Validation and Optimization of an Open-Source Novel Nonlinear Froude-Krylov Model for Advanced Design of Wave Energy Converters

Manual of the Nonlinear Froude-Krylov Matlab demonstration toolbox

GIUSEPPE GIORGI

H2020-MSCA-IF-2018 (832140): OpenWave
Politecnico di Torino, Turin, Italy
October 2019

giuseppe.giorgi@polito.it

Manual of the Nonlinear Froude-Krylov Matlab demonstration toolbox

Abstract

This document purports to describe in detail the mathematical framework for a computationally efficient computation of nonlinear Froude-Krylov forces (NLFK) in 6 degrees of freedom (DoFs) for axisymmetric floating objects. Additionally, this document also acts as a reference manual to a set of Matlab scripts, forming a demonstration toolbox to show the capabilities of the NLFK approach and provide an easy, operative, and ready-to-use implementation of the method. The toolbox is openly available at [1]. This document and the toolbox are licensed with a Creative-Commons-By-Attribution-Share-Alike (CC-BY-SA) license [2]. Note that this is the first version of the toolbox, so any feedback and potential corrections are welcome. Moreover, the user is highly invited to contact the author for any doubt, ideas or suggestions, deeper investigation, higher-complexity problems, and eventually collaboration. Finally, note that this toolbox is the precursor of an open source software, coded in a lower-level coding language than Matlab, hence much faster, which will be virtually shared by mid 2021.

1. Overview

The content of this toolbox and manual has been developed in the context of nonlinear hydrodynamic modelling for wave energy converters. Therefore, most of the discussion and considerations are related to the challenges and requirements of this specific area of ocean engineering. However, the tools herein provided have general validity and broader applicability, so the user is invited to use them in the most creative way. In fact, this mathematical framework allows to easily displace an axisymmetric geometry in 6 DoFs and to compute arbitrary surface integrals on a time-varying portion of the body. In the specific application case of the computation of NLFK forces, the integrand is the undisturbed pressure field, but any other function could be integrated using the same mathematical framework.

The rationale behind the development of this methodology is the critical need for a better compromise of accuracy and computational time with respect to state-of-the-art mathematical models in wave energy conversion applications. Economic viability is the sine qua non condition for commercial competitiveness and industrial sustainability of the wave energy conversion sector and it can be achieved only through substantial reduction of costs and increase in performance. Both such objectives can be accomplished only through holistic design, control, and optimization studies, the effectiveness of which strongly depends on the accuracy of mathematical models. However, a major limiting factor for the feasibility of application of a particular mathematical model is the computational time.

Exploring the computation/fidelity continuum [3], the NLFK approach aims at medium-high fidelity at a low computational cost. Considering the power-production region, hence excluding extreme events and survivability conditions, partially-nonlinear potential-flow-based models have

the potential to realize an appropriate trade-of between computational time and fidelity [4]. In particular, the dynamic response of point absorbers (device relatively small compared to the wavelength) is mainly influenced by Froude-Krylov forces, while radiation and diffraction efforts are normally relatively small and linear [5]. Therefore, NLFK approaches for point absorbers are particularly accurate, where the undisturbed pressure field is integrated onto the instantaneous wetted surface. However, mesh-based approaches, necessary for geometry of arbitrary complexity, are renown to be slow, due to time-consuming re-meshing of the submerged surface [6]. If the floater is assumed to be axisymmetric (not a restrictive assumptions for point absorbers, which are normally axisymmetric), a computationally efficient NLFK formulation exist, relying on the analytical representation of the wetted surface, hence needless of a mesh. If rotations are negligible, algebraic solutions can be obtained [7], otherwise numerical integration is required [8, 9]. Although a fair comparison is challenging, due to different coding languages (Fortran versus Matlab), such numerical solution to the analytically defined NLFK forces for axisymmetric devices is between one and two orders of magnitude faster than a mesh-based approach [7]. Note that nonlinearity of the pressure field can also be taken into account [10].

This Matlab demonstration toolbox has the objective of providing the user with a set of scripts that can quickly and pragmatically shows the capability of this method, without worrying about the mathematical implementation. The code is completely open-source, modifiable and reusable. The remainder of the manual is structured as follows: Sect. 2 discusses the content and the usage of the Matlab demonstration toolbox; Sect. 3 provides few consideration about the computational time. The detailed theoretical background is provided in the following sections: Sect. 4 presents the frames of reference used and the transformation from one to the other; Sect. 5 describes the integration approach and the parametric representation adopted; Sect. 6 show the particular case of NLFK force integrals.

Contents

2. Matlab demonstration toolbox

This section aims at giving a practical and swift guide to the use of the toolbox. Please refer to Sections from 4 for a detailed description of the theoretical background, necessary for a deeper understanding of the content of the toolbox and this section.

For sake of simplicity and flexibility, this demonstration toolbox is coded in order to be able to describe geometries composed of cylindrical or conical sections only, which cover the vast majority of application cases. Although other surfaces are compatible with the NLFK framework (like spherical or toroidal sections), it is preferred to keep the demonstration toolbox simple. Don't hesitate to contact the author for further details about the implementation of slightly more complex geometries.

The toolbox is composed of:

- *LICENSE-CC-BY-SA.txt*: Creative-Commons-By-Attribution-Share-Alike (CC-BY-SA) license [2]
- Folder: *Floaters*: it contains the computed geometries (the folder is created by one of the scripts)
- Folder: *NEMOH*: it contains the open-source BEM code Nemoh, developed at Ecole Centrale Nantes, as well as a Matlab function script (*FUN_NemohAxi*) modified by the author, and one folder for each geometry simulated
- Folder: *SupportingFunctions*: it contains utility functions created by the author (*setMatlabDefaults*)
- Matlab script *RUN_defineNLFKgeom*: it defines the geometry, plots the displaced floater in both the WF and BF with the free surface elevation and the intersection with it, and computes body properties and hydrostatic stiffness.
- Matlab script *RUN_checkNLFK*: it runs Nemoh and compares the dynamic NLFK in linear conditions against linear results from the BEM code
- Matlab function *FUN_defineNLFKgeom*: it defines the geometry in the NLFK framework
- Matlab function *FUN_plot3D*: it plots the geometry in the WF and BF, as well as the free surface elevation and its intersection with the floater
- Matlab function *FUN_computeBodyProperties*: it computes total and wetted lateral surface, total and submerged volumes, and centre of buoyancy through numerical integration
- Matlab function *FUN_NLFK_st_dy*: it computes the static and dynamic NLFK force
- Matlab function *FUN_NLFK*: it computes the total NLFK force separately

All the content is commented in detail so that, having this manual as a reference, it should be straightforward to interpret the code. However, hereafter a quick guide to a suggested usage of the toolbox is provided.

2.1. *RUN_defineNLFKgeom*

The script *RUN_defineNLFKgeom* is the first one that should be run.

If flag *flags.defineNLFKgeometry* is equal to 1, the geometry is created, otherwise is loaded. A set of example predefined geometry is provided, namely a cylinder, a cone, a hollow cylinder, a hollow cone, and a slightly more complex buoy composed of two cylindrical sections connected through a conical one. These and other geometries are simply defined by a $(2 \times N_p)$ matrix, providing N_p couples of $\hat{x}$ and $\hat{z}$ describing the profile of revolution $f(\varrho)$.

It follows that $N_p - 1$ sections are defined, each one between two consecutive points. Such portions are named *patches*, and they can be either cylinders and cones (implying the use of cylindrical coordinates) or discs (implying the use of polar coordinates).

The order with which the points are provided is meaningful, since it defines the direction of the normals. External surfaces (most common case) are defined with points given in descending or horizontal order. If internal surfaces are present (moonpools of hollow geometries), consecutive points are given in ascendant order.

2.1.1. *FUN_defineNLFKgeom*

The function *RUN_defineNLFKgeom* defines all useful parameters in the NLFK framework. Patches requiring cylindrical and polar parametrization are discriminated, as well as patches that reach the free surface. Note that, for simplicity, it is assumed that the free surface elevation is fully contained in the patch that intersects the still water level.

In case of cylindrical coordinates (for cylinders or cones), the following parameters are defined:

- ϱ_1 and ϱ_2, delimiting the patch at the bottom and the top, respectively
- $f(\varrho)$ and $f(\varrho)' = \frac{df(\varrho)}{d\varrho}$, as defined in Fig. 4 and (47)
- $n1$, $n2$ and $n3$, being the three components of (48)
- $n4$ and $n5$, being the first two components of (49)

In case of polar coordinates (for discs), the following parameters are computed:

- r_1 and r_2, delimiting the patch at the inner and outer radii, respectively
- $n1$, $n2$ and $n3$, being the three components of (45)
- $n4$ and $n5$, being the first two components of (46)

Furthermore, total and wetted lateral surface, and total and submerged volumes are computed algebraically and stored for comparison with numerical integrations, as discussed in Sect. 5.2.

Finally, note that the approach of patches and set of Matlab anonymous functions is flexible and transparent, but computationally inefficient. Once the geometry is defined, a more computationally convenient solution is to replace all the cross-references in the anonymous functions by their numerical value.

2.1.2. FUN_plot3D

Going back to the script *RUN_defineNLFKgeom*, if flag *flags.plotFloater* is equal to 1, the function *FUN_plot3D* plots the body in both the WF and the BF, according to a vector of translational and rotational displacements called *states*. Translations are expressed in the WF $(x_R, y_R, z_R)'$, while rotations are expressed about body-fixed axes, as explained in Sect. 4.2. An example is shown in Fig. 1. If states $\in \emptyset_{6\times1}$, the rest position is shown, and the profile provided by the user is also plotted in the WF.

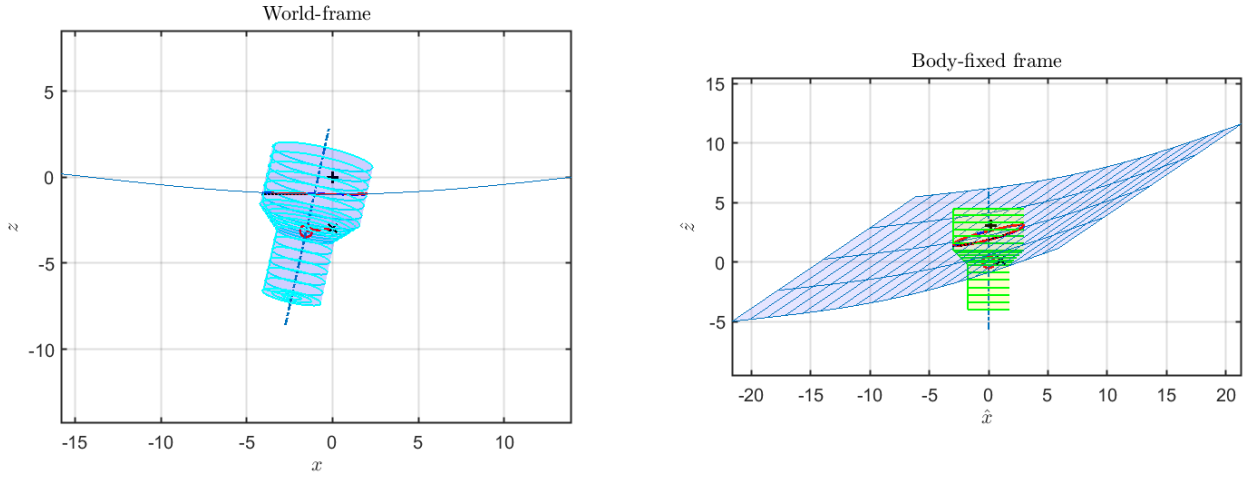

Figure 1: Example of the plot of a buoy, the free surface elevation and their intersection in the world-frame and the body-fixed frame.

Furthermore, is *flags.plotEta* is equal to 1, also the free surface elevation is plotted, in both frames. Moreover, the intersection between the floater and the free surface elevation is computed using the three approaches discussed in Sect. 4.3, namely the exact intersection, the linear approximation and the flat approximation. The author suggests to use the linear approximation, which is almost as fast to compute as the flat approximation, while being almost as accurate as the exact solution for usual operational waves.

2.1.3. FUN_computeBodyProperties and hydrostatic stiffness

Going back to the script *RUN_defineNLFKgeom*, if flag *flags.computeGeometricProperties* is equal to 1, total and wetted lateral surface, total and submerged volumes, and centre of buoyancy are computed through numerical integration, as discussed in Sect. 5.2.

If flag *flags.computeHydrostaticStiffness* is equal to 1, the linear hydrostatic stiffness is computed algebraically, as discussed in Sect. 6.1.1. Moreover, K_h is computed using the NLFK framework, by assuming no incoming wave and an infinitesimal displacement. The function *FUN_NLFK_st_dy* is used, which integrates the static and dynamic pressure separately.

2.2. RUN_checkNLFK

This script implements what discussed in Sect. 6.1.2, namely a comparison between BEM results and NLFK computation in linear conditions. If *flags.runNemoh* is equal to 1, a BEM

simulation is performed in order to obtain hydrodynamic coefficients for the specified buoy. Otherwise, previously computed parameters are uploaded. Note that the accuracy of the solution and the computational time depend on the refinement of the mesh-discretization, which is defined by *Npan* (target number of panels of the mesh) and *n_ang* (number of points for angular discretisation).

If *flags.computeNLFK* is equal to 1, nonlinear dynamic FK coefficients are computed under linear conditions, i.e. zero displacement and very small wave amplitude. Otherwise, previously computed coefficients are uploaded.

Finally, if *flags.plotComparison* is equal to 1, linear and nonlinear FK parameters are compared, both in amplitude and phase. Normally, if the mesh-discretization of Nemoh and the absolute and relative tolerances of the numerical integration are adequate, the two curves should perfectly overlap, as shown in Fig. 7. However, note that Nemoh may have some issues in exactly compute the hydrodynamic curves in presence of moonpools [12], so some small differences may be noticed.

3. Computational time

This sections provides some considerations about the computational time of the functions provided in this toolbox and, more in general, about the NLFK approach.

As already mentioned in Sect. 2.1.1, this toolbox favours flexibility, simplicity, and transparency over computational efficiency. The set of Matlab anonymous functions with internal cross-references are slow when saved in a .mat file. A much faster solution, once the geometry is known, is to save the anonymous functions defined by explicit numerical values. Additionally, all components of equations from (45) to (49) are considered, and the respective integrals computed, even though some of the terms may be zero (the heave component of the lateral surface of cylinders).

As a general rule, the computational time is mainly proportional to the number of numerical integrals computed, namely one for each DoF and for each patch: considering, for example, the buoy in Fig. 1, there are 3 patches with cylindrical coordinates (top cylinder, cone, and bottom cylinder, each requiring 5 integrals) and one patch with polar coordinates (bottom disc), hence 18 integrals in total. If static and dynamic FK forces were computed separately, the number of integrals would double, reaching 36 in the example. Overall, the computational time is closely related to the complexity of the floater.

Future work will investigate the possibility of using alternative solutions to describe the entire surface, ideally having only one function, as opposed to many different patches. More efficient and tailor-made numerical integration schemes will also be studied. Note that, with the current algorithm, the accuracy and computational time of the results depend on the user-set absolute and relative tolerances of the integration algorithm. Finally, as a part of the current fellowship, an open-source software will be written in a lower-level coding language, which is expected to be one order of magnitude faster than the current Matlab implementation.

The user is highly invited to contact the author with any ideas or thoughts on how to improve the computationally efficiency of this model, both on the mathematical framework, the geometry representation, the integration algorithm, the code implementation, or any other aspect discussed in this manual.

4. Reference frames

4.1. World and body-fixed frames

Two right-handed frames of reference are used hereafter:

- World-frame (WF): $(x, y, z)'$
- Body-fixed frame (BF): $(\hat{x}, \hat{y}, \hat{z})'$

A given position point $\mathbf{p}$ in the three-dimensional (3D) space can be expressed (decomposed) in either the WF or the BF, and denoted by the notation $\mathbf{p}_{\text{WF}}$ or $\mathbf{p}_{\text{BF}}$, respectively.

The world-frame of reference $(x, y, z)'$ is inertial, with the origin at the still water level (SWL), the x-axis positive in the direction of propagation of the wave, and the z-axis positive upwards, as shown in Fig. 2. The body-fixed frame of reference $(\hat{x}, \hat{y}, \hat{z})'$ is not inertial, with the origin fixed at a reference point (RefP), usually either at the centre of gravity (CoG) or at the SWL, with the z-axis pointing upwards along the axis of the axisymmetric body, as shown in Fig. 3 for a generic axisymmetric body.

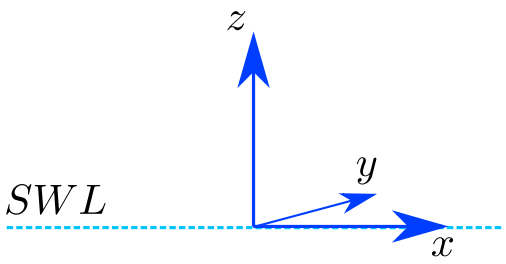

Figure 2: Inertial world-frame (WF)

The lateral surface of a generic axisymmetric geometry can be described by means of cylindrical coordinates. Note that the whole wetted surface of a floating axisymmetric buoy may also need discs, which are more conveniently represented with polar coordinates. Please refer to Sect. 5.1 for further details.

Using cylindrical coordinates, the axisymmetric geometry is defined by the revolution of a generic function $f(\varrho)$, between ϱ_1 and ϱ_2, as shown in Fig. 4. Note that $f(\varrho)$ must be defined so that the origin of the BF (i.e. the RefP) is at $\hat{z} = \varrho = 0$.

Therefore, the parametrization of the axisymmetric geometry, in the BF, is shown in equation (1).

$$\begin{cases} \hat{x}(\varrho, \vartheta) = f(\varrho)\cos\vartheta \\ \hat{y}(\varrho, \vartheta) = f(\varrho)\sin\vartheta \\ \hat{z}(\varrho, \vartheta) = \varrho \end{cases} \quad , \quad \vartheta \in [-\pi, \pi) \wedge \varrho \in [\varrho_1, \varrho_2] \tag{1}$$

The displacement of the RefP in WF coordinates is denoted by $(x_R, y_R, z_R)'_{\text{WF}}$. At rest, e.i. $(x_R = 0, y_R = 0, z_R = 0)'_{\text{WF}}$, the relative position between the WF and the BF is shown in Fig. 5:

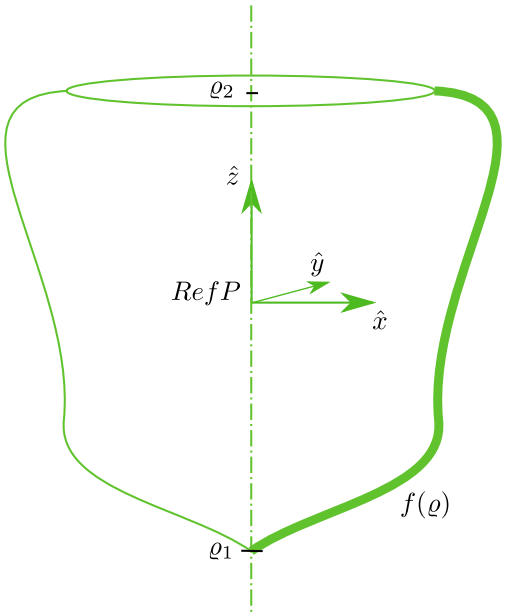

Figure 3: Non-inertial body-fixed frame (BF) for a generic axisymmetric body.

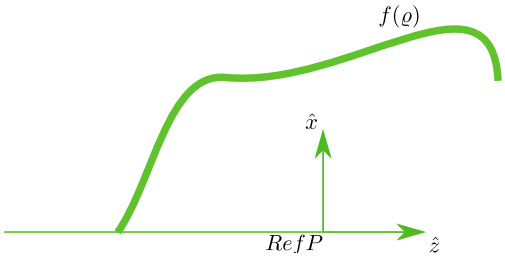

Figure 4: Generic profile of revolution in the BF.

the RefP (which is the origin of the BF) is, in the WF coordinates, at $(0, 0, z_{RefP})'_{\text{WF}}$. It follows that, at rest, a generic point in the BF coordinate is computed as:

$$\text{At rest: } \mathbf{p}_{\text{BF}} = \mathbf{p}_{\text{WF}} - (0, 0, z_{RefP})' \tag{2}$$

4.2. Translation and rotation

Figure 6 shows the body after a displacement $(x_R, y_R, z_R)'_{WF}$ and a rotation $(\phi, \theta, \psi)'$. Note that such rotations are considered about the three body-fixed axes and about the RefP. The 3-2-1 Euler rotation sequence is applied, commonplace in ocean engineering [11], with ϕ being the roll angle around the $\hat{x}$-axis, θ the pitch angle around the $\hat{y}$-axis, and ψ the yaw angle around the $\hat{z}$-axis.

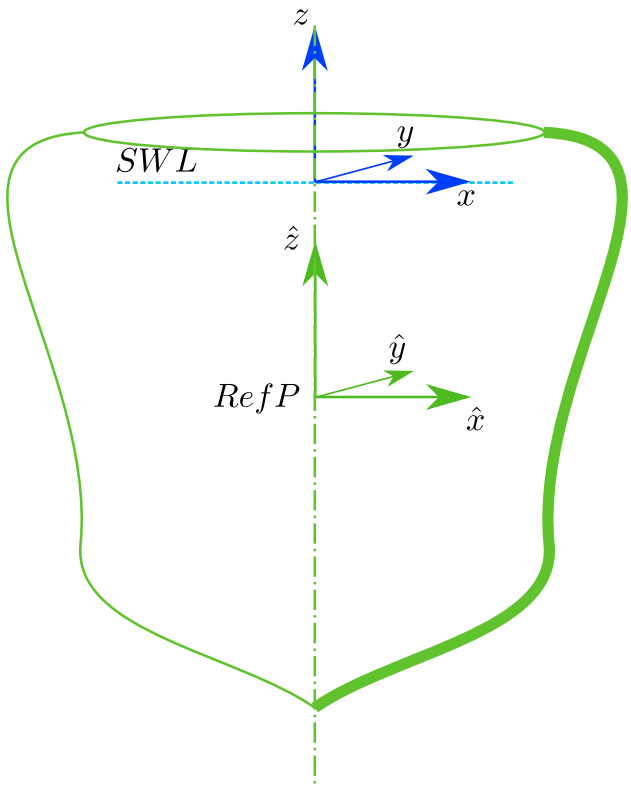

Figure 5: Relative position between frames WF and BF at rest.

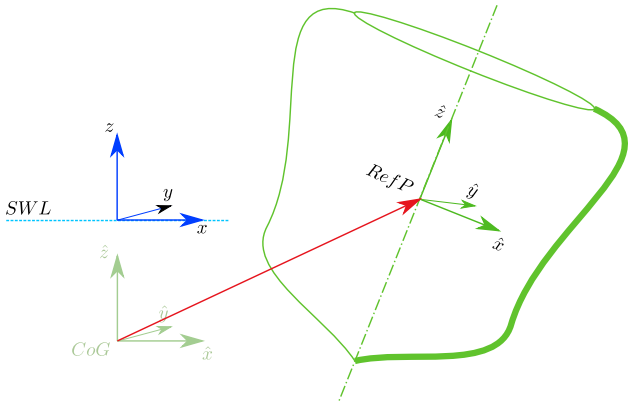

Figure 6: Relative position between frames after displacement.

The 3-2-1 rotation matrix Rot is defined as follows:

$$Rot = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix} \tag{3}$$

Where c and s stand for cosine and sine, respectively. Note that $Rot^{-1} = Rot'$, where Rot^{-1}

denotes the inverse of Rot.

Since rotations are defined around the RefP and about the body-fixed axes, the rotation matrix Rot in (3) pre-multiplies the BF-position vector $(\hat{x}, \hat{y}, \hat{z})'$, applying a rotation around the origin of the BF, which is also the RefP of the body. After the rotation, the axes of the two frames are aligned, and the translation $C = (x_R, y_R, z_R + z_{RefP})'_{\mathsf{WF}}$ can be applied. Note that C is the position of the RefP, in the WF coordinates, after displacement, hence also the relative position of the two frames.

In the $\mathbb{R}^3$ space, rotation and translation are applied sequentially. Conversely, it is more convenient to expand the space to $\mathbb{R}^4$, in order to apply rotation and translation with just one matrix R:

$$R = \left(\begin{array}{ccc|c} & Rot & & C \\ & & & \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \tag{4}$$

It follows that the transformation from BF to WF is given by:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = R \begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \\ 1 \end{pmatrix} \tag{5}$$

Conversely, the transformation from WF to BF is given by:

$$\begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \\ 1 \end{pmatrix} = R^{-1} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{6}$$

As a remark, note that, thanks to the particular structure or R, R^{-1} can be computed using the block matrix inversion theorem, so that

$$R^{-1} = \left(\begin{array}{ccc|c} & Rot' & & -Rot'C \\ & & & \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \tag{7}$$

Finally, note that the coordinates of one point are defined in a column vector. Consequently, an array of n points can be formed by stacking n position vectors, hence defining a matrix $\in \mathbb{R}^{4\times n}$. The whole array can be rotated by pre-multiplying the matrix R.

4.3. Intersection

The main advantage of the approach described in this manual is the ability to analytically represent the instantaneous wetted surface, which implies the analytical definition of the intersection between the body-surface and an arbitrary function describing the free surface elevation (η). In general, a 2D-wave is defined in the WF as a function of x and time (t), usually as a superposition of harmonics (irregular wave):

$$\eta(x,t) = \sum_{i=1}^{N_\omega} a_i \cos\left(\omega_i t + \varphi_i - k_i x\right) \tag{8}$$

where a_i, ω_i, φ_i, and k_i are the wave amplitudes, frequencies, phases, and wave numbers, respectively. Evidently, a regular wave is obtained in the limit case of $N_\omega = 1$. Note that geometric considerations take place at a constant time instant $t = t_0$; therefore, the explicit time-dependence of η will be omitted hereafter.

As further discussed in Sect. 6, it is convenient to define the NLFK integrals in the BF, hence requiring a mapping of the free surface elevation from the WF onto the BF ($\hat{\eta}$). Therefore, with reference to the cylindrical coordinates (ϱ, ϑ), shown in (1), finding the intersection translates into finding $\varrho = \hat{\eta}(\vartheta)$, i.e. a parametric representation of the free surface in the BF.

Let us define η and $\hat{\eta}$ the free surface elevation in WF and BF, respectively. The wave surface is defined, in 4D and in the WF, as:

$$\Gamma = \begin{pmatrix} x \\ y \\ \eta(x) \\ 1 \end{pmatrix} \tag{9}$$

Consequently, according to (6), equation (9) in the BF becomes

$$\hat{\Gamma} = R^{-1}\Gamma \tag{10}$$

The intersection is found by equating ϱ to the third line of $\hat{\Gamma}$, i.e. $\varrho = (0,0,1,0)R^{-1}\Gamma$. However, Γ must be expressed as a function of the parametric variables ϱ and ϑ, i.e. using $x = (1,0,0,0)R(f(\varrho)\cos\vartheta, f(\varrho)\sin\vartheta, \varrho, 1)'$, and $y = (0,1,0,0)R(f(\varrho)\cos\vartheta, f(\varrho)\sin\vartheta, \varrho, 1)'$. Finally, the intersection is the solution of the following equation:

$$\varrho = (0,0,1,0)R^{-1}\begin{pmatrix} (1,0,0,0)R(f(\varrho)\cos\vartheta, f(\varrho)\sin\vartheta, \varrho, 1)' \\ (0,1,0,0)R(f(\varrho)\cos\vartheta, f(\varrho)\sin\vartheta, \varrho, 1)' \\ \eta\left((1,0,0,0)R(f(\varrho)\cos\vartheta, f(\varrho)\sin\vartheta, \varrho, 1)'\right) \\ 1 \end{pmatrix} \tag{11}$$

Using the properties of R, see (7), and using the notation $R(i,j) = R_{ij}$, (11) simplifies to:

$$\begin{aligned} &f(\varrho)\left(\cos\vartheta(R_{11}R_{13}+R_{21}R_{23})+\sin\vartheta(R_{12}R_{13}+R_{22}R_{23})\right)+\\ &\varrho(R_{13}^2+R_{23}^2-1)+R_{33}(\eta(\vartheta,\varrho)-C(3)))=0 \end{aligned} \tag{12}$$

The zeros of (11) represent the *exact* solution of the intersection between the body and the free surface elevation, but a numerical solution to such a nonlinear problem is required.

However, it is reasonable to assume the free surface elevation to be either locally constant, or with a linear dependence on x. In fact, in normal applications, the wave steepness is relatively low, and the wavelength $\lambda = 2\pi/k$ is much longer than the characteristic dimensions of the geometry. If η is assumed constant, $\eta(x_R)$ can be used as a reasonable approximation. Alternatively, a linear regression can be considered instead. Selecting an approximate range of x where the buoy is located, a polynomial fit can be performed, so that the $\eta \simeq p_0 x + p_1$. The solution for a constant η is obtained for $p_0 = 0$ and $p_1 = \eta(x_R)$.

In order to algebraically compute an approximated intersection, it is also assumed that $f(\varrho)$ is either constant or linear, resulting in a cylinder or a cone, respectively. Hereafter, the more general solution for a cone is given, with $f(\varrho) = m\varrho + q$, which simplify in a cylinder when $m = 0$. A solution with other $f(\varrho)$ may be achieved, but they are not common application cases, since the vast majority of buoys, if not the totality, is composed of cones and cylinders.

Therefore, solving (12) with $\eta \simeq p_0 x + p_1$ and $f(\varrho) = m\varrho + q$, it is found that the intersection is given by:

$$\varrho = \frac{R_{33}(C(3) - p_1 - p_0 C(1)) - q h(\vartheta)}{R_{13}^2 + R_{23}^2 - 1 + p_0 R_{13} R_{33} + m h(\vartheta)} \tag{13}$$

where

$$\begin{aligned} h = &\cos\vartheta\left(R_{11}\left(R_{13}+p_0 R_{33}\right)+R_{21}R_{23}\right)+\\ &\sin\vartheta\left(R_{12}\left(R_{13}+p_0 R_{33}\right)+R_{22}R_{23}\right) \end{aligned} \tag{14}$$

5. Integration

While Sect. 4 provides tools for transforming WF-coordinates to BF-coordinates, and vice versa, this section provides the mathematical framework to compute 3D-surface integrals by means of simpler 2D-integrals, taking advantage of the representation of the surface in a convenient parametric space. As further discussed in Sect. 6, it is convenient to define integrals in the BF.

A generic axisymmetric geometry with vertical axis is composed of a lateral surface and up to two circular lids, one at the bottom and one at the top. The lateral surface can be described through cylindrical coordinates, as in Sect. 5.1.1, while the lids through polar coordinates, as in Sect. 5.1.2.

The geometry is described in the BF $(\hat{x}, \hat{y}, \hat{z}) \in \mathbb{R}^3$, then parametrized in $\mathbb{R}^2$, either (ϱ, ϑ) or (r, ϑ), for cylindrical or polar coordinates, respectively. Let us assume a generic change of coordinates $\mathbb{R}^3 \mapsto \mathbb{R}^2 : (\hat{x}, \hat{y}, \hat{z}) \mapsto (u, v)$. Assume to have an integral I of a generic function $g(\hat{x}, \hat{y}, \hat{z})$ over a surface S:

$$I = \iint_S g(\hat{x}, \hat{y}, \hat{z}) dS \tag{15}$$

In order to apply the change of coordinates $(\hat{x}, \hat{y}, \hat{z}) \mapsto (u, v)$, $\|\mathbf{e}_u \times \mathbf{e}_v\|$ must be included:

$$I = \int_u \int_v g(u, v) \|\mathbf{e}_u \times \mathbf{e}_v\| \, du \, dv \tag{16}$$

where $\mathbf{e}_u$ and $\mathbf{e}_v$ are the unity vectors in the u and v direction, respectively:

$$\mathbf{e}_u = \begin{pmatrix} \frac{\partial \hat{x}}{\partial u} \\ \\ \frac{\partial \hat{y}}{\partial u} \\ \\ \frac{\partial \hat{z}}{\partial u} \end{pmatrix} \tag{17}$$

$$\mathbf{e}_v = \begin{pmatrix} \frac{\partial \hat{x}}{\partial v} \\ \\ \frac{\partial \hat{y}}{\partial v} \\ \\ \frac{\partial \hat{z}}{\partial v} \end{pmatrix} \tag{18}$$

5.1. Parametric representations

In the following subsections, cylindrical and polar coordinates are shown, along with their unity vectors and the normal unity vector $\mathbf{n}$, which is useful for some of the following integrals in Sect. 5.2 and 6.

5.1.1. Cylindrical coordinates

$$\begin{cases} \hat{x}(\varrho, \vartheta) = f(\varrho) \cos \vartheta \\ \hat{y}(\varrho, \vartheta) = f(\varrho) \sin \vartheta \\ \hat{z}(\varrho, \vartheta) = \varrho \end{cases} \quad , \quad \vartheta \in [-\pi, \pi) \wedge \varrho \in [\varrho_1, \varrho_2] \tag{19}$$

$$\mathbf{e}_\varrho = \begin{pmatrix} f(\varrho)' \cos \vartheta \\ f(\varrho)' \sin \vartheta \\ 1 \end{pmatrix} \tag{20}$$

$$\mathbf{e}_\vartheta = \begin{pmatrix} -f(\varrho)\sin\vartheta \\ f(\varrho)\cos\vartheta \\ 0 \end{pmatrix} \tag{21}$$

$$\mathbf{e}_\varrho \times \mathbf{e}_\vartheta = f(\varrho) \begin{pmatrix} -\cos\vartheta \\ -\sin\vartheta \\ f(\varrho)' \end{pmatrix} \tag{22}$$

$$\|\mathbf{e}_\varrho \times \mathbf{e}_\vartheta\| = f(\varrho)\sqrt{1 + f(\varrho)'^2} \tag{23}$$

Where $f(\varrho)' = \frac{df(\varrho)}{d\varrho}$. Note that for a cone and a cylinder, $f(\varrho)'$ is extremely simple, being either constant or zero, respectively. Finally, using (20) to (23), it is also possible to compute the unity vector normal to the surface, pointing inwards, as

$$\mathbf{n} = \frac{\mathbf{e}_\varrho \times \mathbf{e}_\vartheta}{\|\mathbf{e}_\varrho \times \mathbf{e}_\vartheta\|} = \frac{1}{\cancel{f(\varrho)}\sqrt{1 + f(\varrho)'^2}} \cancel{f(\varrho)} \begin{pmatrix} -\cos\vartheta \\ -\sin\vartheta \\ f(\varrho)' \end{pmatrix} = \frac{1}{\sqrt{1 + f(\varrho)'^2}} \begin{pmatrix} -\cos\vartheta \\ -\sin\vartheta \\ f(\varrho)' \end{pmatrix} \tag{24}$$

From equations (15), (16), (23), and (24), it follows that:

$$\iint_S \mathbf{n} dS = \int_\vartheta \int_\varrho \cancel{\|\mathbf{e}_\varrho \times \mathbf{e}_\vartheta\|} \frac{\mathbf{e}_\varrho \times \mathbf{e}_\vartheta}{\cancel{\|\mathbf{e}_\varrho \times \mathbf{e}_\vartheta\|}} d\varrho \, d\vartheta \tag{25}$$

5.1.2. Polar coordinates

$$\begin{cases} \hat{x}(r,\vartheta) = r\cos\vartheta \\ \hat{y}(r,\vartheta) = r\sin\vartheta \\ \hat{z}(r,\vartheta) = \hat{z}_0 \end{cases} \quad , \quad \vartheta \in [-\pi, \pi) \wedge r \in [0, R] \ , \quad \hat{z}_0 = \text{const} \tag{26}$$

$$\mathbf{e}_r \times \mathbf{e}_\vartheta = \begin{pmatrix} \cos\vartheta \\ \sin\vartheta \\ 0 \end{pmatrix} \times \begin{pmatrix} -r\sin\vartheta \\ r\cos\vartheta \\ 0 \end{pmatrix} = r \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{27}$$

$$\|\mathbf{e}_r \times \mathbf{e}_\vartheta\| = r \tag{28}$$

$$\mathbf{n} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{29}$$

Note that $\mathbf{n}$ should point inwards. In case of bottom and top discs, the top one will have the opposite normal, i.e. $\mathbf{n}_{bot} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ and $\mathbf{n}_{top} = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}$.

5.2. Geometric properties

Representative integrals for some geometric properties of the body are hereafter proposed. Most of the following quantities can be easily computed with common analytical solutions, which are used in the toolbox as benchmarking to verify the correctness of the numerical implementation.

Note that for a cone and a cylinder, $f(\varrho)'$ is extremely simple, being either constant or zero, respectively. However, following equations are shown for the most general case. Hereafter, a sample geometry with one lateral surface and two discs, one at the bottom and one at the top, is considered.

5.2.1. Surface

The total surface S is composed of the lateral surface S_L plus the bottom (S_{bot}) and top (S_{top}) discs. Lets assume the R_1 and R_2 the radius of the bottom and top discs, respectively. Note that $R_1 = f(\rho_1)$ and $R_2 = f(\rho_2)$. The area of the discs is just the area or a circle. In fact, for the bottom disc, for example:

$$S_{bot} = \iint\limits_{S_{bot}} dS = \int\limits_{\vartheta} \int\limits_{r} \|\mathbf{e}_\vartheta \times \mathbf{e}_r\| \; d\vartheta \; dr = \int\limits_{-\pi}^{\pi} \int\limits_{0}^{R_1} r \; d\vartheta \; dr = \pi R_1^2 \tag{30}$$

For the lateral surface, only the cylindrical part is considered.

$$S_L = \iint\limits_{S_L} dS = \int\limits_{\vartheta} \int\limits_{\varrho} \|\mathbf{e}_\vartheta \times \mathbf{e}_\varrho\| \; d\vartheta \; d\varrho = \int\limits_{\vartheta} \int\limits_{\varrho} f(\varrho)\sqrt{1 + f(\varrho)'^2} \; d\vartheta \; d\varrho \tag{31}$$

5.2.2. Volume

For computing the volume enclosed by S, the integral must be split into S_L, S_{bot} and S_{top}. The volume equivalently can be computed along three directions, i.e. $V = V_{\hat{x}} = V_{\hat{y}} = V_{\hat{z}}$:

$$V = -\iint\limits_{S} n_1 \hat{x} \; dS = -\iint\limits_{S} n_2 \hat{y} \; dS = -\iint\limits_{S} n_3 \hat{z} \; dS \tag{32}$$

where $\mathbf{n} = \begin{pmatrix} n_1 \\ n_2 \\ n_3 \end{pmatrix}$.

For the discs, as shown in (29), n_1 and n_2 are null, while in the vertical direction $\mathbf{n}_{\hat{z},bot} = 1$, and $\mathbf{n}_{\hat{z},top} = -1$. It follows that, according to (25):

$$V_{\hat{x}} = V_{\hat{x},L} + V_{\hat{x},bot} + V_{\hat{x},top} = V_{\hat{x},L} + 0 + 0 \tag{33a}$$

$$V_{\hat{x},L} = -\int\limits_{\vartheta} \int\limits_{\varrho} (\mathbf{e}_\vartheta \times \mathbf{e}_\varrho)_{\hat{x}} \, \hat{x}(\varrho, \vartheta) \; d\varrho \; d\vartheta = -\int\limits_{-\pi}^{\pi} \int\limits_{\varrho_1}^{\varrho_2} -f(\varrho)^2 \cos^2\vartheta \; d\varrho \; d\vartheta \tag{33b}$$

Similarly,

$$V_{\hat{y}} = V_{\hat{y},L} = -\int_{-\pi}^{\pi}\int_{\varrho_1}^{\varrho_2} -f(\varrho)^2 \sin^2\vartheta \; d\varrho \; d\vartheta \tag{34}$$

Finally,

$$V_{\hat{z}} = V_{\hat{z},L} + V_{\hat{z},bot} + V_{\hat{z},top} \tag{35a}$$

$$V_{\hat{z},L} = -\int_{-\pi}^{\pi}\int_{\varrho_1}^{\varrho_2} \varrho f(\varrho) f(\varrho)' \; d\varrho \; d\vartheta \tag{35b}$$

$$V_{\hat{z},bot} = -\int_{-\pi}^{\pi}\int_{0}^{R1} r\varrho_1 \; dr \; d\vartheta = -\pi R_1^2 \varrho_1 \tag{35c}$$

$$V_{\hat{z},top} = -\int_{-\pi}^{\pi}\int_{0}^{R2} -r\varrho_2 \; dr \; d\vartheta = \pi R_2^2 \varrho_2 \tag{35d}$$

5.2.3. Centre of buoyancy

The centre of buoyancy $CoB = (\hat{x}_B, \hat{y}_B, \hat{z}_B)$ is the centre of gravity of the displaced mass of fluid. Therefore, only the submerged part of the body is considered. In particular, the integration limits for the centre of buoyancy are $[\varrho_1, \varrho_0]$, assuming ϱ_0 as the body-frame vertical coordinate of the still water level.

Similar to Sect. 5.2.2, the geometry must be decomposed in lateral, bottom, and top surfaces. The integral to solve are similar to (32):

$$\hat{x}_B = -\frac{1}{2V}\iint_S n_1 \hat{x}^2 \; dS \tag{36a}$$

$$\hat{y}_B = -\frac{1}{2V}\iint_S n_2 \hat{y}^2 \; dS \tag{36b}$$

$$\hat{z}_B = -\frac{1}{2V}\iint_S n_3 \hat{z}^2 \; dS \tag{36c}$$

Since the body is axisymmetric, $\hat{x}_B$ and $\hat{y}_B$ are null. The resulting integral for $\hat{z}_B$ is:

$$\hat{z}_B = -\frac{1}{2V}\left(\int_{-\pi}^{\pi}\int_{\varrho_1}^{\varrho_0} \varrho^2 f(\varrho) f(\varrho)' \; d\varrho \; d\vartheta + \pi R_1^2 \varrho_1^2 - \pi R_2^2 \varrho_0^2\right) \tag{37}$$

6. Nonlinear Froude-Krylov force

The mathematical framework developed in Sect. 4 and 5 is now used to compute NLKF forces in 6-DoFs, which are the result of the gravity force and the integral of the undisturbed pressure field (static and dynamic) over the instantaneous wetted surface. The pressure field (p) is defined, in the WF, according to Airy's theory, with the application of Wheeler stretching [10], as:

$$p(x,z,t) = p_{st} + p_{dy} = -\rho g z + \rho g a \cos\left(\omega t - kx\right) \frac{\cosh\left(k\left(z'+h\right)\right)}{\cosh(kh)} \tag{38}$$

where

- t is the time
- (x, y, z) is the inertial world frame coordinates, with the origin at the still water level, z pointing upwards, x pointing in the direction of wave propagation
- p_{st} and p_{dy} are the static and dynamic pressure, respectively
- ρ is the sea water density
- g is the acceleration of gravity
- a is the wave amplitude
- $\omega = \frac{2\pi}{T}$ is the wave frequency, and T the wave amplitude
- $k = \frac{2\pi}{\lambda}$ is the wave number, and λ the wave length
- h is the water depth
- z' is the change of coordinates required by Wheeler stretching in order to eliminate the free surface boundary condition error: $z' = h\frac{z+h}{\overline{\eta}+h} - h$, where $\overline{\eta}$ is an approximation of the free surface, computed at the reference point (x_R).

Note that $z' + h = h\frac{z+h}{\overline{\eta}+h}$, so that $\frac{\cosh(k(z'+h))}{\cosh(kh)} = \frac{\cosh\left(kh\frac{z+h}{\overline{\eta}+h}\right)}{\cosh(kh)}$.
In infinite water depth conditions, $z' \to (z - \overline{\eta})$, and $\frac{\cosh(k(z'+h))}{\cosh(kh)} \to \exp(k(z-\overline{\eta}))$.
The static FK force is then defined as:

$$\mathbf{F}_{FK_{st}} = \mathbf{F}_g + \iint_{S_w} p_{st}\mathbf{n}\, dS \tag{39}$$

where $\mathbf{F}_g = (0, 0, -mg)'$ is the gravity force, with m the mass of the body, S_w the instantaneous wetted surface, and $\mathbf{n}$ the unity vector normal to the surface.

The dynamic FK force is similarly defined as:

$$\mathbf{F}_{FK_{dy}} = \iint_{S_w} p_{dy}\mathbf{n}\ dS \tag{40}$$

Static and dynamic FK torques are defined as:

$$\mathbf{T}_{FK_{st}} = \mathbf{r}_g \times \mathbf{F}_g + \iint_{S_w} p_{st}\left(\mathbf{r}\times\mathbf{n}\right)\ dS \tag{41}$$

$$\mathbf{T}_{FK_{dy}} = \iint_{S_w} p_{dy}\left(\mathbf{r}\times\mathbf{n}\right)\ dS \tag{42}$$

where $\mathbf{r}$ is the generic position vector, and $\mathbf{r}_g$ is the position vector of the centre of gravity.

Since it is advisable to write the equation of motion in the body-fixed frame (in order to have a constant inertial matrix), the NLFK forces are computed in the BF too. On the other hand, the free surface elevation and the pressure field are defined with respect to the WF. Therefore, two alternative options are available:

a) compute the integrals in the WF, then rotate the forces in the BF;

b) map η and p onto the BF and compute the integrals directly in the BF.

[7] shows that option b) is both simpler and computationally more efficient. The pressure field is mapped onto the body frame, by means of the 4D rotation matrix R, as shown in (5):

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = R \begin{pmatrix} \hat{x}(\varrho,\vartheta) \\ \hat{y}(\varrho,\vartheta) \\ \hat{z}(\varrho,\vartheta) \\ 1 \end{pmatrix} \tag{43}$$

Finally, the upper limit of integration (ϱ_2) should be computed as explained in Sect. 4.3.

Recalling the results provided in Sect. 5, here there are $\mathbf{n}dS$ and $(\mathbf{r}\times\mathbf{n})\,dS$, after the change or coordinate transformation, in polar and cylindrical coordinates:

✓ **Polar**

- $$\mathbf{r} = \begin{cases} \hat{x}(r,\vartheta) = r\cos\vartheta \\ \hat{y}(r,\vartheta) = r\sin\vartheta \\ \hat{z}(r,\vartheta) = \hat{z}_0 \end{cases} \quad , \quad \vartheta \in [-\pi,\pi) \wedge r \in [0,R]\ , \quad \hat{z}_0 = \text{const} \tag{44}$$

- $$\mathbf{n}dS = \mathbf{e}_r \times \mathbf{e}_\vartheta = r \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{45}$$

- $$(\mathbf{r} \times \mathbf{n})\, dS = \mathbf{r} \times (\mathbf{e}_r \times \mathbf{e}_\vartheta) = r^2 \begin{pmatrix} \sin\vartheta \\ -\cos\vartheta \\ 0 \end{pmatrix} \tag{46}$$

✓ **Cylindrical**

- $$\mathbf{r} = \begin{cases} \hat{x}(\varrho, \vartheta) = f(\varrho)\cos\vartheta \\ \hat{y}(\varrho, \vartheta) = f(\varrho)\sin\vartheta \\ \hat{z}(\varrho, \vartheta) = \varrho \end{cases} \quad , \quad \vartheta \in [-\pi, \pi) \wedge \varrho \in [\varrho_1, \varrho_2] \tag{47}$$

- $$\mathbf{n}dS = \mathbf{e}_\varrho \times \mathbf{e}_\vartheta = f(\varrho) \begin{pmatrix} -\cos\vartheta \\ -\sin\vartheta \\ f(\varrho)' \end{pmatrix} \tag{48}$$

- $$(\mathbf{r} \times \mathbf{n})\, dS = \mathbf{r} \times (\mathbf{e}_\varrho \times \mathbf{e}_\vartheta) = \begin{pmatrix} f(\varrho)\sin\vartheta\,(f(\varrho)f(\varrho)' + \varrho) \\ -f(\varrho)\cos\vartheta\,(f(\varrho)f(\varrho)' + \varrho) \\ 0 \end{pmatrix} \tag{49}$$

6.1. Validation in linear conditions

As in Sect. 5.2, it is convenient to use a known benchmark to validate the modelling approach and verify the correctness of implementation. One easy and reliable approach is to compute NLFK forces in linear conditions and compare the results with linear solutions. Extremely linear conditions can be used to ensure the linearity of the solution:

- linear conditions for $\mathbf{F}_{FK_{st}}$ and $\mathbf{T}_{FK_{st}}$: very small displacements and no incoming wave.
- linear conditions for $\mathbf{F}_{FK_{dy}}$ and $\mathbf{T}_{FK_{dy}}$: zero displacement and very small incoming wave.

6.1.1. Hydrostatic stiffness

An infinitesimal displacement δd causes a change in the static FK force and torque δf. The linear hydrostatic stiffness is defined as the ratio between δf and δd. NLFK forces can be computed before and after δd, so that the linear hydrostatic stiffness coefficient can be computed and compared to the known algebraic solutions.

The hydrostatic stiffness K_h is a 6x6 matrix; for vertical axisymmetric geometries, the centre of buoyancy and the centre of gravity are on the axis, so that off-diagonal terms of the K_h are zeros. Finally, the only non-zero elements are $K_h(3,3)$, $K_h(4,4)$, and $K_h(5,5)$.

Let ρ and g be the sea water density and the acceleration of gravity, respectively. Assuming the geometry in equilibrium, the mass m of the body is equal to the mass of the displaced volume of fluid, i.e. $m = \rho V_{sub}$.

Algebraic results for axisymmetric devices are available [11]:

- $K_h(3,3) = \rho g A_{WP}$
- $K_h(4,4) = K_h(5,5) = \rho g \left(I_{WP} - V_{sub}\overline{\mathsf{BG}} \right)$

where A_{WP} is the water plane area, I_{WP} is the geometric moment of inertia of the water plane area, and $\overline{\mathsf{BG}}$ is the distance between the centres of gravity and buoyancy.

The coefficients of K_h can also be computed through integration [11]:

- The **heave** stiffness is computed as:

$$K_h(3,3) = \rho g \iint_{S_w} n_3 \, dS = \rho g \int_{-\pi}^{\pi} \int_{\varrho_1}^{\varrho_0} f(\varrho) f(\varrho)' \, d\varrho \, d\vartheta + \rho g \pi R_1^2 \tag{50}$$

  Note that (50) is equivalent to $\rho g \pi R_0^2 = \rho g A_{WP}$.

- Let us verify that $K_h(3,4) = 0$:

$$\begin{aligned} K_h(3,4) =& \rho g \iint_{S_w} \hat{y} n_3 \, dS = \\ & \rho g \int_{-\pi}^{\pi} \int_{\varrho_1}^{\varrho_0} f(\varrho)^2 f(\varrho)' \sin\vartheta \, d\varrho \, d\vartheta + \rho g \int_{-\pi}^{\pi} \int_{0}^{R_1} r^2 \sin\vartheta \, dr \, d\vartheta = 0 + 0 \end{aligned} \tag{51}$$

  Likewise, $K_(3,5) = -\rho g \iint_{S_w} \hat{x} n_3 \, dS = 0$.

- The **roll** stiffness is computed as:

$$\begin{aligned} K_h(4,4) =& \rho g \iint_{S_w} \hat{y}^2 n_3 \, dS + \rho g V_{sub} \hat{z}_B - mg\hat{z}_G = \\ & \rho g \int_{-\pi}^{\pi} \int_{\varrho_1}^{\varrho_0} f(\varrho)^3 f(\varrho)' \sin^2\vartheta \, d\varrho \, d\vartheta + \rho g \pi \frac{R_1^4}{4} + \rho g V_{sub} \hat{z}_B - mg\hat{z}_G \end{aligned} \tag{52}$$

- Likewise, the **pitch** stiffness, which for an axisymmetric geometry is the same as the **roll** stiffness, is computed as:

$$\begin{aligned} K_h(5,5) =& \rho g \iint_{S_w} \hat{x}^2 n_3 \; dS + \rho g V_{sub} \hat{z}_B - mg\hat{z}_G = \\ & \rho g \int_{-\pi}^{\pi} \int_{\varrho_1}^{\varrho_0} f(\varrho)^3 f(\varrho)' \cos^2\vartheta \; d\varrho \; d\vartheta + \rho g \pi \frac{R_1^4}{4} + \rho g V_{sub} \hat{z}_B - mg\hat{z}_G \end{aligned} \tag{53}$$

- The coupled **roll-pitch** stiffness, is computed as:

$$\begin{aligned} K_h(4,5) =& \rho g \iint_{S_w} \hat{x}\hat{y} n_3 \; dS = \\ & \rho g \int_{-\pi}^{\pi} \int_{\varrho_1}^{\varrho_0} f(\varrho)^3 f(\varrho)' \sin\vartheta\cos\vartheta \; d\varrho \; d\vartheta + \rho g \int_{-\pi}^{\pi} \int_{0}^{R_1} r^3 \sin\vartheta\cos\vartheta \; dr \; d\vartheta = 0 + 0 \end{aligned} \tag{54}$$

6.1.2. Dynamic Froude-Krylov force

Forcing zero displacement of the floater, a set of regular waves can be sent, and the relative amplitude and phase difference with respect to the incoming wave can be computed. If the waves are very small, the results must overlap with the output from any Boundary Element Method (BEM) software, like Nemoh. Figure 7 shows an example of validation of computation of NLFK in surge for a cylinder of 2m radius, 5m radius, using waves of 0.006m amplitude.

Acknowledgement

This work has received funding from the European Research Council under the Horizon 2020 Programme (H2020-MSCA-IF-2018)/ grant agreement no 832140.

References

[1] G. Giorgi, Nonlinear Froude-Krylov Matlab demonstration toolbox, doi:10.5281/zenodo.3517140, 2019.
[2] C. Commons, CCBYSA, URL https://creativecommons.org/licenses/by-sa/4.0/legalcode, 2019.
[3] G. Giorgi, J. V. Ringwood, Nonlinear Froude-Krylov and viscous drag representations for wave energy converters in the computation/fidelity continuum, Ocean Engineering 141 (April) (2017) 164–175, ISSN 00298018, doi:10.1016/j.oceaneng.2017.06.030.
[4] M. Penalba, G. Giorgi, J. V. Ringwood, Mathematical modelling of wave energy converters: a review of nonlinear approaches, Renewable and Sustainable Energy Reviews 78 (2017) 1188–1207, doi:10.1016/j.rser.2016.11.137.
[5] G. Giorgi, J. V. Ringwood, Comparing nonlinear hydrodynamic forces in heaving point absorbers and oscillating wave surge converters, Journal of Ocean Engineering and Marine Energy 4 (1) (2018) 25–35, doi:https://doi.org/10.1007/s40722-017-0098-2.

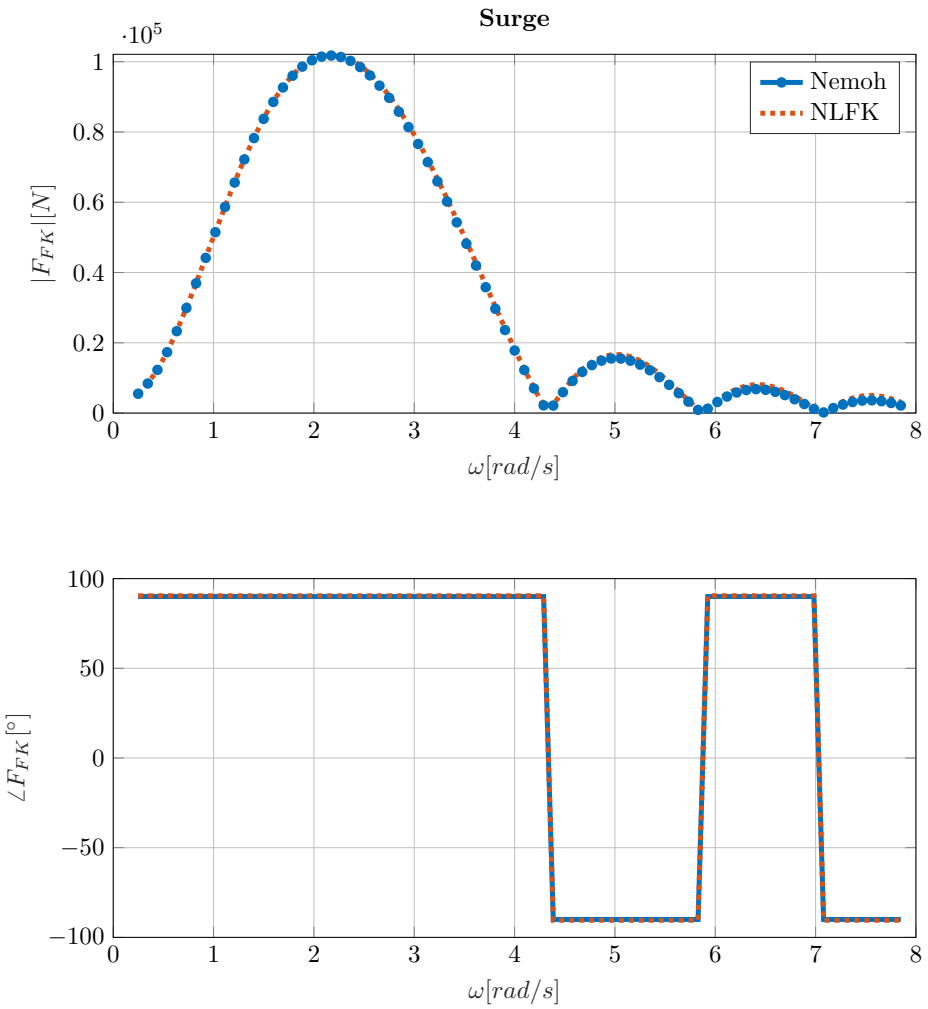

Figure 7: Example of validation of nonlinear Froude-Krylov force in surge for a cylinder.

[6] J.-C. Gilloteaux, Simulation de mouvements de grande amplitude. Application {à} la r{é}cup{é}ration de l'{é}nergie des vagues. .

[7] G. Giorgi, J. V. Ringwood, Analytical representation of nonlinear Froude-Krylov forces for 3-DoF point absorbing wave energy devices, Ocean Engineering 164 (2018) (2018) 749–759, doi:10.1016/j.oceaneng.2018.07.020.

[8] G. Giorgi, J. V. Ringwood, A Compact 6-DoF Nonlinear Wave Energy Device Model for Power Assessment and Control Investigations, IEEE Transactions on Sustainable Energy doi:10.1109/TSTE.2018.2826578.

[9] G. Giorgi, J. V. Ringwood, Articulating parametric resonance for an OWC spar buoy in regular and irregular waves, Journal of Ocean Engineering and Marine Energy 4 (4) (2018) 311–322, ISSN 21986452, doi:10.1007/s40722-018-0124-z.

[10] G. Giorgi, J. V. Ringwood, Relevance of pressure field accuracy for nonlinear FroudeâĂŞKrylov force calculations for wave energy devices, Journal of Ocean Engineering and Marine Energy 4 (1) (2018) 57–71, ISSN 21986452, doi:10.1007/s40722-017-0107-5.

[11] T. I. Fossen, Handbook of marine craft hydrodynamics and motion control, John Wiley & Sons, 2011.

[12] M. Penalba, T. E. Kelly, J. V. Ringwood, Using NEMOH for Modelling Wave Energy Converters : A Comparative Study with WAMIT, Proceedings of the 12th European Wave and Tidal Energy Conference .