FLOATFARM
NEXT GENERATION FLOATING WIND FARMS

DELIVERABLE D7.2

HAWC2 Interfacing Framework

Fanzhong Meng, Andrew Russell and Fabio Pierella (DTU Wind and Energy Systems)

Document track information

Project information	
Project Title	Developing the Next Generation of Environmentally-Friendly Floating Wind Farms with Innovative Technologies and Sustainable Solutions
Starting date	01.01.2024
Duration	48 months
Programme	HORIZON EUROPE
Call Identifier	HORIZON-CL5-2023-D3-01
Grant Agreement No	101136091

Deliverable information	
Deliverable number	7.2
Work package number	7
Deliverable title	HAWC2 Interfacing Framework
Lead beneficiary	DTU Wind and Energy Systems
Authors	Fanzhong Meng, Andrew Russell and Fabio Pierella
Due date	30.04.2025
Actual submission date	30.04.2025
Type of deliverable	Other
Dissemination level	Public

Funded by the European Union

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Innovation Council and SMEs Executive Agency (EISMEA). Neither the European Union nor EISMEA can be held responsible for them.

Revision table

Version	Contributors	Date	Description
V0.1	Andrew Russell, DTU Fanzhong Meng, DTU Fabio Pierella, DTU	31.03.2025	First draft
V0.2	Claudio Lugni, CNR	10.04.2025	Review
V1.0	Andrew Russell, DTU Fanzhong Meng, DTU Fabio Pierella, DTU	30.04.2025	Final version

List of acronyms

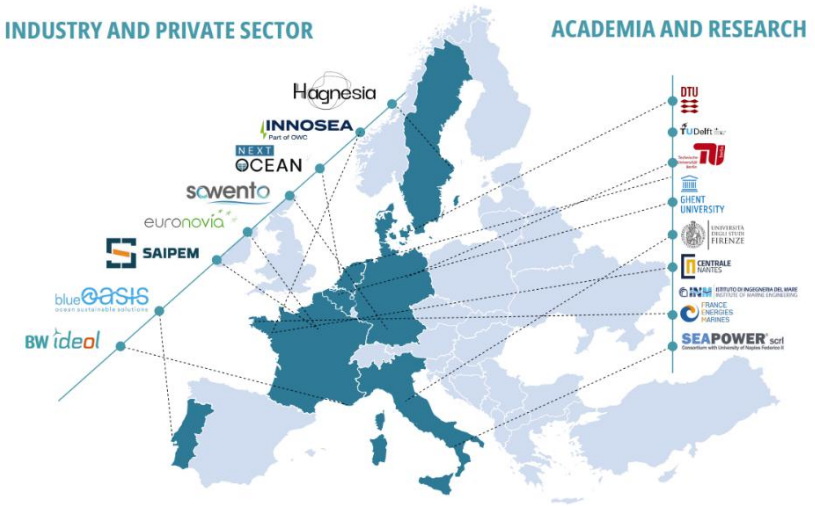
Acronym	Full name
AEP	Annual Energy Production
CNR	Consiglio Nazionale delle Ricerche
DEL	Damage Equivalent Load
DLC	Design Load Case
DLL	Dynamic Link Library
DOF	Degree Of Freedom
DTU	Danmarks Tekniske Universitet
DTUWEC	DTU Wind Energy Controller
FOW	Floating Offshore Wind
FOWT	Floating Offshore Wind Turbine
HAMS	Hydrodynamic Analysis of Marine Structures
HAWC2	Horizontal Axis Wind turbine simulation Code 2nd generation
IO	Input-Output
NREL	National Renewable Energy Laboratory
ROSCO	Reference Open-Source Controller
WEIS	Wind Energy with Integrated Servo-control
WISDEM	Wind-plant Integrated System Design and Engineering Model
WP	Work Package
WSL2	Windows Subsystem for Linux 2

ABOUT FLOATFARM

The FLOATFARM project is a Research and Innovation Action funded by the European Union under the Horizon Europe program. This project is closely linked to the FLOATECH project (2020-2023) and aims to bring the technologies developed within FLOATECH to the next level of technological readiness, complementing them with a significant number of new concepts, innovations and methods.

FLOATFARM aims to significantly advance the maturity and competitiveness of floating offshore wind (FOW) technology by increasing energy production, achieving significant cost reductions within the design and implementation phases, improving the offshore wind value chain and supporting EU companies in this growing sector. Additionally, FLOATFARM aims to decrease negative environmental impacts on marine life and to enhance the public acceptability of FOW, thereby accelerating the EU energy transition.

The FLOATFARM consortium is coordinated by the Technische Universität Berlin and is made up of 17 partners spread across 8 countries, each contributing to technology and scientific excellence in the wind energy sector.



The approach of FLOATFARM can be broken down into three actions:

1) **Turbine Technology**: Development of innovative technologies and methods for improvements on an individual FOW turbine level,

2) **Farm Technology**: Development, investigation and demonstration of technologies that are applicable to an array of turbines within a FOW farm,

3) **Environmental & Socioeconomic Impacts**: Model development, data collection and scenario analysis of environmental, economic and sociological impacts of FOW farms.

Table of contents

1. Introduction 1
1.1. Context within FLOATFARM 1
1.2. Document Structure 1
2. WEIS-HAWC2 1
2.1. Overview and Motivation 1
2.2. WEIS 3
2.3. HAWC2 in the WEIS framework 4
3. Access, Installation and Basic User Guide 6
3.1. Access and Licensing 6
3.2. Installation 7
3.2.1. Important HAWC2 installation requirements 8
3.3. User Guide 9
4. Demonstration test case 10
4.1. IEA 15MW UMaine VolturnUS-S platform 11
5. Conclusions 15
6. References 15

1. Introduction

1.1. Context within FLOATFARM

This document presents the interface between the HAWC2 (Horizontal Axis Wind turbine simulation Code 2nd generation) [1] aero-hydro-servo-elastic simulation tool and the WEIS (Wind Energy with Integrated Servo-control) holistic floating offshore wind turbine optimization framework [2], henceforth termed WEIS-HAWC2, that was developed under Task 7.1 of WP7 of FLOATFARM. This integration provides the basis for several optimization and design tasks within Work Packages (WPs) 1-6 and is therefore essential for the successful achievement of FLOATFARM's objectives.

This document should be read as a guide to the public release of WEIS-HAWC2, which is the actual deliverable.

1.2. Document Structure

The document is divided into three sections. Firstly, the motivation for WEIS-HAWC2 and its capabilities are discussed. Secondly, a guide to installing and running the software is provided. Lastly, results of a test case optimization problem are demonstrated.

2. WEIS-HAWC2

2.1. Overview and Motivation

Floating Offshore Wind Turbines (FOWTs) are inherently interdisciplinary systems that are excited by a multitude of non-linear external disturbances. Figure 1 provides an overview of the main sub-systems that comprise a FOWT and the environment in which they operate. The current generation of wind turbines has rotor diameters exceeding 240 m and rated powers of at least 15 megawatts. The large rotor-swept areas of such turbines cause their structures to experience vastly different aerodynamic conditions and highly variable loadings depending on the rotor azimuthal positions of the blades. At the same time, floating structures are exposed to linear and non-linear wave excitation that potentially cause resonance. The aerodynamic and hydrodynamic forces cause the floating substructure to move within its rigid-body degrees of freedom (DOFs), most prominently in surging (movement forwards and backwards) and pitching (tilting forwards and backwards) motions. FOWTs are therefore prone to negative aerodynamic damping, whereby the magnitude of oscillations increases rather than decreases

if the controller of a FOWT is not co-designed together with other sub-components such as rotor aero-elastic design, tower design, floater design, etc. The continuous motion of the platform adds to the loadings that are experienced by a FOWT, compared to their fixed-bottom or onshore counterparts (e.g. on the intersection between the floating sub-structure and the tower).

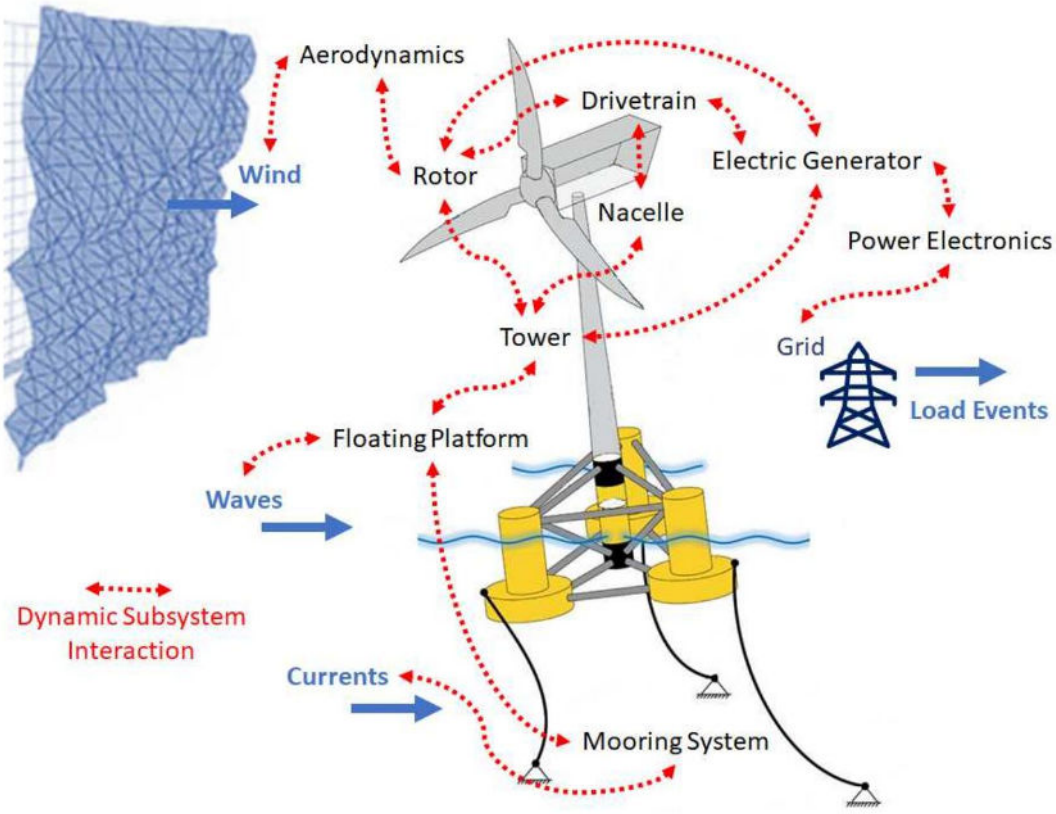


Figure 1 – Sub-systems of a FOWT and environmental excitation. Figure taken from [3].

The complex interaction of FOWT sub-systems poses a challenge to designers of each sub-component as any design decision that is taken might have an unintended consequence on overall system behavior in such a tightly coupled dynamic system. For example, the controller's behavior can directly impact upon the tower base fatigue and the stability of the platform. Currently, this leads to a conservative design of structural elements such as the tower and detuning of the pitch controller in above-rated wind speed conditions (compared to their fixed-bottom counterparts) to avoid negative aerodynamic damping through aversion of coupling of the pitch control actuation with the platform pitching motion.

To cope with this challenge, the design approach of FOWTs can evolve from a sequential one to a Control Co-Design (CCD) approach. In a sequential approach, the mechanical (sub-)systems are typically designed initially, and the controller design begins after the mechanical system is accomplished. Within the design of

the mechanical sub-systems, the components of each group must be designed such that pre-defined constraints and objectives are met. By nature, these constraints and objectives have to be conservative to prevent a highly iterative process that can incur significant computation time [3]. CCD offers the potential to design the aero-hydro-elastic floating offshore wind energy system at the same time as the controller design, thus finding solutions that could not be found in a sequential approach [4] and allowing for design trade-offs to be identified and accounted for within the optimization process. Enabling this type of design with HAWC2 is our objective in WP7 of the FLOATFARM project. Before defining the objective and scope further, we describe the WEIS framework for clarity.

2.2. WEIS

The WEIS toolset is an open-source framework developed by the National Renewable Energy Laboratory (NREL) to design and optimize FOWT structures. WEIS configures various wind turbine design, simulation and analysis tools as components within an OpenMDAO [5] hierarchical optimization framework, whereby the outputs of upstream components are used as inputs to downstream components. Figure 2 shows the tools that make up WEIS and demonstrates the direction of the workflow in a typical optimization or design task.

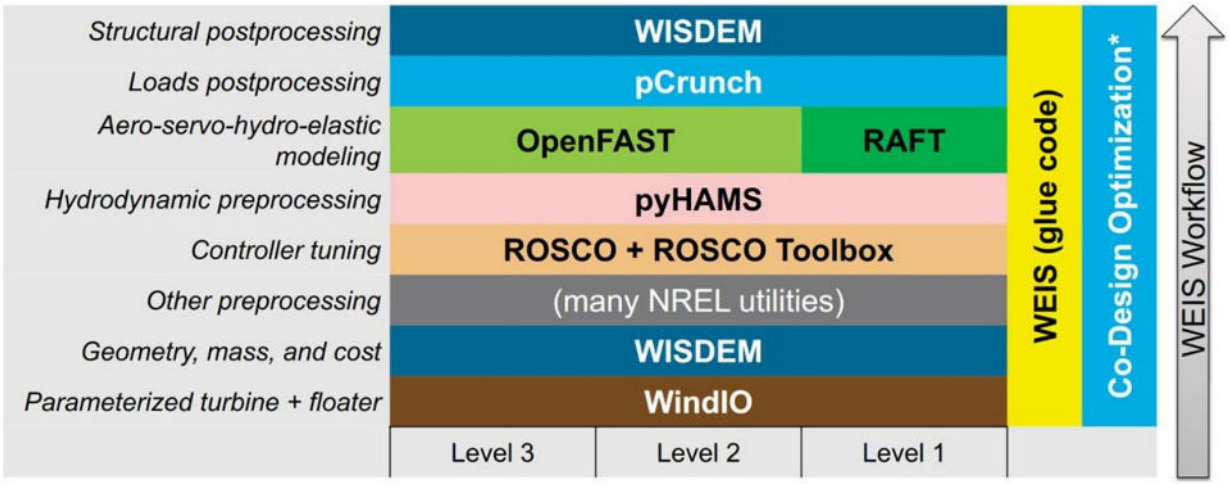


Figure 2 – Components that are integrated in WEIS, taken from [6].

In the present document, the toolchain is briefly described. For a detailed description of the individual tools and the workflow, the reader is referred to publication [6].

WindIO: Parametric definition of the properties of the various structural components of the FOWT [7] such as the airfoils, blade geometry, internal blade layup, floater geometry, materials and component diameters and thicknesses.

WISDEM: Wind-plant Integrated System Design and Engineering Model (WISDEM) [8] is an optimization framework in its own right. WISDEM relies on steady-state tools and includes techno-economic modules to derive masses, costs and structural beam properties. Within WEIS, WISDEM is utilized mostly to evaluate costs and convert the WindIO definition into an OpenFAST/RAFT/HAWC2 model.

ROSCO: Reference Open-Source Controller (ROSCO) [9] is an open-source reference controller that comes with the ROSCO toolbox. The toolbox allows for a generic process to enable CCD.

pyHAMS: Python wrapper for Hydrodynamic Analysis of Marine Structures (HAMS) [10], which is an open-source software to analyze wave-structure interaction in the frequency domain. Added mass, damping coefficients and excitation forces are evaluated for 3D structures so that radiation and diffraction effects can be included in time domain simulations.

OpenFAST [11] (aeroelastic solver): Nonlinear aero-servo-hydro-elastic simulation framework for wind turbine response in the time domain. The typical simulation duration is in the magnitude of minutes. The aeroelastic solver is a key component of the WEIS optimization loop and is critical to the assessment of the viability of the FOWT design. The optimizer instructs the solver to perform time-domain simulations of the FOWT design according to the design load cases (DLCs) specified within the industrial standards [12].

RAFT: Response Amplitudes of Floating Turbines (RAFT) is a linear, frequency domain representation of a FOWT [13]. The typical simulation duration is in the magnitude of seconds.

pCrunch: IO and post processing interface for OpenFAST results [14]. Used within WEIS to analyze load case results and derive parameters usually used as constraints or merit figures (such as DELs, AEP, etc.).

2.3. HAWC2 in the WEIS framework

The objective of the coupling that is described in this document is to expand the stack of tools that WEIS combines with the HAWC2 wind turbine simulation software and thus provide a design engineer with another option next to OpenFAST and RAFT to carry out aero-elastic-servo-hydro-elastic simulations. Following this integration, the user can seamlessly switch between HAWC2, OpenFAST and RAFT inside WEIS, giving birth to a strong, unified optimization framework

The availability of HAWC2 within the WEIS optimization framework provides HAWC2's established and expanding academic and industrial user base with new advanced co-design optimization capabilities. The coupling of WEIS and HAWC2 is further motivated by the fact that HAWC2 possesses some advantages with respect to OpenFAST. First, it uses a nonlinear formulation for the body dynamics that is widely recognized as more accurate than OpenFAST's. Moreover, the use of HAWC2's recently developed unique static solver functionality also has the potential to greatly reduce the computation time of the optimization process. In fact, the static solver is capable of rapidly converging upon a steady-state solution to significantly reduce the duration of start-up transients, leading to expedited optimization studies.

The first step to enabling the use of HAWC2 within WEIS was ensuring that the parameterized windIO.yaml input file could be translated to a HAWC2 model. The challenge lies in the syntax and notation used by HAWC2, which is very different to that of both OpenFAST and the windIO.yaml input file. Moreover, the coordinate systems used by HAWC2 and OpenFAST are different and so these conflict with the axes representations used within the windIO.yaml input file. These issues led to the development of the windIO converter at DTU [15] - a tool capable of performing a coordinate transformation and converting windIO.yaml input files to HAWC2-formatted python dictionaries and finally to accurate HAWC2 input files.

Subsequently, an OpenMDAO component has been created for HAWC2, which can receive the required inputs from WISDEM, pyHAMS etc and map them to HAWC2-specific inputs. Afterward, the control parameters tailored for DTUWEC [16] [17] are designed according to the required inputs, which enables controller optimization and controller co-design approaches. This allows for the iterative changes made (according to the user-specified design variables) to the turbine's subsystems during an optimization namely the blades, hub, drivetrain, tower/monopile, floating platform, moorings and controller to be captured within the HAWC2 input files.

The HAWC2 OpenMDAO component is offered as an alternative to the OpenFAST OpenMDAO component and DTUWEC is offered as an alternative to ROSCO controller toolbox, as indicated in Figure 3.

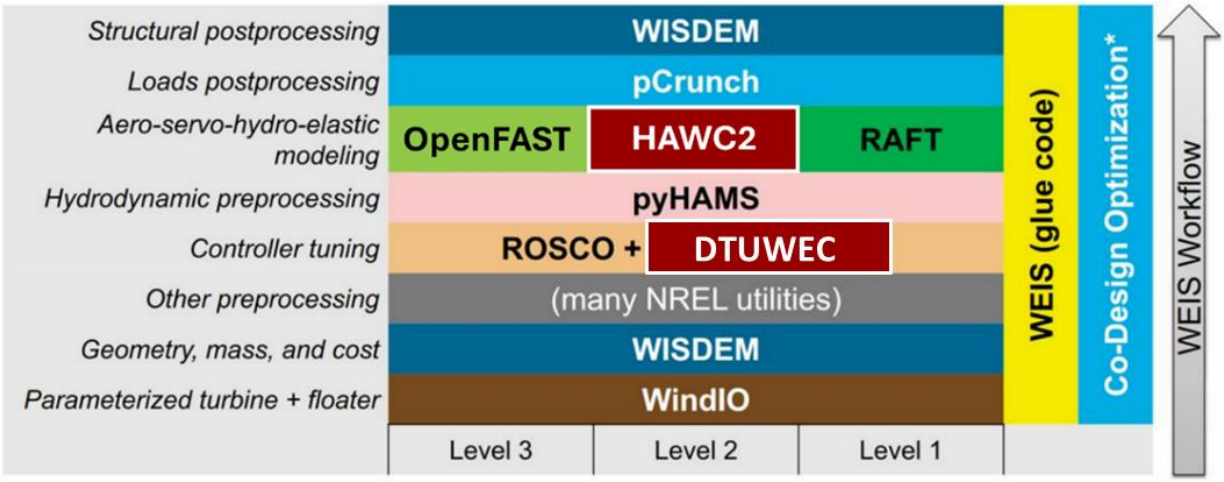


Figure 3 – HAWC2 in the WEIS Framework, taken from [6] and modified.

The capability to perform HAWC2 simulations for user-defined design load cases (DLCs) is included within the coupling and the case-specific HAWC2 input files are generated and called from a HAWC2 wrapper within WEIS. Once simulations are completed, their relevant time series are written out, translated from the binary ".hdf5" format used in HAWC2 to the OpenFAST ASCII format and mapped to channels that can be interpreted by the post-processing tool pCrunch. pCrunch derives parameters such as the AEP, DELs, max platform pitch angle etc from the time series results. These outputs are utilized by the optimizer as constraints or merit figures. The capability to create animations of iterative changes made to the floating platform or tower geometry during the optimization process has also been included.

3. Access, Installation and Basic User Guide

3.1. Access and Licensing

WEIS-HAWC2 is released as a GitLab repository. This was chosen because this allows for the repository to be maintained and updated efficiently. The repository can be accessed via this public link:

https://gitlab.windenergy.dtu.dk/floatfarm/weis-hawc2-coupling

3.2. Installation

WEIS-HAWC2 has been developed and tested on Windows Subsystem for Linux version 2 (WSL2), though it is also expected to function on native Linux as well. While cross-platform compatibility has been considered throughout development, installation on a Windows platform is not recommended at this point. This is one area of focus to develop the code further after the initial release in April 2025.

The installation process is in line with that of the core WEIS code. The following instructions and commands have been tested starting with a baseline WSL distribution (version 2) and Ubuntu-22.04. Installation with Miniforge in a newly created self-contained virtual environment is recommended.

1) **Install WSL2:**
    a. In Windows Terminal (run as administrator), enter:
    `wsl --install -d Ubuntu-22.04`
    b. Restart the terminal again as administrator
    c. Enter WSL by entering `wsl.exe`

    Detailed information regarding WSL2 and the installation can be found via this link: https://learn.microsoft.com/en-us/windows/wsl/

2) **Install the Linux Miniforge installer:**
    a. Download the Linux Miniforge installer:
    `curl -L -O "https://github.com/conda-forge/miniforge/releases/latest/download/Miniforge3-Linux-x86_64.sh"`
    b. Install Miniforge within WSL within the terminal (as administrator) using:
    `bash Miniforge3-Linux-x86_64.sh`
    It should ask you to agree to the license agreement and confirm the installation location. Once completed, it will ask to update your shell profile to automatically initialize `conda`. Enter yes and restart the terminal. Enter WSL2 again using `wsl.exe`.

3) **Clone the WEIS-HAWC2 repository:**
```
git clone https://gitlab.windenergy.dtu.dk/floatfarm/weis-hawc2-coupling.git
cd weis-hawc2-coupling
```

4) **Create and activate a virtual environment:**
```
conda env create --name h2weis-env -f environment.yml
conda activate h2weis-env
```

5) Add the final packages and install the software:

```
conda install -y petsc4py mpi4py
conda install -c conda-forge pyoptsparse
pip install windio-converter
pip install --no-deps -e . -v
```

3.2.1. Important HAWC2 installation requirements

1) Patch WISDEM and RAFT

During development, a bug was found within the WISDEM code that must be corrected manually for HAWC2 to run successfully. As a specific release version of WISDEM (3.16.4) was used to develop the WEIS-HAWC2 coupling, this version must be used within the created virtual environment. Therefore, this bug could not be retrospectively fixed by the WISDEM development team for the specific release of WISDEM that is used for WEIS-HAWC2.

Furthermore, minor modifications were made to the RAFT library so that additional outputs required for DTUWEC tuning could be included.

To ensure that these changes are reflected in the WEIS-HAWC2 virtual environment packages, users should update the "gc_WT_InitModel.py" and "omdao_raft.py" scripts of the WISDEM and RAFT packages, respectively, by simply executing these steps:

```
cd weis/patch_wisdem
python patch_wisdem.py
cd ../..
```

2) Placement of HAWC2 binary, .dll, .so, executable and license files

To use HAWC2 within the WEIS framework, all of the relevant .dll, .so, license and executable files must be downloaded separately and placed within the "HAWC2_bin" folder. The control .dll and .so files have been included in the repository and are housed in their own folder titled "control" within the HAWC2_bin folder (HAWC2_bin/control).

Congratulations, you now have installed WEIS-HAWC2 on your local machine! To test if everything is working, navigate to the "HAWC2_examples" folder and run the example:

```
cd examples/HAWC2_examples
python weis_driver_umaine_semi_hawc2.py
```

3.3. User Guide

When a user plans to use HAWC2 in an optimization problem or design task within WEIS, the workflow follows the same logic as if a level 1 (RAFT) or level 2 or 3 (OpenFAST) optimization with WEIS was to be run – use of HAWC2 has been assigned to a level 5 optimization. Fundamentally, WEIS is accessed by a user through three *.yaml type files and one driver python script that calls each of the yaml files and runs the WEIS framework. The required files are as follows:

windIO.yaml: This file contains the parametric definition of a FOWT including information such as the airfoils, blade geometry, internal blade layup, floater geometry, materials, diameters, wall thicknesses, etc. There are several wind turbine geometries available to the research community:

- NREL5MW-Spar OC3 & -Semi OC4
- IEA 3.4-130 RWT
- IEA-15MW-240-RWT Monopile & VolturnUS-S
- IEA-22MW-280-RWT Monopile & VolturnUS-S

The WEIS-HAWC2 coupling has primarily been developed and tested with the IEA-15MW-240-RWT VolturnUS-S FOWT model [18][19]. Further development is planned to ensure compatibility with all turbine models.

modeling_options.yaml:
The modeling options file lets the user define code specific simulation settings. Here, the user may select between the different aero-servo-hydro-elastic modeling tools (OpenFAST, RAFT, HAWC2) and enter inputs relevant to the simulation setup, the controller and the DLCs. A detailed overview of available inputs, detailed descriptions, entry-types and bounds can be found in the modeling_schema.yaml in this directory (weis-hawc2-coupling/weis/inputs/). Here, all the HAWC2 relevant inputs are found in the "Level5" object and "hawc2_configuration", which is nested in the properties object. The relvant inputs of performing control co-design or controller optimization based on DTUWEC are found in the "DTUWEC" object.

analysis_options.yaml:
The analysis options file determines the context in which the simulations are performed. Either, the chosen load cases are simply run for analysis purposes, an optimization problem is set up or a design of experiments built. If either of the latter two options are chosen, design variables (denoting the turbine parameters that are to be changed between iterations), constraints (denoting the values of parameters that should not be exceeded during the optimization) and, if required, a merit figure (the target parameter that is intended to be maximized or minimized during the optimization) can be chosen. Furthermore, settings concerning the

optimization algorithm such as the maximum number of iterations or the convergence criteria can be defined.

weis_driver.py:
This python script points to the three abovementioned files and invokes the "run_weis()" method, passing the three files as arguments to the function.

A detailed flowchart of the input files' and HAWC2's role within the WEIS optimization framework is shown in Figure 4.

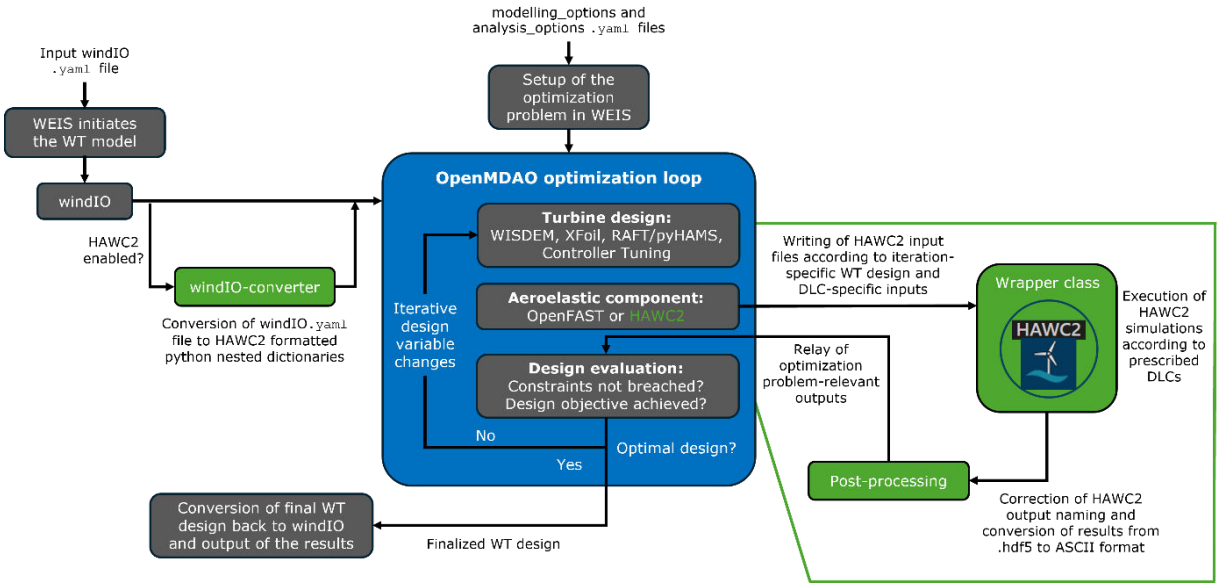


Figure 4 - Flowchart illustrating the WEIS optimization framework and where the newly implemented HAWC2 solver lies within it. Grey boxes represent existing WEIS functionality and green boxes/text represent the additions made to achieve the HAWC2 integration.

4. Demonstration test case

The test case presented below is intended to demonstrate the functional coupling between HAWC2 and WEIS and to show that the optimizer finds a solution to a problem that is easy to interpret. This test case was only performed using short-length HAWC2 simulations covering a reduced number of wind speeds for a single seed to minimize the computation time. This was a suitable approach for this proof-of-concept test case because the goal is to demonstrate that the HAWC2 component can receive the inputs sent from upstream components within the OpenMDAO hierarchy, perform the simulations according to the prescribed DLCs, format and postprocess the results of the simulations and relay the relevant outputs to the downstream components within the OpenMDAO hierarchy.

Therefore, a comprehensive suite of DLC simulations was not required for this demonstration case. As such, the results should not be considered to be indicative or comparable to those of a comprehensive optimization study.

4.1. IEA 15MW UMaine VolturnUS-S platform

In this example, the geometry of the semi-submersible platform of the IEA 15MW UMaine VolturnUS-S was modified with the objective of minimizing the mass of the platform.

We simulated steady conditions at wind speeds of {6, 8, 10, 12, 14, 16, 18, 20} [ms^{-1}]. The simulations did not include a controller and thus the rotor was configured to rotate about a frictionless bearing at a constant speed appropriate to each wind speed. Irregular ocean waves were simulated, with wave heights and periods also appropriate to each wind speed. Thus, this optimization problem was formulated as:

- Minimize (merit figure):
  - Platform mass
- By varying (design variables):
  - Center column diameter
  - Outer column diameters
  - Outer column thicknesses
  - Freeboard height (height of the columns above the still water level)
  - Draft (depth of the columns below the still water level)
  - Center-to-outer column distances
- Subject to (constraints):
  - Maximum platform pitch angle
  - Standard deviation (σ) of the platform pitch angle
  - Maximum nacelle acceleration

Figure 5 provides a diagram of the design variables (except for the outer column thickness) modified during this optimization problem.

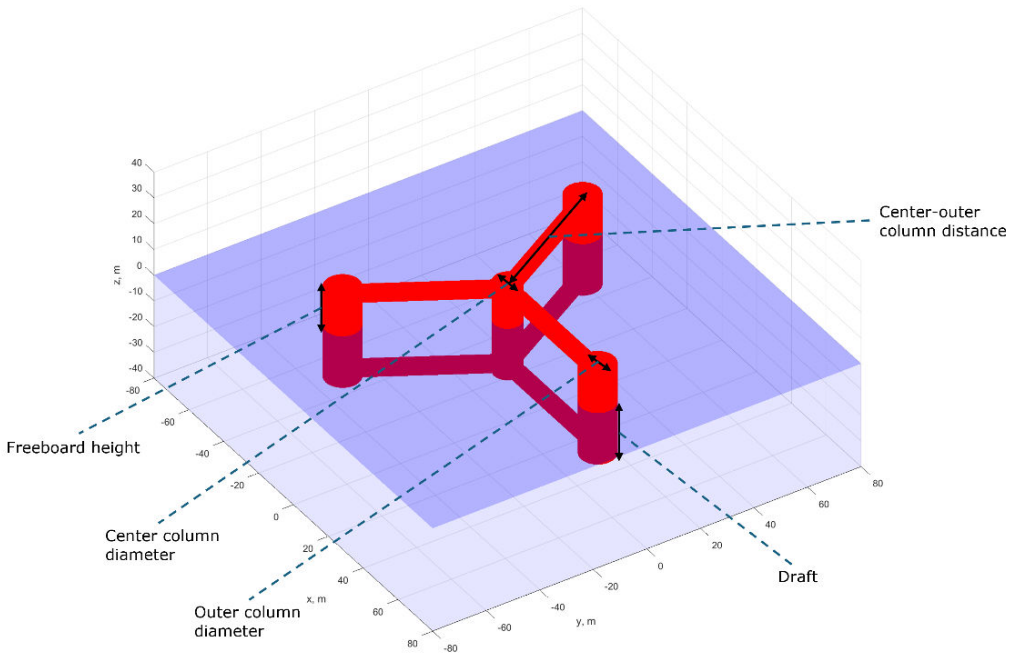


Figure 5 – Properties of the VolturnUS-S platform geometry set as design variables (excluding the outer column wall thickness).

Table 1 shows the original values of the design variables, constraints (obtained from the first iteration simulation results, using the original design) and merit figure, along with the applicable upper and lower bounds.

Table 1 – Original values, upper bounds and lower bounds of the design variables, constraints and merit figure.

Optimization metric	Parameter	Units	Original value	Upper bound	Lower bound
Design variables	Center column diameter	m	10	16	8
	Outer column diameters	m	12.5	16	10
	Outer column thicknesses	m	0.05	0.25	0.05
	Freeboard height	m	15	20	10
	Draft	m	20	50	20
	Center-outer column distances	m	51.75	75	40
Constraints	Max. platform pitch angle	°	4.52	6	-
	σ platform pitch angle	°	0.28	0.5	-
	Max. nacelle acceleration	ms^{-2}	0.95	1	-
Merit figure	Platform mass	kilotons	13.37	-	-

Given that the upper bounds of the constraints were greater than their original values, one would expect the optimizer to reduce the mass and geometry of the

platform to allow for greater motions, while seeking to remain within the upper bounds of the constraints.

The key results of the optimization are shown in Figures 6-8. The platform mass was reduced by 6.8%, from 13.37 kilotons to 12.46 kilotons, as shown in Figure 6. This was achieved primarily by reducing the center column diameter (Figure 7a) and the center-outer column distance (Figure 7d) (thereby reducing the length and mass of the pontoons connecting the columns) by 3.74% and 5.74%, respectively. The reductions to the platform mass and geometry had the expected consequence of increasing the maximum platform pitch angle (Figure 8), but the optimizer worked to ensure that the upper bound of the constraint (6°) was not exceeded.

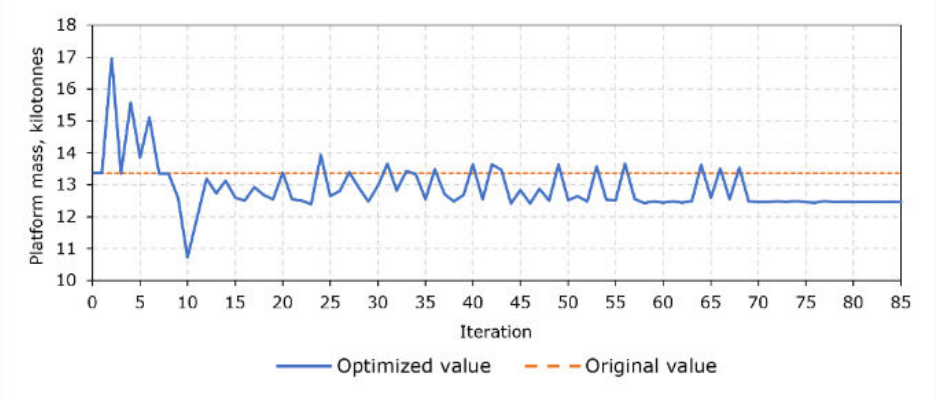


Figure 6 – Iterative changes to the platform mass during the optimization.

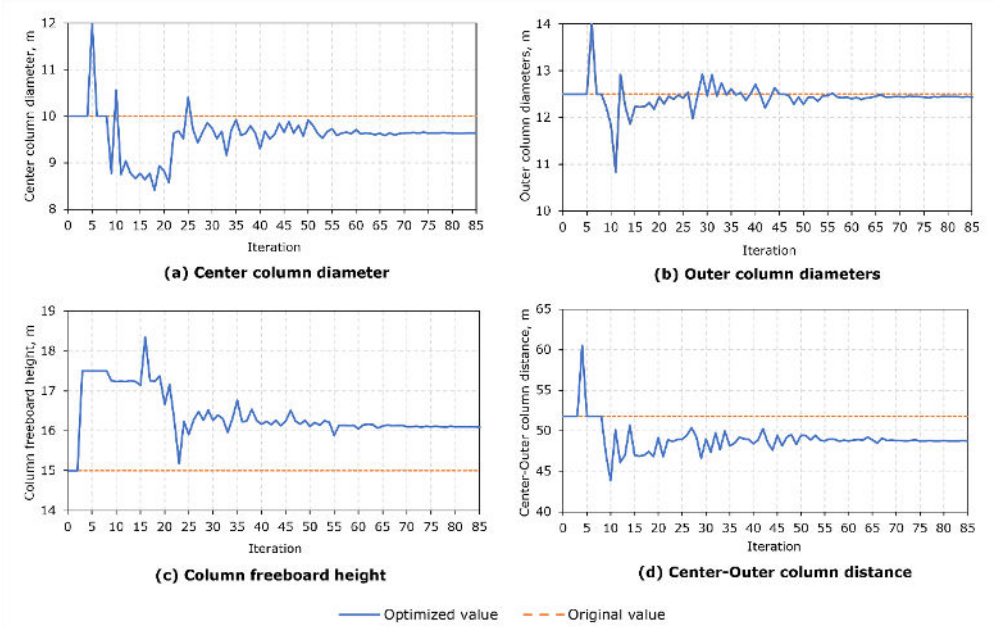


Figure 7 - Iterative changes to the design variables during the optimization.

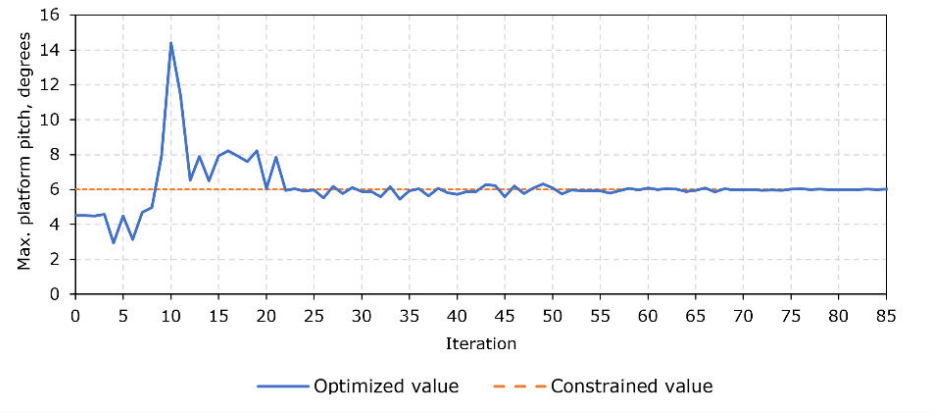


Figure 8 - Iterative changes to the maximum platform pitch angle constraint during the optimization.

A diagram comparing the geometries of the original and optimized platform design is shown in Figure 9. This clearly illustrates the changes made to the center-outer column distances, the center column diameter and the freeboard height.

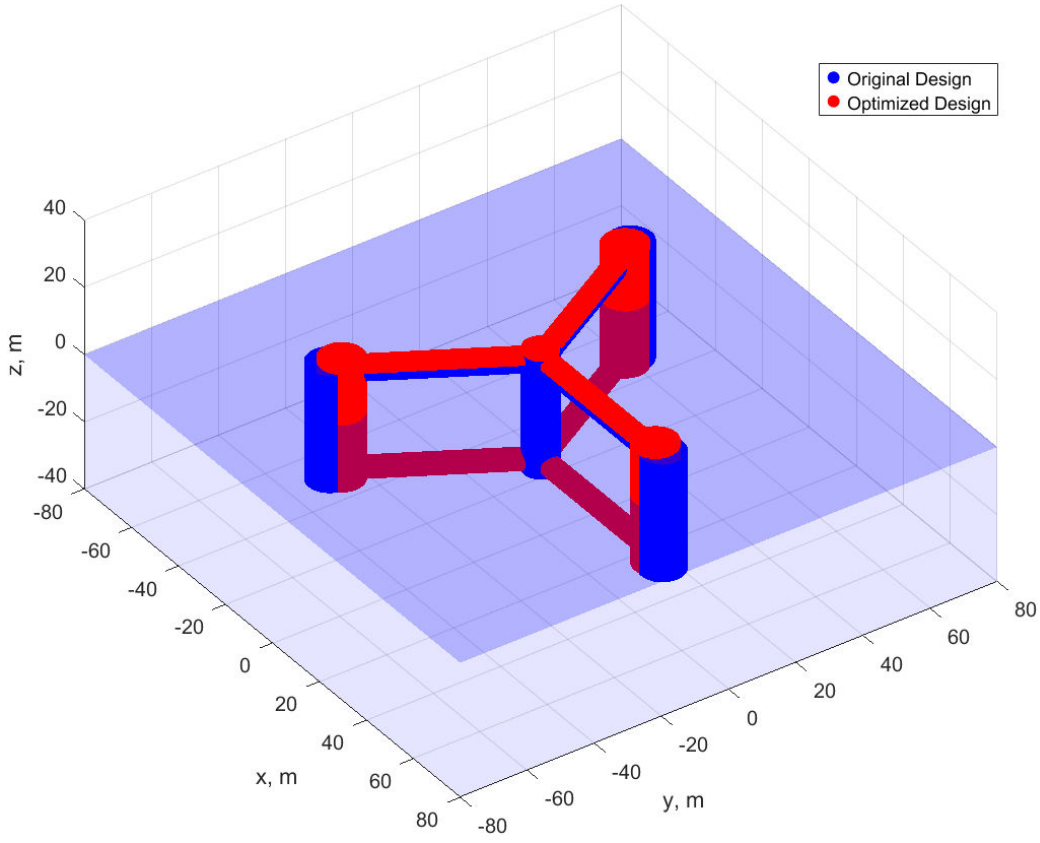


Figure 9 – Comparing the geometries of the original and optimized VolturnUS-S platform.

5. Conclusions

In this deliverable report, an overview of the work carried out in Task 7.1 within WP 7 of the FLOATFARM project is given. Thereby, an integration of the aero-servo-hydro-elastic wind turbine simulation tool HAWC2 into the design and optimization framework WEIS is introduced.

The motivation for such a tool lies in the multi-disciplinary complex problem that is posed to designers and engineers involved in the optimization and design process of FOWTs. WEIS already provides a holistic toolset to carry out multi-disciplinary design analysis and optimization tasks. This toolset is expanded by HAWC2, adding various options and capabilities to the framework, including HAWC2's unique static solver functionality.

A test case was presented for the optimizer where the optimization problem was formulated in such a way that a certain result could be expected based on the given design variables, constraints and merit figure. HAWC2-WEIS performed as expected, reducing the structural mass of a semi-submersible platform.

6. References

[1] T. J. Larsen & A. M. Hansen (2007). How 2 HAWC2, the user's manual. Risø National Laboratory.

[2] J. Jonkman, A. Wright, G. Barter, M. Hall, J. Allison, D. R. & Herber (2021). Functional requirements for the WEIS toolset to enable controls co-design of floating offshore wind turbines. International Conference on Offshore Mechanics and Arctic Engineering (Vol. 84768, p. V001T01A007).

[3] M. Owayjan, A. Houari, R. Achkar, M. Air-Ahmed, M. Tahliati and S. Ferahtia (2024). Robust model-free control of floating offshore wind turbine under high wind & wave conditions. In 2024 International Conference on Control, Automation and Diagnosis (ICCAD) https://doi.org/10.1109/ICCAD60883.2024.10553661.

[4] N. J. Abbas, J. Jasa, D. Zalkind, A. Wright & L. Pao (2023). Control co-design of a floating offshore wind turbine. Applied Energy, 353, 122036. https://doi.org/10.1016/j.apenergy.2023.122036.

[5] J. S. Gray, J. T. Hwang, J. R. R. A. Martins, K. T. Moore, and B. A. Naylor. (2019). "OpenMDAO: An Open-Source Framework for Multidisciplinary Design, Analysis, and Optimization," Structural and Multidisciplinary Optimization.

[6] D. Zalkind and P. Bortolotti (2024). Control Co-Design Studies for a 22 MW Semisubmersible Floating Wind Turbine Platform. Journal of Physics: Conference Series, Volume 2767, System design and multi-fidelity/multi-disciplinary modeling. DOI: 10.1088/1742-6596/2767/8/082020.

[7] IEAWindTask. GitHub - IEAWindTask37/WindIO. GitHub. https://github.com/IEAWindTask37/windIO

[8] WISDEM. Wind Plant Integrated System Design and Engineering Model. GitHub. https://github.com/WISDEM/WISDEM.

[9] ROSCO. GitHub - NREL/ROSCO: a reference open-source controller for wind turbines. GitHub. https://github.com/NREL/ROSCO

[10] YingyiLiu. GitHub - YingyiLiu/HAMS: An open-source computer program for the analysis of wave diffraction and radiation of three-dimensional floating or submerged structures. GitHub. https://github.com/YingyiLiu/HAMS

[11] OpenFAST. GitHub - OpenFAST/openfast: Main repository for the NREL-supported OpenFAST whole-turbine and FAST.Farm wind farm simulation codes. GitHub. https://github.com/OpenFAST/openfast

[12] IEC 61400-3-1:2019 - Wind energy generation systems - Part 3-1: Design Requirements for fixed offshore wind turbines. Tech. Rep. British Standards Institution, London, UK, 2019.

[13] RAFT. GitHub - WISDEM/RAFT: A frequency-domain dynamics model for floating wind turbines. GitHub. https://github.com/WISDEM/RAFT

[14] pCrunch. GitHub - NREL/pCrunch. GitHub. https://github.com/NREL/pCrunch

[15] windio-converter. https://pypi.org/project/windio-converter/0.2.6/

[16] DTUWEC. https://gitlab.windenergy.dtu.dk/OpenLAC/BasicDTUController

[17] F. Meng, A. W. H. Lio and T. Barlas (2020). DTUWEC: an open-source DTU Wind Energy Controller with advanced industrial features. Journal of Physics Conference Series, Volume 1618 022009, DOI 10.1088/1742-6596/1618/2/022009.

[18] E. Gaertner, J. Rinker, L. Sethuraman, F. Zahle, B. Anderson, G. Barter, ... & Viselli, A (2022). IEA wind TCP task 37: definition of the IEA 15-megawatt offshore reference wind turbine. Tech. Rep. NREL, Golden, CO: NREL/TP-5000-75698.

[19] C. Allen, A. Viscelli, H. Dagher, A. Goupee, E. Gaertner, N. Abbas, M. Hall, and G. Barter (2020). Definition of the UMaine VolturnUS-S reference platform developed for the IEA wind 15-megawatt offshore reference wind turbine. Tech. Rep. NREL, Golden, CO); Univ. of Maine, Orono, ME: NREL/TP-5000-75698.

FLOATFARM

NEXT GENERATION FLOATING WIND FARMS

Funded by
the European Union