Politecnico di Milano
Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Ingegneria Meccanica

Hydrodynamics of Offshore Floating Wind Turbines: Modeling for Hardware In the Loop Implementation

Relatore: Prof. Marco BELLOLI
Correlatore: Ing. Ilmas Andrea BAYATI

Tesi di laurea di:
Marco VILLA Matr. 787847

Anno Accademico 2013–2014

Ringraziamenti

Ringrazio il Prof. Marco Belloli e l' Ing. Ilmas Bayati per avermi dato l'occasione di cimentarmi in un lavoro di tesi che mi ha appassionato sin da subito. Soprattutto ringrazio Ilmas, non solo per tutti i consigli e supporto datomi, ma anche per le chiaccherate e risate che mi faranno avere sempre un bel ricordo di quelli che sono stati questi mesi di tesi.
Non posso che ringraziare con tutto il cuore quelli che sono stati gli amici con i quali ho condiviso questi anni di università. Luigi, Edoardo e Lorenzo, che sono stati effettivamente compagni di corso, ma anche Giovanni, Paolo e Fabio, che ho conosciuto in biblioteca. In particolare ringrazio Fabio per avere reso le domeniche a "Piola" un appuntamento che sicuramente troverò a mancare negli anni futuri. Ricorderò sempre con molto affetto e nostalgia le giornate passate in "pk" assieme a loro, tra risate, scleri e le decisioni poco democratiche su dove andare a pranzare.
Infine mi rivolgo ai miei genitori, Adela e Renato. Li ringrazio per essere il mio punto di riferimento, per avermi concesso la fortuna di poter studiare senza dovermi preoccupare di nient'altro, per sopportare soppratutto il mio difficile carattere e per volermi incondizionatamente bene.
Dedico questo lavoro di tesi a loro e ai miei nonni, Giuseppina e Luigi, che tanto avrei voluto avere vicino a me giunti alla fine di questo percorso.

“Valar Morghulis”

Abstract

The main purpose of the following work is to propose a numerical code able to predict the dynamics of a floating wind turbine spar-buoy platform and also to be implemented in a real time "hardware-in-the-loop" system.
The numerical model is designed to solve the dynamics, both for regular and irregular sea state, of a two degrees of freedom system: surge and pitch displacements. The code is structured in different subroutines, each one deals with a different kind of phenomena involved in the hydrodynamics of a floating body maintained in its position by a mooring lines system.
Concerning the memory effect term of the hydrodynamical radiation problem, that in its traditional representation consists in a convolution integral, a fundamental part of this work was dedicated to replace the highly time consuming convolution integral with a simpler state space model representation in order to make possible the implementation of the numerical code in the real time "hardware-in-the-loop" system.
A set of numerical tests were performed taking as wind turbine model the OC3 Hywind. Numerical results are consistent with the ones obtained by FAST, that is considered the simulator code reference in the present day.
Surge and pitch degrees of freedom were chosen in order to support the second part of the work, which consists in the implementation of the code in a 2 degrees of freedom mechanism designed for simulate floating wind turbine hydrodynamics in a wind tunnel facility. The wave simulator mechanism was designed and validated at Politecnico di Milano, and it is able to represent a translational and a rotational motion of the platform. This mechanism is designed so that a scale wind turbine model can be mounted on it, with the possibility to analyze different configurations. The wave simulator mechanism is moved by hydraulic actuators whose inputs are consistent with the real time "hardware-in-the-loop" system that elaborates the acquisitions gathered from the measurement chain. The measurement chain consists in two balances, one at tower base and the other inside the nacelle, two mems accelerometers and the control of the MTS actuators.
In order to take into account the aerodynamic load into the numerical model, which provides platform displacements, a particular data processing of tower base balance and accelerometers acquisitions was performed. A set of experimental tests were performed on order to create the calibration matrix needed for this operation.

Sommario

Lo scopo del seguente lavoro di tesi consiste nel proporre un modello matematico in grado di riprodurre l'idrodinamica di una turbina eolica offshore di tipo spar-buoy, e di poter essere implementato in un sistema "hardware-in-the-loop" per la simulazione in tempo reale il forzamento delle onde indotto sulla piattaforma flottante.
Il modello numerico sviluppato per questo lavoro è in grado di rappresentare diversi stati di mare, sia di tipo regolare che irregolare, e in funzione di questo riprodurre gli spostamenti di un modello di piattaforma a due gradi di libertà: surge e pitch. Il codice numerico è in grado di risolvere i diversi fenomeni che che governano l'idrodinamica di una piattaforma spar-bouy mantenuta in una posizione fissa mediante un sistema di ormeggio.
Per quanto concerne l'effetto memoria del termine di idrodinamico di radiazione, che nella sua formulazione classica è rappresentato da un integrale di convoluzione, gran parte di questo lavoro di tesi è stato dedicato alla trasformazione di tale termine in un semplice modello in forma di stato. Solo in questo modo è stato possibile implementare il modello numerico in un sistema "hardware-in-the-loop" per la simulazione in real time della dinamica della piattaforma.
Surge e pitch sono stati scelti in modo da poter essere di supporto alla seconda parte del lavoro di tesi. Infatti un sistema a due gradi di libertà per la simulazione della dinamica delle onde, progettato e validato al Politecnico di Milano, capace di rappresentare un grado di libertà di traslazione ed uno di rotazione, è stato utilizzato come riferimento per la configurazione del modello "hardware-in-the-loop". Questo meccanismo è stato progettato in modo che venga montato un modello in scala di turbina eolica su di esso, con la possibilità di analizzare diverse configurazioni. Il meccanismo di simulazione delle onde è movimentato da attuatori idraulici, i cui input sono consistenti con il sistema "hardware-in-the-loop" che elabora le misure acquisite dalla catena di misura. Le misure acquisite sono trasformate in un termine che rappresenta il forzamento aerodinamico agente sulla turbina e viene dunque inserito nell'equazione di moto del modello numerico.
Sono state svolte delle prove sperimentali, fuori dalla galleria del vento, al fine di ottenere la matrice di taratura necessaria per questa operazione.

Contents

List of Figures

List of Tables

Introduzione

Oggigiorno lo scenenario delle energie rinnovabili è in continua espansione ed evoluzione. Queste non solo risultano essere una proposta alternativa agli impianti di potenza basati su combustibili fossili, ma anche l'occasione di sfruttare a pieno le risorse naturali messe a disposizione dal nostro pianeta, che per loro intrinseca caratteristica si rigenerano alla stessa velocità con la quale vengono consumate.

Una delle tecnologie più affermate per la produzione energetica da fonti rinnovabili è l'energia eolica, principalmente grazie all'utilizzo di aerogeneratori che trasformano l'energia posseduta dal vento in energia elettrica. Questa è una tecnologià ampiamente consolidata che sta portando velocemente alla saturazione dei siti, sia onshore che offshore, adatti a questo tipo di installazione. Per quanto concerne le installazioni offshore, attualmente queste sono ubicate in acque non paritcolarmente profonde (30-40 m) e adottano le tipiche caratteristiche dei prototipi utilizzati sulla terra ferma.

La necessità di sopperire alla continua crescita di domanda energetica da parte della popolazione, combinata all'esigenza di preservare la società civile da inquinamento visivo ed acustico correlato al funzionamento delle turbine stesse, stà muovendo la comunità scientifica nel cercare soluzioni per l'installazione di parchi eolici sempre più lontano dalla costa.

Diverse sono le proposte avanzate dalla comunità scientifica per la progettazione di piattaforme galleggianti destinate all'istallazione di turbine eoliche. Esse differiscono tra loro per il tipo di base galleggiante, per il sistema di ormeggio e per dimensione. Inoltre, le caratteristiche dello specifico sito di installazione comportano una progettazione della turbina stessa che deve abbandonare i soliti canoni dei prototipi on-shore.

E' dunque necessario lo sviluppo di modelli numerici per la simulazione del forzamento aero- e idro-dinamico, nell'ottica di trovare soluzioni progettuali in grado di massimizzare il rendimento della turbina e che allo stesso tempo garantiscano una opportuna progettazione a fatica degli organi di macchina.

La correzione e validazione di tali modelli necessita un confronto con misure su turbine eoliche al vero o su modelli in scala testati in vasche navali e/o presso gallerie del vento. Dal momento che la prima alternativa non è percorribile, essendo questa tecnologia molto recente e particolarmente costosa, è necessario disporre ambienti speriementali in grado di riprodurre in scala, sia da un punto di

vista aerodinamico che idrodianmico, le condizioni di esercizio a cui sono soggette tali turbine eoliche.
I primi tentativi sperimentali sono stati eseguiti all'interno di vasche navali, dove è possibile riprodurre al meglio l'iterazione onda-piattaforma. Tuttavia queste strutture di ricerca sono carenti per quanto concerne l'aspettto aerodinamico dal momento che dispongono esclusivamente di grandi ventilatori non in grado di garantire il controllo del flusso incidente, sia in termini di precisione del livello di turbolenza sia in termini di definizione del profilo dello strato limite.
Il presente lavoro di tesi si pone l'obbiettivo di sviluppare un codice numerico in grado di riprodurre l'idrodinamica di turbine eoliche flottanti di tipo spar-buoy e che possa essere implementato in un sistema di movimentazione per la simulazione sperimentale delle codizioni di funzionamento di tali macchine.
Le piattaforme galleggianti di tipo spar-buoy hanno forma cilindrica con asse di simmetria coincidente con quello della turbina. Generalmente sono costituite da due regioni cilindriche di diversa dimensione, collegate da una terza regione il cui diametro varia linearmente da una regione all'altra.
Il codice numerico è strutturato in diverse sub-routines comunicanti tra loro. Fondamentalmente ciascuna sub-routine rappresenta e risolve uno dei tanti fenomeni coinvolti nelll'idrodinamica di queste strutture. Questi termini vengono poi predisposti in una opportuna formulazione, in modo che possano essere risolti nel dominio del tempo, sia con metodi di integrazione a passo fisso che a passo variabile.
Durante lo svolgimento del lavoro, molta enfasi è stata posta nella modellazione dell'effetto memoria idrodinamico. Il nome deriva dal fatto che il carico esercitato su una struttura galleggiante da parte delle onde dipende direttaemente dal loro stato, il quale è fortemente influenzato da quelli che sono stati gli spostamenti passati della struttura. L'effetto memoria è un fenomeno che ha luogo ogni qual volta una struttura interagisce con un fluido con densità dello stesso ordine di grandezza della struttura stessa.
La formulazione classica dell'equazione di moto idrodinamica nel dominio del tempo prevede la rappresentazione di questo termine mediante un integrale di convoluzione. Questo richiede un sforzo computazionale maggiore rispetto agli altri termini che compongono l'equazione di moto ed inoltre è poco adatto per implementare logiche di controllo o per sperimentazione per applicazioni in real time.
Il codice prevede a proposito una formulazione alternativa per la rappresentazione dell'effetto memoria. Mediante tecniche di identificazione nel dominio delle frequenze si può ottenere una formulazione parametrica della funzione di memoria, la quale può essere trasformata in un sistema lineare in forma di stato di facile introduzione nel modello numerico e che comporta anche benefici dal punto di vista nel costo computazione.
Il codice di calcolo risolve la dinamica di un modello a due gradi di libertà: surge e pitch. Le ragioni di questa scelta sono dovute alla necessità di assumere in prima istanza padronanza e controllo di un modello matematico complesso che rappre-

senta la dinamica di un sistema multicomposto (piattaforma, turbina, ormeggio e onde). Inoltre, lo scopo ultimo del codice è quello di essere implementato in un sistema si movimentazione a due gradi di libertà, uno di traslazione ed uno di rotazione, nell'ottica di simulare sperimentalmente in galleria del vento la dinamica di un modello di turbina in scala montato su di esso.
Tale meccanismo è azionato mediante due attuatori idraulici, uno per ciascun grado di libertà, in modo da poter studiare diverse configurazioni di galleggiamento. In questo lavoro è stata predispota una procedura di controllo del moto ondoso in tempo reale mediante un sistema "hardware-in-the-loop", coerente con i calcoli idrodinamici ottenuti dal modello numerico. Per tenere conto dei carichi aerodinamici nel modello numerico è stata opportunamente studiata una catena di misura in grado di tradure le misure acquisite sul modello di turbina in un opportuno termine aggiuntivo nell'equazione di moto.
Per quanto concerne la formulazione del codice numerico al fine di essere implemento in real time, è fondamentale rappresentare l'effetto memoria in forma di stato. Infatti, il costo computazionale richiesto dall'integrale di convoluzione non è idoneo per una simulazione sperimentale "hardware-in-the-loop" in tempo reale.
Sono state svolte prove numeriche prendendo come riferimento il modello di turbina eolica OC3 Hywind. I risultati sono consistenti con quelli presenti in letteratura e forniti da FAST, che è da riternersi il codice di riferimento per questo genere di applicazioni.
Per quanto concerne l'apetto sperimentale del lavoro, sono state fatte delle prove di taratura ai fini di ottere una opportuna matrice di taratura necessaria all'estrapolazione della forza aerodinamica da inserire nel codice.

Chapter 1

Offshore Floating Wind Turbines

1.1 Wind Energy

As is common knowledge, nowadays wind energy employment is the result of a continuos refining process of an energy production method which has been exploited by humans since the old age.

For more than two millennia wind power machines have ground grain and pumped water through mills (figure 1.1a). Since the development of electric power, wind power found new applications through wind turbines (figure 1.1b) in lighting buildings remote from centrally-generated power. Throughout the 20th century parallel paths developed small wind plants suitable for farms or residences, and larger utility-scale wind generators that could be connected to electricity grids for remote use of power. Today wind powered generators operate in every size range between tiny plants for battery charging at isolated residences, up to near-gigawatt sized offshore wind farms that provide electricity to national electrical networks. A wind farm is a group of wind turbines in the same location used for production of electricity. A large wind farm may consist of several hundred individual wind turbines distributed over an extended area. A wind farm may also be located offshore.

Wind power is an affordable, efficient and abundant source of domestic electricity. It's pollution free and in some countries tends to be cost competitive with energy from coal and gas fired power plants. It is also a renewable energy which means that comes from resources which are naturally replenished on a human timescale. Renewable energies are derived from natural processes that are replenished constantly. In its various forms, wind energy derives directly from the sun.

Wind energy is the kinetic energy of air in motion. Wind power in an open air stream is proportional to the third power of the wind speed. Even though the obtainable energy from a wind flow can't be higher than the 60% (Betz Theory, [2]) so wind turbines for grid electricity need to be especially efficient.

The strength of wind varies, and an average value for a given location does not alone indicate the amount of energy a wind turbine could produce there. To assess the frequency of wind speeds at a particular location, a probability distribution function is often fit to the observed data [20].

(a) Wind Mill

(b) Wind Turbine

The wind industry has been growing rapidly in recent years. Worldwide there are now over two hundred thousand wind turbines operating, with a total nameplate capacity of 282,482 MW as of end 2012. The European Union alone passed some 100,000 MW nameplate capacity in September 2012, while the United States surpassed 50,000 MW in August 2012 and China's grid connected capacity passed 50,000 MW the same month. World wind generation capacity more than quadrupled between 2000 and 2006, doubling about every three years. At the end of 2013, worldwide nameplate capacity of wind-powered generators was 318 gigawatts (GW), growing by 35 GW over the preceding year [7], [5].

1.1.1 Offshore Wind Energy

Offshore wind power refers to the construction of wind farms in bodies of water to generate electricity from wind. Offshore wind power includes inshore water areas such as lakes, fjords and sheltered coastal areas, utilizing traditional fixed-bottom wind turbine technologies, as well as deep-water areas utilizing floating wind turbines. A subcategory within offshore wind power can be nearshore wind power [21].
Since better wind speeds are available offshore compared to on land, the installation of wind farms offshore results highly interesting. Stronger and more regular winds brings to higher efficiencies and lower operation costs. For example the more regular wind leads to less turbulent loads on blades, making fatigue less

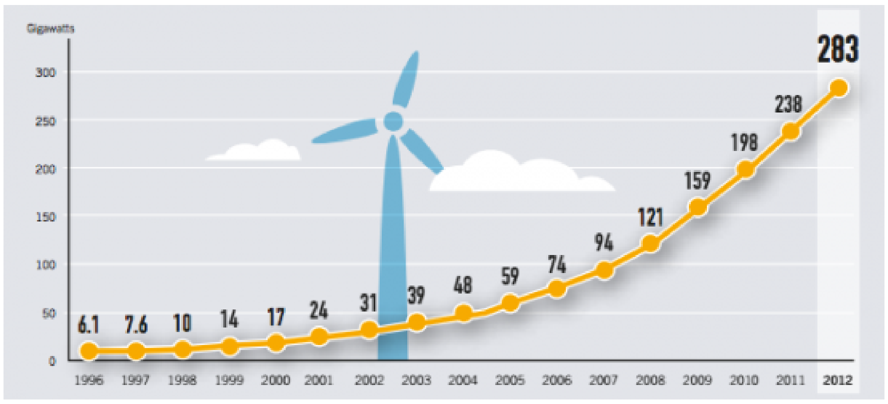


Figure 1.1: Global cumulative installed wind capacity [MW]

critical. Also opposition to construction is usually weaker because installing wind turbines sufficiently far from the shore can eliminate visual impact and noise issues and in general not interfere with daily life of the nearest population. There is no apparent limit on the the size of the turbine so it's possible to take the maximum advance of scale effect and go toward bigger rotor diameters than the ones installed on the onshore machines. Actually the bigger is rotor diameter the slowly the blades rotates, so the gearbox ration tends to increase and its efficiency to decrease [9], [25].

On the other hand, this technology entails to a lot of negative issues. First of all, according to the US Energy Information Agency, offshore wind power is the most expensive energy generating technology being considered for large scale deployment. Prices can be in the range of 2.5-3.0 million Euro/MW. Tower, foundations and underwater cabling require investments 1.5-2 times more expensive and repairs are estimated to be 5-10 times more due to the difficult availability of the turbine location [27]. Furthermore, underwater cables have a very low transportation efficiency, not only for the long distance but also for the environment, that weights a lot on the global efficiency of the conversion system [3].

Offshore wind turbines are exposed to more severe environmental conditions compared to the land based ones. Their foundations or platforms have to deal with waves and currents loads and also the possibility of ice loading and corrosion from salty water. Even the seaside salty air can promote corrosion of critical parts of the turbine, such as for example the gearbox or the generator. For this reason the nacelle needs an air condition system in order to prevent this kind of damages [10].

1.1.1.1 Offshore Wind Turbines.

Offshore wind turbines are classified is two main groups depending on the depth of the installation site. They can have ground-fixed foundations (figure 1.2) or they can be mounted on a floating structure that allows the turbine to generate electricity in water depths where bottom-mounted towers are not fiscally and economically feasible. (figure 1.3).

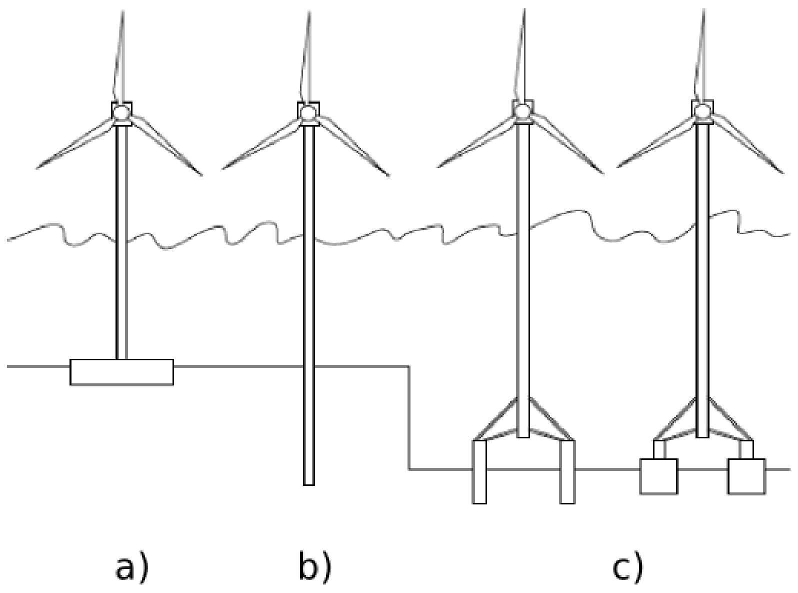


Figure 1.2: Ground Based Offshore Wind Turbines: a) Ground Based b) Monopile c) Multiple Footing Suction Caissons.

All the European offshore wind turbines installed to date are on fixed-bottom substructures. These turbines have mostly been installed in water shallower than 20 m by driving mono-piles into the seabed or by relying on conventional concrete gravity bases. The choice between these two solutions rely on the type of seabed soil. Monopile solutions are anchored into the seabed and expensive drilling operations are needed, while gravity based solutions are maintained on the sea bottom due to heavy concrete caissons fixed to the turbine tower extreme [9].
Bottom fixed offshore wind turbines are a consolidated technology. The the first offshore wind farm was installed in Denmark in 1991. On the other hand, floating wind turbines concepts are very recent and also very attracting due to the opportunity of wide ocean areas exploitation.
Numerous floating support platform configurations are possible for offshore wind turbines, particularly considering the variety of mooring systems, tanks, and ballast options that are used in the offshore oil and gas (O&G) industries. The three principal concepts, classified in terms of how the floating system achieve static stability, are:

1. A shallow drafted barge, achieving pitch restoring via water plane area moment.

2. A ballasted deep-drafted spar buoy with pitch restoring by ballasting. This concept, which can be moored by catenary or taut lines, achieves stability by

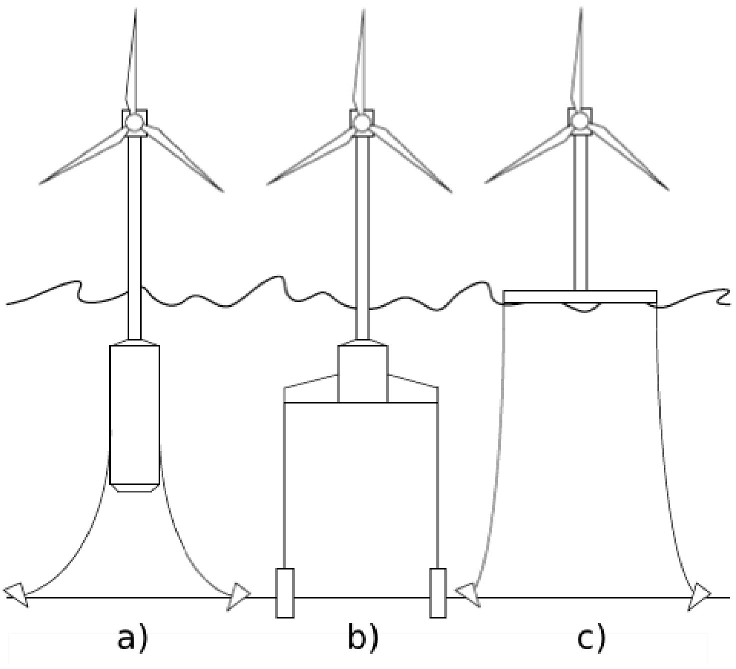


Figure 1.3: Floating Offshore Wind Turbines: a) Spar buoy b) TLP c) Barge

using ballast to lower the center of mass (CM) below the center of buoyancy (COB).

3. An unballasted tension leg platform TLP, for which pitch restoring mainly is provided by the mooring system brought about by excess buoyancy in the tank.

The barge and spar-buoy types can be anchored to the seabed either with slack catenary or with taut vertical mooring lines, but the TLP must be equipped with taut mooring lines. There are various types of possible mooring cables, such as chains, steel or synthetic fibers, or a combination of these. Numerous anchor systems exist, ranging from simple deadweight anchors and conventional “mushroom” anchors to more sophisticated screw-in and suction anchors [14].

1.2 Offshore Code Comparison Collaboration

As mentioned before, floating wind turbines are considered a new energy reality. This energy conversion systems are very recent and nowadays only a few opera-

tional deep-water floating large-capacity wind turbine are installed. One of those is the Hywind prototype, owned by Statoil, a Norwegian multinational oil and gas company.
The Hywind was towed out to Norwegian sea in early June 2009. The 2.3 MW turbine was constructed by Siemens Wind Power and mounted on a floating tower with a 100 meter deep draft. Hywind was inaugurated on 8 September 2009 and is still operating and generating electricity for the Norwegian grid.
Nowadays, the scientific community is very dedicated and put a lot of efforts in finding better floating wind farms design solutions in order to be cost-effective. Research is focused on scale modeling and computer modeling attempt to predict the behavior of large scale wind turbines in order to avoid costly failures and to expand the use of offshore wind power The research in this field include :

1. Numerical models:

   Overview of integrated dynamic calculations for floating offshore wind turbines.

   Fully coupled aero-hydro-servo-elastic response.

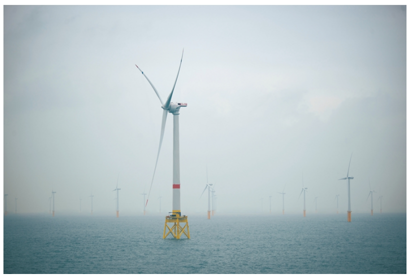
   Basic research tool to validate new designs.

2. Scale models:

   Water tank studies on 1:100 scale Tension-Leg Platform and Spar Buoy platforms.

   Dynamic response dependency on the mooring configuration.

One way to identify the inaccuracies and errors in new numerical models and simulations is to compare their outputs under a wide range of conditions. To compare the codes used to design support structures for offshore wind turbines, the Offshore Code Comparison Collaboration (OC3) began in 2004, within Subtask 2 of the IEA Wind Task 23 Offshore Wind Technology and Deployment.
Experts from eighteen organisations in ten countries made code-to-code comparisons of the same wind turbine dynamics to improve the codes predictive capability for offshore wind energy structural loads. By 2009, the collaboration completed four work packages or phases, all built on a common 5 MW reference wind turbine model developed in previous work by the National Renewable Energy Laboratory in the United States. These four phases were focus both on ground fixed offshore wind turbines and the barge, spar-buoy and TLP platforms for offshore floating wind turbines (figure 1.3).
Task 30 OC4 continues the work begun in Task 23. Over an additional four-year period (2010 through 2013). The operating agent organizations (managers) of the work are the National Renewable Energy Laboratory in the United States and the Fraunhofer Institute for Wind Energy and Energy System Technology IWES in Germany. For the OC4 project there are two work packages (1.5 years each) focused on jacket support structure and floating semisubmersible, respectively (figure 1.4).


(a) Jacket

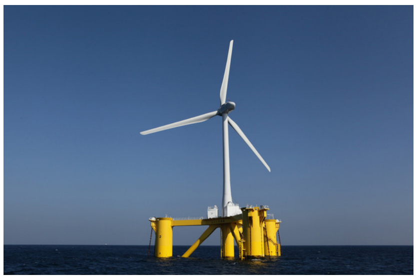

(b) Semi Submersible

Figure 1.4: OC4 phases

An experts meeting was held in 2012. Each work package and the experts meeting of verification data sets results in final reports. Both the OC3 and OC4 projects thus far have been focused on the verification of offshore wind modeling tools through code-to-code comparisons. Verification of the modeling tools is just one step towards ensuring the accuracy of the results that these tools provide. Another step is validation, which involves the comparison of simulated results to measured data from a physical test [11], [28], [23], [17]
The OC5 project is already started, but is still in a validation process.

1.2.1 Definition of the Floating System for Phase IV of OC3

This study is based on the numerical and experimental simulation of the OC3-Hywind. This wind turbine was chosen because is considered by scientific community the reference model in order to make code comparisons and a lot of data from numerical tests is available. The main characteristics of the floating wind turbine will be presented in this part of the chapter.

1.2.1.1 Wind Turbine Properties

Because of the large portion of system costs in the support structure of an offshore wind system, a deepwater wind system is cost-effective if each individual wind turbine is rated at 5 MW or higher, [16].
The 5 MW machine has a rotor radius of about 63 m. In order to minimize the overturning moment acting on an offshore substructure the hub height for the baseline wind turbine should be 90 m from mean sea level. This give a 15 meters air gap between the blade tips at their lowest point when the wind turbine is undeflected and an estimated extreme 50 year individual wave height of 30 m (i.e., 15 m amplitude).
The additional gross properties chosen for the NREL 5-MW baseline wind turbine are presented in table 1.1.

Rating	5 [MW]
Rotor Orientation, Configuration	Upwind, 3 blades
Control	Variable Speed, Collective Pitch
Drivetrain	High Speed, Multiple-Stage Gearbox
Rotor, Hub Diameter	126, 3 [m]
Blade Precone	2.5 [°]
Hub Height above SML	90 [m]
Cut in, Rated, Cut out Wind Speed	3, 11.4, 25 [m/s]
Cut in, Rated Rotor Speed	6.9, 12.1 [rpm]
Rotor Mass	110000 [kg]
Nacelle Mass	240000 [kg]
Tower Mass and CM	249,718 [kg]
Tower Top Heigth above SML	87.6 [m]
Tower Base Heigth above SML	10 [m]
CM Location of Tower above SML	43.4 [m]

Table 1.1: Gross Properties Chosen for OC3 Hywind

The (x,y,z) coordinates of the overall center of mass (CM) location of the wind turbine are indicated in a tower-base coordinate system, which originates along the tower centerline at ground or mean sea level (MSL). The x-axis of this coordinate system is directed nominally downwind, the y-axis is directed transverse to the nominal wind direction, and the z-axis is directed vertically from the tower base to the yaw bearing. The rotor diameter indicated in Table 1.1 ignores the effect of blade precone, which reduces the actual diameter and swept area. The exact rotor diameter in the turbine specifications (assuming that the blades are undeflected) is actually $(126\,m) \times cos(2.5°) = 125.88m$ and the actual swept area is $(\pi/4) \times (125.88\,m)^2 = 12,445.3\,m^2$ *[16], [15].*

1.2.1.2 Floating Platform Properties

The tower is cantilevered at an elevation to the top of the floating platform. The draft of the platform is 120 m. Between the top and bottom of the platform, the OC3-Hywind spar-buoy consists of two cylindrical regions connected by a linearly tapered conical region. The cylinder diameter of 6.5 m above the taper is more slender than the cylinder diameter of 9.4 m below the taper to reduce hydrodynamic loads near the free surface. The linearly tapered conical region extends from a depth of 4 m to a depth of 12 m below the SWL. These properties are all relative to the undisplaced position of the platform. Mass and inertia of the platform system are presented in table 1.2.

Depth to Platform Base Below SWL (Total Draft)	120 [m]
Elevation to Platform Top (Tower Base) Above SWL	10 [m]
Depth to Top of Taper Below SWL	4 [m]
Depth to Bottom of Taper Below SWL	12 [m]
Platform Diameter Above Taper	6.5 [m]
Platform Diameter Below Taper	9.4 [m]
Platform Mass, Including Ballast	7,466,330 [kg]
CM Location Below SWL Along Platform Centerline	89.9155 [m]
Platform Roll Inertia about CM	4,229,230,000 [kgm^2]
Platform Pitch Inertia about CM	4,229,230,000 [kgm^2]
Platform Yaw Inertia about Platform Centerline	164,230,000 [kgm^2]

Table 1.2: OC3 Hywind Platform Characteristics

1.2.1.3 Mooring Lines Properties

In order to simplify the analysis of the mooring system within the OC3 project, however, only the specifications of an effective system are presented for the OC3-Hywind system. Three simplifications are made. First, the delta connection is eliminated, which requires that the mooring system be augmented with a yaw spring to achieve the proper overall yaw stiffness. Second, each of the multisegment lines is replaced with an equivalent homogenous line, with properties derived as the weighted-average values of the mass, weight, and stiffness (weighted based on the unstretched lengths of each segment). Third, all mooring system damping, including the hydrodynamic drag and line-to-seabed drag, is neglected. These three simplifications are acceptable for static analysis, but may not be appropriate in all dynamical conditions. From these simplifications, the fairleads (body-fixed locations where the mooring lines attach to the platform) are located at a depth of 70.0 m below the SWL and at a radius of 5.2 m from the platform centerline. The anchors (fixed to the inertia frame) are located at a (water) depth of 320 m below the SWL and at a radius of 853.87 m from the platform centerline. One of the lines is directed along the positive X-axis (in the XZ-plane). The two remaining lines are distributed uniformly around the platform, such that each line, fairlead, and anchor is 120º apart when looking from above [15].

All the properties of the mooring system are presented in table 1.5..

Number of Mooring Lines	3
Angle Between Adjacent Lines	120 [°]
Depth to Anchors Below SWL (Water Depth)	320 [m]
Depth to Fairleads Below SWL	70 [m]
Radius to Anchors from Platform Centerline	853.87 [m]
Radius to Fairleads from Platform Centerline	5.2 [m]
Unstretched Mooring Line Length	902.2 [m]
Mooring Line Diameter	0.09 [m]
Equivalent Mooring Line Mass Density	77.7066 [kg/m]
Equivalent Mooring Line Weight in Water	698.094 [N/m]
Equivalent Mooring Line Extensional Stiffness	384,243,000 [N]
Additional Yaw Spring Stiffness	98,340,000 [Nm/rad]

Figure 1.5: Mooring Lines Characteristics

1.3 State of the Art of Floating Wind Turbines Numerical Codes

1.3.1 FAST 7

As a reference for this thesis the simulator code FAST version 7 with AeroDyn and HydroDyn modules has been used.
FAST 7 is considered nowadays (2014) the most reliable tool for the prediction of floating wind turbine performances .
FAST, which stands for Fatigue, Aerodynamics, Structures and Turbulence, is a code created, developed and verified by Jonkman and Buhl at NREL. It is a comprehensive hydro-aero-servo-elastic simulator capable of predicting both the extreme and fatigue loads of two and three bladed horizontal axis wind turbines (HAWTs). It can also extract linear state-space models for controls design and can be used to to generate MSC.ADAMS models.
The hydrodynamical code has been design especially for the spar-buoy, barge and TLP systems taking in consideration for the OC3 project, but also can solve the dynamics of a huge variety of offshore floating platforms.
The aerodynamics are calculated in the FAST module AeroDyn, which uses a state of the art blade element momentum approach (BEM) with empirical corrections to calculate the rotor aerodynamics. The empirical corrections consider the losses caused by the airflow around the blade tip and at the rotor hub. The effect of a turbulent wake state that occurs if the rotor strongly decelerates the axial airflow is considered, as well as unsteady airfoil aerodynamics and wake inertia and 3-D effects such as stall delay. AeroDyn also is able to apply the generalized dynamic wake theory to account for the effects of dynamic inflow.

The FAST module HydroDyn adds the capability of simulating time domain hy-

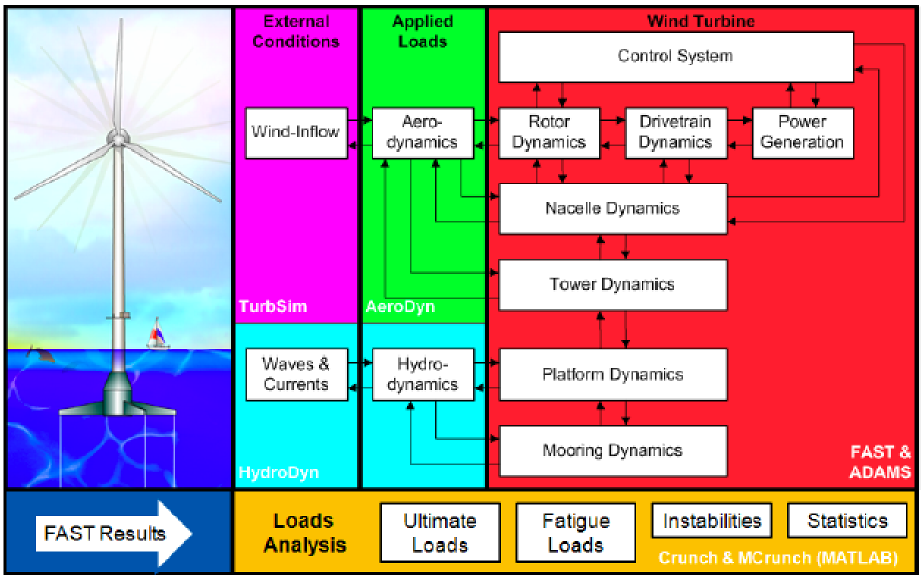


Figure 1.6: FAST v7 Modules Scheme

drodynamic effects from linear hydrostatic restoring, added-mass and damping contributions from linear radiation, including free surface memory effects. It shapes memory effect via convolution integral, as it is represented in the classical form of Cummins equation.
HydroDyn can also simulate incident wave excitation from linear diffraction, nonlinear viscous drag, including sea current loading.
The code also includes a nonlinear quasi-static mooring line module. It uses Newton Raphson method to find the tensions of the lines at the fairleads and so the restoring forces and moments acting on the platform.
These models are of higher fidelity than most of the models that have been used in the past to analyze floating turbines and which neglected important hydrodynamic and mooring system effects.
Both AeroDyn and HydroDyn for FAST 7 solves the floating wind turbine dynamics in time domain using a variable time step integration method [13].

1.3.2 FAST framework: FAST 8

NREL has recently put considerable effort into improving the overall modularity of its FAST wind turbine aero-hydro-servo-elastic tool to:

1. improve the ability to read, implement, and maintain source code.
2. increase module sharing and shared code development across the wind community.
3. improve numerical performance and robustness.

4. greatly enhance flexibility and expandability to enable further developments of functionality without the need to recode established modules.

The new FAST modularization framework supports module-independent inputs, outputs, states and parameters.
It is envisioned that the new modularization framework will transform FAST into a powerful, robust, and flexible wind turbine modeling tool with a large number of developers and a range of modeling fidelities across the aerodynamic, hydrodynamic, servo-dynamic, and structural-dynamic components.
The main differences between FAST v7 and FAST v8 are presented in figure 1.7.

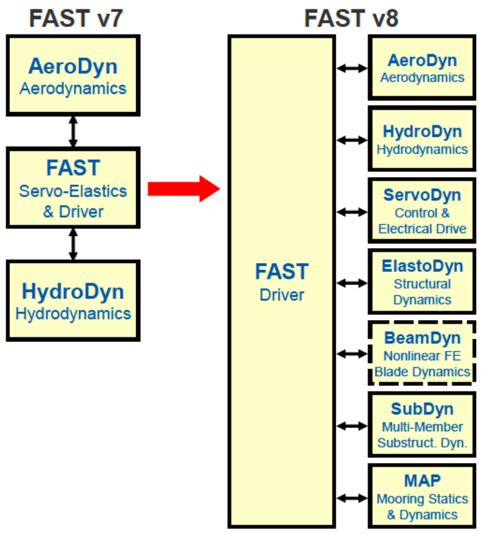


Figure 1.7: Architectural comparison of FAST 7 and FAST 8

The modules of FAST (AeroDyn, HydroDyn, etc.) correspond to different physical domains of the coupled aero-hydro-servo-elastic solution, most of which are separated by spatial boundaries. Figure 1.8 shows the control volumes associated with each module for floating offshore wind turbines [12].

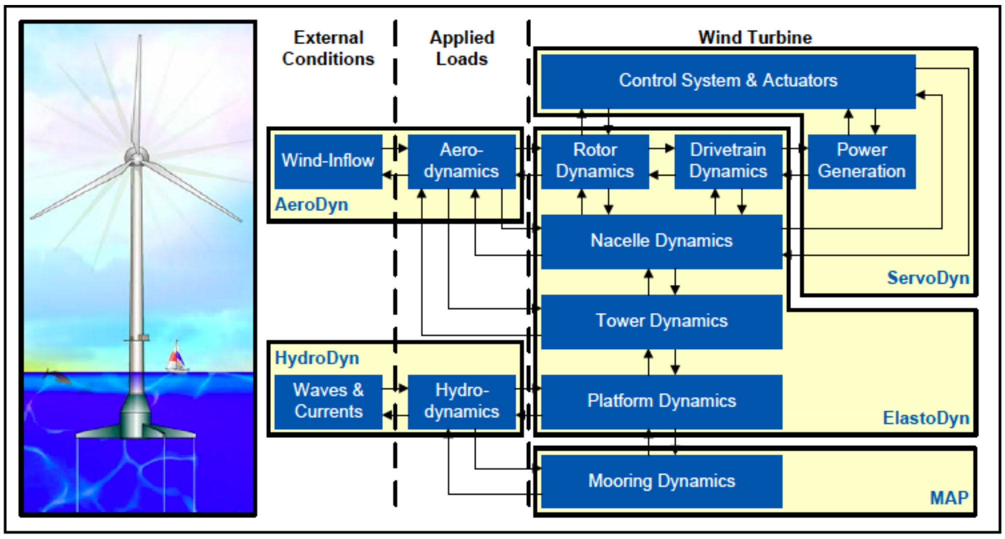


Figure 1.8: FAST control volumes for floating systems

Chapter 2

Hydrodynamical Numerical Model

This chapter presents the hydrodynamic model used for the representation of the wind turbine platform motion of only two degrees of freedom: surge and pitch. The model takes into account different kind of phenomena, which take place when a structure interacts with a fluid as water. In linear hydrodynamics, the hydrodynamic problem can be splitted into three separate and simpler problems: one for radiation, one for diffraction and one for hydrostatics [18]. In addiction to the linear problem will be included into the model two typically non linear terms: the effect of viscosity of the flow and the effect of the mooring lines. In this chapter these phenomena will be described separately so as to highlight their nature and role in the mathematical model. Before linear problems description, the chapter will summarize the main caractetists and properties of sea state because they are essential for the formulation of the external force acting on the structure. Offshore industry typically uses hydrodynamic models based on frequency domain assuming that the system is linear, such that its behavior is linearly related to its displacement, velocity and acceleration. To overcome this limitation for transient analysis, in which nonlinear effects, transient behavior, and irregular sea states are important, it is necessary revert to the direct solution of the equation of motion as a function of time. For these reasons the model used in this work is based on time domain solution. Unfortunately time domain direct solution implies some negative aspects, as for example it is highly time-consuming. In fact the damping term due to the radiation problem, the so called "memory effect", is represented in time domain through a convolution integral, and for its resolution a buffer containing the history of surge and pitch velocities is also needed [8]. In order to be able to realize experimental tests in real time it's necessary to transform memory effect from a convolution integral into a state space formulation across a parametric transformation of the retardation function in frequency domain [24]. In the present study the radiation and diffraction data is provided by the software WAMIT, which uses a panel method based on potential flow theory [22]. Once the mathematical model is presented, the chapter will illustrate how the matlab-simulink code has been structured and will provide the numerical results of the floating turbine dynamics for regular and irregular wave states and some sensitive cases for the energy production of the turbine. Furthermore the numerical results of this study are compared with the simulator FAST.

2.1 Ocean Surface Waves

Water waves causes periodic loads on all sort of man-made structures in the sea. Before the discussion of the different hydrodynamic problems it is worth introducing the principles of wave theory.
Water waves can be generated in many different ways:

- they can be generated by a floating body which is moving.
- they can be generated by the interaction between wind and sea surface.
- They can be generated by astronomical forces as tides or also submarine landslides as tsunamis.

Wind waves are very irregular. They can be seen as a superposition of regular components, each with its own amplitude, length, frequency and direction of propagation. To analyze complicated wave systems, it's necessary to know the properties of simple harmonic components, called *regular waves.*

2.1.1 Regular Waves

Figure 2.1 shows a harmonic wave as seen from two different perspective. The left part shows what one would observe in a snapshot photo. The wave profile is seen as a function of distance x. The right part is a sort of time record of the water level observed in one location. The origin of the coordinate system is at the

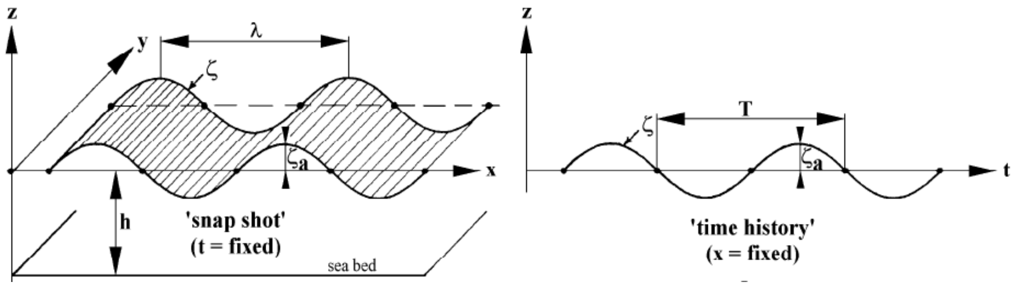


Figure 2.1: Harmonic Wave Definition

still water level with the positive z-axis directed upward and x-axis is positive in the direction of wave propagation. The water depth, *h,* is measured between the sea bed *(z=-h)* and the still water level. If the wave is described as a sine wave, then its amplitude η_a is the distance between still water level to the crest or the trough. The wave height H is measured vertically from wave trough level to the wave crest level. Obviously:

$$H = 2\eta_a \tag{2.1.1}$$

The horizontal distance between two successive wave crest is the wave length λ, and time between two wave crest at the same location is the wave period T. The ratio between the wave height and wave length it is called wave steepness H/T. Since sine and cosine waves are expressed in terms of angular arguments, λ and T are converted in angles using:

$$\text{Wave Number: } k = \frac{2\pi}{\lambda} \text{ [rad/m]} \tag{2.1.2}$$

$$\text{Circular Wave Frequency: } \omega = \frac{2\pi}{T} \text{ [rad/s]} \tag{2.1.3}$$

If the wave moves in the positive x-direction, the wave profile it is expressed as in equation 2.1.4, as a function of both x and t.

$$\eta = \eta_a cos(-\omega t + kx) \tag{2.1.4}$$

Regular wave theory, also called Airy theory, is based on potential flow, which is a linear theory. In order to use it it's necessary to assume that water surface slope is very small, or in terms of a-dimensional quantities, the wave steepness must be very small.
When applied to hydrodynamics of floating bodies on water free surface, the velocity potential, $\varphi = \nabla \mathbf{V}$, where $\mathbf{V}$ is the velocity of the flow, must satisfy four requirements:

1. Continuity condition:
$$\nabla \cdot \mathbf{V} = 0 \tag{2.1.5}$$

2. Sea bed condition: vertical velocity of water particles at sea bed is zero.
$$w = \frac{\partial \varphi_w}{\partial z} = 0 \qquad \text{for: } z = -h \tag{2.1.6}$$

3. Free surface dynamic boundary condition: pressure at the free surface is equal to the atmospheric pressure p_0.
$$p = p_0 \qquad \text{for: } z = \eta \tag{2.1.7}$$

4. Free surface kinematic boundary condition: vertical velocity of water particles at free surface is identical to vertical velocity of the free surface itself:
$$\frac{dz}{dt} = \frac{d\eta}{dt} \qquad \text{for: } z = \eta \tag{2.1.8}$$

The free surface kinematic boundary condition leads to the definition of the relation between the wave period and the wave number shown in equation 2.1.9. This relation it's called *dispersion relationship* and it is good for any water depth [19].

$$\omega^2 = kgtanh(kh) \tag{2.1.9}$$

Figure 2.2 shows that for deep water, where floating wind turbines are installed, and up to a certain dimensionless wave steepness, regular wave theory is a good approximation. In the figure the dimensionless wave steepness is $\frac{H}{gT^2}$ and the dimensionless water depth is represented by $\frac{d}{gT^2}$, both dimensionalized by gravity g and the square of the wave period T [14].

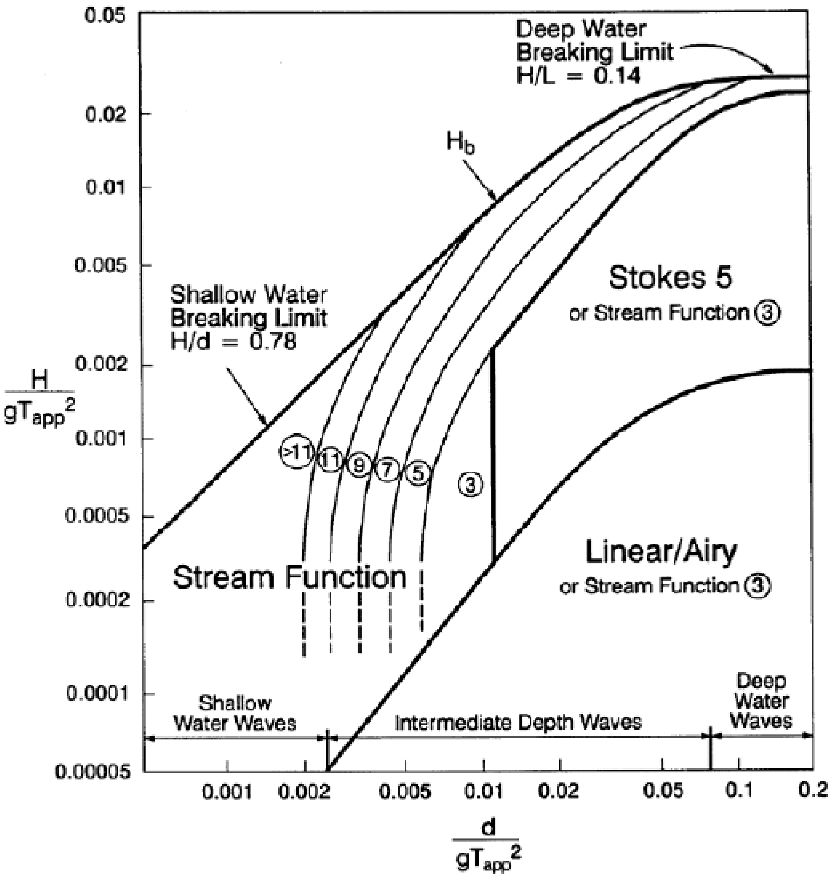


Figure 2.2: Validity of different wave theories

2.1.2 Irregular Waves

Now that the background of regular waves has been discussed, a realistic image of the sea will be presented. Often the sea surface looks very confused ; its image changes continuously with time without repeating itself. To obtain a description

of its randomness it is assumed that the sea can be described as a stationary random process, which generally is valid for limited periods [19].
With this assumption it is possible to represent irregular sea state using a linear superposition of regular wave components. Figure 2.3 illustrates the relationship between the time domain solution $\eta(t)$ of the waves and the frequency domain representation of the waves by a wave spectrum $S(\omega)$.

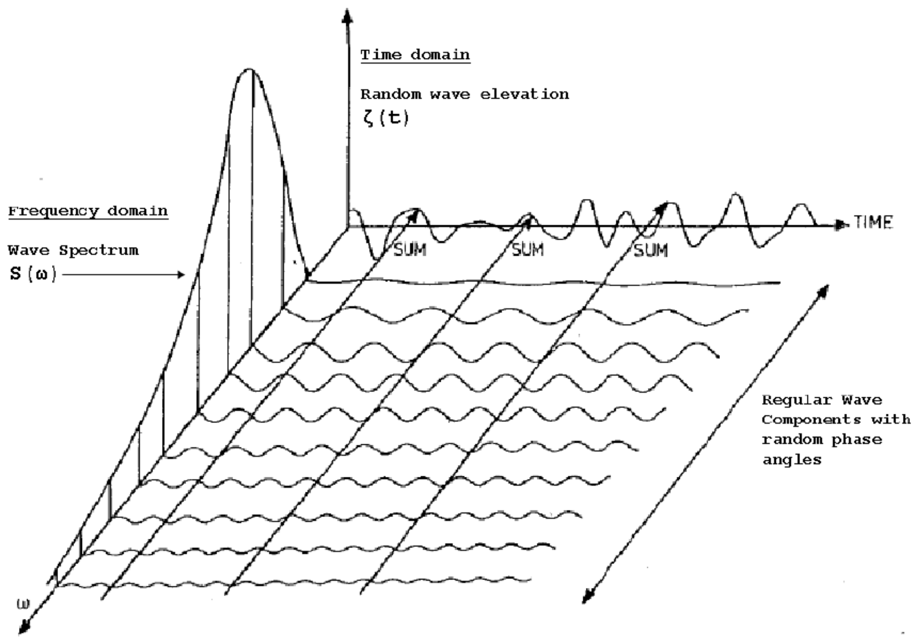


Figure 2.3: Wave elevation of irregular waves in the time domain as a combination of regular waves and typical wave spectrum.

A typical wave spectrum is the JONSWAP spectrum defined from the IEC 61400-3 [6] design standard as follows:

$$PSD_{\eta}(\omega) = \frac{5}{16}\frac{T_p}{2\pi}H_s^2\left(\frac{\omega T_p}{2\pi}\right)^{-5} exp\left[-\frac{5}{4}\left(\frac{\omega T_p}{2\pi}\right)^{-4}\right](1-0.287\log\gamma)\,\beta \tag{2.1.10}$$

$$\beta = \gamma^{exp\left[-\frac{1}{2}\left(\frac{\frac{\omega T_p}{2\pi}-1}{\sigma(\omega)}\right)^2\right]} \tag{2.1.11}$$

where H_s is the significant wave height (i.e., the mean of one-third highest waves), T_p is the peak spectral period, γ is the peak factor and σ is the scaling factor:

$$\sigma(\omega) = \begin{cases} 0.07 & \text{for } \omega \leq \frac{2\pi}{T_P} \\ 0.09 & \text{for } \omega > \frac{2\pi}{T_P} \end{cases} \tag{2.1.12}$$

$$\gamma = \begin{cases} 5 & \text{for } \frac{T_p}{\sqrt{H_s}} \leq 3.6 \\ exp\left(5.75 - 1.15\frac{T_p}{\sqrt{H_s}}\right) & \text{for } 3.6 < \frac{T_p}{\sqrt{H_s}} \leq 5 \\ 1 & \text{for } \frac{T_p}{\sqrt{H_s}} < 5 \end{cases} \tag{2.1.13}$$

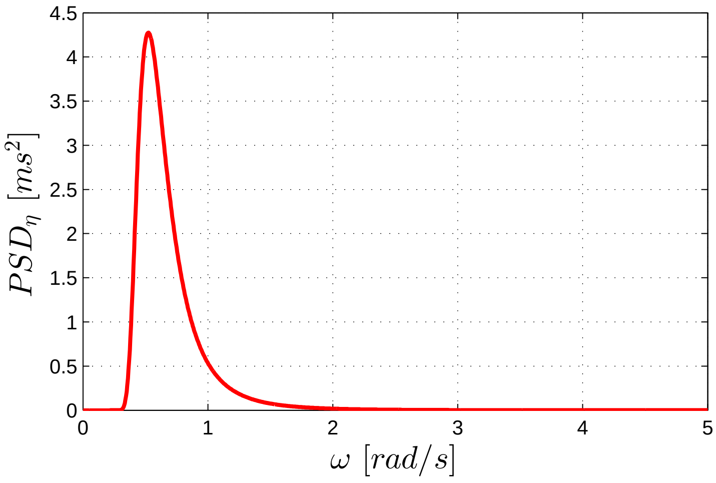


Figure 2.4: Jonswap Spectrum: $H_s = 5\,m$ and $T_P = 12\,s$

Once the Wave Spectrum Density is defined it's possible to retrieve the amplitude of each frequency component η_n of the irregular wave state through the following relation:

$$PSD_\eta(\omega)\triangle\omega = \sum_{n=1}^{N}\frac{|\eta_n|^2}{2} \tag{2.1.14}$$

Basically, for obtain the irregular wave elevation formulation in time domain, multiple regular waves of different frequencies and with random phase angles are summed with respect to their probability spectral density according to the wave spectrum used for the specified site.
The wave elevation of irregular sea, propagating along the positive x axis, can be written as the sum of a large number of regular waves components in the frequency domain:

$$\eta(t) = \sum_{n=1}^{N}\eta_n cos(k_n x - \omega_n t + \phi_n) \tag{2.1.15}$$

in which, for each component, n:

- η_n is the wave amplitude for the n component [m].
- ω_n is the circular frequency component [rad/s].
- $k_n$is the wave number component [rad/m].

- ϕ_n is the random phase angle component [rad].

2.1.3 Wave Kinematics

The kinematics of a water particle is found from the velocity components in the x- and z-directions, obtained from the velocity potential given in equation 2.1.9. The resulting velocity components in their most general form are expressed as:

$$u = \omega \frac{cosh(k(d+z))}{sinh(kd)} \eta_a cos(kx - \omega t) \tag{2.1.16}$$

$$w = \omega \frac{sinh(k(d+z))}{sinh(kd)} \eta_a sin(kx - \omega t) \tag{2.1.17}$$

Equations 2.1.16 and 2.1.17 can be used for any kind of sea bed depth.
In deep water, the water particle velocities can be written in a simpler form:

$$u = \omega e^{kz} \eta_a cos(kx - \omega t) \tag{2.1.18}$$

$$w = \omega e^{kz} \eta_a sin(kx - \omega t) \tag{2.1.19}$$

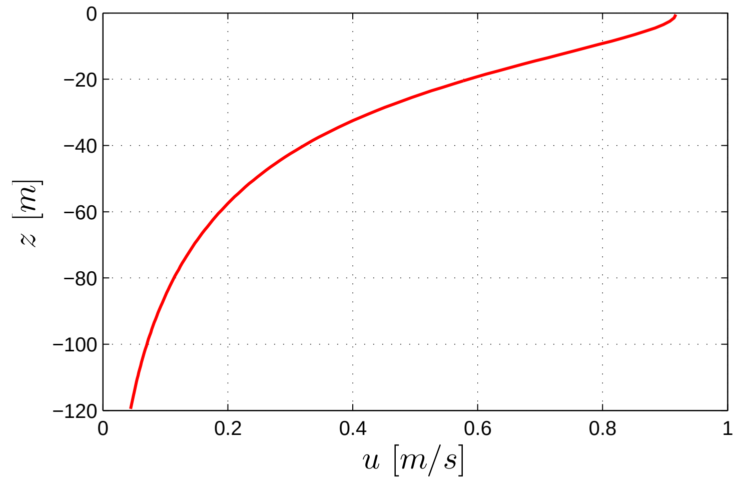


Figure 2.5: Wave Velocity

Wave velocity is the maximum at the sea level and it decreases exponentially with the distance below the surface, as it's possible to see in figure 2.5 [19].

2.2 Time domain model

As previously mentioned, for a transient state analysis is necessary to solve the dynamics in time domain.
The hydrodynamic forces and moments, due to time varying platform motions, can be described using the classic formulation given by Cummins, 1962 [8].

The complete nonlinear time-domain equations of motion of the global system, given by wind turbine, floating platform and mooring lines system are of the general form [14]:

$$[M_{sys}]\ddot{\mathbf{x}} = \sum_i \mathbf{F}_i = \mathbf{F}^{HS} + \mathbf{F}^{Rad} + \mathbf{F}^{Diffr} + \mathbf{F}^{Visc} + \mathbf{F}^{Moor} + \mathbf{F}^{Ad.\ Damp} \tag{2.2.1}$$

where $[M_{sys}]$is the inertia mass matrix, which depends on the set of system's degrees of freedom (DoF) $\mathbf{x} = \begin{bmatrix} x & \theta \end{bmatrix}^T$. In the equations presented in this study, a right Cartesian coordinate system *(X,Y,Z)* fixed to the mean position of the wind turbine is used as shown in figure 2.6. The positive Z axis is vertically upwards through the center of gravity of the platform in its undisplaced position; the origin of the centre of the system (CS) is in the plane of the undisturbed free surface, the mean sea level, and the X axis is parallel to the nominal downwind direction.

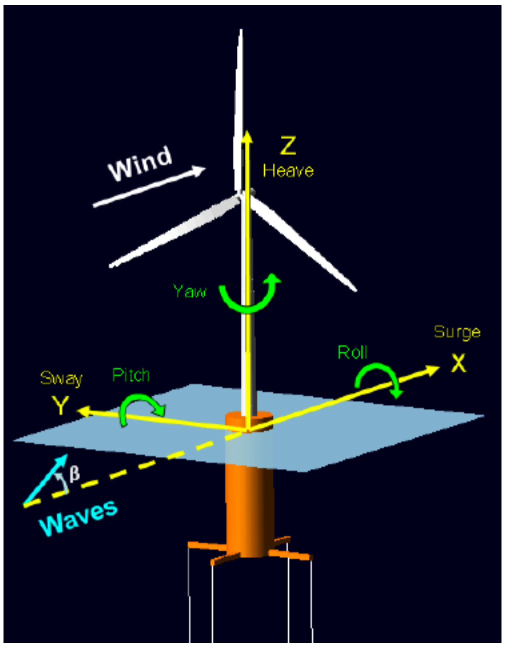


Figure 2.6: Cordinate System

The term on handside of the equation 2.2.1 represents the forces due to diferent contributions: hydrostatics, radiation, diffraction, viscosity of the flow, mooring lines effects and an additional damping force. The last one was introduced because linear radiation damping and nonlinear viscous-drag do not capture all the hydrodynamic damping for the motion of the real platform [15].
In the following paragraphs the terms of equation 2.2.1 will be described separately so as to highlight their role in Cummins model.

2.2.1 Hydrostatics

The first term on the right-hand side of equation 2.2.1 represents the hydrostatic force. It consists in two terms: the first is Archimede's force, which is responsible for the static position of the system, and the second is the restoring force. Restoring force depends on platform position and is responsible of restoring the system, if perturbed, in its static equilibrium position:

$$\mathbf{F}^{HS} = \mathbf{F}^{Arch} - [C^{HS}]\mathbf{x} \tag{2.2.2}$$

where $[C^{HS}]$ is called *hydrostatic restoring matrix.*
Static equilibrium position is found by the resolution of the hydrostatic problem in still water.
In absence of movement, the forces involved are: the different structure parts weight, the preload of the mooring system (i.e the weight of lines in water) and the Archimede's force.

$$\begin{cases} \mathbf{F}^{Arch} = \rho g V_0 \\ \mathbf{P} = \sum_i M_i g \\ \mathbf{F}^{Arch} = \mathbf{P} + \mathbf{F}^{Lines,0} \end{cases} \tag{2.2.3}$$

where ρ is water density, g is gravity acceleration, V_0 is the displaced volume of fluid when the platform is in its undispaced position, M_i are the masses of the system and $\mathbf{F}^{Lines,0}$ is the preload of the mooring lines.
From the system of equations 2.7 it is possible to retrieve V_0. Once V_0 is known, centres of gravity of each part of the floating wind turbine and platform's centre of buoyancy (COB) position Z_i can be defined with respect to the reference system frame (X, Y, Z).

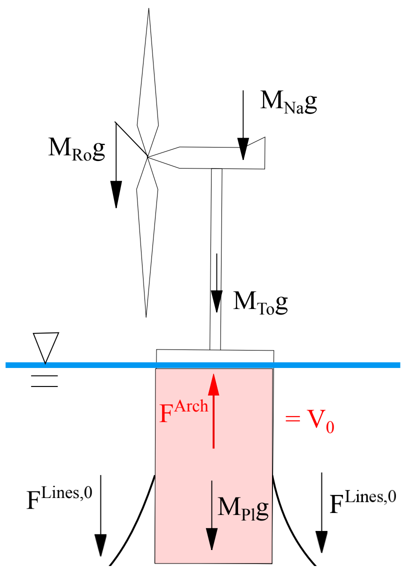


Figure 2.7: Static Equilibrium

Since the platform geometry is a cylinder, COB position and Z_i remain unchanged during the platform movements (surge and pitch).
In order to define equilibrium position it is necessary apply a force and moment equilibrium balance around the system's reference frame, as it is show in figure 2.8. As it is shown it's possible to obtain pitch equilibrium position θ_0, which is essentially zero-valued: $\theta_0 \tilde{=} 0$.

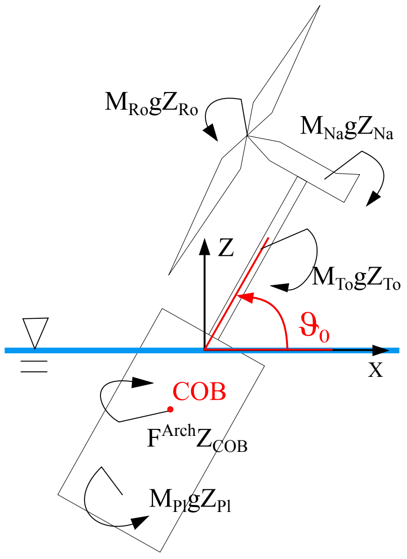


Figure 2.8: Hydrostatic Moment Equilibrium

When equilibrium static position is known, hydrostatic restoring matrix $[C^{HS}]$ can be defined. If body-fixed yz-plane of the submerged portions of the structure is plane of symmetry, hydrostatic restoring matrix has only three components different from zero, which are the diagonal terms of heave, roll and pitch: C_{zz} ,$C_{\alpha\alpha}$, $C_{\theta\theta}$. This terms are defined as follows:

$$\begin{gathered} C_{zz} = \rho g A_0 \\ C_{\alpha\alpha} = \rho g J_{0y} + \mathbf{F}^{Arch} Z_{COB} - g \sum_i M_i Z_i \\ C_{\theta\theta} = \rho g J_{0x} + \mathbf{F}^{Arch} Z_{COB} - g \sum_i M_i Z_i \end{gathered} \tag{2.2.4}$$

where A_0, J_{0x} and J_{0y} are respectively the surface and the inertia moment respect x and y axis of the platform at still water. The numerical model developted for this thesis consideres only surge and pitch DoFs, and the hydrostatic matrix implemented in the numerical code is:

$$[C^{HS}] = \begin{bmatrix} 0 & 0 \\ 0 & C_{\theta\theta} \end{bmatrix} \tag{2.2.5}$$

2.2.2 Mass Matrix

The mass matrix is defined through the derivation of kinetic energy according to Lagrange method. Small displacement hypothesis were made in order to avoid non-linear forms in the equation of motion of the structure. The system consists in four main components: rotor, nacelle, tower and platform. Any of those contributes to the total kinetic energy which is defined as follows:

$$K = \frac{1}{2}M_{Pl}V_{Pl}^2 + \frac{1}{2}M_{To}V_{To}^2 + \frac{1}{2}M_{Na}V_{Na}^2 + \frac{1}{2}M_{Ro}V_{Ro}^2 + \frac{1}{2}J_{Pl}\omega_{Pl}^2 + \frac{1}{2}J_{To}\omega_{To}^2$$

Velocity terms of the whole system are shown in figure 2.9.

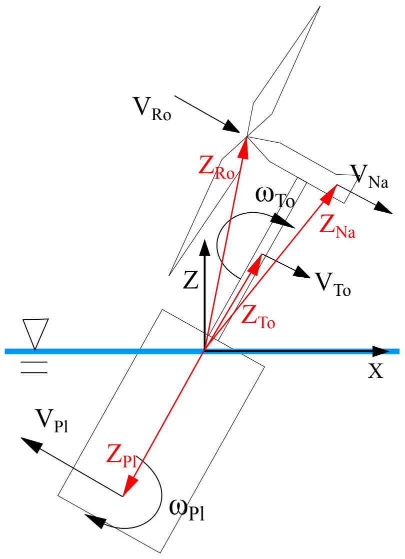


Figure 2.9: Wind Turbine Kinematics

For this case the derivation of kinetic energy according to Lagrange method consist in:

$$\left\{ \begin{array}{c} \frac{d}{dt}\frac{\partial K}{\partial x} \\ \frac{d}{dt}\frac{\partial K}{\partial \theta} \end{array} \right\} = \left\{ \begin{array}{c} m_{xx}\ddot{x} + m_{x\vartheta}\ddot{\theta} \\ m_{\theta x}\ddot{x} + m_{\theta\vartheta}\ddot{\theta} \end{array} \right\} = \left[\begin{array}{cc} m_{xx} & m_{x\vartheta} \\ m_{x\vartheta} & m_{\theta\vartheta} \end{array} \right] \left\{ \begin{array}{c} \ddot{x} \\ \ddot{\theta} \end{array} \right\}$$

For the structure analyzed the terms of the system mass matrix terms are defined as follows:

$$\begin{aligned} m_{xx} &= M_{Pl} + M_{To} + M_{Na} + M_{Ro} \\ m_{x\vartheta} &= M_{Pl}Z_{Pl} + M_{To}Z_{To} + M_{Na}Z_{Na} + M_{Ro}Z_{Ro} \\ m_{\theta x} &= M_{Pl}Z_{Pl} + M_{To}Z_{To} + M_{Na}Z_{Na} + M_{Ro}Z_{Ro} \\ m_{\theta\vartheta} &= M_{Pl}Z_{Pl}^2 + M_{To}Z_{To}^2 + M_{Na}Z_{Na}^2 + M_{Ro}Z_{Ro}^2 + J_{Pl} + J_{To} \end{aligned} \tag{2.2.6}$$

2.2.3 Radiation Problem

The second term of the equation 2.2.1 is the radiation force.
The radiation problem can be treated separately from the diffraction problem since they are both two linear problems. The resulting motion in waves can be seen as a superposition of the motion of the body in still water (radiation) and the motion induced by waves beating on the restrained body (diffraction).

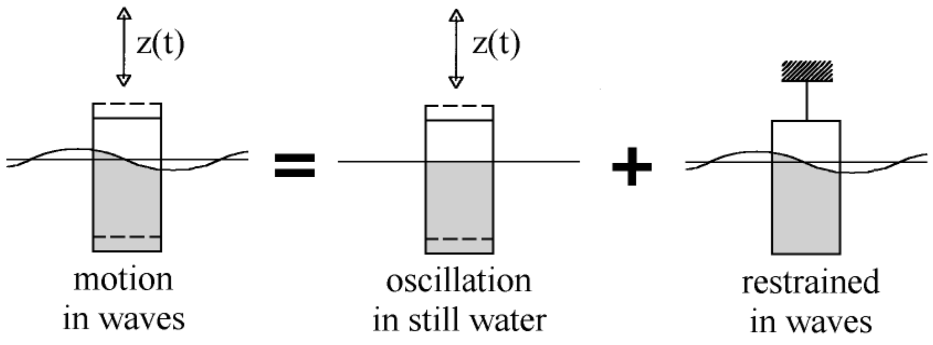


Figure 2.10: Superposition of radiation and diffraction problem.

The radiation force includes contributions from hydrodynamic added mass and damping. In time domain it is expressed as follows:

$$F^{Rad}(t) = -\left[A_{\infty}\right]\ddot{\mathbf{x}} - \int_{-\infty}^{t} \left[K(t-\tau)\right]\dot{\mathbf{x}}(\tau)d\tau \qquad (2.2.7)$$

The first term is the impulsive hydrodynamic added mass component. It is proportional to the acceleration of the platform and represents the integration over the wetted surface of the outgoing wave pressure induced by a unit of acceleration. Like the system mass matrix $[M_{sys}]$, the impulsive hydrodynamic mass $[A_{\infty}]$ is symmetric. Unlike the $[M_{sys}]$ and depending on the shape of the support platform, $[A_{\infty}]$ can contain off-diagonal components that couple modes of motion that cannot be coupled through body inertia.
In the radiation problem, the free surface brings about the existence of memory effects, denoting that the wave radiation load depends on the history of motion of the support platform. The memory effect is represented by the final term of equation (2.8), that is a convolution integral representing the load contribution from wave radiation damping. In the expression τ is a dummy variable with the same units as the simulation time, t, and $[K(t)]$ is wave-radiation-retardation kernel matrix, that is the velocity-impulse response function of the radiation problem. It is called memory effect because an impulse in structure velocity induces a pressure field within the fluid domain that persists for as long as the wave radiates away.

The memory effect time $t = \bar{t}$:

$$\mu(\bar{t}) = \int_{-\infty}^{\bar{t}} [K(\bar{t} - \tau)]\, \dot{\mathbf{x}}(\tau) d\tau \tag{2.2.8}$$

is seen as a the response to the superposition of a succession of individual velocity impulses. Each response is calculated with an appropriate time delay from the instance of the corresponding impulse. The impulses must be considered as occurring closer and closer together until finally it is possible to integrate the responses rather than summing them.
In practice, the convolution integral extremes must be a finite number keeping in mind that in order to achieve the best result the integration interval must be as big as possible.
In this study the integration interval $\triangle t$ were chosen of 60 seconds:

$$\mu(\bar{t}) = \int_{t-\triangle t}^{t} [K(\bar{t} - \tau)]\, \dot{\mathbf{x}}(\tau) d\tau \tag{2.2.9}$$

Both $[A]$ and $[K(t)]$ are indirectly retrieved from the data provided by WAMIT. WAMIT's outputs for radiation problem are the added mass matrix $[A(\omega)]$ and the added damping matrix $[B(\omega)]$. These matrices terms are reported in figures 2.11 and 2.12.
The hydrodynamic added mass component of equation (3.6) and the wave-radiation-retardation kernel matrix $[K(t)]$ are defined as follows and the results are shown in equation 2.2.10 and 2.13:

$$[A_\infty] = \lim_{\omega \to \infty} [A(\omega)] = \begin{bmatrix} 7.98e6 & -4.86e8 \\ -4.86e8 & 3.8e10 \end{bmatrix} \tag{2.2.10}$$

$$[K(t)] = \frac{2}{\pi} \int_{-\infty}^{+\infty} [B(\omega)]\, cos(\omega t) d\omega \tag{2.2.11}$$

2.2.4 Diffraction Problem

The third term of equation (3.1) is the diffraction force. It represents the excitation load from incident waves on the structure. Since the flow is considered incompressible and irrotational (potential flow), the diffraction force accounts only for the effect of the pressure on wetted surface of the platform:

$$\mathbf{F}^{Diff} = \iint_{S_{wet}} p dS \tag{2.2.12}$$

The excitation force from the incident waves is closely related to the wave elevation.
In case of regular waves, that are simple sine waves, the diffraction force depends only on the wave amplitude, wave frequency and the geometry of the body.

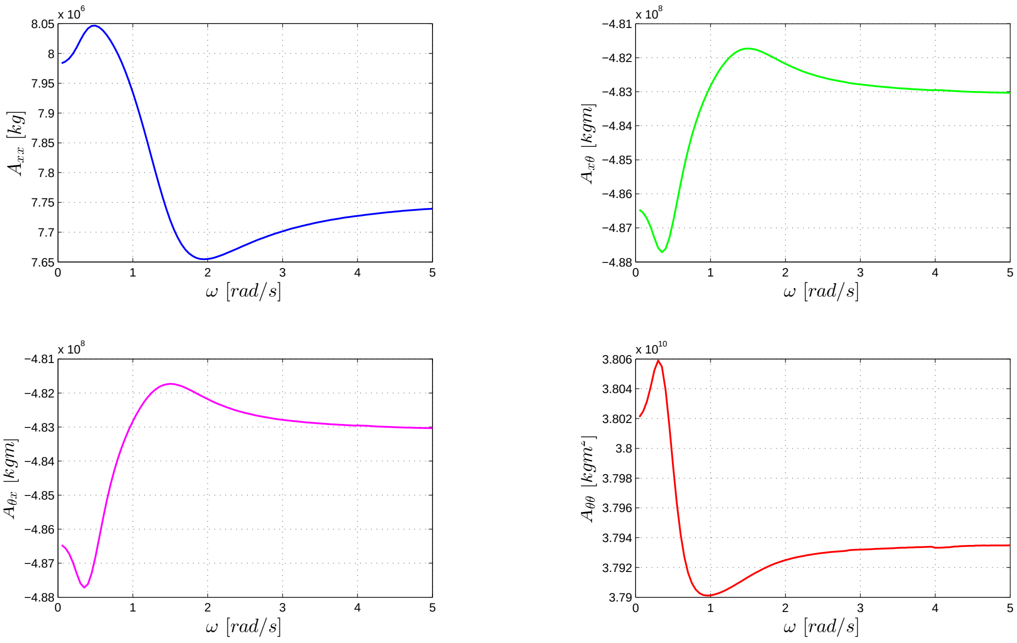


Figure 2.11: Added Mass Matrix Terms

In case of irregular waves, the diffraction force is obtained directly from the superposition principle. This assumption implies that the magnitude of the wave excitation force from multiple superimposed waves is the same as the sum of the wave excitation force produced by each individual wave component.
Equation (3.11) can be expressed in the following form:

$$\mathbf{F}^{Diff}(t) = \int_{-\infty}^{+\infty} \chi(t-\tau)\eta(\tau)d\tau \tag{2.2.13}$$

In equation 2.2.13 τ is a dummy variable with the same units as the simulation time, t, and $\chi(t)$ is the time- and direction- dependent incident-wave-excitation force on the support platform normalized per unit wave amplitude:

$$\chi(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \overline{\mathbf{X}}(\omega,\beta)e^{j\omega t}d\omega \tag{2.2.14}$$

where $\overline{\mathbf{X}}(\omega,\beta)$ is a complex valued array that represents the force on the body normalize per unit wave amplitude; the imaginary component makes the force out of phase with the wave elevation.
$\overline{\mathbf{X}}(\omega,\beta)$ depends on the geometry of the structure, the frequency of the wave component ω and the propagation direction of the wave β. This study takes into account only waves propagating along the X axis of the referent system frame: $\beta = 0$.

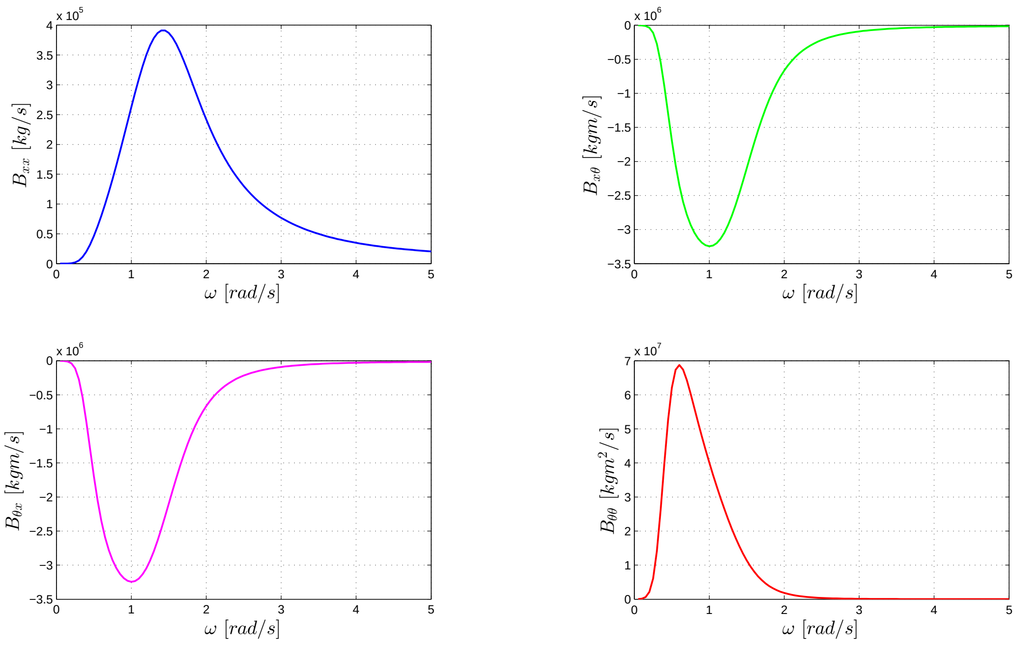


Figure 2.12: Added Damping Matrix terms

$\overline{\mathbf{X}}(\omega, \beta)$ is also provided by WAMIT. The data for surge and pitch is illustrated in figure 2.14.

Radiation, diffraction and hydrostatics are all linear contributions tha are collected into the Cummins's equation. Together form what is called *Cummins's equation* (equation *2.2.15)*, which is the usual representation of a hydrodynamic problem in time domain:

$$([M_{Sys}] + [A_\infty])\ddot{\mathbf{x}}(t) + \int_{t-\triangle t}^{t} [K(\bar{t} - \tau)]\,\dot{\mathbf{x}}(\tau)d\tau + [C^{HS}]\mathbf{x}(t) = \mathbf{F}^{Diff}(t) \quad (2.2.15)$$

2.2.5 Viscous Problem

The viscous term $\mathbf{F}^{Visc}$ of equation (3.1) is due to the water viscous forces described by *Morrison Representation* of slender cylinders hydrodyamics. *Morrison Representation* is an alternative way to formulate the hydrodynamical problem and can substitute Cummins's formulation for surge, sway, roll and pitch. It can be used, in conjunction with strip theory, to compute the linear wave load and nonlinear viscous drag loads. It is typically used for slender offshore substructures.

The total force acting on the structure, subdivided in strips, is found integrating over the length of the cylinder the loads acting on each strip:

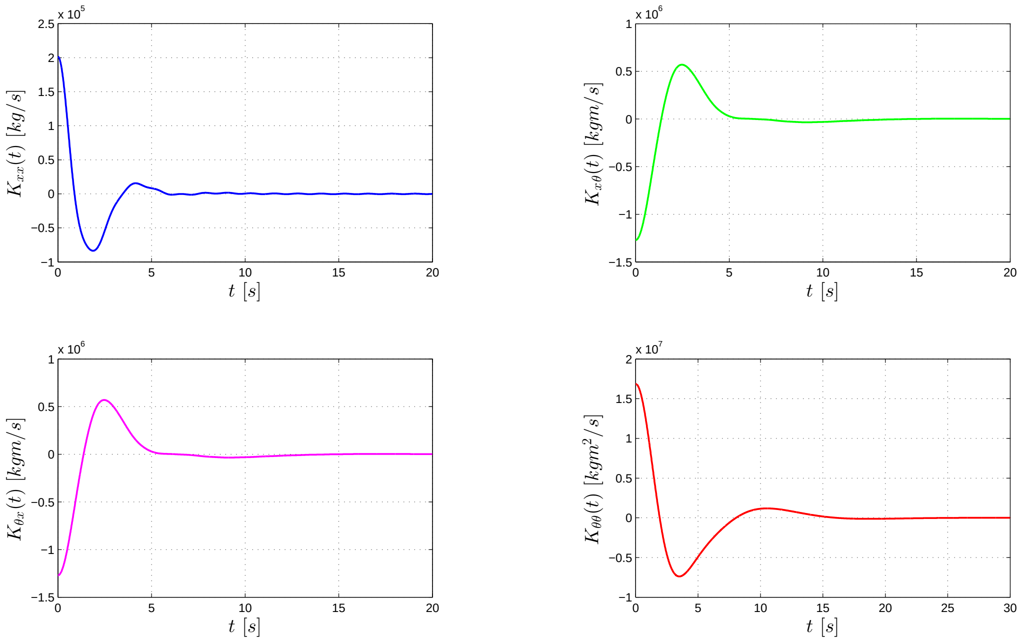


Figure 2.13: Wave Radiation Retardation Kernel Matrix Terms

$$\mathbf{F}^{Morrison} = \int_{height} d\mathbf{F}^{Morrison} \tag{2.2.16}$$

$d\mathbf{F}^{\text{Morrison}}$ consists in a term due to the added mass contribution and in a term due to the nonlinear viscous drag. These coefficients depend, among other factors, on Reynolds and Keulegan-Carpender number, which is defined as follows:

$$K = \frac{VT}{D}$$

where D is the diameter of the cylinder, V is the amplitude of the velocity normal to the cylinder and T is the wave period. Keulegan-Carpenter number is a dimensionless quantity describing the relative importance of the drag forces over inertia forces for bluff objects in an oscillatory fluid flow. Or similarly, for objects that oscillate in a fluid at rest. For small Keulegan-Carpenter number inertia dominates, while for large numbers the (turbulence) drag forces are important. For the OC3-Hywind spar-buoy, the Keulegan-Carpenter and oscillatory Reynolds numbers increse with severity in the wave conditions and decrease with depth along the spar. Flow separation occurs when Keulegan-Carpenter number exceeds 2. For values lower than 2, potential flow theory applies. Consequently, potential flow model can be properly adopted for all along the spar except for upper portion of the platform, where separationwill occur.
In the present study the viscous term of Morrison's equation it is added to Cummins equation. The reason of that is not only because it permits to consider real

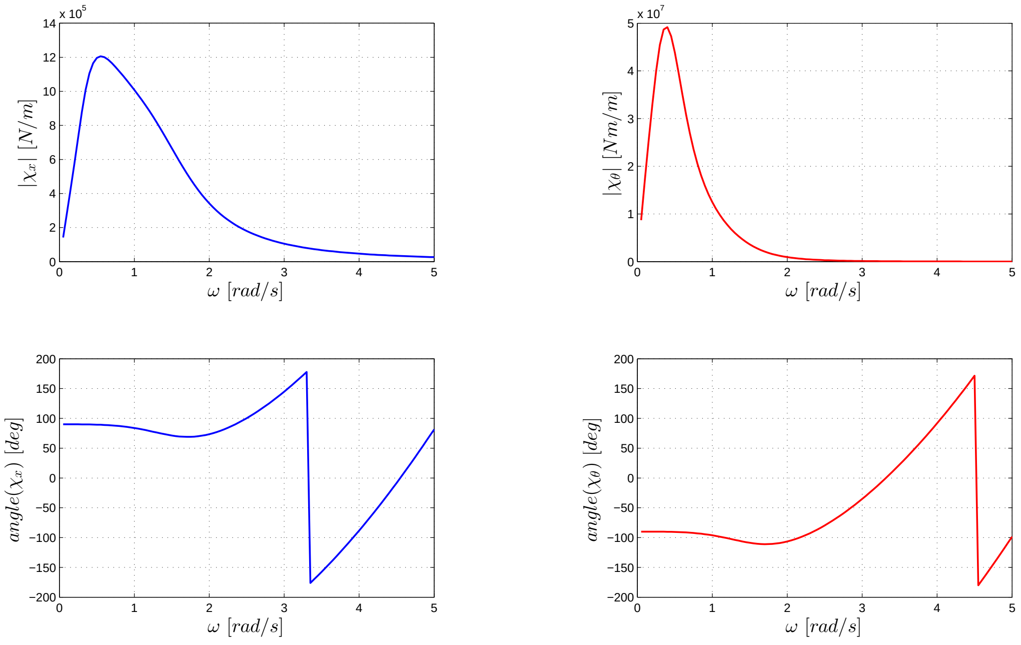


Figure 2.14: Wave Force per unit Wave Amplitude

effects of the flow, neglected in Cummins equations because of potential flow hypotesys, but also beacuse is an important source of hydrodynamic damping when the motions of the structure are very small.

The viscous term of Morrison 's equation is here reported:

$$d\mathbf{F}^{Visc} = \left\{ \begin{array}{c} \frac{1}{2}\rho C_D D \left\| V_{Rel}(Z,t) \right\| V_{Rel}(Z,t) dZ \\ \frac{1}{2}\rho C_D D \left\| V_{Rel}(Z,t) \right\| V_{Rel}(Z,t) Z dZ \end{array} \right\} \tag{2.2.17}$$

$$V_{Rel}(Z,t) = U(Z,t) - V_{Pl}(Z,t) \tag{2.2.18}$$

where D is the diameter of the cylinder, Z is the coordinate of the strip's geometrical centre, dZ is the length of the differential strip of the structure, C_D is the viscous drag coefficient, which is function of Re and roughness of the surface, ρ is water density, $U(Z,t)$ is the undisturbed fluid-particle velocity and $V_{Pl}(Z,t)$ is the platform velocity:

$$V_{Pl}(Z,t) = \dot{x} + Z\dot{\theta} \tag{2.2.19}$$

where $\dot{x}$ and $\dot{\theta}$ are respectively surge and pitch velocities.

Reynolds number is at least on the order of 10^7 and the surface is considered as smooth. For this reasons C_D has been chosen 0.6.

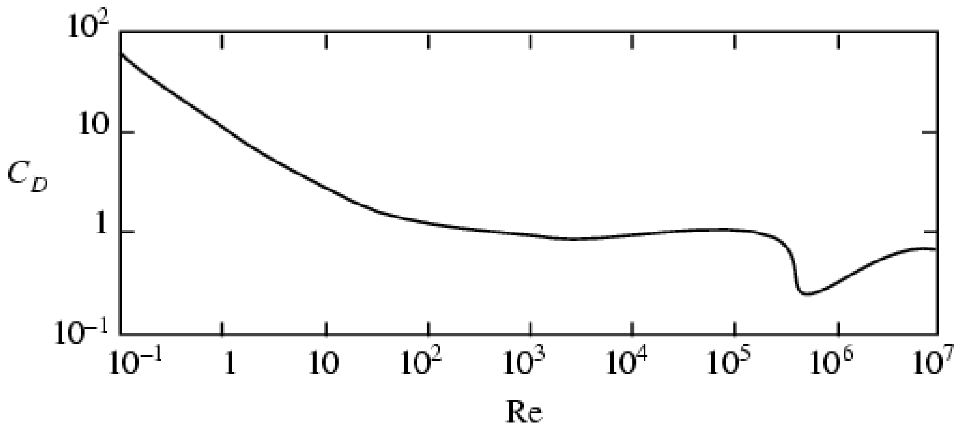


Figure 2.15: Drag Coefficient for Cylinders

2.2.6 Mooring Lines Problem

Mooring systems are used as a means of station-keeping and holding a floating platform against wind, waves, and current and also in some support platform designs, such as in a TLP, they are responsible for stability.
A mooring system is made up of a number of cables that are attached to the floating support platform at fairlead connections, with the opposite ends anchored to the seabed. Restraining forces at the fairleads are established through tension in the mooring lines. This tension depends on the buoyancy of the support platform, the cable weight in water, the elasticity in the cable, seabed friction of each line and the geometrical layout of the mooring system. As the fairleads move with the support platform in response to unsteady environmental loading, the restraining forces at the fairleads change with the changing cable tension.

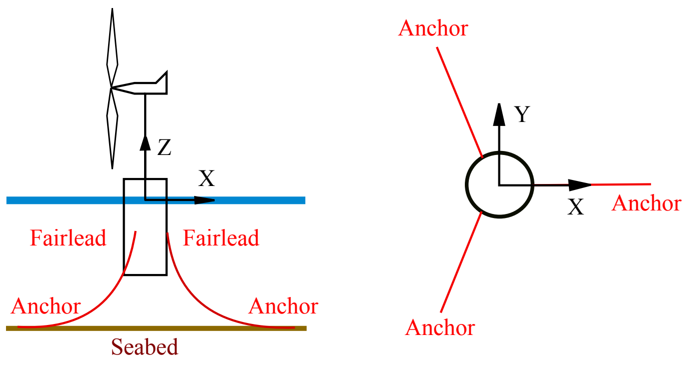


Figure 2.16: Mooring Lines system geometry

In order to simplify the analysis of the mooring system within the OC3 project, however, three simplifications are made in order to deal with a "quasi-static"

Number of Mooring Lines	3
Angle Between Adjacent Lines	120 deg
Depth to Anchors Below SWL (Water Depth)	320 m
Depth to Fairleads Below SWL	70 m
Radius to Anchors from Platform Centerline	853.87 m
Radius to Fairleads from Platform Centerline	5.2 m
Unstretched Mooring Line Length	902.2 m
Mooring Line Diameter	0.09 m
Equivalent Mooring Line Mass Density	77.7066 kg/m
Equivalent Mooring Line Weight in Water	698.094 N/m
Equivalent Mooring Line Extensional Stiffness	384243000 N

Table 2.1: Mooring System Properties

problem. Mpre specifically each of the multisegment lines is replaced with an equivalent homogenous line, with properties derived as the weighted-average values of the mass, weight, and stiffness. Also all mooring system mass and damping, including the hydrodynamic drag and line-to-seabed drag, is neglected.
Table 2.1 and figure summarizes OC3 HyWind mooring lines properties.

In general the mooring system dynamics is non-linear in nature. The mooring dynamics also often includes nonlinear hysteresis effects, where energy is dissipated as the lines oscillate with the support platform around its mean position.
Mooring lines Module implemented in Fast solves a non-linear system of two equations in two unknowns for each line. The unknowns are the horizontal and vertical components of the actual tension in the mooring line at the fairlead and the variables in input are the fairleads position, also a function of the platform dofs displacements. Once the tension at fairleads is known, it's possible to retrieve their effects in terms of restoring forces and momentums on the platform.

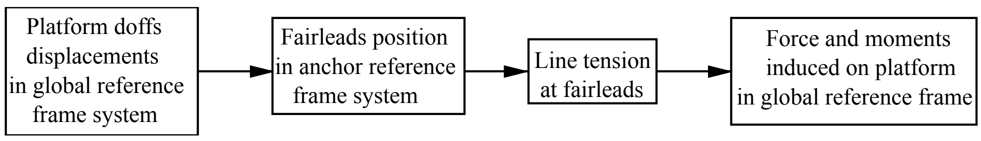

Platform doffs displacements in global reference frame system
Fairleads position in anchor reference frame system
Line tension at fairleads
Force and moments induced on platform in global reference frame

Figure 2.17: Mooring Lines Forces Calculation

Figure 2.18 shows the nonlinear relationships for the surge forces and pitching moments associated with surge and pitch displacements.
As it can be seen from figure, for small displacements around system's undispalced position, the total load on the support platform from the contribution of all mooring lines $\mathbf{F}^{Moor}$ (from equation (3.1)) can be written in a linear form:

$$\mathbf{F}^{Moor} = \mathbf{F}^{Lines,0} - \left[C^{Lines}\right]\mathbf{x} \tag{2.2.20}$$

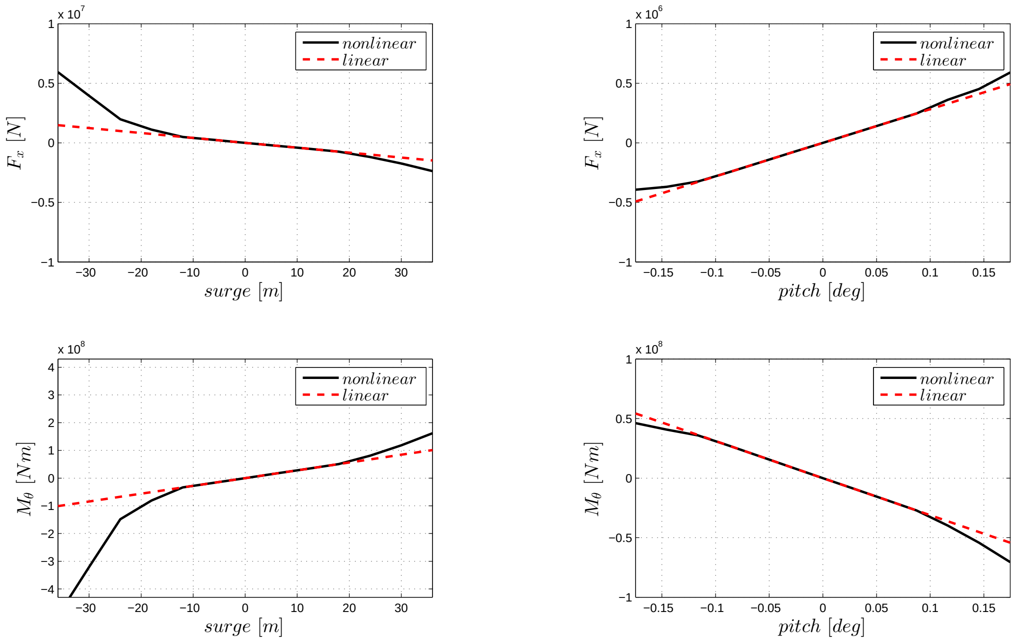


Figure 2.18: Non-linear relationship for the surge forces and pitching moments associated with surge and pitch displacements.

where $\mathbf{F}^{Lines,0}$ is the mooring system load acting on the structure in its undisplaced position (x_0, θ_0). For catenary mooring lines it represents the pretension in Z direction at the fairleads caused from the weight of those cables not layng on the seabed. $\left[C^{Lines}\right]$ is the mooring lines matrix linearized in the undisplaced poistion (x_0, θ_0). It is the combined result of mooring lines elastic stiffness, the geometric stiffness due to the relative position of each fairlead to the anchor, and the weight of cables in water. It depend on the layout of the system and for OC3 Hywind platform has the following values for surge and pitch:

$$\left[C^{Lines}\right] = \begin{bmatrix} 41180\,N/m & -2821000\,N/rad \\ -2816000\,Nm/m & 311100000\,Nm/rad \end{bmatrix} \tag{2.2.21}$$

2.2.7 Additional Damping Problem

The linear radiation damping (from potential-flow theory, which is small) and the nonlinear viscous-drag (from the relative form of Morison's formulation), when summed, do not capture all of the hydrodynamic damping for the motions of the real Hywind platform. Therefore, Statoil recommended that the hydrodynamics models for the OC3-Hywind system, as described above, be augmented with additional linear damping. To determine the amount of additional damping required in the OC3-Hywind system, still-water free-decay responses tests for all six rigid-body modes-of-motion of the a scale model was made in a by Statoil in a water tank. These responses were used as target responses for the OC3-Hywind system. This damping applies to the platform motions directly and not to the relative motion between the platform and wave particle velocities. Additional damping force is added in equation 2.2.1 in linear form as in equation 2.2.22.

$$F^{Additional\,Damping} = -\left[B^{Ad\,Damp}\right]\dot{\mathbf{x}} \tag{2.2.22}$$

For surge and pitch $\left[B^{Ad\,Damp}\right]$ is made by the following values:

$$\left[B^{Ad\,Damp}\right] = \begin{bmatrix} 100000\,Ns/m & 0 \\ 0 & 0 \end{bmatrix} \tag{2.2.23}$$

2.3 State Space Model Representation fo Memory Effect

Time domain model of marine structures based on frequency domain data usually rely on the Cummins equation. This type of model is a vector integro-differential equation which involves a convolution term extensively discussed above. This convolution term implies several disadvantages:

- it is not convenient for analysis and design of motion control.
- it complitcates the linearization of the structure model system.
- it is time consuming for real time applications.

Because convolution is a linear operation, different approaches can be followed to obtain an approximately equivalent linear system in the form of either transfer function or state space models.
This process involves the use of system identification. Several methods have been proposed in literature to perform the identification. In this study a frequency domain based identification model has been adopted. The aim is to build a parametric transfer function $[\tilde{K}(\omega)]$ of the retardation function in frequency domain $[K(\omega)]$, defined in equation 2.3.1, and then to determine the equivalent state-space model representation in order to retrieve the memory effect μ [24]..

$$[K(\omega)] = [B(\omega)] + j\omega([A(\omega)] - [A_\infty]) \tag{2.3.1}$$

$$[K(\omega)] \longrightarrow [\tilde{K}(\omega)] \longrightarrow \begin{cases} \dot{\mathbf{y}} = [A_r]\mathbf{y} + [B_r]\dot{\mathbf{x}} \\ \\ \mu = [C_r]\mathbf{y} \end{cases} \tag{2.3.2}$$

These operations can be performed assuming that the system is causal and time invariant, and also due to the Markovian property of state space models, which guaranties that any future state of the system depends only on the present value of the system state, so that no past information are required to be stored as for the convolution integral.

2.3.1 Properties and Quality of the Parametric Model

The parametric model of the retardation function must be constructed in the following way:

$$\tilde{K}_{ij}(s,\theta) = \frac{P(s,\theta)}{Q(s,\theta)} = \frac{p_m s^m + p_{m-1}s^{m-1} + ... + p_0}{s^n + q_{n-1}s^{n-1} + ... + q_0} \tag{2.3.3}$$

Where ϑ is the vector with the different parameters of numerator $P(s,\theta)$ and denominator $Q(s,\theta)$, i and j are the entries of the retardation matrix and $s = j\omega$. The parametric function of the retardation function $[\tilde{K}(\omega)]$ have to fullfill certatain proprieties apriori known.

1. The low-frequency asymptotic value has to be 0:

$$\lim_{s\to 0} \tilde{K}_{ij}(s) \neq 0 \tag{2.3.4}$$

A structure cannot radiate waves at zero-frequency.

2. The high-frequency limit of the retardation function has to be 0:

$$\lim_{s\to \infty} \tilde{K}_{ij}(s) = 0 \tag{2.3.5}$$

To guarantee this propriety, the transfer function $\tilde{K}_{ij}(s)$ has to be strictly proper, that is $deg\,\{Q(s,\theta)\} > deg\,\{P(s,\theta)\}$.

3. Initial time value different from zero.

$$\lim_{t\to 0} \tilde{K}_{ij}(t) \neq 0 \tag{2.3.6}$$

which implies in Laplace domain that:

$$\lim_{s\to \infty} s\tilde{K}_{ij}(s) = \lim_{\omega\to \infty} s\frac{P(s)}{Q(s)} = \frac{p_m s^{m+1}}{s^n} \tag{2.3.7}$$

From the previous equation it is clear that in order to force the limit to be finite and different from 0, the relative order between the denominator and numerator must be one ($n = m+1$). Combining this requirement with property 1, it is easy to conclude that the minimum order of the parametric function $\tilde{K}_{ij}$ is second order.

4. The response of a stable system to an impulse should tend to zero when time tends to infinite. This propriety establishes the bounded-input bounded-output stability of the radiation system and it is given by the limit:

$$\lim_{t\to \infty} \tilde{K}_{ij}(t) = 0 \tag{2.3.8}$$

Therefore the poles of the transfer function $K(\omega)$, poles of the denominator $Q(s)$, must have a negative real part.

5. The retardation matrix must be positive define in the frequency domain:

$$\mathfrak{Re}\left\{[\tilde{K}(\omega)]\right\} > 0 \tag{2.3.9}$$

In order to evaluate the proper fitting, the R^2 value is computed using:

$$R^2 = 1 - \frac{\sum\limits_{l}\left(K_{ij} - \tilde{K}_{ij}\right)^2}{\sum\limits_{l}\left(K_{ij} - \bar{K}_{ij}\right)^2} \qquad 0 \leq R^2 < 1 \tag{2.3.10}$$

were K_{ij} represents the reference retardation function, $\tilde{K}_{ij}$ the parametric model and $\bar{K}_{ij}$ is the mean value of the reference retardation function. The summations are performed across all frequencies (for frequency response) or time (for impulse response). This is a measure of the amount of variability of the parametric function $\tilde{K}_{ij}$ that is captured by the model. The closer to 1, the better is the quality of the fit.

2.3.2 FDI Toolbox

In this study it has been used the Frequency Domain Identification Toolbox developed by Perez & Fossen [24]. This method is based the optimization of the Least Squares Method:

$$\theta = arg\left\{\underset{\theta}{min}\left[\sum_l w_l\left(K_{ij}(s,\theta) - \widetilde{K}_{ij}(s,\theta)\right)^2\right]\right\} \tag{2.3.11}$$

where ϑ is the vector with the different parameters of numerator $P(s,\theta)$ and denominator $Q(s,\theta)$, w_l is the weight factor, $K_{ij}(s,\theta)$ and $\tilde{K}_{ij}(s,\theta)$ are the retardation function and its parametric model.
In particular the method solves an iterative linear LS problem, using as weight factors the denominator values calculated at the previous iteraction.
The toolbox uses the following algorithm:

1. Set the appropriate range of frequencies according to the user defined weight factors;

2. Scale the data:
$$K_{ij}^{'} = \alpha K_{ij}; \qquad \alpha = \frac{1}{max|K_{ij}|} \tag{2.3.12}$$

3. Select the order of the approximation $n = deg\left\{Q(s)\right\}$. The minimum order approximation n=*2* is the first attempt, as previously mentioned.

4. Estimate the parameters θ using the iterative LS method, according to:
$$\theta = arg\left\{\underset{\theta}{min}\sum_l\left|\frac{K_{ij}^{'}(s,\theta)}{s} - \widetilde{K}_{ij}^{'}(s,\theta)\right|\right\} \tag{2.3.13}$$

5. Check the stability by computing the roots of $Q(s)$ and change the real part of these roots with positive real part to a negative real part.

6. Construct the desired transfer function by dividing by scaling and incorporating the s the numerator.
$$\widetilde{K}_{ij} = \frac{s\widetilde{K}_{ij}}{\alpha} \tag{2.3.14}$$

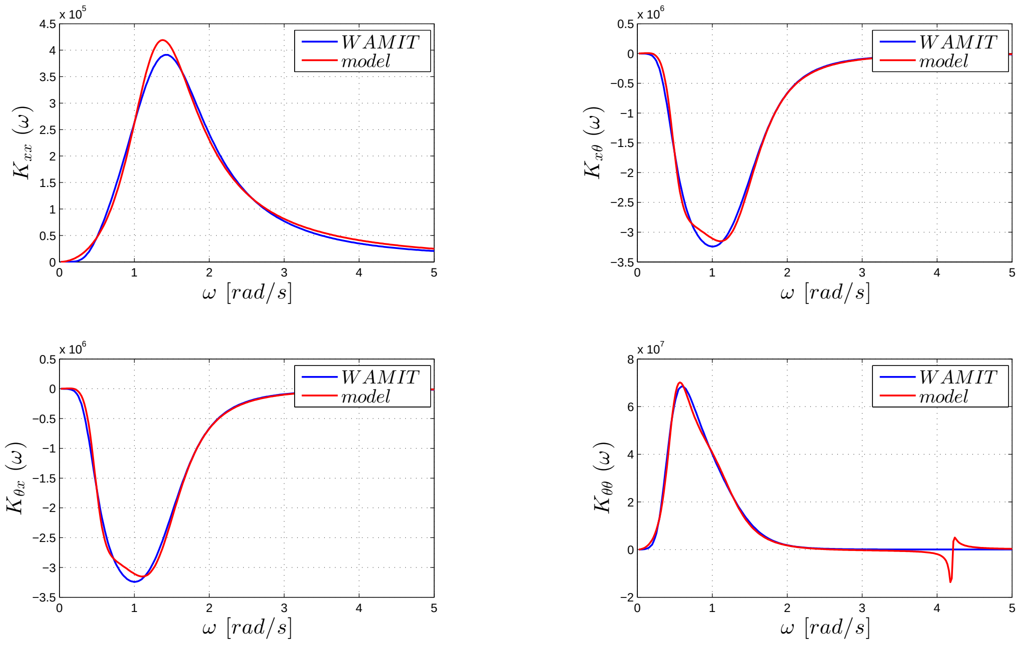


Figure 2.19: Comparison of the retardation function $K(\omega)$ and the parametric form obtained $\widetilde{K}(\omega)$.

7. Estimate the added mass matrix $[\widetilde{A}(\omega)]$ and added damping matrix $[\widetilde{B}(\omega)]$ based on the identified parametric approximation via equations 2.3.15, and compare with the added mass and damping matrix given by WAMIT. Added mass and damping terms are obtained as follows:

$$\begin{cases} \widetilde{A}_{ij}(\omega) = \mathfrak{Im}\left\{\tilde{K}_{ij}\right\} + A_{\infty,ij} \\ \\ \widetilde{B}_{ij}(\omega) = \mathfrak{Re}\left\{\tilde{K}_{ij}\right\} \end{cases} \tag{2.3.15}$$

   The quality of the fit is assessed using the parameter R^2 for the added mass and added damping terms. If the fit it is not satisfactory, the code increases the order of the approximation.

8. Check that the retardation matrix must be positive define in the frequency domain: $\mathfrak{Re}\left\{[\widetilde{K}(\omega)]\right\} > 0$.

9. From the parametric transfer function it is easy to obtain the state space model, so the state space matrices $[A_r]_{ij}$, $[B_r]_{ij}$ and $[C_r]_{ij}$.

In appendix A a matlab code able to compute the FDI toolbox steps is presented. With this code parametric retardation function of surge-surge, surge-pitch, pitch-surge and pitch-pitch dofs with a quality value R^2 of 0.99 imposed have been obtained. Results are show in figure 2.19.

2.3.3 Matrix Assembly

Using the method described above, a set of state space systems are obtained, ones for each significant entry ij of the retardation matrix $[K(\omega)]$. In order to obtain the complete state-space system, each matrices $[A_r]_{ij}$, $[B_r]_{ij}$, $[C_r]_{ij}$ and the state vectors $\mathbf{y}_{ij}$ have to be assembled into a global state space system according to equations 2.3.16, 2.3.17, 2.3.18 and 2.3.19:

$$[A_r] = \begin{bmatrix} [A_r]_{11} & & & & & & & & \\ & \ddots & & & & & & & \\ & & [A_r]_{1m} & & & & & & \\ & & & [A_r]_{21} & & & & & \\ & & & & [A_r]_{22} & & & & \\ & & & & & \ddots & & & \\ & & & & & & [A_r]_{m1} & & \\ & & & & & & & \ddots & \\ & & & & & & & & [A_r]_{mm} \end{bmatrix} \tag{2.3.16}$$

$$[B_r] = \begin{bmatrix} [B_r]_{11} & & \\ \vdots & & \\ [B_r]_{1m} & & \\ & [B_r]_{21} & \\ & \vdots & \\ & [B_r]_{2m} & \\ & & [B_r]_{m1} \\ & & \vdots \\ & & [B_r]_{mm} \end{bmatrix} \tag{2.3.17}$$

$$[C_r] = \begin{bmatrix} [C_r]_{11} & & & [C_r]_{21} & & & [C_r]_{m1} & & \\ & \ddots & & & \ddots & & & \ddots & \\ & & [C_r]_{1m} & & & [C_r]_{22} & & & [C_r]_{mm} \end{bmatrix} \tag{2.3.18}$$

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_{11} \\ \vdots \\ \mathbf{y}_{1m} \\ \mathbf{y}_{21} \\ \vdots \\ \mathbf{y}_{22} \\ \mathbf{y}_{m1} \\ \vdots \\ \mathbf{y}_{mm} \end{bmatrix} \tag{2.3.19}$$

In appendix A also a code for assembling each entry ij is presented.

2.3.4 Comparison between convolution Integral and State Space Model Results

At this stage of the study a comparison between convolution integral (CONV) and state space model (SS) results was performed. The convolution integral needs at least 10 seconds to be computed in order to reach a good accuracy. So it has been taken a 60 seconds of impulse response duration with a time step of 0.01 seconds with the aim of avoinding accuracy problems.
Before computing the state space model into the complete numerical model of platform hydrodynamics two simpler tests were performed:

1. The response due to a cosine wave input with an amplitude value of 1 and circular frequency value of 2.

   For this test the surge-surge retardation function was taken as reference. Results are shown in figure 2.20.

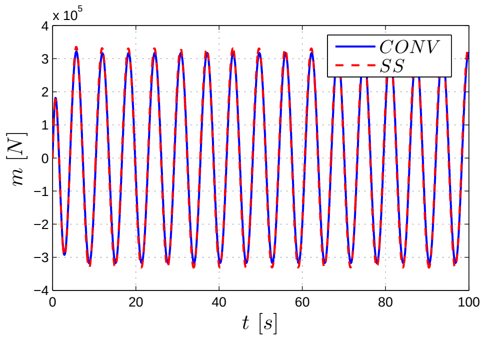


Figure 2.20: State space model and convolution model comparison test comparison with single sine wave input

2. A free decay test on the surge and pitch platform dofs in still water, neglecting the viscous effect:

$$([M_{Sys}] + [A_\infty])\ddot{\mathbf{x}}(t) + \mu + [C^{HS} + C^{Moor}]\mathbf{x}(t) = 0$$

A comparison to free decays from FAST was also performed.

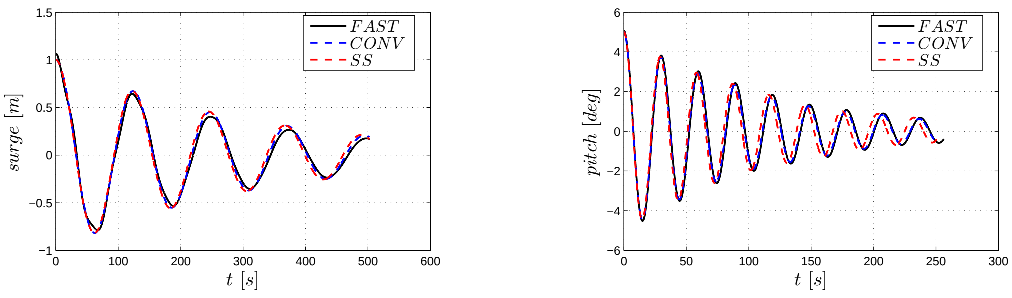


Figure 2.21: Free decay comparison between SS model, Convolution model and FAST

Convolution Model	15 min
State Space Model	< 1 s
FAST	5 min

Table 2.2: Free Deacy Test Time Resolution Duration

Both tests demonstrate that state space model can substitute convolution model with an acceptable approximation. Figure 2.20 shows that the convolution integral dissipates slightly more energy than the parametric retardation function in state space form.
Free decay tests show a perfect fit between the convolution model and FAST. The state space model is found to be extremely accurate for the surge decay test, while for the pitch decay test a little discrepancy in the damped natural circular frequency exists.
The primary advantage of the state space model is the reduction of the computational cost for the resolution and it can be appreciated in table 2.2 that shows time resolution duration of the three free decay test. This makes this approach suitable for real-time implementations.

2.4 Numerical Model Development

One of the main parts of the work is the development of a numerical model for the hydrodynamics of a spar-buoy floating wind turbine (FOWT).
The goal of this code is to return the wave elevation, surge and pitch displacements for a user defined sea wave conditions and floating wind turbine geometry.
This code is completely inspired by HydroDyn module of FAST 7.
The numerical model needs the system geometry (dimensions, mass and inertia) as input and also, like FAST code, the radiation and diffraction matrix and vectors parameters, which directly depend on platform geometry. Radiation and diffraction parameters can be obtain only in two ways, which are experimental or linear potential flow analysis. FAST uses WAMIT computer program to solve the linear potential flow problem and retrieve $[A(\omega)]$, $[B(\omega)]$ and $\chi(\omega)$. WAMIT is based on a three-dimensional numerical-panel method in the frequency domain for the resolution of the waves-structure iteration problem.
The model presented in this chapter solves the floating platform dynamics of only two degrees of freedom: *surge* and *pitch*. The reason of that is because the aim of the numerical code was in first place to support the "hardware-in-the-loop" (HIL) experimental analyses which motion mechanism is design only for this two platform DoFs. More details of the wave simulator mechanism will be presented in chapter 3.
The code presented consists in a group of matlab functions. It solves the system's equation of motion in matlab or in simulink depending on how the radiation memory effect is represented. In the former case a state-space model is used and it's easy to build a state space model of the whole system and solve it in matlab, while in the latter case of a convolution integral representation the numerical model passes throug simulink enviroment for the resolution of the system's equation of motion.
For both cases it can be used an integration method based on variable or fixed time step. In particular the methods chosen are respectively ode45 and ode4.

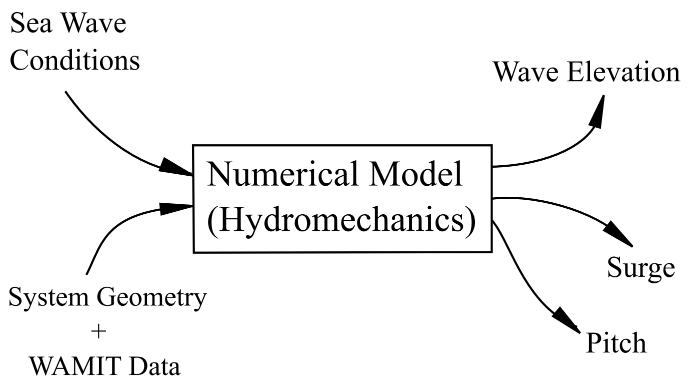


Figure 2.22: Code Flow Idea

2.4.1 Code structure

The structure of the numerical code is presented in figure 2.23.

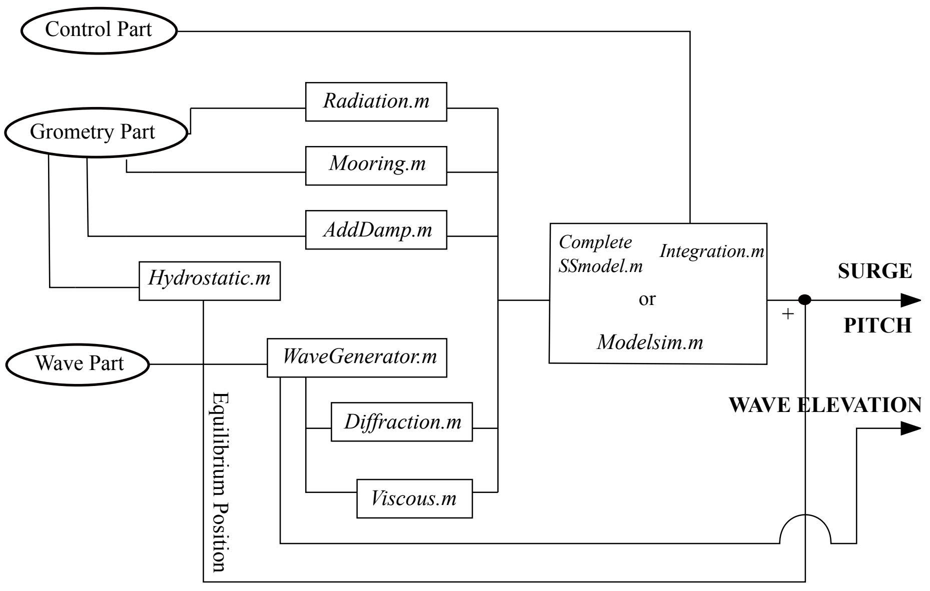


Figure 2.23: Code structure

The model can solve the dynamics both form regular or irregular incoming exciting waves.
The main differences between the code designed for this thesis and FAST 7 are here reported:

- The equilibrium position is firslty calculated from the solution of the hydrostatic problem. Once defined it does not change because it is a design requirement that diffraction and viscous forces and moments cannot change platform's equilibrium position.
- The model cannot solve the nonlinear dynamics of mooring lines. A linear form of the mooring lines forces and moments must be defined in input by user.
- Radiation memory effect resolution can be solved both via convolution integral or state space model. This option is already implemented in FAST 8.
- User can decide which integration method to use for the solution of the equation of motion. The available methods are ode4 and ode45.
- The model neglects inertia moment and gyroscopic effect of the wind turbine rotor.

2.4.1.1 Input File

The Input File is divided in three parts:

1. *Geometry Part.*
2. *Wave Part.*
3. *Control Part.*

Geometry Part collects all the geometrical and inertial characteristics of the floating wind turbine, the hydrodynamical data (radiation and diffraction matrix and vectors) previously calculated with WAMIT and the linearized model of the mooring lines. Also an additional damping matrix can be introduced.
Wave Part defines wave characteristics and includes the possibility to decide memory effect representation. User can switch from regular to irregular excitation of the system. For regular case wave amplitude and wave circular frequency must be defined, while the irregular case requires the user to specify the significant height and peak period of the wave elevation PSD_η. It is also possible to excite the system with a user-defined wave elevation. Concerning the memory effect resolution, the user can solve this term both by state space model or via convolution integral. *Wave Part* requires seabed depth for wave kinematics calculation, wave direction propagation for the diffraction force and drag coefficient for viscous force.
Control Part sets the integration parameters of the numerical model. Simulation duration and time discretization must be specified. In this part the user must choose witch integration method implement.

2.4.1.2 Numerical Model

This model is based on nine functions:

1. *wavegenerator.m*
2. *hydrostatic.m*
3. *mmooringLines.m*
4. *addamp.m*
5. *radiation.m*
6. *diffraction.m*
7. *viscous.m*
8. *completeSSmodel.m*
9. *integration.m*
10. modelsim.m

wavegenerator.m

wavegenerator.m generates the wave elevation and calculates the particle velocity along the platform draft. In case of regular wave the function builds a sine wave function with amplitude and frequency defined by user in the input file. For irregular case it uses the subroutine *jonswap.m* for the PSD_η creation and *boxmuller.m* for the creation of $WNG(\omega)$, which stands for white gaussian noise, a uniformly distribuited source with random phase across all the frequency domain. The analytical wave amplitude $\eta_a(\omega)$ for all the frequency domain is obtained from wave elevation PSD_η through equation 2.1.4.

Wave elevation spectrum is defined as :

$$E(\omega) = \eta_a(\omega)WGN(\omega) \tag{2.4.1}$$

and from this time domain formulation is obtained by superposition theory:

$$\eta(t) = \sum_{\omega} |E(\omega)|\, cos(\omega t + angle\, \{E(\omega)\}) \tag{2.4.2}$$

wavegenerator.m includes also the possibility to use a user-defined wave elevation.

The particle velocity is calculated with equation 2.1.16.

hydrostatic.m

hydrostatic.m calculates the equilibrium position of the platform, the hydrostatic restoring matrix and the mass matrix.

First of all it computes the volume V_0 of the platform resting in still water. It is assumed that the platform consists in two cylindrical regions connected by a linearly tapered conical region as described in figure 2.24. Once V_0 is defined, COB position is obtained with a weight average of each submerged region of the platform (step 3 of figure 2.24) and also the position of each structure part's centre of gravity relative to the system reference frame is known. With this information it is possible to retrieve the equilibrium position (x_0, θ_0), the hydrostatic restoring matrix $\left[C^{HS}\right]$ and mass matrix $[M^{sys}]$.

Figure 2.24 shows the first steps of the function for the COB position determination. Once Z_{COB} is defined, $\left[C^{HS}\right]$ and $[M^{sys}]$ are obtained with equations 2.2.5 and 2.2.6.

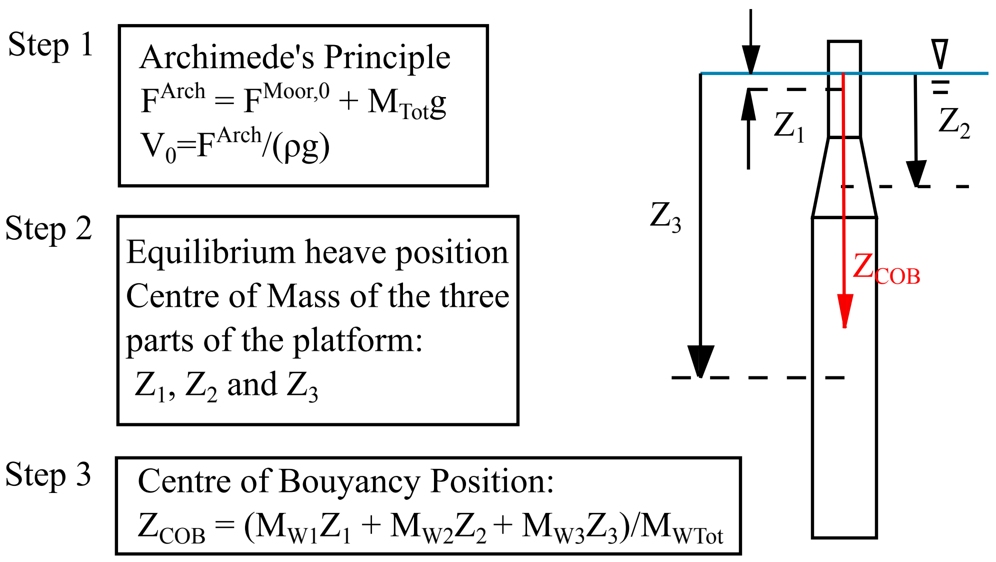

Step 1
Archimede's Principle
$F^{Arch} = F^{Moor,0} + M_{Tot}g$
$V_0=F^{Arch}/(\rho g)$
Step 2
Equilibrium heave position
Centre of Mass of the three
parts of the platform:
Z_1, Z_2 and Z_3
Step 3
Centre of Bouyancy Position:
$Z_{COB} = (M_{W1}Z_1 + M_{W2}Z_2 + M_{W3}Z_3)/M_{WTot}$
Z_1
Z_2
Z_3
Z_{COB}

Figure 2.24: Platform in its undiplaced position

M_{Wi} in figure 2.24 represents the submerged volume of the three parts that constitute the platform.

mooringLines.m

mooringLines.m assembles the mooring line matrix $\left[C^{Lines}\right]$.

addamp.m

addamp.m assembles the addtional damping matrix $\left[B^{AddDamp}\right]$.

radiation.m

radiation.m works in two different ways according to which memory effect representation is chosen from input file.

First of all it assembles added mass $[A(\omega)]$ and damping matrix $[B(\omega)]$ from the input file.

In case of convolution integral representation it builds the kernell-retardation matrix $[K(t)]$ from WAMIT data and a memory buffer of the velocity platform DoFs $\mathbf{V}_{buffer}$. The dimension of this buffer depends on the duration and discretization of radiation time t_{rad}, which is the time interval of the convolution integral, also defined by user in the input file. In case of state space model representation of the memory effect *radiation.m* uses the subroutine *ssmarco.m,* presented in Appendix A, for the construction of the matricies $[A_r]$, $[B_r]$ and $[C_r]$.

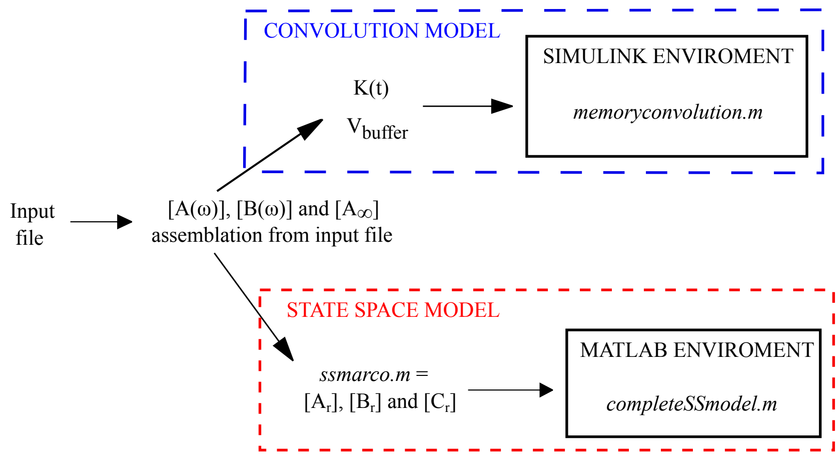


Figure 2.25: *Radiation.m* structure

diffracton.m

diffracton.m is responsible for the construction of the diffraction force. It uses the wave elevation spectrum $E(\omega)$ previously calculated by *wavegenerator.m* and the force per unit wave amplitude vector $\chi(\omega, \beta)$ from WAMIT data. $\chi(\omega, \beta)$ also depends on the wave propagation direction β, that must be provided by user in input file. Since the turbine model does not consider the inertia moment of rotor and nacelle, the wave propagation direction β is not influent in the dynamics of the system.

In first place *diffracton.m* takes diffraction terms for the selected wave propagation direction and assemble them in $\chi(\omega)$ vector. Diffraction force in time domain is calculated by superposition theory from its spectrum $\chi E(\omega)$. This terms are defined as follows:

$$\chi E(\omega) = \chi(\omega)E(\omega) \tag{2.4.3}$$

$$F^{Diff}(t) = \sum_{\omega} |\chi E(\omega)| \, cos(\omega t + angle\{\chi E(\omega)\}) \tag{2.4.4}$$

viscous.m

viscous.m calculates the relative velocity of the platform and the viscous force term with respectively equations 2.2.18 and 2.2.17.

completeSSmodel.m

In case of memory effect representation through state space model the code enables *completeSSmodel.m* function in order to solve the hydrodynamics in time domain with a state space model of the whole system.

As simplification the following substitutions are introduced:

$$[M] = [M^{sys}] + [A_\infty]$$

$$[R] = \left[B^{Ad}\right]$$

$$[K] = \left[C^{HS}\right] + \left[C^{Lines}\right]$$

$$\mu = [C_r]\,\mathbf{y}(t)$$

The model is summarized in the following system of equations:

$$\begin{cases} [M]\,\ddot{\mathbf{x}}(t) + [R]\,\dot{\mathbf{x}}(t) + [K]\,\mathbf{x}(t) + [C_r]\,\mathbf{y}(t) = \mathbf{F}^{Diff}(t) + \mathbf{F}^{Visc}(t) \\ \dot{\mathbf{y}}(t) = [A_r]\,\mathbf{y}(t) + [B_r]\,\dot{\boldsymbol{x}}(t) \end{cases} \tag{2.4.5}$$

The state space model transformation is presented as follows:

$$\begin{Bmatrix} \ddot{\mathbf{x}}(t) \\ \dot{\mathbf{x}}(t) \\ \dot{\mathbf{y}}(t) \end{Bmatrix} = \begin{bmatrix} -[M]^{-1}[R] & -[M]^{-1}[K] & -[M]^{-1}[C_r] \\ [I] & [0] & [0] \\ [B_r] & [0] & [A_r] \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{x}}(t) \\ \mathbf{x}(t) \\ \mathbf{y}(t) \end{Bmatrix} + \\ + \begin{bmatrix} -[M]^{-1} \\ [0] \end{bmatrix} \left(\mathbf{F}^{Diff}(t) + \mathbf{F}^{Visc}(t)\right) \tag{2.4.6}$$

$$\dot{\mathbf{X}}(t) = [A_{State}]\,\mathbf{X}(t) + [B_{input}]\,\mathbf{U}(t) \tag{2.4.7}$$

$[A_{State}]$, $[B_{input}]$ and $\mathbf{X}(t)$ has respectively the dimensions of [17x17], [17x2] and [17x1].

integration.m
If the code calls *completeSSmodel.m* then for the integration calls *Integration.m.* This is a function that implements the resolution method. It can switch from ode4 to ode45.

modelsim.m
In case of memory effect representation via convolution integral, the code pass through to simulink enviroment for the resolution of platform's equation of motion. *Modelsim.m* is a matlab code that calls the simulink file of the hydrodynamic model and returns matlab's workspace the wave elevation and platform displacements.

The input file, all matlab functions and the simulink model are presented in Appendix B.

2.5 Numerical Results

In this chapter will show a comparison between the results obtained with the code developed for this thesis and the ones provided by FAST is reported.
The compariosons performed are based on the following aspects:

- Wave elevation analysis:
  - Frequency domain comparison.
  - Energy content comparison.
- System displacement analysis: Surge and Pitch.
  - Time domain comparison.
  - Frequency domain comparison.
  - Memory effect model comparison.
  - Energy content comparison.
- Simulation's duration.

Several simulations were performed, both with regular and irregular state of the sea. In particular four relevant conditions of sea were chosen for tests with irregular waves. These four sea states correspond to a wind velocity that represent characteristics points of the OC3 HyWind wind velocity-power curve.

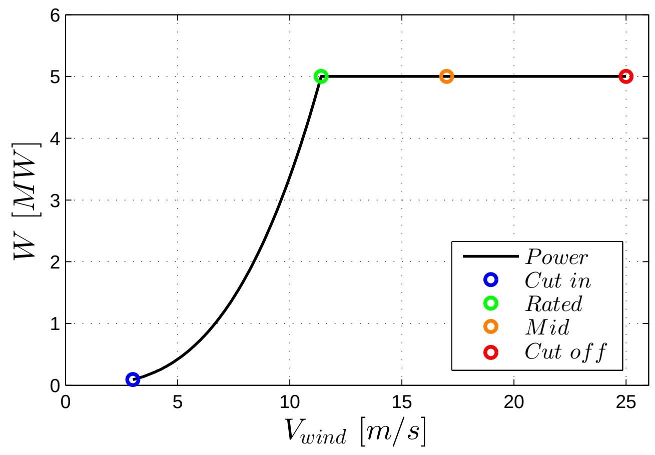


Figure 2.26: Wind Turbine Wind-Velocity Power Curve

Simulations have been resolved with 4th-order Runge Kutta method (rk4) with a variable time step, but in order to have an idea of the computational time necessary for the "hardware-in-the-loop" (HIL) real time simulations, a fixed time step integration method was used for the resolution of a set of cases.

A duration of almost 21 minutes were imposed for all simulations. Within this amount of time the transient state is not completely extinguished in terms of spectral content, but it is enough to highlight the steady behavior of the platform in time domain.

In order to highlight only the difference of numerical model representation based on the hypothesis mentioned in section 4 of this chapter, some tests for irregular waves have been carried out imposing the same input for both codes. In this way same spectral content of the system's excitation forces is guaranteed.

2.5.1 Regular Wave Tests

For regular wave simulations just a wave elevation case were considered. This wave has the following characteristics (Table 2.3):

ω_η	0.5 rad/s
η_a	2 m
ϕ_η	1 rad

Table 2.3: Regular Wave Elevation

For these kind of simulations the reproduction of the same wave elevation for both codes it is ensured, because there is no random factor in wave generation. Same input for both code systems is guaranteed as reported in figure 2.27.

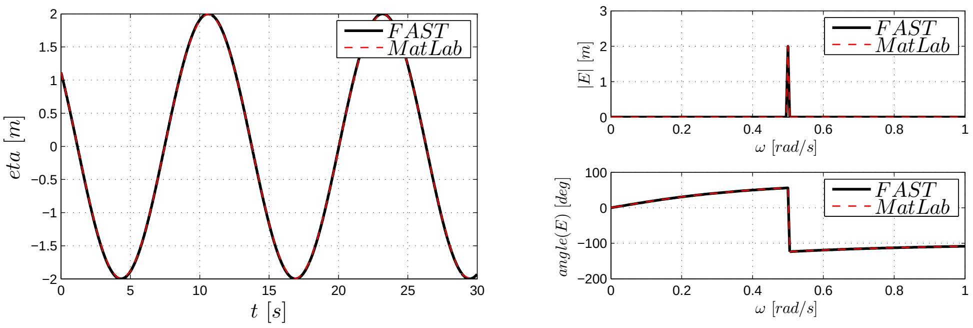


Figure 2.27: Regular Wave Elevation

Results are presented in figures 2.30 to 2.31.

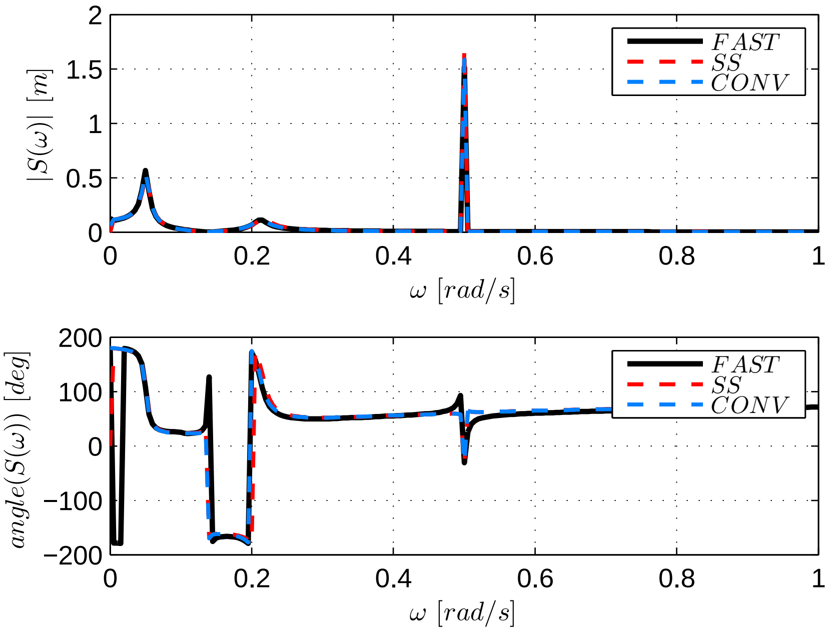


Figure 2.28: Regular: Surge in frequency domain

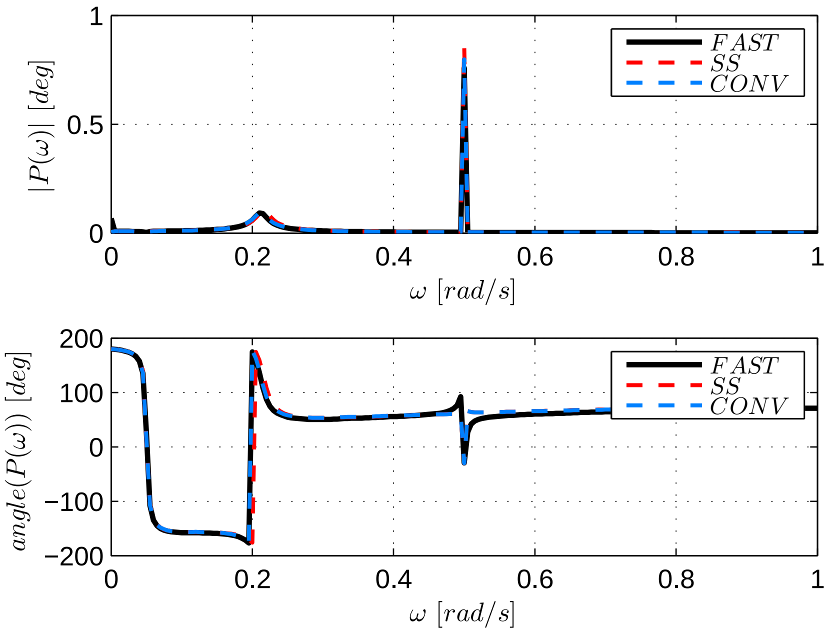


Figure 2.29: Regular: Pitch in frequency domain

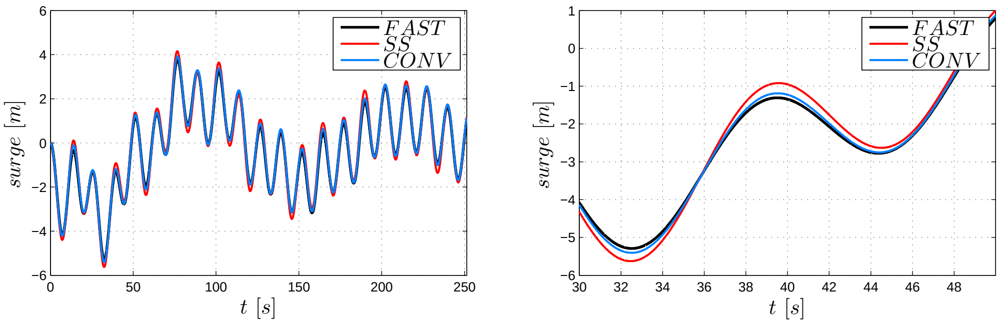


Figure 2.30: Regular: Surge in frequency domain

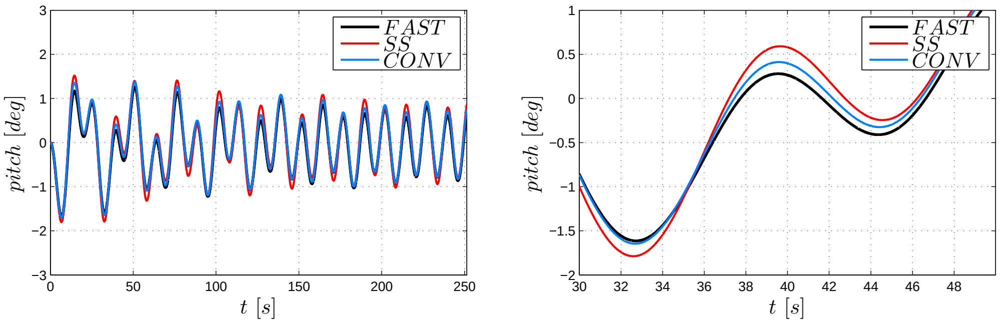


Figure 2.31: Regular: Pitch in time domain

Figures show a good match between the two codes. As proved in section 3, memory effect's state space model dissipates less energy than convolution integral model. This can be appreciated by observing the peaks of the frequency responses of surge and pitch which are lower for the tests that use convolution integral for radiation problem.
Also some minor differences in surge and pitch phases compared to FAST can be noted. The cause must be the different representation of mooring lines forces which leads to a different poles location in the state matrix of the whole system $[A_{state}]$. This difference is not appreciable in time domain and it is possible to assume that a linear model for mooring lines term is a good approximation.
The execution time of the models are reported in the following chart.

	conv rk4	conv rk45	SS rk4	SS rk45	FAST
Simulation Duration [s]	4804.6	1201.2	300.05	4.72	482.1
$\frac{\text{Duration}_\text{i}}{\text{Duration}_\text{FAST}}$	99.659	2.4916	0.6224	0.0098	1

Table 2.4: regular tests execution time.

$H_S\,[m]$	$T_P\,[s]$	$V_{Wind}\,[m/s]$
0.09	2	2.5
0.67	4.8	7.5
0.88	5.4	8.6
1.4	6.5	10.5
1.86	7.2	12.1
2.44	8.1	13.6
3.66	9.7	17.6
4.57	10.5	22
5.49	11.3	25.8
6.71	12.1	35.1

Table 2.5: Wind velocity and wave state corrispondeces

As expected convolution integral model is highly time consuming compared with the state space model representation of the memory effect. Table 2.4 shows that *completeSSmodel.m* is much faster than *modelsim.m* and FAST. Also it's evident that FAST, which solved not only the radiation problem via convolution but also the mooring lines term with an iteratively model, is highly optimized.

2.5.2 Irregular Wave tests

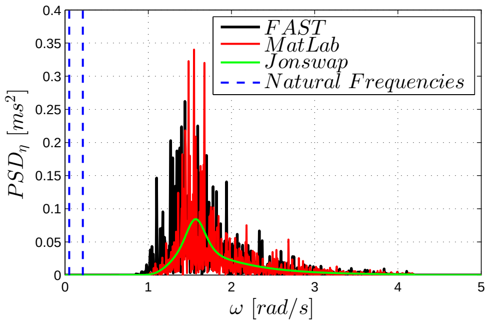
As mentioned in the introduction, for simulations with irregular sea waves the following cases were considered.

1. Cut in.
2. Rated.
3. Mid point between Rated and Cut off.
4. Cut off.

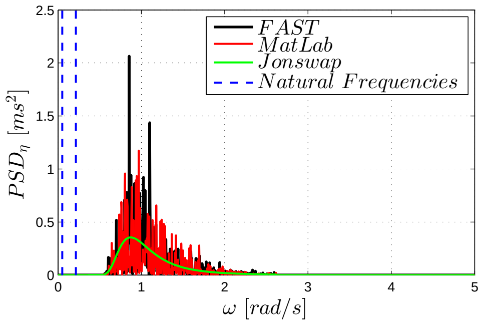
Table 2.5 shows a corrispondence between wind velocity and sea wave conditions. Since wave elevation formulation for irregular waves assumes a random factor, which is the phase generation of each wave frequency component, for this kind of simulations it is impossible to guarantee the same input in terms of amplitudes and phases. Therefore, the only reasonable comparison among the code inputs is to compare the energy content of the wave elevation power spectral densities generated by the two codes.
Figure 2.32 shows wave elevation power spectral densities PSD_η generated by *jonswap.m*, FAST and Jonswap analitical formulation, of the four cases taken in consideration.

Wave Energy	Jonswap	Matlab	Fast	$\frac{Matlab}{\text{FAST}}$	$\frac{\text{Jonswap}}{\text{FAST}}$
Cut in	4.98e-4	4.94e-4	5.2e-4	1.051	1.0411
Rated	0.2129	0.1999	0.2062	1.0314	0.9683
Midpoint	0.8244	0.7815	0.7623	0.977	0.9247
Cut off	1.8628	1.8531	1.8201	0.9822	0.9771

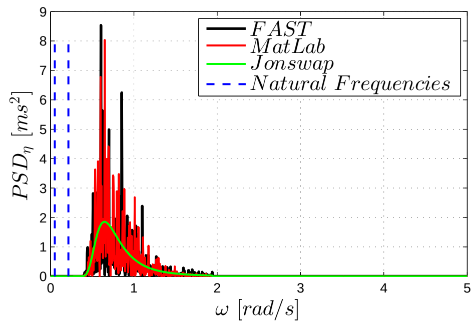
Table 2.6: Wave Elevation PSD Energy Content



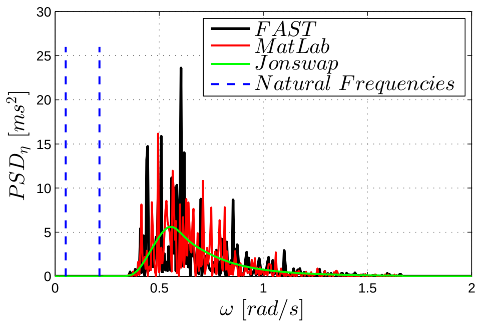
(a) Cut in conditions: $H_s = 0.09$ [m], $T_p = 2$ [s]



(b) Rated condition: $H_s = 1.86$ [m], 7.2 [s]



(c) Mid conditions: $H_s = 3.66$ [m], $T_p = 9.7$ [s]



(d) Cut off condition: $H_s = 5.49$ [m], $T_p = 11.3$ [s]

Figure 2.32: Wave Elevation Power Spectral Densities

Figure 2.32 and table 2.6 show that WGN inevitably leads to some differences in the PSD_η, but anyway their energy content is comparable.

From figure 2.32 it is also possible to verify that these wave state conditions do not excite the natural frequencies of the platform, indicated in table 2.7, and resonance phenomena is avoided for the entire operating range of the floating wind turbine. This is consistent with the deign requirements of tis kind of loating platform. However TLP-like floating platforms are designed to have the natural frequency falling within a higher range than the sea spectrum.

	$\omega_0\ [rad/s]$
Surge	0.051
Pitch	0.2135

Table 2.7: Natural circular frequencies of the platform

Concernig numerical simulations cut off sea wave state was taken as refernce, and for this a detailed analysis, of the same kind of the regular wave case, were performed. Cut off was chosen because is the most critical sea wave condition of the four. It can lead to off design operating conditons of wind turbine that can drastically penalize power generation.

In addiction, because of the random nature of irregular waves, an energy content analysis of surge and pitch was made for the comparison with FAST.

Moreover, for a few tests wave elevation generated by FAST was imposed as input . This simulations aim to highlight just the differences of the two codes outputs.

2.5.2.1 Cut off simulations

Cut off simulations are divided in two groups depending on the parameters assigned as input:

1. Wave elevation power spectral density PSD_η parameters.

2. Wave elevation defined by user.

Simulations of type 1

The results obtained are presented in figures from 2.33 to 2.36.

Tables 2.8 and 2.9 show the energy content of surge and pitch and the duration of each simulation.

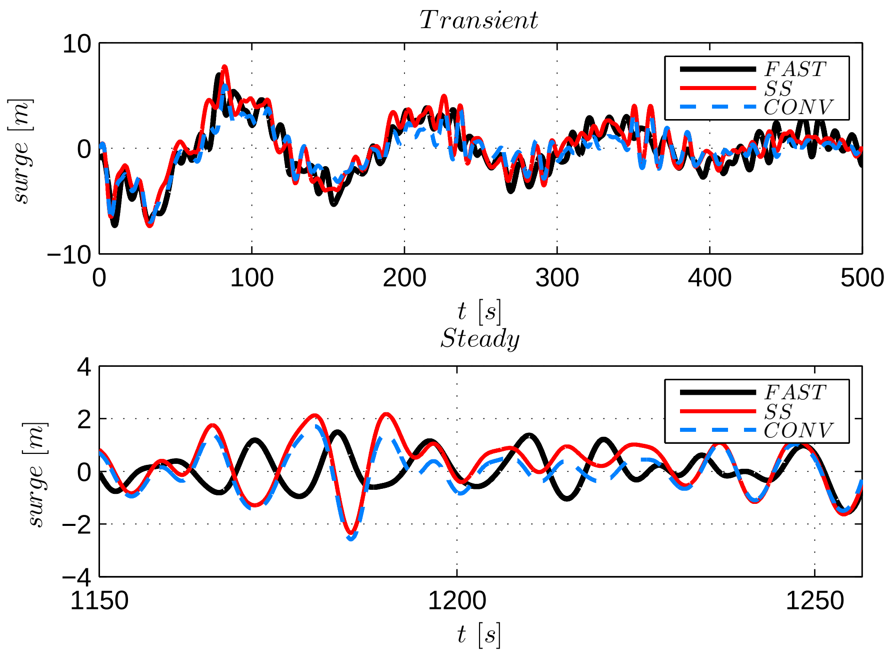


Figure 2.33: Cut off conditions: Surge in frequency domain

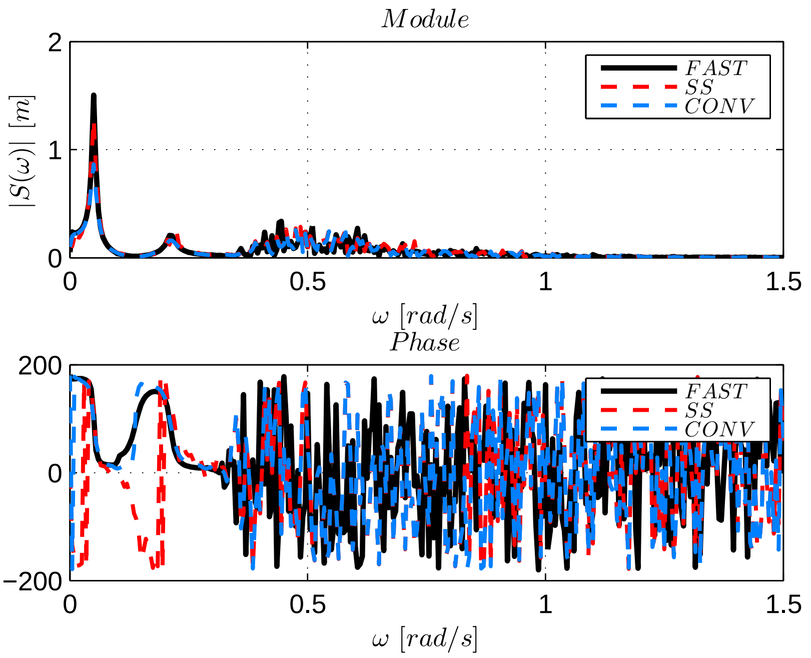


Figure 2.34: Cut off conditions: Surge in frequency domain

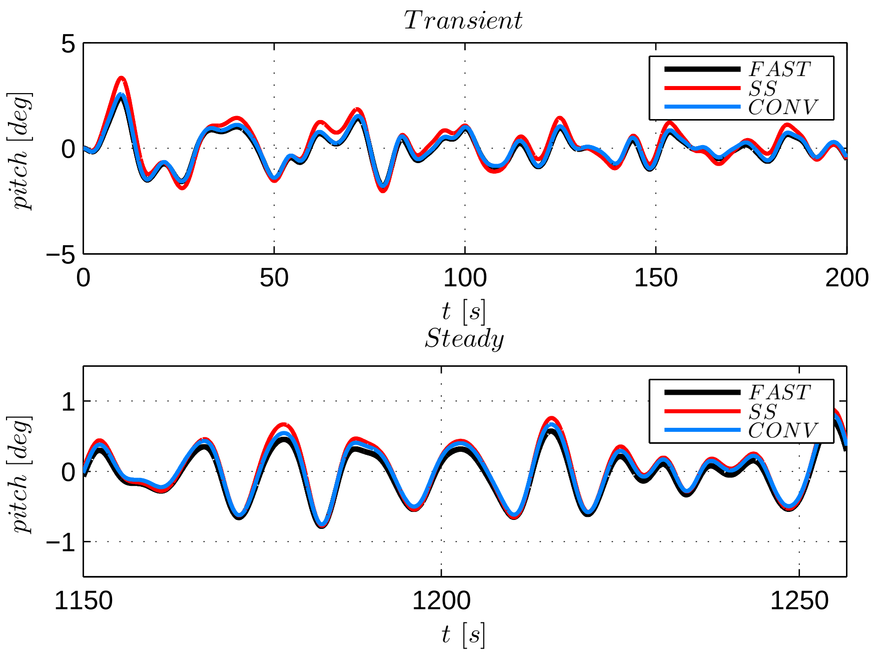


Figure 2.35: Cut off conditions: Pitch in time domain

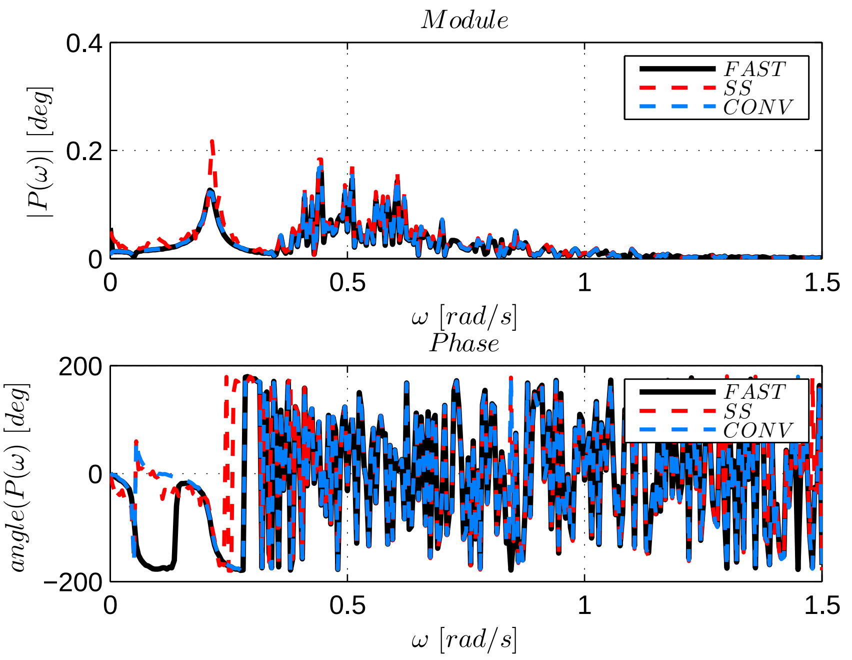


Figure 2.36: Cut off conditions: Pitch in frequency domain

	SS rk45	SS rk4	conv rk45	conv rk4	FAST
Wave Elevation Energy	0.1135				0.1102
Surge Energy	0.1043	0.1043	0.0938	0.0935	0.0976
Pitch Ene rgy	0.0433	0.0433	0.0399	0.0392	0.382
Surge Energy/Wave ElevationEnergy	0.9188	0.9188	0.8264	0.8236	0.8862
Surge Energy/Wave ElevationEnergy	0.3814	0.3814	0.3510	0.3458	0.3468

Table 2.8: Energy Content of the Output Parameters of the Cut Off Simulation

	SS rk45	SS rk4	conv rk45	conv rk4	FAST
Duration [min]	0.14	8.1	55	63	25
$\frac{\text{Duration}_i}{\text{Duration}_{\text{FAST}}}$	0.0056	0.324	2.2	2.52	1

Table 2.9: Irregular Simulation Duration

Simulations of type 2

The results obtained are presented in figures from 2.37 to 2.40.

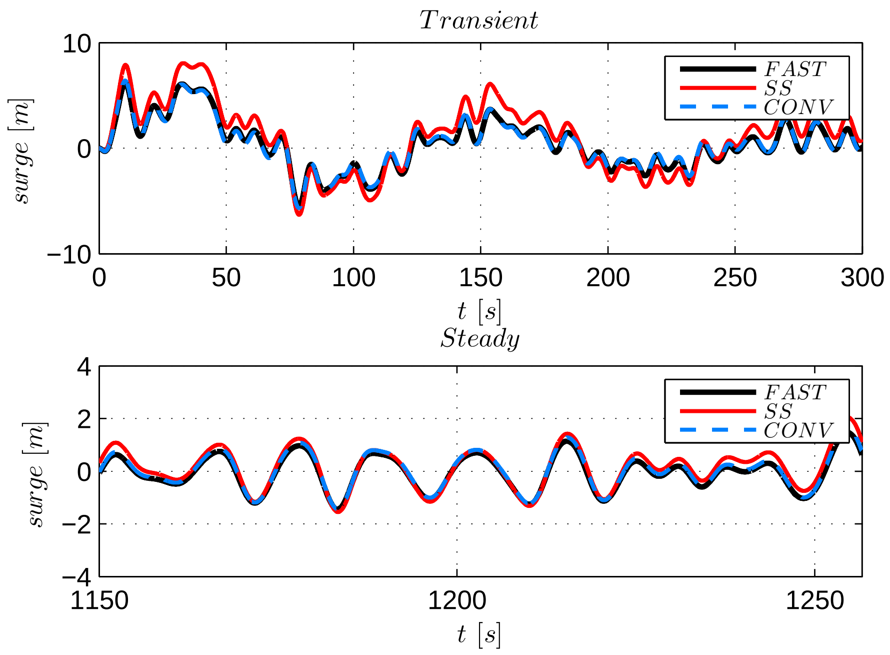


Figure 2.37: Cut off conditions: Surge in frequency domain

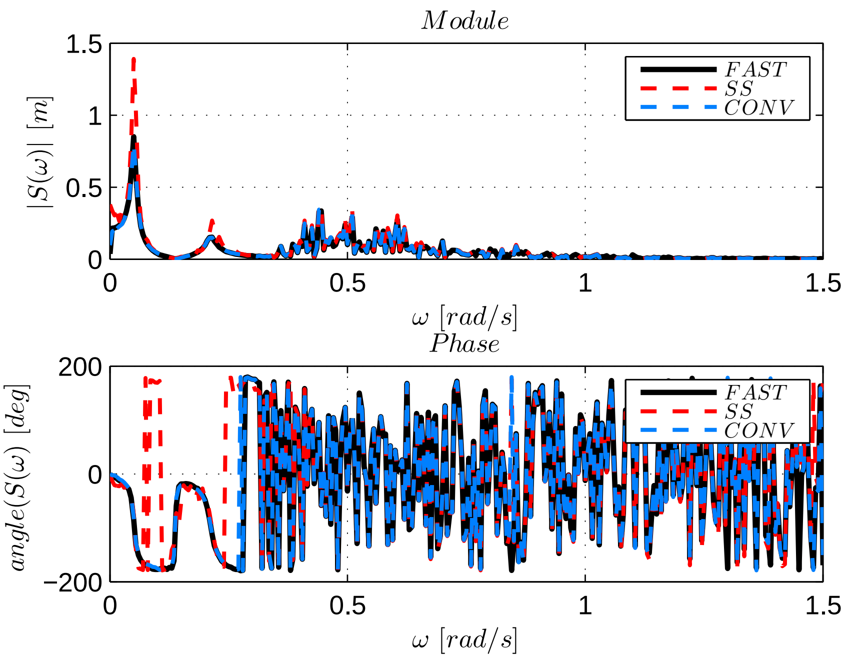


Figure 2.38: Cut off conditions: Surge in frequency domain

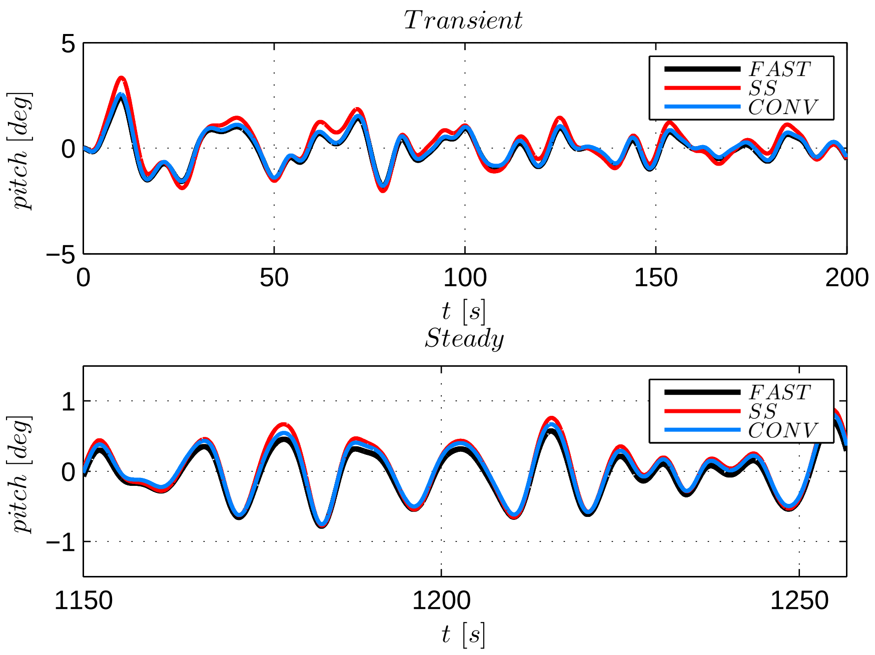


Figure 2.39: Cut off conditions: Pitch in time domain

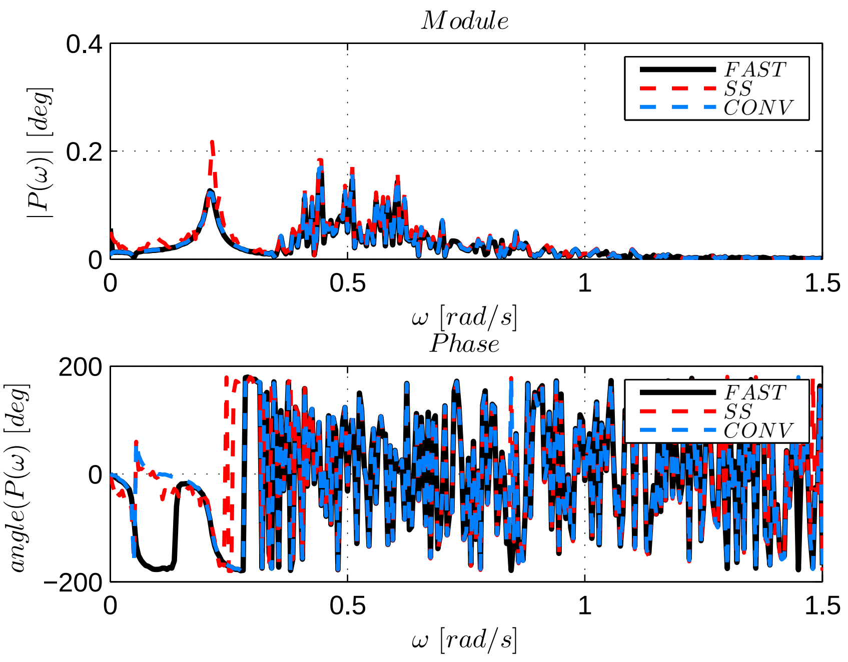


Figure 2.40: Cut off conditions: Pitch in frequency domain

Irregular wave tests show a good matching with FAST results for both simulations groups. Both time and frequency domain comparisons lead to the same conclusions of the regular tests. Not only figures from 2.33 to 2.44 but also table 2.8 show that state space model dissipates less energy than the convolution model. Convolution model simulations, especially the ones solved with the fixed time step method, represent the best fit for reproduce FAST results. The negative aspect is that it takes more than double of the FAST simulation duration. Even for the irregular case tests mooring lines forces linearisation has the same behavior of the non-linear model in time domain.

2.5.2.2 Other simulations

For the other three cases only the results of the model which uses state space system for memory effect representation are presented in this work. Results are shown in figures from 2.41 to 2.52

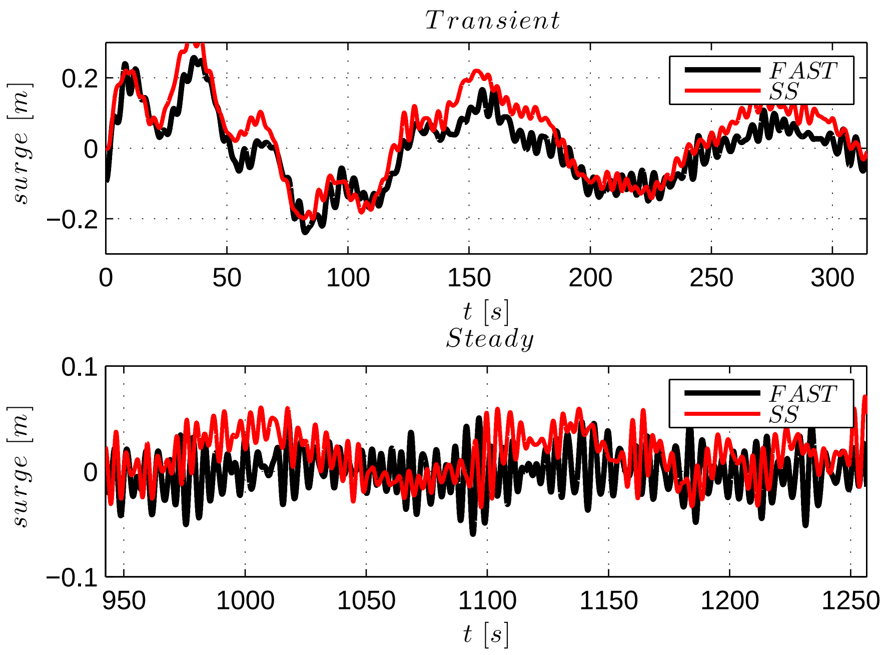


Figure 2.41: Cut in conditions: Surge in frequency domain

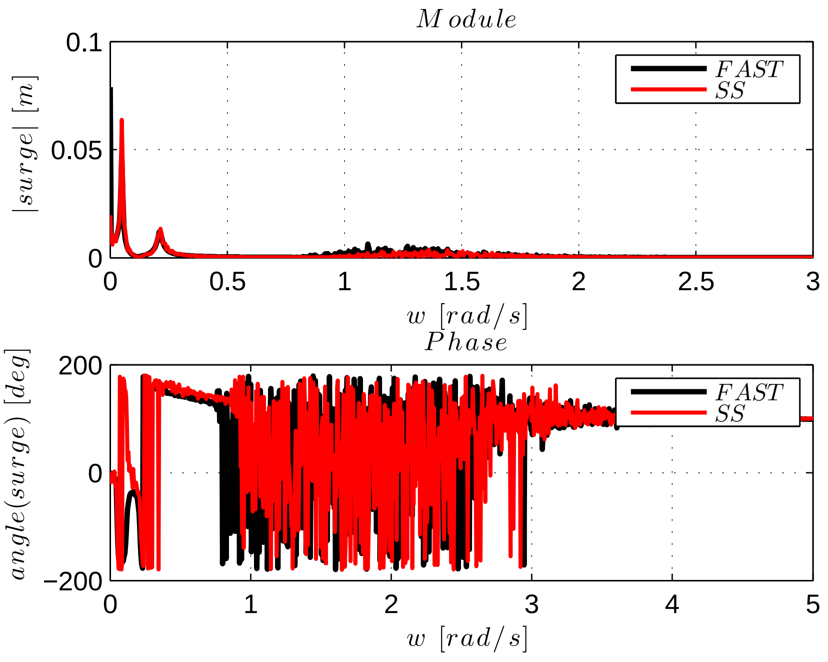


Figure 2.42: Cut in conditions: Surge in frequency domain

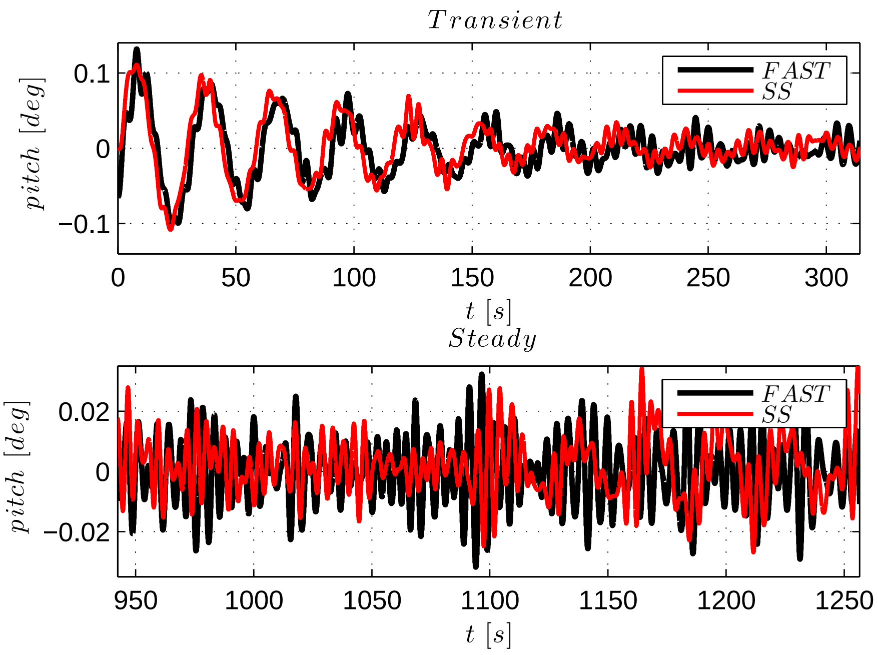


Figure 2.43: Cut in conditions: Pitch in time domain

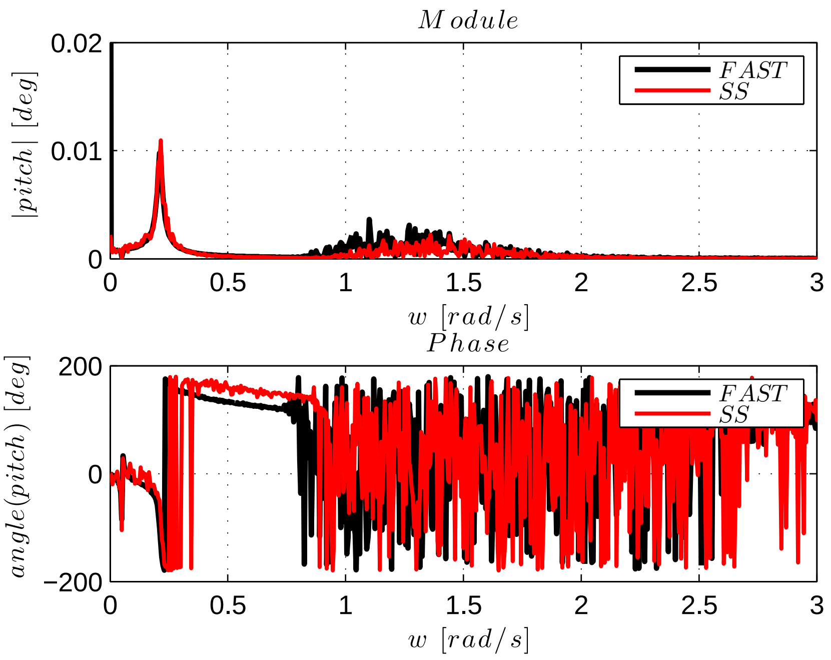


Figure 2.44: Cut in conditions: Pitch in frequency domain

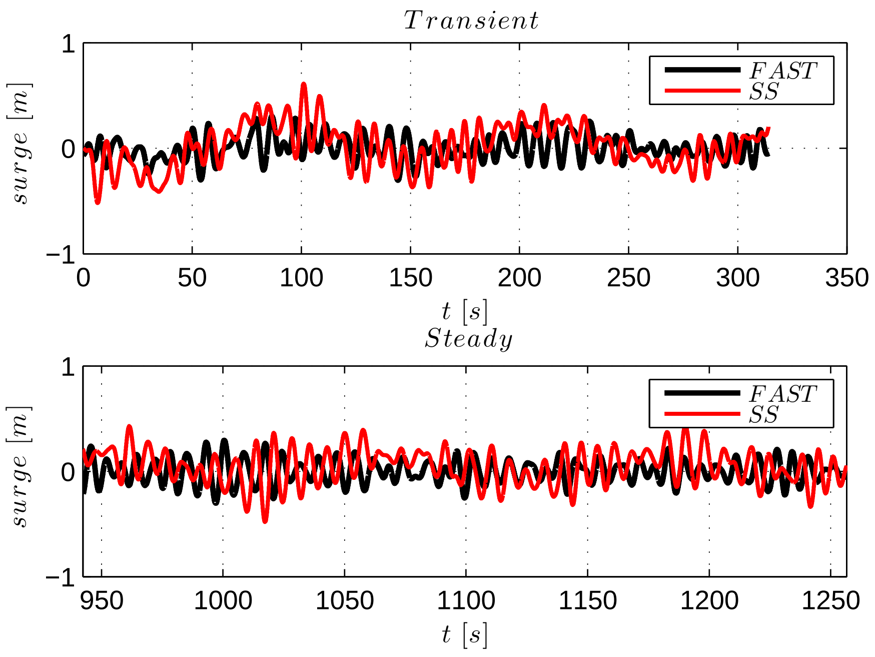


Figure 2.45: Rated conditions: Surge in frequency domain

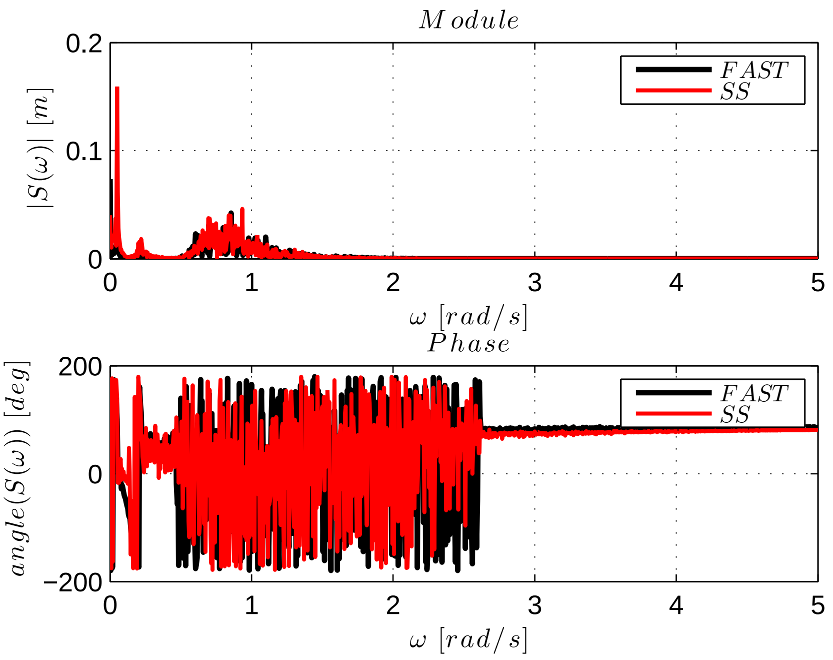


Figure 2.46: Rated conditions: Surge in frequency domain

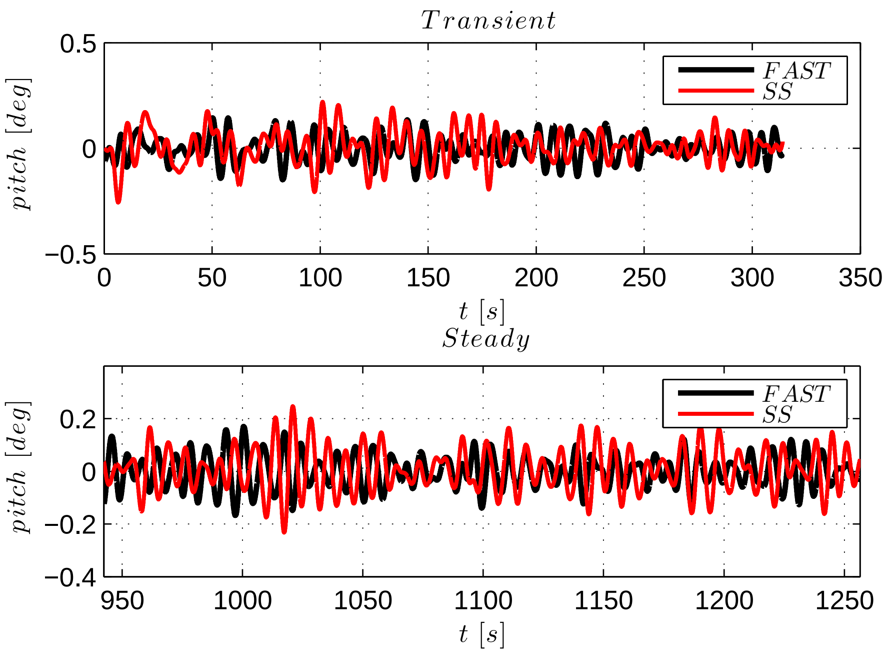


Figure 2.47: Rated conditions: Pitch in time domain

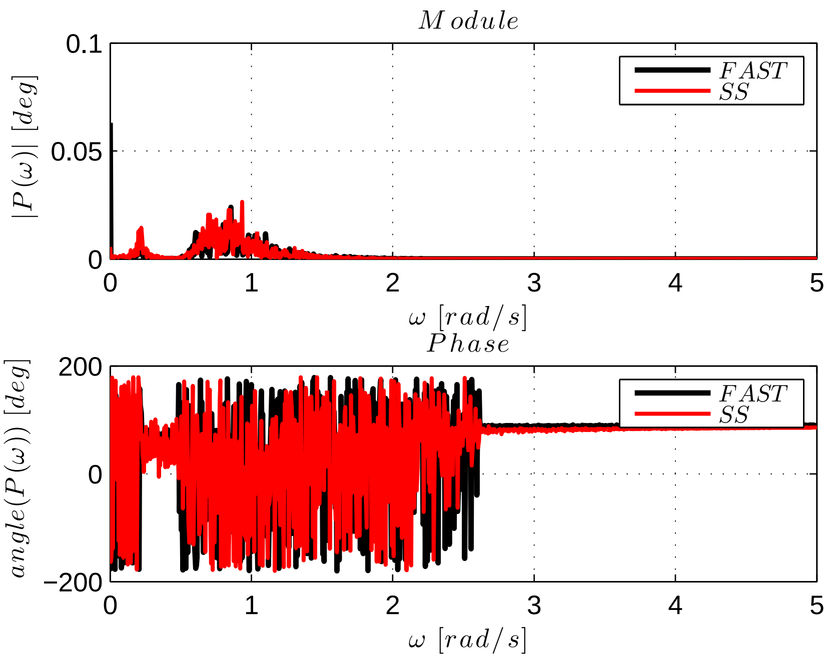


Figure 2.48: Rated conditions: Pitch in frequency domain

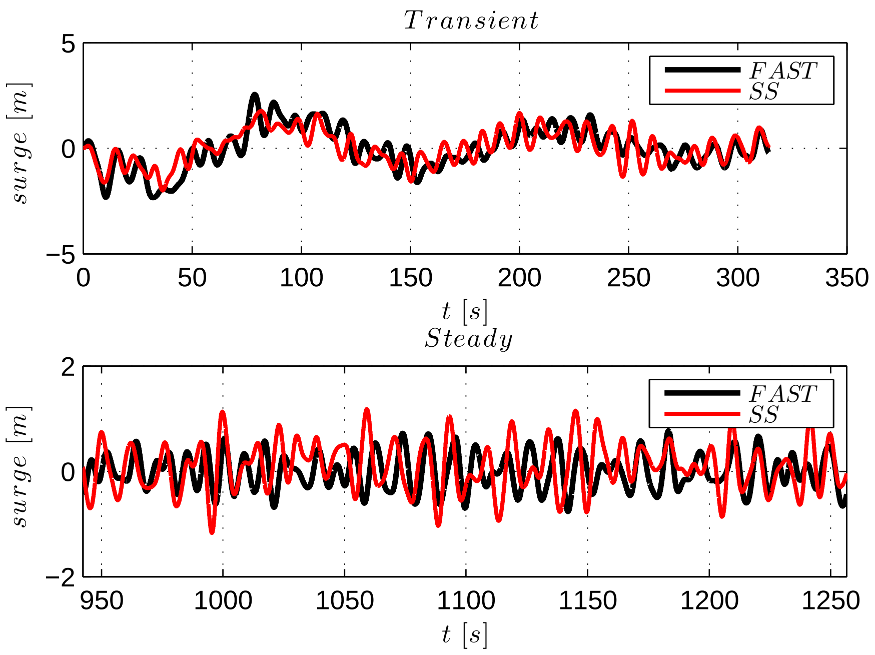


Figure 2.49: Mid conditions: Surge in frequency domain

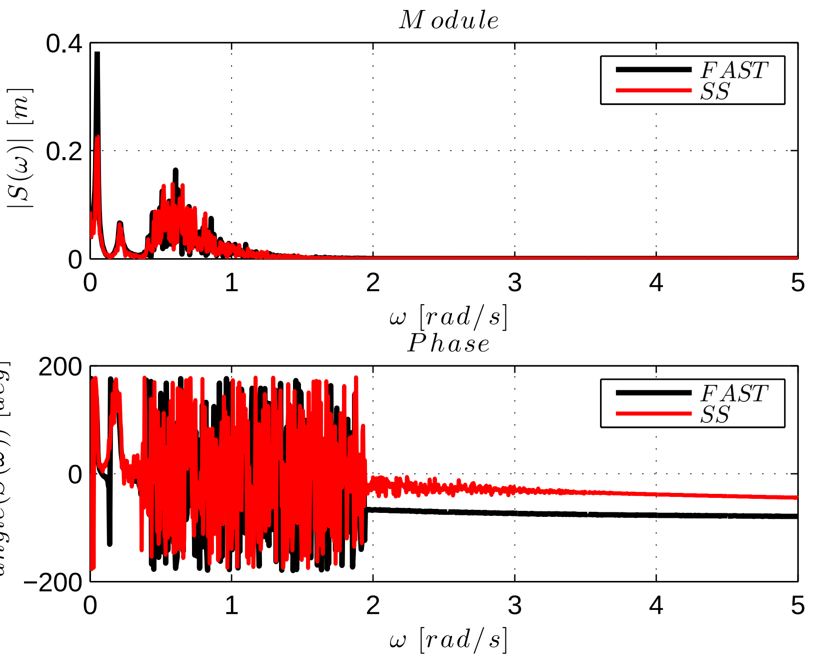


Figure 2.50: Mid conditions: Surge in frequency domain

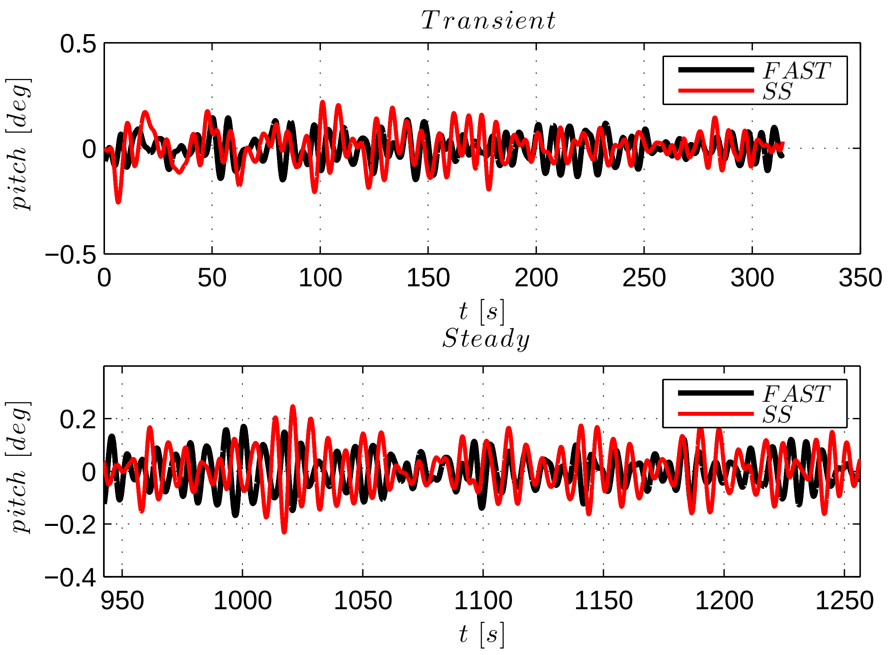


Figure 2.51: Mid conditions: Pitch in time domain

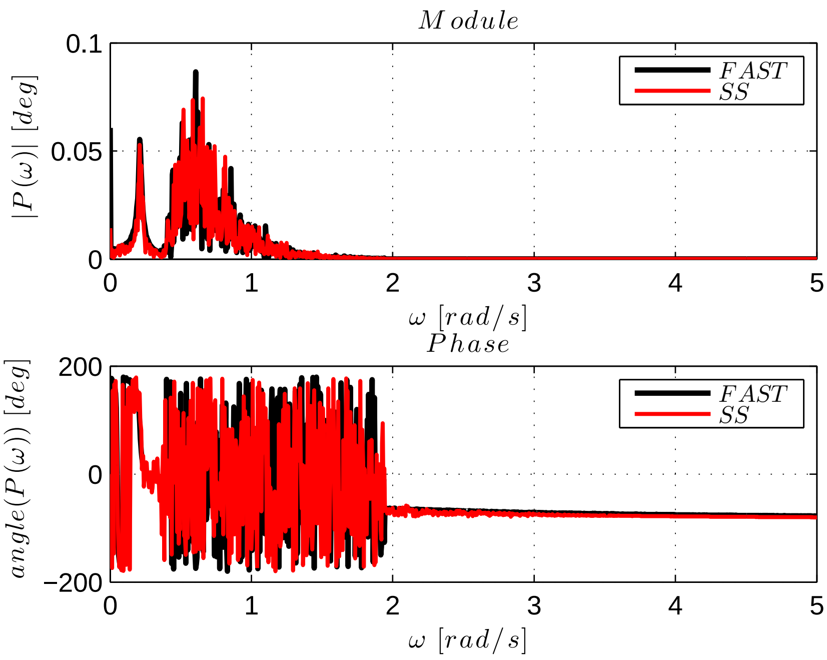


Figure 2.52: Mid conditions: Pitch in frequency domain

Chapter 3

Modeling for Hardware in the Loop Implementation

As floating wind turbines are becoming a great object of interest among international engineering community, experimental tests are necessary to validate numerical simulators. Also, measurements on real prototypes are still not available, or scares to this aim. In order to validate numerical codes experimental campaigns on proper scaled models are required. Based on the parameters coming from experimental campaigns it is possible to validate and correct aero-hydrodynamical algorithms (e.g the additional damping term introduced in section 2.7 of chapter 2).

The final goals of this work is to reproduce in a wind tunnel facility the operational conditions that occurs in relation to the nominal functioning of a floating wind turbine, with the remarkable outcome of implementing and comparing strategies for pitch and rotational speed control in order to optimize power generation due to specific environment(i.e aero- and hydro-dynamic inputs).

This chapter deals with implementation of the numerical code reported in chapter 2 in a real time hardware-in-the-loop (HIL) experimental rig. A two degrees of freedom mechanism for the simulation of the platform hydrodynamics was adopted. The mechanism is designed so that a wind turbine model can be mounted on it and is moved by means of two hydraulic actuators whose inputs are consistent with the real-time HIL system (i.e the output of the model reported in chapter 2).

In order to run efficiently the model of Chapter 2 through a real time implementation, great effort was put in the simplification of the hydrodynamic model. Therefore the classical Cummins representation of the radiation problem was replaced by the Fossen&Perez state-space formulation [24], as thoroughly explained in section 2.3 of chapter 2. This turned out to define a model suitable for real-time HIL implementation

In order to implement a real time simulation it's beneficial, and sometimes necessary, that computer system manages a low computational cost mathematical model. Therefore, the development and adoption of a state space model for memory effect and a linear model for mooring lines term were necessary for a correct and fast implementation of the model.

3.1 Hardware In the Loop Simulation

Hardware-in-the-loop (HIL) simulation, is a technique that is typically adopted in the development and test of complex real-time embedded systems. HIL simulation provides an effective platform by adding the complexity of the plant under control to the test platform. The complexity of the plant under control is included in test and development by adding a mathematical representation of all related dynamic systems. These mathematical representations are referred to as the "*plant simulation*". The embedded system to be tested interacts with this plant simulation. A HIL simulation must include electrical emulation of sensors and actuators. These electrical emulations act as the interface between the plant simulation and the embedded system under test. The value of each electrically emulated sensor is controlled by the plant simulation and is read by the embedded system under test (feedback). Likewise, the embedded system under test implements its control algorithms by outputting actuator control signals. Changes in the control signals result in changes to variable values in the plant simulation.

3.2 Scaled Hydrodynamics

3.2.1 Similitude Theory

A model is a representation of a physical system often used for predicting the behavior of a system in a certain condition. Generally the model has different dimensions with respect of the real system but if similitude principle is respected is possible to draw conclusions even for the real system [4].

For the experimental campaign a scaled rigid model of Vestas V52 wind turbine were used. This model is not consistent in terms of aeroelsticity with the OC3 Hywind wind turbine, but since it is a rigid model, this incongruity do not compromise the experimental procedure's validity, which is the aim of this work.

The geometric characteristics of the Vestas V52 wind turbine model used for the simulations are presents in table 3.1.

Parameter	Unit	Value
Rotor Diameter	[m]	1.04
Hub Height	[m]	1.1
Blades	[]	3
Blade length	[m]	0.506
Rotor Shaft	[°]	6
Nacelle weight	[kg]	0.17
Rotor weight	[kg]	0.08
Tower weight	[kg]	0.506

Table 3.1: Geometric characteristics of wind turbine scale model

A geometric scale factor λ, defined as

$$\lambda = \frac{Scale\,Model\,Dimension}{Full\,Scale\,Model\,Dimension}, \tag{3.2.1}$$

of 1/70 were adopted in order to respect the ratio between the scale model and full scale model tower height. The choice of this specific scale factor is due to the most influent role that tower' s height plays in the mass and hydrostatic matrices definitions.
Moreover working with a high geometrical scale, that makes Reynolds similitude very hard to pursue as scaling method, and also because experimental tests have been performed without aerodynamic loads, a model scale procedure based on Froude's similitude theory was adopted [26]. Froude's similitude imposes:

$$Fr_{Full\,Scale} = \frac{V_{Full\,Scale}}{\sqrt{gL_{Full\,Scale}}} = \frac{V_{Model}}{\sqrt{gL_{Model}}} = Fr_{Model} \rightarrow \lambda_V = \sqrt{\lambda} \tag{3.2.2}$$

where λ_V is the velocity scale.
Air, water and even structures density were considered to have a scale factor of 1 for the model parameters definition.
Also acceleration scale factor must be kept to 1 because the gravitational acceleration is impossible to vary.
Based on the last assumptions, regarding acceleration, density scale factors and Froude's similitude, it is possible to define all scale factors that have a significant role in the hydrodynamical model. These scale factors are reported in table 3.2

3.2.2 Implementation of the scaled Hydrodynamics

For the hydrodynamical part of model, since it is numerically represented, it is possible to obtain a perfect scale for all geometric parameters and scale procedure consists just in the implementation of an input file with floating wind turbine's geometric parameters properly scaled based on equations in table 3.2.
Also the hydrodynamics must be scaled. Radiation data must be treated as a normal mass and damping matrices terms, so that they can be scaled by $\lambda_{[M]}$ and

Scale Factor	λ_i
Velocity λ_V	$\sqrt{\lambda}$
Time λ_T	$\sqrt{\lambda}$
Frequency λ_f	$1/\sqrt{\lambda}$
Force λ_F	λ^2
Moment λ_C	λ^3
Mass λ_M	λ^3
Inertia Moment λ_J	λ^4

Mass Matrix $\lambda_{[M]}$	$\begin{bmatrix} \lambda^3 & \lambda^4 \\ \lambda^4 & \lambda^5 \end{bmatrix}$
Damping Matrix $\lambda_{[R]}$	$\begin{bmatrix} \lambda^{2.5} & \lambda^{3.5} \\ \lambda^{3.5} & \lambda^{4.5} \end{bmatrix}$
Stiffness Matrix $\lambda_{[K]}$	$\begin{bmatrix} \lambda^2 & \lambda^3 \\ \lambda^3 & \lambda^4 \end{bmatrix}$

Table 3.2: Scale Factors

$\lambda_{[R]}$ parameters of table 3.2. Diffraction data, which represents the force acting on the platform per unit wave amplitude, is scaled with a scale factor defined as the ratio between the force or moment (depending on the DoF) scale factor and the geometric scale factor.

$$\lambda_\chi = \begin{bmatrix} \lambda_F/\lambda \\ \lambda_C/\lambda \end{bmatrix} = \begin{bmatrix} \lambda^2 \\ \lambda^3 \end{bmatrix} \tag{3.2.3}$$

As mentioned in the introduction of this chapter it is necessary to adopt the state space representation for the memory effect in order to implement a real time simulation.

Concerning the parametric transformation of the retardation function based on FDI Toolbox algorithm, several difficulties were found. In fact the retardation function obtained with radiation scaled values parameters, $[A(\omega)]$ and $[B(\omega)]$, does not allow the Least Square Method algorithm to converge with low order, and that makes impossible to obtain a good fit. Figure 3.1 show the results obtained with FDI Toolbox code.

In order to avoid this problem it is necessary to scale retardation function $[K(\omega)]$ obtained from radiation full scale parameters and than apply the Least Square Method algorithm (*marcofit.m).* In this way a good match between the analytical and parametric form of the retardation function was possible to obtain. Results are reported in figure 3.2.

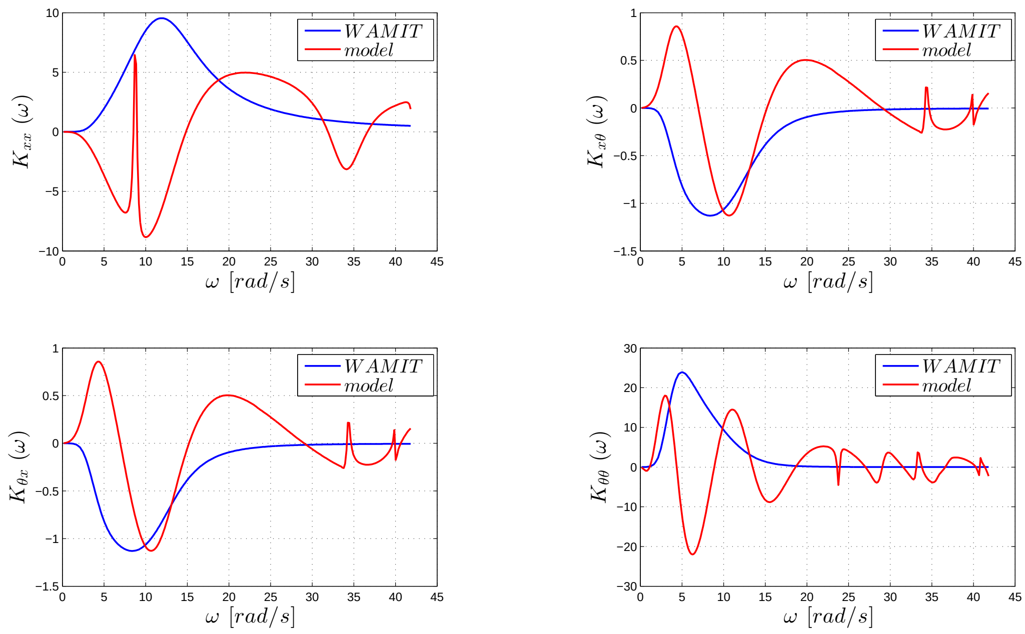


Figure 3.1: Parametric transformation of scaled retardation function from FDI Toolbox

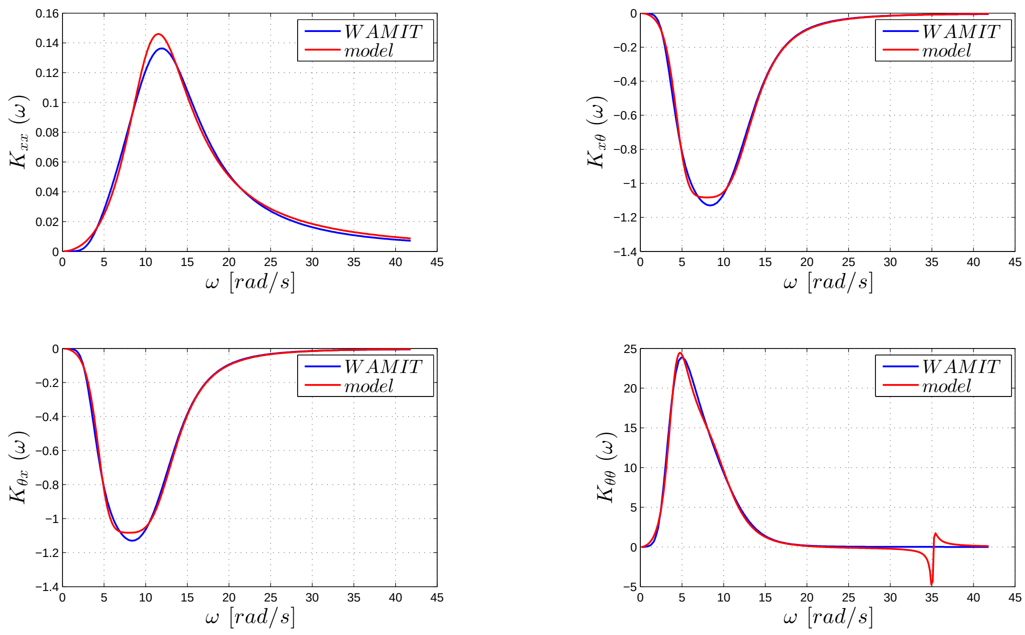
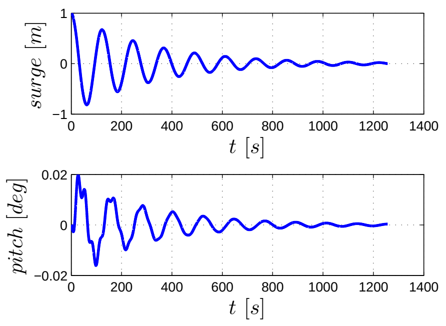


Figure 3.2: Parametric Transformation of Scaled Retardation Function from FDI Toolbox

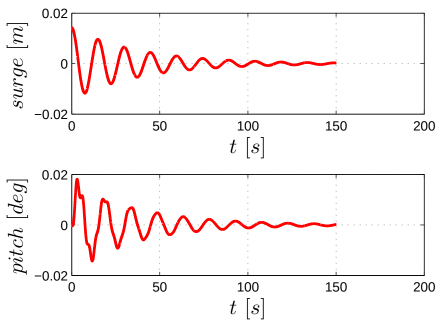
3.2.3 Numerical tests

In order to validate the scaling procedure adopted a free decay test, a regular and irregular excitation tests were performed. Regular and cut off cases presented in chapter 2 are taken as examples.

Results are shown in figures from 3.3 to 3.6.

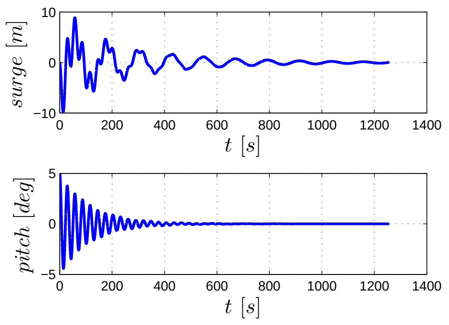


(a) Full scale

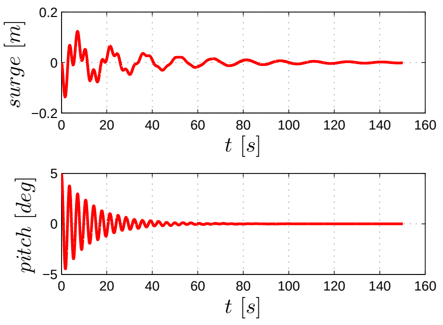


(b) Model

Figure 3.3: Free decay tests on Surge

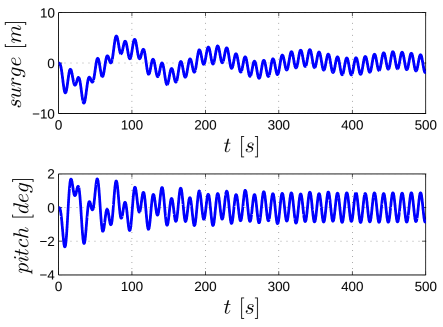


(a) Full scale

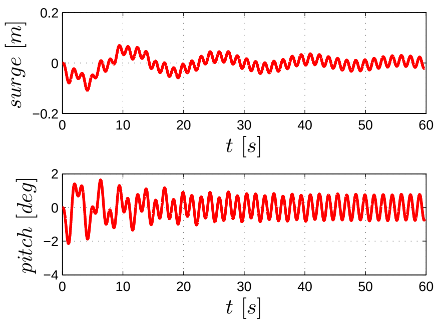


(b) Model

Figure 3.4: Free decay test on Pitch

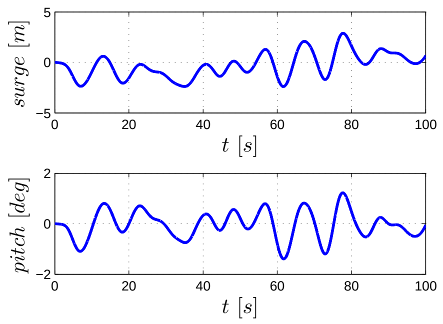


(a) Full scale

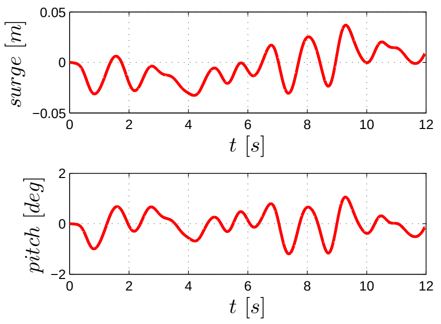


(b) Model

Figure 3.5: Regular test



(a) Full scale



(b) Model

Figure 3.6: Cut off test

3.3 Platform Motion Simulator

In this part of the chapter the main characteristics of the experimental rig for the real time HIL simulations will be reported.

The experimental set up consists in a two degrees of freedom mechanism developed and validated at Politecnico di Milano [1]. The experimental configuration of the wave simulator is depicted in figure 3.7.

The measuring system is design also for the acquisition of aerodynamic forces even though the experimental tests were carried out in absence of wind, because the simulator system, once is validated, will be tested in the wind tunnel facility.

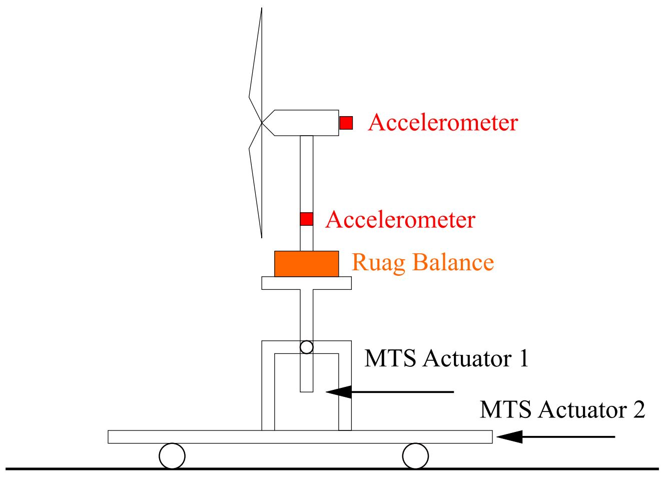


Figure 3.7: Motion mechanism configuration

3.3.1 Mechanism features

With reference of Fig 3.8 the mechanism system is able to reproduce the Pitch/Roll rotations and the Surge/Sway displacements of a floating wind turbine scale model.

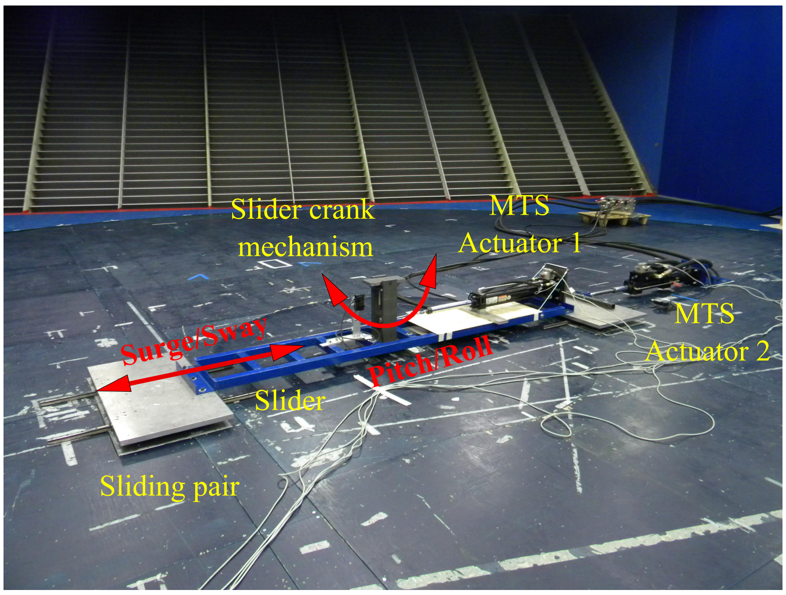


Figure 3.8: Wave simulator mechanism

The platform consists in a long slider able to slide on a rail fixed to the ground, which is mechanically coupled to the slider through a sliding pair. Horizontal motion is provided by a MTS hydraulic actuator (actuator 2) giving the motion

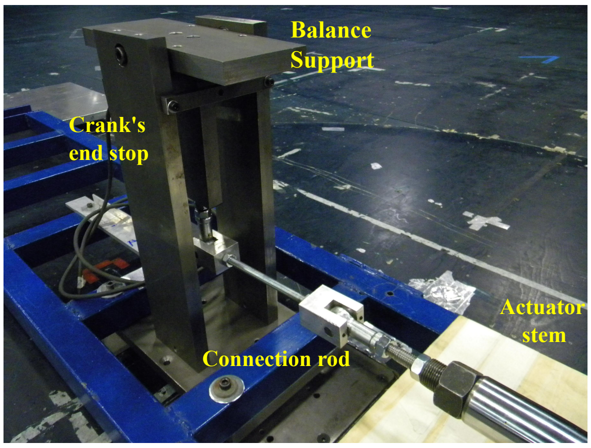


Figure 3.9: Rotational mechanism detail

thrust to the slider through a coupling bar fixed on the slider plate. An other MTS hydraulic actuator (actuator 1) is mounted upon the slider itself. Its main function is to provide motion along the rotational degree of freedom and it also works to prevent any motion of the actuator due to inertial forces acting on during the operation. This is obtained through a slider-crank mechanism connected to the actuator and mounted upon the slider. As it can be seen in Fig 3.9, crank's stroke is limited by an end stop bar mounted for safety reasons, in order to prevent dangerous rotations of the wind turbine model, in cases of wrong digital signal sent to actuator's control system during the test. A similar safety solution has been developed for the linear stroke of the other actuator.

The wind turbine model is mounted upon the mechanism through a Ruag balance, that it is in turn bound to the mechanism itself by an appropriate support. The coupling between the wind turbine's tower and the balance is allowed by a shadow mask.

3.3.2 Measuring System

The measuring system includes Ruag and Ati balances, two accelerometers mounted on the wind turbine and the position control of the MTS actuator.

Rotational and linear degrees of freedom of the experimental set up are directly measured by the MTS hydraulic actuators control. Two mems accelerometers are mounted on the turbine model along X direction of floating wind turbine model's reference frame (figure 3.10).

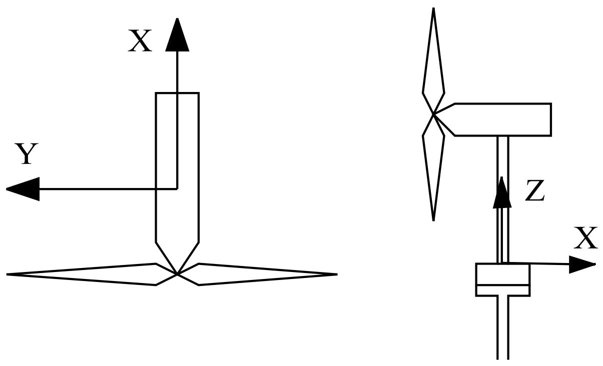


Figure 3.10: Wind turbine model reference frame

In tables 3.11a, 3.11b and 3.3 the main characteristics of the Ruag and Ati balances, the accelerometers and the MTS hydraulic actuators are reported.

Components	Load	$\epsilon_e/\epsilon_a[\%]$
F_X	$1500\,N$	88.24
F_Y	$1000\,N$	15.70
F_Z	$5000\,N$	20.78
M_X	$500\,Nm$	53.3
M_Y	$1000\,Nm$	36.59
M_Z	$600\,Nm$	84.74

(a) Ruag balance data sheet

	Single Axis Overload
$F_{XY}\,[N]$	± 5100
$F_Z\,[N]$	± 10000
$M_{XY}\,[Nm]$	± 110
$M_Z\,[Nm]$	± 140

(b) Ati-Ia balance mini 45 specifications

Figure 3.11: Balances characteristics

	Full Scale	Sensitivity
Accelerometers	$\pm 20\,gm/s^2$	$100\,mV/g$
MTS Hydraulic Actuator 1	$\pm 125\,mm$	$12.6\,mm/V$
MTS Hydraulic Actuator 2	$\pm 250\,mm$	$25.4\,mm/V$

Table 3.3: Devices data sheet

3.4 HIL Model

In this part of the chapter will be discussed how the numerical model presented in chapter 2 is implemented in the wave simulator mechanism for a real time experimental application.

Fig 3.12 shows the main features of the HIL simulation model.

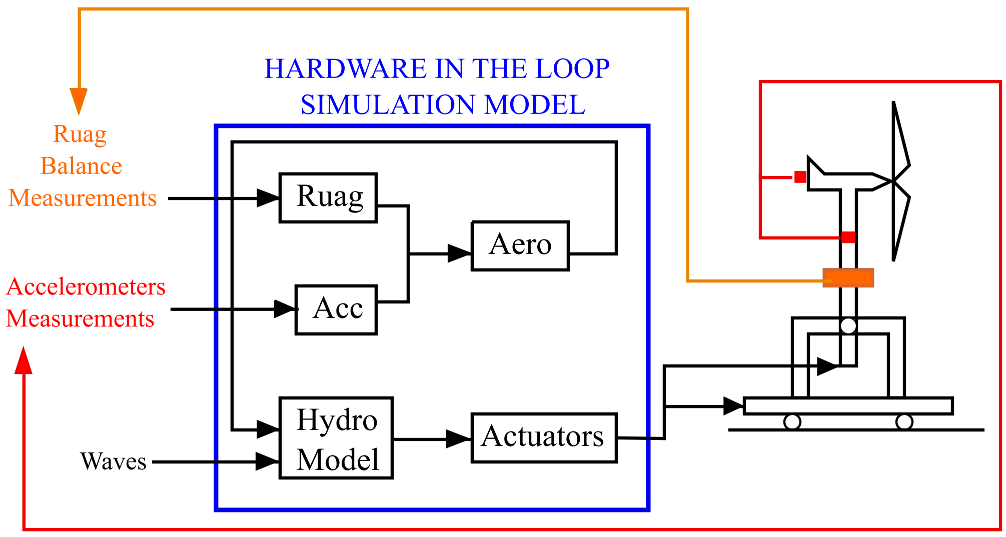


Figure 3.12: HIL simulation model scheme

As can be noted from Fig 3.12 , the plant simulation for the HIL model consists in five submodels:

1. *Hydro Model*
2. *Actuators*
3. *Faero*
4. *Ruag*
5. *Acc*

Hydro Model is basically the hydrodynamical model of chapter 2. As mentioned before, for the HIL model system, the memory effect must be represented by a state space model in order to make possible a real time application. Unlike the numerical model presented in chapter 2, platform DoFs displacements are now affected by aerodynamical loads, and it is necessary to add this term to the equation of motion:

$$[M]\ddot{\mathbf{x}} + [R]\dot{\mathbf{x}} + [K]\mathbf{x} + \mu = \mathbf{F}^{Waves} + \mathbf{F}^{Aero} \tag{3.4.1}$$

where $[M]$, $[R]$, $[K]$ represent the equivalent mass, damping and stiffness matrices, μ is the memory effect, $\mathbf{F}^{Waves}$ represent the diffraction and viscous forces and $\mathbf{F}^{Aero}$ is the aerodynamical force. *Hydro Model* has as inputs the wave state characteristics, necessary to build the $\mathbf{F}^{Waves}$ term, and the aerodynamic force acting on the wind turbine model referred to the platform's reference frame, that

in the scale model corresponds to the top of the Ruag balance. The outputs of this submodel are zero-mean surge and pitch displacements.
Actuators is a submodel which transforms the platform DoFs in the MTS actuators displacements.
Figure 3.13 and equations 3.4.2 show the kinematic link between surge and pitch DoFs and actuators displacements.

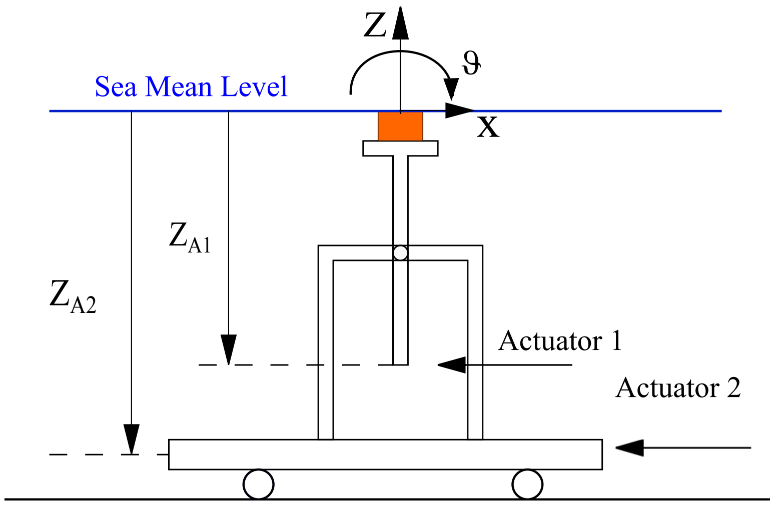


Figure 3.13: Kinematic relationship between surge and pitch DoFs and actuators displacements.

$$\begin{aligned} Actuator\,1 &= Z_{A1}\theta \\ Actuator\,2 &= -x + Z_{A2}\theta \end{aligned} \tag{3.4.2}$$

Also a notch filter is applied on the actuators signals in order to remove the spectral content in correspondence of the first natural frequency of the wind turbine scale model and be sure to avoid resonance excitation. The first natural circular frequency of the rigid model of wind turbine, provided from previous tests, corresponds to:

$$\omega_0^{Turbine} = 33.3\,[rad/s] \tag{3.4.3}$$

Faero submodel is meant to form the aerodynamic force $\mathbf{F}^{Aero}$ to introduce into *Hydro Model.*
As forces measured by the balances at tower's base and in the nacelle involve a not negligible contribute of inertial terms linked to the motion mechanism, signals have to be purified by this contribution in order to isolate aerodynamic loads.

$$\mathbf{F}^{Aero} = \left[F^{Aero}\, C^{Aero}\right]^T = \mathbf{F}^{Ruag} - \mathbf{F}^{Iner} \tag{3.4.4}$$

where $\mathbf{F}^{Ruag}$ is the force at tower base calculated by the Ruag balance. It represents the total force acting at tower base. $\mathbf{F}^{Iner}$ is the inertial force of the wind turbine model referred to the tower base.

Ruag is a submodel that transforms, through a calibration matrix $[K_{Ruag}]$, the seven tension acquisitions $\mathbf{T}$ of Ruag balance into the 3 forces and 3 moments components of $\mathbf{F}^{Ruag}$:

$$\mathbf{F}^{Ruag} = \left[K^{Ruag}\right]\mathbf{T} \tag{3.4.5}$$

These output values are deducted from their mean value in order to not consider the weight of the system, but only the dynamical loads. In particular only F_x and C_y are significant for this application, so only these two are considered.

$$\mathbf{F}^{Ruag} = \left\{ \begin{array}{c} F_x \\ C_y \end{array} \right\} \tag{3.4.6}$$

Acc is a submodel that transforms accelerometers acquisitions into the inertial force and moment of the wind turbine referred to tower base. This transformation is obtained by a calibration matrix:

$$\mathbf{F}^{Iner} = \left[M^{Iner}\right]\mathbf{A} \tag{3.4.7}$$

where $\mathbf{F}^{Iner} = [F^{Iner}\ C^{Iner}]^T$ is the inertial force, $\left[M^{Iner}\right]$ is the inertial calibration matrix and $\mathbf{A} = [A_1\ A_2]^T$ is the acceleration vector, containing the accelerometers acquisitions multiplied by its sensitivity. Also this term must be deducted from its mean value in order not to consider the gravity acceleration. $\left[M^{Iner}\right]$ terms, which links acceleration to the inertial forces of the wind turbine model referred to the tower base, are obtained through an experimental procedure, reported in the following.

3.4.1 Strumentation and System Configuration

For a real time application a controller is needed to interface the embedded plant system to the numerical model.

In this work Ni-PXI chassy was used with a FPGA integrated circuit, and Ni-Veristand as software environment for the configuration between plant and numerical model.

The complete HIL model, presented in section 3.3 of chapter 3, in order to be imported in Ni-Veristand had to be compiled in one, or more, dll file from simulink enviroment. The dll file works as “bitstream”, meaning it contains the information about how the FPGA gates should be wired together.

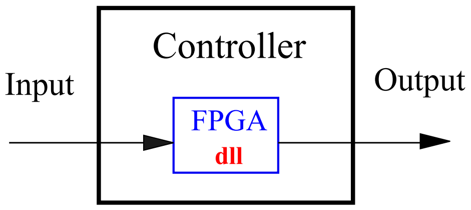


Figure 3.14: Plant controller scheme

In order to simulate in real time, due to PXI's limited skills, it was necessary to split *Hydro Model* in two or three submodels, depending wether the sea state was respectively regular or irregular. As a matter of fact, the irregular sea state is more computational consuming than regular case. Regular and irregular cases also differs for the control on wave elevation. For regular waves it was set the possibility to freely change the wave amplitude or frequency. For irregular waves it is not possible since dll files cannot deal with for-cycles, which are necessary for PSD_η generation. Therefore, for irregular waves it was configured the possibility to choose directly the wave elevation spectrum of a set of wave states loaded in the dll file. Fig 3.15 shows the irregular waves control in simulink enviroment.

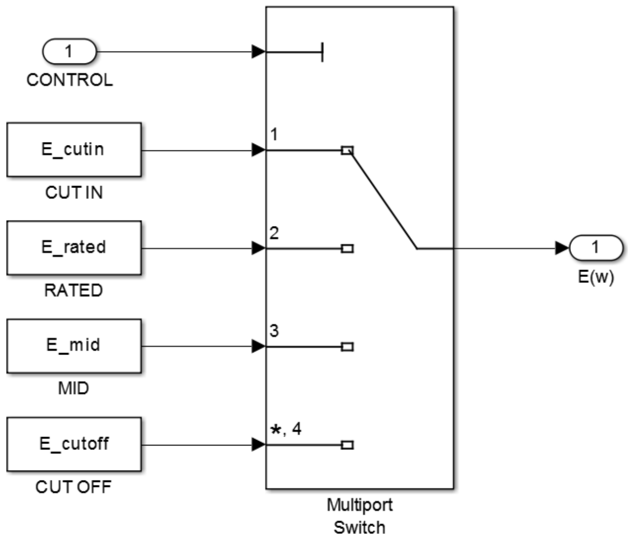


Figure 3.15: Irregular waves control in the plant simulation

All submodels were configured to use Bogaki-Shampine integration method, with a time step of *1 ms*. Once the dll time step is defined, in order to have a real time execution of the simulation, it is mandatory to set the PCL rate of the controller to the same value. In this case the PCL rate needed to be configured to *1000 Hz*.

3.4.2 Inertial Calibration Matrix

Experimental tests are needed to define $\left[M^{Iner}\right]$ terms. For this purpose two sets of tests were performed: surge only, for the translational motion, and pitch only, for the rotational motion.
Tests were carried out with different amplitudes of platform's surge and pitch at different frequencies in order to obtain a mass matrix $\left[M^{Iner}\right]$ that suits all the functioning range of the wave simulator.

amp [mm]	5, 10, 15
frequency [Hz]	0.25:0.25:2.5

(a) Surge tests

amp [mm]	15, 25
frequency [Hz]	0.25:0.25:2.5

(b) Pitch tests

Accelerometers were installed respectively at a distance from the Ruag balance of:

$$L_1 = 0.046\,m$$
$$L_2 = 1.790\,m$$

Translational motions tests (Fig 3.17a) aimed to obtain the equivalent mass and centre of mass (CM) position of the wind turbine model. Equations 3.4.8 and 3.4.9 represent the dynamical equilibrium of forces and moments in this configuration:

$$\begin{cases} F_{CM} = mA_{CM} = F^{Ruag} \\ A_1 = A_2 = A_{CM} \end{cases} \rightarrow m = \frac{F^{Ruag}}{A_{CM}} \tag{3.4.8}$$

$$F_{CM} b = C^{Ruag} \rightarrow b = \frac{C^{Ruag}}{F^{Ruag}} \tag{3.4.9}$$

On the other hand, rotational tests (Fig 3.17b) allowed to retrieve the translational and rotational accelerations of the model CM and also its equivalent inertia moment.
From the kinematic relation between rotational and translational acceleration of accelerometers it's possible to define the relation between the rotational acceleration of the turbine model CM and the measurements/signals from accelerometers. Also, the centre of mass of the wind turbine scale model acceleration can be related to its geometry and accelerometers signals (equations 3.4.10 and 3.4.11):

$$\ddot{\vartheta}\left(L_2 - L_1\right) = A_2 - A_1 \rightarrow \ddot{\vartheta} = \frac{A_2 - A_1}{L_2 - L_1} \tag{3.4.10}$$

$$A_{CM} = A_1 + \ddot{\vartheta}(b - L_1) = A_1 + \frac{A_2 - A_1}{L_2 - L_1}(b - L_1) \tag{3.4.11}$$

Inertia moment is retrieved from dynamical equilibrium of forces and moments, as it's shown equations 3.4.12 and 3.4.13:

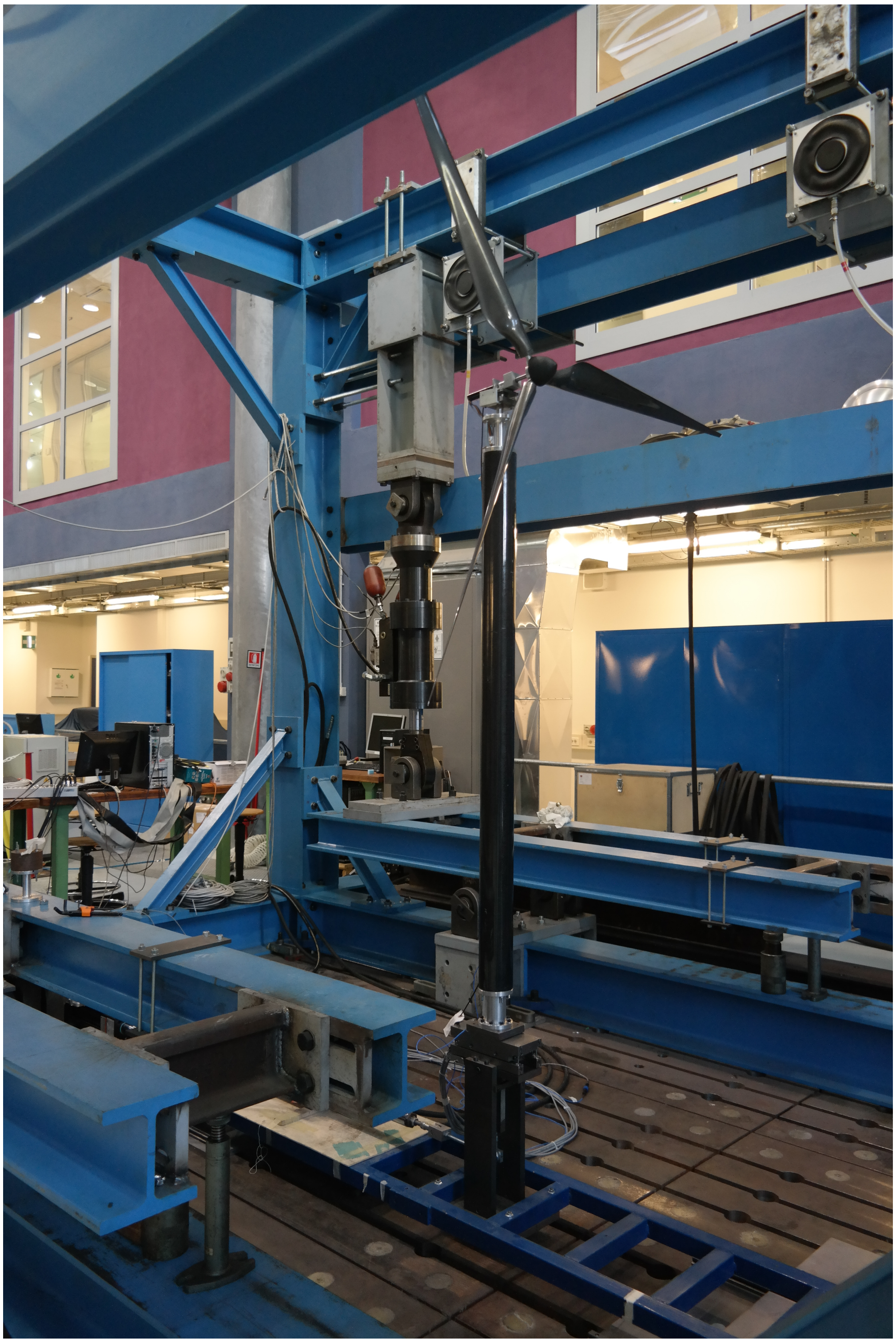
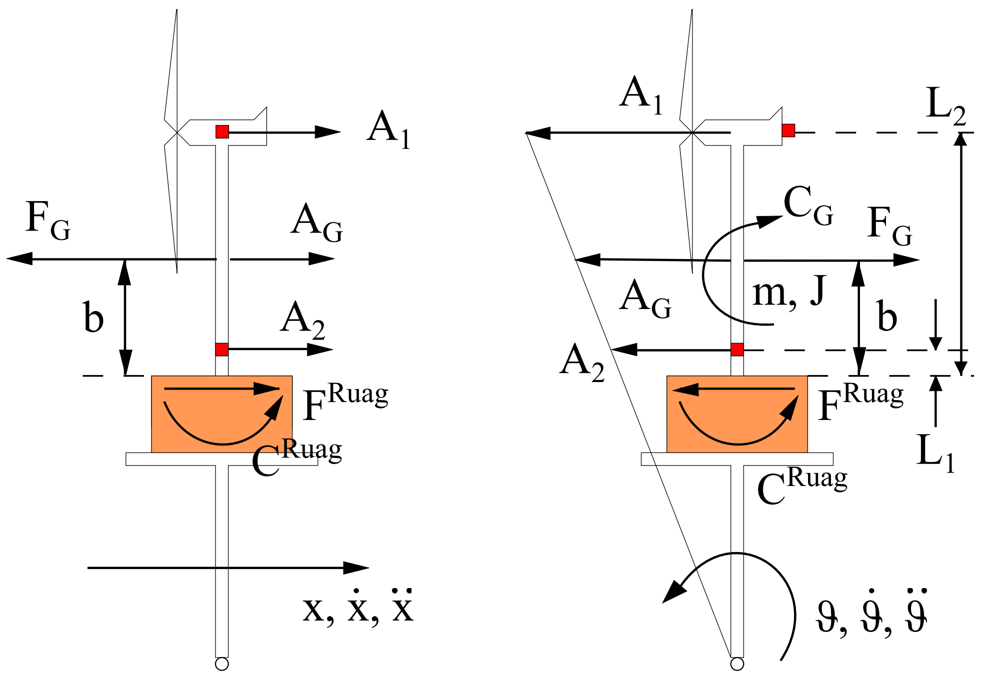


Figure 3.16: Experimental set up



(a) Translational motion (b) Rotational motion

Figure 3.17: Experimental tests for $[M^{Iner}]$ definition

$$F_{CM} = mA_{CM} = F^{Ruag} \tag{3.4.12}$$

$$J\ddot{\vartheta} + F_{CM}b = C^{Ruag} \rightarrow J = \frac{C^{Ruag} - F_{CM}b}{\ddot{\vartheta}} \tag{3.4.13}$$

The results obtained from the experimental campaign are shown in Fig 3.18 to 3.22.

In Fig 3.18, the amp_{nom} values represent:

$$A_{CM}^{nom} = amp(2\pi f)^2 \tag{3.4.14}$$

while the others are the mean value of the accelerometers acquisitions.

In Fig 3.21 the "*inertia*" values are obtained with $F_{CM} = mA_{CM}$, with A_{CM} defined as $A_1 + \frac{A_2 - A_1}{L_2 - L_1}(b - L_1)$, and the *"measured"* values are obtained from the Ruag balance acquisition.

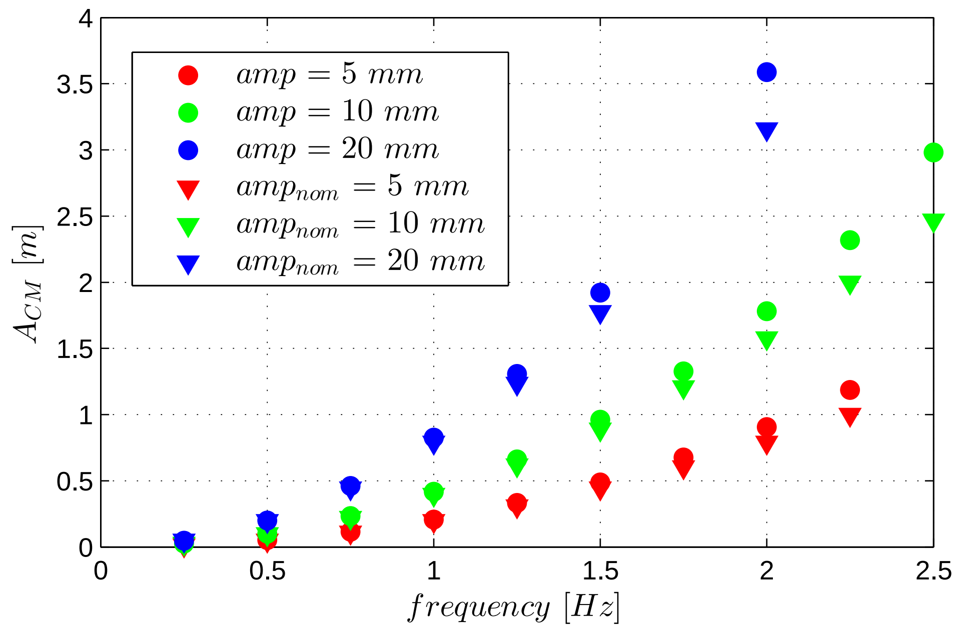


Figure 3.18: Rigid model centre of mass acceleration in surge tests.

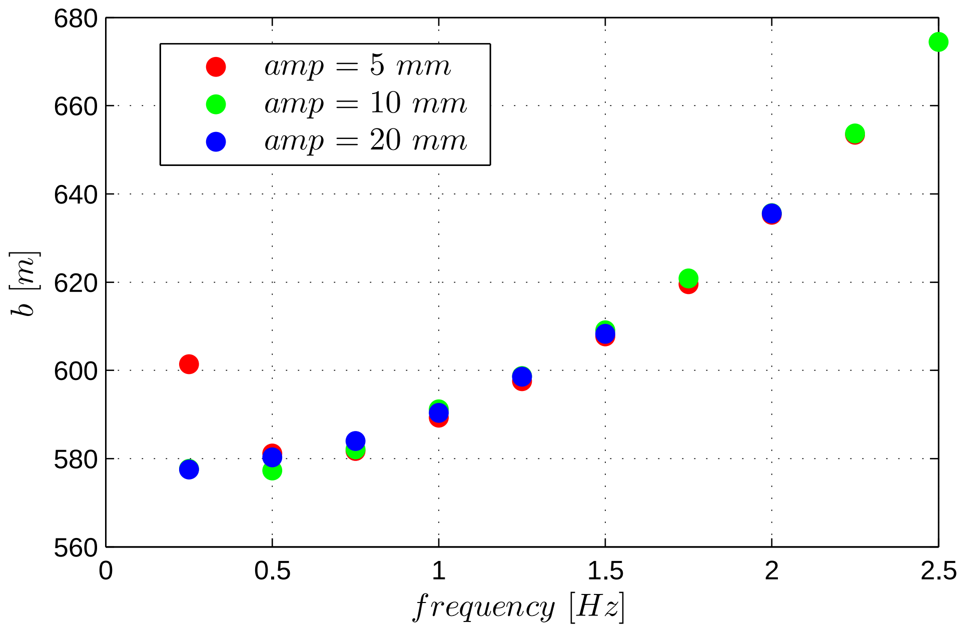


Figure 3.19: Distance of model centre of mass from the top of the Ruag balance.

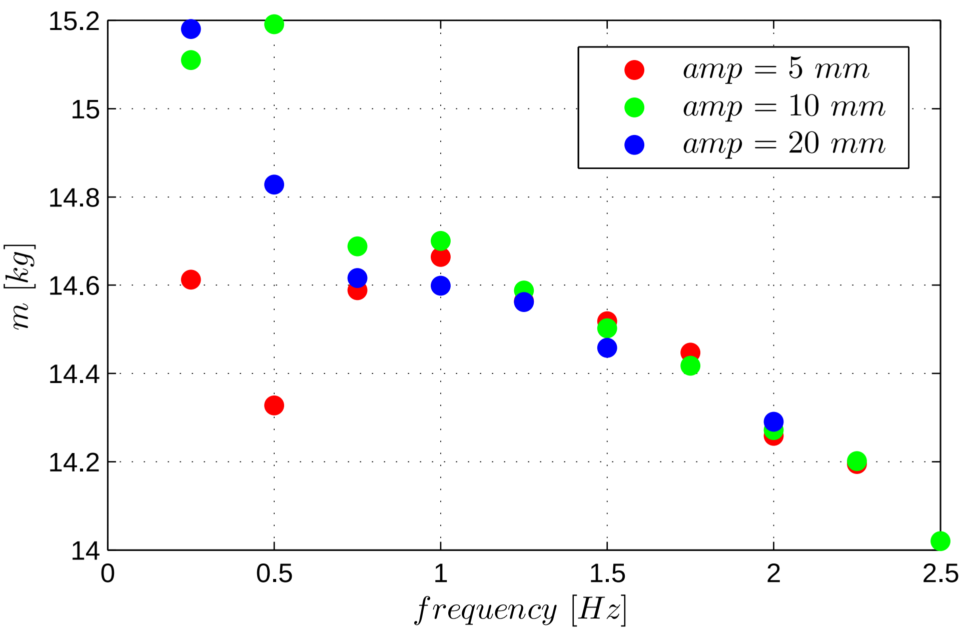


Figure 3.20: Mass of the rigid model

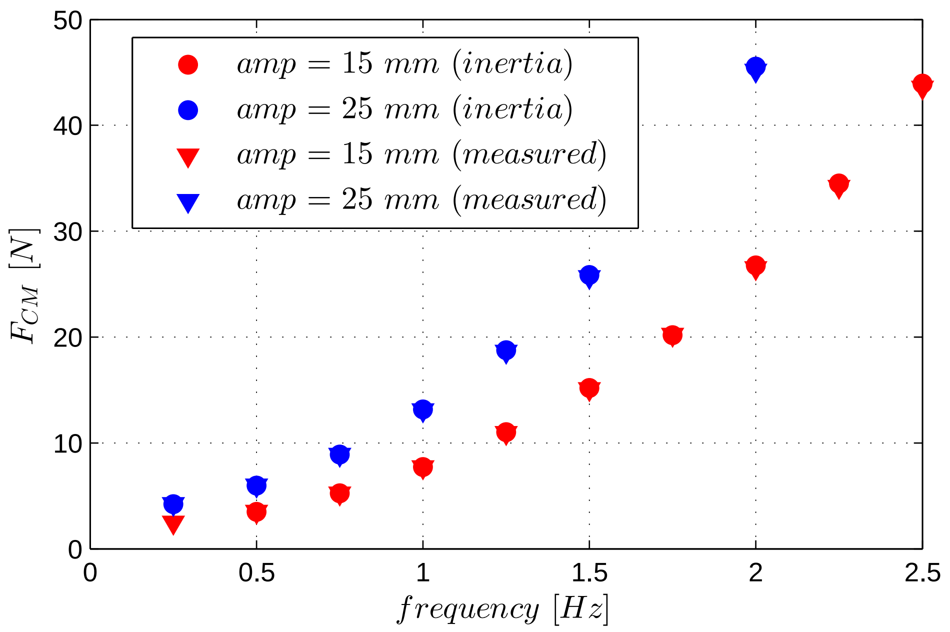


Figure 3.21: Force at model centre of mass calculated in pitch tests.

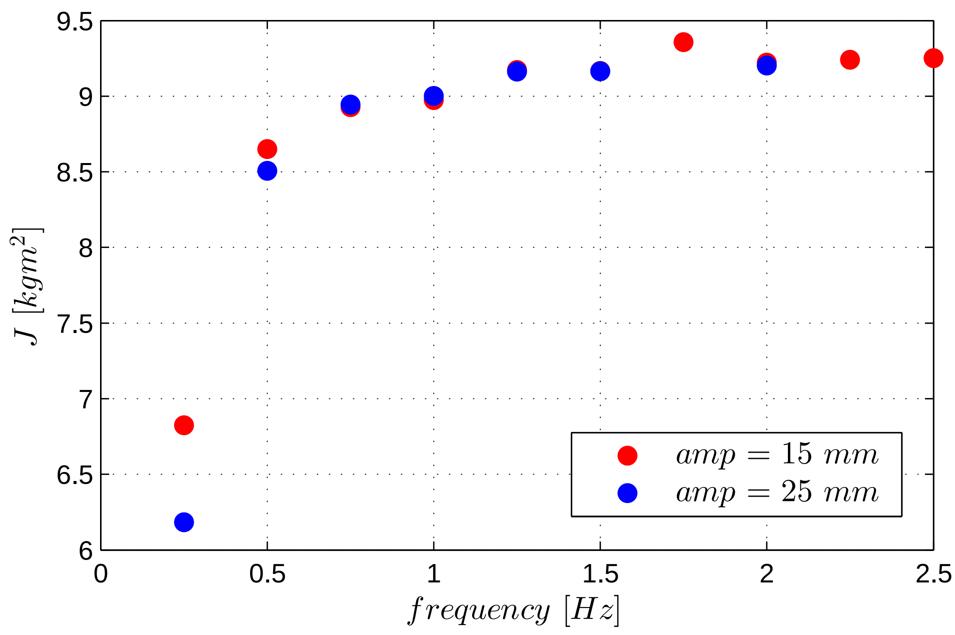


Figure 3.22: Model's inertia moment.

Then, once m and J are defined, it's possible to define the inertial mass matrix $\left[M^{Iner}\right]$ which relates the signals from accelerometers with the inertial force and moment at tower base:

$$\begin{cases} F^{Iner} = mA_{CM} = m\left(A_1 + \frac{A_2-A_1}{L_2-L_1}(b-L_1)\right) = \hat{m}_{11}A_1 + \hat{m}_{12}A_2 \\ C^{Iner} = J\ddot{\vartheta} + mA_{CM}b = J\frac{A_2-A_1}{L_2-L_1} + m\left(A_1 + \frac{A_2-A_1}{L_2-L_1}(b-L_1)\right)b = \hat{m}_{21}A_1 + \hat{m}_{22}A_2 \end{cases} \tag{3.4.15}$$

where $\hat{m}_{ij}$ are defined as follows:

$$\begin{aligned} \hat{m}_{11} &= m\left(1 - \frac{b-L_1}{L_2-L_1}\right) \\ \hat{m}_{12} &= m\frac{b-L_1}{L_2-L_1} \\ \hat{m}_{21} &= \hat{m}_{11} - \frac{J}{L_2-L_1} \\ \hat{m}_{22} &= \hat{m}_{12} + \frac{J}{L_2-L_1} \end{aligned} \tag{3.4.16}$$

In table 3.4 and figures are presented the results of these previous experimental tests.

m $[kg]$	14.55
b $[m]$	0.606
J $[kgm^2]$	8.76
$\hat{m}_{11}$ $[kg]$	9.99
$\hat{m}_{12}$ $[kgm]$	4.75
$\hat{m}_{21}$ $[kgm]$	0.083
$\hat{m}_{22}$ $[kgm^2]$	8.3

Table 3.4: Previous experimental campaign results

Chapter 4

Conclusions and Future Development

In this work a numerical model for the hydrodynamics of floating wind turbines was developed.
The comparison with FAST simulator, that must be considered the main reference in this field, shows a consistent matching between the two codes, both in time and frequency domain, and also from energy content point of view.
All the effort put in the modeling of the radiation memory effect, in order to avoid the convolution representation, led to the creation of a tool able to transform this term in a simpler state space model. This tool is to be considered a simplified version of the Perez&Fossen "FDI Toolbox", and can also be used for other hydrodynamical applications.
Concerning the numerical code, several improvements need to be done in order to get closer to all the potentialities of FAST.
From the hydrodynamical point of view, the model should be augmented to all the six degrees of freedom and a more sophisticated model for the mooring lines dynamics should be added. Whereas the aerodynamics were not considered in the development of the code, even though they are fundamental for the power generation of the wind turbine. A dedicated subroutine should be implemented, with the possibility to solve also the aeroelasticity of tower and blades.
The second part of the work consisted in the implementation of the code in a two degrees of freedom mechanism for a real time "hardware-in-the-loop" simulation.
In this part the numerical code needed to be modified and imported in Simulink enviroment. Also a dedicated wave elevation control configuration in the simulation plant was developed in order to vary the sea state during the real time application. Different submodels were added for the modeling of the embedded system, in order to control measurement chain's acquisitions and hydraulic actuators.
Also an experimental campaign was performed for the calibration of the inertial matrix $[M^{Iner}]$, necessary for aerodynamic load acquisition and implementation

in the numerical code.
An experimental campaign, in order to test both regular and irregular sea states, will be necessary for the validation of the real time “hardware-in-the-loop” model. Once will be validated, this experimental rig will be a remarkable tool for the recreation of spar-buoy floating wind turbine functioning. For the porpoise of testing aeroelastic models of wind turbine, which is the future aim of this project, will be also necessary to develop a more sophisticated data processing for the acquisition, and implementation in the numerical model, of the aerodynamical loads. At this point the experimental rig will offer the possibility to study and develop strategies for pitch and rotational speed control in order to overcome the loss of power generated during these operating conditions.

Bibliography

[1] ILMAS ANDREA BAYATI. Design and validation of a hardware in the loop experimental rig for wind tunnel tests on offshore wind turbines. 2011.

[2] A Betz. *Introduction to the Theory of Flow Machines.* Pergamon Press., 1966.

[3] Paola Bresesti, Wil L Kling, Ralph L Hendriks, and Riccardo Vailati. Hvdc connection of offshore wind farms to the transmission system. *Energy Conversion, IEEE Transactions on*, 22(1):37–43, 2007.

[4] Edgar Buckingham. The principle of similitude. *Nature*, 96:396–397, 1915.

[5] Xia Changliang and Song Zhanfeng. Wind energy in china: current scenario and future perspectives. *Renewable and Sustainable Energy Reviews*, 13(8):1966–1974, 2009.

[6] International Electrotechnical Commission et al. Iec 61400-3. *Wind TurbinesÑPart 3: Design Requirements for Offshore Wind Turbines*, 2009.

[7] Global Wind Energy Council. Global wind report, annual market update, 2011.

[8] WE Cummins. The impulse response function and ship motions. Technical report, DTIC Document, 1962.

[9] Leon L Freris. *Wind energy conversion systems.* Prentice Hall, 1990.

[10] Phyllis Heronemus and William Heronemus. Offshore wind turbine, 2003.

[11] J Jonkman, Sandy Butterfield, P Passon, T Larsen, T Camp, James Nichols, J Azcona, and Alfredo Martinez. Offshore code comparison collaboration within iea wind annex xxiii: phase ii results regarding monopile foundation modeling. In *2007 European Offshore Wind Conference & Exhibition, 4–6 December 2007, Berlin, Germany*, 2007.

[12] Jason M Jonkman. The new modularization framework for the fast wind turbine cae tool. In *51st AIAA Aerospace Sciences Meeting and 31st ASME Wind Energy Symposium, Grapevine, Texas*, 2013.

[13] Jason M Jonkman and Marshall L Buhl Jr. Fast userÕs guide. *Golden, CO: National Renewable Energy Laboratory*, 2005.

[14] Jason Mark Jonkman. *Dynamics modeling and loads analysis of an offshore floating wind turbine.* ProQuest, 2007.

[15] Jason Mark Jonkman. *Definition of the Floating System for Phase IV of OC3.* National Renewable Energy Laboratory, 2010.

[16] Jason Mark Jonkman, Sandy Butterfield, Walter Musial, and G Scott. *Definition of a 5-MW reference wind turbine for offshore system development.* National Renewable Energy Laboratory Golden, CO, 2009.

[17] Jason Mark Jonkman and Denis Matha. *A quantitative comparison of the responses of three floating platforms.* National Renewable Energy Laboratory, 2010.

[18] JMJ Journe and WW Massie. *Offshore hydromechanics.* TU Delft, 2000.

[19] JMJ Journée and WW Massie. *Offshore hydromechanics.* TU Delft, 2000.

[20] CG Justus, WR Hargraves, Amir Mikhail, and Denise Graber. Methods for estimating wind speed frequency distributions. *Journal of applied meteorology*, 17(3):350–353, 1978.

[21] Jacob Ladenburg. Attitudes towards on-land and offshore wind power development in denmark; choice of development strategy. *Renewable Energy*, 33(1):111–118, 2008.

[22] Chang-Ho Lee and J Nicholas Newman. Wamit user manual. *Dept. of Ocean Engineering, Massachusetts Institute of Technology, Cambridge, MA*, 1999.

[23] Chenyu Luan, Zhen Gao, and Torgeir Moan. Modelling and analysis of a semi-submersible wind turbine with a central tower with emphasis on the brace system. In *ASME 2013 32nd International Conference on Ocean, Offshore and Arctic Engineering*, pages V008T09A024–V008T09A024. American Society of Mechanical Engineers, 2013.

[24] Tristan Perez and Thor I Fossen. Practical aspects of frequency-domain identification of dynamic models of marine structures from hydrodynamic data. *Ocean Engineering*, 38(2):426–435, 2011.

[25] Susan L Sakmar. *Energy for the 21st Century: Opportunities and Challenges for Liquefied Natural Gas (LNG).* Edward Elgar Publishing, 2013.

[26] Irving Herman Shames and Irving Herman Shames. *Mechanics of fluids.* McGraw-Hill New York, 1982.

[27] GJW Van Bussel and MB Zaaijer. Reliability, availability and maintenance aspects of large-scale offshore wind farms, a concepts study. In *Proceedings of MAREC*, volume 2001, 2001.

[28] Fabian Vorpahl, Wojciech Popko, and Daniel Kaufer. Description of a basic model of the" upwind reference jacket" for code comparison in the oc4 project under iea wind annex xxx. *Fraunhofer Institute for Wind Energy and Energy System Technology (IWES)*, 4, 2011.

Appendix A

Appendix A

In this appendix will be presented a matlab code that aims to fit a parametric model of the retardation function. Once the retardation function is found the code creates an equivalent state space model that could be used in equation 2.2.1 in order to replace the memory effect's convolution integral.
The code is structurated in four matlab functions:

1. *ssmarco.m*
2. *marcofit.m*
3. *MakeStable.m*
4. *assembling.m*

A.0.3 *ssmarco.m*

ssmarco.m returns the parametric form of the retardation function and its state space model representation for the entry selceted by the user. Also a maximum order of the transfer function must be insert. *ssmarco.m* uses *marcofit.m* iteratively until the accuracy level of the fitting is achived. This threshold has been set at 99%.

```matlab
function[Kw_hat,Ar,Br,Cr] = ssmarco(dof,ordermax);

%% Wamit Data Loading
% Sea Water density rho = 1025;
% Wamit frequency discretization
dw = 0.05;
w_max = 5;
Nw = w_max/dw;
w = dw*(0:Nw)';
data = load('HydroData/spar.1');

%% Pick the Dof
if dof(1) == 1 && dof(2) == 1
    aa = 11;
```

```
elseif dof(1) == 1 && dof(2) == 2
    aa = 12;
elseif dof(1) == 2 && dof(2) == 1
    aa = 18;
elseif dof(1) == 2 && dof(2) == 2
    aa = 19;
else     disp('enter new dof')
    return
end

%% Added Mass and Damping Matrixes
Ainf = data(aa,4)*rho;
A = data(aa:10:end,4)*rho;
B = data(aa:10:end,5)*rho.*w;
% Increase w, A and B values via interpolation in order
% to increase interpolation accuracy. Discretization must be
  divisible by 2
% in order to get the best match.
dwi = w_max/(2^8-1);
wi = dwi*(0:2^8-1)';
Aint = interp1(w,A,wi);
Bint = interp1(w,B,wi);
clear A
clear B
A(:,1) = Aint(2:end);
B(:,1) = Bint(2:end);
W(:,1) = wi(2:end);

% Retardation Function
Kw = B+complex(0,W).*(A-Ainf*ones(size(A)));

%% Inizialization
% marcofitt.m finds the parametric form of numerator and
  denominator
% polynomials retardation function Kw in W range starting from a
  defined
% order and executing iter iteractions in order to obtain the
  best fit.
% 2 is the minimun order of the retardation function based on
  SS_Fitting
% theory
order = 2;
iter = 20;
[P,Q]=marcofitt(W,Kw,order,iter);
Kw_hat=freqs(P,Q,W);

% The quality of the model is valued based on R^2 values of A_hat
  and B_hat
% which are the added and damping matrixes obtained with Kw_hat.
Brecfd = real(Kw_hat);
Arecfd = imag(Kw_hat)./W+Ainf*ones(size(W));
SSEB =
(B-Brecfd)'*(B-Brecfd);
SSTB =(B-mean(B)*ones(size(B)))'*(B-mean(B)*ones(size(B)));
```

```
R2B = 1 - SSEB/SSTB;
SSEA = (A-Arecfd)'*(A-Arecfd);
SSTA =(A-mean(A)*ones(size(A)))'*(A-mean(A)*ones(size(A)));
R2A = 1 - SSEA/SSTA;

% Accuracy Target   r2Thres = 0.99;
% marcofit.m runs in a loop until r2Thres is obtaoned
while  (R2B<=r2Thres) && (R2A<=r2Thres),
    order = order +1;
    if order> ordermax,
        break
    end
    [P,Q]=marcofitt(W,Kw,order,iter);
    Kw_hat=freqs(P,Q,W);
    Brecfd = real(Kw_hat);
    Arecfd = imag(Kw_hat)./W+Ainf*ones(size(W));
    SSEB = (B-Brecfd)'*(B-Brecfd);
    SSTB =(B-mean(B)*ones(size(B)))'*(B-mean(B)*ones(size(B)));
    R2B = 1 - SSEB/SSTB;
    SSEA = (A-Arecfd)'*(A-Arecfd);
    SSTA =(A-mean(A)*ones(size(A)))'*(A-mean(A)*ones(size(A)));
    R2A = 1 - SSEA/SSTA; end

%% Comparison by Plot.
figure
subplot(211)
plot(W,abs(Kw),W,abs(Kw_hat),'LineWidth',2)
grid on
xlabel('w')
ylabel('|K(w)|')
legend('wamit','fitting')
subplot(212)
plot(W,angle(Kw)*180/pi,W,angle(Kw_hat)*180/pi,'LineWidth',2)
grid on
xlabel('w')
ylabel('angle(K(w))')
legend('wamit','fitting')

%% State Space Trasformation
[Ar,Br,Cr] = tf2ss(P,Q);
end
```

A.0.4 *marcofit.m*

marcofit.m takes for input the retardation function and its frequency discretization and returns denominator and numerator terms of the parametric retardation form. Also the starting order of the parametric function and maximum number of iterations must be defined. *marcofit.m* uses *MakeStable.m* in order to guarantee negative real part of retardation function poles.

```
function [P,Q]=marcofitt(W,Kw,order,iter)

```

```
% marcofitt.m finds the parametric form of numerator and
  denominator
% polynomials retardation function Kw in W range starting from a
  defined
% order and executing iter iteractions in order to obtain the
  best fit.

% In first place a normalization of Kw(w) respect to highest
  value
% has to be done in order to improve interpolation quality
alfa=max(abs(Kw));
Kws = Kw/alfa;
Fresp= Kws./complex(0,W);

%Frequency response fitting
ord_den = order;
ord_num = order-2; % because p0 is added later weight=ones(size(W
  ));
for k=1:iter,
    [P,Q]=invfreqs(Fresp,W,ord_num,ord_den,weight);
    % MakeStable chages the sign of the Q poles
    % if they have positive real part.
    Q = MakeStable(Q);
    % weigths assume (Q values)^2 of the previous iteraction
    for m=1:length(Q);
        weight(m)=1/abs(polyval(Q,complex(0,W(m))))^2;
    end
end

% Rescale and incorporate the zero P=alfa*[P 0];
end
```

A.0.5 *MakeStable.m*

MakeStable.m cheks if a polynomial has roots in the closed right half plane, and if so, the unstable roots are reflected about the immaginary axis.

```
function out=MakeStable(p)

% This function checks if a polynomial has roots in the
% closed right half plane, and if so, the unstable roots are
% reflected about the imaginary axis.
r=roots(p);
for k=1:length(r)
    if real(r(k)) > 0,
    r(k) = -r(k);
end
end

out=poly(r);
```

A.0.6 *assembling.m*

assembling.m is a code that deals with matrix assemblation. It takes $[A_r]$, $[B_r]$ and $[C_r]$ of each entry and assemble them according to 2.3.16, 2.3.17 and 2.3.18. For this case only the surge-surge, surge-pitch, pitch-surge and pitch-pitch entries are taken in consideration and assumed that they are saved in a folder of the same name.

```
% Load State Space Matrix
addpath C:\......\surge_surge\ load Ar Arss = Ar;
clear Ar
load Br
Brss = Br;
clear Br
load
Cr
Crss = Cr;
clear Cr
rmpath  C:\......\surge_surge\
addpath C:\......\surge_pitch\
load Ar
Arsp = Ar;
clear Ar
load Br
Brsp = Br;
clear Br
load Cr
Crsp = Cr;
clear Cr
rmpath  C:\.......\surge_pitch\
addpath C:\.......\pitch_surge\
load Ar
Arps = Ar;
clear Ar
load Br
Brps = Br;
clear Br
load Cr
Crps = Cr;
clear Cr
rmpath  C:\......\pitch_surge\
addpath C:\......\pitch_pitch\
load Ar
Arpp = Ar;
clear Ar
load Br
Brpp = Br;
clear Br
load Cr
Crpp = Cr;
clear Cr
rmpath  C:\......\pitch_pitch\

% Ar Assembling
Arsize = size(Arss) + size(Arsp) + size(Arps) + size(Arpp);
l_Arss = length(Arss); l_Arsp = length(Arsp);
l_Arps = length(Arps);
l_Arpp = length(Arpp);
Ar = zeros(Arsize);
Ar(1:l_Arss,1:l_Arss) = Arss;
Ar(l_Arss+1:l_Arss+l_Arsp,l_Arss+1:l_Arss+l_Arsp) = Arsp;
Ar(l_Arss+l_Arsp+1:l_Arss+l_Arsp+l_Arps,l_Arss+l_Arsp+1:l_Arss+
  l_Arsp+l_Arps) = Arps;
Ar(Arsize-l_Arpp+1:Arsize,Arsize-l_Arpp+1:Arsize) = Arpp;

% poles check poles = eig(Ar);
% Br Assembling
Br = zeros(length(Ar),2); Br(1:l_Arss,1) = Brss;
Br(l_Arss+1:l_Arss+l_Arsp,1) = Brsp;
Br(l_Arss+l_Arsp+1:l_Arss+l_Arsp+l_Arps,l_Arss+l_Arsp+1,2) = Brps
  ;
Br(Arsize-l_Arpp+1:Arsize,2) = Brpp;

% Cr Assembling
Cr = zeros(2,length(Ar));
Cr(1,1:l_Arss) = Crss;
Cr(2,l_Arss+1:l_Arss+l_Arsp) = Crsp;
Cr(1,l_Arss+l_Arsp+1:l_Arss+l) = Crps;
Cr(2,Arsize-l_Arpp+1:Arsize) = Crpp;
```

Appendix B

Appendix B

In this appendix the input file and the different matlab functions that costitute the numerical model described in section 3 of chapter 2 will be presented.

B.0.7 Input File

This script contains the geometrical properties of the system and let the User to choose type of wave state and intensity. Also integration parameters of the simulation must be defined.

```
%% File Input

%% Geometry Part

%% Plaform
% Mass
Mpl = 7466330;
% Inertia Momentum
Jpl = 4229230e3;
% Top Diameter
Dup = 6.5;
% Bottom Diameter
Ddown = 9.4;
% Top heigth
h1 = 14;
% Mid heigth
h2 = 8;
% Bottom height
h3 = 108;
% Centre of Mass
CMpl = [0 30.0900];
% Drag Coefficient
c_drag = 0.6;
% Radiation and Diffraction Data
% Select the path to the file containing radiation and
  diffraction data
data_diffraction = load('HydroData\diffractionfile.txt');
data_radiation = load('HydroData\radiationfile.txt');
```

```
% Memory Effect
rad_flag = 2;        % State Space = 1; Convolution = 2;
% Radiation Time for Convolution Model
Trad = 60;
% Initial Condition od Surge and Pitch
V0surge = 0; V0pitch = 0; X0surge = 0; X0pitch = 0;
%% Tower
% Mass
Mto = 249718;
% heigth
hto = 87.6;
% Centre of Mass
CMto = [0 33.4];
%% Nacelle
% Mass
Mna = 240E3;
% Centre of Mass
CMna = [1.9,80];
%% Rotor
% Mass
Mro = 110E3;
% Centre of Mass
CMto = [-5.01910,80];
%% Mooring Lines
% Preload
Flines0 = 1607e3;
% Linearized Mooring Lines Matrix
Clines = [  41180 -2821e3;
          -2816e3 311.1e6];
%% Additional Damping
% Matrix
Bad = [1e5 0;0 0];




%% Wave

%% Wave Elevation
% Regular = 1; Irregular = 2;
flageta = 2;
% Regular case
    amplitude = 2;
    frequency= 0.5;
    phase = 1;
% Irregular case
    ww_stop = 1.665;
    Hs = 5.49;
    Tp = 11.3;
% Wave direction
beta0 = 0;


%% Control
%% Time
% Discretization
dt = 0.01;
% Duration of the Simulation
T = 1000;
%% Frequency
% Discretization
dw = 0.005;
%% Solver
% ode45: solver = 1; % ode4: solver = 2;
solver = 1;
```

B.0.8 *Hydrostatics.m*

Hydrostatics.m is a function that returns the equilibrium position of the system in still water. Also mass matrix and hydrostatic restoring matrix are calculated as function of the equilibrium position.

```
function [ output ] = hydrostatics( geometry )
% Global Parameters
rho = 1025; g = 9.80665;

% Masses
Mpl = geometry.platform.mass;
Jpl = geometry.platform.inertia;
Mto = geometry.tower.mass;
Mna = geometry.nacelle.mass;
Mro = geometry.rotor.mass;
Mtot = Mpl + Mto + Mna + Mro;

% Platform Geometry
Dup = geometry.platform.topdiameter;
Ddown = geometry.platform.bottomdiameter;
Aup = pi/4*Dup^2; Adown = pi/4*Ddown^2;
h1 = geometry.platform.height3;
h2 = geometry.platform.height2;
h3 = geometry.platform.height1;
Vpl1 = Aup*h1;
Vpl2 = (pi/12)*h2*(Dup^2+Ddown^2+Ddown*Dup);
Vpl3 = Adown*h3;
Vpl = Vpl1+Vpl2+Vpl3;

% Tower Geometry
hto = geometry.tower.heigth;
Jto = Mto*hto^2/12;

% CM
CMpl = geometry.platform.CM;
CMto = geometry.tower.CM;
CMna = geometry.nacelle.CM;
CMro = geometry.rotor.CM;

```

```
% Mooring Lines Preload;
Flines0 = geometry.mooring.preload;

% Archimede's Principle
Farch = Flines0 + Mtot*g;
V0 = Farch/(rho*g);
% Distance of the top of the platform from slm
hslm = round((Vpl-V0)/Aup);

% COB
% Referemce Frame is on slm and platform axis
% Submerged Water
Mw1 = rho*Aup*(h1-hslm); Mw2 = rho*Vpl2; Mw3 = rho*Vpl3;
Mw = rho*V0;

% Definition: Mw*ZCOB = Mw1*ZCOB1+Mw2*ZCOB2+Mw3*ZCOB3
ZCOB1 = -(h1-hslm)/2;
ang_coef = (Ddown-Dup)/h2;
ZCOB2 = -(h1-hslm) - 1/Vpl2*pi*(((Dup/2)^2)*h2^2/2 + ang_coef^2*
  h2^4/4 +...
             2*ang_coef*(Dup/2)*h2^3/3);
              % The second term is the integral of x*pi*r^2 between
                 0 and h2.
              %  x =  Dup/2 + coef_ang*x.
              % integral( x*pi*(Dup/2^2 + ang_coef*x)^2 )dx between
                 0 e h2.
ZCOB3 = -(h1-hslm) - h2 - h3/2;
ZCOB = 1/Mw*( Mw1*ZCOB1 + Mw2*ZCOB2 + Mw3*ZCOB3 );

% Equilibrium Position
% Distance System CMs from reference frame
draft = abs(hslm-(h1+h2+h3)); Z
pl = -(draft-hslm-CMpl(2));
Zto = CMto(2)+hslm;
Zna = CMna(2)+hslm;
Zro = CMro(2)+hslm;
Xro = CMro(1);
Xna = CMna(1);
Rna = sqrt(Xna^2+Zna^2);
Rro = sqrt(Xro^2+Zro^2);
alfa_ro = atan(Xro/Zna);
alfa_na = atan(Xna/Zna);

% Equilibrium Position is found by a equilibrium of momemnts:
% Equilibrium Position of Surge is assumed 0.
deno = (Farch - Mpl*g*Zpl + Mto*g*Zto + Mna*g*Rna*cos(alfa_na)...
         + Mro*g*Rro*cos(alfa_ro));
nume = -(Mna*g*Rna*sin(alfa_na) + Mro*g*Rro*sin(alfa_ro));
theta0 = atan(nume/deno); X0 = [0 theta0];

%% Linear Hydrostatic Restoring Matrix
J0 = pi/64*Dup^4;
Chs(2,2) = rho*g*J0 + ZCOB*Farch - g*(Zpl*Mpl + Zto*Mto + Rna*Mna
   + Rro*Mro);
```

```

%% System Mass Matrix
Msys(1,1) = Mtot;
Msys(1,2) = Mpl*Zpl + Mto*Zto + (Mna+Mro)*Zna; Msys(2,1) = Msys
  (1,2);
Msys(2,2) = Jpl + Jto + Zpl^2*Mpl + Zto^2*Mto + Zna^2*(Mna+Mro);

output = struct('X0',[],'Chs',[],'Msys',[],'draft',[]);
output.X0 = X0;
output.Chs = Chs;
output.Msys = Msys;
output.draft = draft;

end
```

B.0.9 *WaveGenerator.m*

WaveGenerator.m is a function that creates the wave elevation and current velocity from the data provided by User in the input file.

```
function [ output ] = wavegenerator( waveopt,geometry,hsoutput,
  control )

flag = waveopt.type;
memoryflag = geometry.platform.radiationstru.memoryrep;

% flag = 1: regular case
% flag = 2; irregular case

dt = control.disctime; T = control.timeend; t = 0:dt:T;
dw = control.discfreq; w = 0:dw:5;

% Frequency
dw_op = .005;
wmax_op = 5;
w_op = 0:dw_op:5;
Nw = wmax_op/dw_op;

% Time
dt_op = 0.01;
T_op = 2*pi/dw_op;
t_op = 0:dt_op:T_op;
Nt = T_op/dt_op;

% Global Parameters
g = 9.81;
% Platform Geometry
draft = hsoutput.draft;
dz = geometry.platform.dz;
z = -(dz/2:dz:draft-dz/2)';

% Platform Geometry
draft = hsoutput.draft;
```

```
dz = geometry.platform.dz;
z = -(dz/2:dz:draft-dz/2)';

if flag == 1,
    %% REGULAR
    % Wave Elevation
    % time domain
    eta_a = waveopt.eta_a;
    w_eta = waveopt.w_eta;
    phi_eta = waveopt.phi;
    eta_fun = @(t) eta_a*cos(w_eta*t + phi_eta);
    eta = eta_fun(t);
    eta_op = eta_a*cos(w_eta*t_op + phi_eta);
    % frequency domain
    E2 = fft(eta_op)/Nt;
    E_op(1) = E2(1);
    E_op(2:Nw+1) = 2*E2(2:Nw+1);
    E = interp1(w_op,E_op,w);
    % Wave Velocity
    k = w_eta^2/g;
    f_z = exp(k*z);
    u = @(t) w_eta*f_z*eta_fun(t);
    if memoryflag == 2;
        for i = 1:length(t)
             u_op(:,i) = u(t(i));
        end
        u_sim = timeseries(u_op,t);
        output = struct('eta',[],'E',[],'u',[],'u_sim',[]);
        output.eta = eta;
        output.E = E;
        output.u = u;
        output.u_sim = u_sim;
    else
        output = struct('eta',[],'E',[],'u',[]);
        output.eta = eta;
        output.E = E;
        output.u = u;
    end
end

if flag == 2
    %% IRREGULAR
    % Wave Elevation
    % PSD construction
    Hs = waveopt.sigheight;
    Tp = waveopt.period;
    w_stop = waveopt.cutfreq;
    PSD  = jonswap( w, Hs, Tp, w_stop );
    WGN  = boxmuller( w );
    % frequency domain
    eta_a = sqrt(2*dw_op*PSD);
    E_op = eta_a.*WGN;
    E_abs = abs(E_op);
    E_angle = angle(E_op);
    E = interp1(w_op,E_op,w);
    eta_w_fun = @(t) E_abs.*cos(w*t+E_angle);
    eta_fun = @(t) sum( E_abs.*cos(w*t+E_angle) );
    for it = 1:length(t)
        eta(it) = eta_fun(t(it));
    end
    % Wave Velocity
    k = w.^2/g;
    f_z = exp(z*k);
    u = @(t) f_z*(w.*eta_w_fun(t))';
    if memoryflag == 2;
        for i = 1:length(t)
            u_op(:,i) = u(t(i));
        end
        u_sim = timeseries(u_op,t);
        output = struct('eta',[],'E',[],'u',[],'u_sim',[],'PSD'
          ,[]);
        output.eta = eta;
        output.E = E;
        output.u = u;
        output.u_sim = u_sim;
        output.PSD = PSD;
    else
        output = struct('eta',[],'E',[],'u',[],'PSD',[]);
        output.eta = eta;
        output.E = E;
        output.u = u;
        output.PSD = PSD;
    end
end

end
```

B.0.10 *Radiation.m*

Radiation.m is a function that import radiation data from the file specified in the input file and returns added mass matrix $[A_\infty]$ and memory effect μ.
For the representation of memory effect via state space model *radiation.m* uses the function *ssmarco.m* presented in Appendix A.
In case of convolution integral model of memory effect the retardation radiation kernell matrix is calculated. This will be used in the simulink model.

```
function [ output ] = radiation( geometry, control )

% Global Parameters
rho = 1025;
dw = control.discfreq;
w = dw:dw:5;
% Loading Wamit Data
data_rad = geometry.platform.radiationstru.radiationterms;
rad_flag = geometry.platform.radiationstru.memoryrep;
Trad = geometry.platform.radiationstru.time;
```

```

% memoryrep = 1: statespace representation
% memoryrep = 2: convolution representation
T = data_rad(21:10:end,1); w_file = [0; 2*pi./T];

Ainf_ss = data_rad(11,4)*rho; A_ss = data_rad(11:10:end,4)*rho;
B_ss = data_rad(11:10:end,5)*rho.*w_file;

Ainf_sp = data_rad(12,4)*rho; A_sp = data_rad(12:10:end,4)*rho;
B_sp = data_rad(12:10:end,5)*rho.*w_file;

Ainf_ps = data_rad(18,4)*rho; A_ps = data_rad(18:10:end,4)*rho;
B_ps = data_rad(18:10:end,5)*rho.*w_file;

Ainf_pp = data_rad(19,4)*rho; A_pp = data_rad(19:10:end,4)*rho;
B_pp = data_rad(19:10:end,5)*rho.*w_file;

comparisonplot = 0;
output.Ainf = [Ainf_ss Ainf_sp;                Ainf_ps Ainf_pp];

if rad_flag == 1
    % surge-surge term
    [Ar_ss,Br_ss,Cr_ss] = ssmarco(A_ss,B_ss,Ainf_ss,w_file,
      comparisonplot);
    % surge-pitch term
    [Ar_sp,Br_sp,Cr_sp] = ssmarco(A_sp,B_sp,Ainf_sp,w_file,
      comparisonplot);
    % pitch-surge term
    [Ar_ps,Br_ps,Cr_ps] = ssmarco(A_ps,B_ps,Ainf_ps,w_file,
      comparisonplot);
    % pitch-pitch term
    [Ar_pp,Br_pp,Cr_pp] = ssmarco(A_pp,B_pp,Ainf_pp,w_file,
      comparisonplot);
    % Assembling
    n_ss = length(Ar_ss);
    n_sp = length(Ar_sp);
    n_ps = length(Ar_ps);
    n_pp = length(Ar_pp);
    N = n_ss+n_sp+n_ps+n_pp;
    % Ar
    Ar = zeros(N);
    Ar(1:n_ss,1:n_ss) = Ar_ss;
    Ar(n_ss+1:n_ss+n_sp,n_ss+1:n_ss+n_sp) = Ar_sp;
    Ar(n_ss+n_sp+1:n_ss+n_sp+n_ps,n_ss+n_sp+1:n_ss+n_sp+n_ps) =
      Ar_ps;
    Ar(n_ss+n_sp+n_ps+1:end,n_ss+n_sp+n_ps+1:end) = Ar_pp;
    % Br     Br = zeros(N,2);
    Br(1:n_ss,1) = Br_ss;
    Br(n_ss+1:n_ss+n_sp,1) = Br_sp;
    Br(n_ss+n_sp+1:n_ss+n_sp+n_ps,2) = Br_ps;
    Br(n_ss+n_sp+n_ps+1:end,2) = Br_pp;
    % Cr
    Cr = zeros(2,N);
    Cr(1,1:n_ss) = Cr_ss;
    Cr(2,n_ss+1:n_ss+n_sp) = Cr_sp;
    Cr(1,n_ss+n_sp+1:n_ss+n_sp+n_ps) = Cr_ps;
    Cr(2,n_ss+n_sp+n_ps+1:end) = Cr_pp;

    output.Ar = Ar;
    output.Br = Br;
    output.Cr = Cr;
end

if rad_flag == 2
    dt = control.disctime;
    trad = (0:dt:Trad)';
    w_file(end) = w(end);
    B_ss = interp1(w_file,B_ss,w);
    B_sp = interp1(w_file,B_sp,w);
    B_ps = interp1(w_file,B_ps,w);
    B_pp = interp1(w_file,B_pp,w);
    Kt_ss = zeros(length(trad),1);
    Kt_sp = zeros(length(trad),1);
    Kt_ps = zeros(length(trad),1);
    Kt_pp = zeros(length(trad),1);
    for n = 1:length(trad)
    % surge-surge term
    Kt_ss(n,1) = dw/pi*(2*sum(B_ss(1:length(w)-1).*...
                    cos(w(1:length(w)-1)*trad(n)))+...
                    B_ss(length(w))*cos(w(length(w))*trad(n)));
    % surge-pitch term
    Kt_sp(n,1) = dw/pi*(2*sum(B_sp(1:length(w)-1).*...
                    cos(w(1:length(w)-1)*trad(n)))+...
                   B_sp(length(w))*cos(w(length(w))*trad(n)));
    % pitch-surge term
    Kt_ps(n,1) = dw/pi*(2*sum(B_ps(1:length(w)-1).*...
                    cos(w(1:length(w)-1)*trad(n)))+...
    B_ps(length(w))*cos(w(length(w))*trad(n)));
                    % pitch-pitch term
    Kt_pp(n,1) = dw/pi*(2*sum(B_pp(1:length(w)-1).*...
                    cos(w(1:length(w)-1)*trad(n)))+...
                    B_pp(length(w))*cos(w(length(w))*trad(n)));
    end

   output.Kt_ss = Kt_ss;
   output.Kt_sp = Kt_sp;
   output.Kt_ps = Kt_ps;
   output.Kt_pp = Kt_pp;
   output.trad = trad;

end

end
```

B.0.11 *Diffraction.m*

Diffraction.m calculates the diffraction force. It uses the wave elevation spectrum provided by *wavegenerator.m* and diffraction data from the file specified in the input file.

```
function [ output ] = diffraction(  geometry, waveopt, waveoutput
   , control  )

% Global Parameters
rho = 1025; g = 9.81;
dw = control.discfreq;
w = 0:dw:5;

E = waveoutput.E;
data_diff = geometry.platform.diffractionterms;
beta_target = waveopt.direction;

% Loading Wamit
Data T_alldof(:,1) = data_diff(:,1);
beta(:,1) = data_diff(:,2);
dof(:,1) = data_diff(:,3);
ii = 0; jj = 0;
for kk = 1:size(data_diff,1)
    if beta(kk) == beta_target
        if dof(kk) == 1
          ii = ii+1;
         indsurge(ii,1) = kk;
        elseif dof(kk) == 5;
          jj = jj+1;
          indpitch(jj,1) = kk;
        end
    end
end

% Transfer Function Fdiff(w)/E(w)
% frequency
T_file = T_alldof(indsurge);
w_file = 2*pi./T_file';

% module Xmod = [rho*g*data_diff(indsurge,4)';
rho*g*data_diff(indpitch,4)'];
% phase Xphase = [data_diff(indsurge,5)'*pi/180;
data_diff(indpitch,5)'*pi/180];


% module
w_file(end) = w(end);
Xmod_surge = interp1([0 w_file],[0 Xmod(1,:)],w);
Xmod_pitch = interp1([0 w_file],[0 Xmod(2,:)],w);
% phase
Xphase_surge = interp1([0 w_file],[0 Xphase(1,:)],w);
Xphase_pitch = interp1([0 w_file],[0 Xphase(2,:)],w);

```

```
% Xmod*exp(jXphase)
Xdiff = [(Xmod_surge.*exp(1j*Xphase_surge));
         (Xmod_pitch.*exp(1j*Xphase_pitch))];

% Diffraction Force XE = Xdiff.*[E; E];
XE_abs = abs(XE); XE_angle = angle(XE);
Fdiff = @(t) [sum(XE_abs(1,:).*cos(w*t + XE_angle(1,:)));
              sum(XE_abs(2,:).*cos(w*t + XE_angle(2,:)))];

dt = control.disctime;
T = control.timeend;
t = 0:dt:T;

memoryflag = geometry.platform.radiationstru.memoryrep;
if memoryflag == 2;
    for i = 1:length(t)
        Fdiff_op(:,i) = Fdiff(t(i));
    end
    Fdiff_sim = timeseries(Fdiff_op,t);
    output = struct('XE',[],'Fdiff',[],'Fdiff_sim',[]);
    output.XE = XE;
    output.Fdiff= Fdiff;
    output.Fdiff_sim = Fdiff_sim;
else
    output = struct('XE',[],'Fdiff',[]);
    output.XE = XE;
    output.Fdiff= Fdiff;
end

end
```

B.0.12 *Viscous.m*

Viscous.m calculates the viscous force using the wave velocity calculated by *wave-generator.m* and platform's draft calcualted by *hydrostatics.m*.

```
function [ output ] = viscous( geometry, waveoutput, hsoutput )

% Global Parameters
rho = 1025;

% Platform z-coordinate
draft = hsoutput.draft;
dz = geometry.platform.dz;
z = -(dz/2:dz:draft-dz/2)';
Nz = length(z);

% Relative Velocity of Platform
u = waveoutput.u;
Vpl = @(x) x(1)*ones(Nz,1) + x(2)*z;
Vrel = @(t,x) u(t) - Vpl(x);

% Drag Coeffiencient
```

```
c_drag = geometry.platform.dragcoef;
D = geometry.platform.bottomdiameter;
visc_coef = 1/2*rho*c_drag*D;
Fvisc = @(t,x) visc_coef*[ sum( abs(Vrel(t,x).*Vrel(t,x) ) );
                           sum( abs(Vrel(t,x).*Vrel(t,x).*z ) )];

output = struct('Vpl',[],'Vrel',[],'Fvisc',[]);
output.Vpl = Vpl;
output.Vrel = Vrel; output.Fvisc = Fvisc;

end
```

B.0.13 *Mooring.m* and *AddDamp.m*

Mooring.m and *AddDamp* just transform the input data of mooring line into a matrix.

```
function [ Cmoor ] = mooringlines( geometry )
    Cmoor = geometry.mooring.matrix;
end

function [ Bad ] = addDamp( geometry )
    Bad = geometry.additionaldamping.matrix;
end
```

B.0.14 *CompleteSSmodel.m*

This function is used only if memory effect is repbresented with state space model. This is a function that assembles all terms previously calculated in a state and input matrix of a globla state space system.

```
function [ output ] = completeSSmodel( diffoutput, viscoutput,
  radoutput, hsoutput, Cmoor, Bad );

Ainf = radoutput.Ainf;
Ar = radoutput.Ar;
Br = radoutput.Br;
Cr = radoutput.Cr;
n_r = length(Ar);

Msys = hsoutput.Msys;
Chs = hsoutput.Chs;
M = Msys + Ainf;
R = Bad;
K = Cmoor + Chs;

Fdiff = diffoutput.Fdiff; Fvisc = viscoutput.Fvisc;

% State Matrix
Astato = [ -inv(M)*R                -inv(M)*K                -inv(M
  )*Cr;
```

```
            eye(2)              zeros(2)            zeros(2,
              n_r);
                 Br              zeros(n_r,2)
                                       Ar];

% Input Matrix Binput = zeros(length(Astato),2);
Binput(1:2,1:2) = inv(M);

% Input  U = @(t,x) Fdiff(t) + Fvisc(t,x);

output = struct('Astato',[],'Binput',[],'U',[]);
output.Astato = Astato;
output.Binput = Binput;
output.U = U;

end
```

B.0.15 *Intgration.m*

Integration.m solves the differential problem of equation according on the parameters provided by User in the input file.

```
function [output] = integration(SSoutput,geometry,control)

dt = control.disctime; T = control.timeend; t = 0:dt:T;
flag = control.solver;

Astato = SSoutput.Astato;
Binput = SSoutput.Binput;
U = SSoutput.U;

x0dof = geometry.platform.initialconditions;
x0 = zeros(length(Astato),1); x0(1:4) = x0dof;

fun = @(t,x) Astato*x + Binput*U(t,x);

output = struct('surge',[],'pitch',[],'duration',[]);

if flag == 1
    clock_start = clock;
    [tout,x] = ode45(fun,t,x0);
    clock_stop = clock;
    clear tout
    duration = clock_stop-clock_start;
    output.surge = x(:,3);
    output.pitch = x(:,4);
end
if flag == 2
    clock_start = clock;
    [t,x,xp]=rk4(fun,T,dt,x0);
    clock_stop = clock;
    duration = clock_stop-clock_start;
    output.surge = x(3,:);
    output.pitch = x(4,:);
end
output.duration = duration;

end
```

B.0.16 Modelsim.m

Modelsim.m is a matlab script that assembles all the terms previolusly calculated in order to be solved in simulink enviroment (figure B.1). It is only used when memory effect is represented by convolution integral. This term is modeled in simulink through a "interpreted matlab function" named *memoryconvolution.m.* This function uses $[K_t]$ and t_{rad} calculared by *radiation.m.* and calculates $\mu(t)$ as function of the platform dofs velocity u.

```
function [ mu ] = memoryconvolution( u )

load trad dt = mean(diff(trad));
Trad = trad(end);
Nrad = Trad/dt;

load Vbuffer
Vbuffer(1:Nrad,1) = Vbuffer(2:Nrad+1,1);
Vbuffer(1:Nrad,2) = Vbuffer(2:Nrad+1,2);
Vbuffer(Nrad+1,1) = u(1);
Vbuffer(Nrad+1,2) = u(2);
save('Vbuffer','Vbuffer')

load Kt Kt_ss(:,1) = Kt(:,1);
Kt_sp(:,1) = Kt(:,2); K
t_ps(:,1) = Kt(:,3); Kt_pp(:,1) = Kt(:,4);

mu = zeros(2,1);
mu(1) = sum( flipud(Kt_ss).*Vbuffer(:,1) )*dt + sum( flipud(Kt_sp
  ).*Vbuffer(:,2) )*dt;
mu(2) = sum( flipud(Kt_ps).*Vbuffer(:,1) )*dt + sum( flipud(Kt_pp
  ).*Vbuffer(:,2) )*dt;

end
```

```
%% Modelsim.m

% Memory Effect
Kt_ss = radoutput.Kt_ss;
Kt_sp = radoutput.Kt_sp;
Kt_ps = radoutput.Kt_ps;
Kt_pp = radoutput.Kt_pp;
Kt = [Kt_ss Kt_sp Kt_ps Kt_pp];
save('Kt','Kt')

trad = radoutput.trad;
```

```
save('trad','trad')

Vbuffer = zeros(length(trad),2);
save('Vbuffer','Vbuffer')

% Matrix
Ainf = radoutput.Ainf;
Msys = hsoutput.Msys;
Chs = hsoutput.Chs;
M = Msys + Ainf;
R = Bad;
K = Cmoor + Chs;

% Time
dt = control.disctime;
T = control.timeend;
t = 0:dt:T;

% Viscous
draft = hsoutput.draft;
dz = geometry.platform.dz;
z = -(dz/2:dz:draft-dz/2)';
u_sim = waveoutput.u_sim;
rho = 1025;
c_drag = geometry.platform.dragcoef;
D = geometry.platform.bottomdiameter;
visc_coef = 1/2*rho*c_drag*D;

% Diffraction
Fdiff_sim = diffoutput.Fdiff_sim;

solver = control.solver;
if solver == 1
    sim('Model45');
end
if solver == 2
    sim('Model4');
end

simoutput = struct('t',[],'surge',[],'pitch',[]);
simoutput.t = t;
simoutput.surge = yout(1,1,:);
simoutput.pitch = yout(2,1,:);
```

B.0.17 Main.m

This script shows how is effectively structured the code.

```

input;

control = struct('disctime',[],'discfreq',[],'timeend',[],'solver',[]);
control.disctime = dt;
control.discfreq = dw;
```

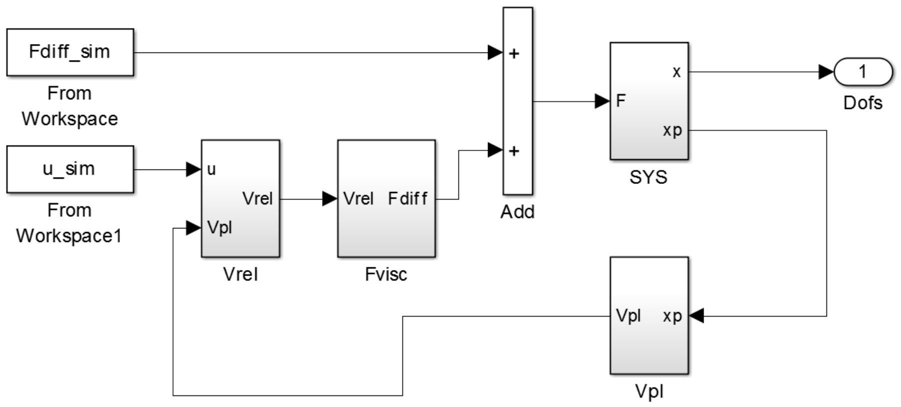


Figure B.1: Simulink Model

```
control.timeend = T;
control.solver = 1;

waveopt = struct('type',[],'w_eta',[],'eta_a',[],'phi',[],'sigheight',[],...
                'cutfreq',[],'period',[],'direction',[]);
waveopt.w_eta = frequency;
waveopt.eta_a = amplitude;
waveopt.phi = phase;
waveopt.sigheight = Hs;
waveopt.cutfreq = ww_stop;
waveopt.period = Tp;
waveopt.direction = beta0;
waveopt.type = flageta;

radiationstru = struct('radiationterms',[],'memoryrep',[],'time',[]);
radiationstru.radiationterms = data_radiation;
radiationstru.memoryrep = rad_flag;
radiationstru.time = Trad;

platform = struct('mass',[],'inertia',[],'CM',[],'bottomdiameter',[],'topdiameter',[],...                     'heigth1',[],'heigth2',[],'height3',[],'draft',[],'dz',[],'dragcoef',[],...                     'diffractionterms',[],'radiationstru',[],'initialconditions',[]);
platform.mass = Mpl;
platform.inertia = Jpl;
platform.CM = CMpl;
platform.bottomdiameter = Ddown;
platform.topdiameter = Dup;
platform.height1 = h3 ;
platform.height2 = h2;
platform.height3 = h1;
platform.dz = 1;
platform.dragcoef = c_drag;
platform.diffractionterms = data_diffraction;
platform.radiationstru = radiationstru;
platform.initialconditions = X0;

tower = struct('mass',[],'heigth',[],'CM',[]);
tower.mass = Mto;
```

```matlab
tower.heigth = hto;
tower.CM = CMto;

nacelle = struct('mass',[],'CM',[]);
nacelle.mass = Mna;
nacelle.CM = CMna;

rotor = struct('mass',[],'CM',[]);
rotor.mass = Mro; rotor.CM = CMro;

mooring = struct('preload',[],'matrix',[]);
mooring.preload = 1607e3;
mooring.matrix = Clines;

additionaldamping = struct('matrix',[]);
additionaldamping.matrix = Bad;

geometry = struct('platform',[],'tower',[],'nacelle',[],'rotor',[],...
                                   'mooring',[],'additionaldamping',[]);
geometry.platform = platform;
geometry.tower = tower;
geometry.nacelle = nacelle;
geometry.rotor = rotor;
geometry.mooring = mooring;
geometry.additionaldamping = additionaldamping;


[ hsoutput ] = hydrostatics( geometry );
[ waveoutput ] = wavegenerator( waveopt, geometry, hsoutput, control );
[ Cmoor ] = mooringlines( geometry );
[ Bad ] = add_damping( geometry );
[ viscoutput ] = viscous( geometry, waveoutput, hsoutput);
[ diffoutput ] = diffraction( geometry, waveopt, waveoutput , control );
[ radoutput ] = radiation( geometry, control );

memoryflag = geometry.platform.radiationstru.memoryrep;

if memoryflag == 1,
    [ SSoutput ] = completeSSmodel( diffoutput, viscoutput, ...
                                                    radoutput, hsoutput
      , Cmoor, Bad );
    [output] = integration(SSoutput,geometry,control);
    surge = output.surge;
    pitch =     output.pitch;
end

if memoryflag == 2,
    Modelsim;
    surge = simoutput.surge;
    pitch = simoutput.pitch;
end

X0 = hsoutput.X0;
surge0 = X0(1);
pitch0 = X0(2);

eta = waveoutput.eta;
surge = surge0 + surge;
pitch = pitch0 + pitch;
```