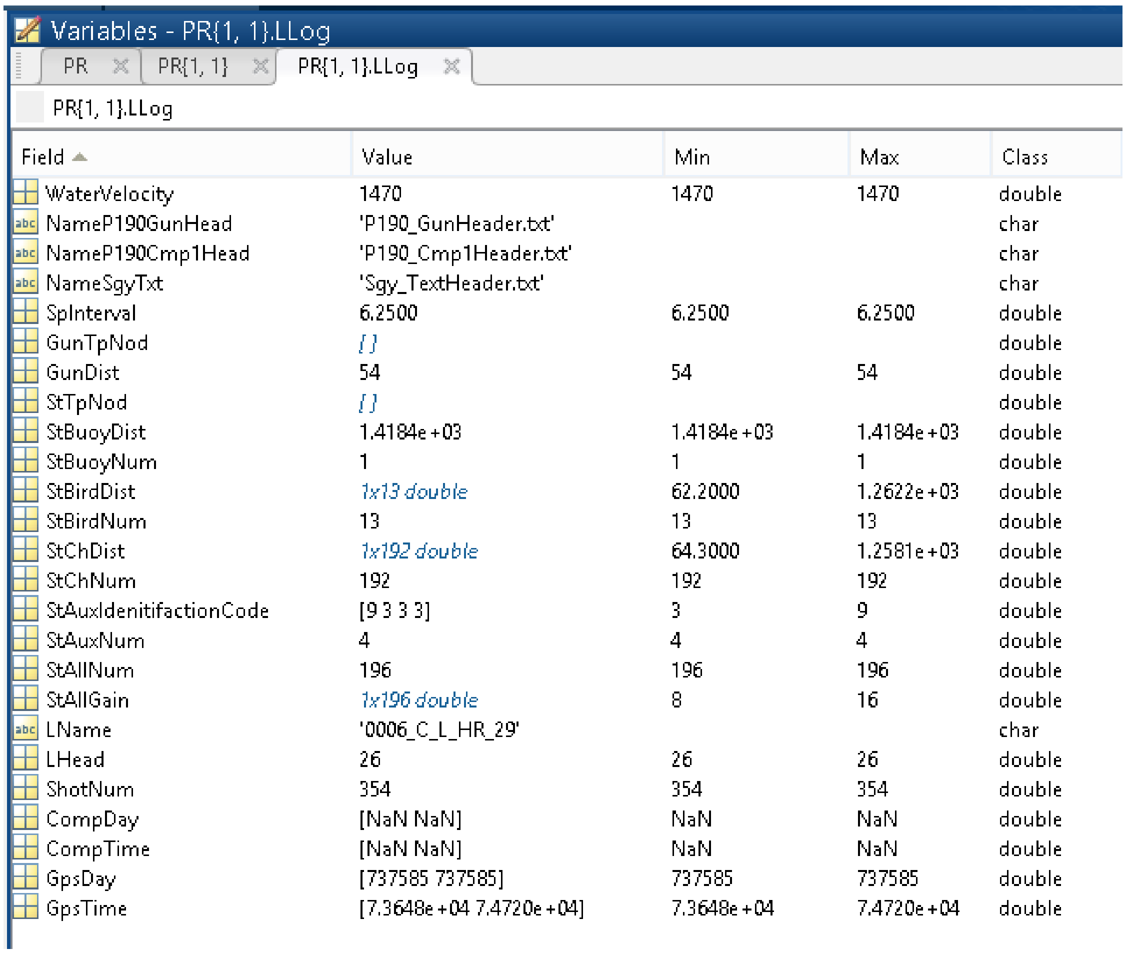
-- GpsDay, GpsTime – time marks for GPS time or UTC (usually taken from GPS's messages),

(2) fields, copied from SU-variable as "potentially changed general settings for Survey Line".



Field	Value	Min	Max	Class
LLog	1x1 struct			struct
ProcLog	1x45 char			char
Bathy	1x1 struct			struct
StTp	1x1 struct			struct
StBuoy	1x1 struct			struct
StNav	1x1 struct			struct
StDpt	1x1 struct			struct
StTens	1x1 struct			struct
GunTp	1x1 struct			struct
GunLog	1x1 struct			struct
BirdLog	1x1 struct			struct
SgyHead	1x1 struct			struct
Head	1x1 struct			struct

Figure 6.4 PR{1} fields example (seismic streamer geometry)



Field	Value	Min	Max	Class
WaterVelocity	1470	1470	1470	double
NameP190GunHead	'P190_GunHeader.txt'			char
NameP190Cmp1Head	'P190_Cmp1Header.txt'			char
NameSgyTxt	'Sgy_TextHeader.txt'			char
SpInterval	6.2500	6.2500	6.2500	double
GunTpNod	[]			double
GunDist	54	54	54	double
StTpNod	[]			double
StBuoyDist	1.4184e+03	1.4184e+03	1.4184e+03	double
StBuoyNum	1	1	1	double
StBirdDist	1x13 double	62.2000	1.2622e+03	double
StBirdNum	13	13	13	double
StChDist	1x192 double	64.3000	1.2581e+03	double
StChNum	192	192	192	double
StAuxIdenitifactionCode	[9 3 3 3]	3	9	double
StAuxNum	4	4	4	double
StAllNum	196	196	196	double
StAllGain	1x196 double	8	16	double
LName	'0006_C_L_HR_29'			char
LHead	26	26	26	double
ShotNum	354	354	354	double
CompDay	[NaN NaN]	NaN	NaN	double
CompTime	[NaN NaN]	NaN	NaN	double
GpsDay	[737585 737585]	737585	737585	double
GpsTime	[7.3648e+04 7.4720e+04]	7.3648e+04	7.4720e+04	double

Figure 6.5 PR{1}.LLog fields example (seismic streamer geometry)

6.4.5 GL (“geophysical processing” data)

The cell vector with “geophysical processing data” (the data created as the result of PR-variable’s data processing). The examples: SBP-sections picking results (seismic horizons), magnetic targets picking and position/mass estimation result. Each cell includes a number of field-names levels in accordance with processing method’s notion for own survey type.

The processing data are discussed in own papers, specified for each processing method.

7. Processing-flow script

7.1 Script's commands

In general, the SU, PR, RS, ML and GL variables are not used as function parameters for RAM using optimization; the single variable's cell can use as function parameter only (see ***Script's command function***). To create compact variable's space and exhaustive processing flow code, the processing flow is realized using MatLab scripts. The graphic interface is not used for input (it means menu and forms) to keep code compact view; for example, the SU fields values are defined in script code.

The Ge0Mlib library used the **special commands format for own scripts**:

{'ComName',Par1,Par2,...,ParN,ParDef1,...ParDefN};ScriptName;

where

{....} – the script initialization data (command name and parameters);

ComName – the command name;

Par1..ParN – the additional data and parameters for command execution; this type of parameters is mandatory and must be defined;

ParDef1..ParDefN – the additional data and parameters for command execution; if this type of parameters is not writes in command line, than defined values wrote in the script-code are used;

ScriptName – the name of script will execute.

For example: {'PR_Set','0006_C_L_HR_29',26,1};gUhrGeoEelGeomR02;

The script's command is works with SU, PR, RS, ML and GL variables. Other variables located in general MatLab workspace can be used as command parameters.

The prefer to use variable names in a script body, which have last symbol "_"; so the out-of-script variables would not deleted or changed. The all "additional" variables used in script's command body is recommend to delete in a last command's code line.

The script for "Field distribution creation" task (***Figure 3.1***) is read files with sensors data and formed Pre-processed Set of data. It includes the follow steps (each of them is realized as one or a number script's command):

(1) Set/load SU fields – the global project's settings input into the SU-fields, using script's code.

(2) Load ML cells – the "extraneous data" input into the ML-cells cells, using script's code. There is formalized code with "known" objects and formats.

(3) Load sensors to RS cells (or PR cells) – the sensors data loading from files to MatLab working space. The each sensor's data is loaded to own cell, using script's code. There is formalized code with "known" objects and formats. The RS cells are used if we have uninterrupted sensors' record; the next step is "cut" data from record using Survey Line's time. If we have sensors' data for survey lines' time only, we can load sensors' data to PR-cells without RS-cells using.

(4) Cut sensors data to PR cells – using the Survey Lines' time from on-line log, we cut the sensors' data to PR-cells; the each cell is the a all sensors record for Survey Line's time. The operator can use "common" name for sensor field or set another name. There is mainly formalized code with "known" objects and formats; the sensors naming is exception.

(5) Sensor's initial Quality Control – the usually initial control is the control of measurements step (for measurements with defined frequency), spikes control, "other nature" trends control (for example, the vessel's heading oscillation can "projected" to towed fish USBL positioning data) and high frequency random noise filtering. Usually, the initial QC is the formalized code with "known" procedures. In this stage the original data saved to fields with postfix "Row"; to sensor fields write "processed" data created using initial control procedures. The QC-flag is created for each sensor too. This step main task is to try removing "specific sensor's and data's features" and creating "ideal sensors data" (see ***Data Abstraction Levels***).

(6) Sensor's interpolation to general time-points – the all sensors' data fields are interpolated to target-sensor data fields (it means the sensor measured the "target geophysical data"). By the result we have a time-synchronized set with different sensors. There is formalized code with "known" objects and formats.

(7) Pre-processed Set forming using Sensor's combining – this step is a calculation with using all sensors data to create Pre-processed Set (see ***Data Abstraction Levels***). In a general case the Pre-processed Set contained physical field distribution – the physical field values, time and space coordinates for values. The example 1: for magnetic gradiometer's measurements we create dataset with each magnetic sensor XYZ-coordinates, measurement time and magnetic field value (the additional "method-specified" information is measurements quality like signal strength and environment's information like sensor depth, sensor altitude, Earth's magnetic field variations and some other). The example 2: for seismic streamer we create dataset with source and receiver groups' XYZ-coordinates, shot and trace-record time and physical "deformation field record" (the additional "method-specified" information is sources synchronization/power data, near and/or far field hydrophones records, the sources' and receivers' depth and altitude and some other). This code need to create and formalized for each new equipment set; for state survey process this is the formalized code with "known" objects and formats.

(7) Pre-processed Set export and QC-estimations – the Pre-processed Set is exported to files for using with industrial software (the Processing and Interpretation Procedures; see ***Data Abstraction Levels***); the QC-values and QC-plots are created for data quality descriptions. The exported data examples: P1/90 files with objects coordinates, Nav merged SGY-file for seismic data, database with "magnetic field survey" data for Oasis Montaj. This code need to create and formalized for each new equipment set; for state survey process this is the formalized code with "known" objects and formats.

(8) Load and save SU, RS, PR-variables – this step needs for scripts workspace save and load.

The some additional processing steps can be added to “basic steps”, for example, using high-frequency sensors’ data from RS{...} cells we can calculate the additional “virtual sensor” and use it to follow steps. But for the most parts of scripts the “basic steps” are enough.

7.2 MatLab session example with Script’s commands

The script-commands example for UHR streamer’s geometry calculation (quick start guide for gUhrGeoEelGeomR02 script commands) is presented below:

{'SU_Set'};gUhrGeoEelGeomR02;

Load general Setup parameters for script.

{'ML_Set'};gUhrGeoEelGeomR02;

Define "Map layers". The current script contained only one layer, this is “Line planning”.

{'Main_Load'};gUhrGeoEelGeomR02;

Load PR variable with processed survey lines (or Create PR-variable).

{'PR_Set','0006_C_L_HR_29',26,3};gUhrGeoEelGeomR02;

Create dataset for single line and load Logs for dataset creation. The defined Line Name is 0006_C_L_HR_29. The line direction is 26 degree. The line will upload to PR-variable cell number 3.

The follow logs and Header-files are loaded:

1) streamer’s towpoint position with 1-second period;

2) cluster towpoint position with 1-second period;

3) tail-buoy position with 1-secont period;

4) birds log (used exported from Qinsy);

5) GeoEel seismic station’s three txt-logs (Nav for shots log, Depth log, Tension log);

6) GunLink txt-log (includes shots time, cluster depth, pressure, synchronization errors, etc.);

7) MBES survey “belt” for current seismic line;

8) Sgy-file created by GeoEel seismic station;

9) txt-file with P1/90 header template;

10) txt-file with Sgy’s text-header template.

{'Fix&Shot_QC',3};gUhrGeoEelGeomR02;

The visual Miss Shot’s searching and analysis. Used survey line loaded to PR-variable cell number 3.

{'Time_QC','StTp','Gps',1,1:3};gUhrGeoEelGeomR02;

Delete spikes and smooth Time for field 'StTp' (streamer towpoint). The time postfix is 'Gps' (usually can be 'Gps' or 'Comp'). Time smooth window is 1 (no smooth). The use data in PR-variable cell number from 1 to 3.

{'Time_QC','StBuoy','Gps',1,1:3};gUhrGeoEelGeomR02;

Delete spikes and smooth Time for field 'StBuoy' (streamer buoy position). The time postfix is 'Gps' (usually can be 'Gps' or 'Comp'). Time smooth window is 1 (no smooth). The use data in PR-variable cell number from 1 to 3.

{'Time_QC','GunTp','Gps',1,1:3};gUhrGeoEelGeomR02;

Delete spikes and smooth Time for field 'GunTp' (seismic cluster towpoint). The time postfix is 'Gps' (usually can be 'Gps' or 'Comp'). Time smooth window is 1 (no smooth). The use data in PR-variable cell number from 1 to 3.

{'Coord_QC','StTp','Gps',10,1:3};gUhrGeoEelGeomR02;

Delete spike&smooth for Coordinates, field 'StTp' (streamer towpoint). The time postfix is 'Gps' (need for deleted coordinates interpolation; usually can be 'Gps' or 'Comp'). Time smooth window is 10. The use data in PR-variable cell number from 1 to 3.

{'Coord_QC','StBuoy','Gps',10,1:3};gUhrGeoEelGeomR02;

Delete spike&smooth for Coordinates, field 'StBuoy' (streamer buoy position). The time postfix is 'Gps' (need for deleted coordinates interpolation; usually can be 'Gps' or 'Comp'). Time smooth window is 10. The use data in PR-variable cell number from 1 to 3.

{'Coord_QC','GunTp','Gps',10,1:3};gUhrGeoEelGeomR02;

Delete spike&smooth for Coordinates, field 'GunTp' (seismic cluster towpoint). The time postfix is 'Gps' (need for deleted coordinates interpolation; usually can be 'Gps' or 'Comp'). Time smooth window is 10. The use data in PR-variable cell number from 1 to 3.

{'PR_CreateSet',1:3};gUhrGeoEelGeomR02;

The create dataset with sensors were synchronized; the dataset time is the Shots moments.

{'OutPromaxGeom',1:3};gUhrGeoEelGeomR02;

Create spreadsheet file for ProMax with Gun position. The use data in PR-variable cell number from 1 to 3.

{'OutP190Geom','Gun',2018,1:3,10,'S'};gUhrGeoEelGeomR02;

Create file P1/90 and AutoCad-script for Gun position. The use data in PR-variable cell number from 1 to 3. Year 2018. The points step in catalogue-file 10. The P1/90 Record Identifier is 'S' (Source).

{'OutP190Geom','Cmp1',2018,1:3,10,'C'};gUhrGeoEelGeomR02;

Create file P1/90 and AutoCad-script for Cmp1. The use data in PR-variable cell number from 1 to 3. Year 2018. The points step in catalogue-file 10. The P1/90 Record Identifier is 'C' (Common Mid Point).

{'OutSgyGeom',1:3};gUhrGeoEelGeomR02;

Save Sgy file with Navigation merged. The use data in PR-variable cell number from 1 to 3 (save three files).

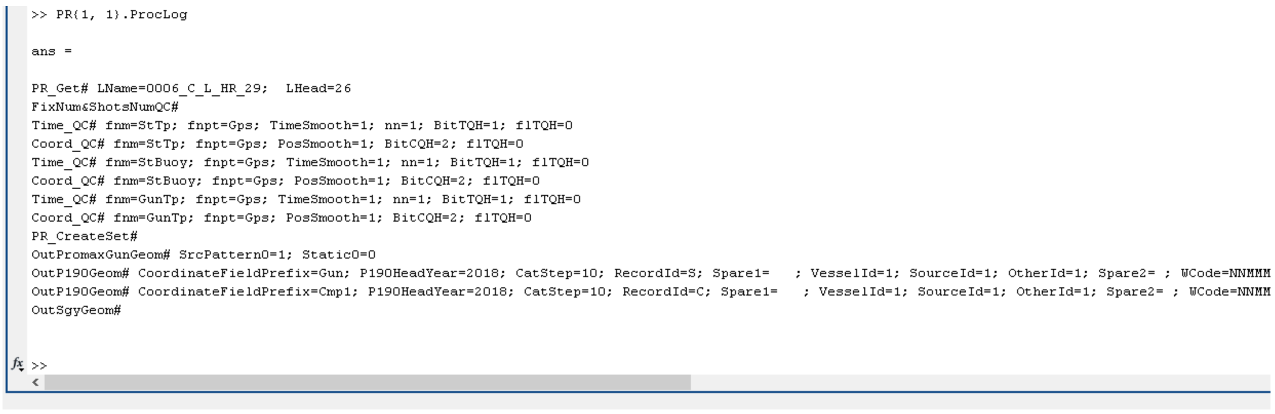
{'Main_Save'};gUhrGeoEelGeomR02;

The PR variable Save.

7.3 Survey Line processing log

To memorize commands were sent to script for Survey Line's sensor processing the "ProcLog" field is used – the PR{n}.ProcLog contained the processing commands log for current Survey Line. There is a String with LF (0Axh) lines delimited.

The PR{n}.ProcLog example is shown in ***Figure 7.1***. The ProcLog based MatLab session is shown for "***MatLab session example with Script's commands***".



```
>> PR{1, 1}.ProcLog

ans =

PR_Get# LName=0006_C_L_HR_29;  LHead=26
FixNum&ShotsNumQC#
Time_QC# fnm=StTp; fnpt=Gps; TimeSmooth=1; nn=1; BitTQH=1; flTQH=0
Coord_QC# fnm=StTp; fnpt=Gps; PosSmooth=1; BitCQH=2; flTQH=0
Time_QC# fnm=StBuoy; fnpt=Gps; TimeSmooth=1; nn=1; BitTQH=1; flTQH=0
Coord_QC# fnm=StBuoy; fnpt=Gps; PosSmooth=1; BitCQH=2; flTQH=0
Time_QC# fnm=GunTp; fnpt=Gps; TimeSmooth=1; nn=1; BitTQH=1; flTQH=0
Coord_QC# fnm=GunTp; fnpt=Gps; PosSmooth=1; BitCQH=2; flTQH=0
PR_CreateSet#
OutPromaxGunGeom# SrcPattern0=1; Static0=0
OutP190Geom# CoordinateFieldPrefix=Gun; P190HeadYear=2018; CatStep=10; RecordId=S; Spare1=   ; VesselId=1; SourceId=1; OtherId=1; Spare2= ; WCode=NNMMM
OutP190Geom# CoordinateFieldPrefix=Cmp1; P190HeadYear=2018; CatStep=10; RecordId=C; Spare1=   ; VesselId=1; SourceId=1; OtherId=1; Spare2= ; WCode=NNMM
OutSgyGeom#


>>
```

Figure 7.1 PR{n}.ProcLog example (seismic streamer geometry)

The code with PR{n}.ProcLog forming need to add after script's operations, before variables clearing.

The "ProcLog" is not "total control instrument"; there are not logged manipulations with data can takes using Command Window (it is logged in MatLab session) or in a script's body code (it is not logged anywhere), but ProcLog-field is a useful instrument for state-processing steps and commands parameters logging. The script's commands which not deal with Survey Lines data (PR-variable's cells) are not logged into PR{n}.ProcLog field.

7.4 Script's command functions

A number of scripts commands usually are realized using the same code, for example – TimeQC (a sensor's time marks initial Quality Control). This code is "moved" to "command functions" – this functions code can be easy "moved" back (using copy-paste) to script code. It demands the specific variable's names in function code and some "specific programming style".

This feature is realized, in one hand – for script's code simplifies and string number decrease, in other hand – if we need "specify processing" (which need changes in code), we can "move back" "command function's" code to script and adopt it to processing task.

Citation