A robust algorithm for computational floating body dynamics

J. Roenby^{1,3*}, S. Aliyar², and H. Bredmose²

Stromning Aps, Luftmarinegade 62, 1432 København K, DK
 Department of Wind and Energy Systems, Technical University of Denmark, Nils Koppels Alle, Kgs. Lyngby 2800, DK
 Department of Science and Environment, Roskilde University, Universitetsvej 1, 4000 Roskilde, DK
 *johan@ruc.dk

October 3, 2023

Abstract

We present a non-iterative algorithm, FloatStepper, for coupling the motion of a rigid body and an incompressible fluid in computational fluid dynamics (CFD) simulations. The purpose of the algorithm is to remove the so-called added mass instability problem, which may arise when a light floating body interacts with a heavy fluid. The idea underlying the presented coupling method is to precede every computational time step by a series of prescribed probe body motions in which the fluid response is determined, thus revealing the decomposition of the net force and torque into two components: 1) An added mass contribution proportional to the instantaneous body acceleration, and 2) all other forces and torques. The algorithm is implemented and released as an open source extension module to the widely used CFD toolbox, OpenFOAM, as an alternative to the existing body motion solvers. The accuracy of the algorithm is investigated with several single-phase and two-phase flow benchmark cases. The benchmarks demonstrate excellent stability properties, allowing simulations even with massless bodies. They also highlight aspects of the implementation, such as the mesh motion method, where more work is needed to further enhance the flexibility and predictive capabilities of the code.

1 Introduction

Accurate modelling of floating body motion is important for the design of offshore structures. This requires a robust numerical approach to both free surface flow and wave-structure interaction. While many basic response effects are well described by linear and second-order radiation-diffraction theory [1], this relies on the assumption of small wave steepness and small body motion. To describe response effects from large waves, slamming from breaking waves, green water flow on the structure and viscous damping, more accurate modelling is needed.

During the last decades, Computational Fluid Dynamics (CFD) for free surface flow has been matured to a level where it is now a viable solution for engineering calculation of design wave events. Among the various CFD methodologies, the Finite Volume Method (FVM) in combination with the Volume of Fluid technique (VoF) for the free surface treatment [2], has shown robust performance with ability to calculate breaking wave loads on e.g. monopiles, see for example [3, 4, 5]. Several publications have shown good comparisons to measured force and pressure for such breaking wave impacts, e.g. [6]. Compared to potential flow solvers, CFD allows the fluid topology to change, for example through release of droplets. Much research effort has gone into the detailed VoF schemes to avoid numerical smearing of the air-water interface. Well–known methods after the original paper of Hirt & Nicols [2] include the works in [7, 8], as well as the isoAdvector scheme [9], which can be applied on unstructured meshes.

Given the successful results for fixed structures, application of finite volume CFD to floating body motion appears to be a natural next development. In principle, the body motion can be treated by calculation of its acceleration in each time step through Newtons second law, with input of the integrated fluid pressure as a force on the right hand side. Several studies with FVM-VoF based floating body CFD have been published in recent years, especially within ship motion, wave energy generation and floating wind turbine motion see e.g. [10, 11, 12, 13] and [14]. A thorough review on various solver types is given by Windt et al. [15]. Further, Ransley et al. [16] presented a comparative study for the response of focused wave groups for a hemispherical-bottomed buoy and a truncated cylinder with a cylindrical moon-pool with both potential flow solvers and CFD. A straight-forward implementation of the above steps, however, have shown to be unstable for bodies with low structural mass. The inherent problem is that acceleration of the body requires simultaneous acceleration of the surrounding fluid, such that extra fluid mass must be added to the body mass to achieve the truly needed force through Newtons second law. This fluid mass is denoted 'added mass' and is generally a 6×6 matrix that depends on the instantaneous fluid topology.

The need for a proper treatment of added mass in floating body CFD has been discussed already by Söding [17] in a conference paper, that appears to have only little recognition. Bettle [18] discussed the stability problem in the context of CFD for submarine maneuvering and devised a coupling algorithm with iterations between body and fluid motion. A floating body solver along the lines of Bettle's work is found in the widely used open source CFD code, OpenFOAM. This was improved in the work of Dunbar et al. [19] and Chow et al. [20] using dynamic relaxation techniques, and by Bruinsma et al. [21] who stabilised solutions by relaxing fluid pressure in the iteration loop. The latter concluded that more work is needed to achieve a robust solution for the added mass problem, since the stabilization techniques lead to larger computational effort. Further steps in the solution of the added mass problem have been proposed by Devolder et al. [22] in terms of an acceleration technique for the added mass iterations with 1 degree of freedom (DoF), and by Veldman et al. [23], in terms of an approximate initial added mass term.

While the iterative methods can give accurate results upon convergence, a robust solution of the equations of motion, requires separation of the added mass force from the overall fluid force lumping the body and added mass together to properly isolate the acceleration in the force equation. This is the core idea of the present paper. We develop an algorithm, where the added mass is determined explicitly in each time step and thus allows an accurate and direct calculation of the true body acceleration without the need for outer iterations. We implement the algorithm and demonstrate its robustness through a series of numerical experiments.

It is worth to note, that the idea of an explicit added mass matrix in floating body modelling has been presented also by other researchers. In this respect, our approach has strong similarities to the algorithm outlined by Söding [17] and applied by Shigonov et al [24] for aircraft landing on water in three degrees of freedom. In Söding's paper, the need for an explicit added mass matrix for numerical stability is explained and an iteration based method for its determination is formulated. In a study by Meyer et al. [25] this algorithm was applied to calculate the motion of a yacht in head waves. To the best of our knowledge, no thorough demonstration of the stability properties exists in the literature, and no open source CFD implementation is available to the scientific community.

The contribution of the present work is to develop and implement in a fully parallelised unstructured FVM CFD code, OpenFOAM, a 6–DoF added mass aware algorithm which in contrast to earlier works is free of outer iteration loops. We analyse and demonstrate its stability properties and validate it against five test cases. Two of these have analytical solutions, a rising disc and a wiggling ellipse, and two contain comparison to experimental results, namely a freely floating box and a moored box in regular waves. Our hope is that the new robust method, and the release of the implementation as an OpenFOAM extension module [26], will provide a simpler approach to CFD simulations of floating body problems.

2 The added mass instability problem

We consider the numerical coupling of a floating rigid body with a surrounding incompressible fluid. The fluid may either be a single fluid or two immiscible fluids separated by a sharp fluid interface. It may either be inviscid or viscous with slip or no-slip boundary condition on the body so the instantaneous distribution of fluid pressure — and possibly shear stress — exerts a force and a torque on the body.

When an external force, **F**, such as gravity or a spring works on a rigid body immersed in fluid, the body will accelerate and so must the surrounding fluid to accommodate the body displacement. In a small time

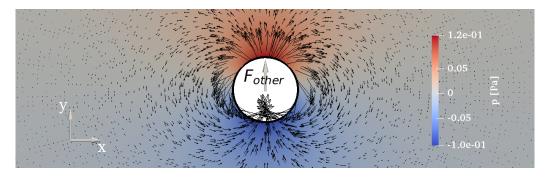


Figure 1: Pressure and velocity field around circular body exposed to a force along the y-axis.

step, δt , the momentum of the body-fluid system will increase by $\mathbf{F}\delta t$ and from a numerical point of view the difficulty is that we do not know a priori how this increase in system momentum is distributed between the fluid and the body. It is the task of a coupling algorithm to find this distribution and advance the system accordingly.

For illustrative purposes let us consider a rigid cylinder floating in 2D ideal fluid of uniform mass density ρ_f , as shown in Figure 1. The body has radius R and uniform mass density ρ_b , and hence mass (per unit length) $m_b = \rho_b \pi R^2$. If the body is exposed to a net force, F, along the y-axis, this can be written as

$$F = F_{\text{other}} - m_a a,\tag{1}$$

where m_a is the added mass of the body, a is the instantaneous y-acceleration of the body, and F_{other} represents all other forces on the body (gravity, buoyancy, mooring lines etc.). In this simple example, the added mass is known a priori to be $m_a = \rho_f \pi R^2$. Equating the total force, F, to $m_b a$, and isolating a, we get the body acceleration,

$$a = \frac{F_{\text{other}}}{m_a + m_b}. (2)$$

In computational fluid dynamics simulations we often do not know m_a and/or F_{other} . Therefore, in partitioned coupling algorithms, we typically resort to iteration between

- 1. Calculating the body acceleration as $a = F/m_b$ (or some relaxed variant of this), where F includes the force from the surrounding fluid flow, and
- 2. Calculating the fluid flow and force on the body, F, resulting from the body acceleration, a.

The hope is that iterations between these two steps will eventually converge to the physically correct body acceleration and fluid response, which is assumed to be reached when reiteration no longer changes the results (to within a tolerance). It is, however, well–known that this iterative procedure is unstable when the body mass is smaller than the added mass [27].

Let us first consider a loose body-fluid coupling algorithm without any iterations. Each time step contains a single update of the body state followed by an update of the fluid state. We assume the only force on our circular body is gravity, $\mathbf{g} = -g\hat{\mathbf{y}}$, and so F_{other} is constant in time. The body state is then represented by the body position and velocity $(\mathbf{x}_b, \mathbf{v}_b)$, here restricted to motion along the y-axis. The fluid state is represented by the velocity field and pressure field (\mathbf{u}, p) . The algorithm could look as sketched in Algorithm 1. In the fluid initialisation in Step 1 and in Step 7, it is vital to ensure that proper boundary conditions are specified for the velocity and pressure fields on the body boundary since these contain the coupling between body and fluid (will be detailed in Sections 3.1.1 and 4.6).

In Step 4, we know – but for now ignore – that part of the force experienced by the body is due to its instantaneous acceleration, cf. Eqn. (1). Even if we did not know the specific values of F_{other} and m_a , we can still explore the implications of ignoring the added mass force in Step 4. Using Eqn. (1), we have

$$a_{n+1} = \frac{F_{\text{other}}}{m_b} - \frac{m_a}{m_b} a_n,\tag{3}$$

Algorithm 1 A simple loose coupling algorithm.

- 1: Initialise body state, $(\mathbf{x}_b, \mathbf{v}_b)$, and fluid state, (\mathbf{u}, p) .
- 2: Increment time by Δt .
- 3: Calculate the net force, \mathbf{F} , on the body including p integrated over its surface
- 4: Calculate the body acceleration using Newton's 2nd law: $\mathbf{a} = \mathbf{F}/m_b$
- 5: Numerically integrate **a** to get the updated velocity $\mathbf{v}_h^{\text{new}}$, and position $\mathbf{x}_h^{\text{new}}$.
- 6: Update body state, $(\mathbf{x}_b, \mathbf{v}_b) = (\mathbf{x}_b^{\text{new}}, \mathbf{v}_b^{\text{new}})$, and update mesh accordingly.
- 7: Update fluid boundary conditions on body in correspondence with calculated body state and acceleration.
- 8: Update fluid state, (\mathbf{u}, p) , using the fluid solver.
- 9: If end time reached, stop, otherwise go to Step 2.

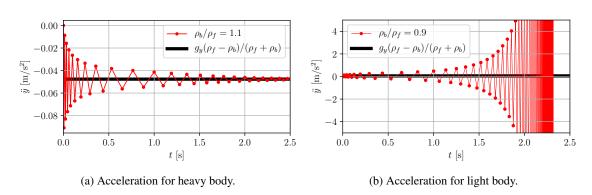


Figure 2: Convergence/divergence of body acceleration in simulation of a circular body of radius R = 1 m in gravity $g_y = -1$ m/s², with density higher/lower than the surrounding inviscid, incompressible fluid ($\rho_f = 1$ kg/m³). Black line marks theoretical value. Outer domain boundary with slip placed at 40R in both cases.

where the subscript n indicates time step. If we call $F_{\text{other}}/m_b = a_0$, insert the corresponding expression for a_n in terms of a_{n-1} , and so forth until we reach n = 0, we get

$$a_{n+1} = a_0 \sum_{k=0}^{n} \left(-\frac{m_a}{m_b} \right)^k \to \begin{cases} \frac{F_{\text{other}}}{m_b + m_a} & \text{if } m_a < m_b, \\ \pm \infty & \text{if } m_a > m_b, \end{cases}$$

$$(4)$$

i.e. an alternating geometric series bound to diverge in an oscillating manner when the added mass exceeds the body mass. This is the added mass instability in a nutshell. It is an inherent problem in any partitioned coupling mechanism [28], and many codes exhibit the instability. This also includes the most widely used open source CFD code, OpenFOAM [29, 22, 19], which we use in this work as our implementation platform. Figure 2 illustrates the change from stable to unstable solver behaviour when the body becomes lighter than the surrounding fluid in the case of a circular body accelerating in a fluid due to gravity. The simulations were run with OpenFOAM's interFoam solver for the fluid motion [8] and the sixDoF-RigidBodyMotion module [30] for body motion.

To circumvent the added mass instability, many codes including OpenFOAM, introduce an outer corrector loop for stronger coupling between body and fluid solution within each time step. In these new iterations, an under-relaxed acceleration is used as shown in Algorithm 2.

The introduction of the acceleration relaxation factor, $\gamma \in [0, 1]$, modifies the iterative process in Eqn. (3) to

$$a_{n+1} = \gamma \left(a_0 - \frac{m_a}{m_b} a_n \right) + (1 - \gamma) a_n,$$
 (5)

where the subscript is now an iteration counter rather than a time step counter. Tracking this iterative

Algorithm 2 Strong coupling algorithm with outer corrections and under-relaxation.

- 1: Set number of outer correctors, N_{OC} , acceleration relaxation, $\gamma \in [0, 1]$ and initial body acceleration, **a**.
- 2: Initialise body state, $(\mathbf{x}_b, \mathbf{v}_b)$, and fluid state, (\mathbf{u}, p) .
- 3: Increment time by Δt
- 4: Calculate the net force, \mathbf{F} , on the body including p integrated over its surface
- 5: Store the body acceleration from previous iteration, $\mathbf{a}^{\text{prev}} = \mathbf{a}$.
- 6: Calculate body acceleration using Newton's 2nd law: $\mathbf{a}^* = \mathbf{F}/m_b$
- 7: Under–relax body acceleration, $\mathbf{a} = \gamma \mathbf{a}^* + (1 \gamma)\mathbf{a}^{\text{prev}}$
- 8: Numerically integrate **a** to get the updated body velocity $\mathbf{v}_b^{\text{new}}$, and position $\mathbf{x}_b^{\text{new}}$.
- 9: Update body state, $(\mathbf{x}_b, \mathbf{v}_b) = (\mathbf{x}_b^{\text{new}}, \mathbf{v}_b^{\text{new}})$ and mesh.
- 10: Update fluid boundary conditions on body in correspondence with calculated body state and acceleration.
- 11: Update fluid state, (\mathbf{u}, p) , using the fluid solver.
- 12: If Steps 4-11 were performed less than $N_{\rm OC}$ times, go to Step 4, otherwise continue.
- 13: If end time reached, stop, otherwise go to Step 3.

equation back to the zero'th iteration, we get the modified geometric series,

$$a_n = \gamma a_0 \sum_{k=0}^n \left[1 - \gamma \left(1 + \frac{m_a}{m_b} \right) \right]^k. \tag{6}$$

This converges if the square bracket has absolute value smaller than one, leading to the stability criterion,

$$\gamma < \gamma_c = \frac{2}{1 + m_a/m_b},\tag{7}$$

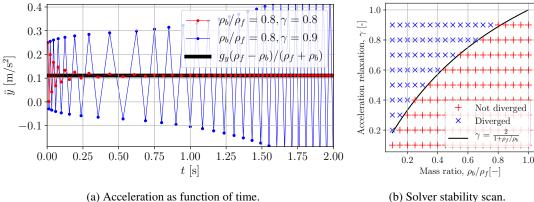
depending on the instantaneous ratio between added and body mass. This stability criterion was also stated in [17] and derived in a slightly different manner in [22]. It is clear that the under-relaxation procedure enables convergence for body mass smaller than the added mass. Figure 3a demonstrates the change to convergence for $\gamma < \gamma_c$ in an OpenFOAM simulation of a light circular body rising in a heavy inviscid fluid. Figure 3b shows the stability criterion in Eqn. (7) plotted together with numerically obtained convergence tests using OpenFOAM for a scan of acceleration relaxation values and density ratios. This clearly demonstrates that the theoretically obtained convergence criterion indeed applies to the specific body–fluid coupling implemented in OpenFOAM.

When the body-to-fluid density ratio approaches zero, so does γ_c , and hence also the change in acceleration between iterations diminishes, cf. Eqn. (5). In other words, a larger number of iterations is necessary for light bodies. This becomes computationally expensive, since each outer iteration involves a full CFD update of the fluid state. Also, for problems involving a body near a free surface, or body motion close to other boundaries, the added mass will vary in time, and so will γ_c therefore. There exist methods for dynamic relaxation which have been used with some success [19, 20]. Here we take a different approach: Instead of trying to fix the trial-and-error approach to finding a consistent body acceleration, we attempt to exploit the freedom CFD grants us to directly measure the zero-acceleration force, F_{other} , in a virtual time step before the actual time step is taken. Likewise, we use a simplified version of the CFD solver, to obtain the instantaneous added mass. This allows us to calculate the body acceleration directly, which is then used to find the new body velocity and position. In this way, we obtain a coupling mechanism that is completely freed from the added mass instability and outer iterations.

3 Governing equations

3.1 Fluid motion

The fluid motion is assumed to be governed by the incompressible Navier-Stokes equations. It may be a single phase with constant density, ρ_f , or two immiscible phases separated by a sharp fluid interface across



(b) Solver stability scan.

Figure 3: (a): Circle with $\rho_b = 0.8 \text{ kg/m}^3$ in fluid with $\rho_f = 1 \text{ kg/m}^3$ in gravity $g_v = -1 \text{ m/s}^2$. Illustration of transition from convergence to divergence when acceleration relaxation, γ , is changed from 0.8 to 0.9 with $\gamma_c \approx 0.0889$. (b): 171 simulations with $\rho_b/\rho_f = [0.1:0.05:1]$ and $\gamma = [0.1:0.1:0.9]$. Blue dot indicates simulation crash due to divergence. Red dot means simulation reached t = 3s without crash. Black curve is the theoretical limit from Eqn. (7).

which the fluid mass density jumps from the value ρ^+ in the reference fluid to ρ^- in the other fluid. The fluid equations of motion are then

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{8a}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{8b}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{8b}$$

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \mathbf{u}) = -\nabla p + \rho \mathbf{g} + \nabla \cdot (\mu \nabla \mathbf{u}) + \mathbf{f}, \tag{8c}$$

where g is the gravity vector and f represents other forces such as surface tension or external forces. The fluid state at any time is represented by the density field, ρ , the pressure field, p, and the three components of the velocity field, **u**. As ρ , the dynamic viscosity, μ , takes constant values, μ^+ and μ^- in the the two fluids. Both ρ and the dynamic viscosity, μ , may be expressed in terms of an indicator function, $H(\mathbf{x},t)$, which is a Heaviside function taking the value 1 in the reference fluid and 0 in the other:

$$\rho = \rho^{+}H + \rho^{-}(1 - H)$$
 and $\mu = \mu^{+}H + \mu^{-}(1 - H)$, (9)

Eqn. (8a) can then be replaced by the equivalent equation

$$\frac{\partial H}{\partial t} + \nabla \cdot (H\mathbf{u}) = 0. \tag{10}$$

This is the starting point for derivations of VoF schemes used to track the sharp fluid interface. For incompressible single–phase flows ρ is constant in the whole domain, and Eqn. (10) is trivially satisfied.

Above we have used (\mathbf{u}, p) to specify the state of the fluid. This is sufficient for single phase flows. For two-phase flows with a sharp fluid interface, we need to augment \mathbf{u} and p with a description of the instantaneous fluid interface position. Since this is encoded in the density field, jumping from one value to another at the interface, we will henceforth use the triplet (ρ, \mathbf{u}, ρ) to represent the fluid state.

3.1.1 Boundary conditions

The gradient of the indicator field, H, is a 3-dimensional Dirac δ -function which is zero everywhere except on the fluid interface, where it is infinite and points along the interface normal, $\hat{\bf n}_I$, into the reference fluid,

$$\nabla H = \hat{\mathbf{n}}_I \delta(\mathbf{x} - \mathbf{x}_I). \tag{11}$$

On domain boundaries one can therefore specify a desired contact angle, or rather interface orientation, by specifying ∇H . Often this angle is not of practical interest and one simply uses a zero gradient Neumann boundary condition, $\hat{\mathbf{n}}_b \cdot \nabla H = 0$, on walls and outlets, where $\hat{\mathbf{n}}_b$ is the unit normal of the boundary. For inlet boundaries, one must use a Dirichlet boundary condition to specify the interface position of the inflowing fluid

For the velocity field, \mathbf{u} , we either use a slip condition or a no-slip condition on walls. For slip we must have $\hat{\mathbf{n}}_b \cdot \mathbf{u} = \hat{\mathbf{n}}_b \cdot \mathbf{v}_b$, where \mathbf{v}_b is the velocity of the boundary point. The tangential velocity component can be written $\mathbf{u}_t = (I_3 - \hat{\mathbf{n}}_b \hat{\mathbf{n}}_b)\mathbf{u}$, where $\hat{\mathbf{n}}_b \hat{\mathbf{n}}_b$ is the outer product of the vector $\hat{\mathbf{n}}_b$ with itself. For slip a Neumann condition can be applied for the tangential component, $\hat{\mathbf{n}}_b \cdot \nabla \mathbf{u}_t = 0$. In case of no-slip, the full velocity on the boundary must follow the velocity of the boundary point, $\mathbf{u} = \mathbf{v}_b$. Again, for inlet boundaries we must specify the velocity value and for domain outlets we can use $\hat{\mathbf{n}}_b \cdot \nabla \mathbf{u} = \mathbf{0}$.

For the pressure, boundary conditions are formulated by requiring consistency with the Navier-Stokes equations Eqn. (8c) also on the boundaries of fixed and moving walls. Isolating the pressure gradient and dotting with the boundary normal, we get the Neumann boundary condition,

$$\hat{\mathbf{n}}_b \cdot \nabla p = -\hat{\mathbf{n}}_b \cdot \left(\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \mathbf{u}) - \rho \mathbf{g} - \nabla \cdot (\mu \nabla \mathbf{u}) - \mathbf{f} \right). \tag{12}$$

As we will see below, if the boundary is accelerating, care must be taken to encode the acceleration correctly in the implementation of the boundary condition.

3.2 Rigid body motion

For the body equations of motion, we start by defining a body-fixed coordinate system centred at some chosen point, $\mathbf{x}_0(t)$, and with coordinate axes spanned by the three mutually orthogonal unit vectors $\mathbf{q}_1(t)$, $\mathbf{q}_2(t)$ and $\mathbf{q}_3(t)$. Together, \mathbf{x}_0 and the orthogonal orientation matrix, $Q = [\mathbf{q}_1 \ \mathbf{q}_2 \ \mathbf{q}_3]$, define the instantaneous configuration of the body. For convenience, we will sometimes denote the body configuration with the shorthand notation,

$$x_b = [\mathbf{x}_0 \ Q]. \tag{13}$$

The body translational velocity is

$$\dot{\mathbf{x}}_0 = \mathbf{v}_0(t),\tag{14}$$

and the rotational velocity is given in laboratory coordinates by $\boldsymbol{\omega}(t) = [\omega_1 \ \omega_2 \ \omega_3]^T$ such that

$$\dot{\mathbf{q}}_i = \boldsymbol{\omega} \times \mathbf{q}_i \quad \text{for} \quad i = 1, 2, 3.$$
 (15)

If we define for any vector $\mathbf{v} = [v_1 \ v_2 \ v_3]^T$ the skew–symmetric matrix,

$$\mathbf{v} \times = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}, \tag{16}$$

then Eqn. (15) can also be written in matrix form as

$$\dot{Q} = \boldsymbol{\omega} \times Q. \tag{17}$$

The acceleration equations are given by Newton's 2nd law

$$\frac{d\mathbf{p}}{dt} = \mathbf{F}, \text{ and } \frac{d\mathbf{L}}{dt} = \mathbf{\tau},$$
 (18)

where the body linear and angular momentum, force and torque (with respect to the laboratory frame origin) are, respectively,

$$\mathbf{p} = \int_{\mathcal{D}} \rho \mathbf{v} dV, \quad \mathbf{L} = \int_{\mathcal{D}} \mathbf{x} \times \rho \mathbf{v} dV, \quad \mathbf{F} = \int_{\mathcal{D}} \mathbf{f}_{\text{ext}} dV, \quad \text{and} \quad \mathbf{\tau} = \int_{\mathcal{D}} \mathbf{x} \times \mathbf{f}_{\text{ext}} dV.$$
 (19)

Here \mathscr{B} denotes the body region, and $\mathbf{f}_{\mathrm{ext}}(\mathbf{x},t)$ denotes the external force on the body. The velocity in (and on the surface of) \mathscr{B} is given by

$$\mathbf{v} = \mathbf{v}_0 + \boldsymbol{\omega} \times (\mathbf{x} - \mathbf{x}_0). \tag{20}$$

We define a body-fixed coordinate system such that the representation of a point in the laboratory frame, x, and in the body-fixed coordinates, \tilde{x} , are related by

$$\tilde{\mathbf{x}} = Q^T(\mathbf{x} - \mathbf{x}_0) \quad \Leftrightarrow \quad \mathbf{x} = \mathbf{x}_0 + Q\tilde{\mathbf{x}}.$$
 (21)

Here we have exploited that $Q^{-1} = Q^T$ for the orthogonal matrix, Q. Since the body is rigid, the mass density, ρ , in \mathcal{B} is constant in time as viewed from the body-fixed coordinates, i.e. $\rho = \rho(\tilde{\mathbf{x}})$. The body volume, mass, centre of mass and moment of inertia are, respectively,

$$V_b = \int_{\mathscr{B}} dV, \quad m_b = \int_{\mathscr{B}} \rho dV, \quad \mathbf{x}_{cm}(t) = \frac{1}{V_b} \int_{\mathscr{B}} \rho \mathbf{x} dV, \quad \tilde{I}_0 = \int_{\mathscr{B}} \rho(\tilde{\mathbf{x}}) (|\tilde{\mathbf{x}}|^2 I_3 - \tilde{\mathbf{x}} \tilde{\mathbf{x}}) dV. \tag{22}$$

Here \tilde{I}_0 is the moment of inertia with respect to \mathbf{x}_0 , represented in the body–fixed coordinates, and I_3 is the 3×3 identity matrix.

The equations for linear and angular acceleration are obtained by inserting Eqns. (19) and Eqns. (22) in Eqns. (18) giving

$$\begin{bmatrix} m_b I_3 & -m_b Q \tilde{\mathbf{x}}_{cm} \times \\ m_b Q \tilde{\mathbf{x}}_{cm} \times & I_0 \end{bmatrix} \begin{bmatrix} \dot{\mathbf{v}}_0 \\ \dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \mathbf{F} + \boldsymbol{\omega} \times m_b Q \tilde{\mathbf{x}}_{cm} \times \boldsymbol{\omega} \\ \boldsymbol{\tau}_0 - \boldsymbol{\omega} \times I_0 \boldsymbol{\omega} \end{bmatrix}. \tag{23}$$

Note the zero subscript on the torque, τ_0 , indicating that it is the torque on the body with respect to the point x_0 . Together with Eqns. (14) and (17), Eqn. (23) comprise the rigid body equations of motion. For convenience, we will use the more compact notation

$$M = \begin{bmatrix} m_b I_3 & -m_b Q \tilde{\mathbf{x}}_{cm} \times \\ m_b Q \tilde{\mathbf{x}}_{cm} \times & I_0 \end{bmatrix}, \quad v_b = \begin{bmatrix} \mathbf{v}_0 \\ \boldsymbol{\omega} \end{bmatrix}, \quad f = \begin{bmatrix} \mathbf{F} + \boldsymbol{\omega} \times m_b Q \tilde{\mathbf{x}}_{cm} \times \boldsymbol{\omega} \\ \boldsymbol{\tau}_0 - \boldsymbol{\omega} \times I_0 \boldsymbol{\omega} \end{bmatrix}.$$
(24)

so that Eqn. (23) can be written simply as

$$M\dot{v}_b = f. ag{25}$$

3.3 Added mass

The hydrodynamic force and torque on the body are given by

$$\mathbf{F}_{h} = \int_{\mathscr{L}} (-p\mathbf{I}_{3} + \mu \nabla \mathbf{u}) \cdot d\mathbf{S}, \quad \boldsymbol{\tau}_{h,0} = \int_{\mathscr{L}} (\mathbf{x} - \mathbf{x}_{0}) \times (-p\mathbf{I}_{3} + \mu \nabla \mathbf{u}) \cdot d\mathbf{S}, \tag{26}$$

where $\mathscr{S} = \partial \mathscr{B}$ is the body surface and $d\mathbf{S}$ is the differential area vector pointing out of the body region. Because the fluid is incompressible and the body is rigid, the absolute value of the pressure is immaterial. Only variations in pressure along the surface are relevant for the dynamics.

The pressure parts of the hydrodynamic force and torque contain components, that are proportional to the instantaneous body acceleration, and which go into the total force-torque vector, f, in Eqn. (25). Just as we did in the introductory 1-DoF example, we can conceptually split f into a part, $-A\dot{v}_b$, containing all terms that are proportional to \dot{v}_b , and another part, f_{other} , containing all other forces and torques,

$$f = f_{\text{other}} - A\dot{v}_b, \tag{27}$$

where A is the 6-by-6 added mass matrix, and $f_{\text{other}} \equiv f + A\dot{v}_b$.

For the added mass matrix, we recall its definition from potential flow theory [31]. In the case of a body moving in an unbounded, incompressible and inviscid fluid, the velocity field can be expressed in terms of a velocity potential, $\mathbf{u} = \nabla \phi$. The velocity potential, ϕ , can be decomposed into contributions proportional to the six velocity components (linear and angular) of the body:

$$\phi = v_1 \phi_1 + v_2 \phi_2 + v_3 \phi_3 + \omega_1 \phi_4 + \omega_2 \phi_5 + \omega_3 \phi_6. \tag{28}$$

The functions, $\phi_1, ..., \phi_6$ are called the unit potentials because they correspond to unit motion along each of the six degrees of freedom. ϕ_1 is found by solving a Laplace equation, $\nabla^2 \phi_1 = 0$, requiring that $\nabla \phi_1$ approaches zero at infinity and that $\hat{\mathbf{n}}_b \cdot \nabla \phi_1 = \hat{\mathbf{n}}_b \cdot (1 \ 0 \ 0)^T$ m/s on the body boundary. The other unit

potentials are found in a similar manner, setting the corresponding body velocity component to one and all other to zero. The added mass matrix can then be expressed in terms of the unit potentials as

$$A_{ij} = -\int_{\mathscr{S}} \rho_f \phi_i(\nabla \phi_j) \cdot d\mathbf{S}, \quad i, j = 1, ..., 6.$$
 (29)

In words, the 1st column of the added mass matrix is the linear and angular momentum (or impulse) of the fluid associated with the body moving with unit velocity along the 1st axis in the coordinate system in which the matrix is represented. Likewise, the 2nd and 3rd column contain, respectively, the fluid linear and angular momentum associated with unit body motion along the 2nd and 3rd coordinate axis. The 4th, 5th and 6th columns are populated with the fluid linear and angular momenta corresponding to unit angular velocity around the three coordinate axes, respectively.

For a body moving in an unbounded fluid the added mass matrix relative to body-fixed coordinates is constant and entirely determined by the body shape. There are many practical situations where the boundaries are so far away that the domain can be regarded as unbounded.

Even in the presence of viscosity and vorticity, the velocity field can be Helmholtz decomposed into purely potential part and a purely vortical part [32], and the added mass force on the body can be shown to be unaltered from the potential flow version [33, 34]. In other words, the forces and torques on the body from vortices in the fluid – including vorticity in the boundary layer – do not depend on the instantaneous acceleration of the body and hence do not contribute to its added mass coefficients. The independence of the added mass on wake vorticity and on the magnitude of the body acceleration has been numerically verified in [35].

If the domain is bounded, or there are other objects in the fluid, the unit velocity potentials associated with unit linear and angular motion of the body will still be well-defined, but will now depend on the instantaneous geometry of the domain. Hence, as the body moves and reorients relative to the other fluid domain boundaries, the unit potentials and added mass coefficients will also change. Especially if the body pierces a water surface, the instantaneous shape of this surface and the body position and orientation relative to it will influence the added mass matrix. As extreme example is an object falling from air into water, which will give rise to an increase in added mass by a factor of $\rho_{\text{water}}/\rho_{\text{air}} \approx 830$ as the object penetrates the water surface.

4 The FloatStepper algorithm

Performing CFD simulations with fixed boundaries, or boundaries moving in a prescribed way, is a standard task performed every day by thousands of engineers and scientists around the world. It is peculiar that adding just six degrees of freedom to the often millions of degrees of freedom used to represent the fluid state can cause severe numerical difficulties. Of all the infinitely many body paths we could prescribe, exactly one corresponds to the path the body would follow if it was free to move in response to the net forces and torques exerted on it, including hydrodynamic forces and other external forces. It is the job of our coupling algorithm to predict the body acceleration that leads us down this particular path when we use it in our prescription of the body motion. What makes this job so hard is the added mass force and its proportionality to the instantaneous body acceleration with a proportionality constant that we do not know in advance. The most widely used methods for solving the implicit acceleration equation is to employ expensive iterations between fluid and body state solvers. Here, we attempt instead to calculate the acceleration directly and non-iteratively. Inserting the decomposed force from Eqn. (27) into the body equations of motion, Eqn. (25), and isolating the acceleration, we get

$$\dot{v}_b = (M+A)^{-1} f_{\text{other}}.\tag{30}$$

This equation is well established in linear radiation-diffraction theory for floating bodies [1], where the added mass can be computed at the bodies equilibrium position. For it to be useful in unsteady CFD, and without assumption of small waves or body motion, we need to devise methods to calculate the unknown and time dependent vector, f_{other} , and matrix A.

4.1 The zero–acceleration step

Let us think of a snapshot of our body-fluid system with body state (x_b, v_b) , fluid state, (ρ, \mathbf{u}, p) , and possibly a number of external forces acting on the body. At this point in time, we need to decide where the

body should go in the next time step in order for its motion to represent free motion. Suppose we took a time step with the same v_b as in the previous time step. This would be experienced by the fluid as a step with zero acceleration, $\dot{v}_b = 0$. According to Eqn. (27), the force experienced by the body during such a zero–acceleration time step would be

$$f = f_{\text{other}}. (31)$$

In other words, taking a zero–acceleration time step with our CFD solver, and measuring the resulting hydrodynamic response force and torque, reveals the non-added mass part of Eqn. (26), which, together with gravity, mooring lines, self propulsion etc., comprises f_{other} .

4.2 Rewinding system

In our process of developing FloatStepper, we initially attempted to take the zero–acceleration time step without actually moving the mesh, as we otherwise do in real CFD time steps. This was, however, found to lead to wrong estimates of $f_{\rm other}$. Instead we take the zero–acceleration time step using mesh motion and exactly the same CFD solver settings as in the real time step. This ensures accurate estimation of $f_{\rm other}$. It also requires a careful time reversal step where, once $f_{\rm other}$ is obtained, the fluid, body and mesh are brought back exactly to their state before the zero–acceleration time step.

4.3 Added mass estimation

To numerically measure the instantaneous added mass matrix, we exploit its definition as the constant of proportionality between hydrodynamic force and body acceleration. We also exploit that the added mass in a viscous fluid with vorticity is identical to the one obtained in the corresponding potential flow situation. In the added mass calculation the convective and viscous terms can be neglected (see e.g. [36]), so the equation to solve is simply

$$\frac{\partial \rho \mathbf{u}}{\partial t} = -\nabla p \tag{32}$$

We now discretise the time derivative using the Euler scheme,

$$\frac{\partial \rho \mathbf{u}}{\partial t} \approx \frac{\rho^{n+1} \mathbf{u}^{n+1} - \rho^n \mathbf{u}^n}{\Delta t},\tag{33}$$

where the superscript denotes time step. In potential flow theory, the added mass associated with motion along the x-axis is obtained by impulsively changing the body velocity from zero to $\mathbf{v}_0 = (1,0,0)$ m/s amounting to a boundary condition, $\mathbf{n}_b \cdot \mathbf{u} = \mathbf{n}_b \cdot (1,0,0)$ m/s for \mathbf{u}^{n+1} . Inserting Eqn. (33) in Eqn. (32) and taking the divergence, we get

$$\nabla \cdot \left(\frac{1}{\rho^{n+1}} \nabla(\Delta t p_1)\right) = 0, \tag{34}$$

where we have used that $\mathbf{u}^n = \mathbf{0}$, required $\nabla \cdot \mathbf{u}^{n+1} = 0$, and marked the pressure with a subscript 1 to indicate that it is the pressure corresponding to acceleration $a_1 = 1 \text{m/s}/\Delta t$ along the first degree of freedom (here chosen to be the *x*-axis). We have also collected $\Delta t p_1$ in a bracket in Eqn. (34) to emphasise that this impulse approaches a constant as $\Delta t \to 0$. Physically, this means that if we impose the body velocity over a very short (long) time step, Δt , then the pressure amplitude required to set the fluid into corresponding motion must be very high (low) such as to keep $\Delta t p$ constant. In the limit $\Delta t \to 0$, the density ρ^{n+1} approaches ρ^n , and so in our added mass calculation step, we do not update the fluid interface position encoded in ρ , i.e. we simply use ρ^n instead of ρ^{n+1} in Eqn. (34).

Once the pressure, p_1 , corresponding to the acceleration, $a_1 = 1$ m/s $/\Delta t$ of the body along the x-axis is found, the corresponding force and torque on the body are calculated as

$$f_1 = \int_{\mathscr{S}} p_1 \begin{bmatrix} I_3 \\ (\mathbf{x} - \mathbf{x}_0) \times \end{bmatrix} d\mathbf{S}.$$
 (35)

and the first column of the added mass matrix is then given by

$$A_1 = -f_1/a_1. (36)$$

The second to sixth column of the added mass matrix are calculated in the same way, calculating the pressure force corresponding to unit body velocity along the other two axes, and unit angular velocity around the three coordinate axes.

In our current implementation of the added mass matrix calculator, the boundary conditions on all other boundaries than the rigid body are copied from the fields used in the real time step. Solver settings for the pressure equation are also copied from the real time step pressure solution. The added mass calculator is implemented as a copy of the PISO step in the interFoam solver except that the convective and viscous terms have been removed from the momentum equation. Thus, the calculation is fully parallelised, using the same domain decomposition as the real time step. The method allows the user to specify which degrees of freedom should be active in a simulation, for instance first, second and sixth for a body freely moving and rotating body in the xy-plane. It also allows the user to specify a parameter, MaddUpdateFreq, to only update the added mass matrix every MaddUpdateFreq'th time step. This may save computation time in simulations where the added mass is known to only change slowly. Finally, we mention that the added mass calculator class is derived from an abstract base class, allowing future addition of alternative added mass calculator classes (for instance a panel method based calculator) with runtime selection of the preferred method specified in the case setup files.

4.4 Body state update

Once f_{other} and A have been calculated, as described above, the body acceleration is calculated directly from Eqn. (30). This brings us to the actual integration of the body acceleration and velocity to obtain the new velocity and position. This can be done with standard ODE solvers, and while the choice of integration scheme here can have important consequences for energy conservation etc., it is not a point of focus for our work here. We simply use Euler integration, $x^{n+1} = x^n + \dot{x}^n \Delta t$, except for the orientation matrix, Q, which we update using the Rodrigues rotation formula,

$$Q^{n+1} = \left(I_3 + \sin(|\boldsymbol{\omega}|\Delta t) \left(\frac{\boldsymbol{\omega}}{|\boldsymbol{\omega}|} \times\right) + \left[1 - \cos(|\boldsymbol{\omega}|\Delta t)\right] \left(\frac{\boldsymbol{\omega}}{|\boldsymbol{\omega}|} \times\right)^2\right) Q^n, \tag{37}$$

to ensure that it stays orthogonal.

4.5 Mesh motion

From the fluid side, the body is represented by a boundary patch on which discretised versions of the boundary conditions in Section 3.1.1 are applied. Thus, after the body position and velocity have been updated to their newly found values, the mesh must follow along. In our current implementation, we use the deforming (or "morphing") mesh functionality of OpenFOAM with the body boundary patch moving rigidly, and the mesh points in a region around the patch deforming to accommodate this motion. In this approach all mesh points closer to the body patch than a user defined innerDistance follow the body in its rigid body motion. Mesh points outside a user specified outerDistance from the body patch are kept stationary. Mesh points between these two distances adapt their position smoothly using Spherical Linear Interpolation (SLERP) based on their distance to the body [37]. This leads to acceptable mesh quality as long as the body displacement and rotation are not too large.

4.6 Updating boundary conditions on a rigid body

In the meshed fluid domain the boundary patch representing the rigid body consists of polygonal faces, each with a face centre, \mathbf{x}_f , which is updated in each time step. We have implemented a velocity boundary field class called floaterVelocity, which holds a reference to a floating body object from which it reads a body position, \mathbf{x}_0 , velocity, \mathbf{v}_0 , and angular velocity, $\boldsymbol{\omega}$. For each face on the rigid body boundary patch it then sets the velocity field using Eqn. (20) with $\mathbf{x} = \mathbf{x}_f$. The boundary condition takes a boolean parameter called slip. If this is set to true, the boundary condition only takes the normal component from Eqn. (20). The velocity component tangential to the face is directly copied from the tangential component of the velocity vector at the centre of the cell to which the face belongs. This is to lowest order (ignoring the curvature of the surface) a symmetry condition on the tangential velocity component.

In the rigid body pressure boundary condition, Eqn. (12), the dependency on body acceleration appears indirectly in the first term on the right hand side. This can be seen by writing it as

$$\frac{\partial \rho \mathbf{u}}{\partial t} = \rho \dot{\mathbf{v}} + \mathbf{v} \frac{\partial \rho}{\partial t} = \rho \left[\dot{\mathbf{v}}_0 + \dot{\boldsymbol{\omega}} (\mathbf{x} - \mathbf{x}_0) + \boldsymbol{\omega} \times \left\{ \boldsymbol{\omega} \times (\mathbf{x} - \mathbf{x}_0) \right\} \right] + \left[\mathbf{v}_0 + \boldsymbol{\omega} \times (\mathbf{x} - \mathbf{x}_0) \right] \frac{\partial \rho}{\partial t}, \quad (38)$$

where we have inserted and differentiated the rigid body velocity, v, from Eqn. (20). This acceleration dependency is the very origin of the added mass force on the body and therefore is important to capture. We remark that in OpenFOAM, this dependency is treated indirectly by the PISO solution procedure. Discretising the momentum equation, Eqn. (8c), it can be written

$$a^{\mathbf{u}}\mathbf{u}^{n+1} = \mathbf{H} - \nabla p,\tag{39}$$

where $a^{\mathbf{u}}\mathbf{u}^{n+1}$ is a collection of all terms proportional to the new time velocity, \mathbf{u}^{n+1} , and \mathbf{H} (a vector not to be confused with the indicator field introduced in Section 3.1) contains all other terms except the pressure gradient, ∇p . If the time derivative in Eqn. (8c) is for instance discretised using the Euler method (Eqn. (33)), then $a^{\mathbf{u}}$ will contain a term, $\rho^{n+1}/\Delta t$, and \mathbf{H} will contain a term, $\rho^n\mathbf{u}^n/\Delta t$. For boundary faces on the rigid body, these old and new velocities are determined by the specified body acceleration, which is then conveyed to the pressure by imposing a boundary condition for p on the body surface of the form,

$$\mathbf{n}_b \cdot \nabla p = \mathbf{n}_b \cdot \left(\mathbf{H} / a^{\mathbf{u}} - \mathbf{u}^{n+1} \right) a^{\mathbf{u}}. \tag{40}$$

Here, for the acceleration to be included correctly, care must be taken when building the $\mathbf{H}/a^{\mathbf{u}}$ -field, and imposing boundary conditions on it. In particular, when assembling the pressure Poisson equation, the standard OpenFOAM solver calls a function, constrainHbyA, which sets $\mathbf{H}/a^{\mathbf{u}}$ equal to \mathbf{u}^{n+1} on boundaries where the \mathbf{u} boundary condition is of Dirichlet type, including rigid bodies. This results in $\mathbf{n}_b \cdot \nabla p = 0$, which we have just argued is incorrect when the body is accelerating. In our simulations, we have found this to give rise to an erroneous behaviour, where the velocity in the cell layer closest to the accelerating body has only around half of the correct magnitude regardless of the thickness of this layer. To avoid this, we make use of the build—in pressure boundary condition called fixedfluxExtrapolatedPressure. Using this, $\mathbf{H}/a^{\mathbf{u}}$ retains its boundary value obtained as a sum of all the terms from which it is composed, including the previous time velocity, \mathbf{u}^n , and we no longer observe the erroneous velocity behaviour.

4.7 Fluid state update

The exact procedure for solving the fluid equations is not the focus of the work presented here, and will therefore only be described briefly. The implementation is based on OpenFOAM's interfacial flow solver, interIsoFoam, (version v2206) employing the finite volume method to solve the motion of two immiscible fluids on arbitrary unstructured meshes with cell-centred collocated field representation.

The fluid solver for updating the fluid state, (ρ, \mathbf{u}, p) , starts by updating the fluid interface position using the VoF method, where the interface is represented by a volume fraction field expressing for each cell how much of its volume is occupied by the reference fluid. There exist many VoF methods, and the FloatStepper algorithm does not depend on this choice. Here we use the geometric VoF method called isoAdvector, which ensures a sharp interface and accurate, efficient interface advection on arbitrary unstructured meshes [9, 38]. The solver can be run in single-phase mode simply by setting the volume fraction field to 1 in all cells and on all boundaries. In this case, isoAdvector will find no interface cells and hence will do nothing.

After the interface advection step the pressure and velocity fields are updated using the PISO algorithm [39]. Details of the OpenFOAM specific solution procedure can be found in [8, 40, 41]. When the mesh is moving, the convection of mass and momentum in Eqn. (8) is made relative to the mesh motion. This is done by subtracting the flux due to motion of mesh faces from the physical fluxes across faces in the discretised convective terms, as described e.g. in [42].

4.8 Summary of algorithm

In Algorithm 3 we summarize the FloatStepper coupling algorithm. We remark that a similar approach to separating the force into an added mass contribution and everything else was briefly and elegantly sketched by Söding in [17]. Our method differs from Söding's by being non-iterative. The added mass estimate in

Söding's algorithm is found via a minimisation process and used as a good initial guess for an iterative solution procedure for the implicit acceleration equation. We, on the other hand, attempt to calculate $f_{\rm other}$ and A directly and sufficiently accurately, so that iterations can be avoided. Söding provides no validation and only few implementation details. In [22] Devolder et al. presented a 1 DoF OpenFOAM implementation of a similar iterative approach, and showed its favourable stability properties for a heaving floater.

Algorithm 3 The FloatStepper algorithm.

- 1: Initialise the body state, (x_b, v_b) , and fluid state, (ρ, \mathbf{u}, p) .
- 2: Increment time by Δt
- 3: Take a probe time step with zero acceleration and measure the resulting force and torque, which by Eqn. (31) is equal to f_{other} .
- 4: Rewind body, mesh and fluid states to state just before Step 3
- 5: Calculate updated added mass matrix, A (optionally, only every MaddFreqUpdate'th time step).
- 6: Calculate body acceleration as $\dot{v}_b = (M+A)^{-1} f_{\text{other}}$.
- 7: Time integrate \dot{v}_b to get v_b^{new} , and v_b to get x_b^{new} .
- 8: Move body (and mesh) accordingly and update fluid boundary conditions on its surface.
- 9: Calculate new fluid state as if the found body displacement was prescribed.
- 10: If end time reached, stop, else go to Step 2.

5 Validation

5.1 Light disc in gravity

In the introduction, we illustrated the added mass instability with a light disc rising in a heavy, inviscid fluid. We recall that in such an ideal fluid the force on the body is obtained by integrating the pressure over the body surface with the pressure given by the unsteady Bernoulli equation,

$$p = -\rho_f \frac{\partial \phi}{\partial t} - \rho_f \frac{1}{2} |\mathbf{u}|^2 + C, \tag{41}$$

where C is an arbitrary constant and ϕ is the velocity potential. For pure translational motion along the y-axis ϕ can be written as $v_y \hat{\phi}_y$, where $\hat{\phi}_y$ is the unit velocity potential associated with unit body velocity along the y-axis. The unit potential can be obtained by employing the Milne-Thomson circle theorem [43]. Thus, it is the time derivative of ϕ in Eqn. (41), which gives the added mass force contribution proportional to the instantaneous acceleration, \dot{v}_y , and to ρ_f , when integrated over the body surface. Isolating the acceleration in the resulting force expression, one obtains the theoretical constant body acceleration [31],

$$a_y = \frac{\rho_f - \rho_b}{\rho_f + \rho_b} g_y. \tag{42}$$

We remark that the ρ_f in the denominator comes from added mass term and prevents the acceleration from diverging when $\rho_b/\rho_f \to 0$. This term is sometimes omitted in the literature, for instance in section 13.10 of *Computational Methods for Fluid Dynamics* by Ferziger et al. [44] although their example has $\rho_b/\rho_f = 0.01$. It is exactly this omission in numerical coupling methods that leads to large overestimation of the body acceleration and numerical instability caused by unphysical kinetic energy injection.

Figure 4a shows the relative acceleration error (relative to a_y from Eqn. (42)) as a function of time for a FloatStepper simulation (blue), where the light circle is released to rise buoyantly at time zero. The FloatStepper acceleration is very close to constant with a relative error of around 0.06 %. For comparison we also show in Figure 4a the results obtained with the sixDoFRigidBodyMotion library of OpenFOAM with 1, 3 and 5 outer correctors in Algorithm 2. For those simulations the initial acceleration is miscalculated due to the initial estimate $a = F/m_b$ built into the algorithm. Increasing the number of outer iterations makes the simulation converge faster, but we cannot completely avoid the faulty initial accelerations, and the computational cost increases in proportion to the number of outer correctors.

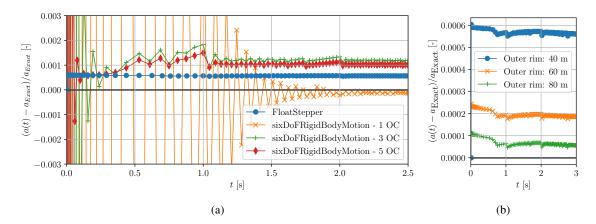


Figure 4: (a) Relative error in predicted acceleration for a circle of radius 1 m and density $\rho_b = 0.8 \text{ kg/m}^3$ surrounded by inviscid fluid of density $\rho_f = 1 \text{ kg/m}^3$. The <code>sixDofRigidBodyMotion</code> simulation is done with 1, 3 and 5 outer correctors (the former identical to the stable solution shown in Fig. 3a), acceleration relaxation $\gamma = 0.8$ and Euler time integration of the body equations of motion. FloatStepper and <code>sixDof-RigidBodyMotion</code> simulations use the same mesh and adaptive time step with maximum CFL of 0.5. (b) FloatStepper simulation repeated with outer domain at 40 m (same as in (a)), 60 m and 80 m.

The sixDoFRigidBodyMotion results in Figure 4a are run with acceleration relaxation $\gamma=0.8$ based on our knowledge of γ_c from Eqn. (7) and Figure 3b to ensure convergence. We remind the reader that this was only possible because of the simplicity of the case, a circle with known, constant added mass. In practical simulations, the added mass is normally not known and may vary with time. Indeed, one of the frustrating aspects of working with the sixDoFRigidBodyMotion is the guesswork going into setting the acceleration relaxation and the number of outer correctors for a given simulation situation. The user often faces a choice between excessive simulation time and reduced accuracy at best, or numerical instability at worst. Eliminating this guesswork is one of the main motivations for developing FloatStepper.

We have numerical investigated the cause of the constant 0.06% error of FloatStepper in Figure 4a. We have found that the error is unaltered by reducing the time step size or increasing the mesh resolution. The simulation was done with a circle of radius 1 m and with the circular outer rim of the domain placed 40 radii away. We have found that if we repeat the numerical experiment with radius of 60 m and 80 m instead of 40 m, the observed deviation from the theoretical value is reduced, see Figure 4b. This suggests that the deviation from the theoretical (infinite domain) value, is in fact not a numerical error but rather a finite domain size effect.

It is a peculiar thought that even at the very moment when the body is released, but has no velocity yet, it experiences a large added mass force from the surrounding fluid due to the momentum the fluid were to gain, if it was set into motion by the body. This is an artifact of the strict fluid incompressibility condition, causing the fluid and body to be as tightly dynamically coupled as the two halves of the rigid body. In reality no physical fluid is perfectly incompressible, so any real body acceleration will cause a pressure wave with corresponding fluid compression to emanate from the body at the speed of sound. Indeed, fluid solver approaches like Smoothed–Particle Hydrodynamics (SPH), where an artificial compressibility is introduced, do not appear to suffer from the added mass instability.

5.2 Disc in gravity hitting water surface

In the previous case we recalculated the added mass at every time step although it was essentially constant due to the long distance to boundaries and absence of a fluid interface. We now test our added mass calculation procedure with a test case involving a large sudden change in added mass, namely a circular body falling from air into water. Only the vertical component of the body motion is active. Figure 5 shows the initial configuration (black) as well as the body position and water surface at time t=1.3s (red), where the body is fully immersed in water, and at the end of the simulation, t=10s (green). In Figure 5b we

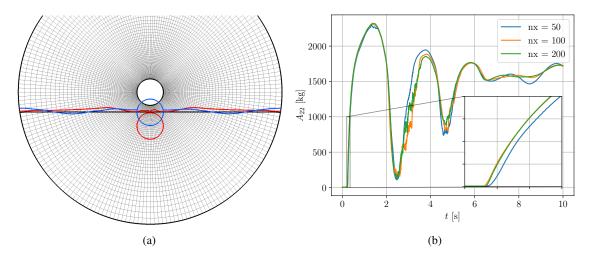


Figure 5: (a) A circular domain of radius 10 m centred at the origin and a water surface placed at y=-1.5m. A circular body of radius R=1 m is initialised at the origin with downward velocity $v_y=-1$ m/s. Gravity is $g_y=-9.81$ m/s², air density is $\rho_a=1$ kg/m³, water density is $\rho_w=1000$ kg/m³, and the body has density $\rho_b=500$ kg/m³. Body position and water surface shown for time t=0 s (black), t=1.3 s (red) and t=10 s (blue). (b) Evolution of the vertical added mass component with time for three different mesh resolutions.

show the time evolution of the added mass during the simulation for three different mesh resolutions. As expected, we observe a sudden rise in added mass as the body hits the water surface with a maximum when the body is fully immersed in water. The entry of the body into the water creates waves that are reflected at the domain walls, causing an irregular heaving motion of the body as the added mass settles to its equilibrium value dictated by the density ratios. The convergence with mesh resolution in the initial phase, where the body hits the water surface, is very good as shown in the inset of Figure 5. At later stages the correspondence between the three simulations is also good, although with small variations, presumably due to difference at different mesh resolutions in the ability to capture the details of the complicated, reflected wave field. This example demonstrates how the FloatStepper algorithm is able to robustly handle large and abrupt changes in added mass.

5.3 Free ellipse in infinite ideal fluid

When a rigid body free to translate and rotate is immersed in a fluid, the hydrodynamic forces introduce a coupling so that translation can induce rotation and vice verse. It is important to verify that our algorithm captures this coupling correctly. To this end, we consider a benchmark case with a rigid body moving through inviscid fluid with all boundaries far away. According to Howe [33] the hydrodynamic force associated with the body motion can be written

$$\mathbf{F}_h = -\frac{\partial}{\partial t} (T\mathbf{v}_0 + S\boldsymbol{\omega}) \tag{43a}$$

$$\boldsymbol{\tau}_{h,0} = -\frac{\partial}{\partial t} (S^T \mathbf{v}_0 + J\boldsymbol{\omega}) - \mathbf{v}_0 \times (A\mathbf{v}_0 + C\boldsymbol{\omega})$$
(43b)

where T, S and J are the 3×3 added mass submatrices,

$$A = \begin{bmatrix} T & S \\ S^T & J \end{bmatrix}. \tag{44}$$

To see how these added mass coefficients change in time, we note that the matrices T, S and J represented in the body-fixed basis, $\{\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3\}$, are constants in time determined solely by the body geometry. We will call these matrices \hat{T}, \hat{S} and \hat{J} and note that we have $T = Q\hat{T}Q^T$ and so on. Then for instance the first

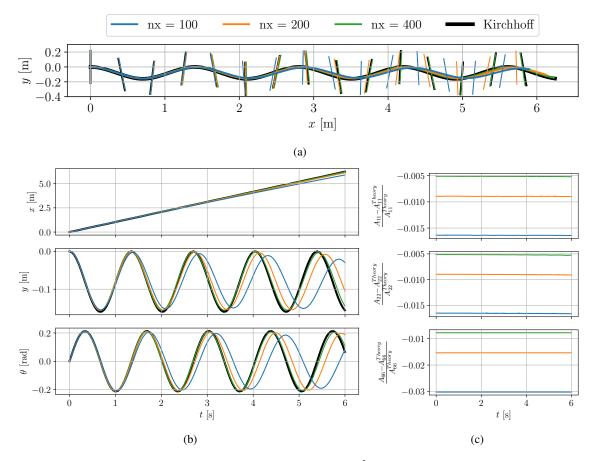


Figure 6: (a) Trajectory of an elliptic body of density $\rho_b = 0 \text{ kg/m}^3$ moving through an infinite 2D ideal fluid of density $\rho_f = 1 \text{ kg/m}^3$. Initial body centre at (0,0) with body minor-axis aligned with the *x*-axis. Initial velocity is $(v_x, v_y, \omega) = (1,0,1)$. Black curve is exact solution while simulations with three different mesh resolutions are shown in colours. Body orientation (major axis) shown at 15 locations along the trajectory. (b) Body coordinates as function of time. (c) Relative deviation of added mass coefficients from infinite domain values in Eqn. (49).

term in Eqn. (43) becomes

$$\partial_t(T\mathbf{v}_0) = \dot{Q}\hat{T}Q^T\mathbf{v}_0 + Q\hat{T}\dot{Q}^T\mathbf{v}_0 + Q\hat{T}Q^T\dot{\mathbf{v}}_0. \tag{45}$$

Noting that $\dot{Q} = \boldsymbol{\omega} \times Q$ and that $(\boldsymbol{\omega} \times)^T = -\boldsymbol{\omega} \times$, we can rewrite this to

$$\partial_t(T\mathbf{v}_0) = \boldsymbol{\omega} \times T\mathbf{v}_0 - T\boldsymbol{\omega} \times \mathbf{v}_0 + T\dot{\mathbf{v}}_0. \tag{46}$$

Using this for all the matrix-vector products in Eqn. (43), we get

$$\begin{bmatrix} \mathbf{F}_h \\ \mathbf{\tau}_{h,0} \end{bmatrix} = \begin{pmatrix} A \begin{bmatrix} \boldsymbol{\omega} \times & 0_3 \\ 0_3 & \boldsymbol{\omega} \times \end{bmatrix} - \begin{bmatrix} \boldsymbol{\omega} \times & 0_3 \\ 0_3 & \boldsymbol{\omega} \times \end{bmatrix} A \end{pmatrix} \begin{bmatrix} \mathbf{v}_0 \\ \boldsymbol{\omega} \end{bmatrix} - A \begin{bmatrix} \dot{\mathbf{v}}_0 \\ \dot{\boldsymbol{\omega}} \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ \mathbf{v}_0 \times (T \mathbf{v}_0 + J \boldsymbol{\omega}) \end{bmatrix}. \tag{47}$$

Inserting this as the force and torque in the body equations of motion, Eqn. (23), one obtains the Kirchhoff equations [45, 36] for a rigid body moving through an infinite, ideal fluid. They are valuable for evaluating fluid-structure coupling algorithms, like FloatStepper, because they are a set of ODE's that can be solved easily and fast on a computer and encompass non-trivial body-fluid interaction. In fact, for 3–dimensional motion they exhibit chaotic motion [46].

If we restrict ourselves to motion in the infinite xy-plane and choose coordinate axes such that S = 0 (always possible because of symmetry of added mass matrix), then Eqn. (47) simplifies to

$$\mathbf{F}_h = T(\boldsymbol{\omega} \times \mathbf{v}_0) - \boldsymbol{\omega} \times T\mathbf{v}_0 - T\dot{\mathbf{v}}_0 \tag{48a}$$

$$\mathbf{\tau}_{h\,0} = -J\dot{\mathbf{\omega}} - \mathbf{v}_0 \times T\mathbf{v}_0,\tag{48b}$$

where now $\mathbf{F}_h = (F_x, F_y, 0)^T$, $\mathbf{v}_0 = (v_x, v_y, 0)^T$, $\boldsymbol{\omega} = (0, 0, \omega)^T$ and $\boldsymbol{\tau}_{h,0} = (0, 0, \tau)^T$. For such planar motion, the equations are integrable, but still exhibit interesting dynamics and coupling between the three degrees of freedom. From Eqn. (48) we see that the hydrodynamic force depends both on body velocity and on body acceleration, when $\boldsymbol{\omega} \neq \mathbf{0}$. Also it shows (last term on right hand side of 48b) that the torque with respect to \mathbf{x}_0 depends on velocity, even if there is no body rotation.

Many useful coupling validation cases can be constructed based on the Kirchhoff equations and their solutions. Here we consider a body of elliptic shape with major and minor axes $R(1+a^2)$ and $R(1-a^2)$, respectively, where $a \in [0,1]$. For such a body the added mass coefficients are,

$$A_{11} = \rho_f \pi R^2 (1 - a^2)^2$$
, $A_{22} = \rho_f \pi R^2 (1 + a^2)^2$, $A_{66} = 2\rho_f \pi R^4 a^4$. (49)

In our numerical experiment we use R=1 m, a=0.5, fluid density $\rho_f=1$ kg/m³ and body density $\rho_b = 0$ kg/m³. The body is initialised with its centre at the origin and its minor axis aligned with the x-axis. The initial velocity is chosen to be $v_x = 1$ m/s, $v_y = 0$ m/s and $\omega = 1$ rad/s, which gives rise to an undulatory motion along the x-axis while the body wiggles with an angular amplitude of around 11°. Our simulation is started at t = 0 s and ends at time t = 6 s corresponding to around 4.5 motion periods. Adaptive time stepping was used with a maximum Courant-Friedrichs-Lewy (CFL) number of 0.1. The circular outer rim is placed at 40R. Simulations were done with three different mesh resolutions with 100, 200 and 400 cells in the radial direction with grading such that the inner most cells are 50 times smaller than the outer most cells. The corresponding azimuthal resolutions were 120, 240 and 480 cells. Simulations obtained with the three mesh resolutions are shown in Figure 6a together with the exact solution obtained by integrating the Kirchhoff equations directly. Figure 6b shows the horizontal, vertical and angular body components and their convergence to the exact solution (black curves) with mesh refinement. The added mass coefficients are recalculated in each time step. Figure 6c shows the relative error in the added mass coefficients as a function of time with respect to Eqn. (49). The added mass is very close to constant throughout each simulation, and the error is seen to diminish with increased mesh resolution. As in the rising circle case it is possible that some of this error is due to the finite domain size.

5.4 Freely floating box in regular waves

We now increase the level of complexity by considering a case combining a free surface with several active degrees of freedom. We choose the benchmark case presented in Ren et al. [47] with a box floating freely in regular waves in a wave flume. Here we try to reproduce their experimental data, including recorded wave elevation and box surge (x), heave (y) and pitch (θ) motion. The physical wave flume is 23 m long, 44 cm wide and filled to a water depth of d = 40 cm. The floating box is 30 cm long, 20 cm high and 40 cm wide, leaving a gap of 2 cm to each side wall of the flume. The box is made of 8 mm thick Perspex plates and has a compartment at the middle which is filled with a granular material to give it an overall density of 500 kg/m³, while retaining its centre of mass at its geometric centre. The total mass of the floater is then 12 kg. Assuming the density of Perspex to be 1180 kg/m³, and that the granular filling material is evenly distributed in the inner cross sectional area of the box, we calculate the moment of inertia with respect to its long centre axis to be $I_{\text{box}} = 0.151 \text{ kg m}^2$. The box is initialised in equilibrium, half immersed in water ($\rho_w = 1000 \text{ kg/m}^3$), with its centre 2 m from a piston-type wave generating wall placed at one end of the flume. To minimize wave reflections, a wave absorber is placed in the other end of the flume. Ren et al. [47] perform two free floater tests with regular waves of wave height H = 0.04 m and H = 0.10m, respectively, both with wave period T = 1.2 s. No detailed information is provided about the type of waves produced. In our numerical setup we generate waves using a custom made piston-type wave maker. It works by squeezing and stretching the cells in a region in front of the wave piston wall such as to make the piston wall move in accordance with a user supplied displacement file. The piston displacement signal we use is given from wave piston theory by

$$X(t) = -H \frac{\sinh(kd)\cosh(kd) + kd}{4\sinh^2(kd)}\sin(\omega t), \tag{50}$$

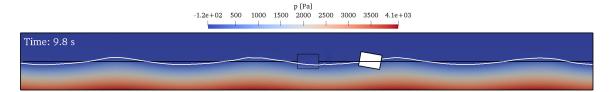


Figure 7: Snapshots of water surface and box position at time t = 0 s (black) and t = 9.8 s (white) for the H = 0.10 m case of Ren et al. [47].

where $\omega = 2\pi/T$, and k is found numerically by solving the dispersion relation,

$$\omega^2 = gk \tanh(kd),\tag{51}$$

for the given choice of period, depth and gravity, g. Given the two–dimensional nature of the problem with only a small gap between the box and flume walls, we choose to model the experiment in a two–dimensional setup. We choose a domain height of 0.8 m with the initial horizontal water surface in the middle at y=0 m. To limit the mesh size, we truncate the flume 8 m from the wave–piston wall and use the active wave absorption boundary condition build into OpenFOAM to minimise wave reflections [48, 49]. The cells in our coarse base mesh are squares with side length 1 cm, giving a mesh size of 63.400 cells. We also use a fine mesh with the region between y=0.2 m and y=0.55 m (covering the water surface and the box) refined into squares of side length 0.5 cm giving a total of 145.600 cells. The simulations are run on the 16 cores of an AMD EPYC 7301 processor with time steps adjusted to keep the maximum CFL number in the domain below 0.5. On the coarse mesh numerical experiments were also done with CFL < 0.1. Snapshots of the H=0.10 m case simulation are shown in Figure 7.

Ren et al. [47] provide experimental data for five wave periods on the interval $t \in [0, 6]$ s. This data is plotted with black dots in Figure 8, and shows almost periodic motion with only minor differences in amplitude from one period to another. No data or information is given about the preceding time interval where the waves and body motion was building up. Unless the box was kept fixed during this ramp up period, it will have drifted some distance from its initial position at x = 2 m. There is therefore some uncertainty about the offset for the surge motion shown in the second row of Figure 8. In our numerical experiments we have observed that this distance may be of importance because the waves reflected from the box interacts with the incoming waves in the region between wave piston and box. Thus, if we start our box at x = 2 m we generally overestimate the surge drift. Domínguez et al. [50] have also tried to numerically reproduce the results in [47], and have successfully reproduced the surge drift in their highest resolution simulation. They do not explicitly mention their starting position for the box, but from their figures, we infer that they started it at x = 4 m. We will therefore use this starting position in our simulations. Our FloatStepper results for the two mesh and time resolutions are shown in Figure 8.

For the free surface elevation in the top row, Ren et al. [47] do not mention where in the domain it is recorded. We have chosen to record the surface elevation just in front of the wave piston at x = 0.3 m. We compensate for the phase difference caused by the different numerical and experimental wave gauge positions by shifting the experimental wave data by 4.4T along the time axis for the H = 0.04m case, and by 3.4T for the H = 0.10 m case. The results demonstrate a good match in wave height and period between the experimental data and all three numerical runs.

The second row of panels in Figure 8 shows the box surge motion which is characterised by oscillations superimposed on a steady drift in the direction of wave propagation. For the H=0.04 case the coarse simulations with CFL = 0.1 and 0.5 capture both oscillations and drift very well. The fine simulation overestimates the drift for the last two periods. We do not currently have a good explanation for this behaviour. For the H=0.10 m case all three simulations give virtually identical surge, but with an overestimation of the drift motion in all five wave periods. As mentioned, this may be due to differences in horizontal initial box position. We note that the accumulated drift of around 0.7 m during the five wave periods causes significant mesh distortion with the currently available SLERP based mesh deformation in OpenFOAM. The two coarse meshed simulations crash at around t=10 s due to this, and the fine meshed simulation crashes at t=8 s. In future work we will incorporate an improved mesh deformation method that allows for larger lateral displacement without compromising mesh quality. We also plan to couple FloatStepper with

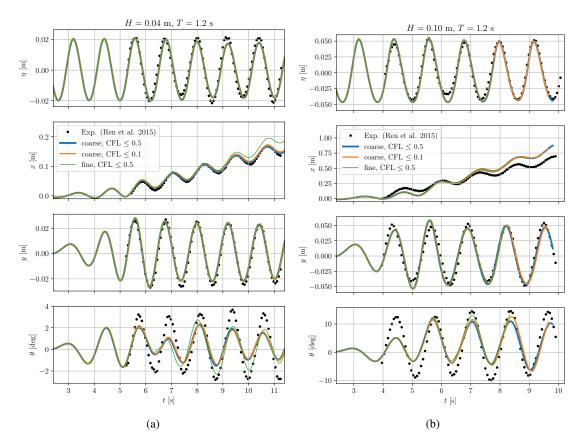


Figure 8: Freely floating box exposed to regular waves. Experimental results from Ren et al. [47] reproduced with FloatStepper using different mesh and time resolution.

the overset mesh implementation in OpenFOAM, which will allow for arbitrarily large body displacements and rotations without the problem of deteriorating mesh quality.

The heave motion of the box is shown in the third row of Figure 8. This is captured very well for both wave heights with virtually no difference between the simulations with different mesh and time resolution.

We show the pitch motion of the body in the last row of Figure 8. For both wave heights, the experimental pitch data is characterized by oscillations with minor irregularities in amplitude. All our simulations underestimate the amplitude of the pitch oscillations. For the H=0.04 case, the numerical oscillations are quite irregular and with minor, but noticeable difference between the coarse and fine simulations. For the H=0.10 m case the simulated pitch oscillations are more regular, but with a slightly longer period than the experiments. This may be due to the overestimated numerical surge drift, causing a Doppler–like shift in the period of the pitch forcing from the incoming waves.

Taking the uncertainties about wave signal, ramp-up procedure, wave gauge position and initial body position into account, we regard the numerical results as satisfactory.

5.5 Moored floating box in regular waves

Our last test case focuses on a moored floating box in regular waves. In order to enhance the usability of FloatStepper in Marine and Offshore Engineering, we have employed an integrated approach that combines the solver with state of the art mooring line algorithms. Many approaches to mooring dynamics have been explored throughout the years such as MDD [52], OrcaFlex [53], Moody [54], and MoorDyn [55]. It has been demonstrated that for floating structures subjected to significant motions, the behaviors described by a dynamic mooring model as opposed to a quasi-static mooring model produce a notable difference

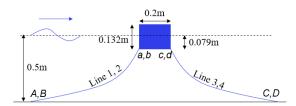


Figure 9: Numerical setup for the moored floating box experiment [51], including the fairlead and anchor points denoted as [a, b, c, d] and [A, B, C, D], respectively. The coordinates for these points are provided in the Table 1.

Parameter	Value
Box length	0.2 m
Box width	0.2 m
Box height	0.132 m
Box weight	3.148 kg
Centre of gravity	(0,0,-0.0126)
Box draft	0.0786 m
Mooring diameter	0.003656
Mooring weight	0.0607 kg/m
Mooring length	1.455 m
Axial Stiffness	29 N
Fairlead a,b,c,d	$\pm 0.1, \pm 0.1, -0.0736$
Anchor A,B,C,D	$\pm 1.385, \pm 0.423, -0.5$

Table 1: Box and mooring parameters along with coordinates of the mooring line anchor and fairlead connections from the experiment [51]

in mooring tensions [56]. In the present work, FloatStepper is coupled with the dynamic mooring line tool MoorDyn, which is an open-source library designed to seamlessly integrate with rigid body solvers. This comprehensive library encompasses catenary moorings, seabed friction, axial and bending stiffness, hydrodynamic drag and added mass effects, making it a versatile and complete mooring model. The coupling between the FloatStepper and MoorDyn follows a similar methodology as that adopted in [56]. The FloatStepper conveys the floating body's position and velocity to MoorDyn. MoorDyn, in turn, computes fairlead kinematics and updates the mooring system states, including the position, velocity of mooring line nodes, and segment tension. Subsequently, MoorDyn returns the mooring restraining forces and moments, which result from the sum of all fairlead tensions. These values are sent back from MoorDyn to the body motion solver, facilitating the update of the body's acceleration. MoorDyn's fundamental routines are built into a shared library to allow for the dynamic loading of functions within the FloatStepper.

The coupled model is validated against experimental measurements [51] involving a floating box moored using four catenary lines and exposed to regular wave conditions. The physical tests were conducted in a 30 m long and 1 m wide wave flume with a water depth of 0.5 m. Box dimensions and mooring configuration is shown in Figure 9 and listed in Table 1. The box is constructed from lightweight PVC material, with a net density of 570 kg/m³. A wooden plate was attached to the front of the box, positioned to face the incoming waves. Light reflecting markers were strategically positioned on this plate to track the 6-DoF motion. The mooring system comprises four chains of equal length arranged in a catenary configuration. To reduce the computational cost, we shorten our the numerical wave flume to 10 m. The domain is 1 m high and 1 m wide covered by a mesh of approximately 5 million cells. The mesh was decomposed into 100 sub-domains and simulations were executed on an AMD EPYC based HPC cluster.

Three different cases with differing wave conditions are found in the related references ([50], [57], [58]). We have run all three cases and found similar degree of correspondence between experiments and FloatStepper results in all of them. Here, we therefore only show the case with the most severe wave

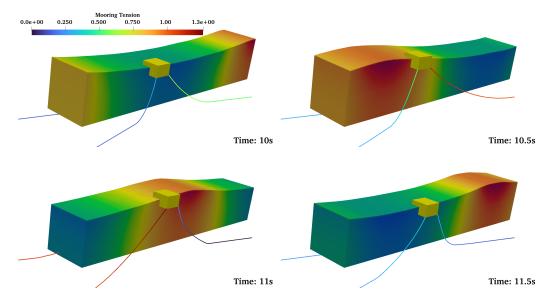


Figure 10: Instantaneous pressure distribution and a catenary mooring system configuration over one wave cycle for the box interacting with regular waves (T = 2 s, H = 0.12 m).

condition with a wave length of 4.116 m, a wave height of 0.12 m, and a wave period of 2 s (Case 3 from [50], Case 2 from [57]). The case was run for 20 simulation seconds corresponding to 10 wave periods with a maximum CFL number limit of 0.5. No turbulence model was activated. This approach simplified mesh generation and reduced computational effort, as it eliminated the need for strong grid refinement in the body boundary layers. Different turbulence models were tested in [57] for the same test case and it was found that the choice of turbulence model had minimal impact on the results. Figure 10 depicts snapshots during a single wave period, illustrating the instantaneous conditions of the fluid domain and mooring lines. In this representation, the colour of the fluid domain corresponds to the dynamic pressure variation, and the colour of the mooring lines indicates the real-time tension within each segment of the line.

Figure 11 presents a comparison of the floating box's motion, focusing on surge (x), heave (y), pitch (θ) and fairlead tension of Line 1 (T_{Line1}) . While our predictions for heave motion exhibit satisfactory agreement, there is a slight overestimation in surge motion and mooring tension T_{Line1} amplitudes. Such discrepancies were also found in previous numerical results [57] and [50] reported. In terms of pitch motion, our simulation matches maxima, minima, and phase, although there is a trend to overestimate the secondary peak amplitude between the main peaks, a phenomenon also observed in other numerical studies [57, 50, 58]. In these studies, the pitch discrepancies were attributed to uncertainties in experimental measurements, variations in geometry definitions, and the absence of the wooden plate for camera markers in the numerical simulations, which may alter the floater mass properties. In our current investigation, we analysed the effect of changing the center of gravity, inertia, and mooring parameters, to better understand the observed discrepancies. Notably, shifting the fairlead point just 1 cm closer to the center of gravity had a significant impact on the behaviour of the secondary peaks in pitch motion, as shown in Figure 11. We have also noticed a similar level of sensitivity when examining other wave case scenarios, mainly in the context of secondary peaks in the pitch motion. Given the reported uncertainties in experimental parameters, and the sensitivity of the rotational motion to these, we regard the moored floating body test as an acceptable validation result for FloatStepper.

6 Summary and discussion

We have demonstrated the feasibility of a new coupling algorithm, FloatStepper, for FVM based CFD simulation of an incompressible fluid and a rigid body. The method is based on direct calculation of the instantaneous added mass matrix, which allows separation of the added mass force from the other hydro-

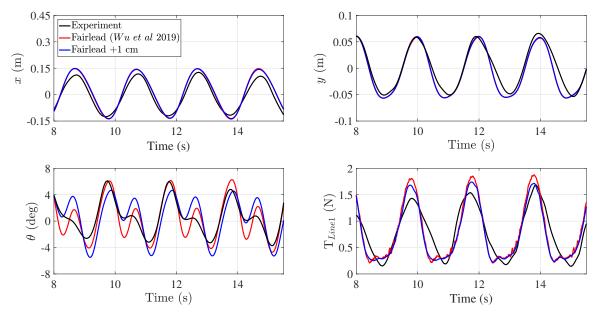


Figure 11: Moored floating box exposed to regular wave with T=2 s and H=0.12 m. CFD results with fairlead attachment as described in [51] (red) and with fairlead coordinate displaced by 1 cm (blue).

dynamic forces. Hereby the equations of motion can be solved robustly without iteration. While other researchers have proposed to introduce explicit added mass calculation, the combination of direct evaluation, non-iterative form and accessibility in a widely used open source CFD software framework is a novelty of our work.

The robustness of the algorithm has been demonstrated through five simple test cases. First, for a rising disc in unbounded fluid, the solver is able to determine the acceleration with a relative error of less than 0.01%. Next, for a disk falling into flat water, the solver is able to handle the abrupt change in added mass by a factor of $\rho_w/\rho_a=830$ at the initial entry and we demonstrated convergence with mesh refinement. The hydrodynamic coupling between translational and rotational degrees of freedom was tested against a benchmark with a wiggling ellipse travelling through unbounded fluid, where the Kirchhoff equations provide an exact solution. For this test, a zero body mass was used to demonstrate the absence of added mass instability, and the solver was shown to converge to the analytical solution upon mesh refinement.

The solver performance for floating structures in waves was next benchmarked in two test cases with a box floating freely and exposed to regular waves. Body motion was found to be reasonably well predicted with only small dependency on mesh and time resolution but with some overestimation of surge drift for the case with largest wave height and underestimation of pitch amplitude for both cases. Some of the observed discrepancy may be ascribed to the lack of exhaustive description of the experimental setup details such as wave and floater behaviour prior to the recorded experimental time interval.

Our last test case included a coupling with the MoorDyn library to compute the motion of a moored box in regular waves. The surge was well matched while the pitch motion deviated through a larger amplitude of a secondary motion peak in between the main peaks. The mooring line tension amplitude was overestimated by around 15%. For this experiment, some uncertainty for the mass properties of the setup has been described by other researchers, who have discussed similar deviations. Although a closer match in pitch for both box cases is desirable, we note the good match for the ellipse case and also good matches for comparison to test results for a wind turbine floater presented at the Wind Energy Science Conference 2023 [59]. On this basis, we regard the present test cases as an acceptable validation of the new algorithm.

The current implementation in OpenFOAM is published as open source [26] in the hope that it will be used an extended by the CFD community, scientists and engineers working with floating objects. The shared code is at a proof-of-concept level of maturity. This means that, while it can certainly be used for production CFD runs for floating object simulations, there is still plenty of room for improvement. In

particular, the code can most likely be optimized in terms of speed and memory usage. A central aspect here is the explicit added mass calculation. This comes at the price of solving six Poisson equations in the full domain, but eliminates the need for outer corrections, and – just as importantly – the uncertainty associated with choosing a safe, yet efficient, value for the acceleration relaxation parameter. In principle, FloatStepper with 6 active DoFs will cost the same as running with eight outer corrector (one zero–acceleration time step and six added mass column calculations in addition to the real time step). However, our experience so far is, that an added mass column calculation is not nearly as expensive as the full PISO time step of an outer corrector iteration. A quantification of this computational cost difference will be the subject of further studies, where we will also investigate the effect on accuracy and efficiency of reducing the added mass updating frequency.

Several numerical aspects, such as the ODE solver for the 6–DoF update, the interface advection method, and the type of mesh morphing, are currently hardcoded in the FloatStepper implementation. Our plan is to extend the code to allow the user various choices of schemes and methods and to easily add and test own customised methods. An important future extension would be to couple the method with overset mesh and immersed boundary methods to allow more extreme body motions than what is feasible with the deforming mesh method. Another relevant extension area would be the ability to handle multiple rigid bodies, which would enable simulations e.g. of the interaction of an installation vessel with a floating offshore wind turbine foundation.

A robust floating body algorithm is a prerequisite for realising the full potential of CFD as an engineering tool within fluid-structure interaction. It is our hope that the open source release of FloatStepper will help realise this potential, and foster collaboration in the CFD community to further improve the predictive capabilities of floating body CFD.

Acknowledgement

The work presented here was funded by the FloatStep Grand Solution project (8055-00075B) from Innovation Fund Denmark to Stromning Aps and Technical University of Denmark. JR also acknowledges partial funding from the DFF Sapere Aude Research Leader grant, InterFlow, to Roskilde University by Independent Research Fund Denmark (9063-00018B). JR thanks Henning Scheufler for useful discussions about code structure, and Željko Tuković for useful discussions about pressure boundary conditions.

References

- [1] Newman JN. 2018 Marine hydrodynamics. The MIT press.
- [2] Hirt C, Nichols B. 1981 Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics* **39**, 201–225.
- [3] Christensen E, Bredmose H, Hansen E. 2005 Extreme wave forces and wave run-up on offshore wind-turbine foundations. In *Proceedings of Copenhagen Offshore Wind Conference*. Copenhagen Offshore Wind 2005; Conference date: 26-10-2005 Through 28-10-2005.
- [4] Bredmose H, Hansen E, Skourup J, Christensen E, Pedersen L, Mitzlaff A. 2006 Numerical reproduction of extreme wave loads on a gravity wind turbine foundation. In *Proceedings of the ASME 25th International Conference on Offshore Mechanics and Arctic Engineering* vol. 1 pp. 279–287 United States. American Society of Mechanical Engineers. The 25th International Conference on Ocean, Offshore and Arctic Engineering, OMAE2006; Conference date: 04-06-2006 Through 09-06-2006.
- [5] Bredmose H, Jacobsen N. 2010 Breaking wave impacts on offshore wind turbine foundations: Focused wave groups and CFD. In *OMAE2010* p. 20368 United States. American Society of Mechanical Engineers. 29th International Conference on Ocean, Offshore and Arctic Engineering: Offshore Measurement and Data Interpretation, OMAE 2010; Conference date: 06-06-2010 Through 11-06-2010.
- [6] Ghadirian A, Bredmose H. 2020 Detailed force modelling of the secondary load cycle. *Journal of Fluid Mechanics* 889.
- [7] Ubbink O, Issa R. 1999 A Method for Capturing Sharp Fluid Interfaces on Arbitrary Meshes. *Journal of Computational Physics* **153**, 26–50.
- [8] Deshpande SS, Anumolu L, Trujillo MF. 2012 Evaluating the Performance of the Two-Phase Flow Solver interFoam. *Computational Science & Discovery* **5**, 014016.
- [9] Roenby J, Bredmose H, Jasak H. 2016 A computational method for sharp interface advection. *Royal Society Open Science* **3**, 160405.
- [10] Schmitt P, Elsaesser B. 2015 On the use of OpenFOAM to model oscillating wave surge converters. *Ocean Engineering* **108**, 98–104.
- [11] Sarlak Chivaee H, Pegalajar Jurado A, Bredmose H. 2018 CFD Simulations of a Newly Developed Floating Offshore Wind Turbine Platform using OpenFOAM. 21st Australasian Fluid Mechanics Conference.
- [12] Wang Jh, Zhao Ww, Wan Dc. 2019 Development of naoe-FOAM-SJTU solver based on OpenFOAM for marine hydrodynamics. *JOURNAL OF HYDRODYNAMICS* **31**, 1–20.
- [13] Begovic E, Gatin I, Jasak H, Rinauro B. 2020 CFD simulations for surf-riding occurrence assessment. *Ocean Engineering* **218**, 107975.
- [14] Wang L, Robertson A, Jonkman J, Kim J, Shen ZR, Koop A, Borràs Nadal A, Shi W, Zeng X, Ransley E, Brown S, Hann M, Chandramouli P, Viré A, Ramesh Reddy L, Li X, Xiao Q, Méndez López B, Campaña Alonso G, Oh S, Sarlak H, Netzband S, Jang H, Yu K. 2022 OC6 Phase Ia: CFD Simulations of the Free-Decay Motion of the DeepCwind Semisubmersible. *Energies* 15.
- [15] Windt C, Davidson J, Ringwood JV. 2018 High-fidelity numerical modelling of ocean wave energy systems: A review of computational fluid dynamics-based numerical wave tanks. *Renewable and Sustainable Energy Reviews* **93**, 610–630.
- [16] Ransley E, Yan S, Brown S, Hann M, Graham D, Windt C, Schmitt P, Davidson J, Ringwood J, Musiedlak PH, Wang J, Wang J, Ma Q, Xie Z, Zhang N, Zheng X, Giorgi G, Chen H, Lin Z, Qian L, Ma Z, Bai W, Chen Q, Zang J, Ding H, Cheng L, Zheng J, Gu H, Gong X, Liu Z, Zhuang Y, Wan D, Bingham H, Greaves D. 2020 A Blind Comparative Study of Focused Wave Interactions with Floating Structures (CCP-WSI Blind Test Series 3). INTERNATIONAL JOURNAL OF OFFSHORE AND POLAR ENGINEERING 30, 1–10.
- [17] Söding H. 2001 How to integrate free motions of solids in fluids. In 4th Numerical towing tank symposium, Hamburg vol. 52.
- [18] Bettle M. 2012 Unsteady Computational Fluid Dynamics Simulations of Six Degrees-of-Freedom Submarine Manoeuvres. PhD thesis University of New Brunswick.

- [19] Dunbar AJ, Craven BA, Paterson EG. 2015 Development and Validation of a Tightly Coupled CFD/6-DOF Solver for Simulating Floating Offshore Wind Turbine Platforms. *Ocean Engineering* 110, 98– 105
- [20] Chow J, Ng E. 2016 Strongly Coupled Partitioned Six Degree-of-Freedom Rigid Body Motion Solver with Aitken's Dynamic under-Relaxation. *International Journal of Naval Architecture and Ocean Engineering* 8.
- [21] Bruinsma N, Paulsen B, Jacobsen N. 2018 Validation and application of a fully nonlinear numerical wave tank for simulating floating offshore wind turbines. *Ocean Engineering* **147**, 647–658.
- [22] Devolder B, Troch P, Rauwoens P. 2019 Accelerated Numerical Simulations of a Heaving Floating Body by Coupling a Motion Solver with a Two-Phase Fluid Solver. *Computers & Mathematics with Applications* 77, 1605–1625.
- [23] Veldman AE, Luppes R, Van Der Plas P, Van Der Heiden H, Düz B, Seubers H, Helder J, Bunnik T, Huijsmans R. 2016 Free-surface flow simulations for moored and floating offshore platforms. vol. 4 p. 7515 7531.
- [24] Shigunov V, Söding H, Zhou Y. 2001 Numerical Simulation of Emergency Landing of Aircraft on a Plane Water Surface. .
- [25] Meyer J, Graf K, Thomas S. 2017 A New Adjustment-Free Damping Method for Free-Surface Waves in Numerical Simulations.
- [26] Roenby J. 2023 FloatStepper. https://doi.org/10.5281/zenodo.8146516.
- [27] Causin P, Gerbeau J, Nobile F. 2005 Added-Mass Effect in the Design of Partitioned Algorithms for Fluid-Structure Problems. Computer Methods in Applied Mechanics and Engineering 194, 4506– 4527.
- [28] Förster C, Wall WA, Ramm E. 2007 Artificial Added Mass Instabilities in Sequential Staggered Coupling of Nonlinear Structures and Incompressible Viscous Flows. Computer Methods in Applied Mechanics and Engineering 196, 1278–1293.
- [29] Devolder B, Schmitt P, Rauwoens P, Elsaesser B, Troch P. 2015 A review of the implicit motion solver algorithm in OpenFOAM® to simulate a heaving buoy. In *NUTTS conference* vol. 2015 p. 18th.
- [30] Huang L, Li Y, Benites-Munoz D, Windt CW, Feichtner A, Tavakoli S, Davidson J, Paredes R, Quintuna T, Ransley E, Colombo M, Li M, Cardiff P, Tabor G. 2022 A Review on the Modelling of Wave-Structure Interactions Based on OpenFOAM. *OpenFOAM® Journal* 2, 116–142.
- [31] Milne-Thomson LM. 2011 Theoretical Hydrodynamics. New York: Dover Publications 5th edition.
- [32] Eldredge JD. 2019 Mathematical Modeling of Unsteady Inviscid Flows. Interdisciplinary Applied Mathematics. Springer International Publishing.
- [33] Howe MS. 1995 ON THE FORCE AND MOMENT ON A BODY IN AN INCOMPRESSIBLE FLUID, WITH APPLICATION TO RIGID BODIES AND BUBBLES AT HIGH AND LOW REYNOLDS NUMBERS. The Quarterly Journal of Mechanics and Applied Mathematics 48, 401–426.
- [34] Conca C, Osses A, Planchard J. 1997 Added Mass and Damping in Fluid-Structure Interaction. *Computer Methods in Applied Mechanics and Engineering* **146**, 387–405.
- [35] Wakaba L, Balachandar S. 2007 On the Added Mass Force at Finite Reynolds and Acceleration Numbers. Theoretical and Computational Fluid Dynamics 21, 147–153.
- [36] Mougin G, Magnaudet J. 2002 The generalized Kirchhoff equations and their application to the interaction between a rigid body and an arbitrary time-dependent viscous flow. *International Journal of Multiphase Flow* 28, 1837–1851.
- [37] Shoemake K. 1985 Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques* SIGGRAPH '85 pp. 245–254 New York, NY, USA. Association for Computing Machinery.
- [38] Roenby J, Bredmose H, Jasak H. 2019 IsoAdvector: Geometric VOF on General Meshes. In Nóbrega JM, Jasak H, editors, *OpenFOAM*®, pp. 281–296. Cham: Springer International Publishing.

- [39] Issa R. 1986 Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of Computational Physics* **62**, 40–65.
- [40] Moukalled F, Mangani L, Darwish M. 2016 The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM® and Matlab vol. 113Fluid Mechanics and Its Applications. Cham: Springer International Publishing.
- [41] Uroić T. 2019 *Implicitly coupled finite volume algorithms*. PhD thesis University of Zagreb. Faculty of Mechanical Engineering and Naval Architecture.
- [42] Jasak H, Tuković Ž. 2010 Dynamic mesh handling in OpenFOAM applied to fluid-structure interaction simulations. In Proceedings of the V European Conference on Computational Fluid Dynamics ECCOMAS CFD 2010.
- [43] Acheson DJ. 1990 Elementary Fluid Dynamics. Oxford: New York: Clarendon Press 1st edition.
- [44] Ferziger JH, Perić M, Street RL. 2020 Computational Methods for Fluid Dynamics. Cham: Springer 4th edition.
- [45] Lamb SH. 1932 Hydrodynamics. Cambridge Univ. Press. 6th edition.
- [46] Aref H, Jones SW. 1993 Chaotic motion of a solid through ideal fluid. *Physics of Fluids A: Fluid Dynamics* 5, 3026.
- [47] Ren B, He M, Dong P, Wen H. 2015 Nonlinear simulations of wave-induced motions of a freely floating body using WCSPH method. *Applied Ocean Research* **50**, 1–12.
- [48] Higuera P, Lara JL, Losada IJ. 2013a Realistic wave generation and active wave absorption for Navier–Stokes models: Application to OpenFOAM®. *Coastal Engineering* **71**, 102–118.
- [49] Higuera P, Lara JL, Losada IJ. 2013b Simulating coastal engineering processes with OpenFOAM®. *Coastal Engineering* **71**, 119–134.
- [50] Domínguez JM, Crespo AJ, Hall M, Altomare C, Wu M, Stratigaki V, Troch P, Cappietti L, Gómez-Gesteira M. 2019 SPH simulation of floating structures with moorings. *Coastal Engineering* 153, 103560.
- [51] Wu M, Wang W, Palm J, Eskilsson C. 2018 CFD Simulation of a Passively Controlled Point Absorber Wave Energy Converter. In Ölçer AI, Kitada M, Dalaklis D, Ballini F, editors, *Trends and Challenges in Maritime Energy Management*, WMU Studies in Maritime Affairs pp. 499–511. Cham: Springer International Publishing.
- [52] Dewey RK. 1999 Mooring Design & Dynamics—a Matlab® package for designing and analyzing oceanographic moorings. *Marine Models* 1, 103–157.
- [53] Randolph M, Quiggin P. 2009 Non-linear hysteretic seabed model for catenary pipeline contact. In International Conference on Offshore Mechanics and Arctic Engineering vol. 43437 pp. 145–154.
- [54] Ferri F, Palm J. 2015 Implementation of a Dynamic Mooring Solver (MOODY) into a wave to wire model of a simple WEC. Department of Civil Engineering.
- [55] Hall M, Goupee A. 2015 Validation of a lumped-mass mooring line model with DeepCwind semisub-mersible model test data. *Ocean Engineering* **104**, 590–603.
- [56] Aliyar S, Ducrozet G, Bouscasse B, Bonnefoy F, Sriram V, Ferrant P. 2022 Numerical coupling strategy using HOS-OpenFOAM-MoorDyn for OC3 Hywind SPAR type platform. *Ocean Engineering* 263, 112206.
- [57] Chen H, Hall M. 2022 CFD simulation of floating body motion with mooring dynamics: Coupling MoorDyn with OpenFOAM. Applied Ocean Research 124, 103210.
- [58] Jeon W, Park S, Cho S. 2023 Moored motion prediction of a semi-submersible offshore platform in waves using an OpenFOAM and MoorDyn coupled solver. *International Journal of Naval Architec*ture and Ocean Engineering 15, 100544.
- [59] Aliyar S, Bredmose H. 2023 CFD analysis of the installation of floating wind SPAR substructure in waves, Wind Energy Science Conference (WESC). https://zenodo.org/record/8325970.