Proceedings of the ASME 2019 38th International &
Conference on Ocean, Offshore and Arctic Engineering
OMAE 2019
June 9-14, 2019, Glasgow, Scotland

OMAE2019-96051

DRAFT: VIRTUAL PROTOTYPING AND SIMULATION OF MULTIBODY MARINE OPERATIONS USING WEB-BASED TECHNOLOGIES

Ícaro A. Fonseca[*]**, Felipe F. de Oliveira, Henrique M. Gaspar**
Department of Ocean Operations and Civil Engineering
Norwegian University of Science and Technology
Ålesund, Norway

ABSTRACT

This paper focuses on virtual prototyping and simulation of marine operations based on web technologies. The ship is represented as a digital object, which can be used to perform different types of analyses and simulations. The presented simulations are: motion of a single hull and of multiple hulls in regular waves calculated with closed-form expressions, induced pendulum motion response to a lifted load, and motion of a barge with initial movements in still water calculated with equations of motion.

The simulations are developed as web applications in JavaScript and HTML, with graphical user interfaces and 3D renders of the operations. Relevant parameters of the simulations such as wave characteristics and design dimensions are linked to interactive dashboards, allowing the user to modify them and visualize the results in real-time. The applications are lightweight enough to be executed locally in the web browser of most modern devices.

The work employs an open source approach, relying most notably on the Vessel.js library. This aims to foster reuse of models and collaboration with external contributors.

NOMENCLATURE

t	Time.
j	Vessel motion mode (3 for heave, 4 for roll and 5 for pitch.
$\eta_j(t)$	Vessel motion.
ϕ_j	Amplitude of the vessel motion.
ω	Angular wave frequency.
k	Wave number.
$dist$	Orthogonal distance between the vessel's position and the origin line of the wave train.
θ_j	Phase angle of the motion mode.
J	Transformation matrix from body-fixed coordinate to world coordinate.
F	External forces (gravitational force is included in this element).
$C_{RB}(\dot{\eta})$	Rigid-body Coriolis and centripetal matrix.
$C_A(\dot{\eta})$	Hydrodynamic added Coriolis and centripetal Matrix.
B	Damping matrix.
$C(\eta)$	Restoring forces.
M_{RB}	Rigid-body matrix of inertia.
M_A	Added mass matrix of inertia.

[*]Contact author: icaro.a.fonseca@ntnu.no

INTRODUCTION

Simulations and virtual prototyping (VP) have been important in marine engineering design for years, and the overall usage of simulations and VP through the marine life cycle has been recently increasing. Virtual prototypes allow testing of engineering systems for different purposes in the life cycle, for instance: evaluation of proposals during conceptual design, virtual commissioning of the system and planning of operations. These features are quite desirable in the context of the marine operations, where the high risk, complexity and cost of the systems is prohibitive to the usage of physical prototypes in general.

Virtual prototyping also poses the advantage of allowing sharing of models among distributed agents for usage, verification and validation. Given the high number of stakeholders involved in the vessel's life cycle, it becomes important to share data among distributed agents as efficiently as possible, allowing them to easily access the data that is relevant to their activities. In this context, the web-based approach brings useful features to make vessel data accessible to a great number of users while reduce complications associated with complex engineering frameworks management.

For instance, web-applications are compatible with any modern device that has a web browser, avoiding compatibility issues from multiple sources. This ubiquity was attainable in great part due to the reliance of web technologies in open standards, allowing developers to freely use and implement them. In fact, two of the three core technologies of the web, HTML and CSS, are open standards, while the third one, the JavaScript programming language, is an implementation of an open standard, ECMAScript.

In practice, this implies that during development it is not necessary to target a specific operational system or device configuration. On the other hand, the user is not required to install any software or environment in order to execute the application, and they always have access to the latest version of the app without being required to install updates manually.

When applied to simulate marine operations, the web-based approach allows the creation of interactive visualizations with realistic 3D graphics including textures and lighting. The applications can be useful in different stages of the life cycle: during early design phase, they may give the user a better perception of the physical meaning of the results; during operation, they may be used for training of personnel or planning of activities.

WEB-BASED VIRTUAL PROTOTYPING AND SIMULATION OF MARINE OPERATIONS

Web-based development is supported by a wide variety of open source libraries for different purposes: they can be applied not only for calculations performance and solving differential equations, but also for creating elaborated graphical user interfaces, 2D and 3D visualizations, and so on. Gaspar [1] gives an overview of JavaScript development in the context of maritime design and engineering, listing various useful open source libraries.

WebGL is one of the most relevant JavaScript APIs for rendering graphics in a web browser. It supports GPU acceleration for physics and image processing. The Three.js library can be used to draw and load 3D shapes in a canvas using WebGL, making it easier to create animations with lights, textures and other graphic resources. All the simulators described in the following paragraphs use visualizations created with Three.js.

There are already some web-applications related to the scope of this work. In terms of virtual prototyping, the CAD platform CAESES released a generator of Wageningen B-series propeller geometry [2]. The user is allowed to configure all relevant propeller characteristics (e.g., diameter, expanded area, pitch, thickness) and the propeller geometry is automatically created in a remote server running the CAD environment. When satisfied, the user can download the final model as a file in STEP or STL format for posterior use. STL is suitable for 3D printing (in fact, the name is an abbreviation of "stereolithography"). A STL model is defined with triangular facets forming a 3D shape. STEP is a CAD format which can be used for CAE analyses. It is an open standard developed by ISO and therefore is suitable for model exchange, being supported by various proprietary engineering software.

Hatledal et al. [3] present an architecture for simulations based on web technologies and the Functional Mock-up Interface (FMI). FMI is an open standard for dynamic simulation models. It is widely used in the automotive industry, but can be applied to other domains as well. FMI allows development of modules that can be exchanged and assembled into complex simulations. It is adequate for distributed co-simulations, where multiple geographically disperse users interact with different aspects of the operation in the same simulation environment. The architecture presented in the work executes the simulation modules in the server and synchronizes the results with the client browser, where the visualization layer renders the graphics. The architecture was applied to virtual prototyping and operation of maritime cranes.

The research group with which this work is involved has been consistently developing web applications for marine design and engineering, including some simulations. Chaves [4] presented a 3D simulator for ship virtual prototype and motion prediction in regular waves. It allows configuration of design characteristics for visualization purposes (e.g., type of propulsion, type of bow, size of superstructure) and variation of vessel main dimensions, which directly influence the predicted motion response.

VESSEL.JS FOR SIMULATIONS

Vessel.js is a JavaScript library for investigation of common issues of conceptual ship design [5]. The library follows a web-based and object-oriented approach. It is open source and collaborative, welcoming reuse of code and input from external contributors. Vessel.js is currently developed by the Ship Design and Operation lab at NTNU in Ålesund.

Vessel.js supports the simulations presented in the following sections, from virtual prototyping of a vessel to simulation of vesse behavior in operation. The simulations are based on a taxonomy comprising three sub-models: entities, states and processes [6, 7]. The entity model collects data about the simulated system. It may represent an actual vessel or a design concept during the design stage. The state model defines static constraints to which the vessel is subjected. It is a static simulation or analysis, e.g., calculation of floating condition or resistance for a given speed. Finally, the process model is a succession of states, which may be arranged to create a dynamic simulation, e.g., a simulation of an operation. In the Vessel.js library, the entity model translates to a ship object, possibly complemented by other objects representing additional systems, the state model translates to modules that receive the ship object and other arguments to calculate the states, and the process model to web applications where the states are combined to simulates the ship behavior.

The next sections explains how the ship virtual prototype is defined with the Vessel.js, and the following one explains how the library calculates states based on the ship definition and on the simulation constraints. These principles are used to perform the time-domain simulations presented later in this work.

VIRTUAL PROTOTYPING WITH VESSEL.JS

A ship design is described with objects for compartments, structure and additional systems. The compartments are created with ”base” and ”derived” objects. The base objects defines weight data, dimensions and link to 3D files. It functions as a ”template” of a compartment, which can be replicated in different positions inside the vessel. This is done with the derived objects, which contain the coordinates where the element will be placed inside the vessel. The ship's structure comprises hull, decks and bulkheads. The decks and bulkheads are defined with geometric dimensions (i.e., spans, equivalent thickness) and material density; and the hull is defined with a table of offsets. The weights of the decks and bulkheads are derived directly from the physical dimensions of the elements, while the hull weight is estimated with empirical formulas in order to overcome the lack of structural detailing during conceptual design stage. Finally, additional subsystems (e.g., propulsion, lifting equipment) are modeled in the library with specific approaches depending on the intended purpose of the model and the requirements of the simulation.

Once a ship object is defined with Vessel.js, it can be visualized in WebGL. A function was specifically developed to create a 3D visualization in Three.js from the ship object. The function automatically generates the hull visualization from its table of offsets. The base objects are represented with STL files provided by the user. If no file is provided for a given base object, it will be represented in the visualization with a cuboid of equivalent dimensions. The function returns an object ready to be loaded to a scene in the web browser, where it can be visualized, as pictured in Fig. 1, and manipulated.

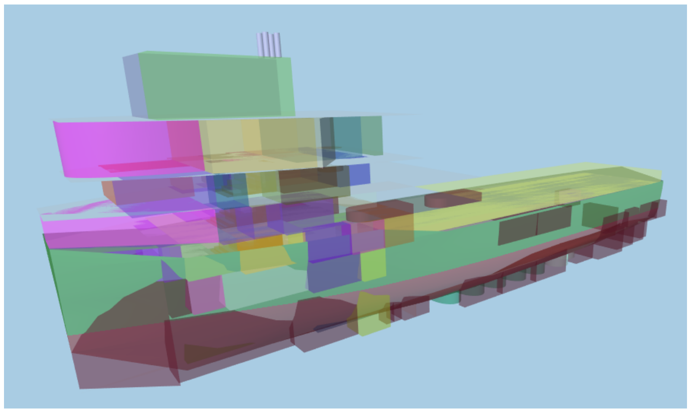


FIGURE 1. VISUALIZATION OF A PSV SPECIFICATION GENERATED WITH THE VESSEL.JS LIBRARY.

A ship object created with Vessel.js can be serialized as a specification and stored for posterior use in various applications developed with the library. Vessel.js uses the JavaScript Object Notation (JSON) as the standard for serialization. Besides being ubiquitous across programming languages and libraries, JSON is also human-readable. This is a crucial feature to allow semantic interpretation of data, facilitating inspection of the specifications.

CALCULATION OF STATES

Vessel.js provides methods for calculate a broad types of ship states, which can be used to perform a design analysis or to assemble a dynamic simulation. The handling of states follows a certain degree of modularization, being calculated independently from each other when possible.

The Vessel.js library includes an object prototype to handle all the states calculated during a simulation. The object is able to handle both discrete states which do not need to be constantly updated, e.g. the filling ratios of tanks, and continuous states which are constantly tracked during the simulation, more notably the vessel's position in the six degrees of freedom. The positional states can be directly linked to Three.js library in order to visualize the vessel's motion. The following paragraphs detail the simulation models used in the simulations developed for this work.

Loading Condition

The vessel's loading condition can be defined by assigning filling ratios and positions to its derived objects, which are intended to represent its tanks and compartments. It is possible to define the filling ratio of each tank individually or in groups (e.g., group of ballast tanks, group of fuel tanks). When the user requests the library to calculate the vessel's displacement and center of gravity, the library combines the vessel's lightweight with the current loading condition to provide the resulting values.

Floating Condition

The vessel's floating condition is defined by confronting the vessel's current displacement and center of gravity with the hull table of offsets to calculate its floating dimensions, hydrostatic coefficients and stability coefficients numerically. This includes calculation of draft, water plane dimensions and coefficients, form coefficients and position of metacenters, among others. Trim is also calculated for small angles (that is, inclining angles small enough for the metacenter position to remain approximately the same). The scheme in Fig. 2 illustrates how a ship can be associated to states describing different loading conditions, which lead to different floating conditions.

Wave Motion Response Amplitude

The amplitude of wave motion response is estimated with closed-form expressions by Jensen et al. [8]. The method estimates amplitude response for heave, roll and pitch in regular

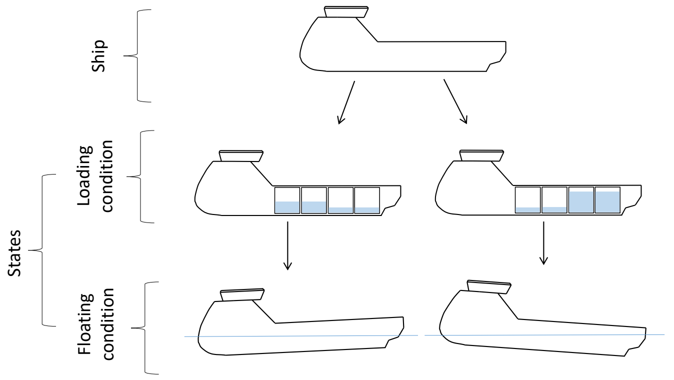


FIGURE 2. SHIP SUBJECTED TO DIFFERENT LOADING STATES, LEADING TO DIFFERENT FLOATING CONDITIONS.

waves based on the hull's main dimensions and its form parameters. The hull is modeled as a box-shaped barge, for heave and pitch, and as a combination of two box shapes, for roll. It neglects coupling between heave and pitch so that the total vertical motion amplitude is estimated by assuming a 90° phase difference between both movements. The results serve as an approximation of the motion which can be used during early design stage.

In the Vessel.js library, the regular wave characteristics are handled by an object with frequency, amplitude and direction in relation to the environment. The ship state should also includes the ship direction in relation to the environment. When the response amplitude is calculated, the wave and ship directions are compared in order to derive the ship heading in relation to waves. The scheme in Fig.3 below illustrates how the wave motion response is calculated based on a given floating condition excited by an incident wave.

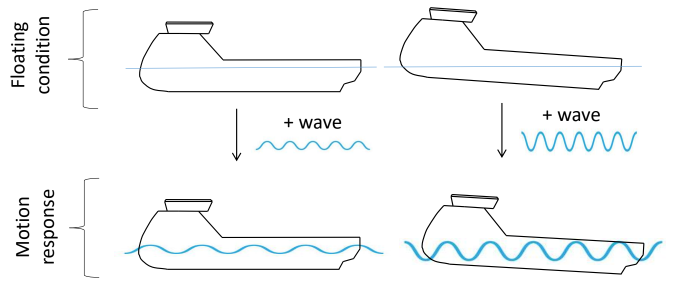


FIGURE 3. MOTION RESPONSE STATES CALCULATED FROM A FLOATING CONDITION EXCITED BY AN INCIDENT WAVE.

Time-Domain Response of Hull with Closed-Form Expressions

The response amplitudes calculated with the formulation in the previous section for heave, pitch and roll can be converted to a sinusoidal series with Eqn. (1) and then synchronized with an incident regular wave in a 3D visualization in order to represent hull motion over time. If the vessel is not positioned in the wave origin, the motion phase may need to be corrected for the orthogonal distance in relation to the wave train's origin in order to keep the hull and wave motions in synchronization.

$$\eta_j(t) = \phi_j cos(\omega \cdot t - k \cdot dist + \theta_j) \qquad j = 3,4,5. \tag{1}$$

Pendulum Response of Load During Lifting Operation

A module was created to simulate the pendulum motion of a load lifted by an A-frame in a ship moving in regular waves. The model considers the hanging load as a spherical pendulum with moving pivot. The equations of motion are derived with the Lagrangian formulation describing the motion with Euler angles [9,10]. This system of coordinates can be easily represented in a Three.js visualization.

The accelerations of the pivot, the load's hanging point, are derived from the motion response calculated with the closed-form expressions, as presented in the previous section. The motion is calculated on the hanging point of the load and substituted on the equations for each time-step of the simulation. The system of equations is solved with the Dormand-Prince method implementation in the Numeric.js library [11], yielding the angular position and velocity of the pendulum over time. The scheme in Fig. 4 illustrates the calculation approach, where the ship motion is combined with the lifting equipment to derive the pendulum response of the lifted load. The pendulum model is purely kinematic, not taking into consideration the forces induced by the load and the motion interaction between load and ship.

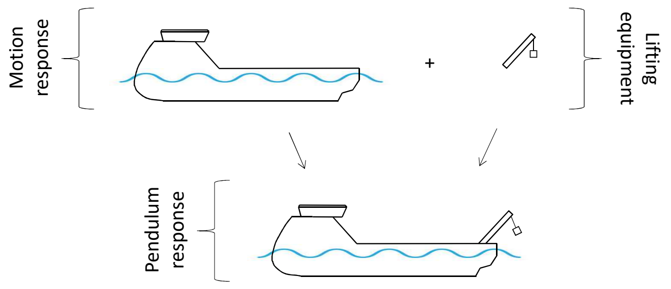


FIGURE 4. SHIP MOTION RESPONSE INDUCING A PENDULUM MOTION IN THE INSTALLED LIFTING EQUIPMENT.

Time-Domain Response of Hull with Equations of Motion

According to Fossen and Fjellstad [12], the ship model movement can be represented by Newton's second law with six degrees of freedom with the following equation:

$$\ddot{\eta} = \frac{[J^{-1} \cdot F - (C_{RB}(\dot{\eta}) + C_A(\dot{\eta})) \cdot \dot{\eta} - B \cdot \dot{\eta} - C(\eta)]}{M_{RB} + M_A} \quad (2)$$

Assuming the initial states $\dot{\eta}$ and η as known, it is possible to solve Eqn. (2) to calculate the acceleration of the rigid body. The force F represents the sum of external forces applied to the rigid body, e.g., waves, current or mooring forces. Thus, this formulation is suitable for a wide range of marine operations. However, this work simplifies the equation by considering the hull floating freely on still water, so the only one external force acting on the body is the gravity.

One difficulty encountered when solving Eqn. (2) is to estimate the added mass and damping matrix, since this information relies on hydrodynamic effects that are usually represented by non-linear formulations. However, for some conditions related to the amplitude of movement and ship main dimensions, it is possible to identify closed-form expressions for those parameters in the literature. The added mass matrix is included in the simulation with the formulation presented by Bergdahl [13], while the damping coefficients are inserted by the user and taken as constants.

The system of equations can be solved with the Dormand-Prince method, which allows calculation of the position and velocity components of the rigid body for each time step, therefore simulating the ship movement over time. Part of the code for the equations of motion was adapted from an open source application previously developed by Monteiro et al. [14].

State Handling with Vessel.js

The states calculated with the models presented in this section are stored in a ship state object in the Vessel.js library. The object organizes the states in two categories: discrete and continuous, as illustrated in Fig. 5. Discrete states are assumed to remain constant for longer intervals during the simulation, such as the loading condition, floating condition and the response amplitude calculated with the closed-form expressions for regular waves. They are stored in groups and are marked with cache systems in order to identify when the stored results need to be recalculated. The continuous states experience continuous variation during the simulation, and thus need to be stored and modified constantly, such as the instantaneous positions of the ship and of the lifted load. They are directly linked to the 3D visualizations and are updated at the simulation's frame rate.

TIME-DOMAIN SIMULATIONS

The dynamic simulations are performed by combining the states as described in the previous section to simulate vessel behavior over time. The simulations presented in the next section grow in scope from behavior of one hull to behavior of several vessels and accompanying subsystems. Continuous states are

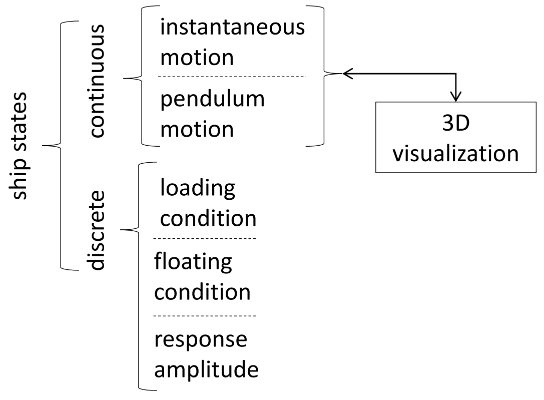


FIGURE 5. SCHEME OF THE SHIP STATE OBJECT IN THE VESSEL.JS LIBRARY.

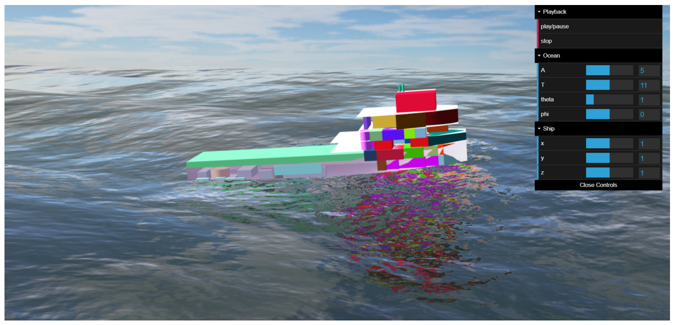
synchronized with the visualization, and therefore are calculated at the update rate of the visualization. Discrete states are calculated at the beginning of the simulation and are only updated in case there is a significant change which requires updating, e.g., recalculation of stability coefficients due to rearrangement of weights inside the vessel.

The applications use the Vessel.js features to allow virtual prototyping of vessel and subsystems behavior. Relevant parameters of the simulations such as wave characteristics and design dimensions are linked to interactive dashboards, allowing interaction of the user with the simulations to evaluate performance of different design proposals under different sea conditions. Every time the user modifies the simulation with a new parameter such as wave period, wave height, vessel main dimension or lifting equipment dimensions, the applications recalculate the results and updates the visualization accordingly. The mathematical models are lightweight enough to allow the web browser to execute all operations locally in real-time.

Single Hull Motion Response

The first simulation assesses motion response of a single hull subjected to regular waves, as shown in Fig. 6. By default, the simulation loads with a PSV model. The main dimensions of the model (length, beam and draft) can be scaled by the user, and the simulation automatically updates with the results for the scaled ship. When one dimension is modified by the user, the entire design is scaled, which includes recalculation of tank capacities, structural weight and weight distribution, which in turn influences the floating condition. The user can also configure the amplitude, period and direction of the incident wave. The wave length is automatically adjusted based on the dispersion relation for deep waters considering the chosen period. For any modification the user performs in the simulation parameters, the results are automatically recalculated and rendered.

The flowchart in Fig. 7 shows the main components of this simulator grouped in input, calculation (process) and output. The following paragraphs explain each represented component fol-



A 5
T 11
theta 1
phi 0
x 1
y 1
z 1

FIGURE 6. SCREENSHOT OF THE SINGLE HULL MOTION SIMULATOR WITH INTERACTIVE DASHBOARD.

lowing the numbering convention in the figure:

0. GUI: a graphical user interface with simple sliders allows the user to control the ship main dimensions and the wave parameters in the simulation (items 1.2 and 1.3, respectively).
1. Input: the 3D files and ship specification define the ship object and 3D model. The ship dimensions and wave parameters are simulation inputs that can be modified while the application is being executed.
    1.1. Ship specification (.json): a JSON ship specification as described previously in the section Virtual Prototyping with Vessel.js.
    1.2. Ship dimensions: the user can scale the main dimensions (length, beam and depth) of the ship specification. This redefines the hull, structure and compartments by proportionally applying the scaling coefficient. The scaled 3D model is automatically displayed with the recalculated motion response as the user defines the scaling coefficients.
    1.3. Wave parameters: the user can modify the wave period, amplitude and direction angle. The wave length is defined based on the period, by applying the dispersion relation for deep waters. The wave geometry is automatically adjusted in the visualization as the user defines the wave parameters.
    1.4. 3D files (.stl): it is possible to use stored STL files to display in the 3D vessel model. They need to be referred in the ship specification in order to be included.
2. Calculation: calculation is handled with objects encapsulating relevant parts of the simulation, which can be reused in other applications.
    2.1. Ship object: the Vessel.js ship object, defined from the JSON specification.
    2.2. Motion state calculation object: a module containing the closed-form expressions for estimation of ship motion response amplitude.
    2.3. Ocean rendering library: an open source Three.js water shader library [15] used to render an ocean with a single regular wave.
    2.4. Ship 3D model: a Three.js ship 3D model generated from the JSON specification and the 3D files, as shown in Fig. 1.
3. Output: the output is the rendered scene, which can be decomposed in two main components that are reproduced in synchronization.
    3.1. Ship motion visualization: the ship motion is visualized by moving the 3D model in the scene according to a sinusoidal function with the motion response amplitude.
    3.2. Wave visualization: the wave is generated with the water shader library according to the parameters defined in the GUI sliders.

While the motion visualization in this simulation is similar to a previous work [4], there are important differences between the approaches of the two applications. The previous simulator performed all the calculations based on the minimum set of design characteristics required to estimate the motion response

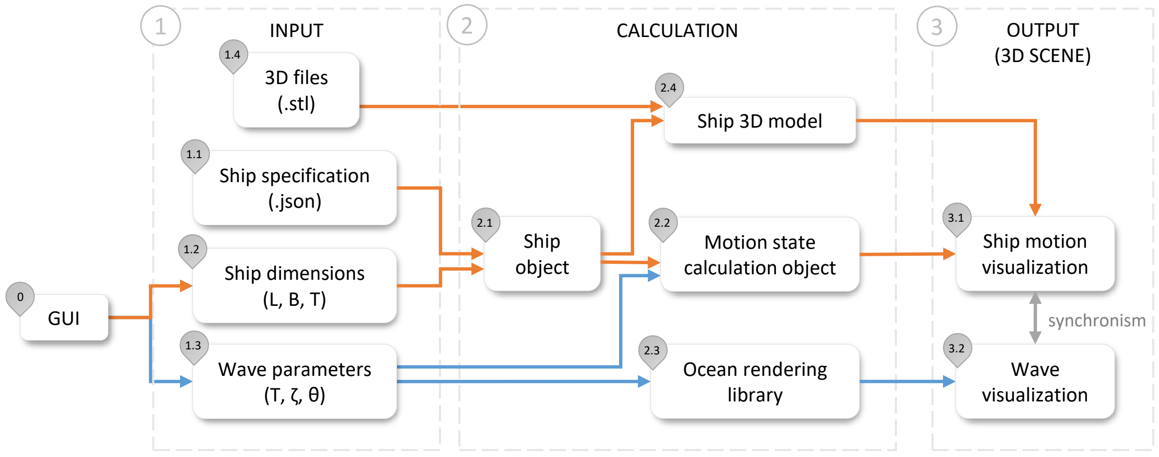
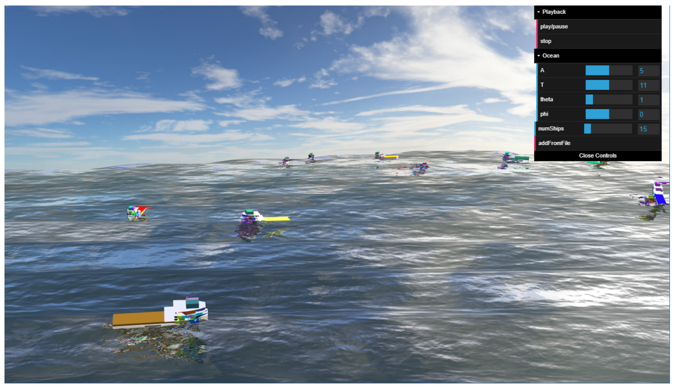


FIGURE 7. FLOWCHART OF THE SINGLE HULL SIMULATOR. INSPIRED BY CHAVES AND GASPAR [4].

with the closed-form model and rendered the visualization with a simplified 3D model of the vessel. The new simulator uses a ship design defined with the Vessel.js library and estimates the motion response based on the characteristics derived for a certain state of that design. While the first version of the simulator is basically a 3D visualization of the wave motion response, the new one works as an extension of the Vessel.js library, providing the same 3D motion visualization for a design defined by the user.

Multibody Motion Response

The second simulation calculates motion response of multiple vessels simultaneously subjected to regular waves, as shown in Fig. 8. It is very similar to the previous simulation, but adjusted to calculate and render the motion response for multiple floating vessels.



A 5
T 11
theta 1
phi 0
numShips 15

FIGURE 8. SCREENSHOT OF THE MULTIBODY MOTION SIMULATOR.

The flowchart in Fig. 9 illustrates how this is done with an object-oriented approach. The new flowchart is similar to the previous one (Fig. 7), but with the components related to the ship motion reproduced to account for multiple hulls floating simultaneously. The GUI allows the user to add hulls to or remove them from the simulation. The flowchart exemplifies a simulation with two hulls, but the same structure can be expanded to more ship instances.

These ship instances are encapsulated and handled independently, which is a suitable approach for the evaluation of multiple motion response amplitudes with different calculation parameters. This way, the main script can perform the required calculations for each ship by invoking a method in the corresponding object, then access the results to move the ship 3D model in the visualization. The simulation does not consider wave interaction due to the presence of multiple vessels.

Pendulum Motion of Lifted Load

The pendulum application is similar to the Single Hull Motion simulator, but with the addition of an A-frame with pendulum motion, as shown in Fig. 10. The motion of the pendulum responds in real-time to the ship motion, which in turn is influenced by the wave parameters set by the user (wave amplitude, period and direction).

The organization of the simulator is very similar to the flowchart in Fig. 7, but adapted to include a geometric definition of the A-frame, contained in an object, and a 3D model of the A-frame generated automatically from that definition. Furthermore, a new module is also necessary to calculate the pendulum states, position and velocity, induced by the hull response to waves.

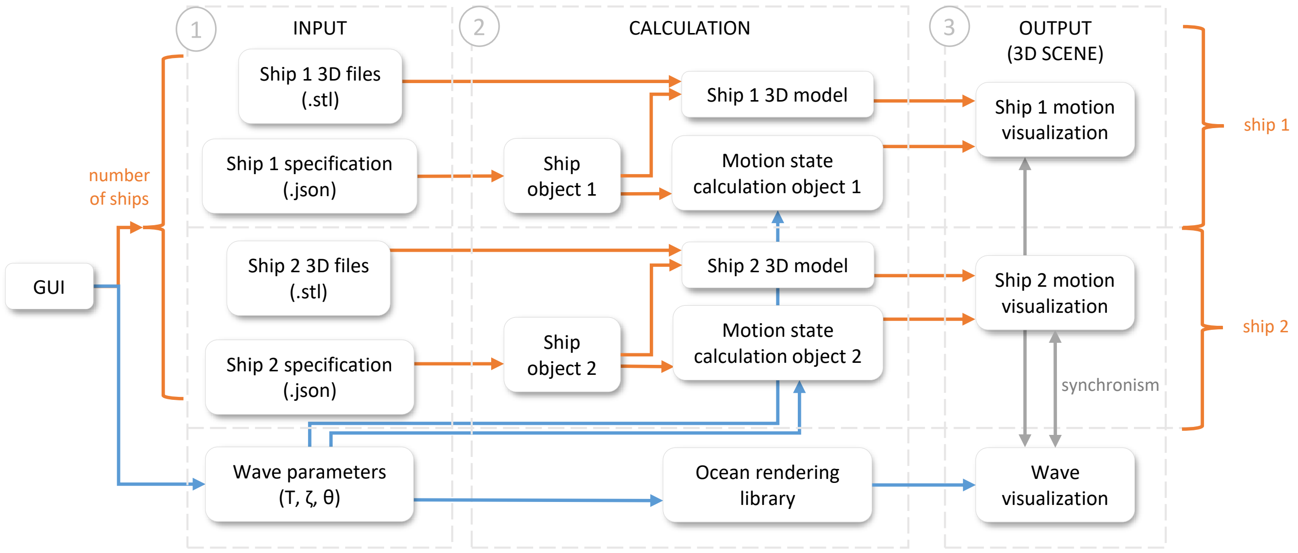
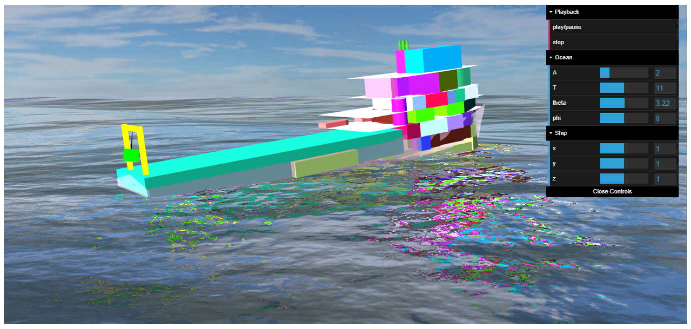


FIGURE 9. FLOWCHART OF THE MULTIBODY SIMULATOR.



A 2
T 11
theta 3.22
phi 0
x 1
y 1
z 1

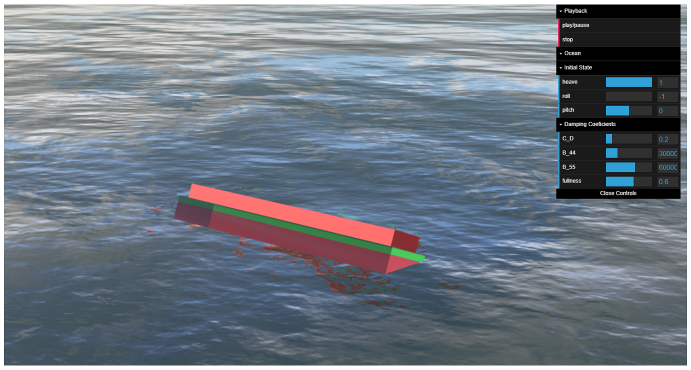
FIGURE 10. SCREENSHOT OF THE PENDULUM MOTION SIMULATOR.

Motion of Free Floating Hull

Differently from the previous examples, this simulation does not use predefined equations based on experimental methods to evaluate hull motion. Instead, it uses the equation of movement to calculate the vessel position over time. Fig. 11 shows a barge with initial heave and roll conditions different from zero. As the simulation advances, the barge will oscillate until all energy dissipates due to damping. The box-shaped barge geometry was chosen because it is simple to represent its motion coefficients, particularly the added mass. However, in the future it is possible to use more complex formulations in order to represent added masses for distinct types of geometries.

The flowchart in Fig. 12 shows the components of the simulator. Note that this simulation does not use any parameter to configure the ocean because it is always considered to be in the calm water condition. It is worthwhile to have a deeper look into two items from the chart inputs, because they are distinct from the other simulations:

1.3. Initial state: the user can change the vessel's heave, roll, and pitch in order to simulate its movement trough time. The initial state will be changed in the ship state object, which is translated to the movement of the ship 3D model in the visualization.
1.4. Damping coefficients: these are the coefficients responsible for the movement decay. C_D is used to calculate the damping in the three linear directions (surge, sway and heave), which are taken as the same magnitude in the simulation. $B_4$4 and $B_5$5 account for the damping in roll and pitch, respectively.



heave 1
roll -1
pitch 0
C_D 0.2
B_44 30000
B_55 60000
fullness 0.6

FIGURE 11. SCREENSHOT OF THE FREE MOTION SIMULATOR.

DISCUSSION

The web applications presented in the work were successful in performing time-domain simulations of motion with 3D visualizations in real-time on the client-side (that is, without relinquishing the calculations to a server). The framework for state handling, which had its development started in previous works [7], now has its foundations in place, providing capability to handle both discrete and continuous states during a simulation.

At this point, the web applications still present some limita-

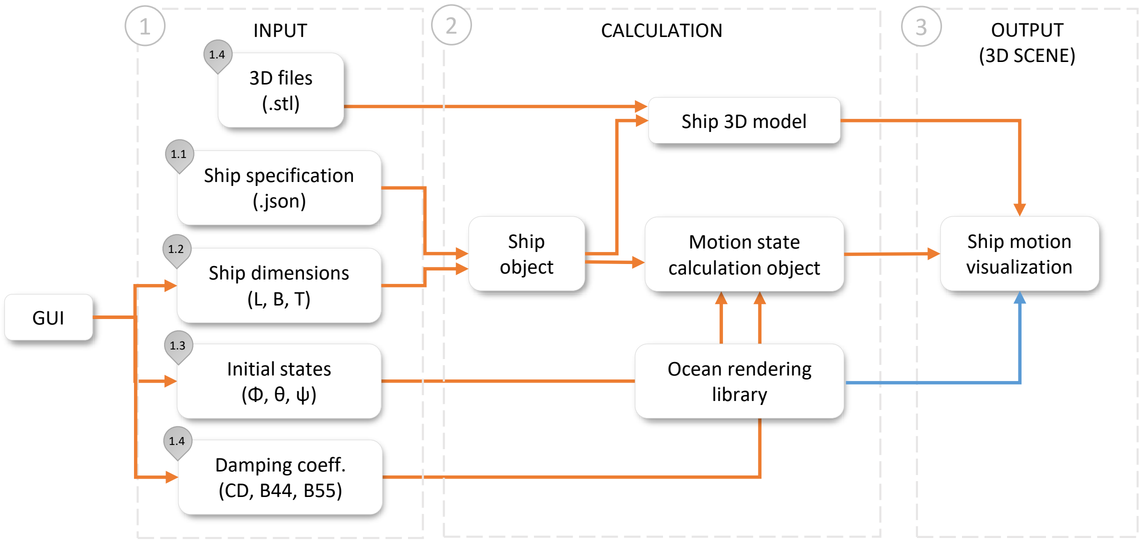


FIGURE 12. FLOWCHART OF THE FREE MOTION SIMULATOR.

tions in terms of scope and accuracy to account for the simulation of an entire marine operation. Heave and pitch responses calculated with the closed-form expressions are exaggerated in some cases, as acknowledged in the source material [8]. Likewise, the motion simulation based on equations of motion is very incipient, and does not yet account for interaction with waves.

However, the applications put to test the potential of the web-based approach and serve as a starting point for the forthcoming work. The approach yields useful features regarding accessibility of simulation models, allowing one to configure them online and share with geographically distributed users with minor complications. It also demonstrated the potential of web technologies in supporting user interaction, allowing creation of interfaces and visualizations, and open source development, with multiple open libraries available for usage in engineering applications.

The simulations gives the first step towards simulation of motion with differential equations for the Vessel.js library. Given the computational performance of the developed applications, the web-based approach still provides potential to accommodate more demanding mathematical models. In the future, they could be further developed to incorporate strip-theory methods.

CONCLUSION

This work presented a web-based approach to ship virtual prototyping and simulation of marine operations. The approach is applied to web-applications with simulations of motion response of a single hull in regular waves, of multiple hulls in regular waves, of a load lifted from an A-frame and of a hull floating in still-water. The motion response of the hulls in regular wave is calculated with closed-form expressions, while the motion of the load and of the hull in still water are calculated with by solving the equations of motion numerically.

The web-based and open source approaches supported the development of the applications, allowing interactive visual presentation, assuring accessibility and compatibility of the applications, and allowing usage of various open libraries, more notably Vessel.js.

FUTURE WORK

At the moment, the development of the Vessel.js library is focused towards simulations of subsea operations and motion interaction between vessel and mooring or towing lines.

Furthermore, the library may also be linked to FMI, which is now being proposed as a standard for exchange of simulation models in the maritime industry [16]. This would provide the benefits of the web-based approach to the FMI simulations, while allowing organization of the functional mock-up units in a more comprehensive framework supported by the library.

SOURCE CODE

The webpage of the Vessel.js library can be accessed via the address: `https://vesseljs.org/`. Besides the source code, it includes documentation, examples of applications and live tutorials. The library and webpage are currently under active development and should still undergo improvements after the publication of this work. As the project aims to be collaborative, anyone is welcome to use and contribute with the project.

ACKNOWLEDGMENT

This research is connected to the Ship Design and Operation Lab at NTNU in Ålesund. The research is partly supported by the EDIS project, in cooperation with Ulstein International AS (Norway) and the Research Council of Norway, and by the INTPART Subsea project in cooperation with the University of São Paulo (USP) and the Research Council of Norway.

REFERENCES

[1] Gaspar, H. M., 2017. "JavaScript applied to maritime design and engineering". In 16th Conference on Computer and IT Applications in the Maritime Industries, pp. 253–269.

[2] Harries, S., Lorentz, K., Palluch, J., and Praefke, E., 2018. "Appification of propeller modeling and design via CAESES". In 17th Conference on Computer and IT Applications in the Maritime Industries, pp. 292–307.

[3] Hatledal, L. I., Schaathun, H. G., and Zhang, H., 2015. "A software architecture for simulation and visualisation based on the functional mock-up interface and web technologies". In Proceedings of the 56th Conference on Simulation and Modelling, Linköping University Electronic Press.

[4] Chaves, O., and Gaspar, H., 2016. "A web based real-time 3D simulatorfor ship design virtual prototype and motion prediction". In 15th Conference on Computer and IT Applications in the Maritime Industries.

[5] Gaspar, H. M., 2018. "Vessel.js: an open and collaborative ship design object-oriented library". In Marine Design Conference (IMDC'18).

[6] He, B., Wang, Y., Song, W., and Tang, W., 2015. "Design resource management for virtual prototyping in product collaborative design". *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture,* ***229***(12), pp. 2284–2300.

[7] Fonseca, Í. A., 2018. "An open and collaborative object-oriented taxonomy for simulation of marine operations". Master's thesis, NTNU.

[8] Jensen, J. J., Mansour, A. E., and Olsen, A. S., 2004. "Estimation of ship motions using closed-form expressions". *Ocean Engineering,* ***31***(1), pp. 61–85.

[9] Myhre, T. A. Spherical pendulum dynamics. Available at `https://www.torsteinmyhre.name/snippets/spherical_pendulum.html`.

[10] Myhre, T. A., and Egeland, O., 2016. "Collision detection for visual tracking of crane loads using a particle filter". In IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society, IEEE.

[11] Loisel, S. Numeric.js. Available at `https://github.com/sloisel/numeric`.

[12] Fossen, T. I., and Fjellstad, O.-E., 1995. "Nonlinear modeling of marine vehicles in 6 degrees of freedom". *Mathematical Modeling of Systems,* ***1***(1), pp. 1–11.

[13] Bergdahl, L., 2009. "Wave loads on and motions of a ship in regular waves". In Wave-induced loads and ship motion, pp. 65–112.

[14] Monteiro, T. G., Xu, J., and Gaspar, H. M. Animated linear roll + heave ship model (6dof model). Available at `http://www.shiplab.hials.org/app/6dof/`.

[15] Bouny, J. Ocean - realistic water shader for three.js. Available at `https://github.com/jbouny/ocean`.

[16] Chu, Y., Hatledal, L. I., Æsøy, V., Ehlers, S., and Zhang, H., 2017. "An object-oriented modeling approach to virtual prototyping of marine operation systems based on functional mock-up interface co-simulation". *Journal of Offshore Mechanics and Arctic Engineering,* ***140***(2), nov, p. 021601.