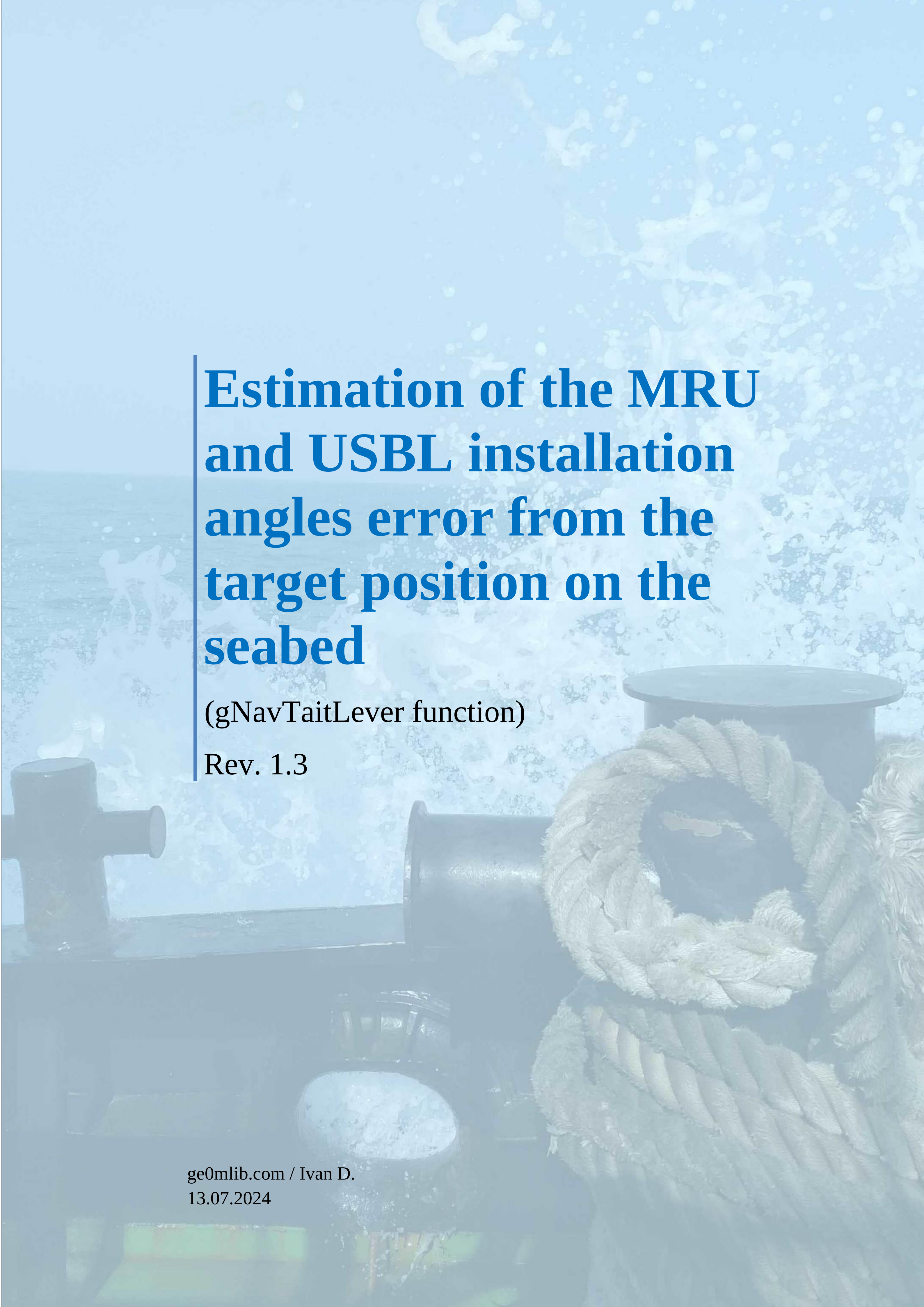


Estimation of the MRU and USBL installation angles error from the target position on the seabed

(gNavTaitLever function)

Rev. 1.3

ge0mlib.com / Ivan D.

13.07.2024

Contents

History of changes

Version	Date	Description of changes
1.0	2021-05-21	Document created // Ivan D.
1.1	2021-05-30	Changes were made to Model-2, the text was corrected // Ivan D.
1.2	2021-07-06	The text about the coordinate conversion graph was added; Appendix 2 was added // Ivan D.
1.3	2024-07-13	Text corrected // Ivan D.

Figures list

Introduction

The document discusses the use of the gNavTaitLever function from the ge0mlib library in context of the MRU and USBL installation angles error estimation based on the position of a fixed target on the seabed. Such a target can be either a USBL transponder or a seabed feature (a relief feature, an object on the seabed) recorded on data of the SSS or MBES that are installed on a USBL-positioned device.

The first chapter provides a brief description of the used mathematical apparatus: a coordinate system and a coordinate axes naming system for vessel navigation are introduced; rotation (Tait-Bryan) matrices and sequences of rotation matrices ("manipulator model") are considered.

The second chapter discusses the first approximation model (Model-1), which includes:
-- USBL with installation angles that include setting angle errors,
-- MRU, which is oriented precisely along the geometric axes of the vessel hull.
The following terms are also introduced: discrepancies for USBL installation angles error estimation, array of discrepancies (dst), ellipsoid of discrepancies, total array of discrepancies. Examples of how to use the gNavTaitLever function are provided.

The third chapter discusses the model (Model-2), which includes:
-- USBL with installation angles that include setting angle errors;
-- MRU with installation angles that include setting angle errors;
-- offset of LeverUSBL relative to the metacenter.
A special case is considered when the USBL and MRU installation angles are equal; The algorithm and calculation results for "selection of USBL installation angles" to compensate for the installation angles error are presented.

All constructions described in this document are static; that is, they are performed on the basis of elementary geometric constructions without taking into account the dynamic characteristics involved in object models.

1 Definition of coordinate systems and operations with rotation matrices

1.1 Coordinate systems

To designate the coordinate axes, we introduce the naming system described below.

GCS – Global Coordinate System

The axes are specified in a geographic coordinate system (geographic frame); The X axis points north, the Y axis points east, the Z axis points upward, positive rotation angles are counted clockwise. For the convenience of geometric calculations, coordinates can be specified in meters on a rectangular projection with low distortion. Axes are used to display the position of various “objects” (targets, vessel position, measurements of physical fields, etc.).

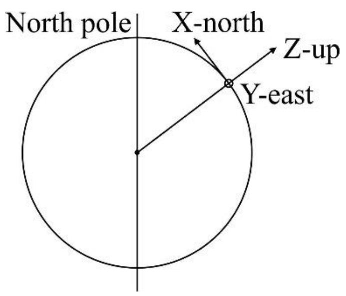


Figure1.1 GCS axes

If the axes are specified in the GCS direction, but the origin of coordinates coincides with the position of a particular device or point, then the coordinates origin point name is added to the GCS through a hyphen:

GCS-META – axes are specified in the GCS direction, but the origin coincides with the metacenter.

GCS-MBES – the origin coincides with the position of the MBES acoustic center;

GCS-MRU – the origin coincides with the position of the MRU;

GCS-USBL – the origin coincides with the position of USBL.

VCS – Vessel's Coordinate System

The axes are specified by the geometry of the vessel hull; the origin of coordinates is located in the metacenter, the X axis points diametrically to the bow of the vessel, the Y axis points to the starboard side, the Z axis points upward, positive rotation angles are counted clockwise. Axes are used to account the quantities associated with the inclinations, displacements and accelerations of the vessel.

If the axes are specified in the VCS direction, but the origin of coordinates coincides with the position of a particular device or point, then the coordinates origin point name is added to the VCS through a hyphen: VCS-MBES; VCS-MRU; VCS-USBL. The VCS and VCS-META designations are equivalent.

ECS – Equipment Coordinate System

The axes are specified for a specific device in accordance with its design features, for example, the direction of the transmitting and receiving antennas (let us assume that the direction of the axes and the origin of the ECS coordinates are marked on the device hull). In the designations of the ECS axes, the name of the device is written through an underscore: ECS_MBES, ECS_MRU, ECS_USBL; with this designation it is assumed that the origin of coordinates is at the point where the device is positioned (the designation ECS_MRU is identical to ECS_MRU-MRU). If the origin of coordinates is shifted relative to the device, then the name of the origin point is added to the designation through a hyphen; for example, if the origin of coordinates is located at the metacenter for axes with the direction of “MRU device axes”, then these axes will be designated as ECS_MRU-META.

The deviation of the device axes (ECS) direction from the vessel hull axes (VCS) will be called installation angles.

1.2 Elements of the transformation matrix

To transition between coordinate axes, we will use rotation matrices (a special case of transformation matrices).

In the two-dimensional case (X axis is vertical; Y axis is horizontal, clockwise rotation is positive), the new coordinates of the point (in the “old” coordinate axes) are calculated as the multiplication of the transformation matrix by the coordinate matrix:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a1 & a2 & tx \\ b1 & b2 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Calculations are performed using the formulas:

$$x' = a1x + a2y + tx;$$
$$y' = b1x + b2y + ty.$$

If a conditional object is specified by a set of points, then performing calculations for each point, we obtain the following transformations of the object:

a1 – horizontal scale change (reflection with negative values);

b1 – vertical shear (skew; change of shear direction with negative values);

a2 – horizontal shear (skew; change of shear direction with negative values);

b2 – vertical scale change (reflection with negative values);

tx – x-axis displacement (translation);

ty – y-axis displacement (translation).

Rotation is a combination of scaling and shear to preserve the proportions of the object, sines and cosines are used:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

When using the rotation matrix, we can say:

-- about “rotation of a vector” (directed from the coordinates origin to a point the coordinates of which are rotated), when the coordinates of a vector rotated by a given angle in “fixed” coordinate axes are calculated;

-- about “rotation of axes”, when it is assumed that the vector maintains its direction, and the coordinate axes rotate; accordingly, after calculations we obtain the coordinates of the “fixed vector” in the axes rotated by a given angle.

In the second case, when substituting into the formula, the sign of the rotation angle must be changed. In another way, this case can be formulated as follows: positive rotation angles are counted clockwise if it is assumed that a certain vector rotates relative to the axes; positive rotation angles are counted counterclockwise if the axes are assumed to rotate relative to some vector.

In the case of rotation in three coordinate axes, a 4x4 transformation matrix is used:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a1 & a2 & a3 & tx \\ b1 & b2 & b3 & ty \\ c1 & c2 & c3 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

To optimize computer calculations, we will use 3x3 “rotation matrices for three coordinate axes” instead of “4x4 transformation matrices”. The shift of the coordinates origin by values [tx ty tz] will be performed by a separate operation of “transfer of the coordinates origin”.

1.3 Rotation matrices for three coordinate axes

We will perform the rotation in a previously defined coordinate system: the X axis points to the north, the Y axis points to the east, the Z axis points upward; positive rotation angles are counted clockwise. We will bring the readings of all devices to this coordinate axes configuration, recalculating, if necessary, the devices readings obtained in their own coordinate axes, different from those described.

Let us accept the following order of rotations (the rotation intrinsics are known as: yaw, pitch and roll): (a) rotation of the axes relative to Z; (b) rotation of the axes relative to the previously rotated Y; (c) rotation of the axes relative to the previously rotated X. Let us denote this order as "direct rotation" (forward). If we have the values [Heading, Pitch, Roll], then we can recalculate “some point coordinates on the vessel hull after the vessel has turned”; to do this, the vector is rotated relative to the GCS-META axes (point coordinates will also be calculated in GSC-META axes).

We denote the reverse order of rotations as "reverse rotation". With the measured coordinates of the object relative to the vessel in VSC (using USBL), according to the MRU readings, we can recalculate the coordinates of the object in GSC-META. In this case, “reverse rotation” is used (in the order of Roll, Pitch, Heading); Since the coordinate axes are rotated, we change the sign of the rotation angles (positive angles are counted counterclockwise).

Reverse rotation (VSC >> GCS-META) is performed using the following formula:

$$R = R_z(\alpha)\,R_y(\beta)\,R_x(\gamma) = \overbrace{\begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}}^{\text{yaw}} \overbrace{\begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}}^{\text{pitch}} \overbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}}^{\text{roll}}$$

$$R = \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix}$$

https://en.wikipedia.org/wiki/Rotation_matrix

In the first line, each of the three matrices corresponds to a rotation around one of the coordinate axes (Z1Y2X3). Moreover, the order of rotation corresponds to the order of multiplication of the angle matrices by the coordinate matrix (the matrix on the right is multiplied first). In the above formula, rotation is first performed by Roll (X3), then by Pitch (Y2), then by Heading (Z1).

There are formulas below for a different order of rotations around coordinate axes. In this case, “forward rotation [Heading, Pitch, Roll]” corresponds to the order X1Y2Z3, and “reverse rotation [Roll, Pitch, Heading]” corresponds to the order Z1Y2X3.

Proper Euler angles	Tait–Bryan angles
$X_1Z_2X_3 = \begin{bmatrix} c_2 & -c_3s_2 & s_2s_3 \\ c_1s_2 & c_1c_2c_3 - s_1s_3 & -c_3s_1 - c_1c_2s_3 \\ s_1s_2 & c_1s_3 + c_2c_3s_1 & c_1c_3 - c_2s_1s_3 \end{bmatrix}$	$X_1Z_2Y_3 = \begin{bmatrix} c_2c_3 & -s_2 & c_2s_3 \\ s_1s_3 + c_1c_3s_2 & c_1c_2 & c_1s_2s_3 - c_3s_1 \\ c_3s_1s_2 - c_1s_3 & c_2s_1 & c_1c_3 + s_1s_2s_3 \end{bmatrix}$
$X_1Y_2X_3 = \begin{bmatrix} c_2 & s_2s_3 & c_3s_2 \\ s_1s_2 & c_1c_3 - c_2s_1s_3 & -c_1s_3 - c_2c_3s_1 \\ -c_1s_2 & c_3s_1 + c_1c_2s_3 & c_1c_2c_3 - s_1s_3 \end{bmatrix}$	$X_1Y_2Z_3 = \begin{bmatrix} c_2c_3 & -c_2s_3 & s_2 \\ c_1s_3 + c_3s_1s_2 & c_1c_3 - s_1s_2s_3 & -c_2s_1 \\ s_1s_3 - c_1c_3s_2 & c_3s_1 + c_1s_2s_3 & c_1c_2 \end{bmatrix}$
$Y_1X_2Y_3 = \begin{bmatrix} c_1c_3 - c_2s_1s_3 & s_1s_2 & c_1s_3 + c_2c_3s_1 \\ s_2s_3 & c_2 & -c_3s_2 \\ -c_3s_1 - c_1c_2s_3 & c_1s_2 & c_1c_2c_3 - s_1s_3 \end{bmatrix}$	$Y_1X_2Z_3 = \begin{bmatrix} c_1c_3 + s_1s_2s_3 & c_3s_1s_2 - c_1s_3 & c_2s_1 \\ c_2s_3 & c_2c_3 & -s_2 \\ c_1s_2s_3 - c_3s_1 & c_1c_3s_2 + s_1s_3 & c_1c_2 \end{bmatrix}$
$Y_1Z_2Y_3 = \begin{bmatrix} c_1c_2c_3 - s_1s_3 & -c_1s_2 & c_3s_1 + c_1c_2s_3 \\ c_3s_2 & c_2 & s_2s_3 \\ -c_1s_3 - c_2c_3s_1 & s_1s_2 & c_1c_3 - c_2s_1s_3 \end{bmatrix}$	$Y_1Z_2X_3 = \begin{bmatrix} c_1c_2 & s_1s_3 - c_1c_3s_2 & c_3s_1 + c_1s_2s_3 \\ s_2 & c_2c_3 & -c_2s_3 \\ -c_2s_1 & c_1s_3 + c_3s_1s_2 & c_1c_3 - s_1s_2s_3 \end{bmatrix}$
$Z_1Y_2Z_3 = \begin{bmatrix} c_1c_2c_3 - s_1s_3 & -c_3s_1 - c_1c_2s_3 & c_1s_2 \\ c_1s_3 + c_2c_3s_1 & c_1c_3 - c_2s_1s_3 & s_1s_2 \\ -c_3s_2 & s_2s_3 & c_2 \end{bmatrix}$	$Z_1Y_2X_3 = \begin{bmatrix} c_1c_2 & c_1s_2s_3 - c_3s_1 & s_1s_3 + c_1c_3s_2 \\ c_2s_1 & c_1c_3 + s_1s_2s_3 & c_3s_1s_2 - c_1s_3 \\ -s_2 & c_2s_3 & c_2c_3 \end{bmatrix}$
$Z_1X_2Z_3 = \begin{bmatrix} c_1c_3 - c_2s_1s_3 & -c_1s_3 - c_2c_3s_1 & s_1s_2 \\ c_3s_1 + c_1c_2s_3 & c_1c_2c_3 - s_1s_3 & -c_1s_2 \\ s_2s_3 & c_3s_2 & c_2 \end{bmatrix}$	$Z_1X_2Y_3 = \begin{bmatrix} c_1c_3 - s_1s_2s_3 & -c_2s_1 & c_1s_3 + c_3s_1s_2 \\ c_3s_1 + c_1s_2s_3 & c_1c_2 & s_1s_3 - c_1c_3s_2 \\ -c_2s_3 & s_2 & c_2c_3 \end{bmatrix}$

https://en.wikipedia.org/wiki/Euler_angles#Intrinsic_rotations

1.4 Manipulator model (sequence of rotations for joints)

“To describe the rotational and translational connections between adjacent links of the manipulator, Denavit and Hartenberg proposed a matrix method for sequentially constructing of coordinate systems related to each link of the kinematic chain. The meaning of the Denavit-Hartenberg representation (DH representation) is to form a homogeneous transformation matrix with a dimension of 4x4 and describing the position of the coordinate system of each link relative to the coordinate system of the previous link. This makes it possible to sequentially transform the coordinates of the manipulator’s grip from the reference system associated with the last link into the base reference system, which is the inertial coordinate system for the dynamic system under consideration.” (Краснобаев Е.А. Теоретические основы робототехники. - курс лекций, УО “ВГУ им. П.М. Машерова”, Витебск 2012.)

The model of the manipulator, as a sequence of articulated joints (rotation of coordinate axes) and levers (transfer of the coordinates origin) is “physically clear”, so we will further use the analogy with the manipulator both in the text and in the figures (for example, Figure 2.1).

The calculations will also use sequences of coordinate axes rotation matrices and shifts of the origin (for the current coordinate axes), similar to the Denavit-Hartenberg representation. However, a “single transformation matrix” will not be formed; each transition between "manipulator joints" will be performed as a separate operation.

For mathematical constructions the following condition must be met: when transferring coordinates between “manipulator joints”, the axes of the coordinate system of one joint must be deployed parallel to the coordinate system axes of the other joint.

1.5 MatLab functions

The description of the models will be given both in the form of mathematical formulas and based on MatLab program code. This will use the gNavTaitLever function (check Appendix 1. gNavTaitLever function text), which is described below.

XYZ_Lev=gNavTaitLever(Head,Pitch,Roll,Lever,RotDirect)

where

Head – row vector that contains the angle of rotation in the X0Y plane in degrees, for times t1...tn;

Pitch – row vector that contains the angle of rotation in the Z0Y plane in degrees, for times t1...tn;

Roll – row vector that contains the angle of rotation in the Y0Z plane in degrees, for times t1...tn;

Lever – three lines containing the offset value in meters [Lx1... Lxm; Ly1... Lym; Lz1... Lzm];

RotDirect – rotation direction key: 'frw' - “forward” rotation; 'rev' - “reverse” rotation;

XYZ_Lev – coordinates of Lever after rotation; the variable contains a three-dimensional array: three lines with values [Lx(t1)...Lx(tn); Ly(t1)...Ly(tn); Lz(t1)...Lz(tn)] (i.e., offset coordinates after rotation for times t1...tn); along the third dimension there are the calculation results for each of the offsets Lever_1...Lever_m. If the Roll or Pitch or Head value is a scalar, then a row vector of the required length will be formed from it.

Rotation of the offsets (Lever) is performed in the coordinate system: the X axis points to the north, the Y axis points to the east, the Z axis points upward; positive rotation angles are counted clockwise. When rotating the coordinate axes (relative to the Lever offsets), the sign of the rotation angles must be changed.

The library also contains two functions gNavTaitNLever and gNavTaitLeverN. The specified functions form arrays with calculation results of the same dimension as the input arrays. Also, these functions operate with the large arrays 100-300 times faster than gNavTaitLever, but use more RAM. An increase in the speed of functions is achieved by eliminating the use of loops in the functions body. For a 301x301x301 array, for example, when running gNavTaitTest (check Appendix 2. Script for calculations using Model-2) for calculations using gNavTaitLever, we get the following calculation time:

```
>> gNavTaitTest;
Using for-cycle
Elapsed time is 4256.362563 seconds.
```

When running gNavTaitTest (check Appendix 2. Script for calculations using Model-2) for calculations using gNavTaitNLever and gNavTaitLeverN, we get a speed increase of 366 times:

```
>> gNavTaitTest;
Not using for-cycle
Elapsed time is 11.625237 seconds.
```

gNavTaitNLever

Allows you to specify Head, Pitch, Roll arrays at the input. The output is X, Y, Z arrays, the same size as the Head, Pitch, Roll arrays. Valuescan be calculated for multiple offsets with the results for each coordinate of each offset being output in a separate variable.

gNavTaitLeverN

Allows you to specify arrays of the offsets Lx, Ly, Lz at the input. The output is arrays X, Y, Z the same size as arrays Lx, Ly, Lz. Values can be calculated for several rotation angles: Heading, Pitch, Roll, and the results for each angle are being output in a separate variable.

Functions gNavTaitNLever and gNavTaitLeverN are used when operating with large amounts of data to reduce calculation time and avoid the use of loops in the main program.

1.6 Summary of Section 1

1) Coordinate axes directions are specified, in which further calculations will be made: the X axis points to the north, the Y axis points to the east, the Z axis points upward; positive rotation angles are counted clockwise.
We consider that the readings of all devices are brought to this configuration of coordinate axes.

2) The naming system for designating coordinate axes is introduced in application to vessel navigation.

3) Matrices for "forward" and "reverse" rotation are considered in three coordinate axes.
The terms "vector rotation in the coordinate axes" and "rotation of coordinate axes relative to a fixed vector" are separated.

4) When performing calculations, the use of coordinate axes rotation matrices sequences and origin shifts (for the current coordinate axes) is declared similar to the Denavit-Hartenberg representation.

5) The gNavTaitLever function (MatLab) is described, which is used further in calculations and in describing mathematical models.

2 Model-1: USBL installation angles, heading and vessel motions

2.1 Description of Model-1

For simplicity, let’s assume that the installation angles, i.e. the deviation of the instrument axes (ECS) direction from the vessel hull axes (VCS), are zero. Errors in the installation angles of the device are the values referred to as dH, dP, dR by which the axes of the device are rotated (in the current coordinate axes) during the sequence of dH, dP, dR rotations (if an MRU was installed on the device hull, it would display the values of the dH , dP, dR angles). Further, we will add a letter to the dH, dP, dR designations indicating the type of device.

Let's consider a model that contains USBL and MRU. USBL is installed with an error in the installation angles (dHu, dPu, dRu). The MRU is installed exactly in the direction of the vessel hull axes (VCS) without errors and displays the heading and vessel motions angles (Heading, Pitch, Roll). The coordinates of the target tr are determined by USBL in the ECS-USBL axes; it is necessary to calculate the target coordinates in the GCS-META axes (Figure 2.1).

To solve the problem, we will perform a sequential rotation of the coordinate axes from ECS-USBL towards GCS-META so that the direction of the previous axes coincides with the direction of subsequent axes (then the target coordinates will be “transmitted between the axes”). We will use 3x3 rotation matrices, if necessary, shifting the origin of the axes to a new rotation point.

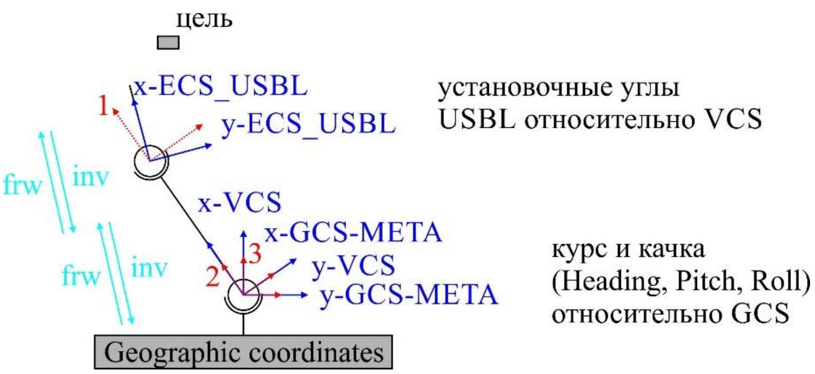


Figure 2.1 Model-1, shifts and rotations of coordinate axes

Let us describe the sequence of rotations-shifts shown in Figure 2.1:

-- the target is located in the ECS_USBL coordinate axes (axes with errors in the USBL installation angles);

-- we rotate the ECS_USBL axes to the installation angles error, obtaining the target coordinates in axes 1, which are located in the direction of the vessel axes (a more accurate definition is to recalculate the target coordinates from the ECS_USBL axes to axis 1);

-- we shift axes 1 (USBL installation point) to the VCS axes (vessel metacenter), we obtain the target coordinates in axes 2;

-- as a result of rotating and shifting the origin of the axes we obtain the target coordinates not in the ECS_USBL axes (axes of the USBL transducer with errors in the installation angles), but in the VCS axes (axes of the vessel hull, origin of coordinates in the metacenter);

-- we rotate axes 2 to the heading and pitching angles (Roll, Pitch, Heading) to the GCS-META axes (direction of the geographic coordinate axes), we obtain the target coordinates in axes 3;

-- as a result of two rotations and one displacement we recalculated the target coordinates from the USBL transducer axes (ECS_USBL) into axes with the direction that coincides with the direction of the geographic coordinate axes, and the origin is in the metacenter of the vessel (GCS-META).

2.2 USBL installation angles error

If there is an error in the USBL installation angles, the target coordinates are displayed in the “rotated” (relative to the vessel hull) ECS_USBL axes; the assumption that the directions of the USBL and the vessel hull (VCS) axes are the same leads to "an error in calculating the target coordinates due to an error in the USBL installation angles."

Let USBL be installed at the vessel metacenter (no LeverUSBL offset), and the vessel hull axes (VCS) be located in the GCS-META direction. Target coordinates in GCS-META – tr=[120; 60; -80].

If the error in the USBL installation angles is increased for all angles by +2 degrees (dHu, dPu, dRu), then the ECS_USBL axes will rotate by 2 degrees relative to the VCS coordinate axes; USBL will show the target coordinates in ECS_USBL.

Let's determine the target coordinates in the ECS_USBL axes with an error in the installation angles of +2 degrees, using the following MatLab code:

```
tmp001=gNavTaitLever(-2,-2,-2,tr,'frw')
tmp001=
124.738496964113
53.0998924855046
-77.5932264644392
```

The sign of the rotation angles changes, since the coordinate axes are rotated relative to the “fixed” given vector. The USBL axes are rotated as 'frw' (forward – Heading, Pitch, Roll), that is, “the axes are twisted”.

When compensating for errors in installation angles, the USBL axes are rotated as 'rev' (reverse – Roll, Pitch, Heading). In this case, the sign of the rotation angles [-2 -2 -2] (rotation of the axes) changes, since the rotation is performed by “unwinding the axes”:

```
gNavTaitLever(2,2,2,tmp001,'rev')
ans =
120
60
-80
```

The target coordinates returned to the initial value tr=[120; 60; -80].

2.3 USBL installation angles error, vessel pitching and heading

Let USBL be installed at the vessel metacenter (no LeverUSBL offset), and the vessel hull axes (VCS) be located in the GCS-META direction. Target coordinates in GCS-META – tr=[120; 60; -80].

Let the MRU readings (at the time of fixing the target coordinates) be [10 8 5] degrees (Heading, Pitch, Roll) for a vessel with an error in the USBL installation angles of +2 degrees, then the tr target coordinates can be recalculated from the GCS-META axes to the ECS_USBL axes in the following way:

-- rotate GCS-META to the VCS direction using MRU angles:

```
tmp002=gNavTaitLever(-10,-8,-5,tr,'frw')
tmp002=
138.478183407984
32.7603577655034
-64.4247753515762
```

-- rotate the VCS to the ECS_USBL direction using the USBL installation angles error:

```
tr1=gNavTaitLever(-2,-2,-2,tmp002,'frw')
tr1 =
141.700536213638
25.8135117529133
-60.4534585259582
```

To move from target coordinates according to ECS_USBL (measured using USBL data) to GCS-META, you must perform:

-- transition from ECS_USBL to VCS, “compensating” for USBL installation angles error:

```
tmp002=gNavTaitLever(2,2,2,tr1,'rev')
tmp002=
138.478183407984
32.7603577655034
-64.4247753515762
```

-- transition from VCS to GCS-META direction, “removing” heading and pitching angles measured by MRU:

```
tr2=gNavTaitLever(10,8,5,tmp002,'rev')
tr2 =
120
60
-80
```

When attempting to perform transition from target coordinates using ECS_USBL to GCS-META using the wrong order of entering rotations - first the MRU angles, and then the USBL angles, we will get the wrong result:

```
tmp002=gNavTaitLever(10,8,5,tr1,'rev')
tmp002=
124.863584386837
53.4789489131471
-77.1303268320328
```

```
tr3=gNavTaitLever(2,2,2,tmp002,'rev')
tr3 =
120.128874737449
60.3673920212812
-79.5288088359389
```

That is, attempting to estimate the USBL installation angles error after applying the current MRU angles is incorrect (changing the order of the rotation matrices for USBL and MRU changes the meaning of the action, as changing the order of the rotation matrices for the Pitch, Roll, Heading angles does).

2.4 Metacenter coordinates in GCS

Let the USBL be installed at the metacenter of the vessel (no LeverUSBL offset) with zero installation angles error. The position of the metacenter is specified in GCS coordinates – sh=[120; 200; 0]. Target coordinates in GCS – tr=[120; 60; -80].

Under the given conditions, before rotations it is necessary to recalculate the coordinates of the target tr=[120; 60; -80] from GCS to GCS-META:

tr-sh=[0; -140; -80].

Let the vessel have MRU readings of [10 8 5] degrees (Heading, Pitch, Roll), then the coordinates of the tr target in the VCS (after entering the MRU values) will be calculated using the following MatLab code:

```
tmp01=gNavTaitLever(-10,-8,-5,tr-sh,'frw')
tmp01=
-12.9403062986162
-144.547923523288
-70.2740797022897
```

If we want to calculate the "target coordinates in GCS after rotating the vessel at the MRU angles [10 8 5]", then we need to add the value sh2 (metacenter coordinates in GCS) to the "metacenter-target" offset (target coordinates in VCS):

```
tmp02=gNavTaitLever(-10,-8,-5,tr-sh,'frw')+sh
tmp02=
107.059693701384
55.4520764767123
-70.2740797022897
```

2.5 Determination of USBL installation angles error when fixing the target coordinates for two vessel positions

Let the USBL be installed at the metacenter of the vessel (no LeverUSBL offset), with the USBL installation angles error. If we have two target coordinates (tr1 and tr2) defined in the GCS at two USBL positions (sh1 and sh2; assuming the vessel is moving in the GCS), then the task of determining the USBL installation angles error can be formulated as follows:

it is necessary to find the value of the USBL installation angles at which the distance between the target positions "corrected for the USBL installation angles" (in GCS) will be minimal.

In the extreme case, when entering corrections for installation angles, the position of the targets must coincide for both USBL (vessel) locations.

In matrix form the task will be written as follows:

[M2_sh2]*[M1]*(tr2-sh2)+sh2=[M2_sh1]*[M1]*(tr1-sh1)+sh1

where

[M2_sh1] – rotation matrix, calculated from the Heading, Pitch, Roll values for heading and pitching angles determined by MRU (reverse, rotation of coordinate axes), for the first USBL position;

[M2_sh2] – rotation matrix, calculated from the Heading, Pitch, Roll values for heading and pitching angles determined by MRU (reverse, rotation of coordinate axes), for the second USBL position;

[M1] – rotation matrix, calculated from the values of dHu, dPu, dRu for USBL installation angles or installation angles error (reverse, rotation of coordinate axes);

sh1 – first position (coordinates) of USBL in GCS, for example [0; 0; 0];

sh2 – second position of USBL in GCS, for example [120; 200; 0];

tr1 – target coordinates in GCS at the first USBL position;

tr2 – target coordinates in GCS at the second USBL position.

If we transform the matrix representation into a scalar form, we get a system of three equations with three unknown installation angles errors. For the calculated USBL installation angles error, the target coordinates for the first and second USBL positions will be the same.

Let us assume that at the first and second USBL positions **the Heading, Pitch, Roll angles are equal to zero** (the vessel is located in the direction of the geographic coordinate axes); then the factors [M2_sh1] and [M2_sh2] can be taken equal to one. For given conditions, the distance between the recalculated target positions (hereinafter referred to as the discrepancy) can be found using the following MatLab code:

(a) Let's set the installation angles error to [1 0.5 0.8] degrees and calculate the target position in the ECS_USBL axes:

```
tr1=gNavTaitLever(-1,-0.5,-0.8,tr-sh1,'frw')+sh1
tr2=gNavTaitLever(-1,-0.5,-0.8,tr-sh2,'frw')+sh2
```

(b) Calculate the discrepancy for each combination of installation angles USBL:

```
r1=gNavTaitLever(dHu(n),dPu(n),dRu(n),tr1-sh1,'rev')+sh1;
r2=gNavTaitLever(dHu(n),dPu(n),dRu(n),tr2-sh2,'rev')+sh2;
dst(n)=sqrt(sum((r1-r2).^2));
```

where

r1 – coordinates of the target in GCS, at the first position of the USBL and with corrections to the USBL installation angles of dHu(n), dPu(n), dRu(n);

r2 – coordinates of the target in GCS, at the second position of USBL and with corrections to the USBL installation angles of dHu(n), dPu(n), dRu(n);

dst(n) – discrepancy (distance between "targets" in GCS, at the first and second position of USBL).

Let's perform calculations for a variety of installation angles dHu(n), dPu(n), dRu(n). To do this, set the range of change for all three installation angles from -3 to 3 degrees, in increments of 0.05 degrees, using the following MatLab code:

```
[dHu,dPu,dRu]=meshgrid(-3:.05:3,-3:.05:3,-3:.05:3);
```

For coordinates sh1=[0; 0; 0]; sh2=[120; 200; 0]; tr=[120; 60; -80] we obtain a three-dimensional array of dst discrepancies as a result of calculations. In Figure 2.2 the sections of the discrepancies array are displayed by planes [1 0.5 0.8]; all discrepancy values greater than 10 meters are assigned NaN (transparent color).

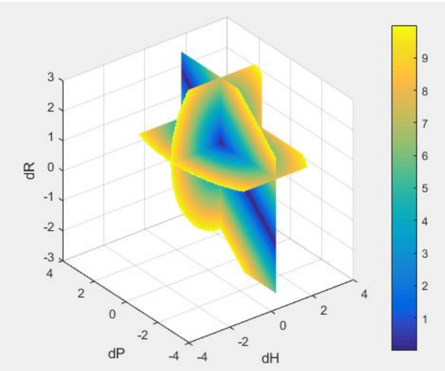


Figure 2.2 Sections of a discrepancies array by planes [1 0.5 0.8] with errors in the USBL installation angles equal to [1 0.5 0.8]

In the dRu&dPu plane there is a solution uncertainty - the set of dRu&dPu pairs correspond to small discrepancy values in fractions of millimeters, which "line up". This line is the major axis of the "discrepancy ellipsoid" (a three-dimensional figure similar in shape to an ellipsoid), on which the solution is weakly conditioned (Figure 2.3).

In this case, the exact solution with a discrepancy "equal" to zero will be unique – [1 0.5 0.8].

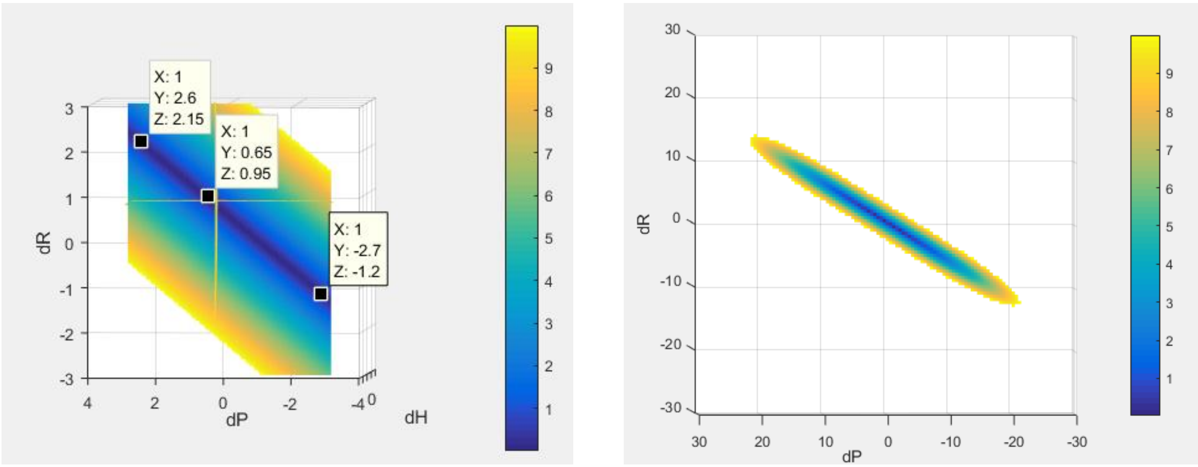


Figure 2.3 Displaying the major axis of the discrepancy ellipsoid

At a qualitative level, the appearance of the ellipsoid major axis can be explained as follows. For two USBL (vessel) positions in GCS, two spheres with a radius of “distance to USBL” can be drawn. The spheres will intersect along a circle, and due to a small change in the coordinates of the vessels along the Z axis, the circle will lie in a plane almost parallel to the Z axis. Further, within certain limits, we can correct dRu&dPu (errors in determining the USBL inclination angles) so that the target coordinates will move close to the circle without much deviation. The area of such movement will form the “major axis” on the dRu&dPu plane.

Let us calculate the position of targets for three “small discrepancies” selected on the dRu&dPu plane (shown in Figure 2.3 by markers).

Recalculated target coordinates at first USBL position sh1=[0; 0; 0]	**Recalculated target coordinates at the second USBL position sh2=[120; 200; 0]**	**XYZ distance between recalculated target coordinates**
Correction of discrepancies: dHu=0;dPu=0;dRu=0 tr1=gNavTaitLever(-1,-0.5,-0.8,tr-sh1,'frw')+sh1 tr2=gNavTaitLever(-1,-0.5,-0.8,tr-sh2,'frw')+sh2		
tr1 = 121.722382236441 56.7887438491782 -79.7414586944469	tr2 = 118.254878973433 58.9177376424443 -78.0560683142346	tr1-tr2 = 3.46750326300841 -2.12899379326615 -1.68539038021223
Correction of discrepancies: dHu=1;dPu=2.6;dRu=2.15 r1=gNavTaitLever(1,2.6,2.15,tr1-sh1,'rev')+sh1 r2=gNavTaitLever(1,2.6,2.15,tr2-sh2,'rev')+sh2		
r1 = 117.018361302192 61.7919922944682 -82.9967035877334	r2 = 116.888440095016 61.8700462180799 -83.129020583442	r1-r2 = 0.129921207176096 -0.0780539236116411 0.132316995708592
Correction of discrepancies: dHu=1;dPu=0.65;dRu=0.95 r1=gNavTaitLever(1,0.65,0.95,tr1'-sh1,'rev')+sh1' r2=gNavTaitLever(1,0.65,0.95,tr2'-sh2,'rev')+sh2'		
r1 = 119.788296878018 60.2028040226261 -80.1647448625456	r2 = 119.782783016295 60.2062072630099 -80.3594943301005	r1-r2 = 0.00551386172379864 -0.0034032403837756 0.194749467554885
Correction of discrepancies: dHu=1;dPu=-2.7;dRu=-1.2 r1=gNavTaitLever(1,-2.7,-1.2,tr1-sh1,'rev')+sh1 r2=gNavTaitLever(1,-2.7,-1.2,tr2-sh2,'rev')+sh2		
r1 = 124.417955281592 57.2864277532088 -75.0895305540264	r2 = 124.283714217508 57.3670337868774 -75.0831987980597	r1-r2 = 0.134241064084875 -0.0806060336685874 -0.00633175596675528
Correction of discrepancies: dHu=1;dPu=0.5;dRu=0.8 r1=gNavTaitLever(1,0.5,0.8,tr1-sh1,'rev')+sh1 r2=gNavTaitLever(1,0.5,0.8,tr2-sh2,'rev')+sh2		
r1 = 120 60 -80	r2 = 120 59.9999999999999 -80	r1-r2 = -1.4210854715202e-14 6.3948846218409e-14 -1.4210854715202e-14

The presence of the “discrepancy ellipsoid major axis” makes it impossible to find an exact analytical solution to the system of equations for determining the USBL installation angles for two vessel positions. Such a solution will obviously be weakly conditioned - the angles will change by tens of degrees with a small error in the initial data. To take into account the “solution uncertainty”, it is necessary to analyse all values of the three-dimensional array of dst discrepancies.

2.6 Determination of USBL installation angles error when fixing target coordinates for two vessel positions with Heading, Pitch, Roll not equal to zero

When the Heading, Pitch, Roll angles are not equal to zero, the inclination angles and the size of the discrepancy ellipsoid axes will change depending on the heading and pitching angles of the vessel.

Let's perform the calculation for coordinates sh1=[0; 0; 0]; sh2=[120; 200; 0]; tr=[120; 60; -80], while the values of Heading, Pitch, Roll will be set in various combinations for the first and second positions of the USBL (vessel). We will set the error of the installation angles to [1 0.5 0.8] degrees. For calculations we use the following MatLab code:

(a) Let us recalculate the position of the targets in the ECS_USBL axis:

```
tmp1=gNavTaitLever(-Head1,-Pitch1,-Roll1,tr-sh1,'frw');
tr1=gNavTaitLever(-1,-0.5,-0.8,tmp1,'frw')+sh1;
tmp2=gNavTaitLever(-Head2,-Pitch2,-Roll2,tr-sh2,'frw');
tr2=gNavTaitLever(-1,-0.5,-0.8,tmp2,'frw')+sh2;
```

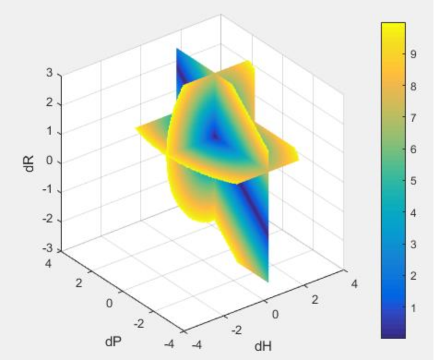
(b) Set the range for changing the installation angles from -3 to 3 degrees, in increments of 0.05 degrees:

```
[dHu,dPu,dRu]=meshgrid(-3:.05:3,-3:.05:3,-3:.05:3);
```

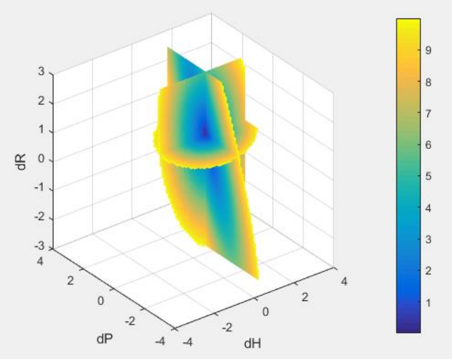
(c) Calculate the discrepancy for each combination of USBL installation angles:

```
tmp1=gNavTaitLever(dHu(n),dPu(n),dRu(n),tr1-sh1,'rev');
r1=gNavTaitLever(Head1,Pitch1,Roll1,tmp1,'rev')+sh1;
tmp2=gNavTaitLever(dHu(n),dPu(n),dRu(n),tr2-sh2,'rev');
r2=gNavTaitLever(Head2,Pitch2,Roll2,tmp2,'rev')+sh2;
dst(n)=sqrt(sum((r1-r2).^2));
```

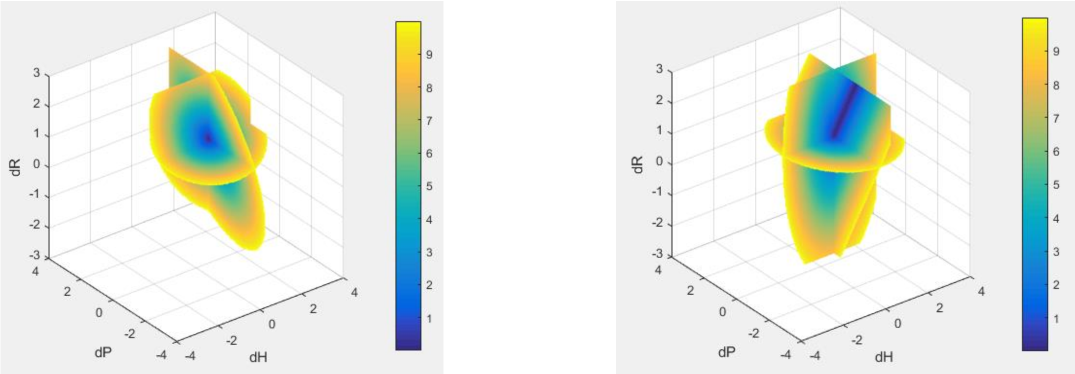
As a result of the calculations, we obtain a three-dimensional array of dst discrepancies. In Figure 2.4 sections of two discrepancies arrays by planes [1 0.5 0.8] are displayed; all discrepancy values greater than 10 meters are assigned as NaN (transparent color). The first array - with all Heading, Pitch, Roll equal to zero is shown earlier in Figure 2.2.



a) sh1: Head1=0, Pitch1=0, Roll1=0;
sh2: Head2=0, Pitch2=0, Roll2=0;



b) sh1: Head1=0, Pitch1=0, Roll1=0;
sh2: Head2=60, Pitch2=3, Roll2=5;



c) sh1: Head1=-60, Pitch1=-3, Roll1=-5;
sh2: Head2=60, Pitch2=3, Roll2=5;

d) sh1: Head1=-89, Pitch1=0, Roll1=0;
sh2: Head2=-89, Pitch2=0, Roll2=0;

Figure 2.4 Sections of discrepancy arrays by planes [1 0.5 0.8] with errors in installation angles USBL equal to [1 0.5 0.8]

2.7 Summation of discrepancy arrays for USBL installation angles at three vessel positions

Let the USBL be installed at the metacenter of the vessel (no LeverUSBL offset) with zero installation angles. Let three USBL positions in GCS be given – sh1=[0; 0; 0]; sh2=[120; 200; 0]; sh3=[200; 0; 0]; true target coordinates in GCS – tr=[120; 60; 80]; USBL installation angles error [1 0.5 0.8]. Let us assume that for all three USBL positions **the Heading, Pitch, Roll angles are equal to zero** (the vessel is located in the direction of the geographic coordinate axes).

Let's calculate arrays of dst discrepancies for USBL position combinations sh1&sh2, sh1&sh3, sh2&sh3.

For sh1=[0; 0; 0]; sh2=[120; 200; 0] sections of the discrepancies array by planes [1 0.5 0.8] are shown in Figure 2.5.

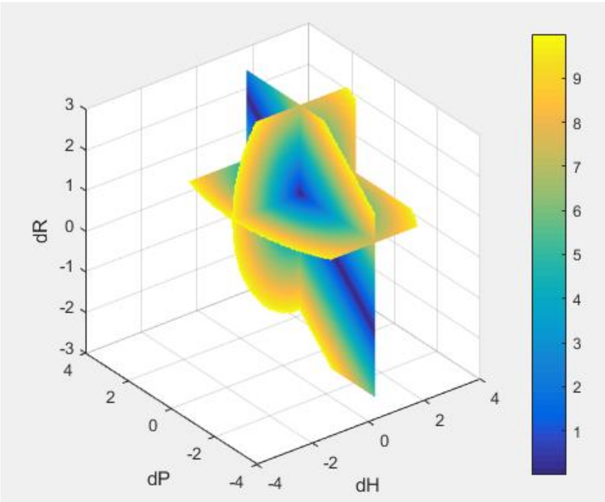


Figure 2.5 Sections of a discrepancies array by planes [1 0.5 0.8] at USBL sh1&sh2 positions

For sh1=[0; 0; 0]; sh3=[200; 0; 0] sections of the discrepancies array by planes [1 0.5 0.8] are shown in Figure 2.6. When the vessels are positioned along the X axis, the major axis of the ellipse is oriented along the change in dRu (perpendicular to the X axis), which corresponds to movement along the line of contact of the spheres “distance to USBL” (check description in Section 2.5).

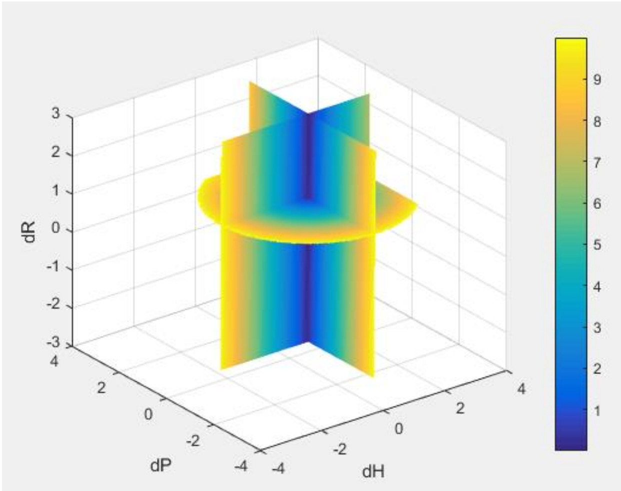


Figure 2.6 Sections of a discrepancies array by planes [1 0.5 0.8] at USBL sh1&sh3 positions

For sh2=[120; 200; 0];sh3=[200; 0; 0] sections of the discrepancies array by planes [1 0.5 0.8] are shown in Figure 2.7.

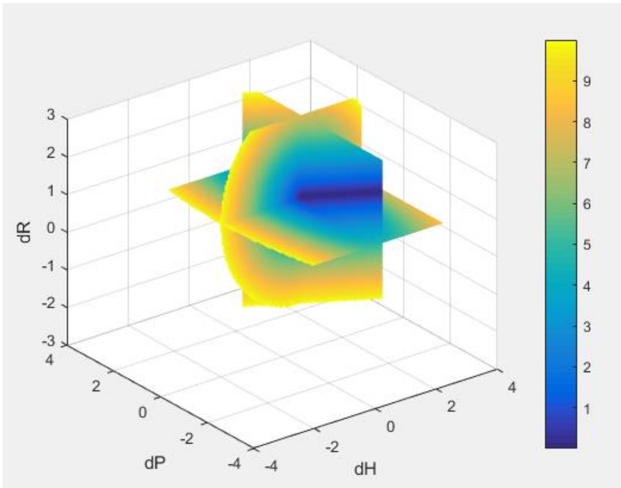


Figure 2.7 Sections of a discrepancies array by planes [1 0.5 0.8] at USBL sh2&sh3 positions

By adding all three arrays of discrepancies, we reduce the uncertainty of finding the minimum discrepancy (Figure 2.8):

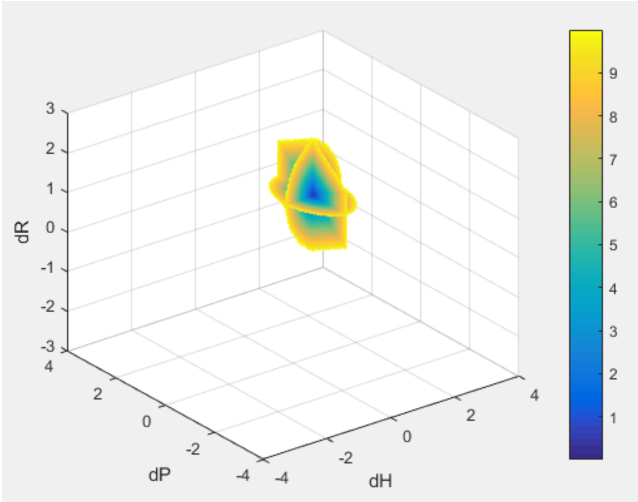
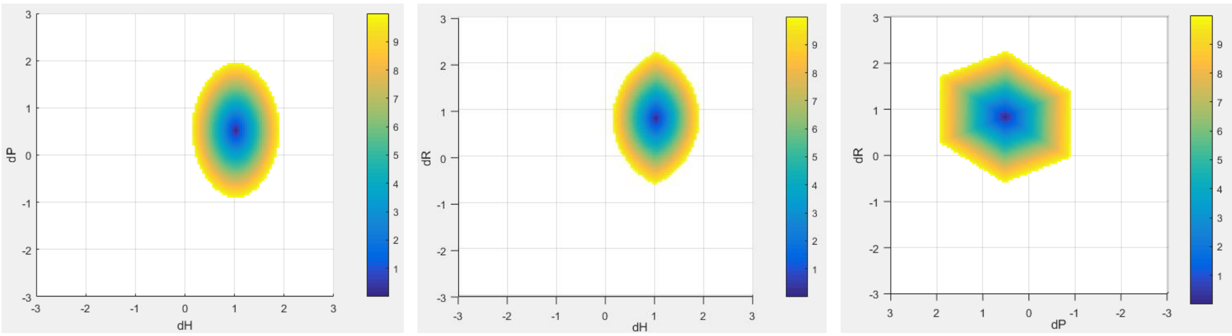


Figure 2.8 Sections of the total discrepancies array by planes [1 0.5 0.8]

In Figure 2.9 sections of the total discrepancies array are shown in each plane separately.



dRu=0.8 dPu=0.5 dHu=1

Figure 2.9 Section planes of the total discrepancies array at dRu=0.8, dPu=0.5, dHu=1

Perhaps, in the presence of errors other than the USBL installation angles (USBL error in determining the distance to the target, error in determining the position of the vessel's metacenter), the distribution pattern of small discrepancies may be more complex than the “discrepancy ellipsoid”.

2.8 Summary of Section 2

1)The main objective of section 2 was Estimation of USBL installation angles error based on target coordinates on the bottom.

2)A mathematical model of the first approximation (Model-1) for converting target coordinates from ECS_USBL to GCS-META (Figure 2.1) was considered. The model contains USBL, MRU and “vessel hull” (VCS axes). For simplicity, it was assumed in all calculations that the USBL is installed at the metacenter of the vessel (there is no LeverUSBL arm).

The model uses axes of: ECS_USBL, VCS, GCS-META. The target coordinates were recalculated by sequential rotation of the previous axes parallel to the subsequent ones; rotations can be performed as forward (Heading-Pitch-Roll) or reverse (Roll-Pitch-Heading).

The order of rotation of the ECS_USBL, VCS axes is fixed (when the order of rotations is changed, the angles of rotation of the axes actually change, and an incorrect calculation result is obtained).

3)To determine the coordinates of the tr target in the ECS_USBL axes when introducing an error in the installation angles of +2 degrees relative to the VCS, the following MatLab code is used:

```
tmp2=gNavTaitLever(-2,-2,-2,tr,'frw')
```

To recalculate the coordinates of the tr target from the ECS_USBL axes to the VCS axes when compensating for the error of the USBL installation angles (error +2 degrees), the following MatLab code is used:

```
tmp1=gNavTaitLever(2,2,2,tr,'rev')
```

4)As the measure for assessing the “correctness of calculation” of USBL installation angles the distance between the “calculated target coordinates after introducing corrections to the USBL installation angles” has been introduced; in this case, calculations are made for the same target on the bottom at two positions of the vessel (USBL).

The term array of discrepancies (dst) was introduced, calculated from a given set of installation angles (dHu, dPu, dRu).

It is shown that for Model-1, when evaluating the USBL installation angles, we get an ellipsoid of discrepancies (a three-dimensional figure close to an ellipsoid in shape) on the axes dHu, dPu, dRu. The central part of the ellipsoid contains values with small discrepancies. With an “elongated” ellipsoid shape, small discrepancies can be obtained for combinations of installation angles when these angles change by tens of degrees (Figure 2.3).

The shape of the discrepancy ellipsoid depends on (1) the relative position of the target on the bottom and the two vessel positions (USBL) (for which the discrepancy is calculated), (2) the Heading, Pitch, Roll angles for each of the vessel positions (Figure 2.4).

5)The term total array of discrepancies was introduced, which is calculated for several positions of the vessel when fixing one target (summing is also possible when fixing different targets). When summing the discrepancies, the uncertainty in finding the minimum discrepancy can decrease (Figure 2.8).

3 Model-2: USBL and MRU installation angles, heading and pitching

3.1 Description of Model-2

The direction of the VCS axes during operation of navigation equipment is often determined by the MRU; it is assumed that the direction of the ECS_MRU axes coincides with the VCS. However, in practice, the MRU is installed with "MRU installation angles error".

Let's consider a model that contains USBL and MRU. The USBL is installed with an error in the installation angles [dHu, dPu, dRu] relative to the VCS or with an error [dH, dP, dR] **relative to the MRU axes**. The MRU is installed with an error in the installation angles [dHm dPm dRm] relative to the VCS. MRU shows the values of the heading and pitch angles (Heading, Pitch, Roll) relative to its coordinate axes (ECS_MRU). The coordinates of the tr target are determined by USBL in the ECS-USBL axes; it is necessary to calculate the target coordinates in the GCS-META axes (Figure 3.1).

The error in the installation angles of the MRU (relative to the axes of the vessel hull) will be those values dHm, dPm, dRm that the MRU will show when installing the corresponding axes of the vessel hull in the north and plumb directions. Drawing physical analogies, we can consider it as if the MRU readings are taken from “measuring dials” installed on the MRU hull (positive values are located on the left side of the dial). In this case, all further MRU measurements will be issued in the “system of rotated measuring dials”. Rotations of the MRU to compensate for errors in installation angles will be performed in the sequence dRm, dPm, dHm.

Since MRU readings are given relative to the ECS_MRU axes, it is convenient to refer to the ECS_MRU-META axes rather than the VCS to the GCS-META axes. Then "incorrect installation of the MRU" can be considered as “incorrect installation of the vessel hull relative to the MRU”. In this case, the USBL installation angles (ECS_USBL) must be evaluated relative to the ECS_MRU axes, and the “incorrectly installed hull” will affect the equipment offsets. The LeverUSBL offset will have to be recalculated from the VCS coordinate axes to the ECS_MRU coordinate axes.

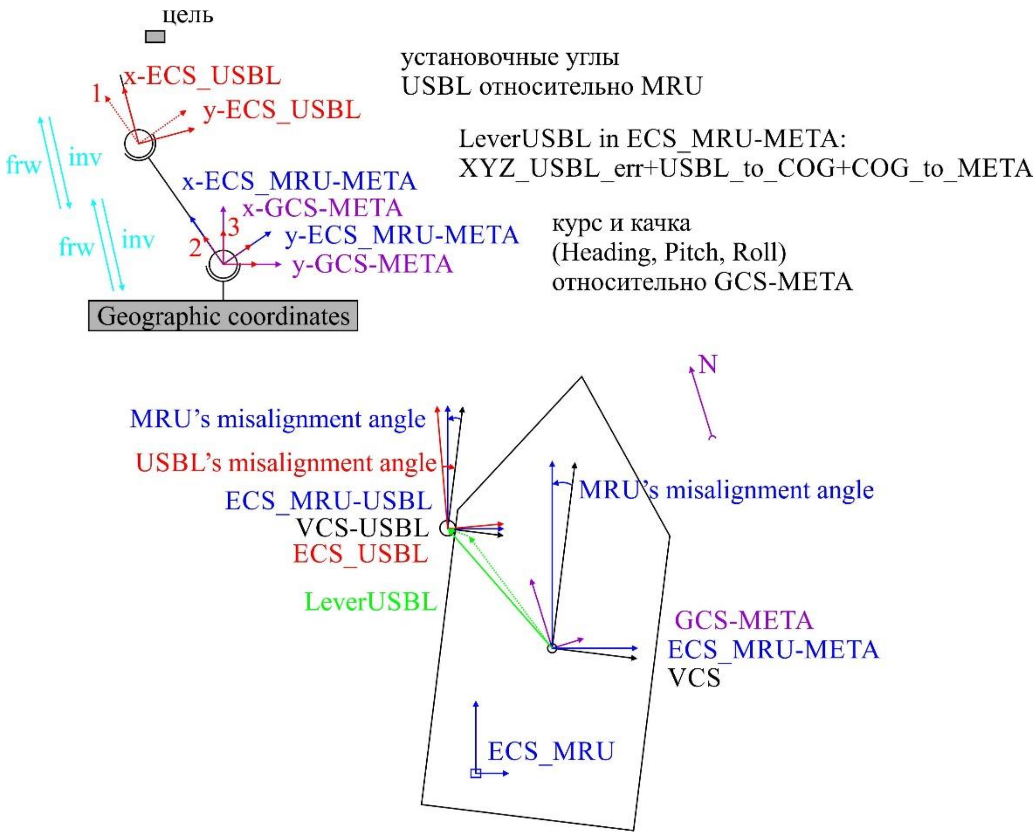


Figure 3.1 Model-2, shifts and rotations of coordinate axes

Let us describe the sequence of rotations-shifts shown in Figure 3.1:

-- the target is located in the ECS_USBL coordinate axes (axes with errors in the USBL installation angles);

-- rotate the ECS_USBL axes to the USBL installation angles error **relative to the MRU axes**, obtaining axes 1 (these axes are located in the direction of the ECS_MRU axes);

-- shift axes 1 (USBL installation point) to the ECS_MRU-META axes (vessel metacenter), we get axes 2; the displacement value must be calculated in the ECS_MRU axes, and not the VCS (to do this, it is necessary to know the error of the MRU installation angles relative to the vessel hull in order to calculate the projection of the offsets on the ECS_MRU);

-- as a result of rotation and displacement of the axes, we obtain the target coordinates not in the ECS_USBL axes (USBL transducer with errors in the installation angles), but in the ECS_MRU-META axes (metacenter of the vessel with axes according to MRU);

-- we rotate axes 2 to the heading and pitching angles (Roll, Pitch, Heading) to the GCS-META axes (the direction of the geographic coordinate axes), we get axes 3;

-- as a result of two rotations and one displacement, we obtained the target coordinates in axes direction of which coincides with the direction of the geographic coordinate axes, and the origin is located in the metacenter of the vessel.

The described model is implemented as a sequential multiplication of the coordinate matrix by the rotation matrix:

tr_GCS-META = [M_rev(Head,Pitch,Roll)]*[M_rev(dH,dP,dR)+LeverUSBL]*tr_ECS-USBL

tr_ECS-USBL = [M_frw(-dH,-dP,-dR)]*[M_frw(-Head,-Pitch,-Roll)-LeverUSBL]*tr_GCS-META

The LeverUSBL offset is used to set the USBL offset in the current ECS_MRU-META coordinate axes.

To calculate the target coordinates during the sequence of rotations and shifts shown in Figure 3.1, to move from GCS-META to ECS_USBL you need to perform calculations using the following MatLab code:

(a) recalculate the coordinates of the tr target from the GCS-META axes to the ECS_MRU-META axis

```
tmp1=gNavTaitLever(-Head,-Pitch,-Roll, tr,'frw')
```

(b) shift the axes from the metacenter (META) to the USBL installation point

```
tmp2=tmp1-LeverUSBL
```

in this case, the LeverUSBL value is entered in the ECS_MRU-META axes

(c) recalculate the target coordinates from the ECS_MRU-USBL axes to the ECS_USBL axis (this implies entering the error of the USBL installation angles exactly “relative to the ECS_MRU-META axes”)

```
tr1=gNavTaitLever(-dH,-dP,-dR,tmp2,'frw')
```

where tr1 are the target coordinates “according to USBL data” (on the ECS_USBL axes).

To calculate the target coordinates during the sequence of rotations and shifts shown in Figure 3.1, to move from ECS_USBL to GCS-META you need to perform calculations using the following MatLab code:

(c)`tmp2=gNavTaitLever(dH,dP,dR,tr1,'rev')`

(b)`tmp1=tmp2+LeverUSBL`

(a)`tr2=gNavTaitLever(Head,Pitch,Roll,tmp1,'rev')`

where tr2 are the target coordinates in the GCS-META coordinate system.

3.2 Extended description of Model-2

For a more complete description of Model-2, it is necessary to consider two issues: (1) recalculation of LeverUSBL into ECS_MRU-META; (2) rotation of the ECS_USBL axes in the direction of ECS_MRU.

3.2.1 Recalculation of USBL lever from VCS to ECS_MRU-META

Let the beginning of the LeverUSBL vector be at the metacenter, and the end at the point with the coordinates of the USBL installation. In this case, the LeverUSBL lever can be considered as consisting of the following components:

-- error in determining the USBL installation point,

-- offset between the metacenter and the USBL installation point,

All components are initially tied to the vessel hull with VCS-META axles. Model-2 does not consider META-COG displacement when the vessel metacenter is incorrectly determined.

Let's determine the coordinates of the LeverUSBL vector specified in the VCS on the ECS_MRU-META axes with an error in the installation angles of +2 degrees relative to the VCS. To do this, let’s “rotate” the VCS axes to the MRU installation angles with the “fixed” LeverUSBL vector, using the following MatLab code:

```
LeverUSBL_ECS_MRU=gNavTaitLever(-2,-2,-2,LeverUSBL_VCS,'frw')
```

or

LeverUSBL_ECS_MRU=gNavTaitLever(-dHm,-dPm,-dRm,LeverUSBL_VCS,'frw')

3.2.2 Conversion of target coordinates from ECS_USBL to ECS_MRU direction via VCS

When rotating the ECS_USBL axes to the direction of the ECS_MRU axes, there is no semantic connection for the rotation angles to the installation angles of the equipment. It is more expedient to perform this rotation through the direction of the vessel hull axes (VCS). In this case, we will be able to operate separately on the “USBL installation angles relative to the vessel axes” and “installation angles MRU relative to the vessel axes.”

Let us divide the rotation of the ECS_USBL axes to the direction of the ECS_MRU axes into two actions:

1) Let's recalculate the target coordinates from the ECS_USBL axes to the VCS-USBL axes. When compensating for the error of the USBL installation angles (error +2 degrees), the ECS_USBL axes are rotated as 'rev' (reverse – Roll-Pitch-Heading) to the direction of the VCS axes:

tmp1=gNavTaitLever(2,2,2,tr0,'rev')

or

tmp1=gNavTaitLever(dHu,dPu,dRu,tr0,'rev')

2) Determine the target coordinates in the ECS_MRU axes when introducing an error in the MRU installation angles of +2 degrees relative to the direction of the VCS axes. To do this, let’s “turn” the VCS-USBL axes to the MRU installation angles using the following MatLab code:

tmp2=gNavTaitLever(-2,-2,-2,tmp1,'frw')

or

tmp2=gNavTaitLever(-dHm,-dPm,-dRm,tmp1,'frw')

3.2.3 MatLab code for Model-2

Taking into account the supplemented description for recalculating target coordinates from ECS_USBL to GCS-META (with a sequence of rotations-shifts shown in Figure 3.1) perform calculations using the following MatLab code:

0) recalculation of LeverUSBL lever from VCS to ECS_MRU-META

LeverUSBL=gNavTaitLever(-dHm,-dPm,-dRm,LeverUSBL_VCS,'frw')

1) USBL installation angles error compensation (rotate ECS_USBL to VCS-USBL)

tmp1=gNavTaitLever(dHu,dPu,dRu,tr0,'rev')

2) entering an error into the MRU installation angles (rotating VCS-USBL to ECS_MRU)

tmp2=gNavTaitLever(-dHm,-dPm,-dRm,tmp1,'frw')

3) offset of the origin from USBL to META in the ECS_MRU-USBL coordinate axes

tmp3=tmp2+LeverUSBL

4) taking into account MRU readings (rotate ECS_MRU-META to GCS-META)

tr02=gNavTaitLever(Head,Pitch,Roll,tmp3,'rev')

where tr02 are the target coordinates in the GCS-META coordinate system.

To recalculate target coordinates from GCS-META to ECS_USBL (with a sequence of rotations-shifts shown in Figure 3.1) perform calculations using the following MatLab code:

0) recalculation of LeverUSBL lever from VCS to ECS_MRU-META

```
LeverUSBL=gNavTaitLever(-dHm,-dPm,-dRm,LeverUSBL_VCS,'frw')
```

4) taking into account MRU readings (rotate GCS-META to ECS_MRU-META)

```
tmp3=gNavTaitLever(-Head,-Pitch,-Roll,tr02,'frw')
```

3) offset of the origin from META to USBL, in the ECS_MRU-META coordinate axes

```
tmp2=tmp3-LeverUSBL
```

2) compensation for errors in MRU installation angles (rotate ECS_MRU to VCS-USBL)

```
tmp1=gNavTaitLever(dHm,dPm,dRm,tmp2,'rev')
```

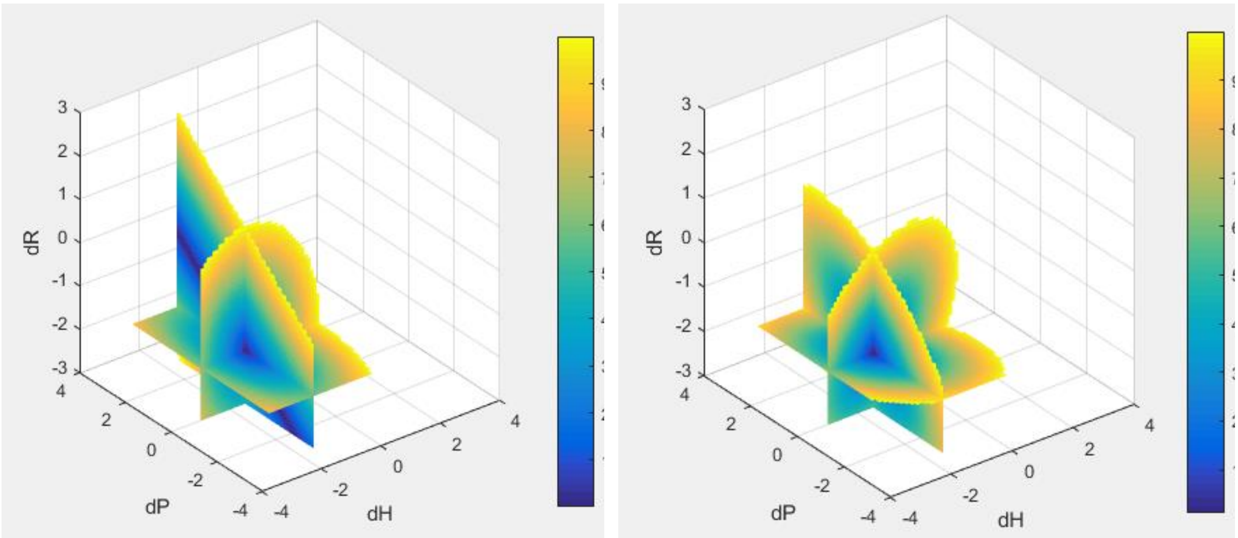
1) entering an error in USBL installation angles (rotating VCS-USBL to ECS_USBL)

```
tr2=gNavTaitLever(-dHu,-dPu,-dRu,tmp1,'frw')
```

3.2.4 Effect of the error in MRU installation angles

Let's perform the calculation for coordinates sh1=[0; 0; 0]; sh2=[120; 200; 0]; tr=[120; 60; -80], while the values of Heading, Pitch, Roll will be set in various combinations for the first and second positions of the USBL (vessel). We will set the USBL installation angle error to [-0.5 0.5 -1] degrees. The value of LeverUSBL_VCS is set to [0; 0; 0] meters. We will set the error of the MRU installation angles to [1;0.5;0.8] degrees; in the calculations we will use the “erroneous” value [0;0;0] degrees.

The calculation results are shown in Figure 3.2. As can be seen from the figure, the shape of the discrepancy ellipsoid (for combinations of angles Heading, Pitch, Roll) is approximately preserved relative to Figure 2.4. The minimum discrepancy value is shifted to **approximately** [-1.5 0 -1.8]=[-0.5 0.5 -1]-[1 0.5 0.8] degrees (**approximately** the USBL installation angles error minus the MRU installation angles error).



a) sh1: Head1=0, Pitch1=0, Roll1=0;
sh2: Head2=0, Pitch2=0, Roll2=0;
LeverUSBL_VCS=[0;0;0];
for dst calculation LeverUSBL_VCS=[0;0;0];

b) sh1: Head1=0, Pitch1=-3, Roll1=-5;
sh2: Head2=60, Pitch2=3, Roll2=5;
LeverUSBL_VCS=[0;0;0];
for dst calculation LeverUSBL_VCS=[0;0;0];

Figure 3.2 Sections of discrepancy arrays by planes [-1.5 0 -1.8] with errors in USBL installation angles equal to [-0.5 0.5 -1] and MRU equal to [1 0.5 0.8]; for calculations, an error was entered into the MRU installation angles

Approximate equality is a consequence of small values of the discrepancy angles. With an accurate calculation (taking Heading, Pitch, Roll equal to zero):

```
tmp1=gNavTaitLever(1,0.5,0.8,tr-sh1,'rev');
tr1=gNavTaitLever(-(-0.5),-0.5,-(-1),tmp1,'frw')+sh1
```

we get the target coordinates converted from GCS-META to ECS_USBL

```
tr1 =
118.359476949244
65.5851230982138
-78.036054772736
```

With an approximate calculation for [-1.5 0 -1.8]=[-0.5 0.5 -1]-[1 0.5 0.8], we obtain the following approximate target coordinates, recalculated from GCS-META to ECS_USBL:

```
gNavTaitLever(-(-1.5),-0,-(-1.8),tr-sh1,'frw')+sh1
ans =
118.388262098594
65.6023877797828
-77.977856567525
```

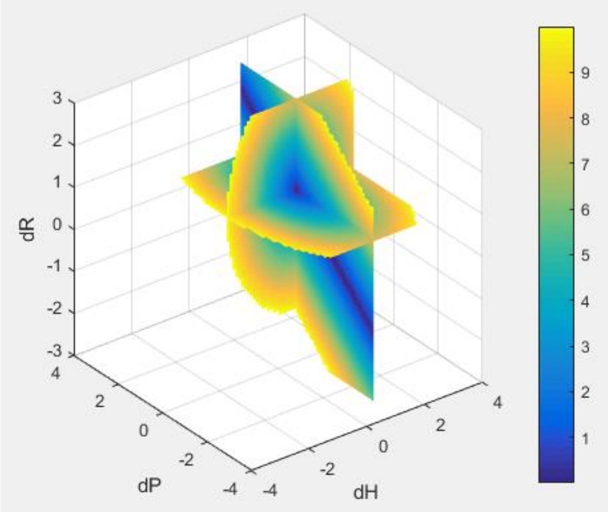
The difference in coordinates, for the conditions in the example given, will be centimeters:

```
tr1-ans
ans =
-0.0287851493501989
-0.0172646815689603
-0.0581982052109566
```

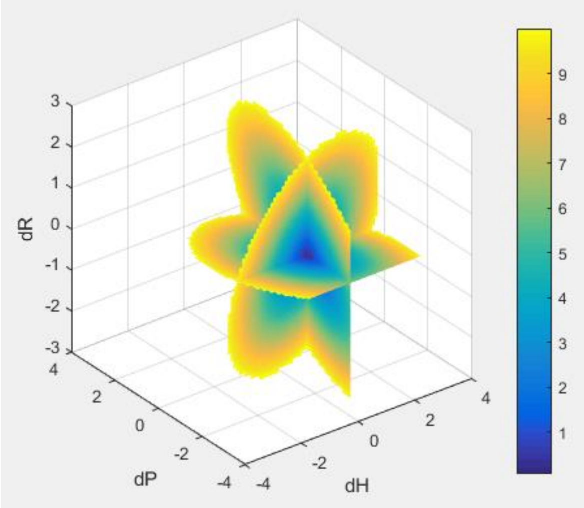
3.2.5 *Effect of the error in LeverUSBL lever length*

Let's perform the calculation for coordinates sh1=[0; 0; 0]; sh2=[120; 200; 0]; tr=[120; 60; -80], while the values of Heading, Pitch, Roll will be set in various combinations for the first and second positions of the USBL (vessel). We will set the USBL installation angles error to [1 0.5 0.8] degrees; We will set the MRU installation angles error to [1;0.5;0.8] degrees. Initially, the value of LeverUSBL_VCS is set to 3 meters; in the calculations we will use the “erroneous” lever value of the 0 meters.

The calculation results are shown in Figure 3.3.



a) sh1: Head1=0, Pitch1=0, Roll1=0;
sh2: Head2=0, Pitch2=0, Roll2=0;
LeverUSBL_VCS=[3;0;0];
for dst calculation LeverUSBL_VCS=[0;0;0];



b) sh1: Head1=-60, Pitch1=-3, Roll1=-5;
sh2: Head2=60, Pitch2=3, Roll2=5;
LeverUSBL_VCS=[3;0;0];
for dst calculation LeverUSBL_VCS=[0;0;0];

Figure 3.3 Sections of discrepancy arrays by planes (a) [0.5 -1 -0.1], (b) [1 0.5 0.8]; an error was entered in the length of the USBL lever - 3 meters along the X axis

As can be seen from the figure, the shape of the discrepancy ellipsoid (for combinations of Heading, Pitch, Roll angles) is approximately preserved relative to Figure 2.4.

If the vessels (USBL) are co-directed, then the errors entered into the tr target coordinates due to the incorrect LeverUSBL value are the same. The calculation of the discrepancy between the USBL installation angles under such conditions is performed correctly (Figure 3.3, a).

The magnitude of the error (calculating the discrepancy between the USBL installation angles) increases with the increase in the difference in USBL directions at two vessel positions. So, in the above example, with an error in LeverUSBL of 3 meters, the error in determining the discrepancy is [-0.5 -1.5 -0.9] degrees.

Thus:

-- the effect of the error in the MRU installation angles “transfers” the minimum value in the dst array of discrepancies (discrepancies for the installation angles USBL) to a new constant point for various combinations of Heading, Pitch, Roll,

-- effect of LeverUSBL length error moves the minimum in the dst discrepancies array for each Heading, Pitch, Roll combination.

Assuming that the MRU installation angles are directed along the vessel hull (the error in the MRU installation angles is [0; 0; 0]), having found the minimum value in the dst array of discrepancies (for USBL installation angles) - we obtain “USBL installation angles relative to the axes of the vessel hull". In fact, the USBL installation angles will be determined relative to the ECS_MRU axes.

In this case, errors in the LeverUSBL projection onto ECS_MRU (since the actual error in the MRU installation angles is not zero or is not fully compensated for in the navigation system

corrections) will move the minimum in the dst array of discrepancies for each combination of Heading, Pitch, Roll and create a “scatter of minimums” in total array of discrepancies.

3.2.6 Special case of Model-2: MRU and USBL installation angles errors are equal

If you select the USBL or MRU installation angles error so that the ECS_USBL and ECS_MRU axes coincide in direction (Figure 3.4), then rotations by [dHu,dPu,dRu] and [dHm,dPm,dRm] angles (check MatLab code for Model-2) compensate each other. After this, the “target coordinate calculation error” will be determined only by the LeverUSBL lever calculation error.

The LeverUSBL error, in turn, will be determined by the magnitude of the MRU installation angles error relative to the vessel hull (VCS); in Figure 3.4 the dotted line shows the position of LeverUSBL and the position of USBL at zero value of the MRU and USBL installation angles error (in this case, the vessel hull rotates, compensating for the installation angles error for MRU).

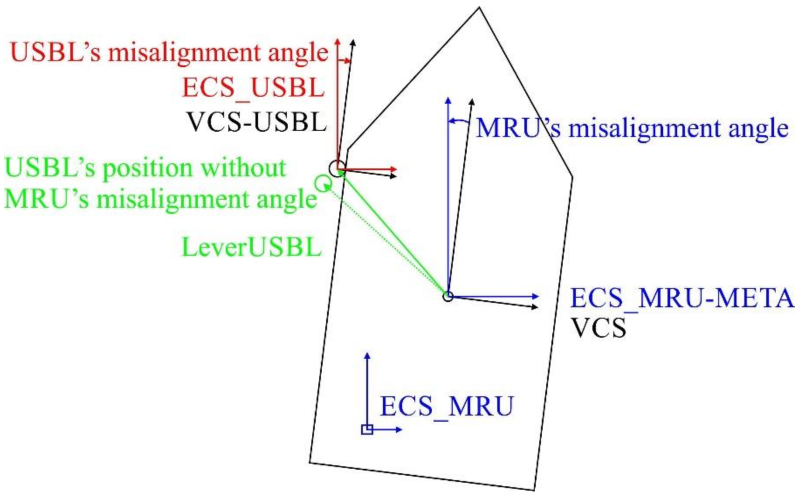


Figure 3.4 Model-2, installation angles error for MRU and USBL are equal

Thus, even for MRU installation angles error that deviate axes significantly from VCS, it is possible to select such “USBL installation angles compensation” angles that the ECS_USBL and ECS_MRU axes coincide in direction (Figure 3.4). Further increasing the “USBL positioning accuracy” by correcting the USBL installation angles will not be possible (or will be possible for a certain course, depth interval and distance to the target). The error will be determined by the impact of the MRU installation angles error on the LeverUSBL projection (on the ECS_MRU axes from the VCS axes), that is, a further increase of the “USBL positioning accuracy” will require correction of the MRU angles.

The value of discrepancy calculated for two positions of the vessel will depend on the combination of Heading, Pitch, Roll angles; the discrepancy will decrease for the same direction of vessels (USBL axes) and increase to a maximum for opposite directions (USBL axes).

3.3 Definition of a coordinate transformation graph

Based on Model-2, it is possible to create a more complete model of vessel navigation. To do this, the following steps are required:

-- add Heave-sensor to the model;

-- add GPS antennas into the model in order to calculate the geographic coordinates of the metacenter in GCS relative to them (for two antennas you can calculate Heading, as well as other inclination angles when operating in RTK);

-- add separate META and COG terms into the model; where META is the metacenter relative to which the actual rotation of the vessel occurs; COG is the point relative to which equipment offsets are calculated and which is taken as the metacenter;

-- add a calculation of the towed devices position into the model based on the length of the etched cable, depth sensors of the towed device and other data not related to hydroacoustic positioning.

When expanding Model-2, not all relations between model components (“manipulator links”) will be described by rotation matrices and displacement of the origin. For example, a towed device can be represented in a first approximation as a “load on a rigid rod”, but subsequent approximations will require a different mathematical model. In this case, it is advisable to move from the “manipulator schematics” to **"coordinate transformation graph"**; In this case, processing will be performed according to the following rules:

-- each node of the graph will have a corresponding “transformation procedure” (in a particular case, this will be a rotation matrix of coordinate axes or a shift of the origin);

-- for each node of the graph there must be a “direct” transformation procedure and a “reverse” transformation procedure;

-- the transition from one graph node to another node must be performed along a given path on the graph (in the prescribed order, performing transformation procedures for each intermediate node).

An example of a transformation graph is shown in Figure 3.5. The graph is given for a typical engineering survey configuration, in which simultaneous bottom relief surveying (MBES), seismic profiling (SBP), and towed deep-sea survey (SSS) and magnetometer surveys are performed. In this case, the following equipment is used to perform the survey:

-- the vessel is equipped with MBES, GPS, MRU&Heave, Draft sensor, Cable counter, USBL, which outputs the coordinates of the transponder beacon in the ECS_USBL and SBP (seismic profiler) axes;

-- a bundle of SSS and a magnetometer is towed through the Cable counter; in this case, the magnetometer is equipped with an altimeter and a depth sensor, and the SSS is equipped with an electronic compass-tilt meter and a depth sensor;

-- a USBL transponder beacon is attached to the towing cable of the bundle.

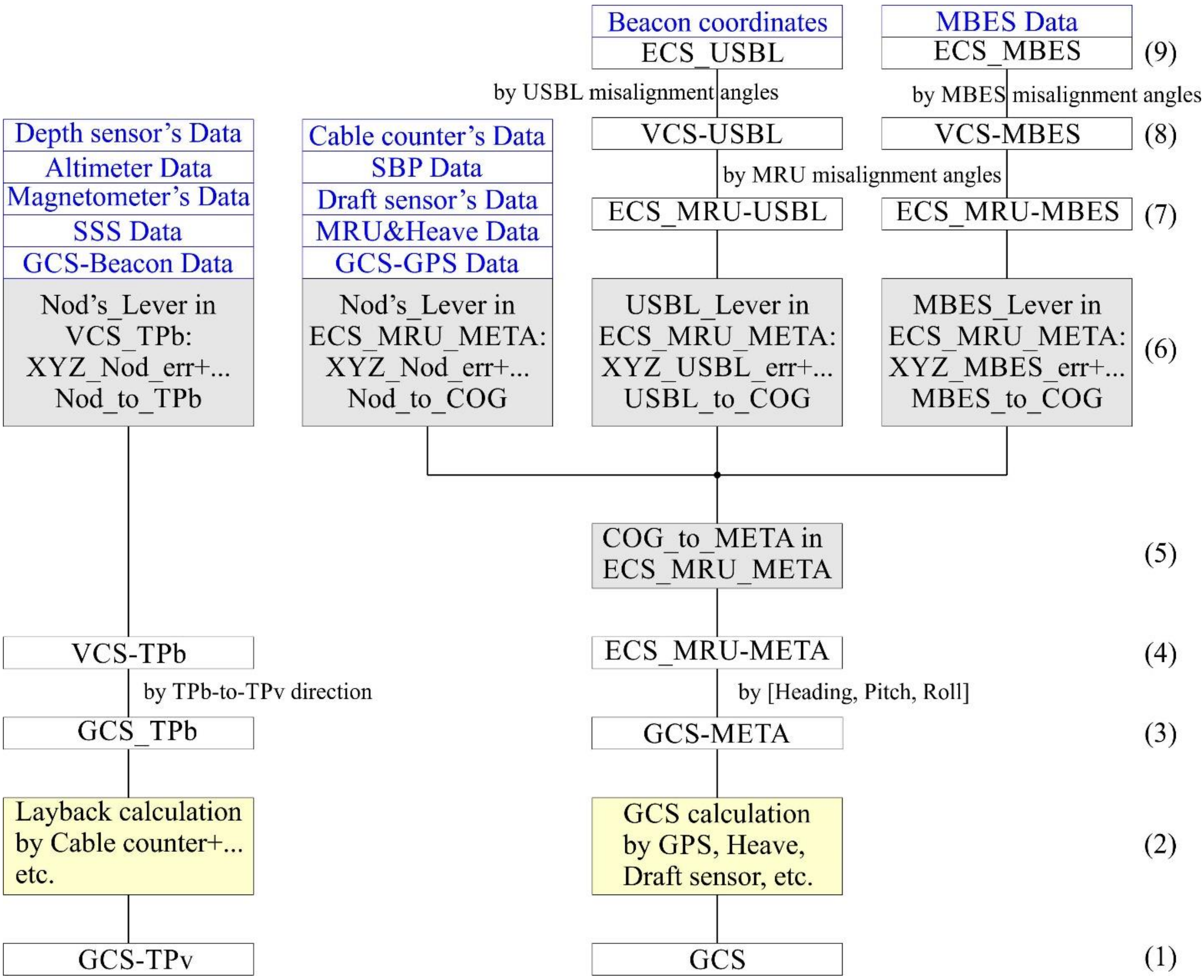


Figure 3.5 An example of a coordinate transformation graph for Model-2

The white rectangles on the graph correspond to the rotation of the coordinate system. Gray rectangles correspond to the origin of coordinates transfer from point to point. Yellow rectangles correspond to coordinate calculation procedures that differ from rotation and transfer of the origin.

Blue color shows data from sensors, which can also be used in "coordinate transformation". For example, knowing the GPS offset relative to the metacenter, we can recalculate it into the GCS-META axes using the [Heading, Pitch, Roll] angles and, further, based on GPS data, link the GCS-META axes to the GCS (due to this we can recalculate into GCS axes the position of any node or target the coordinates of which are known in GCS-META).

Blue color shows the data of the following equipment installed on the vessel: MBES, GPS, MRU&Heave, Draft sensor, Cable counter, SBP, USBL, which outputs the coordinates of the transponder beacon in the ECS_USBL axes.

On the left, the graph shows an example of a branch for SSS and a magnetometer towed in sequence (the magnetometer is towed in a piggy-back; the magnetometer is equipped with an altimeter and a pressure-depth sensor). Positioning is performed using a USBL beacon, controlled by the "Layback" algorithm, which uses the length of the etched cable and other navigation data. If the signal on USBL is lost, positioning will be performed according to the "Layback" algorithm (with filter coefficients selected during

USBL operation). Towed devices course determined by the direction of the vector "position of the USBL transponder beacon – position of the towing point after filtering"; heading control is performed using an electronic magnetic compass-tilt meter installed on the SSS.

Additional control will be performed:

-- by comparing the data of the altimeter and the pressure-depth sensor installed on the magnetometer (linked to coordinates) with the bottom topography according to MBES;

-- by comparing the coordinates of SSS targets on sonograms, when one target is detected on different tacks;

-- by comparing target coordinates on SSS sonograms and MBES data.

3.4 Summary of Section 3

1)A mathematical model (Model-2) is considered for converting target coordinates from ECS_USBL to GCS-META (Figure 3.1). The model contains USBL, MRU and "vessel hull" (VCS axes).

For Model-2, it was assumed that USBL was installed with LeverUSBL offset relative to the metacenter.

Besides USBL installation angles error, the MRU installation angles error are included into the Model-2.

All installation angles within the model are linked to the direction of the vessel hull axes (VCS).

It was shown, that:

-- the effect of the error in the installation angles MRU "transfers" the minimum value in the dst array of discrepancies (discrepancies for the USBL installation angles) to a new constant point for various combinations of Heading, Pitch, Roll,

-- effect of LeverUSBL length error moves the minimum in the dst discrepancies array for each Heading, Pitch, Roll combination.

2)A special case of Model-2 is considered, when MRU and USBL installation angles errors are equal. In this case, there is no component of the installation angles error, which increases with distance. The USBL positioning error will only depend on the incorrect assessment of the LeverUSBL lever position.

Consequently:

-- For any MRU installation angles error (even for angles significantly deviating from VCS), you can select such "USBL installation angles compensation" angles that the ECS_USBL and ECS_MRU axes coincide in direction (Figure 3.4). Further increasing of the "USBL positioning accuracy" by changing the USBL installation angles will not be possible (or will be possible for a certain course, depth interval and distance to the target).

-- The target positioning error (when the MRU and USBL installation angles errors are equal) will be determined by the influence of the MRU installation angles error on the LeverUSBL projection (on the ECS_MRU axes from the VCS axes), that is, further increasing of the "USBL positioning accuracy" will require correction of the MRU angles.

-- The amount of discrepancy calculated for two positions of the vessel (when the MRU and USBL installation angles errors are equal) will depend on the ratio of the Heading, Pitch, Roll angles (at the first and second position); the discrepancy will decrease for the same direction of vessels (USBL axes) and increase to a maximum for opposite directions (USBL axes).

3)The term coordinate transformation graph was introduced.

4 Conclusion

USBL positioning error can be divided into four components:

(a) Random hardware error. For this component, the positioning error increases with increasing distance to the target in accordance with the characteristics of the equipment.

(b) Systematic error of USBL and MRU installation angles. For this component, the positioning error increases with increasing distance to the target. The error is removed by correcting the USBL installation angles, so that they start to coincide with MRU installation angles error.

It is not possible to further increase the target positioning accuracy by correcting the USBL installation angles; MRU installation angles need to be adjusted.

When operating with two MRUs (or transitioning to a backup MRU), their installation angles relative to the vessel hull must be the same, or a separate search for "USBL installation angles" relative to the second MRU must be performed. Also, the occurrence of an error in the MRU readings due to the "operating mode" will create an error in the USBL readings.

(c) Systematic error of LeverUSBL and LeverGPS offsets. For this component, the positioning error will not increase with increasing distance to the target.

The error component will be close to a "permanent offset" relative to the vessel heading; the magnitude of the component will depend on the magnitude of the error in the MRU and USBL installation angles relative to the vessel hull. When passing over the target on one tack, the value of the discrepancy (in the position of the target on the tack) will be close to zero; when passing on opposite tacks, the discrepancy value (at the target position on tacks) will be maximum.

The error is removed by adjusting the MRU installation angles such that the MRU axes begin to coincide with the vessel hull axes (VCS) at which the LeverUSBL and LeverGPS offsets were measured.

(d) "META_to_COG lever bias" error, which is not considered within Model-2. If, with a systematic error of the LeverUSBL and LeverGPS (c) offsets, an error in the location of the "end of the offset vector" is considered, then with the "offset to the META_to_COG lever" there is an error in the position of the "beginning of the offset vector." Since, with this error, the roll and heading will be taken into account relative to the COG, but not the real metacenter of the vessel, the magnitude of the error will depend on the vessel heading and pitching angles.

All calculations in this document are performed using the gNavTaitLever function, ge0mlib libraries. The text of the function is given in Appendix 1.

Script for calculations according to Model-2 based on gNavTaitNLever, gNavTaitLeverN or gNavTaitLever functions is given in Appendix 2.

Reference list

1. https://www.xometry.com/resources/materials/what-is-brass/

Appendix 1. gNavTaitLever function text

```
functionXYZ_Lev=gNavTaitLever(Head,Pitch,Roll,Lever,RotDirect)
%A number LeverArm's coordinates calculation using scalar Heading,Pitch,Roll
        (3D Tait-Bryan matrixes rotation).
%function XYZ_Lev=gNavTaitLever(Head,Pitch,Roll,Lever,RotDirect), where
%Head - row, rotation angle for x0y in degree [H_t1... H_tn];
%Pitch - row, rotation angle for z0x in degree [P_t1... P_tn];
%Roll - row, rotation angle for y0z in degree [R_t1... R_tn];
%Lever - 3 rows with lever arms coordinates [Lx1...Lxm; Ly1...Lym;
        Lz1...Lzm];
%RotDirect - 'frw' for forward rotation; 'rev' for reverse rotation;
%XYZ_Lev - "moved" lever arms coordinates: column is [Lx;Ly;Lz]; rows is
        t1...tn; 3rd-size is Lever_1...Lever_m
%if Roll or Pitch or Head is scalar, then vector with necessary length will
        be created.
%-----------------------
%Coordinate system is:
%^x(forward/roll)
%|
%o---> y(right/pitch)
%z(up/head)
%All right/clockwise rotation sign is + (if vector is rotated relative to
        axis);
%All left/unclockwise rotation sign is + (if axis are rotated relative to
        vector);
%-----------------------
%Example: tr=[120;60;-80];tmp001=gNavTaitLever(-2,-2,-2,tr,'frw');

Head=Head./180.*pi;Pitch=Pitch./180.*pi;Roll=Roll./180.*pi;
Len=max([size(Roll,2) size(Pitch,2) size(Head,2)]);Len=Len(1);
ifall(size(Head)==1), Head=repmat(Head,1,Len);end;
ifall(size(Pitch)==1), Pitch=repmat(Pitch,1,Len);end;
ifall(size(Roll)==1), Roll=repmat(Roll,1,Len);end;
ifall(size(Head)~=size(Roll))||all(size(Pitch)~=size(Roll)),
        error('Roll,Pitch,Head length must be equal or scalar');end;
ifsize(Lever,1)~=3, error('Levers must be 3 rows with lever arms
        coordinates');end;
NumL=size(Lever,2);

XYZ_Lev=zeros(3,Len,NumL);%allocate memory for output
forn=1:Len,%calculate rotation for all levers
 ifisempty(RotDirect)||strcmp(RotDirect,'frw'),%forward rotation >> Head-
        Pitch-Roll >> RollM*PitchM*HeadM*Lever
XYZ_Lev(:,n,:)=[1 0 0; 0 cos(Roll(n)) -sin(Roll(n)); 0 sin(Roll(n))
        cos(Roll(n))]*[cos(Pitch(n)) 0 sin(Pitch(n)); 0 1 0; -sin(Pitch(n)) 0
        cos(Pitch(n))]*[cos(Head(n)) -sin(Head(n)) 0; sin(Head(n))
        cos(Head(n)) 0; 0 0 1]*Lever;
 elseifstrcmp(RotDirect,'rev'),%reverse rotation >> Roll-Pitch-Head >>
        HeadM*PitchM*RollM*Lever
XYZ_Lev(:,n,:)=[cos(Head(n)) -sin(Head(n)) 0; sin(Head(n)) cos(Head(n)) 0; 0
        0 1]*[cos(Pitch(n)) 0 sin(Pitch(n)); 0 1 0; -sin(Pitch(n)) 0
        cos(Pitch(n))]*[1 0 0; 0 cos(Roll(n)) -sin(Roll(n)); 0 sin(Roll(n))
        cos(Roll(n))]*Lever;
 else
error('Incorrect RotDirect value');
 end;
end;

% mail@ge0mlib.com 05/27/2021
```

Appendix 2. Script for calculations using Model-2

```
%script gNavTaitTest;
%Calculate "error ellipsoid" for USBL misalignment angles, using 2 ship's
        location with target's coordinates by USBL.

SU=struct('du',[1;0.5;0.8],'dm',[0;0;0],'LeverUSBL',[3;0;0]);%set USBL and
        MRU misalignment error, set LeverUSBL in VCS [-0.5;0.5;-1]
        [1;0.5;0.8] [10;5;0]
%sh1m=[-60;-3;-5];sh2m=[60;3;5]; %set [Head Pitch Roll] for sh1 and sh2
sh1m=[0;0;0];sh2m=[0;0;0];%set [Head Pitch Roll] for sh1 and sh2
sh1=[0;0;0];sh2=[120;200;0];tr=[120;60;-80];%set ship coords and target
        coords
LeverUSBL=gNavTaitLever(-SU.dm(1),-SU.dm(2),-
        SU.dm(3),SU.LeverUSBL,'frw');%calculate LeverUSBL in ECS_MRU
%calculate 'true' target coords for ship1
tmp3=gNavTaitLever(-sh1m(1),-sh1m(2),-sh1m(3),tr-sh1,'frw');tmp2=tmp3-
        LeverUSBL;tmp1=gNavTaitLever(SU.dm(1),SU.dm(2),SU.dm(3),tmp2,'rev');
tr1=gNavTaitLever(-SU.du(1),-SU.du(2),-SU.du(3),tmp1,'frw')+sh1;
%calculate 'true' target coords for ship2
tmp3=gNavTaitLever(-sh2m(1),-sh2m(2),-sh2m(3),tr-sh2,'frw');tmp2=tmp3-
        LeverUSBL;tmp1=gNavTaitLever(SU.dm(1),SU.dm(2),SU.dm(3),tmp2,'rev');
tr2=gNavTaitLever(-SU.du(1),-SU.du(2),-SU.du(3),tmp1,'frw')+sh2;

%calculate 'dst-residual set'
dd=3;ddd=0.02;[dHu,dPu,dRu]=meshgrid(-dd:ddd:dd,-dd:ddd:dd,-
        dd:ddd:dd);dst=zeros(size(dHu));%set 'USBL angles error corrections'
        by Pich&Roll&Head grid
dHm=SU.dm(1);dPm=SU.dm(2);dRm=SU.dm(3);%set 'false' MRU's misalignment angles
LeverU=[0;0;0];%set 'false' LeverUSBL in ECS_MRU // LeverU=LeverUSBL;
tic;
if0,disp('Using for cycle');
 forn=1:numel(dHu),
 %target coordinate recalculate with 'angles correction' for ship1
tmp1=gNavTaitLever(dHu(n),dPu(n),dRu(n),tr1-sh1,'rev');tmp2=gNavTaitLever(-
        dHm,-dPm,-dRm,tmp1,'frw');tmp3=tmp2+LeverU;
r1=gNavTaitLever(sh1m(1),sh1m(2),sh1m(3),tmp3,'rev')+sh1;
 %target coordinate recalculate with 'angles correction' for ship2
tmp1=gNavTaitLever(dHu(n),dPu(n),dRu(n),tr2-sh2,'rev');tmp2=gNavTaitLever(-
        dHm,-dPm,-dRm,tmp1,'frw');tmp3=tmp2+LeverU;
r2=gNavTaitLever(sh2m(1),sh2m(2),sh2m(3),tmp3,'rev')+sh2;
 %distance between sh1&sh2 target locations
dst(n)=sqrt(sum((r1-r2).^2));
 end;
elsedisp('Not using for-cycle');
[X1,Y1,Z1,X2,Y2,Z2]=gNavTaitNLever(dHu,dPu,dRu,[tr1-sh1 tr2-sh2],'rev');
[X1,Y1,Z1]=gNavTaitLeverN([-dHm;-dPm;-
        dRm],X1,Y1,Z1,'frw');[X1,Y1,Z1]=gNavTaitLeverN(sh1m,X1+LeverU(1),Y1+L
        everU(2),Z1+LeverU(3),'rev');
[X2,Y2,Z2]=gNavTaitLeverN([-dHm;-dPm;-
        dRm],X2,Y2,Z2,'frw');[X2,Y2,Z2]=gNavTaitLeverN(sh2m,X2+LeverU(1),Y2+L
        everU(2),Z2+LeverU(3),'rev');
dst=sqrt((X1+sh1(1)-X2-sh2(1)).^2+(Y1+sh1(2)-Y2-sh2(2)).^2+(Z1+sh1(3)- Z2-
        sh2(3).^2);
end;
toc;
dst(dst>10)=nan;
%draw figures and slices
figure(1);h=slice(dHu,dPu,dRu,dst,1,0.5,0.8);forn=1:numel(h);h(n).EdgeColor='
        none';end;xlabel('dH';ylabel('dP';zlabel('dR');
figure(2);h=slice(dHu,dPu,dRu,dst,0.5,1.5,1);forn=1:numel(h);h(n).EdgeColor='
        none';end;xlabel('dH';ylabel('dP';zlabel('dR');
```

```
figure(3);h=slice(dHu,dPu,dRu,dst,-1.5,0,-
        1.8);forn=1:numel(h);h(n).EdgeColor='none';end;xlabel('dH';ylabel('dP
        ';zlabel('dR');
figure(4);h=slice(dHu,dPu,dRu,dst,0,-0.5,-
        0.2);forn=1:numel(h);h(n).EdgeColor='none';end;xlabel('dH';ylabel('dP
        ';zlabel('dR');
figure(5);h=slice(dHu,dPu,dRu,dst,0.5,-1,-
        0.1);forn=1:numel(h);h(n).EdgeColor='none';end;xlabel('dH';ylabel('dP
        ';zlabel('dR');

% mail@ge0mlib.com 06/07/2021
```