POLITECNICO DI TORINO

Master Degree course in Energy and Nuclear Engineering

Master Degree Thesis

Techno-Economic Modeling and Analysis of Tidal-to-Hydrogen Energy Systems

Supervisors

Prof. Giuseppe Giorgi

Dr. Emiliano Nelson Gorr

Dr. Manuel Corrales González

Candidate

Lorenzo Vero

Academic Year 2024-2025

Abstract

As a response to the current environmental situation, the power sector shifted its focus to renewable sources to provide an affordable energy supply. However, the development of this sector, is facing, especially in Europe, some difficulties due to the limited available land area. One possible answer to this issue is to develop technologies that exploit marine resources that have great potential but are not yet entirely used. Among the different possibilities offered by the offshore environment, this thesis will focus on tidal energy, first as a stand-alone plant, then as a novel symbiosis as the electricity source feeding an electrolyser for green hydrogen production. The proposed analysis is focused on both the energy production and the economic side. The analysis exploits a computational tool that comprises the cost functions of all the components involved for the calculations of the main economic parameters like CapEx and OpEx, as well as the functions for the calculations of the energy production to obtain the LCoE (Levelized Cost of Energy), an economic parameter fundamental for different energy sources comparison. The analysis focuses on the three main technologies for tidal stream energy production: floating, monopile, and gravity-based substructures, while the resource assessment retrieves the needed data from the Copernicus Marine Service. An analogous procedure is performed to compute the hydrogen production technology costs and obtain the LCoH (Levelized Cost of Hydrogen). Furthermore, in order to validate the methodology, a case study is produced, located in Fall of Warness, Scotland, where a real test site is present for hydrogen production via tidal energy, making it possible to compare the different plant layouts proposed with a real case.

Contents

Abstract 3

List of Figures 6

List of Tables 8

1 Theoretical Introduction 9

1.1 Physics of tides 9

1.2 Tidal energy extraction 11

1.3 Green hydrogen production 15

1.4 Electrolyser Selection for Tidal Energy Integration 16

1.5 Cost Assessment of PEM Electrolyser Systems, Compression, and Hydrogen Storage 17

1.5.1 Electrolyser System 18

1.5.2 Compression System 18

1.5.3 Hydrogen Storage 19

1.6 Tidal - Hydrogen integration 20

2 Python Script 24

2.1 Description of scripts 24

2.1.1 `costs_hydrogen_prod()` 35

2.1.2 Required Python libraries 36

2.2 Theoretical background 36

3 Case Study: Fall of Warness 50

3.1 Resource assessment 51

3.2 Power production analysis 57

3.3 Economic Analysis 59
3.4 Hydrogen integration 62

Conclusions 69

Bibliography 73

A Python Model Source Code 78
A.1 TE_0.py 78
A.2 TE_1.py 81
A.3 TE_2.py 93
A.3.1 mooring_prelay_ODYSSEY.py 127
A.3.2 towing_THOR.py 128
A.3.3 mooring_connection_ODYSSEY.py 129
A.3.4 support_mooring_connection_USKMOOR.py 130
A.3.5 blades_connection_ODYSSEY.py 130
A.3.6 installation_costs_NEPTUNE.py 131
A.3.7 installation_costs_AKER_WAYFARER.py 132
A.3.8 monopile_installation_RAMBIZ.py 133
A.3.9 opex.py 134

List of Figures

1.1 Schematic representation of tidal bulges caused by the Moon's gravity and centrifugal force. [1] 10
1.2 Global distribution of the tidal range resource. [2] 11
1.3 Tidal stream energy converters: (a) HATT, (b) VATT, (c) Oscillating hydrofoil, (d) Ducted turbine, (e) Archimedes screw, (f) Tidal kite. [3] 13
1.4 Sabella D10, 1MW, gravity-based foundations. [4] 13
1.5 Seagen, 1.2 MW, monopile foundations. [5] 14
1.6 Orbital O2, 2MW, floating. [6] 14
1.7 ITEG project layout [7] 23

3.1 Bathymetry of the Fall of Warness area. [8] 51
3.2 Horizontal and vertical components of tidal velocity. 53
3.3 Example of tidal current velocity variation at rotor depth (15 m) over one day. 55
3.4 Time series of tidal current velocity at rotor depth (15 m) for the year 2024. 56
3.5 Velocity distributions with fitted normal probability curves for the two selected TEC configurations. 57
3.6 Annual power output comparison for the different TEC configurations at the Fall of Warness site. 58
3.7 Total CapEx as a function of installed capacity for the different TEC configurations. 59
3.8 Annual OpEx as a function of installed capacity for each TEC configuration. 60

3.9 LCoE trend with installed capacity for the different TEC configurations. 61
3.10 Hydrogen hourly production 64
3.11 Hydrogen production and CF variation in response to electric demand increment 66
3.12 Percentage of power allocation for hydrogen production and for electrical demand 67
3.13 LCOH behaviour in response to electricity demand increment. 68

List of Tables

2.1 Consumer price indexes (CPI) and Conversion of currency for several countries 42

2.2 Typical CapEx ranges for major hydrogen system components. 49

3.1 Clearance constraints and installation depth for tidal energy converters (TECs). 54

3.2 Operational velocity parameters for the different tidal energy converter (TEC) configurations. 57

3.3 Summary of hydrogen system costs and resulting LCoh. 65

3.4 Average Power Output and Capacity Factor for the three tidal technologies. 70

Chapter 1

Theoretical Introduction

This chapter aims to provide an overview of the theoretical background this thesis's work is based. This part defines the context of the thesis, the main fields, and the state-of-the-art of the technologies involved. This review will mainly focus on:

1. Physics of tides

2. Tidal energy extraction

3. Green hydrogen production

1.1 Physics of tides

Tides are natural phenomena based on the gravitational interaction between the Earth, the Sun, and the Moon. The Moon's gravitational force pulls the ocean's water in its direction, creating a bulge of water on the side of the Earth that faces the Moon. Meanwhile, the rotation of the Earth-Moon system around its center of mass generates a centrifugal force, creating a second bulge on the opposite side of the Earth. During each period of rotation of the Earth, most coastal regions experience two high tides and two low tides. [1] This is what makes tides such a promising energy source: their high predictability.

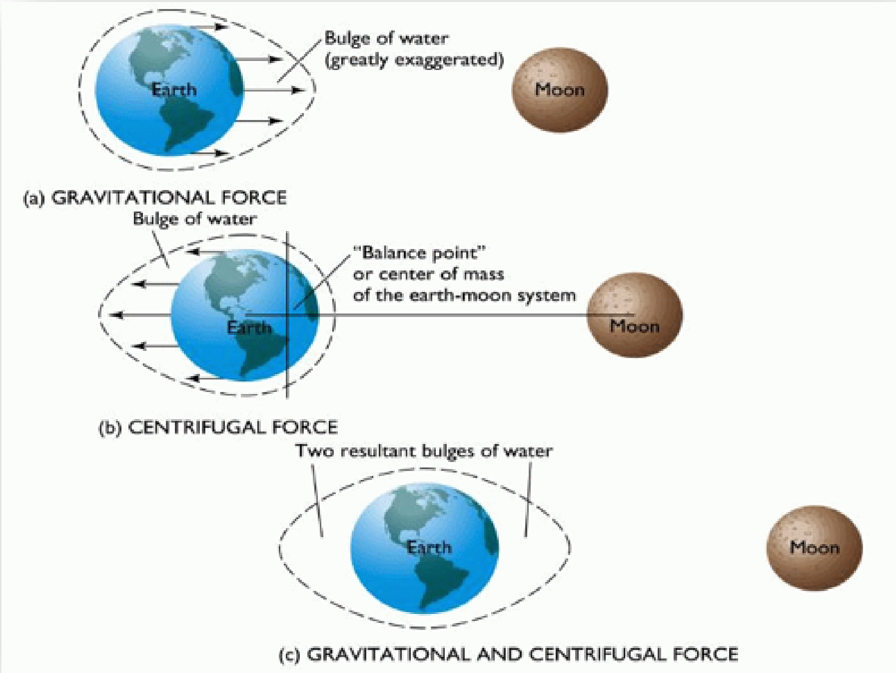


Figure 1.1: Schematic representation of tidal bulges caused by the Moon's gravity and centrifugal force. [1]

The tide's amplitude strongly depends on the Earth, the Sun, and the Moon reciprocal positions; when Earth, Sun, and Moon are aligned their gravitational forces and the centrifugal force sum up producing spring tides, characterized by higher amplitude. When the Sun and the Moon form a 90° angle, their gravitational effects partially cancel out, resulting in neap tides with smaller amplitude. [9]

These factors, of course, affect the amplitude of tides and therefore the potential energy production, as well as the topography of the coastline: ideal sites for tidal energy plants are located in narrow places where the variation between high and low tides is abundant. This happens near natural obstacles as islands, channels, or river mouths.

1.2 Tidal energy extraction

Tidal energy technologies can be divided into two different categories based on the type of energy they exploit:

- Tidal range power plants
- Tidal stream turbines

Tidal range power plants rely on the potential energy due to the difference between high and low tides. To create this difference, a barrage is built to separate the two bodies of water. As the tide rises and falls, the dam blocks the flow, generating a head difference between the outside seawater and the inside retained water. After reaching an optimal level of head difference, the water can pass through the barrage, generating electricity thanks to the turbines placed inside the barrage. With two tidal cycles per day, this head difference is created 4 times each day (as the tide comes in and out). [2]

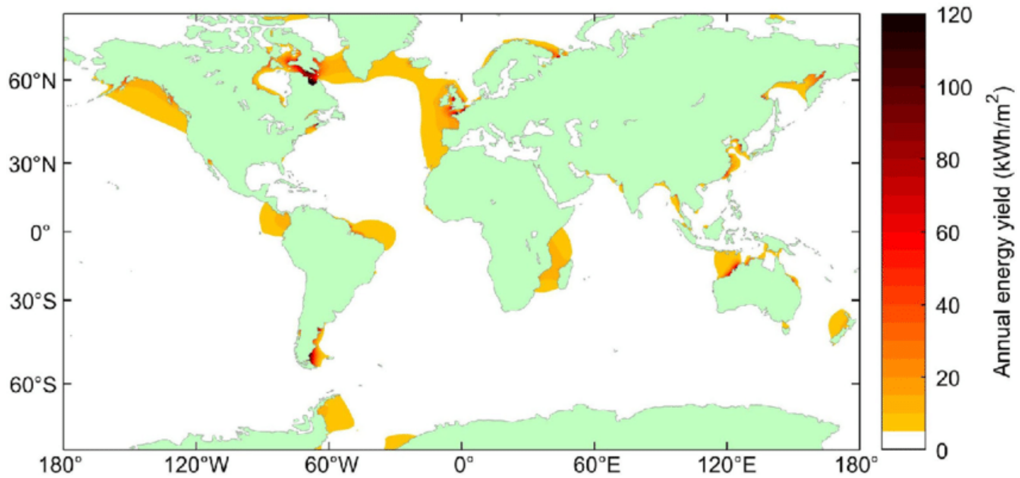


Figure 1.2: Global distribution of the tidal range resource. [2]

On the other hand, tidal stream turbines exploit the kinetic energy of tides and convert it into electricity. These are installed where the variation of the water level caused by tides creates currents of sufficient magnitude. Tidal stream resources are typically greatest in shallow coastal regions where a significant tidal range is present and the flow velocity is intensified by the funneling effect of the surrounding coastline and seabed. Such conditions are commonly found in narrow straits, coastal inlets, around headlands, and between islands, where the movement of water is naturally constrained and accelerated. The functioning and the existing technology are analogous to wind energy, with appropriate modifications; for instance, the density of water is much higher than that of air, thus allowing a significant power generation even with a limited flow velocity.

The main parameters that characterize these systems are the axis orientation, structural configuration, and foundation or mooring design, which allow us to define the following categories: [3]

(a) Horizontal-axis tidal turbine (HATT): they are similar in principle to wind turbines that extract energy from moving air. A horizontal-axis tidal current turbine converts the kinetic energy of free-flowing water into rotational energy, which is then transformed into electrical power through a generator system.

(b) Vertical-axis tidal turbine (VATT): This type of turbine operates on the same general principle as horizontal-axis systems, but its rotors rotate around a vertical axis.

(c) Oscillating hydrofoil: a hydrofoil is attached to an oscillating arm. As the tidal current flows alternately over both sides of the hydrofoil, it generates hydrodynamic lift, causing the arm to oscillate. This motion drives a hydraulic motor, which produces electricity.

(d) Ducted turbine or enclosed tips: these devices consist of a horizontal-axis turbine enclosed within a nozzle or duct. The duct accelerates and channels the water flow, enhancing the turbine's efficiency by increasing the velocity through the rotor. The enclosure also reduces turbulence around the blades and helps to maintain a steady, well-aligned flow.

(e) Archimedes screw: this system employs a helical rotor that operates through a difference in water levels across the screw. As water moves through the helix, it causes the screw to rotate, and this mechanical motion is converted into electrical power.

(f) Tidal kite: it consists of a submerged kite equipped with a small turbine. The kite moves in a controlled trajectory through the water. This motion amplifies the relative flow velocity through the turbine.

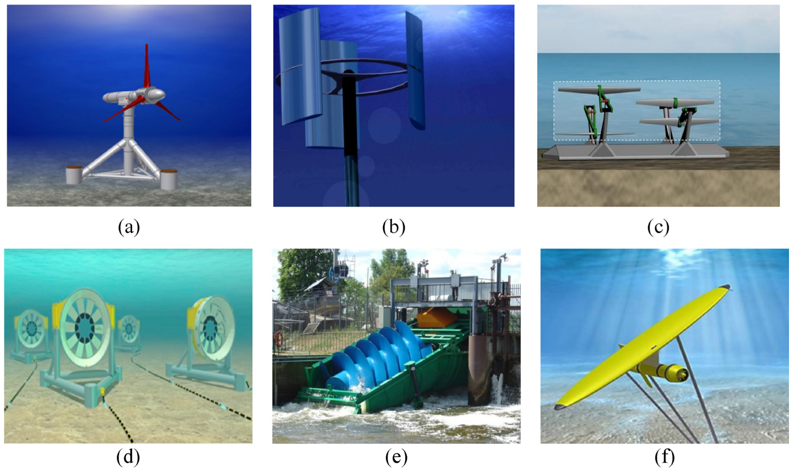


Figure 1.3: Tidal stream energy converters: (a) HATT, (b) VATT, (c) Oscillating hydrofoil, (d) Ducted turbine, (e) Archimedes screw, (f) Tidal kite. [3]

The most widely used and technologically mature designs are based on HATT, whose configuration can vary based on the different installation methods, each characterized by a particular kind of substructure: [10]

- Gravity-based: in this configuration, the system stability is granted by the turbine's own weight, usually equipped with a ballast made of concrete or steel. It is suitable for hard seabed and has the advantage of an easy installation process, even though it requires heavy marine equipment for transport and positioning.

Figure 1.4: Sabella D10, 1MW, gravity-based foundations. [4]

- Monopile: this substructure consists of a large-diameter hollow steel pile that is driven approximately 20–30 meters into the seabed, particularly in areas where the sediment is soft or unconsolidated.

Figure 1.5: Seagen, 1.2 MW, monopile foundations. [5]

- Floating: offers an optimal solution for placing tidal energy devices in deep-water environments. This type of system typically consists of a mounting platform attached to a buoyant vessel that is anchored to the seabed using chains, steel cables, or synthetic mooring lines.

Figure 1.6: Orbital O2, 2MW, floating. [6]

1.3 Green hydrogen production

Hydrogen is the most abundant element on Earth, and it is considered an important energy carrier because of its high energy density (120 MJ/kg), but it always presents itself combined with other elements, as it does in hydrocarbons, where it is combined with carbon, or in water, where it is combined with oxygen. Therefore, the efficient and economically feasible production of hydrogen from these compounds is the real challenge for its large-scale use. Currently, 95% of total hydrogen production is based on fossil fuels through processes as steam methane reforming and coal gasification, emitting around 830 Mt of CO_2 annually. [11] To meet climate targets, production pathways must shift from these carbon-intensive routes toward low-emission or zero-emission alternatives. The main option to achieve this goal is to produce hydrogen via water electrolysis powered by renewable energy sources; this is called green hydrogen. Electrolysis is a process that splits water into hydrogen and oxygen through the following electrochemical reaction:

$$2\,H_2O \longrightarrow 2\,H_2 + O_2 \tag{1.1}$$

Electrolisys presents several variants that can be categorized according to the type of electrolyte and ionic species involved in charge transfer: [11] [12]

- Alkaline Water Electrolysis (AWE): is the most established technology, operating with aqueous KOH or NaOH electrolytes and nickel-based electrodes. It offers long operational lifetimes (up to 60,000 h) and low capital costs but has a slower dynamic response, making it less suited for direct coupling with variable renewable power.

- Proton Exchange Membrane (PEM): employs solid polymer membranes (e.g., Nafion) that conduct protons. They are compact, operate at higher current densities (1–2 A/cm), and respond rapidly to fluctuating power inputs. However, they rely on scarce and expensive noble metal catalysts such as platinum and iridium, which raise costs and limit scalability.

- Anion Exchange Membrane (AEM): still in early development, combines features of both AWE and PEM. They use polymer membranes that transport hydroxide ions, allowing the use of non-noble metal catalysts while maintaining a solid electrolyte architecture. Challenges include limited membrane stability and lower demonstrated lifetimes.

- Solid Oxide Electrolysis (SOE): operating at 700–850 °C, uses ceramic electrolytes such as yttria-stabilized zirconia (YSZ). The high operating temperature allows partial substitution of electrical energy with heat, enabling very high theoretical efficiencies (> 85 %) and potential integration with nuclear or industrial waste heat. However, materials degradation and sealing complexity currently restrict large-scale deployment.

1.4 Electrolyser Selection for Tidal Energy Integration

Among the electrolysis technologies discussed above, alkaline water electrolysers (AWE) and proton exchange membrane (PEM) electrolysers are the most technologically mature. Therefore, a comparative evaluation of these two systems is required to determine which is better suited for integration with tidal turbine power.

Tidal stream generation provides a predictable yet intermittent power profile, with periodic variations in flow velocity and output. When directly coupled to hydrogen production, these fluctuations demand an electrolyser that can ramp rapidly, tolerate frequent start-stop cycles, and maintain efficiency under partial-load operation. Alkaline electrolysis (AWE) and PEM electrolysis represent distinct technological pathways that differ primarily in their electrolyte, operational flexibility, and cost structure.

AWE technology is the most established and commercially available. It offers long-term durability and low capital cost, which make it attractive for large-scale, steady-state hydrogen production [13]. However, AWE systems are generally limited by slower transient response, lower current densities, and potential gas crossover during dynamic operation, which constrains their suitability for variable renewable energy sources [14].

By contrast, PEM electrolysers employ a solid polymer electrolyte membrane (e.g., Nafion®) that conducts protons, enabling high current densities and fast dynamic response. PEM systems can ramp almost instantaneously, operate efficiently at partial loads, and withstand frequent load fluctuations without significant degradation [15, 16]. Although they require noble metal catalysts such as platinum and iridium—resulting in higher capital costs—their compact design, high purity hydrogen output, and ability to operate at elevated pressures make them highly adaptable for integration with renewable power sources [13, 14].

For tidal applications, these characteristics are crucial. Offshore and nearshore tidal turbine systems often impose constraints on space, mass, and maintenance access. PEM electrolysers, due to their compact configuration and capability to produce high-pressure hydrogen directly, minimize the need for additional compression equipment, reducing system footprint and complexity [17]. Moreover, their rapid start-up and shut-down behaviour aligns well with the cyclical power generation profile of tidal currents, improving the overall utilisation factor and reducing the hydrogen levelised cost of energy (LCoE) [14, 16].

While AWE remains advantageous in applications with constant baseload electricity, such as grid-connected or industrial hydrogen plants, its limited dynamic flexibility makes it less suited for direct coupling with marine renewable resources. In contrast, PEM electrolysis has been consistently identified as the reference technology for offshore and variable renewable energy systems, including wind and tidal power integration [18, 19].

Therefore, for tidal turbine-coupled hydrogen production, PEM electrolysis is the preferred choice. Its high efficiency under fluctuating loads, compact and modular design, and superior transient response provide a better match with the operational characteristics of tidal generation, ensuring both technical reliability and economic feasibility for green hydrogen production in marine environments.

1.5 Cost Assessment of PEM Electrolyser Systems, Compression, and Hydrogen Storage

The capital expenditure (CapEx) of a green hydrogen production plant is mainly determined by three components: the electrolyser system, the hydrogen compression unit, and the storage tanks. Each subsystem contributes differently to the overall cost structure, and its economic performance depends on scale, operating pressure, and technological maturity.

1.5.1 Electrolyser System

The electrolyser stack and the associated balance of plant (BOP) represent the largest share of the initial investment. Recent techno-economic assessments show that in 2023 the manufacturing cost of alkaline electrolysers (AEL) lies between 242–388€ /kW, while proton exchange membrane (PEM) electrolysers are higher, ranging from 384–1 071€/kW, depending on stack design and production volume [20].

Projections for 2030 indicate potential cost reductions to 52–79 €/kW for AEL and 63–234 €/kW for PEM systems, achieved through higher current densities, improved material utilisation, and scale economies. A separate National Renewable Energy Laboratory (NREL) report estimates that complete PEM systems—including stack, BOP, power electronics, and enclosure—could reach below 400 €/kW with large-scale manufacturing by 2030 [21].

Operational expenditures (O&M), electricity cost, and the utilisation factor strongly influence the levelised cost of hydrogen (LCoh). Hence, in applications powered by variable renewables such as tidal turbines, PEM electrolysers are preferred due to their rapid dynamic response and efficient operation at partial loads.

1.5.2 Compression System

Hydrogen compression is required to raise the product pressure from the electrolyser outlet (typically 20–30 bar) to storage or distribution levels (350–700 bar). Compressors represent a notable cost element in the system. According to the International Council on Clean Transportation (ICCT), compressor-only capital costs can be approximated as US$ 0.15 per kg H_2 processed annually [22]. When expressed per installed capacity, recent industry data indicate that the specific CapEx of hydrogen compressors ranges between US$ 270–850/kW, equivalent to approximately 240–765 €/kW at current exchange rates [23, 24].

Lower values apply to large-scale (multi-MW) systems, while small-scale or high-pressure designs approach the upper range. A conservative average assumption for design purposes is €250–800/kW installed. These values include mechanical components, motor drive, cooling, installation, and safety systems.

1.5.3 Hydrogen Storage

Hydrogen storage adds another significant cost component and strongly depends on the technology type and operating pressure. Among available options, compressed-gas storage is currently the most commercially mature for decentralised systems.

Ali Saberi Mehr et al. (2024) reviewed the technical and techno-economic performance of different hydrogen storage systems and found that compressed hydrogen storage costs can vary from hundreds to several thousand euros per kilogram of hydrogen capacity, depending on design pressure (300–700 bar), materials, and scale.

To understand the structure of compressed storage systems, hydrogen vessels are classified into four main categories according to their material composition and pressure capability: [25]

- Type I cylinders are made entirely of metals such as stainless steel or aluminium alloys (grades 6061 or 7060). They can withstand pressures up to 300 bar but exhibit a low gravimetric hydrogen density of around 1.7 % in weight.
- Type II vessels feature a metallic liner partially reinforced with a hoop-wrapped composite (fibre + resin). The reinforcement covers only the cylindrical section, offering moderate weight reduction at a slightly higher cost.
- Type III vessels use a fully composite overwrap (hoop and polar winding) with an aluminium liner, supporting 350 bar operation and progressing toward 700 bar. The cost is roughly 10 times higher than Type I or II, but with twice the hydrogen density.
- Type IV cylinders employ polymer liners (polyethylene or polyamide) with composite overwraps. They are lighter than Type III and can operate at similar pressures (300–700 bar). Their main advantage is the ability to eliminate external compression at refuelling stations, enabling direct high-pressure hydrogen transfer.

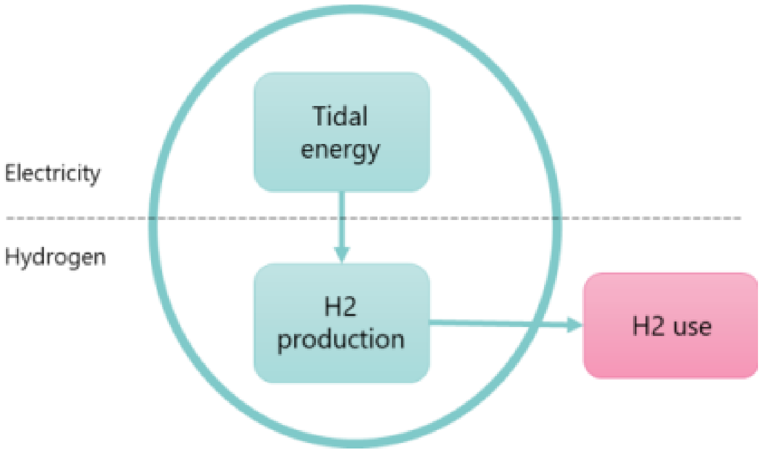
Overall, the PEM electrolyser remains the largest single cost contributor to a green hydrogen system, followed by compression and storage infrastructure. However, all three components are undergoing rapid cost declines. Economies of scale, manufacturing standardisation, and advances in composite materials are expected to significantly reduce both stack and storage costs within the next decade.

1.6 Tidal - Hydrogen integration

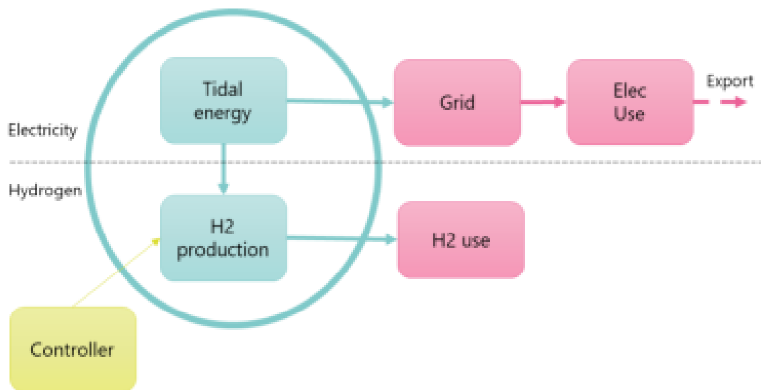
This section provides an overview of the current state of research and real-world applications concerning the integration of tidal energy with hydrogen production systems. The aim is to define the possible system's layout based on existing literature and projects.

The literature has been able to identify three main possible configurations with increasing complexity and grid involvement: [26]

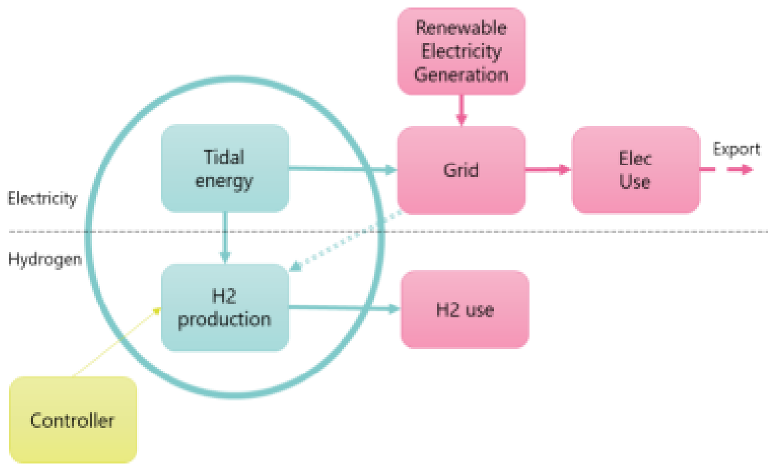
- Closed System: In this configuration, the system operates independently, with no interaction with other electricity sources. The electrolyser produces hydrogen exclusively for a single, consistent type of consumer. The main advantage is that both the tidal turbine and the electrolyser can be sized according to the end-use demand, which remains relatively stable throughout the year. However, this setup is less suitable for applications with fluctuating or seasonal demand, such as heating, unless supported by large-scale energy storage. A key drawback of a closed system is the high level of interdependence between its components, meaning that any malfunction or downtime can directly impact the end use, and implementing contingency measures may be prohibitively expensive. Furthermore, a decline in hydrogen demand could leave the tidal generation system, electrolyser, and associated infrastructure underutilized.



- Tidal generation controlled between electrolyser and the grid: Depending on the chosen control approach, this configuration can maximise the usable energy extracted from the tidal generator and reduce curtailment. In a constrained electricity network, it may also help mitigate fluctuations in electricity prices—by consuming or exporting tidal-generated electricity when prices are high, and producing hydrogen when prices are low. From a cost-based perspective, it is important to note that electrolysers maintain high efficiency even when operating dynamically across a broad range of part loads. Other operational strategies can prioritise maximising operator revenue, increasing hydrogen output, or enhancing energy security and resilience. These strategies are not mutually exclusive and can be integrated to achieve an optimal balance aligned with the overall objectives of the energy system.



- Hybrid marine energy systems: configuration that combines tidal with other renewables. In this configuration, the system is fully integrated with the electrical transmission network, enabling both the import and export of electricity. The local energy system can be managed at the community level with support from network operators. This setup offers greater flexibility by incorporating additional electricity sources, allowing for improved optimisation, energy management, and enhanced resilience in the event of malfunctions or disruptions. Moreover, increased electrification can be achieved without the need for costly upgrades to transmission infrastructure. A grid-connected electrolyser can also be located close to the hydrogen demand point—or even directly on-site—significantly reducing logistics and transportation costs. As with the previous configuration, tidal energy provides a predictable and stable baseload for electrolyser operation, while additional renewable sources can meet extra demand. Electrolysers can absorb excess renewable electricity during periods of low demand and participate in ancillary service markets to help stabilize fluctuating electricity prices. Grid electricity can also be used to increase electrolyser utilization, thereby supporting the growth of hydrogen demand through improved supply reliability.



In practice, several pilot projects have been implemented in the last years. Among them, the most advanced one has been presented by the European Marine Energy Center (EMEC), standing out as a representative case of the second type system, in which tidal energy output is intelligently distributed between a grid connection and a hydrogen electrolyser, demonstrating how such hybrid control can enhance energy flexibility and system efficiency.

At EMEC's Fall of Warness tidal site on Eday, tidal turbines feed variable electricity to a 500 kW PEM electrolyser supplied by ITM Power, producing the world's first tidal-powered hydrogen in 2017 [27]. This hydrogen is compressed and stored on-site, then distributed via tube trailers to Kirkwall, where it is used in fuel cells for port power and local heating, forming a complete island hydrogen loop [28, 29].

Building on this foundation, the ITEG (Integrating Tidal Energy into the European Grid) project, co-funded by Interreg North-West Europe, has further advanced this integration by combining Orbital's 2 MW O2 floating tidal turbine with a 670 kW PEM electrolyser and an Energy Management System (EMS) for optimized power-to-hydrogen conversion and grid interaction [7]. The ITEG layout is hybrid in nature: the tidal turbine supplies power directly to the onshore electrolyser while the EMS coordinates real-time power smoothing and dispatch between grid export, hydrogen production, and on-site load balancing.

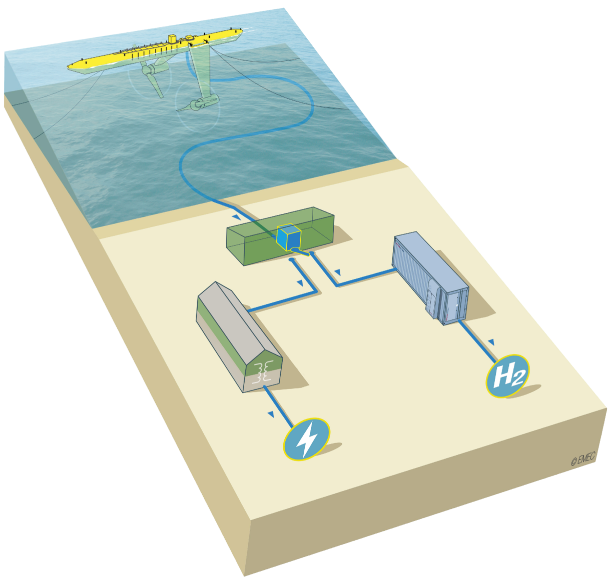


Figure 1.7: ITEG project layout [7]

Chapter 2

Python Script

This section contains the description of the head code made with Python programming for estimating the tidal energy and its descriptive statistics, as well as the estimation of several types of tidal energy converters (TECs) performance indicators, and the estimation of economic indicators. Python programming has been chosen since it has a wide versatility, facilitates the connectivity with other programming languages, including establishing links to web pages and data sharing [30].

This Python script calculates the CapEx (Capital Expenditures) and OpEx (Operational Expenditures) for a tidal energy generation system using Gravity-Based (GBS), Floating, and Monopile structures.

Overall, this chapter is structured as follows: Section 2.1 details the coding which evaluates the tidal energy and its deployment, followed by Section 2.2 where the theory behind the estimations is described.

2.1 Description of scripts

Each technology has a dedicated script that invokes different scripts, or user-created functions, to perform the subtasks assigned to it.

`Main code [·]` ⇒ `User-created Scripts[-]` ⇒ `Particular functions[*]`

- `TE_0.py`
  - `TE_1.py`: Technology performance evaluation
    - `plotLocation()`

    * `distance_from_shoreline()`
    * `tidalCurr_analysis()`
    * `hydrogen_production_dynamic`
  - `TE_2.py`: Techno-economic optimization
    * `calculate_irr()`
    * `economic_parameters_Floating()`: This section relies on a set of functions for accurately estimating the cost inputs for the floating-type TECs. Such functions are: $compute\_mooring\_prelay\_costs\_Odyssey()$, $compute\_mooring\_connection\_costs\_Odyssey()$, $compute\_blades\_connection\_costs\_Odyssey()$, $compute\_towing\_costs\_Thor()$, and $compute\_support\_mooring\_connection\_costs\_Uskmoor()$, and $compute\_opex()$.
    * `economic_parameters_GBS.py`: the required functions in this script are: $compute\_installation\_costs\_Neptune()$,$compute\_installation\_costs\_Aker\_Wayfarer()$, and $compute\_opex()$.
    * `economic_parameters_Monopile()`: the required functions in this section are: $compute\_installation\_costs\_Rambiz()$, $compute{}_installation\_costs\_Aker\_Watfarer()$ and $compute\_opex()$.
    * `costs_hydrogen_prod()`
    * `costs_computations()`
    * `pltinstlld_cpcty()`

`TE_0.py`

This is the main script. Initially, it defines the preamble and general parameters that will be used in the different scripts, such as coordinates of assessed locations, the selected TEC, and more settings as the saving and input data paths. With respect to the spatial interpolation, it is determined at the `WP_1.py`, as the method 'nearest' to extract data from the *xarray* Python library. Moreover, the variable 'TEC_sel' defines the selected Tidal energy converter among the available options, also written as a comment at the end of such variable.

Additionally, this script loads the CMEMS data and calls other functions contained in the scripts `TE_1.py` and `TE_2.py`. Tidal current information can be found at the CMEMS Marine Data website. The variables required for the assessment can be found in the section *Surface currents, hourly*, and extract the tidal current components 'utide' and 'vtide'. The bathymetry data have been retrieved from GEBCO.

With respect to the features of the TECs, since there is a lack of openly available information, only some TECs can be assessed through this script. Specifically, the Floating TEC Orbital O2, Floating Tocardo-T2, and the Monopiles Seagen S-1 and Seagen S-2.

`TE_1.py`

This script contains the different functions needed to determine the tidal current and tidal energy characteristics, computing all the parameters useful to assess the energy production and the hydrogen one. Initially, a function is defined to map the assessed location according to the nearest model coordinates based on the user's requested coordinates. Then, the function `distance_from_shoreline()` computes the minimum distance from the TEC deployment to the shoreline, a distance that will be used in the following economic analysis. The function `tidalCurr_analysis()` uses the tidal current information to firstly determine the upstream current in front of the submerged turbine, and the subsequent downstream, or rated current, as well as the tidal power produced by each turbine. Furthermore, the TEC performance indicators, Annual Electricity Production (AEP) and Capacity Factor (CF), are also estimated. From this information, a series of graphical results is obtained and saved in the designated storage folder.

The `hydrogen_production_dynamic()` function takes as input the values obtained by the previous analysis and uses them to calculate the possible hydrogen production in two different cases: in the first scenario the entire energy produced by the tidal plant is used to power the electrolyser, which produces hydrogen; in the second scenario, a predefined user demand must be satisfied by the tidal plant; only the residual (surplus) energy is subsequently directed to the electrolyser. The sizing of the hydrogen production facilities is determined as a function of the total energy produced by the plant. Also in this case, performance indicators are estimated, such as the Capacity Factor (CF), the total hydrogen production (ton/year), and the percentage distribution of the tidal energy utilized for demand fulfillment and hydrogen generation.

`TE_2.py`

In this script six functions are established. The first function simply estimates the Internal Rate of Return (IRR), as a function of the TEC deployment cashflows and rate of discount (r). The second to fourth functions are responsible for estimating the economic costs related to the Floating, GBS, and Monopile types of TECs; these, in turn, make use of other subroutines for internal calculations. In this contex,t a Hybrid Approach is employed to estimate the TEC-related costs. In this context, Optimal Economic Analysis from the HyA optimization combines the technological explicitness of bottom-up models with the economic comprehensiveness of top-down mode [31], i.e., the major part of costs are estimated directly from catalogs and their intrinsic costs, while some other costs are estimated as a percentage or weighted costs of other WEC components' costs.

After this part, the function `costs_hydrogen_prod()` is defined, with the purpose of calculating the costs related to the electrolysis system. The data needed to complete this part has been retrieved from the literature, assigning a cost to each component strictly related to its size.

The fifth function `costs_computations()` invokes and executes the three previous functions according to the type of selected TEC. This function performs the economic analysis for an increasing number of turbines through a repetition cycle; as a result, the different values of LCoE, CapEx, and installed capacity are accumulated in a vector, to be used in the `pltinstlld_cpcty()`. Finally, `pltinstlld_cpcty()` function generates a 3D map of the Levelised Cost of Energy and CapEx variation, according to the installed capacity.

The following paragraphs will describe in detail the structure of the four initial functions, focusing, at first, on the economic parameters involved in tidal energy generation, and then on the subsequent production of green hydrogen:

`economic_parameters_GBS()`

This Python script calculates capital expenditures (CapEx) and operational expenditures (OpEx) for gravity-based structures (GBS). This function takes some inputs that can be environmental or technical and uses them to compute several parameters that are useful for the cost calculation.

The main function `economic_parameters_GBS.py` takes several input parameters:

- Site parameters: water depth, distance from shore
- Turbine specifics: number of blades, rated current, rotor diameter, rated power
- Array layout: turbines per structure, number of structures, number of rows, and columns
- Electrical parameters: export voltage, number of export cables

The function calculates the mechanical properties of the rotor and drivetrain, including torque, rotational speed, thrust force, and power conversion values based on rated current and rotor geometry. It adjusts historical cost data to 2024 values using Consumer Price Index (CPI) ratios for various countries and converts from USD to EUR.

Components

Computes the costs of each component:

- Blades, hub, pitch system, yaw system, brake, gearbox, generator, shaft, and main bearings
- Electrical components: power converters, transformers, switchgear, wet connectors
- Nacelle and support structures
- Foundation based on thrust force and steel price
- Cost of array and export cables based on cross-sectional area and voltage
- Laying costs for subsea cables and drilled ducts
- Umbilical and wet connector costs

Installation

For the installation part, the costs are calculated by using two external functions which summarize the costs for the operations done by two different vessels: `compute_installation_costs_Aker_Wayfarer()`, which computes the costs for the turbines' installation, and `compute_installation_costs_Neptune()` to model the costs for substructures' installation.

The calculation considers key vessel characteristics, such as crane capacity, installed power, available deck space, transit speed, and crew requirements. These parameters are used to determine the vessel charter rate, the number of substructures or turibines that can be transported per trip, and, consequently, the total number of trips required to complete the installation campaign. The function also incorporates the installation time per structure, which directly affects both vessel rental duration and crew costs. The functions compute also the fuel cost for the trips necessary to complete the installation.

CapEx Calculation

Aggregates all costs, including:

- Structural components and electrical systems
- Cables and connectors
- Installation and offshore base setup
- Development expenditure (DevEx)

OpEx calculation

The OpEx final value is obtained by the sum of maintenance costs and insurance costs considering a project lifetime of 30 years. Maintenance costs are calculated by using an external function, `compute_opex()`, which summarizes the costs of the vessel that performs the maintenance operations for the various components; in particular, it calculates the costs for: spare parts, vessel operation, and workers. Insurance costs are calculated as a percentage of the CapEx.

Output

The function returns:

- Total CapEx in Euros
- Total OpEx in Euros

`economic_parameters_Floating()`

This Python script calculates the capital expenditures (CapEx) and operational expenditures (OpEx) for floating tidal turbines. This function takes several inputs that can be environmental or technical and uses them to compute several parameters useful for the cost calculation. The inputs needed by the function are:

- Site parameters: water depth, distance from shore
- Turbine specifics: number of blades, rated current, rotor diameter, rated power
- Array layout: turbines per structure, number of structures, number of rows, and columns
- Electrical parameters: export voltage, number of export cables

The function calculates the mechanical properties of the rotor and drivetrain, including torque, rotational speed, thrust force, and power conversion values based on rated current and rotor geometry. It adjusts historical cost data to 2024 values using Consumer Price Index (CPI) ratios for various countries and converts from USD to EUR.

Components

Computes the costs of each component:

- Blades, hub, pitch system, brake, gearbox, generator, shaft, and main bearings
- Electrical components: power converters, transformers, switchgear, wet connectors

- Nacelle and support structures
- Foundation based on thrust force and steel price
- Cost of array and export cables based on cross-sectional area and voltage
- Laying costs for subsea cables and drilled ducts
- Umbilical and wet connector costs
- Mooring system: chain and anchor costs are based on the dimensions, density, and steel price. Typically, 4 mooring lines are necessary to keep one platform steady.

Installation

For the installation costs computation, the following external functions are used:

- `compute_mooring_prelay_costs_Odyssey()` computes the costs for the mooring system prelaying.
- `compute_mooring_connection_costs_Odyssey()` computes the costs for mooring connection.
- `compute_support_mooring_connection_costs_Uskmoor()` computes the costs of the operations of a support vessel during mooring connection.
- `compute_towing_costs_Thor()` computes the costs of the operations of a tug vessel towing the on-shore assembled structure.
- `compute_blades_connections_costs_Odyssey()` computes the costs for the connection of the blades.

The calculation considers key vessel characteristics, such as crane capacity, installed power, available deck space, transit speed, and crew requirements. These parameters are used to determine the vessel charter rate, the number of structures that can be transported per trip, and, consequently, the total number of trips required to complete the installation campaign. The function also incorporates the installation time per structure, which directly affects both vessel rental duration and crew costs. The functions compute also the fuel cost for the trips necessary to complete each step of the installation procedure

CapEx Calculation

Aggregates all costs, including:

- Structural components and electrical systems
- Cables and connectors
- Installation and offshore base setup
- Development expenditure (DevEx)

OpEx calculation

The OpEx final value is obtained by the sum of maintenance costs and insurance costs considering a project lifetime of 30 years. Maintenance costs are calculated by using an external function, `compute_opex()`, which summarizes the costs of the vessel that performs the maintenance operations for the various components; in particular, it calculates the costs for: spare parts, vessel operation, and workers. Insurance costs are calculated as a percentage of the CapEx.

Output

The function returns:

- Total CapEx in Euros
- Total OpEx in Euros

`economic_parameters_Monopile()`

This Python script calculates the capital expenditures (CapEx) and operational expenditures (OpEx) for monopile tidal turbines. This function takes several inputs that can be environmental or technical and uses them to compute several parameters useful for the cost calculation. The inputs needed by the function are:

- Site parameters: water depth, distance from shore
- Turbine specifics: number of blades, rated current, rotor diameter, rated power

- Array layout: turbines per structure, number of structures, number of rows, and columns

- Electrical parameters: export voltage, number of export cables

The function calculates the mechanical properties of the rotor and drivetrain, including torque, rotational speed, thrust force, and power conversion values based on rated current and rotor geometry. It adjusts historical cost data to 2024 values using Consumer Price Index (CPI) ratios for various countries and converts from USD to EUR.

Components

Computes the costs of each component:

- Blades, hub, pitch system, brake, gearbox, generator, shaft, and main bearings

- Electrical components: power converters, transformers, switchgear, wet connectors

- Nacelle and support structures

- Monopile cost based on its mass and steel price

- Cost of array and export cables based on cross-sectional area and voltage

- Laying costs for subsea cables and drilled ducts

- Umbilical and wet connector costs

Installation

For the installation costs computation, the following external functions are used:

- `compute_installation_costs_Rambiz()` which computes the costs for the monopile's installation.

- `compute_installation_costs_Aker_Wayfarer()` which computes the costs for the turbines installation.

The calculation considers key vessel characteristics, such as crane capacity, installed power, available deck space, transit speed, and crew requirements. These parameters are used to determine the vessel charter rate, the number of substructures or turibines that can be transported per trip, and, consequently, the total number of trips required to complete the installation campaign. The function also incorporates the installation time per structure, which directly affects both vessel rental duration and crew costs. The functions compute also the fuel cost for the trips necessary to complete the installation.

CapEx Calculation

Aggregates all costs, including:

- Structural components and electrical systems
- Cables and connectors
- Installation and offshore base setup
- Development expenditure (DevEx)

OpEx calculation

The OpEx final value is obtained by the sum of maintenance costs and insurance costs considering a project lifetime of 30 years. Maintenance costs are calculated by using an external function, `compute_opex()`, which summarizes the costs of the vessel that performs the maintenance operations for the various components; in particular, it calculates the costs for: spare parts, vessel operation, and workers. Insurance costs are calculated as a percentage of the CapEx.

Output

The function returns:

- Total CapEx in Euros
- Total OpEx in Euros

2.1.1 `costs_hydrogen_prod()`

This function computes the economic parameters related to the production of green hydrogen, using tidal energy as the only source.

Input

The functions receive the following inputs:

- H_2 produced (`H2_Total`): represent the annual hydrogen production, calculated in `TE_1` in kg.
- Electrolyser nominal power (`PEM_nom`): Nominal power of the PEM stack calculated in `TE_1` as a function of the total AEP of the tidal plant, expressed in kW.
- Compressor nominal power (`COMP_nom`): Nominal power of the compressor (in kW) calculated in `TE_1`. It depends on the average mass flow of hydrogen downstream of the electrolyser.
- Storage nominal capacity (`STORAGE_nom`): Capacity of the tanks for hydrogen storage (in kg) calculated in `TE_1` based on the daily hydrogen production.
- CapEx of the Tidal plant.
- OpEx of the Tidal plant.

Cost of Components

The three main components contributing to the cost of this section of the plant are:

- PEM Electrolyser
- Compressor
- Storage tanks

Their cost functions are based on the existing literature, and they are all dependent on the size of the components.

Output

The function returns:

- Cost of the three components
- CapEx in Euros of the electrolysis system
- OpEx in Euros of the electrolysis system
- Levelized Cost of Hydrogen (LCoH) in €/kg.

2.1.2 Required Python libraries

The programmed routines presented in the description above require various Python libraries to be installed in the Python environment. These libraries are:

- cartopy
- datetime
- geopandas
- matplotlib
- math
- mpl_toolkits
- numpy
- numpy_financial
- os
- pandas
- pyproj
- plotly
- scipy
- seaborn
- sys
- xarray

2.2 Theoretical background

The tidal power (P) can be estimated by the expression(2.1) [32]:

$$P = \frac{1}{2}\rho A\, C_p U^3\,, \tag{2.1}$$

where ρ represents the water density, equal to 1025 kg/m^3, A denotes the rotor cross-sectional area, C_P is the power coefficient usually equal to 0.4, and U is the tidal current. Once the dataset for extracted tidal power is obtained, the annual energy production can be calculated using (2.2),

$$AEP = P_{yearly\,mean} * 8760\,, \tag{2.2}$$

where $P_{yearly\,mean}$ represents the average power extracted over the year, in kW, and 8760 is the total number of hours in a year. The capacity factor (CF) can also be determined; it indicates the amount of power extracted relative to the rated capacity (P_{rated}) [33] of the device as defined in (2.3),

$$CF = \frac{P_{yearly\,mean}}{P_{rated}} \tag{2.3}$$

In the direct cost estimation, a Bottom-Up approach (BuA) is used, which aims to make direct estimates based on the costs associated with each task involved in tidal energy deployment [31]. The direct costs used to estimate CapEx depend mainly on the electrical components, blades, turbine hub, pitch, nacelle, mooring system, and foundation structure. The foundation for the turbine structure determines the type of transport and equipment required for its installation. In this implementation, multiple aspects are considered in estimating direct, initial, and operating costs. The analysis incorporates fixed costs tied to installation vessels and support structures, which vary by TEC type [34]:

1. Floating TECs

   - Mooring Systems: Chains, anchors, and shackles (steel cost: 0.5 –2.5 €/kg).
   - Installation Vessels [35, 36]:
     Odyssey: Pre-lays mooring lines and connects structures.
     Thor: Handles towing operations.
     Uskmoor: Provides support during mooring connections.

   - Blade Installation: Specialized vessels for turbine assembly.

2. GBS TECs

- Gravity-Based Foundations: Mass scales with thrust force (steel cost: 0.8 €/kg)
- Installation Vessels:
  Neptune: Deploys GBS foundations.
  Aker Wayfarer: Installs turbines atop foundations.

3. Monopile TECs

- Monopile Foundations: Steel tubes (cost: 1.2 €/kg) sized to withstand thrust and bending moments.
- Installation Vessels:
  Rambiz: Drives monopiles into the seabed.
  Aker Wayfarer: Turbine installation.

Moreover, assumptions and scalability considerations have been taken into account [37]:
- Vessel Costs: Fuel prices (515€/ton) and labor rates (50€/hour) are fixed. Inflation adjustments are applied to historical cost data.
- Economies of Scale: Projects with multiple structures benefit from reduced per-unit costs (e.g., shared cables, bulk installation).
- Electrical Infrastructure: Costs scale with voltage, current, and cable length (100–282€/m for installation).

Regarding technology towing, vessel-related costs for tidal energy converter installations using a combination of fixed parameters and derived formulas. The bollard pull, a measure of a vessel's towing capacity, is set at 78 tons, while the installed power of the vessel is 4,700 kW. The vessel operates at two speeds: 5 km/h during towing operations and 20 km/h under normal conditions. Labor costs include three workers, each costing 50 € per hour [38]. The vessel charter rate is calculated linearly based on bollard pull, expressed as 508.57 € per ton of bollard pull minus 32,186.00 €, resulting in a daily rate. Fuel costs are derived from the vessel's installed power, fuel price (515 €/ton), and an efficiency factor of 0.8, scaled to a daily consumption:

$$\text{Cost}_{fuel} = \frac{Installedpower_{vessel} \cdot fuel_{price} \cdot 0.8 \cdot 210 \cdot 24}{10^6} [\text{Millions of €}] \tag{2.4}$$

The total vessel cost per day (C_{vessel}) combines the charter rate and fuel costs. For towing operations, two vessels are required, and their total cost depends on the round-trip distance from shore, accounting for both towing and normal speeds:

$$\text{Cost}_{towing} = 2\, C_{vessel} \left(\frac{d_c}{V_{v,\,towing} \cdot 1000 \cdot 24} + \frac{d_c}{V_{v,\,normal} \cdot 1000 \cdot 24} \right) \tag{2.5}$$

where d_c is the distance from the tidal energy converter to the nearest location over the shoreline, $v_{v,towing}$ is the vessel mean speed during towing, and $v_{v,normal}$ is the vessel mean speed during regular transportation. This value is adjusted for inflation. Labor costs for the operation are computed separately, based on the total time spent traveling at both speeds:

$$\text{workers}_{total\ cost} = 3 \cdot 50 \cdot 24 \left(\frac{d_c}{V_{v,\,towing} \cdot 1000 \cdot 24} + \frac{d_c}{V_{v,\,normal} \cdot 1000 \cdot 24} \right) \tag{2.6}$$

These calculations integrate engineering assumptions (e.g., fuel consumption at 210 g/kWh) and operational constraints (e.g., speed differentials) to provide a realistic cost structure for marine logistics. The approach ensures that vessel capabilities, fuel efficiency, and labor are proportionally reflected in the total installation expenses. The final adjusted costs are used in broader economic evaluations, such as LCoE and IRR, to assess project feasibility.

To mention one case on tidal installation, it describes the considerations for the GBS, Floating, and Monopile-based turbines. The economic analysis for GBS tidal turbines incorporates a comprehensive cost model that accounts for turbine components, support structures, electrical infrastructure, and installation logistics. The foundation mass is calculated proportionally to the thrust force exerted by tidal currents, with a coefficient of 1.3726 tons per kN of thrust, multiplied by a steel price of €0.8 per kg for the platform. The thrust force itself derives from hydrodynamic principles, computed as 0.5 times seawater density (1025 kg/m^3) multiplied by the square of flow speed, rotor area (πr^2), and a thrust coefficient of 0.9, then converted to kN. Turbine component costs follow engineering scaling laws. Blade costs scale with rotor diameter (D) raised to the power of 2.7, at 40€ per meter of blade length, adjusted for Spanish inflation. The pitch system cost follows a similar diameter-dependent power law ($2.28 \times 0.2106 \times D^{2.6578}$), while the yaw system cost correlates with the bending moment (thrust $\times$ cover diameter/2 $\times$ safety factor of 3), where cover diameter is 13.33% of rotor diameter. The yaw bearing mass is derived from empirical relations involving the moment and diameter, with a fixed cost multiplier.

Electrical infrastructure costs include power converters (79€/kW), transformers (11,879€/kVA), and switchgear (14,018€ per kV of export voltage), all adjusted for U.S. inflation and converted from dollars to euros. Cable costs depend on current-carrying capacity (CSA) and voltage, with array cables priced at 200€/m multiplied by dimensionless factors for CSA (nCSA = 0.6553 + 0.0035 $\times$ CSA) and voltage (nV). Export cables use analogous pricing but include a 30% premium for umbilicals. Installation costs for cables combine laying (100€/m) and drilled duct (282€/m) methods, with one-third of export cables assumed to require drilling. Installation employs specialized vessels: Neptune for GBS foundation deployment and Aker Wayfarer for turbine installation. Their costs are computed externally but include fuel expenses (515€/ton) and labor (50€/hour for workers). Onshore preparation costs include six workers assembling components (blades, etc.) at 50€/hour per worker, with one hour allocated per component.

The total capital expenditure (CapEx) aggregates these elements: nacelle arrays (scaled by structure count), foundations, connectors (200,000 € per wet connector), electrical systems, cables, and installation. A 5% development cost (devex) is added proportionally. Operational expenditures (OpEx) include insurance (1% of CapEx) and maintenance (15% of component costs), with economies of scale applied for multi-turbine structures via a logarithmic reduction factor. Key financial metrics like LCoE and IRR are derived from these cost streams, energy production (AEP), and project lifetime cashflows, ensuring a holistic assessment of GBS tidal energy projects.

The installation process for floating tidal turbines introduces distinct logistical and cost considerations compared to grounded-based systems (GBS). Floating structures require specialized mooring systems and marine operations, with costs driven by chain and anchor deployment, turbine towing, and offshore assembly.

The mooring system for each floating structure consists of four lines, each comprising chains with a diameter of 0.076 meters, sized to withstand hydrodynamic loads. Chain length scales with water depth at a scope ratio of 5 (length-to-depth), and anchor weight is fixed at 140 tons per line. The chain mass is calculated from its linear density (21,900 × square diameter kg/m), yielding a material cost of 0.5 € /kg for steel. Anchors add 70,000 € per line (140 tons × 0.5 €/kg × 1,000), resulting in a total mooring cost from 1.2 to 1.5 million € per structure, depending on depth.

Installation involves three primary vessel operations [39]: the Odyssey for prelaying mooring lines (22 hours per line) and connecting anchors (12 hours each), the Thor for towing turbines at 5 km/h (with return trips at 20 km/h), and the Uskmoor for supporting mooring connections. Towing costs incorporate fuel consumption (515 €/ton) and labor (3 workers at €50/hour), while blade assembly is performed by the Odyssey at 6 hours per turbine. The total installation cost aggregates these steps, including onshore preparation (6 workers assembling components at 50€/hour). The monopile foundation system presents a third distinct approach for tidal energy converters, combining elements of both floating and GBS systems while introducing unique engineering challenges. The monopile design features steel tubes with a standard diameter of 3.5 meters, sized to withstand the combined thrust forces from all turbines on the structure. The total thrust calculation incorporates a safety factor of 2.6 against the yield strength of steel (248 MPa), with wall thickness optimized through the formula:

$$D_{in} = D_{outer} \left(1 - \frac{32M}{\pi D^3 \sigma_{allowable}}\right)^{0.25} \tag{2.7}$$

where M represents the bending moment from rotor thrust at 30m height. This yields a steel mass between 150-400 tons per monopile, priced at 1.2€/kg. Installation relies on two specialized vessels: the Rambiz for driving monopiles (2.5 days per structure) and the Aker Wayfarer for turbine mounting (1 hour per turbine). The Rambiz's costs incorporate fuel consumption at 515€/tonne and scale with distance from shore, while the Aker Wayfarer's operations account for rotor diameter impacts on installation time. Electrical infrastructure mirrors the GBS approach but eliminates array cables, focusing instead on export cables with 33% drilled duct installation at 282€/m.

Key differences from other allocation systems are listed below:

- Foundation Design: Monopiles require precise geotechnical adaptation, with height calculated as half of the rotor height plus the water depth, contrasting with floating systems' depth-independent mooring.

- Material Efficiency: The crossarm mass (32.09 kg/kN of thrust) and tubular design provide 20-30% steel savings versus GBS foundations.

- Installation Speed: While faster than GBS deployment (2.5 vs 1.5 days), monopile driving creates more seabed disturbance than floating installations.

- Operational Advantages: Benefit from 30% repair time reductions like floating systems, but with GBS-like accessibility for maintenance [32].

The economic model applies these adjustments to component costs, for instance, blade costs use Spain's 1.211 ratio, while power converters apply both the USA 2002 ratio (1.741) [32] and currency conversion (see Table 2.1). Regarding the blades,

Table 2.1: Consumer price indexes (CPI) and Conversion of currency for several countries

Costs basis	Original CPI	2024 CPI	Conversion ratio	Currency adjusment[†]
USA 2002	179.90	313.10	1.741	x 0.92 (USD to EUR)
USA 2021	270.97	313.10	1.156	x 0.92 (USD to EUR)
Spain 2017	95.00	115.00	1.211	NA
England 2013	98.52	133.40	1.354	NA
Ireland 2015	82.80	100.50	1.214	NA
Denmark 2020	103.40	118.70	1.148	NA
Europe 2020	106.10	130.60	1.231	NA

Reference: Organisation for Economic Co-operation and Development (OECD) (2025) [40]

their approximate cost is related to the action diameter of blades D (in m) [41]:

$$\text{Cost}_{blade} = 40\ (0.5 \cdot D)^{2.7} \tag{2.8}$$

Likewise, the turbine's hub cost is estimated similarly as follows:

$$\text{Cost}_{hub} = 1000 \cdot \frac{D}{2} \tag{2.9}$$

The pitch system is an important component of the turbine because it allows the blades to rotate in order to reach the optimal orientation at different flow speeds so that the maximum energy can be extracted [42]. This cost is estimated by:

$$\text{Cost}_{pitch} = 2.28\left(0.2106 \cdot D^{2.6578}\right) \tag{2.10}$$

Additionally, the main bearings are components needed for supporting and guiding the main shaft of the turbine and providing low friction. After having estimated the thrust force over the tidal turbine, it is possible to define the bearing mass by:

$$\text{Mass}_{bearing} = 3.1771\, F_T - 132.26 \tag{2.11}$$

where the thrust force F_T is:

$$F_T = \frac{1}{2}\rho\, C_T A U^2 \tag{2.12}$$

with C_T the thrust coefficient (for a wind turbine, equals 0.8), ρ is the air density, A is the rotor area, and U is the wind speed. Then, the bearing cost in US dollars, considering a $M_{bearing}$ in kg, is determined by:

$$C_{bearing} = 2\, M_{bearing} \cdot 17.6 \tag{2.13}$$

On the other hand, the yaw system is used to rotate the axis of the turbine so that the flow direction is directly facing the rotor, so that the maximum power can be extracted. Most of the tidal current flows are approximately bidirectional, which means that ebb and flow are 180°. To estimate its costs, it is first required to determine the mass of yaw bearings [43]:

$$M_{yaw\ bearing} = 0.0152\left(\frac{M_{max}}{D_{yaw}} - 36\right)^{1.499} \tag{2.14}$$

being M_{max}, the maximum applied moment on the bearings in kN m, and D_{yaw} is the diameter of the yaw bearing. Thus, the cost of the yaw bearing is obtained by the following equation:

$$C_{bearing} = 2\left(M_{yaw\ bearing} \cdot 6.689 + 953\right) \tag{2.15}$$

The maximum applied moment is referred to as the maximum moment between the one acting in the y direction M_y, so the one perpendicular to the yaw axis and parallel to the flow direction, and the one acting in the z direction M_z, so the one generated around the yaw axis during the rotation of the nacelle. Moreover, the mechanical brake is a safety component used for slowing down and, in extreme conditions, stopping the shaft when the current speeds are higher than the rated ones, or an emergency occurs, such as a fault in the electrical system which requires the turbine to be shut down [44]. The mechanical brake system cost is defined as:

$$C_{break} = 1.9894 \cdot P_N \tag{2.16}$$

where P_N is the rated power of the turbine. In addition, the low-speed shaft is a component that transmits the rotational motion from the rotor to the generator through the gearbox. Its cost is defined by:

$$C_{shaft} = 500\frac{D}{2} \tag{2.17}$$

However, since the rotational speed of the rotor is commonly really low, a gearbox is needed in order to transfer the load generated from a low-speed shaft to a high-speed shaft. Two different gearboxes have been considered: a single-stage gearbox and a three-stage gearbox, whose cost is defined by:

$$\begin{aligned} \text{Cost}_{gb\,1stage} &= 529.74, T_{lss}^{0.774} \\ \text{Cost}_{gb\,3stages} &= 700.94, T_{lss}^{0.759} \end{aligned} \tag{2.18}$$

The generator is the element that converts the mechanical energy into electrical energy that is exported after [32]. The direct drive Generator cost is estimated by:

$$C_{generator} = 219.33 \left(\frac{2\,\pi\,T_{shaft}}{3}\right) \tag{2.19}$$

where the T_{shaft} is determined by :

$$T_{shaft} = \frac{P_N}{\omega_{shaft}} \tag{2.20}$$

being the ω_{shaft} the angular velocity of the shaft, in radians per second. Another important element in the directed involved cost of tidal energy is the nacelle cover. This element hosts all the generating components, gearbox, shaft, generator, and brake system. Under the assumption that the nacelle cover is an empty cylinder composed of a material with a given yield strength σ_Y, it is then possible to estimate its costs by considering the mass-dependent equation:

$$Cost_{Nacelle} = \text{Mass}_{nacelle} \cdot P_{mat} \tag{2.21}$$

where P_{mat} is the material price which the nacelle is made of, and the $M_{nacelle}$ is the cylindrical-shaped mass of the nacelle, in which the external and internal nacelle diameters are denoted by D_0 and D_i, respectively:

$$\text{Mass}_{Nacelle} = \rho_{nacelle} \left[\left(D_0^2 - D_i^2 \right) \cdot 0.25 \cdot \pi \cdot L + 0.66 \cdot \pi \left(\left(\frac{D_0}{2} \right)^3 - \left(\frac{D_i}{2} \right)^3 \right) \right] \tag{2.22}$$

Furthermore, power converters are electrical components that allow to conversion of certain electrical energy characteristics to other ones, so to a different frequency and tension, allowing a variable speed operation of the turbine. Since the output of an electrical generator is AC, the power converter used can be an AC/DC converter or an AC/AC converter, depending on the specific design of the system. The cost of the power converter components can be estimated by:

$$\text{Cost}_{powerconv.} = 79 \cdot P_N \tag{2.23}$$

The cost of cables is considered. Submarine power cables are the element that transmits the electrical power from the turbines to the shore. In the tidal energy sector the use of high voltage direct current cabling (HVDC) is limited because they are typically small-sized farms and the distance from the shore is relatively small, and it's not worth installing it for economic reasons, then HVAC systems are employed. The cross-sectional area of cables (CSA) is estimated as:

$$CSA = 28.347 \cdot e^{0.0044 \cdot I} \tag{2.24}$$

where the I is the nominal current passing through the cable. The corrective coefficient nCSA is determined by:

$$n_{CSA} = 0.6553 + 0.0035 \cdot CSA \tag{2.25}$$

It is also required to compute the voltage corrective coefficients that are defined by the expressions:

$$\begin{aligned} n_V &= 0.9819 + 0.0078V\,[kV]\,,\; for\,V >\; 10kV \\ n_V &= -0.0021V^2 + 0.00607V + 0.6076\,[kV]\,,\; for\,V \leq\; 10kV \end{aligned} \tag{2.26}$$

Therefore, the cost of cables is determined by:

$$\text{Cost}_{cables} = 200 \cdot n_{CSA} \cdot n_V \cdot L \tag{2.27}$$

where L is the cable length, and 200 is a reference cost taken from literature [31]. Besides, the cost of the foundation is considered by multiplying the structure mass of the foundation or monopile element by the price per mass of the material employed, usually steel:

$$\text{Cost}_{foundation} = P_{steel} \cdot \text{Mass}_{foundation} \tag{2.28}$$

In case of the floating platforms that support tidal turbines, 4 mooring lines have been considered, which cost is defined as follows:

$$\text{Cost}_{chain} = P_{steel}\left(0.0219 \cdot 10^6 \cdot D_c^2 \cdot 4 \cdot L_m\right) \tag{2.29}$$

in which D_c is the diameter of the chain, and L_m is the length of the line.

Regarding the main financial indicators, the cost of energy, mainly expressed by Cost of Energy (COE) and Levelised cost of Electricity (LCoE), is computed as a function of the Capital and Operational Expenditures (CapEx and OpEx, respectively). CapEx is determined as the sum of all the initial costs of the technology components, whereas OpEx is the sum of the insurance and maintenance of the energy system. It is possible to estimate the COE by considering the AEP as well:

$$COE = \frac{CapEx + \sum_{i=1}^{n} OpEx_i}{\sum_{i=1}^{n} AEP_i}\,, \tag{2.30}$$

Similarly, the LCoE [45] has been previously noted and can be computed as in (2.31) ,

$$LCOE = \frac{CapEx + \sum_{t=1}^{n} \frac{OpEx_t}{(1+r)^t} + \frac{D_c}{1+r)^n}}{\sum_{t=1}^{n} \frac{AEP_t}{(1+r)^t}}\,, \tag{2.31}$$

where D_c is the decommissioning cost, which is discounted at the end of the lifetime of the tidal plant after n=25 years, while r=5% is the discount rate [46] and t is the time in years. On the other hand, the Net Present Value (NPV) is a fundamental financial metric used to evaluate the profitability of an investment. It is defined as the sum of discounted net cash flows over the project lifetime:

$$NPV = -CapEx + \sum_{t=1}^{n} \frac{R_t - OpEx_t}{(1+r)^t} + \frac{D_c}{1+r)^n}\,, \tag{2.32}$$

where R are the revenues from selling the electricity to the grid, expressed as,

$$R = AEP * p_{el}\,, \tag{2.33}$$

in which p_{el} is the selling price of electricity, imposed equal to 207 €/MWh as it has been defined for tidal energy in the UK [47]. Another financial metric to consider is the Return on Investment (ROI), defined as the ratio between the NPV and the total cost [**?**], which, in mathematical terms, becomes:

$$ROI[\%] = \frac{NPV}{C_{tot}}\,, \tag{2.34}$$

where C_{tot} is estimated by equation 2.35:

$$C_{tot} = AEP * CapEx + \sum_{t=1}^{n} \frac{R_t - OpEx_t}{(1+r)^t} + \frac{D_c}{1+r)^n}\,, \tag{2.35}$$

The final cost metric is the Payback Period (PP), which represents the duration required to recoup the Capital Expenditure (CapEx); this means that once the PP has elapsed, profits begin to accumulate. According to its definition, the NPV formula can be equated to zero, allowing the PP to be represented through (2.36):

$$PP = \frac{ln\left(\frac{R-OpEx}{(1+r)^t}\right)}{ln(1+r)}\,, \tag{2.36}$$

Closely related is the Internal Rate of Return (IRR), which is the discount rate that makes the NPV equal to zero:

$$0 = \sum_{t=0}^{T} \frac{R_t - C_t}{(1 + IRR)^t} \tag{2.37}$$

This rate provides an estimate of the expected return of the project and is widely used to compare alternative investments or technologies.

Finally, to estimate the installed capacity (IC) for N TECs it is employed the following expression is employed:

$$IC_{N_{TEC}} = \frac{AEP_{unit} \cdot N}{8760 \cdot CF} \tag{2.38}$$

where the CF corresponds to the Capacity factor, and N is the quantity of conversion units in the TEC array.

To end this theoretical analysis of the cost functions here are exposed the calculation methods for the electrolysis section of the plant are exposed, summarizing the cost of the three main components: PEM electrolyzer, compressor, and storage tanks. The total electrolyser CapEx can be approximated as:

$$\text{CAPEX}_{\text{electrolyser}} = C_{\text{unit}} \times P_{\text{el}} \tag{2.39}$$

where C_{unit} is the cost per kilowatt and P_{el} is the rated electrical power of the system.

The compressor CapEx is generally proportional to flow rate and discharge pressure:

$$\text{CAPEX}_{\text{comp}} = C_{\text{comp,unit}} \times P_{\text{comp}} \tag{2.40}$$

where $C_{\text{comp,unit}}$ is the cost per kW of compression power and P_{comp} the installed compressor capacity.

The storage investment can be represented by:

$$\text{CapEx}_{\text{storage}} = C_{\text{stor,unit}} \times M_{H_2,\text{stored}} \tag{2.41}$$

where $C_{\text{stor,unit}}$ (€/kg H_2) is the specific cost and $M_{H_2,\text{stored}}$ (kg) is the required storage capacity.

In the following table are summarized the typical cost ranges for the components involved in hydrogen production:

Table 2.2: Typical CapEx ranges for major hydrogen system components.

Subsystem	Typical CapEx Range	Units	Key References
PEM Electrolyser	380 – 1,070;	€/kW	[20, 21]
Compressor	250 – 800	€/kW	[22–24]
Type I Cylinder	77.5	€/kg H_2	[25]
Type II Cylinder	80	€/kg H_2	[25]
Type III Cylinder	350	€/kg H_2	[25]
Type IV Cylinder	633	€/kg H_2	[25]

The Total CapEx of the electrolysis system is:

$$\mathrm{CapEx}_{\mathrm{H_2}} = \mathrm{CapEx}_{\mathrm{electrolyser}} + \mathrm{CapEx}_{\mathrm{comp}} + \mathrm{CapEx}_{\mathrm{storage}} \tag{2.42}$$

The OpEx is defined as 1-3 % of the CapEx [22].

The last parameter to be calculated is the LCoh (Levelized Cost of Hydrogen), which has to take into account all the costs for the production of electricity through the tidal plant, in addition to the specific costs of the Hydrogen production process:

$$LCOH = \frac{CapEx + CapEx_{H_2} + \sum_{t=1}^{n} \frac{OpEx + OpEx_{H_2}}{(1+r)^t} + \frac{D_c + D_{c,H_2}}{1+r)^n}}{\sum_{t=1}^{n} \frac{H_2}{(1+r)^t}}, \tag{2.43}$$

where D_c is the decommissioning cost of the tidal plant, D_{c,H_2} is the decommissioning cost of the Electrolyser, both discounted at the end of the lifetime of the plant after n=25 years, while r=5% is the discount rate [44], and t is the time in years.

Chapter 3

Case Study: Fall of Warness

The present study has selected Fall of Warness as a site to test the computational tool presented in the previous chapter; it is a channel located in the Orkney Islands archipelago. This choice is motivated by several reasons, both environmental, the site is characterized by strong tidal currents and adequate seabed conditions, and technical, since it already hosts an operational test infrastructure managed by European Marine Energy Centre (EMEC), which enables the comparison of modeled results with real-world data.

Fall of Warness lies in a narrow channel between the Westray Firth and the Stronsay Firth, through which the tidal flows from the Atlantic Ocean into the North Sea are constricted and thus accelerated. [27] EMEC indeed reports peak velocities of almost 4 m/s at spring tides, which made this site the perfect selection for harvesting energy via tidal streams. The available area is approximately 2 km wide and 4 km long, with depths that vary from 12 m to 50 m.The seabed in this site is described in the scoping report as “scoured and tide-swept bedrock and boulders with mobile sand waves”. [48] These morphological conditions are advantageous for tidal turbine foundations or anchoring solutions (monopile, gravity base, floating) because of the relatively firm substrate and well-defined flow channel. Given the presence of an on-site hydrogen electrolyser (operational at the EMEC substation) and the infrastructure to export energy to both grid and hydrogen production, Fall of Warness offers an invaluable real-world setting for analysing the full chain: tidal energy $\rightarrow$ electricity $\rightarrow$ electrolyser $\rightarrow$ hydrogen. This allows the methodology developed in this thesis to be validated not only theoretically but also using a site where deployment and operation are already underway.

In the following sections, site-specific results will be presented for the three tidal substructure technologies (floating, monopile, gravity-based), compute the associated CapEx/OpEx values, derive LCoE for each configuration, then extend the analysis to hydrogen production and derive LCoH, comparing the outcomes with the existing test site configuration.

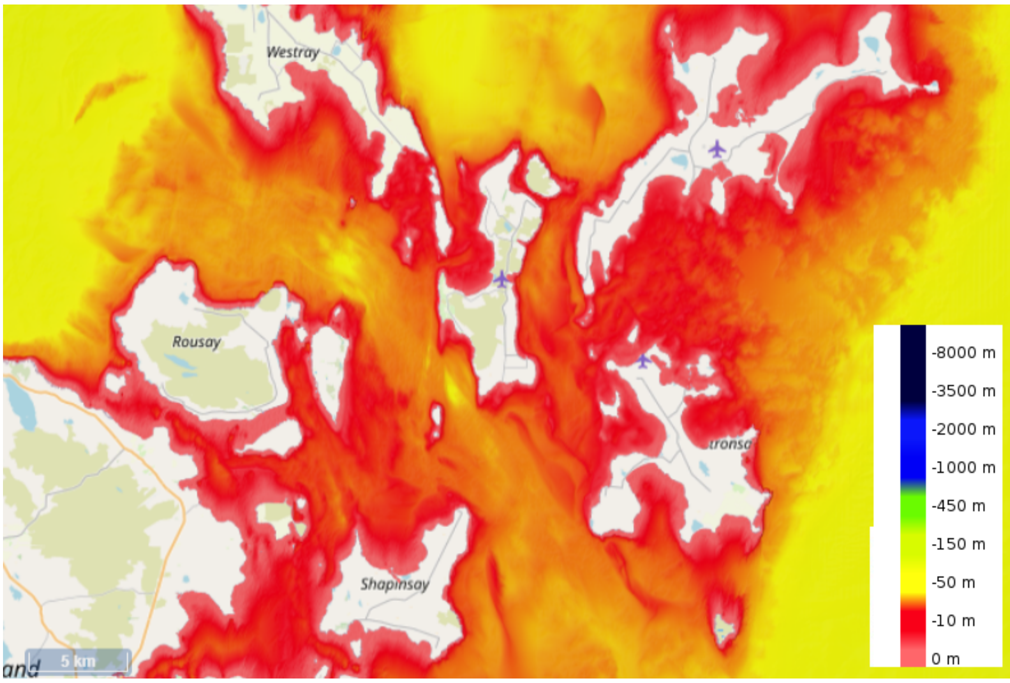


Figure 3.1: Bathymetry of the Fall of Warness area. [8]

3.1 Resource assessment

The geographical point selected to make the resource assessment is located at latitude 59°14’ and longitude 2°81’, and corresponds to the actual position of the Orbital platform where the O2 tidal stream turbine is being tested with the support of the EMEC tidal test site. Here, the water depth provided by the GEBCO dataset is 45, which is consistent with the indications provided by EMEC.

The tidal current velocity data have been retrieved from the Copernicus Marine Service dataset, which provides time series of tidal currents at different spatial coordinates. These data are organized on a regular geographical grid, with a spatial resolution of approximately 1/12°, corresponding to about 8–9 km. Although this resolution is sufficient for large-scale hydrodynamic analyses, it is not fine enough to precisely represent small-scale features or narrow straits such as the Fall of Warness channel.

As a consequence, the exact geographical coordinates of the selected site (Fall of Warness test area) do not coincide with any node of the Copernicus grid. To ensure consistency with the available dataset, the analysis was therefore carried out using the grid point closest to the target location, resulting in a longitude of 2°88' and a latitude of 59°14', where current and depth are, anyway, very similar to the ones expected from the initially chosen location.

Once the site location has been acquired, the distance from shore is calculated through the deputed function `distance_from_shoreline()`, which exploits the coastline data derived from `Natural_Earth_10m` coastline dataset, which provides a representation of the coastal boundaries with a spatial resolution of approximately 1 km. [49] Unfortunately, for the analysed case, the measured distance from the shore was about 4 km instead of the 1.4 km that divides the Orbital platform from the Emec onshore control unit. Since this difference is not negligible and would have increased the expected costs too much, it has been decided to set the distance from shore at 1.4 km in order to obtain more realistic results.

The processing of the tidal current velocity data obtained from the Copernicus Marine Service dataset was carried out using the dedicated `tidalCurr_analysis` function. This function was specifically developed to extract, analyze, and post-process the hydrodynamic time series at the selected location, providing the fundamental inputs for the subsequent energy yield assessment. The Copernicus dataset includes hourly values of the two horizontal current components (u and v) for the entire year 2024, resulting in a total of 8760 temporal records per variable.

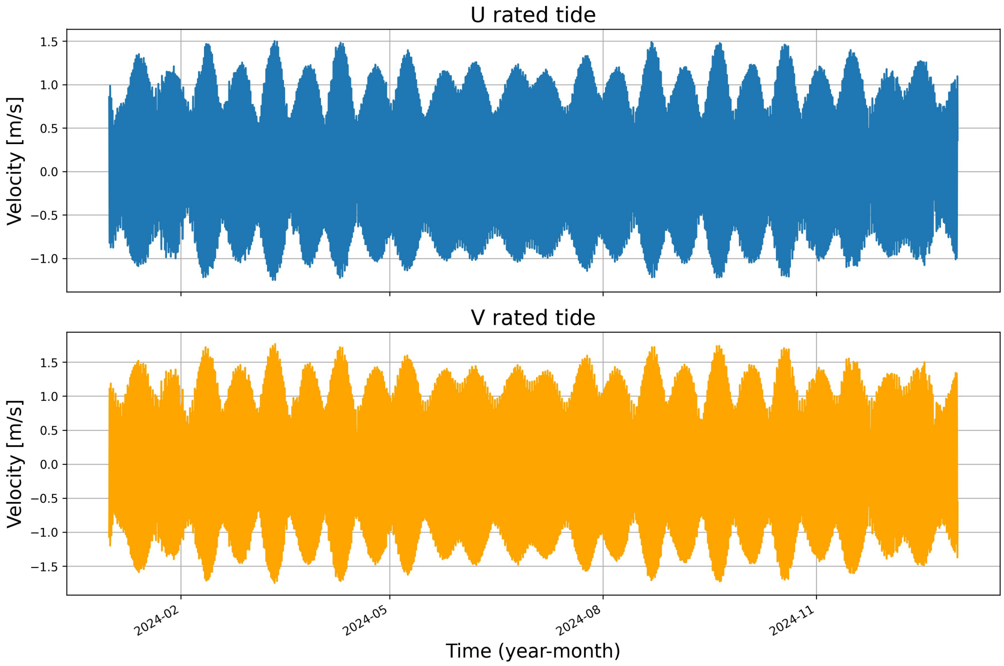


Figure 3.2: Horizontal and vertical components of tidal velocity.

These components represent the east–west and north–south flow velocities, respectively, from which the total current magnitude is derived as:

$$U(t) = \sqrt{u(t)^2 + v(t)^2} \tag{3.1}$$

Since the current data offered by the dataset are measured at a depth of 0.495 m, a depth correction function was implemented in the computational model to estimate the current velocity at the desired rotor hub depth. To do that, the data measured by the ATIR platform where exploited. The values have been measured at a depth of 13.9 m and report a value of the mean current velocity of 1.5 m/s. To estimate the current speed at the desired depth, a power-law vertical velocity profile is applied, expressed as:

$$U(z) = U_{\text{mean}} \left(\frac{h - z}{\beta h} \right)^{1/\alpha} \tag{3.2}$$

where U_{mean} is the depth-averaged current velocity, h is the local water depth, and z is the vertical coordinate of interest (measured upward from the seabed). The parameters $\beta = 0.4$ (bed roughness coefficient) and $\alpha = 7$ (power-law exponent) are known from literature. Since the measure provides U(z=13.9), it is possible to find U_{mean} for the entire time series by inverting the equation (3.2) and using it to calculate the current at the different depths imposed by the technology constraints. [32]

For each turbine technology configuration (floating, monopile, gravity-based), the desired rotor depth was defined as a function of the total water depth H and the necessary clearances from both the sea surface and the seabed, according to installation and operational constraints. The hub depth is therefore computed as:

$$Depth_{rotor,min} = Depth_{clearance} + D/2 \tag{3.3}$$

To accurately determine the rotor hub depth, it is necessary to quantify the minimum clearance distances from both the sea surface and the seabed. Based on the ATIR project specifications, a minimum clearance from the sea surface of 4.4 m has been established. Although this value was originally defined for a floating platform, the same clearance is adopted in this study for the monopile configuration. [32]

For the gravity-based structure (GBS), a different clearance constraint is applied, corresponding to 8 m. With respect to the minimum clearance from the seabed, this parameter is particularly relevant for the GBS configuration, where the turbine foundation directly interacts with the seabed. As a result of the literature review, it has been found that a minimum clearance of 6 m from the seabed is necessary. [32]

Table 3.1: Clearance constraints and installation depth for tidal energy converters (TECs).

Parameter	**Gravity-based (GBS)**	**Floating**	**Monopile**
Minimum clearance from sea surface [m]	8.0	4.4	4.4
Minimum clearance from seabed [m]	6.0	6.0	6.0
Installation depth [m]	30	15	15

As a result of this analysis, the rotor depths for the different TECs have been obtained, making it possible to have a complete overview of the current resource, since its value varies with depth.

The resulting tidal current time series exhibits the expected semi-diurnal pattern, with two high and two low tides per day, typical of the Orkney region. To better visualize the short-term tidal variability, Figure 3.3 shows the current velocity evolution over a representative 24-hour period. The alternating flood and ebb phases are clearly visible, with maximum velocities exceeding 2 m/s and minimal approaching 0.5 m/s, confirming the strong and regular oscillatory nature of the tidal stream.

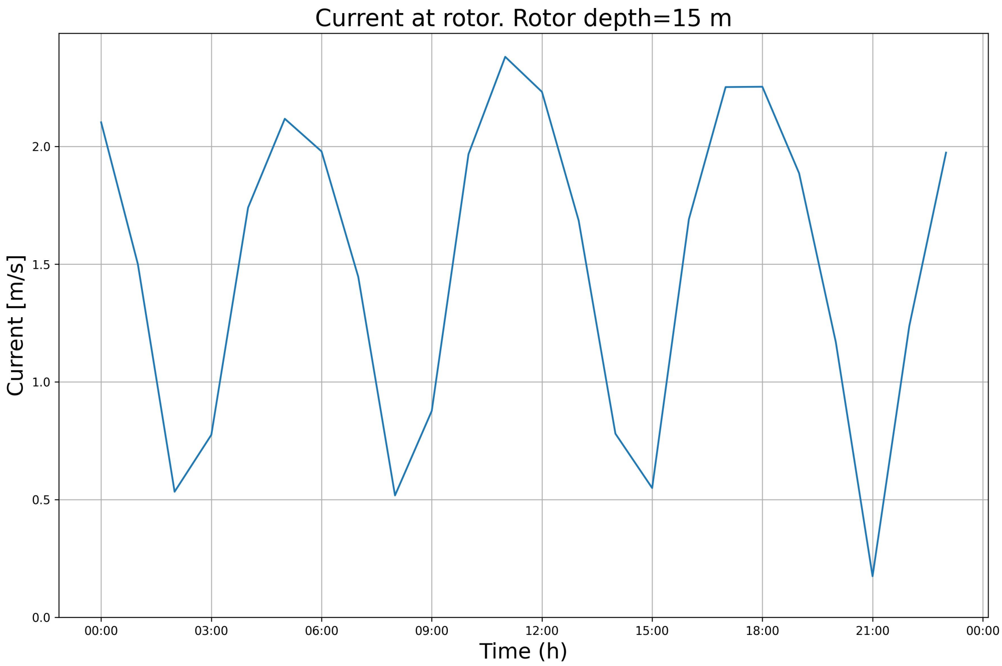


Figure 3.3: Example of tidal current velocity variation at rotor depth (15 m) over one day.

Extending the observation to a longer period, the complete annual time series (Figure 3.4) highlights the seasonal modulation of tidal currents, with slightly higher amplitudes during winter and spring months. The maximum registered value for floating and monopile configurations is 3.31 m/s with an average velocity of 1.49 m/s, while for GBS, whose installation point is deeper, the peak current is 3.12 m/s and the mean value is 1.41 m/s. These data confirm the high energetic potential of the site, underlining at the same time the variations of the currents in relation to the depth at which the study is performed.

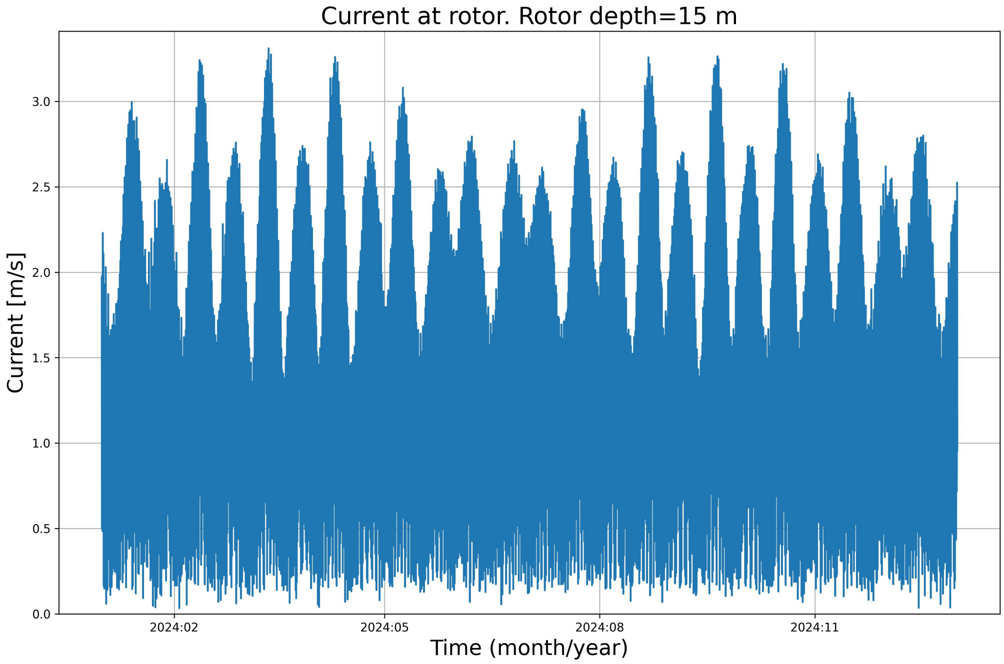


Figure 3.4: Time series of tidal current velocity at rotor depth (15 m) for the year 2024.

To describe the statistical variability of the flow, the velocity distributions were fitted using a normal probability density function. This fitting allows identifying the mean velocity and dispersion around it, both of which are essential parameters for predicting the energy yield of the turbine. Figure 3.5 reports the results for two representative technologies — the Floating Orbital O2 and the GBS Tocardo T2 configurations.

The two histograms show similar behavior, with most of the data concentrated between 1.0 and 2.0 m/s. The Floating Orbital O2, operating closer to the surface, experiences slightly higher peak velocities due to reduced frictional effects. In the case of the GBS configuration, the PDF appears narrower and more peaked, suggesting that the current velocities are more concentrated around a characteristic value, with lower variability compared to the floating system.

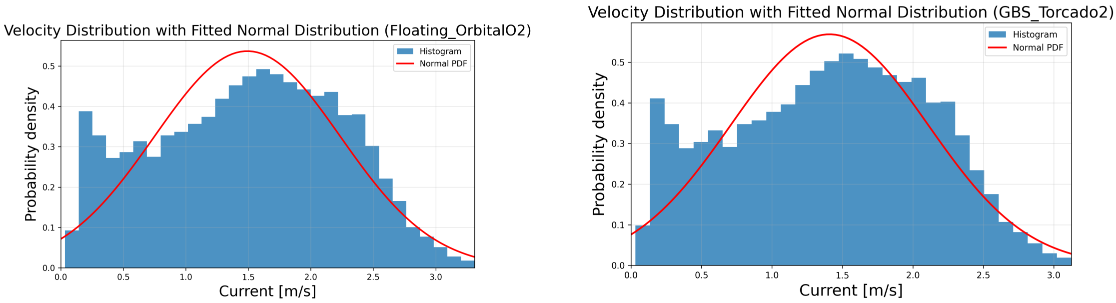


Figure 3.5: Velocity distributions with fitted normal probability curves for the two selected TEC configurations.

3.2 Power production analysis

After the resource characterization, the following step consists of analysing the power performance that results from the site's current velocities. This can be resumed by quantifying the energy conversion potential of the selected tidal energy converters by applying the physical relationships between current velocity, turbine geometry, and conversion efficiency.

The instantaneous mechanical power can be calculated through (2.1). The analysis is carried out by considering a rated power of 1 MW in all the cases. Further consideration needs to be made in order to ensure realistic performance estimation: each turbine has its own characteristics regarding the operational limits, which are summarized in table 3.2. The cut-in velocity represents the minimum current required to start power generation, the rated velocity corresponds to the flow at which nominal power is achieved, and the cut-out velocity indicates the limit beyond which the turbine shuts down for safety reasons.

Table 3.2: Operational velocity parameters for the different tidal energy converter (TEC) configurations.

TEC	**Cut-in velocity [m/s]**	**Rated velocity [m/s]**	**Cut-out velocity [m/s]**
GBS Tocardo T2	1.0	2.00	4.0
Floating Orbital O2	1.0	2.65	4.0
Monopile SeaGen S2	1.0	2.5	4.0

With the implementation of this turbine's characterization, an accurate power curve can be obtained for the three technologies.

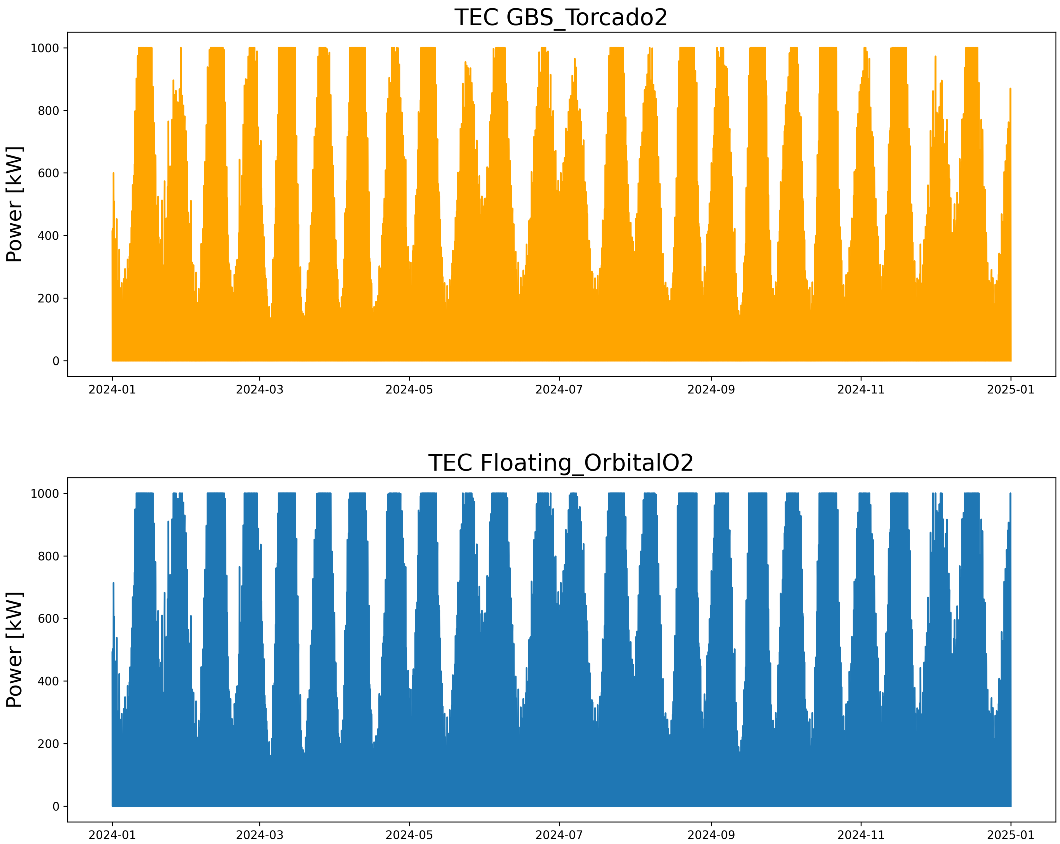


Figure 3.6: Annual power output comparison for the different TEC configurations at the Fall of Warness site.

In figure 3.6, the monopile configuration is not reported since the power series is the same as the floating one. The comparison is useful to underline the lower power values reached by the GBS configuration, a result which is coherent with the fact that lower current values are registered at GBS rotor depth. It's evident that the role of the cut-out velocity, which many times limits the generated power at the rated one. Thanks to these results, it is possible to calculate the mean power production and the Capacity Factor, parameters that reflect the overall behavior of the power series in the different scenarios. For floating and monopile $P_{\text{mean}} = 337$ kW and CF = 33.78%, while for GBS $P_{\text{mean}} = 293$ kW and CF = 29.34%. To conclude the power production, the Annual Energy Production has been calculated,d resulting in 2.97 MWh for floating and monopile and 2.58 MWh for GBS.

3.3 Economic Analysis

The economic assessment aims to compare the three tidal energy converter (TEC) configurations — GBS (Tocardo T2), Floating (Orbital O2), and Monopile (SeagenS2) — in terms of their investment, operational, and energy generation costs. The analysis has been carried out using the `TE_2` script, which computes the costs based on the selected technology. It is important to note that for these analyse,s each selected technology is associated with a real device. This strongly affects the results since for monopile and floating configurations, there are 2 turbines per structure, resulting in a significant cost reduction, while for GBS, there is only one. The indicators considered include the Capital Expenditure (CapEx), Operational Expenditure (OpEx), and the Levelized Cost of Energy (LCoE), evaluated as a function of the installed capacity.

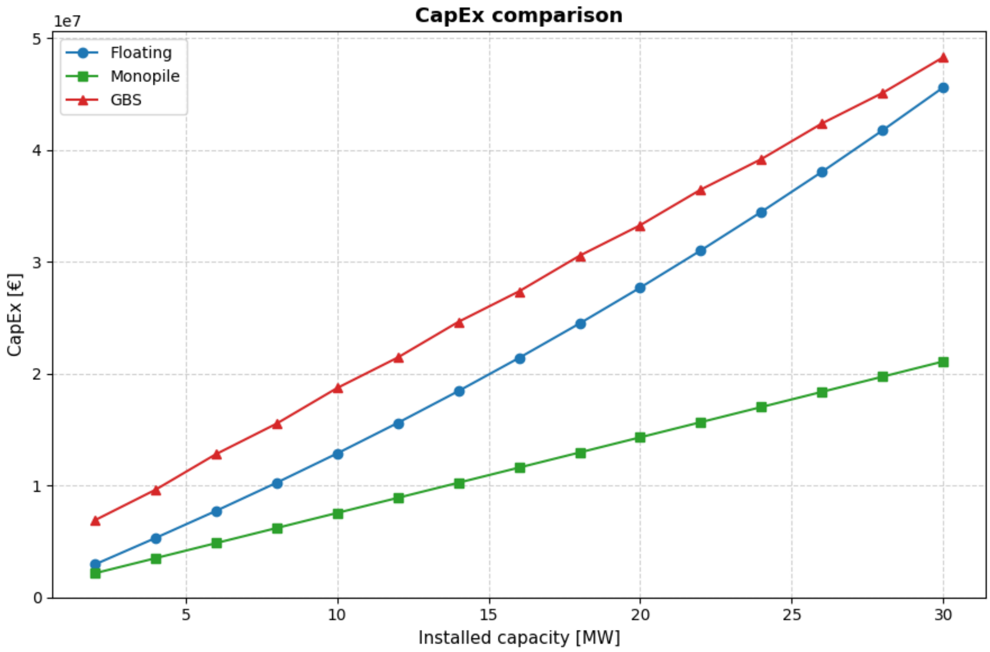


Figure 3.7: Total CapEx as a function of installed capacity for the different TEC configurations.

Figure 3.7 presents the total Capital Expenditure (CapEx) for the three TEC concepts. All curves display an approximately linear trend, confirming that capital costs scale proportionally with the number of installed devices.

The GBS configuration exhibits the highest CapEx, primarily due to the cost-intensive concrete foundation and complex installation procedures typical of seabed-mounted systems. The floating configuration, while avoiding heavy seabed infrastructure, still involves significant investment related to platform fabrication and mooring systems. In contrast, the monopile TEC results in the lowest CapEx, due to its simpler design and reduced offshore assembly requirements.

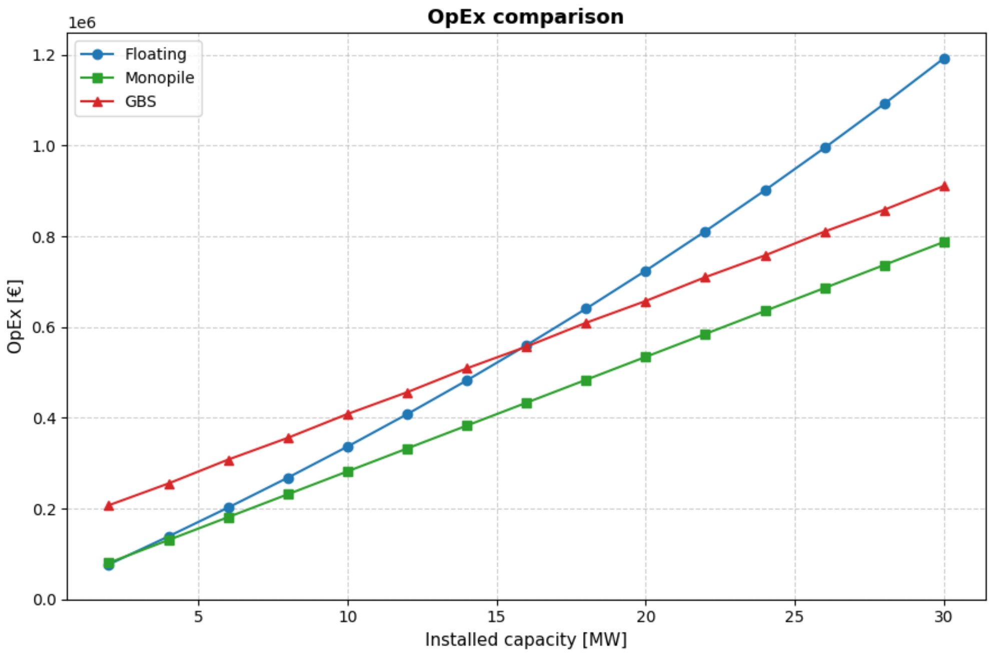


Figure 3.8: Annual OpEx as a function of installed capacity for each TEC configuration.

The trend of Operational Expenditure (OpEx), shown in Figure 3.8, follows a similar linear behavior with increasing installed capacity. OpEx includes maintenance, inspection, and operational costs over the system's lifetime. Among the analyzed systems, the floating configuration presents the highest OpEx, reflecting the more frequent maintenance cycles of moored structures, although operations are easier to perform offshore. The GBS system shows slightly lower OpEx values but suffers from more challenging accessibility and weather-dependent interventions. Finally, the monopile system confirms its cost-effectiveness, with the lowest annual OpEx due to the simplicity of its structure and standardized maintenance procedures.

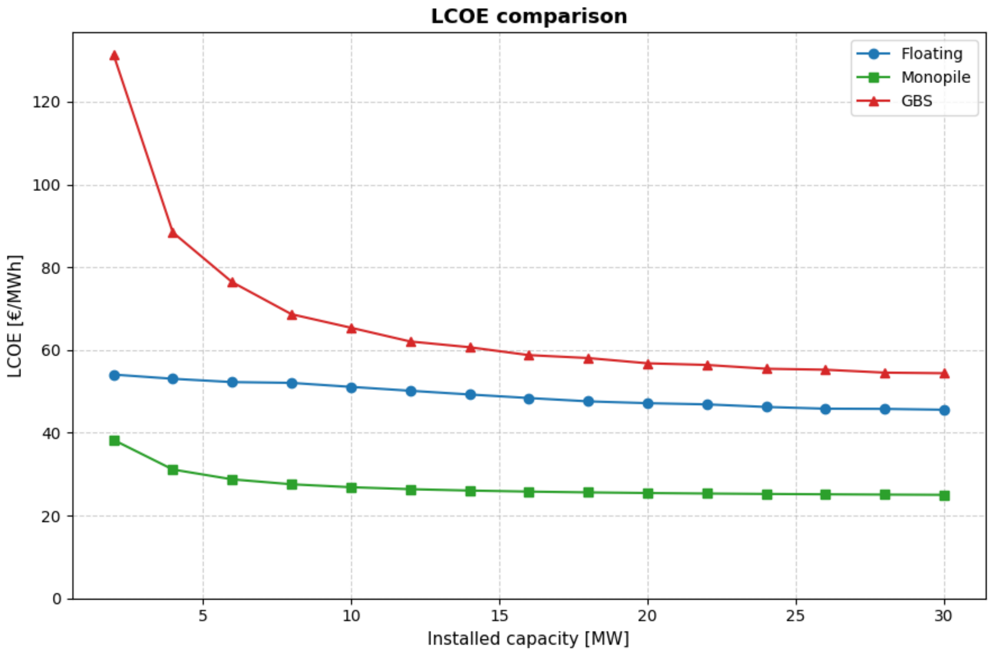


Figure 3.9: LCoE trend with installed capacity for the different TEC configurations.

The Levelized Cost of Energy (LCoE), illustrated in Figure 3.9, combines the effects of CapEx, OpEx, and the energy yield obtained from the hydrodynamic analysis. A decreasing trend is observed for all technologies as installed capacity increases, primarily due to economies of scale and the dilution of fixed investment costs across a higher energy output. The decrement is much more evident for the GBS technology, while for the other two TECs is very limited.

At small array scales (2–5 MW), LCoE values are relatively high, exceeding 100 €/MWh for the GBS system. However, as the farm size increases, LCoE gradually stabilizes toward more competitive values:

- Monopile TEC: lowest LCoE, around 25 €/MWh at 30 MW
- Floating system: intermediate range, approximately 45 €/MWh,
- GBS system: highest values, stabilizing near 55 €/MWh.

These results indicate that the monopile configuration provides the best economic performance, combining moderate capital requirements with reliable energy production. The floating concept, although more expensive, benefits from scalability and easier maintenance, making it attractive for deeper waters or high-current offshore locations. A clear relationship emerges between the production indicators (AEP and CF) and the economic performance of the three tidal technologies. As expected, higher capacity factors lead to larger annual energy outputs, which in turn reduce the levelized cost of energy by distributing the capital and operational expenditures over a greater amount of produced energy. This trend is visible in the floating and monopile configurations, which exhibits the highest CF and AEP, resulting in competitive LCoE values. The GBS structure, with the lowest CF and AEP, correspondingly exhibits the highest LCoE. These comparisons highlight the importance of jointly evaluating production and cost metrics, as only their combined interpretation provides a comprehensive understanding of technology competitiveness.

3.4 Hydrogen integration

The development of the hydrogen production module within the techno-economic framework was carried out in continuous alignment with real-world demonstration projects. Given the limited number of existing tidal-to-hydrogen implementations, the ITEG project, conducted at the European Marine Energy Centre (EMEC), served as the primary reference case. In the original version of the project, the Orbital O2 tidal turbine was coupled with a 500 kW PEM electrolyser, providing a practical benchmark for evaluating the integration between tidal power generation and on-site hydrogen production. The correct sizing of the electrolyser is a key aspect in determining both the technical performance and the economic competitiveness of the integrated tidal-to-hydrogen system. However, the available literature does not provide a unanimous consensus on the optimal design criteria for electrolyser capacity when coupled to variable renewable sources. Most studies indicate that the sizing should be based on a trade-off between maximizing hydrogen production and maintaining a sufficiently high capacity factor of the electrolyser to ensure cost effectiveness. An oversized electrolyser would be able to convert a larger share of the instantaneous tidal power, but it would also operate for a limited number of hours at nominal load, leading to a lower capacity factor and a higher LCoh. On the contrary, an undersized unit would achieve better utilization rates but would not fully exploit the available energy, resulting in a reduced annual hydrogen production. As a consequence, for this study, it has been decided to replicate the ITEG project setup, adopting a 500 kW PEM electrolyser coupled with the Orbital O2 turbine, whose performances have already been assessed in the previous sections of this work. Adopting a similar electrolyser capacity ensures that the present analysis remains consistent with real-world experimental data and allows for a meaningful comparison between the modeled system and existing demonstration projects. Moreover, the selected size represents a reasonable balance between hydrogen output and electrolyser utilization, offering a realistic benchmark for cost and performance evaluation.

To complete the electrolysis module, there is a compressor that compresses hydrogen from 30 bar to 300 bar and a type II storage tank. Their dimensioning is implemented in the script as a function of the hydrogen to be processed. In this particular case, the compressor calculated rated power is 32 kW, and the nominal storage capacity is 124 kg/day.

In the computational tool that has been developed for this work, the hydrogen production analysis has been carried out by simulating two different operational scenarios:

- Full Power-to-Hydrogen Conversion: In this configuration, the entire electrical output of the tidal farm is directed to the PEM electrolyser. This represents an idealized situation in which the tidal power plant operates exclusively for hydrogen generation, with no local or grid-connected electrical demand.

- Hybrid Energy Supply with Local Demand: this configuration is more realistic; it considers a hybrid operation in which the tidal farm primarily supplies a local or grid-connected electrical demand defined by the user, while the excess energy is diverted to hydrogen production. This approach mimics the operational conditions of coastal communities or port infrastructures where the tidal system contributes to local energy autonomy while exploiting surplus power for hydrogen generation.

Full Power-to-Hydrogen Conversion

For this configuration, the resource assessment previously, can be exploited, adapting it to the fact that now a structure with two turbines is being considered. As a result, $P_{mean} = 676$ kW is the average value of instantaneous power produced by the tidal plant.

Once the PEM electrolyser has been dimensioned, the mass flow rate of produced hydrogen needs to be calculated. According to IRENA [50], it can be calculated as:

$$\dot{m}_{\mathrm{H_2}} = \frac{P_{\mathrm{el}} \times \eta_{\mathrm{PEM}}}{\mathrm{LHV}_{\mathrm{H_2}}} \tag{3.4}$$

where:

- $\dot{m}_{\mathrm{H_2}}$ is the hydrogen production rate [kg/h],
- P_{el} is the electrical power supplied to the electrolyser [kW], calculated as the minimum power between the PEM nominal power and the produced tidal power fed to the electrolyser. Note that to the power fed to the electrolyser it has been subtracted the power necessary to run the compressor has been subtracted, whose estimation is 3 kWh/kg [51]
- η_{PEM} is the electrolyser efficiency (0.7) [50]
- $\mathrm{LHV}_{\mathrm{H_2}}$ is the lower heating value of hydrogen (33.33 kWh/kg H_2).

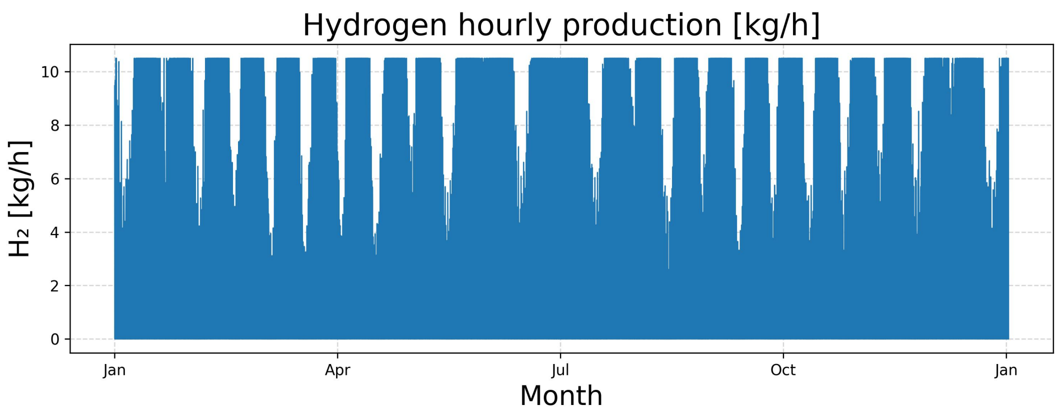


Figure 3.10: Hydrogen hourly production

The pattern represented in Figure 3.10 clearly follows the tidal resource variability. The production peaks, reaching around 10–11 kg/h, occur during the periods of maximum tidal current velocity, when the tidal turbine operates close to its rated power and the electrolyser runs at full capacity. Conversely, sharp drops to zero correspond to slack tide conditions, when the water flow reverses and turbine output momentarily ceases.

Despite these interruptions, the overall production remains highly regular and predictable, reflecting the inherent cyclicity of tidal energy.

The Annual hydrogen production (AHP) is attested at 45 tons/year with a CF for the PEM Electrolyser of 48.2 %, which is an acceptable value for this kind of technology.

The integration of hydrogen production in the model allows the creation of a new parameter, the Levelized Cost of Hydrogen, that, in the analysed case, attests its value at 7.28 €/kg. It represents a realistic outcome, although it remains higher than the current values for green hydrogen production; this is mainly due to the small scale of the application and the relatively high costs of tidal energy production.

Table 3.3: Summary of hydrogen system costs and resulting LCoh.

Component / Parameter	**Cost [€]**
PEM electrolyser	500,000.00
Compressor	9,375.01
Hydrogen storage	1,688.09
Total CapEx (hydrogen system)	**511,063.09**
OPEX (annual)	**10,221.26**
LCoh	**7.28 €/kg H_2**

Hybrid Energy Supply with Local Demand

In this scenario, the hybrid operating mode is analysed, with the subordination of the hydrogen production to a local electrical demand. The user-defined demand is sized to emulate the load of a small coastal energy community (hundreds of kilowatts of average power). To quantify how the hydrogen output reacts to different load levels, the demand was swept from 100 kW up to values that exhaust the average tidal power available to the electrolyser. Throughout the exercise, the PEM size is kept constant (500 kW), so CapEx and OpEx remain unchanged; consequently, the LCoH necessarily increases as the demand grows, because less energy reaches the electrolyser and annual H_2 production declines.

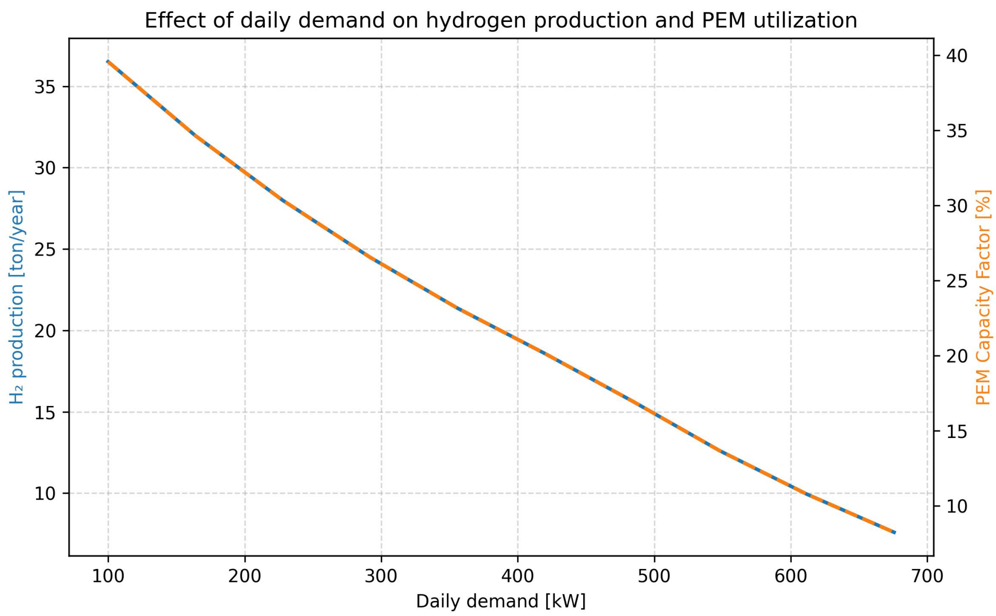


Figure 3.11: Hydrogen production and CF variation in response to electric demand increment

The blue curve in Figure 3.11 shows annual hydrogen production (ton/ear), while the orange dashed curve reports the PEM capacity factor (%). Both decrease monotonically as the local demand rises: more of the tidal power is diverted to the community load, leaving less power for electrolysis and reducing the fraction of time the PEM operates near its rating. At low demand (100–200 kW), the electrolyser maintains a higher CF and yields the maximum H_2 output; towards the upper end of the sweep, the PEM becomes increasingly under-utilised, and production approaches the minimum.

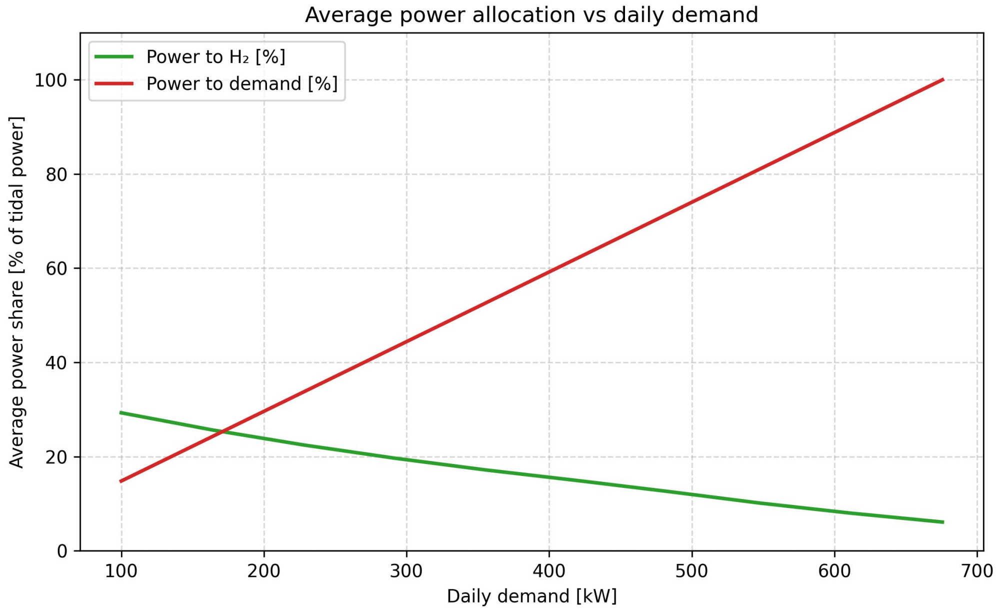


Figure 3.12: Percentage of power allocation for hydrogen production and for electrical demand

The plot in figure 3.12 expresses, on an average-power basis, how the available tidal power is partitioned between the community demand (red) and the electrolyser (green). As the demand increases, the share of demand rises linearly, while the share of H_2 production falls correspondingly. When the demand approaches the average tidal power, almost no power remains for electrolysis, explaining the sharp decline in H_2 output and PEM CF observed in Figure 1.

To confirm the previously done hypothesis, one more plot has been created to observe how the LCoH reacts to the demand variation.

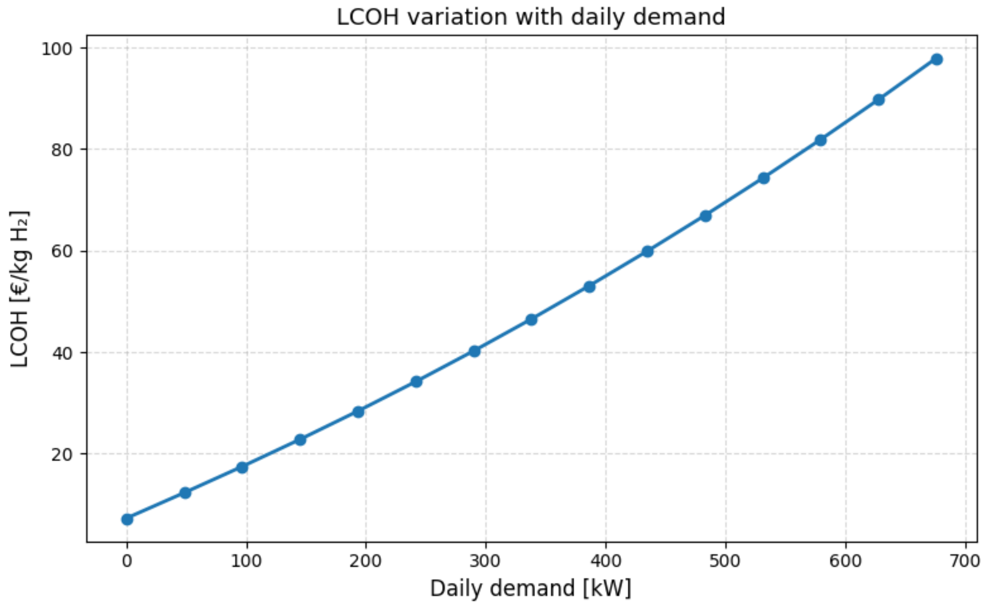


Figure 3.13: LCOH behaviour in response to electricity demand increment.

Because the electrolyser's nominal capacity is fixed, CapEx and OpEx do not change across the demand sweep. Figure 3.13 shows how the decreasing hydrogen production therefore leads, by definition, to a higher Levelised Cost of Hydrogen (LCoH) at higher demand levels. In other words, under a fixed-size PEM, diverting more power to the local load trades economic performance of H_2 for improved local electrification, a system-level choice that depends on project priorities.

Conclusions

The research presented in this thesis has investigated the techno-economic feasibility of tidal stream energy systems and their integration with green hydrogen production through electrolysis. The main objective was to create a computational framework capable of estimating energy yields, costs, and financial indicators for different technological configurations, thus identifying the most promising solutions for future marine renewable developments.

The study has been structured into four main chapters, each addressing a specific component of the analysis — from the theoretical foundations to the numerical modeling and final validation through a real case study.

To begin the analysis, an introduction to the theoretical background has been provided to define the scientific basis of the study. In this chapter, the physical principles of the tides have been described, along with the different energy conversion technologies and the electrolysis process for hydrogen production. Three substructure typologies — gravity-based, monopile, and floating — were identified as the focus of this research. The part on hydrogen production examined the main electrolysis technologies. The Proton Exchange Membrane (PEM) system was selected as the reference technology for tidal integration, given its compact design, fast dynamic response, and operational efficiency under variable load conditions. Then, a summary on the cost structure of electrolyser systems, compressors, and hydrogen storage was provided, based on a literature review. The final part of the chapter introduced possible tidal-hydrogen integration schemes.

The second chapter described the structure of the developed computational tool. Implemented entirely in Python, the model integrates hydrodynamic, energetic, and financial equations to estimate performance metrics such as Annual Energy Production (AEP), Capacity Factor (CF), Levelized Cost of Energy (LCoE), and Levelized Cost of Hydrogen (LCoH). A major achievement of this chapter is the development of a flexible, modular tool capable of simulating various site conditions, turbine layouts, and hydrogen demand profiles. This flexibility was later validated in the case study through a variable-demand scenario, confirming the replicability and robustness of the model under changing boundary conditions.

In the last chapter, the computational model was applied to the Fall of Warness test site in Scotland — one of the most advanced tidal energy research areas managed by the European Marine Energy Centre (EMEC). The site's strong tidal currents (up to 4 m/s) and existing hydrogen production infrastructure made it ideal for validating the proposed methodology. The hydrodynamic data obtained from the Copernicus Marine Service have been used to calculate the energy output for the three substructure types. The mean delivered power and Capacity Factor (CF) values were computed and compared across configurations as showed in Table 3.4. In this section, it has been underlined the fact that the differences in power production are only due to the depth at which the rotor is installed, which is higher for GBS due to structural limitations.

Table 3.4: Average Power Output and Capacity Factor for the three tidal technologies.

Technology	P_{mean} **[kW]**	**Capacity Factor [%]**
Floating	337	33.78
Monopile	337	33.78
Gravity-Based Structure (GBS)	293	29.34

After that, the economic analysis was carried out with the production of several financial indicators: CapEx, OpEx, and LCoE. The comparison of the three technologies under these criteria has pointed out the superiority of the monopile technology from an economic viewpoint. This type of TEC shows the lowest LCoE with values that range from 38 €/kW to 25 €/MWh, while for floating and GBS the ranges are 55-54 €/MWh and 131-54€/MWh respectively. As expected, the LCoE values decrease as the installed capacity increases in all three scenarios.

The integration of a PEM electrolyser was then analysed to evaluate the Levelised Cost of Hydrogen (LCoH), considering the combined influence of electricity cost, electrolyser efficiency, and component sizing. This was done by replicating the setup of the ITEG project in the EMEC test site with an Orbital O2 turbine coupled with a 500 kW electrolyser. In this configuration two different scenarios have been developed: in the first one the tidal turbine produced power was entirely used to feed the electrolyser obtaining an annual production of 45 tons/year with a CF of 48.2% and a consequent LCoH of 7.28 €/kg; in the second one a user defined energy demand has been introduced in the computation to evaluate the response of the tool to a variable-demand setup. In this second configuration, the variation in hydrogen production has been observed with a decreasing trend in response to an increase in the energy demand from the grid. Different plots have been produced to show the difference in power allocation with respect to the daily demand set by the user.

This work enabled the techno-economic modeling of tidal-to-hydrogen energy systems, leading to the main conclusions summarized in the following paragraph.

The results confirm tidal stream energy as a highly predictable and reliable renewable source, characterized by a strong potential for integration into future sustainable energy mixes. The presence of many different technologies makes this source very versatile and well-suited for different applications being installed and adapted to a wide range of sea conditions. The coupling of these tidal energy converters with Proton Exchange Membrane (PEM) electrolysers demonstrated the technical feasibility of converting a variable yet cyclical renewable source into storable hydrogen fuel, providing both energy flexibility and long-term storage capacity.

From an economic perspective, the results indicate that tidal energy is approaching commercial viability, both as a stand-alone plant and when coupled with hydrogen production. Among the three technologies considered, monopile systems emerge as the most economically advantageous, primarily due to their lower installation and maintenance costs. However, the cost gap with floating and gravity-based structures progressively narrows as the installed capacity increases, suggesting that economies of scale play a crucial role in reducing the overall cost of energy across all configurations. Moreover, the addition of the PEM electrolyser does not significantly raise the total capital expenditure of the system. As a result, the combined tidal-to-hydrogen model maintains competitive performance, which is reflected in the favourable LCOH values obtained. These results confirm that integrating hydrogen production into tidal stream systems can be economically viable, especially in remote or off-grid contexts where hydrogen provides strategic value as a storable and transportable energy carrier.

Beyond its technical and economic implications, the study also highlights the environmental and strategic relevance of tidal-to-hydrogen integration. The deployment of such systems can significantly contribute to the decarbonization of coastal and insular regions, where access to stable renewable resources is limited. By providing both clean electricity and a dispatchable, carbon-free fuel, tidal-to-hydrogen plants can play a crucial role in the energy transition, supporting grid stability and complementing intermittent sources such as wind and solar. Their predictability and independence from seasonal weather patterns make them particularly valuable for ensuring long-term energy security and resilience in isolated power systems.

From a methodological standpoint, the thesis offers a significant contribution through the development of a Python-based computational tool that combines detailed engineering modeling with a comprehensive economic assessment. The modular structure of the code, the integration of up-to-date cost functions, and the inclusion of hydrogen production modeling provide a powerful decision-support instrument. This tool enables the evaluation of multiple design configurations and facilitates comparison across technologies.

In summary, this work has demonstrated the strong technical and economic viability of tidal-to-hydrogen systems, highlighting the adaptability and competitiveness of the different turbine technologies, with monopile designs emerging as the most cost-effective option. The integration of PEM electrolysers proved effective in converting a cyclical renewable resource into storable hydrogen without significantly increasing capital costs. The high predictability of tidal energy and the robustness of the developed computational model further reinforce the credibility of the methodology and its usefulness as a decision-support tool.

Looking ahead, several options for future development can be identified. A natural continuation of this work would involve the implementation of real-time energy management strategies to dynamically control the power distribution between tidal generation, grid interaction, and hydrogen production. Furthermore, extending the computational model to include other renewable sources such as offshore wind or wave energy would enable the design of multi-hybrid marine systems, enhancing stability and resource complementarity.

Bibliography

[1] Fabio Di Felice. Lecture presentation. In *Tidal Energy*, 2024.

[2] George Aggidis Shaun Waters. Tidal range technologies and state of the art in review. *Elsevier*, 2016.

[3] I. Goda M. Rouway M. Nachtane, M. Tarfaoui. A review on the technologies, design considerations and numerical models of tidal current turbines. *Elsevier*, 2020.

[4] Sabella. http://www.sabella.fr. Accessed: 2025-09-21.

[5] Strangford lough mct seagen. https://tethys.pnnl.gov/project-sites/strangford-lough-mct-seagen. Accessed: 2025-09-21.

[6] Orbital marine power. https://orbitalmarine.com/. Accessed: 2025-09-21.

[7] Interreg North-West Europe. Iteg: Integrating tidal energy into the european grid, n.d. [Online].

[8] EMODnet (European Marine Observation and Data Network). Emodnet geoviewer. https://emodnet.ec.europa.eu/geoviewer/, 2025. Accessed: 2025-11-10.

[9] National ocean service (noa), tides and water leves. https://oceanservice.noaa.gov/education/tutorial_tides/welcome.html. Accessed: 2025-08-20.

[10] Wei-Haur Lam Long Chen. A review of survivability and remedial actions of tidal current turbines. *Elsevier*, 2015.

[11] Hankwon Lim S. Shiva Kumar. An overview of water electrolysis technologies for green hydrogen production. *Elsevier*, 2022.

[12] International Energy Agency. Global hydrogen review 2025. Technical report, IEA, 2025.

[13] J. Brauns and T. Turek. Alkaline water electrolysis powered by renewable energy: A review. *Chemie Ingenieur Technik*, 2020.

[14] A. López, C. García, J. Ortega, and M. Pérez. Dynamic operation of water electrolysers coupled with renewable energy sources: A review. *Renewable and Sustainable Energy Reviews*, 2023.

[15] A. Sayed-Ahmed, R. El-Emam, and F. Vivas. Performance and degradation of pem water electrolysers under dynamic operation. *International Journal of Hydrogen Energy*, 2024.

[16] Y. Wang, S. Chen, J. Li, and X. Zhao. Comparative analysis of pem and alkaline electrolysers for green hydrogen production under fluctuating renewable power. *Applied Energy*, 2025.

[17] T. Egeland-Eriksen, J. Fjelldal, and S. Seljeseth. Offshore hydrogen production from marine energy: Techno-economic assessment of tidal and wave powered electrolysis. *Renewable Energy*, 2023.

[18] U.S. Department of Energy (DOE). Offshore wind–pem hydrogen integration program: Technical baseline report. Technical report, U.S. Department of Energy (DOE), Washington, D.C., 2024.

[19] Bureau of Ocean Energy Management (BOEM). Offshore hydrogen production: Integration of electrolysis with marine renewable energy systems. Technical Report BOEM-2022-034, Bureau of Ocean Energy Management (BOEM), Washington, D.C., 2022.

[20] S. Krishnan, V. Koning, M. T. de Groot, A. de Groot, P. Granados Mendoza, M. Junginger, and G. J. Kramer. Present and future cost of alkaline and pem electrolyser stacks. *International Journal of Hydrogen Energy*, 83:32313–32330, 2023.

[21] A. Badgett, J. Brauch, A. Thatte, R. Rubin, C. Skangos, X. Wang, R. Ahluwalia, B. Pivovar, and M. Ruth. Updated manufactured cost analysis for proton exchange membrane water electrolyzers. Technical Report NREL/TP-6A20-87625, National Renewable Energy Laboratory (NREL), 2024.

[22] A. Christensen. Assessment of hydrogen production costs from electrolysis: U.s. and europe. Technical report, International Council on Clean Transportation (ICCT), 2020.

[23] Hydrogen electrolysis techno-economic analysis database. Technical report, Electric Power Research Institute (EPRI), 2022.

[24] Thunder Said Energy. Compressors: A simple overview, 2024.

[25] A. S. Mehr, A. D. Phillips, M. P. Brandon, M. T. Pryce, and J. G. Carton. Recent challenges and development of technical and technoeconomic aspects for hydrogen storage, insights at different scales: A state-of-the-art review. *International Journal of Hydrogen Energy*, 70:786–815, 2024.

[26] Energy Systems Catapult. Roadmap study for tidal generation with electrolysis. Technical report, Energy Systems Catapult, 2022.

[27] European Marine Energy Centre (EMEC). Hydrogen projects, 2020. [Online].

[28] BIGHIT Consortium. Building innovative green hydrogen systems in isolated territories, 2020. [Online].

[29] J. L. B. Ferguson. *Technoeconomic Modelling of Renewable Hydrogen Supply Chains on Islands with Constrained Grids*. PhD thesis, University of Edinburgh, 2021.

[30] H Petter Langtangen. *A primer on scientific programming with Python*. Springer, 2016.

[31] Enrico Giglio, Ermando Petracca, Bruno Paduano, Claudio Moscoloni, Giuseppe Giorgi, and Sergej Antonello Sirigu. Estimating the cost of wave energy converters at an early design stage: A bottom-up approach. *Sustainability*, 15(8):6756, 2023.

[32] Mattia Petri, Emiliano Gorr, and Giuseppe Giorgi. Techno-economic modelling and comparative analysis of horizontal axis tidal energy converters. *Thesis dissertation, Politecnico di Torino, Italy*, 2025.

[33] KA Abed and AA El-Mallah. Capacity factor of wind turbines. *Energy*, 22(5):487–491, 1997.

[34] A Martinez and Gregorio Iglesias. Multi-parameter analysis and mapping of the levelised cost of energy from floating offshore wind in the mediterranean sea. *Energy conversion and management*, 243:114416, 2021.

[35] GL DNV. Marine operations and marine warranty. Technical report, Standard DVNGL-ST, 2016.

[36] Sam Porteous. Assessment of vessels required to serve floating offshore wind in the celtic sea. 2023.

[37] Francisco X Correia da Fonseca, Luís Amaral, and Paulo Chainho. A decision support tool for long-term planning of marine operations in ocean energy projects. *Journal of Marine Science and Engineering*, 9(8):810, 2021.

[38] Shanmukh Devarapali, Ashley Manske, Razieh Khayamim, Edwina Jacobs,

Bokang Li, Zeinab Elmi, and Maxim A Dulebenets. Electric tugboat deployment in maritime transportation: detailed analysis of advantages and disadvantages. *Maritime Business Review*, 9(3):263–291, 2024.

[39] Mitra Kamidelivand, Peter Deeney, Fiona Devoy McAuliffe, Kevin Leyne, Michael Togneri, and Jimmy Murphy. Scenario analysis of cost-effectiveness of maintenance strategies for fixed tidal stream turbines in the atlantic ocean. *Journal of Marine Science and Engineering*, 11(5):1046, 2023.

[40] Susanne Schnorr-Bäcker. Statistical monitoring for politics in the european union and the oecd, supporting economic growth and social progress: A comparison between europe 2020 and the oecd better life initiative. *Statistical Journal of the IAOS*, 34(3):353–378, 2018.

[41] Eva Segura, Rafael Morales, and José A Somolinos. Cost assessment methodology and economic viability of tidal energy projects. *Energies*, 10(11):1806, 2017.

[42] L Fingersh, Maureen Hand, and Alan Laxson. Wind turbine design cost and scaling model. Technical report, National Renewable Energy Lab.(NREL), Golden, CO (United States), 2006.

[43] David J Malcolm and A Craig Hansen. *WindPACT Turbine Rotor Design Study:*. National Renewable Energy Laboratory Golden, CO, USA, 2006.

[44] Turaj Ashuri, Joaquim RRA Martins, Michiel B Zaaijer, Gijs AM van Kuik, and Gerard JW van Bussel. Aeroservoelastic design definition of a 20 mw common research wind turbine model. *Wind energy*, 19(11):2071–2087, 2016.

[45] Ross O'Connell, Mitra Kamidelivand, Ioannis Polydoros, Christopher Wright, Paul Bonar, Alison J Williams, and Jimmy Murphy. The integration of tools for the techno-economic evaluation of fixed and floating tidal energy deployment in the irish sea. *Energies*, 16(22):7526, 2023.

[46] M Bianchi, AJ Arnal, M Astorkiza-Andres, J Clavell-Diaz, A Marques, and M Isasa-Sarralde. Life cycle and economic assessment of tidal energy farms in early design phases: Application to a second-generation tidal device. *Heliyon*, 10(12), 2024.

[47] Zerina Maksumic. Uk awards six tidal stream projects with contracts for difference in allocation round 6. https://www.offshore-energy.biz/uk-awards-six-tidal-stream-projects-with-contracts-for-\difference-in-allocation-round-6/, n.d. [Accessed 14-01-2025].

[48] European Marine Energy Centre (EMEC). Fall of warness scoping report. Technical report, European Marine Energy Centre Ltd, 2022.

[49] Natural Earth. 10m coastline — physical vectors. https://www.naturalearthdata.com/downloads/10m-physical-vectors/10m-coastline/, 2025. Accessed 10 November 2025.

[50] International Renewable Energy Agency. Green hydrogen: A guide to policy making, 2020.

[51] National Renewable Energy Laboratory. Hydrogen station compression, storage, and dispensing: Technical report, 2012.

Appendix A

Python Model Source Code

This appendix provides the full Python scripts used in the techno-economic analysis.

A.1 TE_0.py

```
############################################################
# TE_0 MAIN PIPELINE- TIDAL ENERGY
# ##########################################################
# - - - LOAD LIBRARIES - - - - - - - - - -
import matplotlib.pyplot as plt
import xarray as xr
import numpy as np
import os
import pandas as pd
from datetime import datetime
from scipy import stats

from TE_1 import tidalCurr_analysis,distance_from_shoreline,
    hydrogen_production_dynamic
from TE_2 import costs_computations,pltinstlld_cpcty,costs_hydrogen_prod

print("Script␣starts␣at", datetime.now().strftime("%Y-%m-%d␣%H:%M:%S"), "-␣-␣-␣-
    ␣-")
# - - - PREAMBLE - - - - - - - - - -
StartDate="2024-01-01T00:00"
EndDate="2024-12-31T23:00"
```

```
lon,lat=-2.84,59.14 #case SeaQ NL:  5.6,53.42   case UK -4.70, 53.32 # -4.5-0, 57-61
TEC_sel='Floating_Orbital02     '  #Torcado2 or Orbital02 or SeagenS1 or SeagenS2
type='HAT'#'HAT'or 'VAT'or'kite'

basepth='C:/Users/loren/Documents/Poli/Magistrale/Tesi/codice/' #
bathypath='C:/Users/loren/Documents/Poli/Magistrale/Tesi/codice/GEBCO_domain/'
filepath =basepth+'cmems_mod_nws_phy-uv_my_7km-2D_PT1H-i_1762424432480.nc' # it is modified in the function to the data you'll analyse: https://data.marine.copernicus.eu/product/GLOBAL_ANALYSISFORECAST_PHY_001_024/services
svpath = basepth+'graphicsLocation/'#Path for saving outputs
shoreline_file= basepth+"shoreline_NE10m/coastline_NE10m_Orkney.npy"

TECs = {
    "HAT": ["GBS_Torcado2", "Floating_Orbital02", "Monopile_SeagenS1", "Monopile_SeagenS2"],
    "VAT": ["GBS", "Floating", "Monopile"],
    "Kite": ["NA"]
}
TECeval = next((tec for tec in TECs[type] if TEC_sel in tec), None)

export_voltageAll={"GBS_Torcado2":180, "Floating_Orbital02":11, "Monopile_SeagenS1":25, "Monopile_SeagenS2":25}# kV

gebco_file = bathypath + 'gebco_2025_n61.0_s57.0_w-4.0_e0.0.nc'# https://download.gebco.net/
TR=30 # return period [years]
PrjtLftm = 30 # project lifetime, commonly established as 30 years
priceKwh=0.2872#Mean price of kWh in Europe in the 2nd half of 2024. https://ec.europa.eu/eurostat/statistics-explained/index.php?oldid=670662
r=0.05 # discount rate
# - - - EXTRACT TIDAL CURRENT DATA - - - - - - - - -
extractOn = 1
ds = xr.open_dataset(filepath)
ds_gebco = xr.open_dataset(gebco_file)

# Tidal current resource analysis - -
Xtar,Ytar,h,AEP,rated_current,rotor_diameter,nblades,power=tidalCurr_analysis(ds,ds_gebco,StartDate,EndDate,lon,lat,svpath,TECeval)
distShr=distance_from_shoreline(Xtar,Ytar,shoreline_file,svpath)
```

```
print(f'distance:{distShr}m')

#Hydrogen production (senza domanda e con domanda)
H2_results = hydrogen_production_dynamic(power, svpath,electrolyzer_size_factor
    =1.5)

turbines_per_structure=2 #2 for monopile and Floating, 1 for GBS
daily_demand = 0

daily_demand=0
H2_results = hydrogen_production_dynamic(power,turbines_per_structure, svpath,
    demand_daily=daily_demand,electrolyzer_size_factor=1.5               )

PEM_nom = H2_results['PEM_nominal_power']
COMP_nom = H2_results['compressor_nominal_power']
STORAGE_nom = H2_results['storage_nominal_mass']
H2_total = H2_results['H2_total']
# Economic analysis - -
rated_power_per_turbine=np.nanmean(power)
turbines_per_structure=2
export_voltage=export_voltageAll[TECeval]
number_of_structures=1
number_of_export_cables=1
rows_for_array,columns_for_array=number_of_structures,1

rated_current=np.nanmean(rated_current)



Costs=InstallCpct,LCOE,CapEx,Opex,LCOH=costs_computations(TECeval,AEP,h,distShr,
    nblades,rated_current,rotor_diameter,rated_power_per_turbine,
    turbines_per_structure,number_of_structures,export_voltage,
    number_of_export_cables,rows_for_array,columns_for_array, r,PrjtLftm,
    priceKwh,H2_total,PEM_nom, COMP_nom, STORAGE_nom)



cost_PEM, cost_compr, cost_storage, capex_hydrogen, opex_hydrogen, LCOH =
    costs_hydrogen_prod(
    H2_total, PEM_nom, COMP_nom, STORAGE_nom, CapEx, Opex, PrjtLftm, r
)
```

```


pltinstlld_cpcty(Costs,svpath)
print("Script␣ends␣at", datetime.now().strftime("%Y-%m-%d␣%H:%M:%S"), "-␣-␣-␣-␣-
    ")
```

A.2 TE_1.py

```
    ###########################################################
# TE_1 CURRENT ANALYSIS AND TIDAL ENERGY
# ###########################################################
import matplotlib.pyplot as plt
import numpy as np
import os
from scipy import stats
from mpl_toolkits.axes_grid1.inset_locator import inset_axes
import cartopy.io.img_tiles as cimgt
import cartopy.crs as ccrs
import matplotlib.patches as mpatches
import matplotlib.dates as mdates
import geopandas as gpd
import plotly.graph_objects as go
import seaborn as sns
import pandas as pd
from scipy.spatial import cKDTree
from pyproj import Transformer
import matplotlib.pyplot as plt
cartopy.feature as cfeature
import os





def plotLocation(X, Y, svpath):
    # Use satellite imagery tiles from ESRI
    tiler = cimgt.QuadtreeTiles()  # fallback if ESRI fails
    try:
        # ESRI satellite tiles
```

```
        tiler = cimgt.GoogleTiles(style='satellite')  # For a Google-like look (requires newer Cartopy)
    except:
        try:
            from cartopy.io.img_tiles import Stamen
            tiler = Stamen('terrain-background')
        except:
            pass  # fallback to default tiles if others fail

    projection = tiler.crs
    fig = plt.figure(figsize=(12, 10))
    ax_main = fig.add_subplot(1, 1, 1, projection=projection)
    extent_main = [X - 20, X + 25, Y - 12, Y + 12]
    ax_main.set_extent(extent_main, crs=ccrs.PlateCarree())
    ax_main.add_image(tiler, 6)  # zoom level
    ax_main.plot(X, Y, 'rD', markersize=10, transform=ccrs.PlateCarree(), label='DTnode')
    ax_main.legend(loc='lower left')
    ax_main.set_title("Assessed location")
    rect_lon_min = X - 0.9
    rect_lon_max = X + 0.9
    rect_lat_min = Y - 0.9
    rect_lat_max = Y + 0.9
    width = rect_lon_max - rect_lon_min
    height = rect_lat_max - rect_lat_min
    rect = mpatches.Rectangle(
        (rect_lon_min, rect_lat_min), width, height,
        linewidth=1, edgecolor='yellow', facecolor='none',
        transform=ccrs.PlateCarree(), zorder=10
    )
    ax_main.add_patch(rect)
    ax_inset = fig.add_axes([0.56, 0.52, 0.34, 0.35], projection=projection)
    ax_inset.set_extent([rect_lon_min, rect_lon_max, rect_lat_min, rect_lat_max], crs=ccrs.PlateCarree())
    ax_inset.add_image(tiler, 8)
    ax_inset.plot(X, Y, 'rD', markersize=10, transform=ccrs.PlateCarree())
    rect_inset = mpatches.Rectangle(
    (rect_lon_min, rect_lat_min), width, height,
    linewidth=1, edgecolor='yellow', facecolor='none',
    transform=ccrs.PlateCarree(), zorder=10)
    ax_inset.add_patch(rect_inset)
```

```python
    plt.savefig(os.path.join(svpath, "01_LocationMap.jpg"), dpi=300, bbox_inches='tight')
    plt.close()


def distance_from_shoreline(Xtar, Ytar, shoreline_file, svpath):
    # 1) Carico i punti costa (lon, lat)
    pts = np.load(shoreline_file)  # shape (N,2): [lon, lat]
    lons = pts[:,0]; lats = pts[:,1]

    # 2) Proiezione locale in metri (AEQD centrata sul punto target)
    aeqd = f"+proj=aeqd +lat_0={Ytar} +lon_0={Xtar} +datum=WGS84 +units=m +no_defs"
    transformer = Transformer.from_crs("EPSG:4326", aeqd, always_xy=True)
    Xp, Yp = transformer.transform(lons, lats)         # costa in metri
    Xtar_p, Ytar_p = transformer.transform(Xtar, Ytar) # target in metri

    # 3) KDTree nearest neighbor
    tree = cKDTree(np.c_[Xp, Yp])
    dist_m, idx = tree.query([Xtar_p, Ytar_p], k=1)
    closest_lon, closest_lat = lons[idx], lats[idx]

    # 4) Plot
    fig = plt.figure(figsize=(8,6))
    ax = plt.axes(projection=ccrs.PlateCarree())
    ax.add_feature(cfeature.OCEAN, facecolor="lightblue")
    ax.add_feature(cfeature.LAND, facecolor="lightgreen", edgecolor="black")
    ax.scatter(lons, lats, s=1, color='black', label="Coastline pts")
    ax.scatter([Xtar], [Ytar], color='red', s=40, label="Target")
    ax.plot([Xtar, closest_lon], [Ytar, closest_lat], color='cyan',
            lw=2, label=f"Distance: {dist_m/1000:.2f} km")
    ax.set_xlim(Xtar-0.1, Xtar+0.1)  # zoom ~0.1  611 km
    ax.set_ylim(Ytar-0.1, Ytar+0.1)
    ax.legend()
    ax.set_title("Nearest shoreline")
    plt.savefig(os.path.join(svpath, 'DistanceFromShore.png'),
                dpi=300, bbox_inches='tight')
    plt.close(fig)

    print(f"xnode:{Xtar}, ynode:{Ytar}")
    print(f"xshore:{closest_lon}, yshore:{closest_lat}")
```

```
    print(f"distance: {dist_m:.2f} m")
    return dist_m


# Tidal current resource analysis - -
def tidalCurr_analysis(ds,ds_gebco,StartDate,EndDate,lon,lat,svpath,TEC_sel):
    rho = 1024 # water density [kg/m^3]
    g = 9.81 # gravity [m/s^2]
    a,b=7,0.4 #constants of vertical profile
    Cp=0.4#tidal power coefficient
    ratedPowerHAT={"GBS_Torcado2":1000,"Floating_OrbitalO2":1000,"Monopile_SeagenS1":600, "Monopile_SeagenS2":1000}#kW
    ratedCurrHAT={"GBS_Torcado2":2,"Floating_OrbitalO2":2.65,"Monopile_SeagenS1":2.35, "Monopile_SeagenS2":2.5}#m/s
    CutInVelHAT={"GBS_Torcado2":1,"Floating_OrbitalO2":1,"Monopile_SeagenS1":0.7, "Monopile_SeagenS2":1}#m/s
    CutOutVelHAT={"GBS_Torcado2":4,"Floating_OrbitalO2":4,"Monopile_SeagenS1":4, "Monopile_SeagenS2":4}#m/s
    RotDiamHAT={"GBS_Torcado2":20,"Floating_OrbitalO2":20,"Monopile_SeagenS1":20, "Monopile_SeagenS2":20}#m
    RotElevHAT={"GBS_Torcado2":25,"Floating_OrbitalO2":15,"Monopile_SeagenS1":15, "Monopile_SeagenS2":15}#m
    NbladesHAT={"GBS_Torcado2":3,"Floating_OrbitalO2":3,"Monopile_SeagenS1":3, "Monopile_SeagenS2":3}#number
    utide_time=ds['uo'].sel(time=slice(StartDate, EndDate))
    utide_my_loc=utide_time.sel(latitude=lat, longitude=lon, method='nearest')
    utide_urated = utide_my_loc.sel(time=slice(StartDate, EndDate))
    Xr=utide_urated.longitude.values
    Yr=utide_urated.latitude.values
    print(f"Requested coordinates: lon={lon}, lat={lat}")
    print(f"Nearest grid point used: lon={np.round(Xr.item(),3)}, lat={np.round(Yr.item(),3)}")

    vtide_time=ds['vo'].sel(time=slice(StartDate, EndDate))
    vtide_my_loc=vtide_time.sel(latitude=lat, longitude=lon, method='nearest')
    vtide_vrated = vtide_my_loc.sel(time=slice(StartDate, EndDate))

    custom_bin_width = 0.05


    current=np.sqrt(utide_urated**2+vtide_vrated**2)
```

```
print(f"Mean current = {float(np.nanmean(current)):.3f} m/s, Max = {float(np.nanmax(current)):.3f} m/s")

maxi=np.max(current)
mini=np.min(current)
n_bins = int(np.ceil((maxi-mini) / custom_bin_width))
bin_edges = np.arange(-np.ceil(abs(mini) / custom_bin_width) * custom_bin_width, np.ceil(abs(maxi) / custom_bin_width) * custom_bin_width + custom_bin_width, custom_bin_width)
countp, bin_countsp=np.histogram(current, bins=len(bin_edges))
pdfp=countp/sum(countp)
cdfp = np.cumsum(pdfp)

plotLocation(lon,lat,svpath)


fig, axs = plt.subplots(2, 1, figsize=(12, 8))

axs[0].plot(utide_urated['time'], utide_urated, label='U rated')
axs[0].set_title('U rated tide')
axs[0].set_ylabel('Velocity [m/s]')
axs[0].grid()
axs[0].xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))  # <-- Aqu el formato
axs[0].xaxis.set_major_locator(mdates.MonthLocator(interval=3))  # Cada 3 meses, ajusta si quieres

axs[1].plot(vtide_vrated['time'], vtide_vrated, label='V rated', color='orange')
axs[1].set_title('V rated tide')
axs[1].set_ylabel('Velocity [m/s]')
axs[1].set_xlabel('Time (year-month)')  # <-- Etiqueta del eje
axs[1].grid()
axs[1].xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))  # <-- Aqu tambin
axs[1].xaxis.set_major_locator(mdates.MonthLocator(interval=3))

fig.autofmt_xdate()  # Rota las fechas para mayor legibilidad
plt.tight_layout()
plt.savefig(os.path.join(svpath, "02_U_V_ratedTides.jpg"), dpi=300, bbox_inches='tight')
```

```
Dir_deg = (np.degrees(np.arctan2(utide_urated.values, vtide_vrated.values))
+ 360) % 360
Dir_deg = Dir_deg.flatten()

bin_edgesR = np.arange(0, 361, 30)  # [0, 30, 60, ..., 360]
bin_centersR = bin_edgesR[:-1] + 15  # [15, 45, ..., 345]

bin_sums = np.zeros(len(bin_centersR))
bin_counts = np.zeros(len(bin_centersR))

for angle, speed in zip(Dir_deg, current):
    bin_idx = int(angle // 30) % 12  # Asegura rango 0-11
    bin_sums[bin_idx] += speed
    bin_counts[bin_idx] += 1

bin_means = np.divide(bin_sums, bin_counts, out=np.zeros_like(bin_sums),
where=bin_counts!=0)

fig_rose = go.Figure()
fig_rose.add_trace(go.Barpolar(
    r=bin_means,
    theta=bin_centersR,
    width=[30]*len(bin_centersR),
    marker_color='skyblue',
    opacity=0.8,
    name='Current␣Speed␣[m/s]'
))

fig_rose.update_layout(
    title='Rose␣plot␣of␣Current␣Speed',
    polar=dict(
        radialaxis=dict(title='[m/s]', tickfont_size=10, gridcolor='
lightgrey'),
        angularaxis=dict(direction="clockwise", rotation=90)
    )
)
fig_rose.write_image(os.path.join(svpath, f'03_CurrentRosePlot.jpg.png'))
```

```
fig = plt.figure(figsize=(12, 10))
plt.hist(current , bins=bin_edges, edgecolor='k', alpha=0.7)
plt.xlabel('Current [m/s]')
plt.ylabel('Frequency')
plt.title('Bin Analysis of total current')
plt.tight_layout()
plt.savefig(os.path.join(svpath, "04_Bins_totalCurrents.jpg"), dpi=300,
bbox_inches='tight')

fig = plt.figure(figsize=(12, 10))
plt.plot(bin_countsp[1:], pdfp*100, color="red" , label="PDF positive")
plt.plot(bin_countsp[1:], cdfp*100, label="CDF positive")
plt.xlabel('Current [m/s]')
plt.ylabel('Ocurrence [%]')
plt.legend()
plt.tight_layout()
plt.savefig(os.path.join(svpath, "05_PDF_CDF_currents.jpg"), dpi=300,
bbox_inches='tight')


depth_point = ds_gebco['elevation'].sel(lon=Xr.item(), lat=Yr.item(), method
='nearest')
h=abs(depth_point.item())
U_mean_given=1.5 #measured curren

#rotor depth
z=13.9 d#epth measured current
z_mio=0.495 # ADCP or depth of logged records
print('Water depth is:',h,'m')
print('Rotor depth is:',z,'m')

U_mean_depth=current/((h-z_mio)/(b*h))**(1/a)
U_z=((h-z)/(b*h))**(1/a)*U_mean_depth
U_z_mean=U_z.mean(dim='time')


current_scaled=current*U_mean_given/U_z_mean.values
U_mean_depth=current_scaled/((h-z_mio)/(b*h))**(1/a)
U_z=((h-z)/(b*h))**(1/a)*U_mean_depth
U_z_mean=U_z.mean(dim='time')
z=RotElevHAT[TEC_sel]
```

```
    current=((h-z)/(b*h))**(1/a)*U_mean_depth.values
    current = np.real(current)
    print(f"Mean current depth = {float(np.mean(current)):.3f} m/s, Max current depth = {float(np.max(current)):.3f} m/s")

    plt.figure(figsize=(12, 8))
    plt.plot(utide_urated['time'], current)
    plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
    plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=3))
    plt.xlabel('Time (year-month)')
    plt.ylabel('Current [m/s]')
    plt.title(f'Current at rotor. Rotor depth={round(z,2)} m')
    plt.ylim(0, max(current)+0.1)
    plt.grid()
    plt.tight_layout()
    plt.savefig(os.path.join(svpath, "06_Rotordepth_currents.jpg"), dpi=300, bbox_inches='tight')


    current_max=np.max(np.real(current))
    velocity_data = np.real(np.ravel(current))
    velocity_data = velocity_data[~np.isnan(velocity_data)]
    fig, ax = plt.subplots(figsize=(8,5))
    iqr = stats.iqr(velocity_data)
    bin_width = 2 * iqr / (len(velocity_data) ** (1/3))
    num_bins = int(np.ceil((current_max - 0) / bin_width))
    ax.hist(velocity_data, bins=num_bins, density=True, alpha=0.8, label='Histogram')
    mu, sigma = stats.norm.fit(velocity_data)
    x = np.linspace(0, current_max, 100)
    pdf_fitted = stats.norm.pdf(x, mu, sigma)
    ax.plot(x, pdf_fitted, 'r-', lw=2, label='Normal PDF') #label=f'Normal PDF\n={mu:.2f}, ={sigma:.2f}')
    ax.set_xlim(0,current_max)
    ax.set_title(f'Velocity Distribution with Fitted Normal Distribution ({TEC_sel})')
    ax.set_xlabel('Current [m/s]')
    ax.set_ylabel('Probability density')
    ax.legend()
    ax.grid(True, alpha=0.3)
    plt.tight_layout()
```

```
    plt.savefig(os.path.join(svpath, f'07_VelocityDistrNorm{TEC_sel}.png'), dpi
    =300, bbox_inches='tight')

    ### TEC assessed
    U_in= CutInVelHAT[TEC_sel]   #m/s
    U_out= CutOutVelHAT[TEC_sel] #m/s

    power=0.5*rho*Cp*np.abs(current)**3*(np.pi*(RotDiamHAT[TEC_sel]/2)**2)/1000#
    in kW
    power[current<U_in]=0#must be 0
    power[power > ratedPowerHAT[TEC_sel] ] = ratedPowerHAT[TEC_sel]
    power[current>U_out]=0#must be 0
    total_power = power.sum()#in kWh
    mean_power =power.mean()
    AEP=total_power#in KWh
    CF=mean_power/(ratedPowerHAT["Floating_Orbital02"])




    print(f"CF={CF*100}␣%")
    print(f"AEP={AEP}␣kWh")

    months = pd.to_datetime(utide_urated['time'].values).month
    df_tw = pd.DataFrame({
    'tw': power.ravel(),
    'month': months.ravel(),
    'Dir': Dir_deg.ravel()})
    fig, ax = plt.subplots(figsize=(8, 5))
    sns.boxplot(data=df_tw, x='month', y='tw', ax=ax)
    ax.set_title('Monthly␣boxplot␣of␣Tidal␣power␣[kW]')
    plt.tight_layout()
    plt.savefig(os.path.join(svpath, f'08_PowerBoxplots.png'))

    plt.figure(figsize=(13,5))
    plt.plot(utide_urated['time'], power,color='green')
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
    ax.set_xlabel("Time␣(year-month)")
    plt.ylabel('Power␣[kW]')
    plt.title(f'TEC␣{TEC_sel}')
    plt.tight_layout()
```

```
    plt.savefig(os.path.join(svpath, f'09_DeliveredPower{TEC_sel}.png'), dpi=300, bbox_inches='tight')
    AEP=AEP/1000# in MWh



    time = pd.to_datetime(utide_urated['time'].values)
    df_pw = pd.DataFrame(power, index=time)
    Pw_global_mean = df_pw.mean(axis=0)
    unique_months = set(df_pw.index.month)
    full_year = set(range(1, 13)).issubset(unique_months)

    df_pw['month'] = df_pw.index.month
    monthly_mean = df_pw.groupby('month').mean(numeric_only=True)
    # Monthly variability: use correct denominator
    MV_numerator = (monthly_mean.max() - monthly_mean.min())
    MV_denominator = df_pw.mean(axis=0) if not full_year else Pw_global_mean
    MV =(MV_numerator / MV_denominator).values
    MV= round(MV[0],2)
    season_map = {12: 1, 1: 1, 2: 1,
                  3: 2, 4: 2, 5: 2,
                  6: 3, 7: 3, 8: 3,
                  9: 4, 10: 4, 11: 4}
    df_pw['season'] = df_pw.index.month.map(season_map)
    seasonal_mean = df_pw.groupby('season').mean(numeric_only=True)
    # Seasonal variability: one value per node
    SV_numerator = (seasonal_mean.max() - seasonal_mean.min())
    SV = (SV_numerator / Pw_global_mean).values
    SV = round(SV[0],3)
    print(f"MV = {np.round(MV,2)}")
    print(f"SV = {np.round(SV,2)}")


    return Xr,Yr,h,AEP,current,RotDiamHAT[TEC_sel],NbladesHAT[TEC_sel],power #is current or rated current punctual?


def hydrogen_production_dynamic(power_series,turbines_per_structure, svpath, demand_daily=None,
                               electrolyzer_size_factor=1, electrolyzer_eff=0.7,
```

```
                                LHV_H2=33.33, comp_energy_per_kg=3, plot_results=True):

    power_series = np.array(power_series)
    P_avg = np.nanmean(power_series)*turbines_per_structure

    P_PEM_nom = P_avg * electrolyzer_size_factor #dimensionamento
    P_PEM_nom=500 #kW
    comp_factor = 1 / (1 + comp_energy_per_kg / LHV_H2)
    if demand_daily is not None:
        # Se la domanda  in kWh/giorno, la converto in kW medi
        if demand_daily > 1000:
            demand_hourly = demand_daily / 24
        else:
            demand_hourly = demand_daily

        P_available = power_series - demand_hourly
        P_available[P_available < 0] = 0  # nessuna potenza per H se domanda > produzione
    else:
        P_available = power_series  # Tutta la potenza va all'elettrolizzatore

    P_input = np.minimum(P_available * comp_factor, P_PEM_nom)
    H2_hourly = (P_input * electrolyzer_eff) / LHV_H2
    E_comp_hourly = H2_hourly * comp_energy_per_kg
    E_comp_total_MWh = np.nansum(E_comp_hourly) / 1000
    P_comp_hourly = E_comp_hourly
    P_comp_peak = float(np.nanmax(P_comp_hourly))
    H2_total = np.nansum(H2_hourly)
    CF_PEM = np.nanmean(P_input) / P_PEM_nom if P_PEM_nom > 0 else np.nan
    daily_h2_production = H2_total / 12 / 30  # kg/giorno

    #  Bilancio energetico annuale
    E_total_MWh = np.nansum(power_series) / 1000
    if demand_daily is not None:
        E_demand_MWh = (demand_hourly * len(power_series)) / 1000
    else:
        E_demand_MWh = 0
    E_H2_MWh = np.nansum(P_input) / 1000

    # --- Stampa dei risultati ---
```

```
    print("\n---␣DYNAMIC␣HYDROGEN␣PRODUCTION␣ESTIMATION␣---")
    print(f"Average␣tidal␣power:␣␣␣␣␣␣␣␣␣␣{P_avg:.2f}␣kW")
    print(f"Electrolyzer␣nominal␣power:␣␣␣{P_PEM_nom:.2f}␣kW")
    print(f"Capacity␣Factor␣(PEM):␣␣␣␣␣␣␣␣{CF_PEM*100:.1f}%")
    print(f"Compressor␣size␣(peak):␣␣␣␣␣␣␣{P_comp_peak:.2f}␣kW")
    print(f"Compression␣energy␣required:␣␣{E_comp_total_MWh:.2f}␣MWh/year")
    print(f"Total␣H␣produced:␣␣␣␣␣␣␣␣␣␣␣␣{H2_total/1000:.2f}␣ton/year")
    print(f"Nominal␣storage␣(1␣day␣avg):␣␣{daily_h2_production:.2f}␣kg/day")

    print("\n---␣ENERGY␣ALLOCATION␣SUMMARY␣---")
    print(f"Total␣tidal␣energy:␣␣␣␣␣␣␣␣␣␣␣{E_total_MWh:.1f}␣MWh/year")
    print(f"Energy␣for␣demand:␣␣␣␣␣␣␣␣␣␣␣␣{E_demand_MWh:.1f}␣MWh/year␣({
E_demand_MWh/E_total_MWh*100:.1f}%)")
    print(f"Energy␣for␣H␣system:␣␣␣␣␣␣␣␣␣{E_H2_MWh:.1f}␣MWh/year␣({E_H2_MWh/
E_total_MWh*100:.1f}%)")

    # --- Grafico della produzione oraria ---
    if plot_results:
        plt.figure(figsize=(10, 4))
        plt.plot(H2_hourly, color='tab:blue', linewidth=0.8)
        plt.title("Hydrogen␣hourly␣production␣[kg/h]")
        plt.xlabel("Month")
        plt.ylabel("H␣[kg/h]")
        plt.grid(True, linestyle='--', alpha=0.5)

        # Imposta i tick ogni 3 mesi
        hours_per_month = 8760 / 12
        months = np.arange(0, 8760 + hours_per_month * 3, hours_per_month * 3)
        month_labels = ['Jan', 'Apr', 'Jul', 'Oct', 'Jan']

        plt.xticks(months, month_labels)

        plt.tight_layout()
        plt.savefig(os.path.join(svpath, "10_Hydrogen_prod.jpg"), dpi=300,
bbox_inches='tight')
        plt.close()
        # Grafico della potenza disponibile (solo se esiste domanda)
        if demand_daily is not None:
            plt.figure(figsize=(10, 4))
            plt.plot(power_series, label='Power␣available␣[kW]', alpha=0.6)
            plt.plot(P_available, label='Power␣for␣H␣[kW]', color='tab:blue')
```

```
        plt.axhline(demand_hourly, color='red', linestyle='--', label='Demand [kW]')
        plt.legend()
        plt.title("Power allocation: demand vs hydrogen production")
        plt.xlabel("Ore")
        plt.ylabel("Power [kW]")
        plt.tight_layout()
        plt.savefig(os.path.join(svpath, "11_Power_allocation.jpg"), dpi=300, bbox_inches='tight')
        plt.close()

    # --- Ritorno dei risultati ---
    return {
        # Produzione idrogeno
        'H2_hourly': H2_hourly,
        'H2_total': H2_total,
        'storage_nominal_mass': daily_h2_production,
        # Elettrolizzatore e compressore
        'PEM_nominal_power': P_PEM_nom,
        'capacity_factor': CF_PEM,
        'compressor_nominal_power': P_comp_peak,
        'compression_energy_total': E_comp_total_MWh,
        # Bilancio energetico
        'E_total_MWh': E_total_MWh,
        'E_H2_MWh': E_H2_MWh,
        'E_demand_MWh': E_demand_MWh
    }
```

A.3 TE_2.py

```
    ###########################################################
# TE_2 ECONOMIC ANALYSIS ON TIDAL ENERGY
# ##########################################################
import os
import sys
import math
import numpy as np
from numpy_financial import irr, npv
import matplotlib.pyplot as plt
```

```
from scipy.interpolate import griddata
from TE_1 import hydrogen_production_dynamic
sys.path.append(os.path.join(os.path.dirname(__file__), 'additional_func'))
# libraries for Floating TECs
from mooring_prelay_ODYSSEY import compute_mooring_prelay_costs_Odyssey
from towing_THOR import compute_towing_costs_Thor
from mooring_connection_ODYSSEY import compute_mooring_connection_costs_Odyssey
from support_mooring_connection_USKMOOR import
    compute_support_mooring_connection_costs_Uskmoor
from blades_connection_ODYSSEY import compute_blades_connection_costs_Odyssey
from opex import compute_opex

# libraries for GBS TECs
from installation_costs_NEPTUNE import compute_installation_costs_Neptune
from installation_costs_AKER_WAYFARER import
    compute_installation_costs_Aker_Wayfarer
from opex import compute_opex

# libraries for Monopile TECs
import math
from monopile_installation_RAMBIZ import compute_installation_costs_Rambiz
from installation_costs_AKER_WAYFARER import
    compute_installation_costs_Aker_Wayfarer
from opex import compute_opex


def economic_parameters_Floating (AEP,water_depth,
    distance_from_shore,
    number_of_blades,
    rated_current,
    rotor_diameter,
    rated_power_per_turbine,
    turbines_per_structure,
    number_of_structures,
    export_voltage,
    number_of_export_cables,
    rows_for_array,
    columns_for_array, r,PrjtLftm, priceKwh) :

    fuel_price = 515 # /tonn

```

```
    # Assumed Variables
    output_voltage_generator_array = 0.69  # kV
    power_factor = 0.95
    gearbox_ratio = 1/98
    thrust_coefficient = 0.9
    cover_to_rotor_diameter_ratio = 0.1333

    # Other Variables
    # safety_factor_for_yaw =
    chain_diameter = 0.076  # m
    foundation_weigth = 360 # tons
    anchor_weigth = 140
    scope_ratio = 5  # length/depth
    steel_price_anchor_chain = 0.5  # /kg
    steel_price_foundation_platform = 2.5  # /kg
    # monopile_diameter =
    number_of_mooring_lines = 4*number_of_structures
    devex_of_capex = 0.05  #5%
    spare_part_cost = 0.15 #15%
    repair_time_reduction = 0.3 #0%
    insurance_cost_of_capex = 0.01 #1%

    # secondary variables
    # Parametri noti
    rotor_radius = rotor_diameter / 2  # m
    flow_speed = rated_current  #(m/s)

    # TSR: Tip Speed Ratio
    if number_of_blades==3:
        TSR = 4.5
    else:
        TSR = 6

    # Low-speed shaft angular velocity (rad/s)
    omega_low = TSR * flow_speed / rotor_radius

    # Low-speed shaft speed (rpm)
    low_speed_rpm = (omega_low * 60) / (2 * math.pi)

    # Low-speed shaft torque (Nm)
    power_watts = rated_power_per_turbine
```

```
    torque_low = power_watts / omega_low

    # High-speed shaft speed and torque
    gearbox_ratio = 1/98
    omega_high = omega_low / gearbox_ratio
    torque_high = torque_low * gearbox_ratio

    # Apparent Power (kVA)
    apparent_power_single = power_watts / power_factor

    # Array Rated Power
    array_rated_power = rated_power_per_turbine * turbines_per_structure *
    number_of_structures
    array_apparent_power = (array_rated_power) / power_factor

    # Current per turbine (A)
    array_voltage = output_voltage_generator_array
    current_per_turbine = apparent_power_single/(math.sqrt(3)*array_voltage)

    # Array cable voltage (kV)
    Array_cable_voltage=output_voltage_generator_array*rows_for_array

    # Export cable current (A)
    export_cable_current = array_apparent_power/(math.sqrt(3)*export_voltage)/
    number_of_export_cables
    # Cover diameter
    cover_diameter = rotor_diameter * cover_to_rotor_diameter_ratio
    # Thrust force on rotor (N)
    seawater_density = 1025  # kg/m
    rotor_area = math.pi * (rotor_radius ** 2)
    thrust_force = 0.5 * seawater_density * (flow_speed ** 2) * rotor_area *
    thrust_coefficient/1000
    # Device spacing
    spacing_along_row = rotor_diameter * 2.5
    spacing_along_column = rotor_diameter * 10
    # Length each array line
    length_per_line = spacing_along_row * (columns_for_array - 1)
    total_array_length = length_per_line * rows_for_array
    # Transformers and switchgear (1 per row)
    number_of_transformers = 1
    # COST FUNCTIONS
```

```
    # INFLATION AND CURRENCY CORECTIONS
    CPI_USA_2002 =179.9
    CPI_USA_2021 = 270.97
    CPI_Spain_2017 = 95
    CPI_England_2013 = 98.52
    CPI_Ireland_2015 = 82.8
    CPI_Denmark_2020 =103.4
    CPI_Cile_2017 = 2.2
    CPI_Europe_2020 = 106.1

    CPI_USA_2024 = 313.1
    CPI_Spain_2024 = 115
    CPI_England_2024 = 133.4
    CPI_Ireland_2024 = 100.5
    CPI_Denmark_2024 = 118.7
    CPI_Cile_2024 = 4.2
    CPI_Europe_2024 = 130.6

    Ratio_USA_2002 = CPI_USA_2024/CPI_USA_2002
    Ratio_USA_2021= CPI_USA_2024/CPI_USA_2021
    Ratio_Spain = CPI_Spain_2024/CPI_Spain_2017
    Ratio_England = CPI_England_2024/CPI_England_2013
    Ratio_Ireland = CPI_Ireland_2024/CPI_Ireland_2015
    Ratio_Denmark = CPI_Denmark_2024/CPI_Denmark_2020
    Ratio_Cile = CPI_Cile_2024/CPI_Cile_2017
    Ratio_Europe = CPI_Europe_2024/CPI_Europe_2020
    Dollar_to_Euro_conv= 0.92

    # BLADE
    c_blade= 40 # /m
    cost_per_blade = c_blade*(rotor_diameter/2)**2.7
    cost_blades = cost_per_blade*number_of_blades*Ratio_Spain

    # HUB
    c_hub = 1000
    cost_hub = c_hub*rotor_diameter/2*Ratio_Spain

    # PITCH SYSTEM
    cost_pitch = 2.28*0.2106*rotor_diameter**2.6578*Ratio_USA_2002*
    Dollar_to_Euro_conv
    # YAW SYSTEM (not calculated for Floating technology)
```

```
    #max_moment = thrust_force*cover_diameter/2*safety_factor_for_yaw #kN*m
    #yaw_diameter = 0.00009*max_moment+1.53
    #mass_yaw_bearings = 0.0152*(max_moment/yaw_diameter-36)**1.489
    #cost_yaw = 0

    # BRAKE SYSTEM
    brake_mass = 0.19894*rated_power_per_turbine  # kg
    cost_brake = 10*brake_mass*Ratio_USA_2002*Dollar_to_Euro_conv  #

    # GEARBOX
    gearbox_type = "Three-stage"
    gearbox_mass = 70.94*torque_low**0.759  # kg
    cost_gearbox = 10*gearbox_mass*Ratio_USA_2002*Dollar_to_Euro_conv  #

    # GENERATOR
    generator_type = "Three-stage"
    generator_mass = 6.47*(50*math.pi*torque_high)**0.9223  # kg
    cost_generator = 65*50*math.pi*torque_high*Ratio_USA_2002*
    Dollar_to_Euro_conv  #

    # LOW SPEED SHAFT
    cost_shaft_unit= 500 #/m
    cost_low_speed_shaft= cost_shaft_unit*rotor_diameter/2*Ratio_Spain

    # MAIN BEARINGS
    bearings_mass=3.5866*thrust_force-475.35
    cost_main_bearings = 2*17.6*bearings_mass*Ratio_USA_2002*Dollar_to_Euro_conv

    # ROTOR
    cost_rotor=cost_blades+cost_hub+cost_pitch+cost_low_speed_shaft  # blade +
    hub

    # PTO
    cost_PTO= cost_brake + cost_gearbox + cost_generator + cost_main_bearings

    # NACELLE COVER
    cost_fraction= 0.21 #21%
    cost_nacelle_cover=cost_fraction*(cost_rotor+cost_PTO)/(1-cost_fraction)

    #NACELLE
    cost_nacelle = cost_rotor + cost_PTO + cost_nacelle_cover
```

```
    # VARIOUS ELECTRICAL COMPONTENTS
    Cost_Power_Converter = 79*rated_power_per_turbine*Ratio_USA_2002*Dollar_to_Euro_conv
    Cost_Dry_Transformer = 11879*array_apparent_power/1000*Ratio_USA_2021*Dollar_to_Euro_conv
    Cost_Switchgear = 14018*export_voltage*Ratio_USA_2021*Dollar_to_Euro_conv
    Cost_base_offshore = 0
    Cost_wet_connector = 200000*Ratio_England

    # ARRAY POWER CABLE COST
    CSA_array = 28.348*math.exp(0.0044*current_per_turbine) #mm^2
    n_CSA_array = 0.6553+0.0035*CSA_array
    if output_voltage_generator_array<=10:
        n_V_array = -0.0021*output_voltage_generator_array**2+0.0607*output_voltage_generator_array+0.6076
    else:
        n_V_array=0.9819+0.0078*output_voltage_generator_array

    Unit_cost_array_cable = 200*n_CSA_array*n_V_array # /m
    Cost_array_cables = 0 #

    # EXPORT POWER CABLE COST
    CSA_export = 28.348*math.exp(0.0044*export_cable_current)
    n_CSA_export = 0.6553+0.0035*CSA_export
    if export_voltage<=10:
        n_V_export = -0.0021*export_voltage**2+0.0607*export_voltage+0.6076
    else:
        n_V_export=0.9819+0.0078*export_voltage

    Unit_cost_export_cable = 200*n_CSA_export*n_V_export
    Cost_export_cable = Unit_cost_export_cable*distance_from_shore*Ratio_Ireland

    # UMBILICAL POWER CABLE COST
    Umbilical_Unit_cost_export_cable = Unit_cost_export_cable*1.3
    Umbilical_cost_export_cable = Umbilical_Unit_cost_export_cable*water_depth*Ratio_Ireland

    # CABLE INSTALLATION
    unit_cost_laying = 100  # /m
```

```
    unit_cost_drilled_duct = 282  # /m
    fraction_drilled_duct_export = 1/3  # solo per cavo export
    array_cable_cost= unit_cost_laying*total_array_length  #
    export_cable_cost = (unit_cost_laying*(1-fraction_drilled_duct_export)+
    unit_cost_drilled_duct*fraction_drilled_duct_export)*distance_from_shore  #
    # FOUNDATION/PLATFORM COST
    cost_foundation= foundation_weigth*steel_price_foundation_platform*1000 #

    # ONSHORE COMPONENT PREPARATION
    time_per_component = 1  # h
    workers_for_component_prep = 6  # numero di operai
    worker_cost_per_hour = 50  # /h

    # Quantit di componenti
    total_number_of_blades = number_of_blades*turbines_per_structure*
    number_of_structures
    number_of_chain_lines = number_of_mooring_lines
    number_of_anchors = number_of_mooring_lines
    number_of_shackles = 3*number_of_chain_lines
    total_number_of_elements =total_number_of_blades+number_of_chain_lines+
    number_of_anchors+number_of_shackles+number_of_structures

    # Tempi totali e costi
    total_prep_time = total_number_of_elements*time_per_component  # h
    workers_prep_cost = workers_for_component_prep*worker_cost_per_hour*
    total_prep_time  #

    # INSTALLATION

    # Installation time
    Chain_time= 22 # h
    Anchor_time = 12 # h
    Connection_of_elements_for_mooring_time = 10 # h
    Blade_connection_time = 6 # h
    GBS_deployment = 1.5 # days
    Monopile_deployment = 2.5 # days
    GBS_turbine_deployment = 1 # h

    # mooring cost
    mooring_density = 0.0219*(10**6)*chain_diameter**2
    chain_length_per_line= water_depth*scope_ratio
```

```
    mass_chain = mooring_density*chain_length_per_line
    cost_chain_per_line = mass_chain*steel_price_anchor_chain
    anchor_cost = anchor_weigth*steel_price_anchor_chain*1000
    cost_per_mooring_line = cost_chain_per_line+anchor_cost
    cost_mooring_system= cost_per_mooring_line*number_of_mooring_lines

    Cost_mooring_prelay, Workers_mooring_prelay =
    compute_mooring_prelay_costs_Odyssey(Chain_time,Anchor_time,
    number_of_mooring_lines,fuel_price,distance_from_shore,Ratio_Europe)
    Cost_towing, Workers_towing = compute_towing_costs_Thor(fuel_price,
    distance_from_shore,Ratio_Europe)
    Cost_mooring_connection,Workers_mooring_connection =
    compute_mooring_connection_costs_Odyssey(
    Connection_of_elements_for_mooring_time,number_of_mooring_lines,fuel_price,
    distance_from_shore,Ratio_Europe)
    Cost_support_mooring_connection,Workers_support=
    compute_support_mooring_connection_costs_Uskmoor(
    Connection_of_elements_for_mooring_time,number_of_mooring_lines,fuel_price,
    distance_from_shore,Ratio_Europe)
    Cost_blades_connection, Workers_blades_connection =
    compute_blades_connection_costs_Odyssey(Blade_connection_time,
    number_of_blades,turbines_per_structure,fuel_price,distance_from_shore,
    Ratio_Europe)

    Cost_installation = Cost_mooring_prelay+Workers_mooring_prelay+Cost_towing+
    Workers_towing+Cost_mooring_connection+ \
                        Workers_mooring_connection+
    Cost_support_mooring_connection+Workers_support+Cost_blades_connection+ \
                        Workers_blades_connection+workers_prep_cost

    # CAPEX
    Cost_installation_Total = (Cost_installation+export_cable_cost)*
    number_of_structures+array_cable_cost
    Cost_Nacelle_Array= cost_nacelle*turbines_per_structure*number_of_structures
    cost_foundation_array= cost_foundation*number_of_structures+
    cost_mooring_system
    cost_connectors_total= Cost_wet_connector*number_of_structures
    cost_Electrical= (Cost_Power_Converter*turbines_per_structure)*
    number_of_structures+(Cost_Switchgear+Cost_Dry_Transformer)*
    number_of_transformers
```

```
    cost_cables_total= Umbilical_cost_export_cable*number_of_structures+Cost_export_cable*number_of_export_cables+Cost_array_cables
    #Cost_base_offshore =

    devex= devex_of_capex/(1-devex_of_capex)*(Cost_Nacelle_Array+cost_foundation_array+cost_connectors_total+cost_Electrical+cost_cables_total+Cost_installation_Total+Cost_base_offshore)
    capex = Cost_Nacelle_Array+cost_foundation_array+cost_connectors_total+cost_Electrical+cost_cables_total+Cost_installation_Total+Cost_base_offshore +devex
    capex=capex*number_of_structures**-0.1
    if turbines_per_structure>1:
        Pdis = turbines_per_structure ** (math.log(0.15) / math.log(2))
        capex= (1 - Pdis) * capex
        AEP=AEP*turbines_per_structure

    # OPEX
    drivetrain_capex = cost_hub*turbines_per_structure*number_of_structures+cost_brake*number_of_structures+cost_low_speed_shaft*turbines_per_structure*number_of_structures+cost_main_bearings*turbines_per_structure*number_of_structures
    electric_system_capex = Umbilical_cost_export_cable*number_of_structures + (Cost_Switchgear + Cost_Dry_Transformer)*number_of_transformers + Cost_export_cable*number_of_export_cables +Cost_array_cables
    nacelle_capex = cost_nacelle_cover*turbines_per_structure*number_of_structures
    blades_capex = cost_blades*turbines_per_structure*number_of_structures
    support_structure_capex= cost_foundation*number_of_structures + cost_mooring_system*number_of_structures
    pitch_capex= cost_pitch*turbines_per_structure*number_of_structures
    gearbox_capex = cost_gearbox*turbines_per_structure*number_of_structures
    power_converter_capex = Cost_Power_Converter*turbines_per_structure*number_of_structures
    generator_capex = cost_generator*turbines_per_structure*number_of_structures
    control_system_capex = turbines_per_structure*number_of_structures + Cost_wet_connector*number_of_structures

    capex_components = [drivetrain_capex,electric_system_capex,nacelle_capex,blades_capex,support_structure_capex,
                            pitch_capex,gearbox_capex,power_converter_capex,generator_capex,control_system_capex]
```

```python

    components, maintenance_cost= compute_opex(fuel_price,*capex_components, spare_part_cost_percentage=spare_part_cost,inflation_correction=Ratio_Europe,distance_from_shore=distance_from_shore)

    insurance_cost = insurance_cost_of_capex*capex
    if turbines_per_structure == 1:
        opex = insurance_cost + maintenance_cost
    else:
        OMratef = 0.03       # Operation & Maintenance [%/anno]
        InsuRatef = 0.01
        fEconomy =1 - 0.5 * math.log(turbines_per_structure)
        opex = capex * (OMratef + InsuRatef) * fEconomy



    AWecCmp = 0.02*AEP       # Annual TEC consumption 0.02, 2%ofAEP Annual TEC consumpt (MWh/y)
    ExtWecPr = 0.01*AEP      # Extra production, 1%ofAEP Extra Wec production (MWh/y)
    TECAvail= 100 # it is suppose to be available 100% of the time
    InstallCpct = (AEP* TECAvail/ 100) - AWecCmp + ExtWecPr #MWh/y

    capacity= 1*number_of_structures*turbines_per_structure
    Decommisioning=0.21*capex
    DecomCost_disc = Decommisioning /((1 + r) ** PrjtLftm)
    COE = (capex + PrjtLftm * opex + DecomCost_disc) / (PrjtLftm * InstallCpct)
    CRF = (1 - (1 + r) ** -PrjtLftm) / r

    LCOE = (capex + opex * CRF + DecomCost_disc) / (AEP * CRF)

    annual_revenue = AEP * priceKwh  # /year
    cashflows = [-capex] + [(annual_revenue - opex)] * PrjtLftm
    NPV = npv(r, cashflows)
    try:
        IRR = irr(cashflows)
    except:
        IRR = np.nan  # If IRR fails (e.g. all-positive cashflows)

    print(f"Number of structures: {number_of_structures}")
    print(f"Turbines per structure: {turbines_per_structure}")
```

```
    print(f"CAPACITY= {capacity:,.2f} MW")
    print(f"AEP= {AEP:,.2f} MWh")
    print(f"Insurance Costs: {insurance_cost:,.2f}")
    print(f"CapEx: {capex:,.2f}")
    print(f"OpEX: {opex:,.2f}")
    print(f"COE: {COE:,.2f} /MWh")
    print(f"LCOE: {LCOE:,.2f} /MWh")
    print(f"IRR: {IRR:,.2f} ")
    print(f"NPV: {NPV:,.2f} ")

    # --- SAFETY CHECK to prevent NaN propagation ---
    import numpy as np
    for var_name, var_value in {"LCOE": LCOE, "capex": capex, "InstallCpct": InstallCpct}.items():
        if not np.isfinite(var_value) or var_value <= 0:
            print(f" Warning: invalid {var_name} ({var_value}) in Floating economic model  skipping this configuration.")
            return (np.nan, np.nan, np.nan)
    return (LCOE, capex,opex, InstallCpct)


def economic_parameters_GBS(AEP,water_depth,
    distance_from_shore,
    number_of_blades,
    rated_current,
    rotor_diameter,
    rated_power_per_turbine,
    turbines_per_structure,
    number_of_structures,
    export_voltage,
    number_of_export_cables,
    rows_for_array,
    columns_for_array, r,PrjtLftm, priceKwh) :

    fuel_price = 515 # /tonn
    # Assumed Variables
    output_voltage_generator_array = 0.69  # kV
    power_factor = 0.95
    gearbox_ratio = 1/98
    thrust_coefficient = 0.9
    cover_to_rotor_diameter_ratio = 0.1333
```

```
    # Other Variables
    safety_factor_for_yaw = 3
    # chain_diameter =   # m
    scope_ratio = 5  # lunghezza/depth
    #steel_price_anchor_chain =   # /kg
    steel_price_foundation_platform = 0.8  # /kg
    #monopile_diameter = 0.15
    number_of_mooring_lines = 4
    devex_of_capex = 0.05  #5%
    spare_part_cost = 0.15 #15%
    repair_time_reduction = 0 #0%
    insurance_cost_of_capex = 0.01 #1%
    # secondary variables
    # Parametri noti
    rotor_radius = rotor_diameter / 2  # m
    flow_speed = rated_current  # (m/s)

    # TSR: Tip Speed Ratio
    if number_of_blades==3:
        TSR = 4.5
    else:
        TSR = 6
    # Low-speed shaft angular velocity (rad/s)
    omega_low = TSR * flow_speed / rotor_radius

    # Low-speed shaft speed (rpm)
    low_speed_rpm = (omega_low * 60) / (2 * math.pi)

    # Low-speed shaft torque (Nm)
    power_watts = rated_power_per_turbine
    torque_low = power_watts / omega_low

    # High-speed shaft speed and torque
    gearbox_ratio = 1/98
    omega_high = omega_low / gearbox_ratio
    torque_high = torque_low * gearbox_ratio

    # Apparent Power (kVA)
    apparent_power_single = power_watts / power_factor

    # Array Rated Power
```

```
    array_rated_power = rated_power_per_turbine * turbines_per_structure *
    number_of_structures
    array_apparent_power = (array_rated_power) / power_factor

    # Current per turbine (A)
    array_voltage = output_voltage_generator_array
    current_per_turbine = apparent_power_single/(math.sqrt(3)*array_voltage)

    # Array cable voltage (kV)
    Array_cable_voltage=output_voltage_generator_array*rows_for_array

    # Export cable current (A)

    export_cable_current = array_apparent_power/(math.sqrt(3)*export_voltage)/
    number_of_export_cables

    # Cover diameter
    cover_diameter = rotor_diameter * cover_to_rotor_diameter_ratio

    # Thrust force on rotor (N)
    seawater_density = 1025  # kg/m
    rotor_area = math.pi * (rotor_radius ** 2)
    thrust_force = 0.5 * seawater_density * (flow_speed ** 2) * rotor_area *
    thrust_coefficient/1000

    # Device spacing
    spacing_along_row = rotor_diameter * 2.5
    spacing_along_column = rotor_diameter * 10

    # Length each array line
    length_per_line = spacing_along_row * (columns_for_array - 1)
    total_array_length = length_per_line * rows_for_array

    # Transformers and switchgear (1 per row)
    number_of_transformers = 1


    # COST FUNCTIONS

    # INFLATION AND CURRENCY CORECTIONS
    CPI_USA_2002 =179.9
```

```
CPI_USA_2021 = 270.97
CPI_Spain_2017 = 95
CPI_England_2013 = 98.52
CPI_Ireland_2015 = 82.8
CPI_Denmark_2020 =103.4
CPI_Cile_2017 = 2.2
CPI_Europe_2020 = 106.1

CPI_USA_2024 = 313.1
CPI_Spain_2024 = 115
CPI_England_2024 = 133.4
CPI_Ireland_2024 = 100.5
CPI_Denmark_2024 = 118.7
CPI_Cile_2024 = 4.2
CPI_Europe_2024 = 130.6

Ratio_USA_2002 = CPI_USA_2024/CPI_USA_2002
Ratio_USA_2021= CPI_USA_2024/CPI_USA_2021
Ratio_Spain = CPI_Spain_2024/CPI_Spain_2017
Ratio_England = CPI_England_2024/CPI_England_2013
Ratio_Ireland = CPI_Ireland_2024/CPI_Ireland_2015
Ratio_Denmark = CPI_Denmark_2024/CPI_Denmark_2020
Ratio_Cile = CPI_Cile_2024/CPI_Cile_2017
Ratio_Europe = CPI_Europe_2024/CPI_Europe_2020

Dollar_to_Euro_conv= 0.92

# BLADE
c_blade= 40 # /m
cost_per_blade = c_blade*(rotor_diameter/2)**2.7
cost_blades = cost_per_blade*number_of_blades*Ratio_Spain

# HUB
c_hub = 1000
cost_hub = c_hub*rotor_diameter/2*Ratio_Spain

# PITCH SYSTEM
cost_pitch = 2.28*0.2106*rotor_diameter**2.6578*Ratio_USA_2002*
Dollar_to_Euro_conv

# YAW SYSTEM
```

```
    max_moment = thrust_force*cover_diameter/2*safety_factor_for_yaw #kN*m
    yaw_diameter = 0.00009*max_moment+1.53
    ratio_term = max(max_moment / yaw_diameter - 36, 0)
    mass_yaw_bearings = 0.0152 * (ratio_term ** 1.489)
    cost_yaw = 2*(mass_yaw_bearings*6.689+953)*Ratio_USA_2002*
    Dollar_to_Euro_conv

    # BRAKE SYSTEM
    brake_mass = 0.19894*rated_power_per_turbine  # kg
    cost_brake = 10*brake_mass*Ratio_USA_2002*Dollar_to_Euro_conv  #

    # GEARBOX
    gearbox_type = "Three-stage"
    gearbox_mass = 70.94*torque_low**0.759  # kg
    cost_gearbox = 10*gearbox_mass*Ratio_USA_2002*Dollar_to_Euro_conv  #

    # GENERATOR
    generator_type = "Three-stage"
    generator_mass = 6.47*(50*math.pi*torque_high)**0.9223  # kg
    cost_generator = 65*50*math.pi*torque_high*Ratio_USA_2002*
    Dollar_to_Euro_conv  #

    # LOW SPEED SHAFT
    cost_shaft_unit= 500 #/m
    cost_low_speed_shaft= cost_shaft_unit*rotor_diameter/2*Ratio_Spain

    # MAIN BEARINGS
    bearings_mass=3.5866*thrust_force-475.35
    cost_main_bearings = 2*17.6*bearings_mass*Ratio_USA_2002*Dollar_to_Euro_conv

    # ROTOR
    cost_rotor=cost_blades+cost_hub+cost_pitch+cost_low_speed_shaft  # blade +
    hub

    # PTO
    cost_PTO= cost_brake + cost_gearbox + cost_generator + cost_main_bearings

    # NACELLE COVER
    cost_fraction= 0.21 #21%
    cost_nacelle_cover=cost_fraction*(cost_rotor+cost_PTO+cost_yaw)/(1-
    cost_fraction)
```

```

    #NACELLE
    cost_nacelle = cost_rotor + cost_PTO + cost_nacelle_cover + cost_yaw


    # VARIOUS ELECTRICAL COMPONTENTS
    Cost_Power_Converter = 79*rated_power_per_turbine*Ratio_USA_2002*Dollar_to_Euro_conv
    Cost_Dry_Transformer = 11879*array_apparent_power/1000*Ratio_USA_2021*Dollar_to_Euro_conv
    Cost_Switchgear = 14018*export_voltage*Ratio_USA_2021*Dollar_to_Euro_conv
    Cost_base_offshore = 303.09*array_rated_power*Ratio_USA_2021*Dollar_to_Euro_conv
    Cost_wet_connector = 200000*Ratio_England

    # ARRAY POWER CABLE COST
    CSA_array = 28.348*math.exp(0.0044*current_per_turbine) #mm^2
    n_CSA_array = 0.6553+0.0035*CSA_array
    if output_voltage_generator_array<=10:
        n_V_array = -0.0021*output_voltage_generator_array**2+0.0607*output_voltage_generator_array+0.6076
    else:
        n_V_array=0.9819+0.0078*output_voltage_generator_array

    Unit_cost_array_cable = 200*n_CSA_array*n_V_array # /m
    Cost_array_cables = Unit_cost_array_cable * total_array_length*Ratio_Ireland #

    # EXPORT POWER CABLE COST
    CSA_export = 28.348*math.exp(0.0044*export_cable_current)
    n_CSA_export = 0.6553+0.0035*CSA_export
    if export_voltage<=10:
        n_V_export = -0.0021*export_voltage**2+0.0607*export_voltage+0.6076
    else:
        n_V_export=0.9819+0.0078*export_voltage

    Unit_cost_export_cable = 200*n_CSA_export*n_V_export
    Cost_export_cable = Unit_cost_export_cable*distance_from_shore*Ratio_Ireland

    # UMBILICAL POWER CABLE COST
    Umbilical_Unit_cost_export_cable = Unit_cost_export_cable*1.3
```

```
    Umbilical_cost_export_cable = 0*Ratio_Ireland

    # CABLE INSTALLATION
    unit_cost_laying = 100  # /m
    unit_cost_drilled_duct = 282  # /m
    fraction_drilled_duct_export = 1/3  # solo per cavo export
    array_cable_cost= unit_cost_laying*total_array_length  #
    export_cable_cost = (unit_cost_laying*(1-fraction_drilled_duct_export)+
   unit_cost_drilled_duct*fraction_drilled_duct_export)*distance_from_shore  #

    # FOUNDATION/PLATFORM COST
    foundation_mass = 1.3726*thrust_force # tons
    cost_foundation= foundation_mass*steel_price_foundation_platform*1000 #

    # ONSHORE COMPONENT PREPARATION
    time_per_component = 1  # h
    workers_for_component_prep = 6  # numero di operai
    worker_cost_per_hour = 50  # /h

    # Quantit di componenti
    total_number_of_blades = number_of_blades*turbines_per_structure*
   number_of_structures
    number_of_chain_lines = 0
    number_of_anchors = 0
    number_of_shackles = 0
    total_number_of_elements =total_number_of_blades+number_of_chain_lines+
   number_of_anchors+number_of_shackles+number_of_structures

    # Tempi totali e costi
    total_prep_time = total_number_of_elements*time_per_component  # h
    workers_prep_cost = workers_for_component_prep*worker_cost_per_hour*
   total_prep_time  #

    # INSTALLATION

    # Installation time
    Chain = 22 # h
    Anchor = 12 # h
    Connection_of_elements_for_mooring = 10 # h
    Blade_connection = 6 # h
    GBS_deployment = 1.5 # days
```

```
    Monopile_deployment = 2.5 # days
    GBS_turbine_deployment = 1 # h


    Cost_Neptune, Workers_Neptune = compute_installation_costs_Neptune(
    number_of_structures,GBS_deployment,fuel_price,distance_from_shore,
    Ratio_Europe)
    Cost_Aker, Workers_Aker = compute_installation_costs_Aker_Wayfarer(
    number_of_structures,turbines_per_structure,fuel_price,distance_from_shore,
    Ratio_Europe,rotor_diameter,GBS_turbine_deployment)

    Cost_installation_structures = Cost_Neptune+Cost_Aker+Workers_Neptune+
    Workers_Aker+workers_prep_cost

    # CAPEX
    Cost_installation_Total = Cost_installation_structures + array_cable_cost +
    export_cable_cost*number_of_export_cables
    Cost_Nacelle_Array= cost_nacelle*number_of_structures
    cost_foundation_array= cost_foundation*number_of_structures
    cost_connectors_total= Cost_wet_connector*number_of_structures
    cost_Electrical= (Cost_Power_Converter*turbines_per_structure)*
    number_of_structures+(Cost_Switchgear+Cost_Dry_Transformer)*
    number_of_transformers
    cost_cables_total= Umbilical_cost_export_cable*number_of_structures+
    Cost_export_cable*number_of_export_cables+Cost_array_cables
    # Cost_base_offshore

    devex= devex_of_capex/(1-devex_of_capex)*(Cost_Nacelle_Array+
    cost_foundation_array+cost_connectors_total+cost_Electrical+
    cost_cables_total+Cost_installation_Total+Cost_base_offshore)


    capex = Cost_Nacelle_Array+cost_foundation_array+cost_connectors_total+
    cost_Electrical+cost_cables_total+Cost_installation_Total+Cost_base_offshore
     +devex

    if turbines_per_structure>1:
        Pdis = turbines_per_structure ** (np.log(0.15) / np.log(2))
        capex= (1 - Pdis) * capex
        AEP=AEP*turbines_per_structure

```

```python
    # OPEX
    drivetrain_capex = cost_hub*turbines_per_structure*number_of_structures+cost_brake*number_of_structures+cost_low_speed_shaft*turbines_per_structure*number_of_structures+cost_main_bearings*turbines_per_structure*number_of_structures
    electric_system_capex = Umbilical_cost_export_cable*number_of_structures + (Cost_Switchgear + Cost_Dry_Transformer)*number_of_transformers + Cost_export_cable*number_of_export_cables +Cost_array_cables
    nacelle_capex = cost_nacelle_cover*turbines_per_structure*number_of_structures
    blades_capex = cost_blades*turbines_per_structure*number_of_structures
    support_structure_capex= cost_foundation*number_of_structures # + cost_mooring*number_of_structures
    pitch_capex= cost_pitch*turbines_per_structure*number_of_structures
    gearbox_capex = cost_gearbox*turbines_per_structure*number_of_structures
    power_converter_capex = Cost_Power_Converter*turbines_per_structure*number_of_structures
    generator_capex = cost_generator*turbines_per_structure*number_of_structures
    control_system_capex = cost_yaw*turbines_per_structure*number_of_structures + Cost_wet_connector*number_of_structures

    capex_components = [drivetrain_capex,electric_system_capex,nacelle_capex,blades_capex,support_structure_capex,
                        pitch_capex,gearbox_capex,power_converter_capex,generator_capex,control_system_capex]

    components, maintenance_cost= compute_opex(fuel_price,*capex_components,spare_part_cost_percentage=spare_part_cost,inflation_correction=Ratio_Europe,distance_from_shore=distance_from_shore)

    insurance_cost = insurance_cost_of_capex*capex


    if turbines_per_structure==1:
        opex = insurance_cost + maintenance_cost
    else:
        fEconomy = 1 - 0.5 * np.log(turbines_per_structure)
        OMratef = 0.03
        InsuRatef = 0.01
        opex = (capex*(OMratef + InsuRatef)) * fEconomy

```

```python

    AWecCmp = 0.02*AEP       # Annual TEC consumption 0.02, 2%ofAEP Annual TEC consumpt (MWh/y)
    ExtWecPr = 0.01*AEP      # Extra production, 1%ofAEP Extra Wec production (MWh/y)
    TECAvail= 100 # it is suppose to be available 100% of the time
    InstallCpct = (AEP*turbines_per_structure* TECAvail/ 100) - AWecCmp + ExtWecPr #MWh/y

    Decommisioning=0.21*capex
    DecomCost_disc = Decommisioning /((1 + r) ** PrjtLftm)
    COE = (capex + PrjtLftm * opex + DecomCost_disc) / (PrjtLftm * InstallCpct)
    CRF = (1 - (1 + r) ** -PrjtLftm) / r

    LCOE = (capex + opex * CRF + DecomCost_disc) / (AEP * CRF)

    annual_revenue = AEP * priceKwh  # /year
    cashflows = [-capex] + [(annual_revenue - opex)] * PrjtLftm
    NPV = npv(r, cashflows)
    try:
        IRR = irr(cashflows)
    except Exception:
        import numpy as np
        IRR = np.nan  # If IRR fails (e.g. all-positive cashflows)
    print(f"AEP= {AEP:,.2f} MWh")
    print(f"Insurance Costs: {insurance_cost:,.2f}")
    print(f"CapEx: {capex:,.2f}")
    print(f"OpEX: {opex:,.2f}")
    print(f"COE: {COE:,.2f} /MWh")
    print(f"LCOE: {LCOE:,.2f} /MWh")
    print(f"IRR: {IRR:,.2f} ")
    print(f"NPV: {NPV:,.2f} ")
    # --- SAFETY CHECK to prevent NaN propagation ---
    import numpy as np
    for var_name, var_value in {"LCOE": LCOE, "capex": capex, "InstallCpct": InstallCpct}.items():
        if not np.isfinite(var_value) or var_value <= 0:
            print(f" Warning: invalid {var_name} ({var_value}) in GBS economic model  skipping this configuration.")
            return (np.nan, np.nan, np.nan)
    return (LCOE, capex,opex, InstallCpct)
```

```python



def economic_parameters_Monopile(AEP,water_depth,
    distance_from_shore,
    number_of_blades,
    rated_current,
    rotor_diameter,
    rated_power_per_turbine,
    turbines_per_structure,
    number_of_structures,
    export_voltage,
    number_of_export_cables,
    rows_for_array,
    columns_for_array, r,PrjtLftm, priceKwh) :

    fuel_price = 515 # /tonn


    # Assumed Variables
    output_voltage_generator_array = 0.69  # kV
    power_factor = 0.95
    gearbox_ratio = 1/98
    thrust_coefficient = 0.9
    cover_to_rotor_diameter_ratio = 0.1333

    # Other Variables
    safety_factor_for_yaw = 3
    # chain_diameter =   # m
    scope_ratio = 5  # length/depth
    #steel_price_anchor_chain =   # /kg
    steel_price_foundation_platform = 1.2  # /kg
    monopile_diameter = 3.5  # m
    rotor_height = 30 #m
    devex_of_capex = 0.05  #5%
    spare_part_cost = 0.15 #15%
    repair_time_reduction = 0.3 #30%
    insurance_cost_of_capex = 0.01 #1%

    # secondary variables
    # Parametri noti
```

```
    rotor_radius = rotor_diameter / 2  # m
    flow_speed = rated_current  # (m/s)

    # TSR: Tip Speed Ratio
    if number_of_blades==3:
        TSR = 4.5
    else:
        TSR = 6

    # Low-speed shaft angular velocity (rad/s)
    omega_low = TSR * flow_speed / rotor_radius

    # Low-speed shaft speed (rpm)
    low_speed_rpm = (omega_low * 60) / (2 * math.pi)

    # Low-speed shaft torque (Nm)
    power_watts = rated_power_per_turbine
    torque_low = power_watts / omega_low

    # High-speed shaft speed and torque
    gearbox_ratio = 1/98
    omega_high = omega_low / gearbox_ratio
    torque_high = torque_low * gearbox_ratio

    # Apparent Power (kVA)
    apparent_power_single = power_watts / power_factor

    # Array Rated Power
    array_rated_power = rated_power_per_turbine * turbines_per_structure *
    number_of_structures
    array_apparent_power = (array_rated_power) / power_factor

    # Current per turbine (A)
    array_voltage = output_voltage_generator_array
    current_per_turbine = apparent_power_single/(math.sqrt(3)*array_voltage)

    # Array cable voltage (kV)
    Array_cable_voltage=output_voltage_generator_array*rows_for_array

    # Export cable current (A)

```

```
    export_cable_current = array_apparent_power/(math.sqrt(3)*export_voltage)/
    number_of_export_cables

    # Cover diameter
    cover_diameter = rotor_diameter * cover_to_rotor_diameter_ratio

    # Thrust force on rotor (N)
    seawater_density = 1025  # kg/m
    rotor_area = math.pi     * (rotor_radius ** 2)
    thrust_force = 0.5 * seawater_density * (flow_speed ** 2) * rotor_area *
    thrust_coefficient/1000

    # Device spacing
    spacing_along_row = rotor_diameter * 2.5
    spacing_along_column = rotor_diameter * 10

    # Length each array line
    length_per_line = spacing_along_row * (columns_for_array - 1)
    total_array_length = length_per_line * rows_for_array

    # Transformers and switchgear (1 per row)
    number_of_transformers = 1


    # COST FUNCTIONS

    # INFLATION AND CURRENCY CORECTIONS
    CPI_USA_2002 =179.9
    CPI_USA_2021 = 270.97
    CPI_Spain_2017 = 95
    CPI_England_2013 = 98.52
    CPI_Ireland_2015 = 82.8
    CPI_Denmark_2020 =103.4
    CPI_Cile_2017 = 2.2
    CPI_Europe_2020 = 106.1

    CPI_USA_2024 = 313.1
    CPI_Spain_2024 = 115
    CPI_England_2024 = 133.4
    CPI_Ireland_2024 = 100.5
    CPI_Denmark_2024 = 118.7
```

```
    CPI_Cile_2024 = 4.2
    CPI_Europe_2024 = 130.6

    Ratio_USA_2002 = CPI_USA_2024/CPI_USA_2002
    Ratio_USA_2021= CPI_USA_2024/CPI_USA_2021
    Ratio_Spain = CPI_Spain_2024/CPI_Spain_2017
    Ratio_England = CPI_England_2024/CPI_England_2013
    Ratio_Ireland = CPI_Ireland_2024/CPI_Ireland_2015
    Ratio_Denmark = CPI_Denmark_2024/CPI_Denmark_2020
    Ratio_Cile = CPI_Cile_2024/CPI_Cile_2017
    Ratio_Europe = CPI_Europe_2024/CPI_Europe_2020

    Dollar_to_Euro_conv= 0.92

    # BLADE
    c_blade= 40 # /m
    cost_per_blade = c_blade*(rotor_diameter/2)**2.7
    cost_blades = cost_per_blade*number_of_blades*Ratio_Spain

    # HUB
    c_hub = 1000
    cost_hub = c_hub*rotor_diameter/2*Ratio_Spain

    # PITCH SYSTEM
    cost_pitch = 2.28*0.2106*rotor_diameter**2.6578*Ratio_USA_2002*
    Dollar_to_Euro_conv

    # YAW SYSTEM
    max_moment = thrust_force*cover_diameter/2*safety_factor_for_yaw #kN*m
    yaw_diameter = 0.00009*max_moment+1.53
    mass_yaw_bearings = 0.0152*(max_moment/yaw_diameter-36)**1.489
    cost_yaw = 0
    # BRAKE SYSTEM
    brake_mass = 0.19894*rated_power_per_turbine  # kg
    cost_brake = 10*brake_mass*Ratio_USA_2002*Dollar_to_Euro_conv  #

    # GEARBOX
    gearbox_type = "Three-stage"
    gearbox_mass = 70.94*torque_low**0.759  # kg
    cost_gearbox = 10*gearbox_mass*Ratio_USA_2002*Dollar_to_Euro_conv  #

```

```
    # GENERATOR
    generator_type = "Three-stage"
    generator_mass = 6.47*(50*math.pi*torque_high)**0.9223  # kg
    cost_generator = 65*50*math.pi*torque_high*Ratio_USA_2002*Dollar_to_Euro_conv  #

    # LOW SPEED SHAFT
    cost_shaft_unit= 500 #/m
    cost_low_speed_shaft= cost_shaft_unit*rotor_diameter/2*Ratio_Spain

    # MAIN BEARINGS
    bearings_mass=3.5866*thrust_force-475.35
    cost_main_bearings = 2*17.6*bearings_mass*Ratio_USA_2002*Dollar_to_Euro_conv

    # ROTOR
    cost_rotor=cost_blades+cost_hub+cost_pitch+cost_low_speed_shaft  # blade + hub

    # PTO
    cost_PTO= cost_brake + cost_gearbox + cost_generator + cost_main_bearings

    # NACELLE COVER
    cost_fraction= 0.21 #21%
    cost_nacelle_cover=cost_fraction*(cost_rotor+cost_PTO+cost_yaw)/(1-cost_fraction)

    #NACELLE
    cost_nacelle = cost_rotor + cost_PTO + cost_nacelle_cover + cost_yaw

    # VARIOUS ELECTRICAL COMPONTENTS
    Cost_Power_Converter = 79*rated_power_per_turbine*Ratio_USA_2002*Dollar_to_Euro_conv
    Cost_Dry_Transformer = 11879*array_apparent_power/1000*Ratio_USA_2021*Dollar_to_Euro_conv
    Cost_Switchgear = 14018*export_voltage*Ratio_USA_2021*Dollar_to_Euro_conv
    Cost_base_offshore = 303.09*array_rated_power*Ratio_USA_2021*Dollar_to_Euro_conv
    Cost_wet_connector = 200000*Ratio_England

    # ARRAY POWER CABLE COST
    CSA_array = 28.348*math.exp(0.0044*current_per_turbine) #mm^2
```

```
    n_CSA_array = 0.6553+0.0035*CSA_array
    if output_voltage_generator_array<=10:
        n_V_array = -0.0021*output_voltage_generator_array**2+0.0607*
    output_voltage_generator_array+0.6076
    else:
        n_V_array=0.9819+0.0078*output_voltage_generator_array

    Unit_cost_array_cable = 200*n_CSA_array*n_V_array # /m
    Cost_array_cables =0*Ratio_Ireland #

    # EXPORT POWER CABLE COST
    CSA_export = 28.348*math.exp(0.0044*export_cable_current)
    n_CSA_export = 0.6553+0.0035*CSA_export
    if export_voltage<=10:
        n_V_export = -0.0021*export_voltage**2+0.0607*export_voltage+0.6076
    else:
        n_V_export=0.9819+0.0078*export_voltage

    Unit_cost_export_cable = 200*n_CSA_export*n_V_export
    Cost_export_cable = Unit_cost_export_cable*distance_from_shore*Ratio_Ireland

    # UMBILICAL POWER CABLE COST
    Umbilical_Unit_cost_export_cable = Unit_cost_export_cable*1.3
    Umbilical_cost_export_cable = Umbilical_Unit_cost_export_cable*water_depth*
    Ratio_Ireland

    # CABLE INSTALLATION
    unit_cost_laying = 100  # /m
    unit_cost_drilled_duct = 282  # /m
    fraction_drilled_duct_export = 1/3  # solo per cavo export
    array_cable_cost= unit_cost_laying*total_array_length  #
    export_cable_cost = (unit_cost_laying*(1-fraction_drilled_duct_export)+
    unit_cost_drilled_duct*fraction_drilled_duct_export)*distance_from_shore  #


    # FOUNDATION/PLATFORM COST
    monopile_height=0.5*rotor_height+water_depth+9 #m
    total_thrust=turbines_per_structure*thrust_force  #kN
    moment_applied = total_thrust*rotor_height #kNm
    SF_foundation= 2.6
    sigma_yield= 248 #MPa
```

```
    sigma_amm = sigma_yield/SF_foundation
    D_in = monopile_diameter*(1-32*moment_applied/(math.pi*1000*(
    monopile_diameter**3)*sigma_amm))**(1/4)
    thickness = (monopile_diameter-D_in)/2*1000
    tube_mass= (monopile_diameter**2-D_in**2)*math.pi/4*monopile_height*7850
    crossarm_mass = 32.09*total_thrust
    monopile_mass = tube_mass+crossarm_mass
    monopile_cost = monopile_mass*steel_price_foundation_platform


    # ONSHORE COMPONENT PREPARATION
    time_per_component = 1  # h
    workers_for_component_prep = 6  # numero di operai
    worker_cost_per_hour = 50  # /h

    # Quantit di componenti
    total_number_of_blades = number_of_blades*turbines_per_structure*
    number_of_structures
    number_of_chain_lines = 0
    number_of_anchors = 0
    number_of_shackles = 0
    total_number_of_elements =total_number_of_blades+number_of_chain_lines+
    number_of_anchors+number_of_shackles+number_of_structures

    # Tempi totali e costi
    total_prep_time = total_number_of_elements*time_per_component  # h
    workers_prep_cost = workers_for_component_prep*worker_cost_per_hour*
    total_prep_time  #

    # INSTALLATION

    # Installation time
    Chain = 22 # h
    Anchor = 12 # h
    Connection_of_elements_for_mooring = 10 # h
    Blade_connection = 6 # h
    GBS_deployment = 1.5 # days
    Monopile_deployment = 2.5 # days
    turbine_deployment = 1 # h

```

```python
    Cost_Rambiz,Workers_Rambiz = compute_installation_costs_Rambiz(number_of_structures,Monopile_deployment,fuel_price,distance_from_shore,Ratio_Europe)
    Cost_Aker, Workers_Aker = compute_installation_costs_Aker_Wayfarer(number_of_structures,turbines_per_structure,fuel_price,distance_from_shore,Ratio_Europe,rotor_diameter,turbine_deployment)

    Cost_installation_structures = Cost_Rambiz+Cost_Aker+Workers_Rambiz+Workers_Aker+workers_prep_cost


    # CAPEX
    Cost_installation_Total = Cost_installation_structures + export_cable_cost*number_of_export_cables
    Cost_Nacelle_Array= cost_nacelle*number_of_structures
    cost_foundation_array= monopile_cost*number_of_structures
    cost_connectors_total= Cost_wet_connector*number_of_structures
    cost_Electrical= (Cost_Power_Converter*turbines_per_structure)*number_of_structures+(Cost_Switchgear+Cost_Dry_Transformer)*number_of_transformers
    cost_cables_total= Umbilical_cost_export_cable*number_of_structures+Cost_export_cable*number_of_export_cables+Cost_array_cables
    # Cost_base_offshore

    devex= devex_of_capex/(1-devex_of_capex)*(Cost_Nacelle_Array+cost_foundation_array+cost_connectors_total+cost_Electrical+cost_cables_total+Cost_installation_Total+Cost_base_offshore)

    capex = Cost_Nacelle_Array+cost_foundation_array+cost_connectors_total+cost_Electrical+cost_cables_total+Cost_installation_Total+Cost_base_offshore +devex
    if turbines_per_structure>1:
        Pdis = turbines_per_structure ** (math.log(0.15) / math.log(2))
        capex= (1 - Pdis) * capex
        AEP=AEP*turbines_per_structure

    # OPEX
    drivetrain_capex = cost_hub*turbines_per_structure*number_of_structures+cost_brake*number_of_structures+cost_low_speed_shaft*turbines_per_structure*number_of_structures+cost_main_bearings*turbines_per_structure*number_of_structures
```

```
    electric_system_capex = Umbilical_cost_export_cable*number_of_structures + (Cost_Switchgear + Cost_Dry_Transformer)*number_of_transformers + Cost_export_cable*number_of_export_cables +Cost_array_cables
    nacelle_capex = cost_nacelle_cover*turbines_per_structure*number_of_structures
    blades_capex = cost_blades*turbines_per_structure*number_of_structures
    support_structure_capex= monopile_cost*number_of_structures # + cost_mooring*number_of_structures
    pitch_capex= cost_pitch*turbines_per_structure*number_of_structures
    gearbox_capex = cost_gearbox*turbines_per_structure*number_of_structures
    power_converter_capex = Cost_Power_Converter*turbines_per_structure*number_of_structures
    generator_capex = cost_generator*turbines_per_structure*number_of_structures
    control_system_capex = cost_yaw*turbines_per_structure*number_of_structures + Cost_wet_connector*number_of_structures
    capex_components = [drivetrain_capex,electric_system_capex,nacelle_capex,blades_capex,support_structure_capex,
                        pitch_capex,gearbox_capex,power_converter_capex,generator_capex,control_system_capex]

    components, maintenance_cost= compute_opex(fuel_price,*capex_components,spare_part_cost_percentage=spare_part_cost,inflation_correction=Ratio_Europe,distance_from_shore=distance_from_shore)

    insurance_cost = insurance_cost_of_capex*capex

    if turbines_per_structure == 1:
        opex = insurance_cost + maintenance_cost
    else:
        OMratef = 0.03       # Operation & Maintenance [%/anno]
        InsuRatef = 0.01
        fEconomy = turbines_per_structure ** -0.1
        opex = capex * (OMratef + InsuRatef) * fEconomy

    AWecCmp = 0.02*AEP       # Annual WEC consumption 0.02, 2%ofAEP Annual Wec consumpt (MWh/y)
    ExtWecPr = 0.01*AEP      # Extra production, 1%ofAEP Extra Wec production (MWh/y)
    TECAvail= 100 # it is suppose to be available 100% of the time
    InstallCpct = (AEP* TECAvail/ 100) - AWecCmp + ExtWecPr #MWh/y

```

```
    Decommisioning=0.21*capex
    DecomCost_disc = Decommisioning /((1 + r) ** PrjtLftm)
    COE = (capex + PrjtLftm * opex + DecomCost_disc) / (PrjtLftm * InstallCpct)
    CRF = (1 - (1 + r) ** -PrjtLftm) / r
    LCOE = (capex + opex * CRF + DecomCost_disc) / (AEP * CRF)

    annual_revenue = AEP * priceKwh  # /year
    cashflows = [-capex] + [(annual_revenue - opex)] * PrjtLftm
    NPV = npv(r, cashflows)
    try:
        IRR = irr(cashflows)
    except:
        IRR = np.nan  # If IRR fails (e.g. all-positive cashflows)
    print(f"AEP= {AEP:,.2f} MWh")
    print(f"Insurance Costs: {insurance_cost:,.2f}")
    print(f"CapEx: {capex:,.2f}")
    print(f"OpEX: {opex:,.2f}")
    print(f"COE: {COE:,.2f} /MWh")
    print(f"LCOE: {LCOE:,.2f} /MWh")
    print(f"IRR: {IRR:,.2f} ")
    print(f"NPV: {NPV:,.2f} ")

    # --- SAFETY CHECK to prevent NaN propagation ---
    import numpy as np
    for var_name, var_value in {"LCOE": LCOE, "capex": capex, "InstallCpct": InstallCpct}.items():
        if not np.isfinite(var_value) or var_value <= 0:
            print(f" Warning: invalid {var_name} ({var_value}) in Monopile economic model  skipping this configuration.")
            return (np.nan, np.nan, np.nan)
    return (LCOE, capex,opex, InstallCpct)

#hydrogen production costs

def costs_hydrogen_prod (H2_total,PEM_nom,
                         COMP_nom,
                         STORAGE_nom,capex, opex,PrjtLftm,r):

    C_PEM = 1000 # /kW
    C_compr= 500 #/kW
    C_storage = 80 #/kg
```

```
    cost_PEM= C_PEM*PEM_nom
    cost_compr= C_compr*COMP_nom
    cost_storage=C_storage*STORAGE_nom
    capex_hydrogen = cost_PEM+cost_compr+cost_storage
    opex_hydrogen= 0.02*capex_hydrogen # 1-3% of capex
    Decommisioning=0.21*(capex+capex_hydrogen)
    DecomCost_disc = Decommisioning /((1 + r) ** PrjtLftm)
    CRF = (1 - (1 + r) ** -PrjtLftm) / r
    LCOH=(capex_hydrogen+capex+ (opex_hydrogen+opex)*CRF + DecomCost_disc) / (
    H2_total*CRF)
    print(f"H2_produced= {H2_total:,.2f} kg")
    print(f"CapEx hydrogen: {capex_hydrogen:,.2f}")
    print(f"OpEX hydrogen: {opex_hydrogen:,.2f}")
    print(f"LCOH: {LCOH:,.2f} /kg H2")
    print(f"PEM: {cost_PEM:,.2f} ")
    print(f"COMPRESSOR: {cost_compr:,.2f} ")
    print(f"STORAGE: {cost_storage:,.2f} ")





    return cost_PEM,cost_compr,cost_storage,capex_hydrogen,opex_hydrogen,LCOH


def costs_computations(TECeval,AEP,h,distShr,nblades,rated_current,
    rotor_diameter,rated_power_per_turbine,turbines_per_structure,
    number_of_structures,export_voltage,number_of_export_cables,rows_for_array,
    columns_for_array, r,PrjtLftm, priceKwh,H2_total,PEM_nom, COMP_nom,
    STORAGE_nom):
    typeTEC = TECeval.split('_')[0]
    LCOE_farm_vals=[]
    YrlyPrd_farm_vals=[]
    CapExRed_vals=[]
    opex_farm_vals=[]
    LCOH_farm_vals=[]
    electrolyzer_eff = 0.7
    LHV_H2 = 33.33        # kWh/kg
    comp_energy_per_kg = 2.5  # kWh/kg
    electrolyzer_size_factor = 1.5
    for i in range(1,16):
```

```
        number_of_structures=i
        AEP_total=AEP*i
        P_avg = (AEP_total * 1000) / 8760
        if typeTEC=="Floating":
            [LCOE,capex,opex,InstallCpct]=economic_parameters_Floating (
    AEP_total,h,
                distShr,
                nblades,
                rated_current,
                rotor_diameter,
                rated_power_per_turbine,
                turbines_per_structure,
                number_of_structures,
                export_voltage,
                number_of_export_cables,
                rows_for_array,
                columns_for_array, r,PrjtLftm, priceKwh)
        elif typeTEC=="GBS":
            [LCOE,capex,opex,InstallCpct]=economic_parameters_GBS(AEP_total,h,
            distShr,
            nblades,
            rated_current,
            rotor_diameter,
            rated_power_per_turbine,
            turbines_per_structure,
            number_of_structures,
            export_voltage,
            number_of_export_cables,
            rows_for_array,
            columns_for_array, r,PrjtLftm, priceKwh)
        elif typeTEC=="Monopile":
            [LCOE,capex,opex,InstallCpct]=economic_parameters_Monopile(AEP_total
    ,h,
            distShr,
            nblades,
            rated_current,
            rotor_diameter,
            rated_power_per_turbine,
            turbines_per_structure,
            number_of_structures,
            export_voltage,
```

```
            number_of_export_cables,
            rows_for_array,
            columns_for_array, r,PrjtLftm, priceKwh)

        _, _, _, capex_h2, opex_h2, LCOH = costs_hydrogen_prod(
            H2_total, PEM_nom, COMP_nom, STORAGE_nom, capex, opex, PrjtLftm, r
        )
        LCOE_farm_vals.append(LCOE)
        YrlyPrd_farm_vals.append(InstallCpct)
        CapExRed_vals.append(capex)
        opex_farm_vals.append(opex)
        LCOH_farm_vals.append(LCOH)
    MatC= {'LCOEfarm_all' :np.array(LCOE_farm_vals),'YrlyPrd_farm_all':np.array(YrlyPrd_farm_vals),'CapExRed_all':np.array(CapExRed_vals),'opex_farm_all':np.array(opex_farm_vals),'LCOH_farm_all':np.array(LCOH_farm_vals)}

    return MatC


def pltinstlld_cpcty(Costs,svpath):
    """
    Plot 3D surface: CapEx vs Installed Capacity vs LCOE for wave energy farm.
    Parameters
    ----------
    Costs : dict
        Dictionary containing the cost results from the main calculation function.
        Must include keys: 'CapExRed_all', 'YrlyPrd_farm_all', 'LCOEfarm_all'.
    """
    # Extract arrays
    x = np.array(Costs['CapExRed_all'])      # CapEx reduced ()
    y = np.array(Costs['YrlyPrd_farm_all'])  # Installed capacity (MWh/year)
    z = np.array(Costs['LCOEfarm_all'])      #LCOE
    v = np.array(Costs['opex_farm_all'])     # Opex
    w = np.array(Costs['LCOH_farm_all'])     # LCOH (/kg)
    xi = np.linspace(np.min(x), np.max(x), 100)
    yi = np.linspace(np.min(y), np.max(y), 100)
    XI, YI = np.meshgrid(xi, yi)
    print("Capex values:", np.unique(x))
    print("YrlyPrd_farm_all values:", np.unique(y))
    print("LCOEfarm_all values:", np.array(z))
```

```
    print("opex_farm_all values:", np.unique(v))
    print("LCOH_farm_all values:", np.array(w))
    z=np.sort(z)[::-1]

    # Remove invalid or NaN data before interpolation
    mask = np.isfinite(x) & np.isfinite(y) & np.isfinite(z)
    x, y, z = x[mask], y[mask], z[mask]
    if len(x) == 0:
        print(" Nessun punto valido per linterpolazione (tutti NaN o inf). Salto la creazione del grafico.")
        return
    ZI = griddata((x, y), z, (XI, YI), method='nearest')
    fig = plt.figure(figsize=(10, 7))
    ax = fig.add_subplot(111, projection='3d')
    surf = ax.plot_surface(XI, YI, ZI, cmap='viridis', edgecolor='none', alpha=0.95)
    ax.set_xlabel('CapEx []')
    ax.set_ylabel('Installed Capacity [MWh/year]')
    ax.view_init(elev=10, azim=30)
    ax.set_zlabel('LCOE [/MWh]')
    ax.set_title('LCOE vs Installed Capacity and CapEx (Interpolated)')
    fig.colorbar(surf, ax=ax, shrink=0.5, aspect=10, label='LCOE [/MWh]')
    plt.tight_layout()

    os.makedirs(svpath, exist_ok=True)
    plt.savefig(os.path.join(svpath, '08_LCOE_CapEx_Cpcty_Interpolated.png'), dpi=300)
    plt.close()
```

A.3.1 mooring_prelay_ODYSSEY.py

```
    import math

def compute_mooring_prelay_costs_Odyssey(chain_t,anchor_t,mooring_lines,fuel_price,distance_from_shore,inflation_correction):


    loa = 26               # m
    installed_power_vessel = 1790    # kW
    deck_area = 120                 # m
```

```
    vessel_speed =  18.5           # km/h
    number_of_workers = 3           # persone a bordo
    worker_cost_per_hour = 50       # /ora

    vessel_charter_rate = 63.23*loa+1812.4   # /giorno
    installation_time=chain_t+anchor_t
    installation_time_total= installation_time*mooring_lines
    days_for_installation= installation_time_total/24
    Cfuel = installed_power_vessel*fuel_price*0.8*210*24/(10**6)
    Cvessel=vessel_charter_rate+Cfuel
    installation_cost_mooring_prelay=Cvessel*(days_for_installation+2*
    distance_from_shore/vessel_speed/1000/24)
    installation_cost_mooring_prelay_adjusted = installation_cost_mooring_prelay
    *inflation_correction  #  installazione aggiustata
    workers_total_cost_Odyssey = number_of_workers*worker_cost_per_hour*24*(
    days_for_installation+2*distance_from_shore/vessel_speed/1000/24) #  totale
    lavoratori

    return installation_cost_mooring_prelay_adjusted, workers_total_cost_Odyssey


\end{document}
```

A.3.2 towing_THOR.py

```
    import math

def compute_towing_costs_Thor(fuel_price,distance_from_shore,
    inflation_correction):


    bollard_pull = 78            # tons
    installed_power_vessel = 4700   # kW
    # deck_area =                # m
    vessel_speed_towing = 5           # km/h
    vessel_speed_normal = 20       # km/h
    number_of_workers = 3           # persone a bordo
    worker_cost_per_hour = 50       # /ora

    vessel_charter_rate = 508.57*bollard_pull-32186  # /giorno
```

```
    vessel_number_tow = 2
    # installation_time =
    # days_for_installation= installation_time/24
    Cfuel = installed_power_vessel*fuel_price*0.8*210*24/(10**6)
    Cvessel=vessel_charter_rate+Cfuel
    cost_towing=vessel_number_tow*Cvessel*(distance_from_shore/vessel_speed_towing/1000/24+distance_from_shore/vessel_speed_normal/1000/24)
    cost_towing_adjusted =cost_towing*inflation_correction  #  installazione aggiustata
    workers_total_cost_Thor = number_of_workers*worker_cost_per_hour*24*(distance_from_shore/vessel_speed_towing/1000/24+distance_from_shore/vessel_speed_normal/1000/24) #  totale lavoratori

    return cost_towing_adjusted, workers_total_cost_Thor
```

A.3.3 mooring_connection_ODYSSEY.py

```
import math

def compute_mooring_connection_costs_Odyssey(mooring_connection_t,mooring_number,fuel_price,distance_from_shore,inflation_correction):


    loa = 26             # m
    installed_power_vessel = 1790   # kW
    deck_area = 120                 # m
    vessel_speed =  18.5            # km/h
    number_of_workers = 3            # persone a bordo
    worker_cost_per_hour = 50        # /ora

    vessel_charter_rate = 63.23*loa+1812.4   # /giorno
    installation_time = mooring_connection_t*mooring_number
    days_for_installation= installation_time/24
    Cfuel = installed_power_vessel*fuel_price*0.8*210*24/(10**6)
    Cvessel=vessel_charter_rate+Cfuel
    installation_cost_mooring_connection=Cvessel*(days_for_installation+2*distance_from_shore/vessel_speed/1000/24)
    installation_cost_mooring_connection_adjusted = installation_cost_mooring_connection*inflation_correction  #  installazione aggiustata
```

```
    workers_total_cost_Odyssey = number_of_workers*worker_cost_per_hour*(
    days_for_installation*24+2*distance_from_shore/vessel_speed/1000) #  totale
    lavoratori

    return installation_cost_mooring_connection_adjusted,
    workers_total_cost_Odyssey
```

A.3.4 support_mooring_connection_USKMOOR.py

```
    import math

def compute_support_mooring_connection_costs_Uskmoor(mooring_connection_t,
    mooring_number,fuel_price,distance_from_shore,inflation_correction):


    loa = 16            # m
    installed_power_vessel = 294   # kW
    # deck_area =               # m
    vessel_speed =  16.5           # km/h
    number_of_workers = 2           # persone a bordo
    worker_cost_per_hour = 50       # /ora

    vessel_charter_rate = 63.23*loa+1812.4   # /giorno
    installation_time = mooring_connection_t*mooring_number
    days_for_installation= installation_time/24
    Cfuel = installed_power_vessel*fuel_price*0.8*210*24/(10**6)
    Cvessel=vessel_charter_rate+Cfuel
    support_mooring_connection=Cvessel*(days_for_installation+2*
    distance_from_shore/vessel_speed/1000/24)
    support_mooring_connection_adjusted = support_mooring_connection*
    inflation_correction  #  installazione aggiustata
    workers_total_cost_Uskmoor = number_of_workers*worker_cost_per_hour*(
    days_for_installation*24+2*distance_from_shore/vessel_speed/1000) #  totale
    lavoratori

    return support_mooring_connection_adjusted, workers_total_cost_Uskmoor
```

A.3.5 blades_connection_ODYSSEY.py

```python
    import math

def compute_blades_connection_costs_Odyssey(blades_connection_t,blades_number,turbines_per_structure,fuel_price,distance_from_shore,inflation_correction):


    loa = 26            # m
    installed_power_vessel = 1790   # kW
    deck_area = 120                # m
    vessel_speed =  18.5           # km/h
    number_of_workers = 3           # persone a bordo
    worker_cost_per_hour = 50       # /ora

    vessel_charter_rate = 63.23*loa+1812.4   # /giorno
    elements_per_trip = 1
    number_of_trips = elements_per_trip*blades_number*turbines_per_structure
    installation_time = blades_connection_t
    days_for_installation= installation_time/24
    Cfuel = installed_power_vessel*fuel_price*0.8*210*24/(10**6)
    Cvessel=vessel_charter_rate+Cfuel
    installation_cost_blades_connection=Cvessel*number_of_trips*(days_for_installation+2*distance_from_shore/vessel_speed/1000/24)
    installation_cost_blades_connection_adjusted = installation_cost_blades_connection*inflation_correction  #  installazione aggiustata
    workers_total_cost_Odyssey = number_of_workers*worker_cost_per_hour*number_of_trips*(days_for_installation*24+2*distance_from_shore/vessel_speed/1000) #  totale lavoratori

    return installation_cost_blades_connection_adjusted, workers_total_cost_Odyssey
```

A.3.6 installation_costs_NEPTUNE.py

```python
    import math

def compute_installation_costs_Neptune(number_of_structures,GBS_deployment,fuel_price,distance_from_shore,inflation_correction):

```

```

    crane_capacity = 600            # tonnellate (o altra unit specifica)
    installed_power_vessel = 8970   # kW
    deck_area = 2000                # m
    vessel_speed = 0.027            # m/s
    number_of_workers = 3           # persone a bordo
    worker_cost_per_hour = 50       # /ora

    vessel_charter_rate = 64.71*crane_capacity+21448.41   # /giorno
    substructure_area = 25*20                       # m
    elements_per_trip = deck_area/substructure_area           # n elementi per viaggio
    number_of_trips = math.ceil(number_of_structures/elements_per_trip) # totale viaggi necessari
    installation_time_per_substructure = GBS_deployment   # giorni per substructure
    total_installation_time = installation_time_per_substructure*number_of_structures    # t inst (giorni totali di installazione)

    cost_fuel = installed_power_vessel*fuel_price*0.8*210*24/10**6     # carburante
    cost_vessel = vessel_charter_rate+cost_fuel                        # noleggio nave
    installation_cost_substructure = cost_vessel*(total_installation_time+number_of_trips*2*distance_from_shore/vessel_speed/1000/24)   # installazione base
    installation_cost_substructure_adjusted = installation_cost_substructure*inflation_correction  #  installazione aggiustata
    workers_total_cost_Neptune = number_of_workers*worker_cost_per_hour*24*(total_installation_time+number_of_trips*2*distance_from_shore/vessel_speed/1000/24) #  totale lavoratori

    return installation_cost_substructure_adjusted, workers_total_cost_Neptune
```

A.3.7 installation_costs_AKER_WAYFARER.py

```
    import math

```

```
def compute_installation_costs_Aker_Wayfarer(number_of_structures, turbines_per_structure,fuel_price,distance_from_shore,inflation_correction, rotor_diameter,turbine_deployment):


    crane_capacity = 400             # tonnellate (o altra unit specifica)
    installed_power_vessel = 19200   # kW
    deck_area = 1850                # m
    vessel_speed = 17.6             # m/s
    number_of_workers = 3           # persone a bordo
    worker_cost_per_hour = 50       # /ora

    vessel_charter_rate = 26.15*crane_capacity+5842.59   # /giorno
    turbine_deck_area = rotor_diameter**2                       # m
    elements_per_trip = math.floor(deck_area/turbine_deck_area)          # n elementi per viaggio
    number_of_trips = math.ceil(number_of_structures*turbines_per_structure/elements_per_trip)         # totale viaggi necessari
    installation_time = turbine_deployment*number_of_structures*turbines_per_structure   # giorni per substructure
    installation_time_days = installation_time/24    # t inst (giorni totali di installazione)

    cost_fuel = installed_power_vessel*fuel_price*0.8*210*24/10**6     # carburante
    cost_vessel = vessel_charter_rate+cost_fuel                        # noleggio nave
    installation_cost_turbine = cost_vessel*(installation_time_days+number_of_trips*2*distance_from_shore/vessel_speed/1000/24)   # installazione base
    installation_cost_turbine_adjusted = installation_cost_turbine*inflation_correction  #  installazione aggiustata
    workers_total_cost_Aker = number_of_workers*worker_cost_per_hour*24*(installation_time_days+number_of_trips*2*distance_from_shore/vessel_speed/1000/24) #  totale lavoratori

    return installation_cost_turbine_adjusted, workers_total_cost_Aker
```

A.3.8 monopile_installation_RAMBIZ.py

```
    import math

def compute_installation_costs_Rambiz(number_of_structures,monopile_deployment,fuel_price,distance_from_shore,inflation_correction):


    crane_capacity = 1700           # tonnellate (o altra unit specifica)
    installed_power_vessel = 3000   # kW
    #deck_area                 # m
    vessel_speed = 13           # m/s
    number_of_workers = 3          # persone a bordo
    worker_cost_per_hour = 50      # /ora

    vessel_charter_rate = 42.24*crane_capacity+11871.96   # /giorno
    elements_per_trip = 1          # n elementi per viaggio
    number_of_trips = math.ceil(number_of_structures/elements_per_trip) # totale viaggi necessari
    installation_time = monopile_deployment  # giorni per monpile
    installation_time_total = installation_time*number_of_structures    # t inst (giorni totali di installazione)

    cost_fuel = installed_power_vessel*fuel_price*0.8*210*24/10**6     # carburante
    cost_vessel = vessel_charter_rate+cost_fuel                         # noleggio nave
    installation_cost_monopile = cost_vessel*(installation_time_total+number_of_trips*2*distance_from_shore/vessel_speed/1000/24)   # installazione base
    installation_cost_monopile_adjusted = installation_cost_monopile*inflation_correction  #  installazione aggiustata
    workers_total_cost_Rambiz = number_of_workers*worker_cost_per_hour*24*(installation_time_total+number_of_trips*2*distance_from_shore/vessel_speed/1000/24) #  totale lavoratori

    return installation_cost_monopile_adjusted, workers_total_cost_Rambiz
```

A.3.9 opex.py

```
    import math
```

```python

def compute_opex(fuel_price,*capex_values,spare_part_cost_percentage, inflation_correction, distance_from_shore,):

    # vessel for maintenace (MV C-Odyssey)
    Loa= 26 # m
    installed_power = 1790 # kW
    deck_area = 120 # m^2
    Vessel_speed = 18.5 # km/h
    workers_per_vessel= 3\
    cost_workers_per_h= 50 # /h
    vessel_charter_rate = 63.23*Loa+1812.4 # /day
    cost_fuel = installed_power*fuel_price*0.8*210*24/10**6 # /day
    cost_vessel = vessel_charter_rate + cost_fuel # /day


    names = [
        "Drivetrain", "Electric system", "Nacelle", "Blade", "Support structure",
        "Pitch system", "Gearbox", "Power converter", "Generator", "Control system"
    ]
    failure_rates = [0.44, 0.17, 0.12, 0.09, 0.06, 0.04, 0.04, 0.02, 0.02, 0.01]
    repair_times = [42, 3.5, 42, 4.9, 350, 30.8, 31.5, 35, 98, 31.5]
    numbers_of_technicians = [4, 2, 6, 2, 8, 4, 4, 4, 4, 2]

    if len(capex_values) != len(names):
        raise ValueError("The number of CAPEX values must match the number of components.")

    results = []

    total_spare = 0
    total_vessel = 0
    total_workers = 0

    for i in range(len(names)):
        name = names[i]
        failure_rate = failure_rates[i]
        repair_time = repair_times[i]
        capex = capex_values[i]
```

```
        number_of_technicians = numbers_of_technicians[i]

        time_repaire_reduced_for_floating= repair_time*0.7
        spare_part_cost= capex*spare_part_cost_percentage
        vessel_cost_per_component= cost_vessel*(repair_time/24+4*distance_from_shore/1000/Vessel_speed/24)*inflation_correction # 4 bc 2 roundtrips
        workers_cost= number_of_technicians*cost_workers_per_h*(repair_time+4*distance_from_shore/Vessel_speed/1000)
        spare_with_failure_rate=failure_rate*spare_part_cost
        vessel_with_failure_rate= failure_rate*vessel_cost_per_component
        workers_with_failure_rate= workers_cost*failure_rate


        component = {
            "name": name,
            "failure_rate": failure_rate,
            "repair_time": repair_time,
            "capex": capex,
            "number of technicians" : number_of_technicians,
            "spare part cost with failure rate" : spare_with_failure_rate,
            "vessel cost with failure rate" : vessel_with_failure_rate,
            "workers cost with failure rate" : workers_with_failure_rate
        }

        results.append(component)

        # Accumuliamo i totali
        total_spare += spare_with_failure_rate
        total_vessel += vessel_with_failure_rate
        total_workers += workers_with_failure_rate

    maintenance_cost= total_spare+total_vessel+total_workers



    return results,maintenance_cost
```