



Flanders
State of
the Art

25_044.1
FH reports

Shipping6DOF – Python Package for the study of ship motions

Theory and Software Design

DEPARTMENT
MOBILITY &
PUBLIC
WORKS

www.flandershydraulics.be

Shipping6DOF – Python Package for the study of ship motions

Theory and Software Design

Lopez Castaño, S.; Verelst, K.; Vanlede, J.

Legal notice

Flanders Hydraulics is of the opinion that the information and positions in this report are substantiated by the available data and knowledge at the time of writing.

The positions taken in this report are those of Flanders Hydraulics and do not reflect necessarily the opinion of the Government of Flanders or any of its institutions.

Flanders Hydraulics nor any person or company acting on behalf of Flanders Hydraulics is responsible for any loss or damage arising from the use of the information in this report.

Copyright and citation

© The Government of Flanders, Department of Mobility and Public Works, Flanders Hydraulics 2026
D/2026/3241/096

This publication should be cited as follows:

Lopez Castaño, S., Verelst, K., Vanlede, J. (2026). Shipping6DOF – Python Package for the study of ship motions: Theory and Software Design. Version 4.0. FH Reports, 25_044_1. Flanders Hydraulics: Antwerp

Reproduction of and reference to this publication is authorised provided the source is acknowledged correctly.

Document identification

Customer:	Flanders Hydraulics	Ref.:	WL2026R25_044_1
Keywords (3-5):	Ship Motions, 6DOF, Locks		
Knowledge domains:	Hydraulic Structures > Locks > Filling and Emptying system > In-situ measurements		
Text (p.):	11	Appendices (p.):	/
Confidential:	☒ No ☒ Available online		

Auteur(s):	Lopez Castaño, Santiago
------------	-------------------------

Control

	Name	Signature
Reviser(s):	Verelst, K.	Getekend door:Kristof Verelst (Signature) Getekend op:2026-07-15 15:10:39 +02:0 Reden:Ik keur dit document goed
	Vanlede, J.	Signed by:Vanlede Joris Dries S Signed at:2026-09-23 09:41:40 +02:00 Reason:Ik keur dit document goed
Project leader:	Lopez Castaño, S.	Getekend door:LOPEZ CASTAÑO Santia Getekend op:2026-08-17 14:43:17 +02:0 Reden:Ik keur dit document goed

Approval

Head of Division:	Bellafkih, K.	Getekend door:Bellafkih Abdelkarim Getekend op:2026-07-15 20:44:08 +02:0 Reden:Ik keur dit document goed
-------------------	---------------	--

Abstract

This document summarizes the basic theory and implementation of a python package for the study of motions of rigid plates, using sensor measurements. A short explanation of the development of the python package, necessary inputs, and outputs is given. Additionally, the theory behind the study of motions in rigid plates is also described. The present text should be considered supporting material to the Jupyter notebook provided with the installation of shipping6DOF, which features complete tutorials and validations.

Index

Abstract	III
Index	V
List of Figures	VI
1 Introduction	1
2 Requirements	2
2.1 General considerations	2
2.2 Out of Scope	2
3 Theory	3
3.1.1 Fundamentals of the package: mathematical background	3
3.1.2 Determining R and $\mathbf{d}$ from two planes	4
3.2 Frame of reference conversion, change of axes, basis	5
4 Methodology: Domain Driven Design	6
5 Verification and Validation	7
6 Conclusions	9
7 Installation of the package	10
8 References	11

List of Figures

Figure 1 – Basics of vector rotations in 3D space.....	3
Figure 2 – Definition of a plane crossing an object through A	4
Figure 3 – (LEFT) Triangular plates (in red) with sensors O, A, and B. The blue line is the mediatrix and the imaginary sensor is indicated as O'. (RIGHT) Theoretical Results using Geometry.	7
Figure 4 – Results of 6DOF versus Theoretical results. Notice that the datasets are exactly the same, therefore indistinguishable from each other.	8

1 Introduction

Locks form an essential link in the Flemish waterway network. Flanders Hydraulics (FH) executes studies to optimize the filling and emptying systems of inland and maritime locks in Flanders. The aim is to achieve safe and rapid levelling with the most economical design possible. A vessel being levelled in a lock chamber will, as a result of the inflowing or outflowing water, experience forces that translate in motions within the lock's chamber. To avoid contact against the lock's chamber and doors, the ship may be moored and may make use of its thrusters during levelling. Neither the forces in the mooring lines nor the vessel's motions within the chamber should become excessive during levelling. For this reason, it is important to gain a good understanding of both the motions and forces acting on it.

FH has extensive experience in carrying out in-situ measurements of the filling and emptying of locks. Since 2007, there has been a strong focus on measuring the longitudinal water level gradient in the lock chamber during levelling. As this technique has its limitations in terms of both accuracy and safety, a research project was carried out between 2019 and 2024 to develop a setup for contactless measurement of a ship's inclination during transit. These ship inclinations are a good measure of the hydrodynamic force a ship experiences during levelling and can be compared with criteria from literature that are also applied in design and evaluation. This research project showed that both a fully contactless measurement technique based on photogrammetry and a measurement based on surveying total stations are possible. Further development into an operational measurement setup must still to be taken up.

De Vlaamse Waterweg (DVW) as water agency started a test with unmanned lock operations on the Leuven-Dijle Canal. During this project positive experience was also gained in carrying out contactless measurement of a ship's inclination based on surveying total stations. , During this project surveyors from DVW measured the position over time of three prisms located on a ship in the lock. Data were processed using scripts developed within the research project. The processing was ad-hoc and therefore requires the necessary technical knowledge of the methodology and the scripts.

The aim of the current project is to develop a Python package for processing such total station measurements in order to determine the 6 DOF-displacements of a vessel during levelling of a lock.

2 Requirements

Given the various origins of input data, quality, and type it is not practical to define a single interface that accommodates to the requirements of the user. On the other hand, the output may also be interpreted or needed in completely different ways (different units, different axes, as deltas, or cumulatives, etc). These two challenges then suggests an approach general enough for it to be customizable according to the type of input required, and the output that is expected from it.

2.1 General considerations

The following are the invariants considered for the design of the software package dedicated to the analysis of 6DOF displacements:

- The analysis of 6DOF displacements can be performed on 2, 3, and even more input points that may represent lines or surfaces.
- These surfaces and lines representing objects subjected to displacements may need to be aligned with a certain axis that remains static, or moves along with the input surface.
- The angles and displacements (6DOF) are given with respect to two consecutive time frames.
- Angles are always given in Radians.
- Input data is assumed having the same scale, coordinate system, and units.
- The library is intuitive to use.
- The user has basic knowledge of python, data analysis, and linear algebra.

The Python Package is designed having these invariants into considerations.

2.2 Out of Scope

As explained earlier the input data may take not only many forms, but may also come from different sources. Typically, such measurements are the product of survey campaigns where it might be the case that different equipment with different accuracies are used. Synchronization between measurement equipment is also essential: a phase lag of 1 second between two readings (or points) used as input for the 6DOF package may provide completely different results compared to the synchronized results. In some other instances, particularly by extremely sensitive equipment, the timeseries of the input data may need a high-pass filter, or denoising, for example. Finally, readings may have gaps or data may be incomplete for some unknown reason. There are then as many ways to manipulate data, as there's data, so the library will not attempt to approach this challenge at this phase.

On the other hand, also the output data may need to be presented in various formats and scales. Perhaps the required angle is not in degrees or radians, but slope percentages, or weight displacements. It is impractical to consider all possible choices an user may want. The user will be responsible of the use of the output data, and the package will remain within the confines and definitions given by the algebra of tensors and vectors.

In summary, the library will assume the input data is correct, synchronized, and complete, so it will not try to manipulate (or fix) it in any way. Also, it is the task of the user to prepare the input data and process the output data in the desired form. On the other hand, given how the output (particularly angles derived from tensors) will be needed and interpreted in different disciplines (Aerospace, naval, CFD), the package will provide results in the most general way possible: it is responsibility of the user to provide the correct context to the results and "transform" it as necessary.

3 Theory

Given the considerations, invariants, and out-of-scope features that the library should have, the most natural way of implementing this library is via the concept of “Domain Driven Design” (DDD). That is, making the package work in the natural language describing the algorithms behind it. In this case, the study of displacements and rotations are naturally described in terms of vector, and tensor, mathematics (or Linear Algebra).

This section will first explain the basics involved in calculating the 6DOF components of a moving object, and then will integrate those mathematical concepts into an intuitive API to be used for these purposes.

3.1.1 Fundamentals of the package: mathematical background

The displacement of a vector in 3D space is described by the following expression:

$$\mathbf{d} = \mathbf{X}(t_1) - \mathbf{X}(t_0),$$

where the position $\mathbf{X}$ of the point is known at two discrete times t_0 and t_1 . Notice that vectors are indicated in boldface. However, its rotation cannot be determined from the equation above. The rotation of a vector or object in space will be assumed to happen along an axis $\mathbf{k}$ with a rotation angle θ .

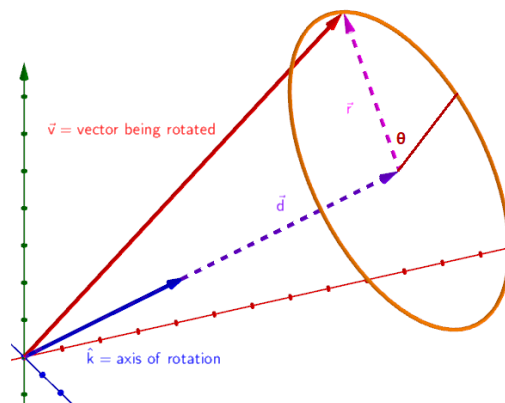


Figure 1 – Basics of vector rotations in 3D space.

The rotation of a vector (or the application of one vector basis to another basis) is then expressed in the following way:

$$\mathbf{v}' = \mathbf{R}\mathbf{v},$$

Where $\mathbf{R}$ is referred to as the rotation tensor or matrix following the properties of linear independence, unitarity, and orthogonality, and $\mathbf{v}$ the vector to be rotated. There are various ways for determining the rotation matrix $\mathbf{R}$, here the formula of Rodrigues is used:

$$\mathbf{R} = \mathbf{I} + (\sin \theta)(\mathbf{k} \times \mathbf{I}) + (1 - \cos \theta)\mathbf{k}\mathbf{k}^T$$

Where $\mathbf{k}$ is the (unitary) axis of rotation, $\mathbf{I}$ the identity tensor, and θ the angle of rotation. Given that the tensor $\mathbf{R}$ forms a basis, it is understood that the first column of $\mathbf{R}$ corresponds to the direction of the first axis, and so on. This means that a coefficient in the $\mathbf{R}$ -tensor, say R_{ij} , corresponds to a component of rotation i around the j axis.

Finally, the vector $\mathbf{d}$, and the rotation matrix R form a complete basis for the transformation (displacement and change of basis) of a vector in 3 dimensions:

$$\mathbf{X}'(t_1) = R(\mathbf{X}(t_0) + \mathbf{d}).$$

The Euler angles (Pitch, yaw, roll) can be easily recovered from the rotation tensor, however, interpretations may differ depending on the given coordinate basis. For example, following the Yaw-Pitch-Roll (Z-Y-X) Sequence:

$$\theta_{pitch} = \arcsin(-R_{31}),$$

$$\theta_{yaw} = \text{atan2}(R_{23}, R_{33}),$$

$$\theta_{Roll} = \text{atan2}(R_{21}, R_{11}).$$

If R_{33} or R_{11} is close to 1 or -1 a gimbal lock occurs, meaning that the euler angle along the Z or X axis cannot be determined.

Notice that the problem description is complete whenever R and $\mathbf{d}$ are known.

3.1.2 Determining R and $\mathbf{d}$ from two planes

Typically, if the object on which displacements and Euler angles are calculated is assumed rigid an arbitrary plane crossing that object can be defined at a point (Figure 1) to be used for our calculations. A plane is defined by a vector to a point in the plane and a normal to that plane. The normal vector $\mathbf{n}$ is defined directly (using two points in space) or calculated via cross-product and Least-Squares fit.

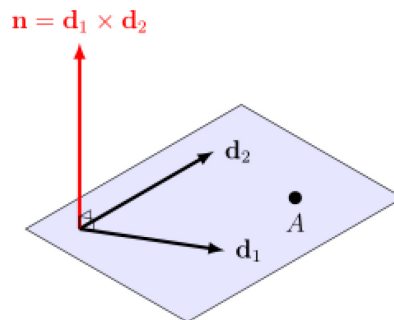


Figure 2 – Definition of a plane crossing an object through A

For defining a plane using three points in space, the points need to be non-collinear and the vector used for displacements is the point of intersection of the two vectors emanating from such intersection. For defining a plane with more than 3 points, a singular value decomposition (SVD) is needed to fit a 3D plane to such points.

The calculation of the rotation vector R involves then the calculation of the angle between the normal vector of the plane at two time instances (t_0 and t_1) along the axis of rotation between the two normals. The angle of rotation between the two normals can be calculated in the following way:

$$\cos \theta = \mathbf{n}(t_0) \cdot \mathbf{n}(t_1),$$

And the axis of rotation between the plane normals is

$$\mathbf{k} = \mathbf{n}(t_0) \times \mathbf{n}(t_1).$$

With these two parameters known the calculation of the Rotation tensor and Euler angles can proceed as explained earlier.

3.2 Frame of reference conversion, change of axes, basis ...

As it could be inferred from the explanations just shown, for the calculation of rotations and displacements all that is required is an origin (a point) and an axis defining a plane at two different time steps. The definition of the axis and origin is a choice left to the user, that is, the user must define the considered plane AND the frame of reference to be used.

Changing of frame of reference and projections follow the rules of basic linear algebra: a change of basis can be conducted by defining linearly independent bases (e_1, e_2, e_3) and creating a basis matrix using the three basis vectors as columns:

$$A = [e_1 \quad e_2 \quad e_3]$$

The rotation tensor and displacements in the new basis are then defined as:

$$R' = AR,$$

$$\mathbf{d}' = A\mathbf{d}.$$

4 Methodology: Domain Driven Design

The theory explained earlier makes heavy use of algebra of vectors and tensors. The development of a library dedicated to 6DOF should reflect this in an elegant way. Namely, the operations explained in previous sections may be expressed as follow:

```
# Create two vectors a, and b
a = vector(2.3,5.8,3.3)
b = vector(1.0,-2.1,0.35)
# create an axis, orthogonal to the plane containing the vectors a and b
axis = (a ^ b) # '^' is cross product
axis = axis/axis.norm() # axes need to be normalized
# create a vector c in space and rotate it 23.0° around the 'axis'
c = vector(1.0,1.0,1.0)
# create rotation matrix corresponding to 'axis' and angle 11.5°
angle = 23.*pi/180./2.
R = create_rot_matrix_from_theta_axis(angle,axis)
# rotate vector c around 'axis', a total of 23°
c_rotated = R*(R*c)
# Calculate the Euler angles from the product R*R, corresponding to c_rotated
roll, yaw, pitch = rotation_matrix_to_euler_angles(R*R)
# The distance between c and c_rotated is defined as its difference:
d = c_rotated - c
# We can calculate the angle between c and c_rotated, it must be 23°
angle = acos(c_rotated*c)*(180.0/pi)
```

Compared to the use of numpy directly, the advantage of this API is that the current interface checks for consistency of the operands and enforces the user to preserve the 3D nature of the problem, avoiding errors that could easily happen while using numpy. The proposed API also allows for more natural commutation, and permutations, without multiple subroutine calls. Also, algorithmically intensive procedures are hidden by the API. Additional utility functions for defining planes from 3 or more points in 3D space are made available in the API. Check the package's tutorial and API for details.

As observed previously, the library offers facilities for calculating planes using 3 or more points (using Singular Value Decomposition methods, or SVD, not explained here)., but the user may perfectly create axes and origins manually using 2 or 3 points.

5 Verification and Validation

This section will briefly describe the verification and validation process of the 6DOF package for oscillations on a triangular plate. A fully detailed description of the process can be found in the Jupyter notebook included within the package. Additional Tutorials can be found there, too.

Imagine a triangular plate of unitary sides, on which sensors are installed at each vertex. One would like to know which would be the oscillations (roll and pitch) of a fourth sensor installed along the mediatrix of the triangular plate but not in the plate itself. The displacements we would like to know at the centroid of the plate. See Figure 3 for reference.

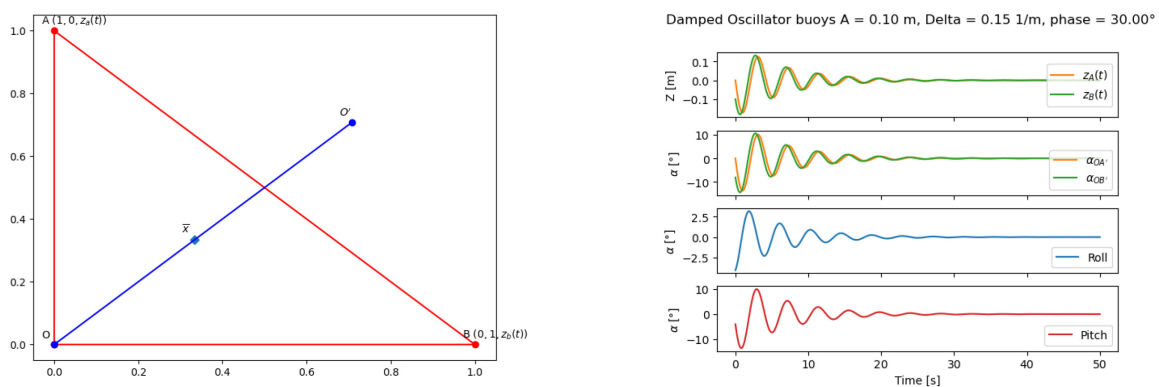


Figure 3 – **(LEFT)** Triangular plates (in red) with sensors O, A, and B. The blue line is the mediatrix and the imaginary sensor is indicated as O'. **(RIGHT)** Theoretical Results using Geometry.

Sensors (or Buoys) A and B are excited by means of damped oscillators, and the corresponding theoretical oscillations in O' can be calculated theoretically by means of Euclidean geometry. Calculating the displacements of the plate's centroid is also possible by means of Euclidean geometry and is left to the reader to perform it. Theoretical calculations are shown in Figure 3.

By creating time series of the sensors' locations one may input this in the 6DOF program and obtain the same exact results (RMSE=0), as shown in Figure 4.

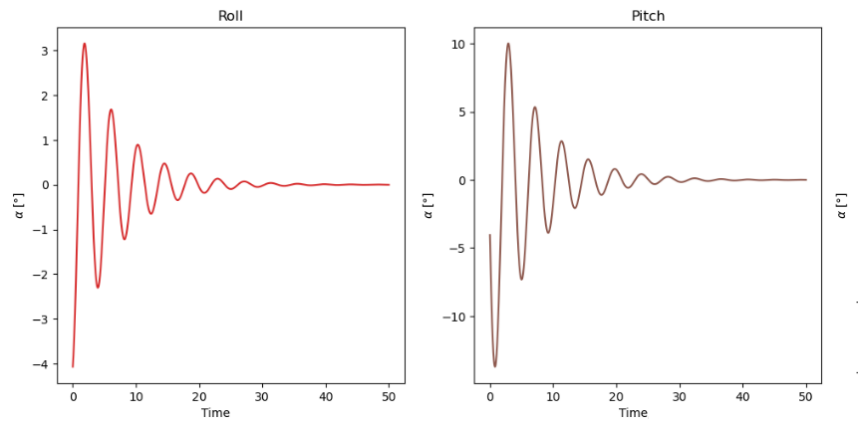


Figure 4 – Results of 6DOF versus Theoretical results. Notice that the datasets are exactly the same, therefore indistinguishable from each other.

6 Conclusions

A python package, shipping6DOF, was developed to calculate the motions of ships by extracting 6DOF motions from planes. Such planes may, for instance, be extracted from sensors (prismas, RTK GPS, etc) placed on the deck of the vessel. The package itself makes no assumption on the origin of the data, making it general for other type of analyses. Finally, the package follows contemporary software design and allows for an intuitive and flexible interface.

The package may be found either in the SVN or github repository of WL. The installation of the package can be done via pip, where all pre-requisites are automatically resolved.

7 Installation of the package

The package can be installed via pip :

```
pip install git+ssh://git@bitbucket.org/departement-mow/25_044-ontwpython6dof.git
```

8 References

Evans, E. (2004). *Domain-driven design: tackling complexity in the heart of software*. Addison-Wesley Professional.

DEPARTMENT **MOBILITY & PUBLIC WORKS**
Flanders hydraulics

Berchemlei 115, 2140 Antwerp

T +32 (0)3 224 60 35

F +32 (0)3 224 60 36

flanders.hydraulics@vlaanderen.be

www.flandershydraulics.be