Algoritmo para caracterizar el oleaje a bordo de boyas de deriva
Algorithm for characterizing wave conditions onboard drifting buoys

Trabajo de Fin de Máster
Curso 2024–2025

Autor
Carlos Gómez Yubero

Director
Luis Piñuel Moreno
Segundo Esteban San Román

Colaborador
Javier Romera Llave

Máster en Internet de las Cosas
Facultad de Informática
Universidad Complutense de Madrid

Algoritmo para caracterizar el oleaje a bordo de boyas de deriva
Algorithm for characterizing wave conditions onboard drifting buoys

Trabajo de Fin de Máster en Internet de las Cosas
Departamento de Informática

Autor
Carlos Gómez Yubero

Director
Luis Piñuel Moreno
Segundo Esteban San Román

Colaborador
Javier Romera Llave

Convocatoria: Junio 2025
Calificación: *8*

Máster en Internet de las Cosas
Facultad de Informática
Universidad Complutense de Madrid

2 de julio de 2025

Agradecimientos

A Javier, a Luis y a Segundo por haberme guiado durante el tiempo de desarrollo de este trabajo.

A mi familia por haberme apoyado durante toda mi vida, tanto en la parte personal como en la académica.

Resumen

Algoritmo para caracterizar el oleaje a bordo de boyas de deriva

Actualmente, se llevan a cabo muchos trabajos sobre la superficie marina, que, por norma general, tiene unas características variables dependiendo de la zona geográfica que se estudie. Conocer el comportamiento del océano puede ayudar a prevenir situaciones de peligro para el personal de trabajo en estas áreas. De esta forma, conociendo los datos del estado de la superficie marina se puede realizar un diseño, tanto estructural para el caso de construcciones o de planificación de rutas para travesías de los barcos, se pueden prever tragedias futuras.

En este trabajo de final de máster, se busca el diseño de un sistema de boyas de deriva capaz de monitorizar el estado del mar a partir de la estimación de la altura y periodo del oleaje, enviando los datos mediante conexiones satelitales o por cobertura móvil cuando esté disponible, aprovechando el avance del IoT sobre las zonas marítimas. El proyecto se enfocará en el diseño e implementación del algoritmo, situado en las boyas desplegadas, el objetivo de este es caracterizar el oleaje por su altura y periodo a partir de las mediciones de aceleración obtenidas desde una IMU. Para desarrollar el algoritmo se diseñó e implementó en MATLAB una simulación capaz de modelar tanto el oleaje como el movimiento realista de una boya frente a dicho oleaje. De este modo, se realizaron múltiples pruebas para estimar con precisión el comportamiento del mar y validar el correcto funcionamiento del algoritmo.

Una vez implementado el algoritmo de estimación, se procederá a probar su eficacia mediante una IMU real sometida a un movimiento vertical, buscando simular el movimiento de un oleaje, para poder analizar los resultados de la estimación en diversas pruebas físicas.

Palabras clave

Unidad de Medición Inercial (IMU), Internet de las Cosas (IoT), Grados de libertad (DOF), JONSWAP, Diseño Basado en Modelo (MBD)

Abstract

Algorithm for characterizing wave conditions onboard drifting buoys

Currently, much work is being carried out on the ocean surface, which exhibits variable characteristics based on the geographic area. A thorough understanding of ocean behavior is crucial for preventing dangerous situations for personnel working in these environments. By knowing the sea surface conditions, we can optimize structural designs for marine construction and refine route planning for ship crossings, ultimately averting potential tragedies.

This master's thesis focuses on designing a drifting buoy system to monitor sea state by estimating wave height and period. The system will transmit data via satellite connections or, when available, mobile coverage, leveraging advancements in IoT for maritime zones. The project will specifically address the design and implementation of an embedded algorithm within these buoys. This algorithm's primary objective is to characterize waves by their height and period using acceleration measurements obtained from an IMU. To develop this algorithm, a MATLAB simulation was designed to model both waves and the realistic motion of a buoy subjected to them. Multiple tests were conducted to accurately estimate sea behavior and validate the algorithm's functionality.

Once the estimation algorithm is implemented, its efficacy will be tested using a real IMU subjected to vertical motion, simulating wave movement. This will allow to analize the estimation results across various physical trials.

Keywords

Inertial Measurement Unit (IMU), Internet of Things (IoT), Degrees of Freedom (DOF), JONSWAP, Model Based Design (MBD)

Índice

Índice de figuras

Índice de tablas

Índice de ecuaciones

Capítulo 1

Introducción

En los últimos años, la implementación de nuevos sistemas inteligentes ha provocado un aumento en el uso del ecosistema del Internet de las Cosas (IoT). Esta extensión se ha visto reflejada tanto sobre la superficie terrestre como en la marítima, donde, para esta última, la conexión satelital y la cobertura móvil han permitido instaurar grandes redes de boyas inteligentes en los océanos, las cuales tienen sensores instalados que permiten realizar mediciones de temperatura, de la salinidad del agua o para caracterizar el oleaje, permitiendo obtener datos precisos y fiables en zonas de difícil acceso, pudiendo así construir modelos digitales del océano para facilitar el estudio de este y prever cómo se modificará la superficie marítima con el paso del tiempo.

En este trabajo se diseñará e implementará un sistema de boyas de deriva capaz de monitorizar el estado del mar a partir de la estimación de la altura y periodo del oleaje. El objetivo principal de este despliegue es mejorar la seguridad en la zona donde se encuentre el sistema de boyas, facilitando la información del oleaje tanto al personal que tiene labores en el mar como para alertar sobre fenómenos adversos en el mismo.

El despliegue de redes de boyas permitiría la creación de un Sistema de Alerta Temprana (EWS), el cual es capaz de detectar eventos críticos en tiempo real y en caso de que se pase de un umbral establecido previamente, poder distribuir una alerta a los responsables de actuación para poder hacer frente a estos eventos y que el evento en cuestión cause los menos daños posibles. El factor a analizar en estos sistemas puede ser variante, por ejemplo, en el trabajo de Chacón et al. (2024), se menciona un sistema capaz de controlar la calidad del agua, detectando floraciones nocivas de algas y cianobacterias (HACBs). También se podría analizar un factor referente al estado del oleaje, como sería nuestro caso, sobre el cual se han encontrado algunos trabajos donde se buscaba alertar a las poblaciones cercanas sobre posibles tsunamis o inundaciones costeras, estos estudios se encuentran en los trabajos de Haq et al. (2019) y Doong et al. (2012). Nuestro objetivo del sistema es implementar, junto a la estimación del oleaje, un EWS capaz de detectar eventos críticos en tiempo real, permitiendo obtener avisos sobre anomalías en el estado del oleaje que podrían ser peligrosas, ya sea para los posibles barcos que estén cerca de la zona o para las zonas costeras próximas donde este oleaje pueda llegar a afectar.

En el sistema diseñado de Andrade et al. (2024), se ha implementado una arquitectura IoT de tres niveles (*Fog-Cloud-Edge*), lo que permite manejar eficientemente el gran volumen de datos y cumplir con los exigentes requisitos de latencia, fiabilidad y escalabilidad.

Para nuestro sistema también se implementará una arquitectura *Fog-Cloud-Edge*, en la

cual, las capas estarán definidas de la siguiente manera:

- **Nivel *Edge*:** en este nivel se encuentran las boyas distribuidas sobre el océano, las cuales dispondrán de una batería recargable mediante un panel solar, un módulo satelital que permitirá enviar los datos necesarios a los centros de control, y también tendrá implementada una IMU que permitirá recoger los movimientos de la boya producidos por el oleaje. Estas mediciones se tratarán mediante un algoritmo de estimación para poder caracterizar el oleaje.

  En esta capa, se recogen y se hace un procesamiento inicial de los datos medidos por la IMU. En este procesamiento se implementan procesos de filtrado para reducir el ruido de las señales con la finalidad de reducir la cantidad de datos totales a tratar, eliminando los erróneos o inválidos. Una vez se hayan filtrado los datos indeseados, se realizará un segundo tratamiento sobre los datos correctos, donde se calculará el desplazamiento vertical de la boya para poder caracterizar el oleaje en tiempo real, obteniendo así la altura y periodo del mismo. Realizando la estimación del oleaje en la propia boya, se consigue que la cantidad de datos a enviar para tratarlos posteriormente a mayor escala, se haya visto enormemente disminuido, consiguiendo así un envío con menor latencia, ancho de banda y por consecuente un menor gasto de batería y se disminuyen los costes de las transmisiones, lo que se agradece en comunicaciones satelitales.

  Finalmente, en este nivel también se implementa la lógica del EWS, el cual será capaz de detectar eventos críticos en tiempo real, permitiendo obtener avisos sobre anomalías en el estado del oleaje que podrían ser peligrosas, ya sea para los posibles barcos que estén cerca de la zona o para las zonas costeras próximas donde este oleaje pueda llegar a afectar.

  Nivel *Fog*: este es el nivel intermedio, en el diseño actual del sistema no se utilizará esta capa, aunque no se descarta implementarla en un futuro dependiendo de la arquitectura de la red de despliegue del sistema.

  Nivel *Cloud*: este es el nivel más alto de la arquitectura y este está formado por centros de datos más avanzados y con mayor capacidad de almacenamiento y cómputo que los anteriores, situándose en territorios más alejados del conjunto de boyas.

  En esta capa se tratan la gran cantidad de datos enviados por las boyas del nivel *Edge*, consiguiendo así la centralización de todos los datos recibidos desde las diferentes boyas. Este conjunto de datos será utilizado para obtener un mapeado del estado del océano, aprovechando el mayor nivel de cómputo respecto a las boyas, permitiendo utilizar algoritmos y modelos avanzados que permitan modelar un oleaje complejo y en algunos casos llegar a realizar predicciones sobre como variará el estado del mar en un intervalo próximo de tiempo. De esta forma los datos procesados, se pueden facilitar tanto al personal que trabaja en labores marítimas, como podrían ser los operadores que se encargan del mantenimiento de las boyas o los pesqueros que utilizan las estimaciones del oleaje en tiempo real para diseñar la ruta diaria, o para enviar las posibles alertas del EWS tanto a las autoridades pertinentes si supone un peligro para la población o en caso de que sea alguna alerta interna, poder revisarla y actuar frente a ella.

El uso de una arquitectura *Fog-Cloud-Edge* ofrece una gran eficiencia, consiguiendo que los cálculos más necesarios y rápidos se realicen cerca del nodo (nivel *Edge*). Y luego, los cálculos más complejos se realizan en la capa *Cloud*, en la cual hay ubicaciones con una mejor infraestructura para soportar un mayor nivel computacional.

1.1. Motivación

La monitorización en tiempo real de las condiciones oceánicas es un pilar fundamental para la seguridad y la eficiencia de un amplio abanico de actividades marítimas. Desde la navegación comercial y la pesca, que dependen de rutas seguras, hasta la protección de infraestructuras costeras y el desarrollo de energías renovables en alta mar, disponer de datos precisos sobre el oleaje es crucial. Los Sistemas de Alerta Temprana (EWS), alimentados por redes de boyas inteligentes, han demostrado ser vitales para mitigar los riesgos asociados a eventos extremos, como oleajes peligrosos o tsunamis, permitiendo una respuesta rápida que puede salvar vidas y reducir pérdidas económicas.

Si bien existen soluciones comerciales muy precisas, estas suelen depender de sensores especializados y costosos, un ejemplo de estas soluciones serían las boyas amarradas, las cuales se instalan ancladas al suelo marino y permaneces prácticamente estáticas en la superficie. Estas boyas también suelen ser robustas y tienen un coste elevado tanto inicialmente como de mantenimiento. Una desventaja de estos dispositivos es la dependencia hacia el estado del fondo marino, puesto que dependiendo del estado de este, podría limitar las zonas posibles de instalación de estas boyas. Estos factores, han impulsado la investigación hacia alternativas más asequibles y de menor consumo energético, facilitando así despliegues a mayor escala, obteniendo como resultado de la investigación las boyas de deriva, las cuales se sueltan en el mar a la deriva, tal y como indica su propio nombre. Una de las ventajas de estas boyas es su bajo coste de producción y mantenimiento, permitiendo repararlas fácilmente en caso de avería. Aparte al tener una mayor capacidad de despliegue, se pueden liberar en prácticamente cualquier zona de la superficie sin tener que depender del fondo marino, permitiendo acceder a zonas remotas del océano en las cuales sea necesario mantener una monitorización periódica. Si que es verdad que las boyas amarradas obtienen una medición con una señal más limpia dada las situación en la que solo le afecta el movimiento vertical y lateral del oleaje, aunque si que en algunos casos, el amarre podría llegar a limitar las mediciones. En cambio a las boyas de deriva les afectan más factores a parte del propio oleaje, como las corrientes, pero estos factores permiten que se obtengan datos más realistas sobre la superficie marítima.

En este contexto, se plantea un desafío tecnológico capaz de caracterizar un oleaje, estimando su altura y periodo mediante el único uso de una IMU, el cual es un sensor económico, integrable con la mayoría de microcontroladores y con un bajo consumo energético, lo cual es un gran punto a favor cuando el dispositivo donde estará instalado no se encuentra en una zona de fácil accesibilidad y funciona a base de baterías.

La motivación central de este trabajo de fin de máster se centra en el desafío planteado anteriormente. Dada la escasa disponibilidad de estudios centrados exclusivamente en la caracterización del oleaje mediante IMUs, se identifica la necesidad de investigar y validar este enfoque, tanto en un entorno simulado que permita controlar todas las variables del sistema, como a través de una implementación física que demuestre su viabilidad en condiciones reales.

Tal como se ha descrito en la arquitectura de tres niveles (*Edge-Fog-Cloud*), las tareas de estimación de parámetros y detección de alertas deben ejecutarse con la mínima latencia posible. Por esta razón, el presente trabajo se centrará específicamente en el diseño, desarrollo y validación del software de cómputo destinado a la capa *Edge*, es decir, el algoritmo que se ejecutará directamente en la boya. Este enfoque es crítico, ya que el software de la boya debe ser computacionalmente ligero, robusto y eficiente desde el punto de vista energético para garantizar su autonomía.

Para abordar el desarrollo de este software de una manera sistemática y fiable, se adoptará la metodología de Diseño Basado en Modelos (MBD). Este es un paradigma de desarrollo en el que un modelo ejecutable del sistema es el artefacto central durante todo el ciclo de vida del proyecto, desde la concepción inicial y el análisis de requisitos hasta la implementación y las pruebas (Andrade et al., 2024).

El MBD es una metodología muy eficaz para el trabajo a realizar, dado que nos permite modelar nuestro sistema de la siguiente manera:

- **Modelado del oleaje:** para empezar, se podrá generar un oleaje simulado, el cual será la base de nuestro entorno virtual.
- **Modelado de la dinámica de la boya:** una vez generado el oleaje, se obtendrá el movimiento de una boya situada sobre el oleaje simulado.
- **Modelado de la IMU:** a partir del movimiento de la boya se generarán las mediciones de la IMU.
- **Algoritmo de estimación:** una vez obtenidos los diferentes modelos, ya se pueden procesar los datos generados para la IMU para estimar las características del oleaje.

La principal ventaja del MBD es la capacidad que ofrece al proyecto de poder diseñar, desarrollar y realizar pruebas sin disponer de ningún tipo de hardware o entorno físico. Esto permite desarrollar e implementar el algoritmo de estimación a partir de modelos y simulaciones. Para lograr esto se han de modelar tanto el oleaje como la dinámica de la boya cuando ésta se somete al oleaje generado y, seguidamente, a partir del modelado del movimiento de la boya, extraer los datos simulados de la IMU, de forma que se puedan obtener las mediciones de la IMU en una ventana de tiempo y a partir de aquí se puede testear la funcionalidad del algoritmo.

Esto facilita mucho el avance del trabajo, dado que a partir del modelado del oleaje se conoce en todo momento el estado de este respecto al tiempo, sin tener que depender de, por ejemplo, una piscina donde colocar la boya físicamente y tener que generar un oleaje real con unas características específicas.

Por otro lado, se disminuyen los factores de error de las implementaciones físicas, consiguiendo que el proceso de desarrollo y depuración sea más sencillo. Aunque siempre se le pueden añadir valores de sesgo y ruido a los modelos para obtener salidas más realistas.

1.2. Objetivos

El objetivo principal de este trabajo es diseñar e implementar una aplicación capaz de caracterizar el oleaje mediante el uso de las mediciones de una IMU instalada en una boya, realizando una estimación de la altura y periodo del oleaje. Se han definido los siguientes objetivos que permitirán alcanzar de forma sencilla el objetivo comentado:

- Implementación de una aplicación capaz caracterizar un oleaje, para la cual se necesitarán seguir los siguientes objetivos específicos:

    1. Modelación del oleaje mediante el espectro JONSWAP, el cual permite obtener un oleaje con características realistas. Este oleaje será la base para realizar la simulación del algoritmo de estimación. Se busca obtener el máximo realismo

en cuanto al oleaje generado para poder obtener una mayor eficacia del sistema al momento de realizar las pruebas con dispositivos reales.

2. El segundo objetivo se centra en la modelación del movimiento de la boya cuando se somete al oleaje generado previamente. Se busca obtener un movimiento realista teniendo en cuenta los diferentes grados de libertad de la boya.
3. Simular las mediciones de la IMU de forma realista, considerando una serie de ruidos y sesgos, los cuales existirán en la implementación con dispositivos físicos.
4. Desarrollar un algoritmo de estimación preciso y eficaz hacía diferentes tipos de oleajes.

- Implementación en Python para caracterizar el oleaje que recibirá los datos de la aceleración a partir de una IMU, los calibrará y los procesará para lograr la estimación de la ola.
- Implementación de una librería genérica desarrollada en C, capaz de estimar el oleaje dada como entrada la aceleración referenciada con una marca de tiempo.
- Realización de pruebas que justifiquen la eficiencia de las implementaciones anteriores. Las pruebas se realizarán a partir de una estructura conectada a un motor paso a paso para simular el movimiento vertical del oleaje.

1.3. Plan de trabajo

El plan de trabajo se divide en las siguientes partes:

1. **Estudio preliminar y revisión bibliográfica:** se realizará un amplio estudio de diferentes trabajos, para los cuales su objetivo del proyecto estén vinculados con la modelación y simulación, tanto para oleajes, boyas o IMUs, así como algunos enfocados a la caracterización del oleaje y las diversas técnicas existentes. Con este estudio inicial se busca obtener una base de conocimiento sobre las diferentes técnicas de modelación y estimación existentes o los diferentes principios y ecuaciones matemáticas propios de los objetivos especificados para nuestro proyecto. Una vez finalizado el estudio se espera obtener una base de información que permita desarrollar un sistema realista.

   Tiempo estimado: 3 semanas

2. **Implementación en MATLAB:** en esta implementación se encontrarán los diferentes modelados y simulaciones a desarrollar, empezando desde el oleaje, seguido por la dinámica de la boya y finalmente realizando la simulación de la IMU, una vez implementado lo anterior se procederá a diseñar el algoritmo de caracterización, en el cual hará falta implementar los modelos matemáticos pertinentes para obtener unos buenos resultados de la estimación de la altura y periodo del oleaje generado. MATLAB nos será muy útil para esta implementación dada su facilidad de uso sobre ecuaciones y modelos matemáticos, así como su facilidad para generar una serie de gráficas que facilitarán el análisis de los resultados obtenidos.

   Tiempo estimado: 3 semanas

3. **Implementación de la estimación en Python:** una vez validado el algoritmo de estimación, se procederá a comprobar el funcionamiento del mismo teniendo como

entrada los datos de un acerelómetro real, los cuales serán procesados antes de pasarlos por el algoritmo. Se ha escogido desarrollar el código en Python dada su similitud en cuando a funciones matemáticas con MATLAB y que también permite graficar los resultados de una forma sencilla.

Tiempo estimado: 2 semanas

4. **Desarrollo del sistema de pruebas:** Para poder realizar las pruebas del algoritmo en un entrono físico, se diseñara un dispositivo mecánico, el cual permitirá generar un movimiento vertical sinusoidal para simular el comportamiento del oleaje sobre una IMU física.

   Tiempo estimado: 1 semana

5. **Implementación de la librería de estimación en C:** como última implementación se hará una librería genérica en C para permitir la estimación del olaje, a partir de datos del acelerómetro obtenidos mediante la lectura de un fichero en formato csv y los cuales ya habrán sido procesados previamente para realizar una estimación precisa. En esta implementación se tendrán que diseñar funciones propias para implementar los diferentes procesos que son necesarios para hacer caracterizar el oleaje.

   Tiempo estimado: 3 semanas

6. **Pruebas a realizar:** para cada una de las implementaciones se realizaran una serie de pruebas para poder analizar los resultados de estimación y de esta manera poder verificar la precisión y eficacia del algoritmo desarrollado.

   Tiempo estimado: 2 semanas

1.4. Esquema de la memoria

Esta memoria está compuesta por los siguientes capítulos:

1. **Introducción:** en este primer capítulo se contextualizarán los diferentes aspectos que han llevado a la motivación del trabajo, así como la serie de objetivos establecidos y un plan de trabajo en el que se destacan las diferentes tareas a realizar en el proyecto.

2. **Estado del arte:** en el segundo capítulo, se ha realizado el estudio bibliográfico de diferentes trabajos realizados referentes a diversas áreas que se tratarán durante la elaboración del trabajo.

3. **Diseño basado en modelo:** el tercer capítulo corresponde a la explicación de los modelos y fundamentos matemáticos que se han seguido para lograr un correcto resultado del trabajo realizado, siguiendo algunas de las bases estudiadas en el capítulo anterior, entre las cuales se destacan el modelado del oleaje, de la dinámica de la boya y de las mediciones de la IMU, pudiendo así implementar el algoritmo que permite la caracterización del oleaje.

4. **Implementación:** en este capítulo se han mencionado y explicado las diferentes implementaciones diseñadas y desarrolladas que han permitido comprobar los resultados del algoritmo de estimación implementado.

5. **Conclusiones y trabajo futuro:** finalmente, en el último capítulo se hace un análisis, tanto de los objetivos establecidos al principio del trabajo como de los resultados obtenidos durante las pruebas realizadas. También se expresa un diagrama explicando la organización real respecto al tiempo de las tareas realizadas así como algunos trabajos futuros para obtener un algoritmo más completo.

Capítulo 2

Estado del arte

En los últimos tiempos, la caracterización precisa del oleaje se ha convertido en un desafío clave para garantizar la seguridad y eficiencia, tanto para las operaciones marítimas como para el diseño de estructuras marítimas y de sistemas flotantes.

Puesto que el trabajo se basa en un diseño basado de modelos, en este capítulo se estudiaran algunos modelos disponibles tanto para el oleaje o para las dinámicas de la boya. El modelado del oleaje, se realiza a partir de modelos espectrales, como podría ser el de Pierson-Moskowitz o JONSWAP, los cuales usan algunos parámetros del oleaje como podrían ser la altura de la ola o el periodo de la misma. A partir de estos fundamentos se desarrollaron algunos algoritmos de síntesis espectral, los cuales son capaces de generar series temporales realistas de elevación mediante la superposición de componentes con fases aleatorias, lo que resulta en la generación de un oleaje realista.

Por otra parte, se repasarán técnicas para estimar el oleaje a partir de los datos proporcionados por una IMU en un sistema lagrangiano. Se estudiarán trabajos en los que se estima el oleaje gracias al uso de IMUs situadas en boyas. De esta forma se pude conseguir una caracterización precisa y fiable del oleaje. Para lograr esta estimación se ha de realizar un procesamiento de las señales obtenidas, así como un filtrado de las mismas en el propio dispositivo flotante.

Á continuación se entrará más en detalle sobre los diferentes modelos y técnicas comentadas al inicio de este capítulo.

2.1. Modelación del oleaje

En este apartado se presentan distintos modelos teóricos y numéricos que permiten generar el oleaje de forma realista. Estos modelos se pueden clasificar en dos categorías principales:

- **Analíticos:** describen el movimiento de la superficie a partir de soluciones analíticas, como podrían ser las ondas de Gerstner o las aproximaciones lagrangianas.
- **Espectrales:** sintetizan el campo de alturas mediante transformadas de Fourier y procesos estocásticos que vienen guiados por parámetros físicos, como podrían ser la velocidad del viento o la profundidad. Entre los espectros empíricos más habituales se encuentran Philips y JONSWAP.

Cada una de estas categorías ofrece un nivel distinto entre el realismo visual, el control paramétrico y el coste computacional. Estos factores se han tenido en cuenta al momento de seleccionar las técnicas a implementar durante la realización de este trabajo.

2.1.1. Fundamentos físicos y matemáticos

Los fundamentos físicos y matemáticos de los modelos de oleaje permiten obtener un modelado realista de la superficie del agua.

2.1.1.1. Ecuación de Navier-Stokes y del flujo potencial

Estos fundamentos matemáticos tienen una gran importancia a la hora de modelar el oleaje.

La ecuación de Navier-Stokes permite calcular el comportamiento de cualquier movimiento del agua. A partir de esta ecuación se obtiene la ecuación de Bernoulli, lo que permite obtener una simulación sobre la envolvente de las olas (Tessendorf, 2001). En forma vectorial se describe como:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + g$$

Ecuación 2.1: Ecuación de movimiento

2.1.1.2. Flujo potencial

El flujo potencial permite definir un escalar ϕ tal que $\mathbf{u} = \nabla\phi$. De forma que la dinámica volumétrica queda reducida en la ecuación de Laplace en todo el dominio (Tessendorf, 2001), la cual es la siguiente:

$$\nabla^2 \phi = 0$$

Ecuación 2.2: Ecuación de Laplace

En la superficie libre $z = h(x, y, t)$ aparecen las dos condiciones siguientes:

- **Cinemática:**

$$\partial_t h + \nabla h \cdot \nabla \phi - \partial_z \phi = 0$$

- **Dinámica:**

$$\partial_t \phi + gh = 0$$

  Se evalúa en $z = 0$.

Al combinar las dos ecuaciones anteriores se obtiene la siguiente ecuación de evolución para h:

$$\frac{\partial^2 h}{\partial t^2} = -g\sqrt{-\nabla^2}h$$

Luego, para obtener la fórmula base de todos los métodos espectrales, donde cada modo **k** evoluciona en el tiempo con fase $\omega(k)$, lo cual se aprovecha tanto para la síntesis por FFT como en la generación de olas mediante espectros empíricos. La fórmula se obtiene a partir de las soluciones armónicas $h \propto e^{i(k\cdot x-\omega t)}$, donde aparece la relación de dispersión:

$$\omega^2 = g \cdot k \cdot \tanh(kh)$$

y que en aguas profundas ($kh \gg 1$) se simplifica a:

$$\omega = \sqrt{g \cdot k} \qquad (Tessendorf,\ 2001)$$

2.1.2. Modelos analíticos

Los modelos analíticos describen el movimiento de la superficie marina, sin tener que modelar el comportamiento del fluido completo, solo de la superficie, para lograr esto se centra en la simulación de cada una de las partículas que forman la superficie para obtener el modelo final. Su principal ventaja es la simplicidad computacional y el control de los parámetros como la altura (H), la longitud de onda (λ) y el período (T).

2.1.2.1. Modelo de Gerstner

El modelo de Gerstner simula el oleaje a partir de partículas que describen una trayectoria circular de radio $r = H/2$, donde H es la altura total de la ola, este desplazamiento circular se realiza alrededor de la posición de reposo x_0, z_0. En dos dimensiones, la posición de cada partícula en función del tiempo viene dada por: (Fournier y Reeves, 1986)

$$\begin{cases} x(t) &= x_0 + r\ \sin(k\,x_0 - \omega\,t) \\ z(t) &= z_0 - r\ \cos(k\,x_0 - \omega\,t) \end{cases}$$

Ecuación 2.3: Ecuaciones paramétricas del modelo Gerstner

Donde:

$$H = 2r, \quad k = \frac{2\pi}{L}, \quad \omega = \frac{2\pi}{T}$$

De esta forma se obtiene, en aguas profundas ($kh \gg 1$), la relación de dispersión:

$$\omega = \sqrt{g \cdot k}$$

Y la velocidad de fase $c = \frac{L}{T} = \frac{\omega}{k}$. (Fournier y Reeves, 1986)

2.1.2.2. Superposición de modos en una dimensión

La superposición de modos permite construir la superficie de un oleaje más complejo mediante la suma de múltiples ondas independientes, cada una de ellas generada con unas características diferentes. Este enfoque mantiene la eficiencia del modelo analítico.

Como se comenta en el artículo de Tessendorf (2001), en el cual se utiliza la siguiente ecuación para el modelado en 1 dimensión:

$$h(x,t) = \sum_i a_i \cos\left(k_i x - \omega_i t + \phi_i\right)$$

Ecuación 2.4: Superposición de modos 1-D

Donde:

- a_i es la amplitud.
- k_i es el número de onda.
- ω_i es la frecuencia.
- ϕ_i es la fase inicial aleatoria.

2.1.3. Modelos espectrales

Los modelos espectrales se centran en un punto de vista estadístico que modela el oleaje a partir de un proceso gaussiano estacionario. Para realizar el modelado se necesita conocer la densidad espectral de energía $S(\omega, \theta)$ para, posteriormente, calcular la altura en cada punto de superficie mediante la superposición de modos. (Tessendorf, 2001)

2.1.3.1. Espectro de Phillips

El espectro propuesto por Owen M. Phillips para el modelado del oleaje de un mar complejo, es el siguiente (Badulin y Zakharov, 2020):

$$E(\omega) = \alpha_{Ph} \cdot g^2 \cdot \omega^{-5}$$

Ecuación 2.5: Espectro de Phillips

Donde:

- $\alpha_{Ph} = 0{,}0081$ es la constante de Phillips.
- g es la aceleración de la gravedad.
- ω es la frecuencia angular.

Phillips estableció un equilibrio entre la fuerza del viento sobre la ola y la disipación de la fuerza de la misma cuando se rompe la cresta de la ola, dado que a frecuencias muy

cortas ($\omega \gg \omega_p$) el viento no añade energía adicional al oleaje, y la disipación generada cuando se rompe la cresta de la ola predomina. El parámetro de saturación

$$B(\omega) = \frac{\omega^5 E(\omega)}{2g^2}$$

Ecuación 2.6: Parámetro de saturación de Phillips

se estabiliza alrededor de $3 \cdot 10^{-3}$, lo que da como resultado la pendiente universal ω^{-5} establecida por Phillips (Badulin y Zakharov, 2020).

Para un campo direccional, la densidad espectral de altura se expresa de la siguiente forma:

$$P_h(k) = A\frac{e^{1/(KL)^2}}{k^4} \cdot \left|\mathbf{k} \hat{\cdot} \hat{\mathbf{w}}\right|^2$$

Ecuación 2.7: Espectro direccional de Phillips

Donde $k = ||\mathbf{k}||$, $L = \frac{V^2}{g}$ es la escala eólica y el factor $A = \frac{\alpha_{Ph}}{4\pi}$ concentra la energía en la dirección media del viento (Tessendorf, 2001).

2.1.3.2. Espectro Pierson-Moskowitz

El modelo de Pierson & Moskowitz describe un oleaje ya desarrollado, asumiendo que ha habido viento durante un tiempo y distancia suficiente para que el flujo de energía se equilibre, con lo que se consigue que el espectro solo dependa de la gravedad g y de la velocidad media del viento U, medida a 19,5 m sobre el nivel del mar. El espectro unidimensional queda de la siguiente manera (Pierson y Moskowitz, 1963):

$$S_{PM}(\omega) = \alpha_{PM} \cdot g^2 \cdot \omega^{-5} \cdot e^{-1{,}25\left(\frac{\omega_p}{\omega}\right)^4}$$

Ecuación 2.8: Espectro Pierson-Moskowitz (ω)

Donde:

- $\alpha = 0{,}0081$ es la constante de Phillips.
- $\omega_p = 0{,}877g/U$ es la frecuencia pico obtenida al imponer $\frac{\partial S}{\partial \omega} = 0$.

2.1.3.3. Espectro JONSWAP

El espectro JONSWAP modela un oleaje no desarrollado y el cual se ha sometido a un periodo de viento finito, a diferencia del espectro Pierson-Moskowitz. Con esto se consigue un pico de energía alto y estrecho en un rango cercano a la frecuencia dominante del oleaje (Guo y Xu, 2011).

En la práctica, el espectro JONSWAP se utiliza tanto en ingeniería costera como en proyectos de simulación numérica y gráfica de oleajes. Este espectro ofrece la posibilidad

de caracterizar el oleaje con su altura significativa (H_s (m)), el periodo pico (T_p (s)) y junto con el factor pico ($\gamma > 1$), se puede obtener un oleaje con más o menos desarrollo temporal, lo que permite simular las diferentes fases del oleaje.

La densidad espectral de altura, siguiendo la base del espectro Pierson-Moskowitz, se expresa como (Guo y Xu, 2011):

$$S(\omega) = \alpha \cdot g^2 \cdot \omega^{-5} \cdot e^{-1{,}25\left(\frac{\omega_p}{\omega}\right)^4} \cdot \gamma^{e^{\left(-\frac{(\omega-\omega_p)^2}{2\cdot\sigma^2\cdot\omega_p^2}\right)}}$$

Ecuación 2.9: Espectro JONSWAP unidimensional

Donde:

- $\alpha = 0{,}076\frac{H_s^2}{T_p^2}$ es la escala de energía.
- $\omega_p = \frac{2\pi}{T_p}$ es la frecuencia pico.
- γ es el factor pico, el valor estándar es de 3,3.
- σ es la anchura del pico.

$$\begin{cases} \sigma = 0{,}07, & \omega \leq \omega_p \\ \sigma = 0{,}09, & \omega > \omega_p \end{cases}$$

Mediante el espectro JONSWAP se pueden modelar oleajes complejos, añadiendo factores direccionales, según Guo y Xu (2011) el espectro bidimensional queda de la siguiente manera:

$$S(\omega,\theta) = S(\omega) \cdot D(\omega,\theta) \qquad \text{que satisface} \qquad \int_{-\pi}^{\pi} D(\omega,\theta) = 1$$

Pero como indican Guo y Xu (2011) se considera que la distribución de energía de la ola es independiente de la distribución direccional, por lo que el espectro JONSWAP bidimensional queda de la siguiente forma:

$$S(\omega,\theta) = S(\omega) \cdot D(\theta)$$

Ecuación 2.10: Espectro JONSWAP bidimensional

Y mencionan dos funciones direccionales comunes:

$$D(\theta) = \frac{2}{\pi} cos^2(\theta), \qquad (|\theta|) \leq \frac{\pi}{2}$$

$$D(\theta) = \frac{8}{3\pi} cos^4(\theta), \qquad (|\theta|) \leq \frac{\pi}{2}$$

2.1.3.4. Superposición de modos en dos dimensiones

La superposición de modos bidimensional permite obtener una malla de alturas definida como $H(x,y,t)$ a partir de un espectro direccional, lo que permite generar un oleaje a lo

largo de una superficie. Tal y como comentan Guo y Xu (2011), realizando la suma doble de ondas planas, se obtiene la siguiente ecuación:

$$H(x,y,t) = \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} a_{ij}\cos\left(k_i \cdot x \cdot \cos\left(\theta_j\right) + k_i \cdot y \cdot \sin\theta_j - \omega_i t + \epsilon_{ij}\right)$$

Ecuación 2.11: Superposición de modos 2-D

Donde:

- $a_{ij} \approx \sqrt{2S(\omega_i,\theta_i)\cdot\Delta\omega_i\cdot\Delta\theta_j}$ es la amplitud.
- k_i es el número de onda.
- ω_i es la frecuencia.
- θ_j es la dirección de propagación.
- ϵ_{ij} es la fase aleatoria.

La malla obtenida por $H(x,y,t)$ reproduce la varianza y la dirección del espectro dado por $S(\omega,\theta)$.

2.2. Modelado del movimiento de una boya

Las boyas tienen un gran papel en cuanto a la monitorización del medio marino. El comportamiento del mar puede llegar a ser muy complejo y irregular, puesto que tiene diversos factores que afectan a este, como el propio oleaje, las mareas o el viento que sopla. Estos factores se han de tener en cuenta al modelar el movimiento de una boya para lograr un sistema estable y poder obtener unos resultados precisos sobre el estudio realizado sobre el mar (Li y Bian, 2021).

En esta sección se tratarán diferentes apartados, donde se hablará de los diferentes movimientos de la boya, de como lograr la estabilidad de la misma y finalmente sobre como se pueden reducir los efectos de factores externos sobre el movimiento de la boya.

2.2.1. Sistemas de referencia y grados de libertad 6-DOF

Para tratar el movimiento de una boya sobre la superficie marítima se pueden usar como referencia dos ejes distintos, el primero que es el eje-Tierra, el cual tiene como referencia el centro de la tierra, por lo que no se tendrán en cuenta los ángulos de rotación del cuerpo, sino que sus desplazamientos, ya sean verticales o horizontales sobre el eje del planeta. Y el segundo eje, el cual se referencia respecto al cuerpo y en este casi si que se tienen en cuenta los ángulos de rotación del objeto, haciendo que los desplazamientos sean más complejos por tener que juntar los movimientos realizados en diferentes ejes teniendo en cuenta los ángulos de rotación de la boya.

El movimiento de un objeto flotante en el mar viene descrito por seis posibles coordenadas independientes, conocidos como los seis grados de libertad (6-DOF). A continuación,

en la figura 2.1 se puede observar un objeto del cual salen los tres ejes de coordenadas y en cada eje están referenciados los grados de libertad correspondientes al mismo.

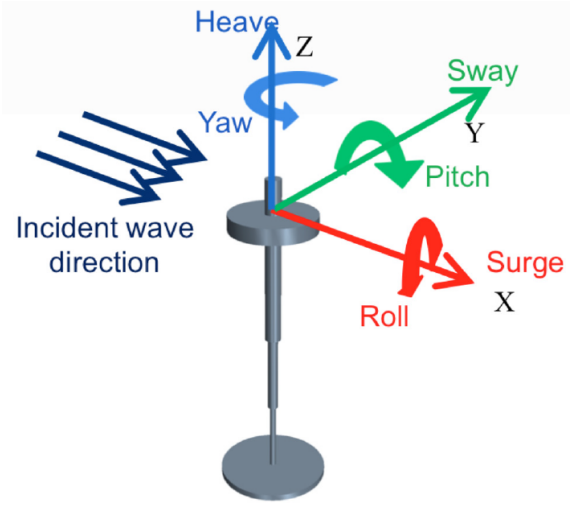

Figura 2.1: Grados de libertad 6-DOF (Johnson y Cotilla-Sanchez, 2020)

Los seis grados de libertad se dividen en dos grupos, de traslación y de rotación.

2.2.1.1. Grados de libertad de traslación

Estos grados de libertad reflejan los desplazamientos lineales de la boya frente a las fuerzas del oleaje que inciden en la misma.

- ***Surge:*** este es el movimiento generado en el eje x, el cual corresponde al avance o retroceso de la boya.
- ***Sway:*** este es el movimiento generado en el eje y, el cual corresponde al movimiento lateral de la boya.
- ***Heave:*** este es el movimiento generado en el eje z, el cual corresponde al desplazamiento vertical de la boya.

2.2.1.2. Grados de libertad de rotación

Estos grados de libertad reflejan la rotación de la boya frente a las fuerzas del oleaje que inciden en la misma.

- ***Roll:*** este es el movimiento de rotación respecto al eje x, este representa la inclinación de la boya lateralmente.
- ***Pitch:***este es el movimiento de rotación respecto al eje y y representa la rotación de la boya frontalmente.
- ***Yaw:*** este es el movimiento de rotación respecto al eje z y representa la orientación de la boya, la cual puede verse afectada por las corrientes del oleaje.

2.2.2. Estabilidad estática y dinámica de la boya

Este apartado está enfocado en la estabilidad de la boya, tanto de forma estática como dinámica.

2.2.2.1. Parámetros de estabilidad estática

Tanto para la rotación en *roll* y en **pitch**, se tiene en cuenta la altura metacéntrica de la boya, tanto de forma transversal como longitudinal respectivamente.

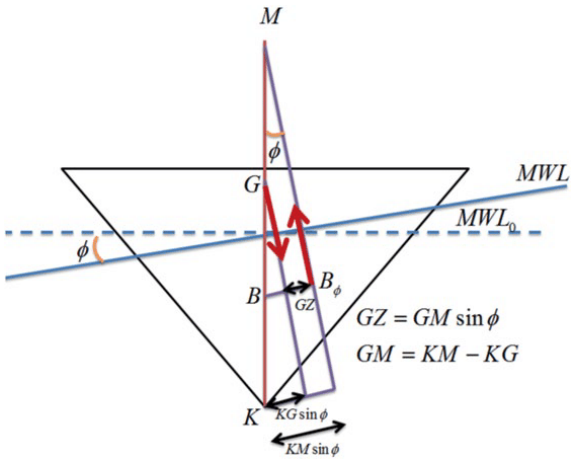

Figura 2.2: Diagrama de puntos de la boya (Karimirad, 2014)

En la figura 2.2 se pueden observar los siguientes puntos, explicados por Zheng et al. (2024):

- **K:** es el punto más bajo de la parte sumergida de la boya.
- **B:** es el centro de flotación de la boya cuando esta se encuentra estable verticalmente.
- B_ϕ**:** es el centro de flotación cuando la boya se encuentra inclinada un ángulo ϕ.
- MWL_0 **y** MWL: estos parámetros indican la línea de flotabilidad de la boya, siendo MLW_0 la línea inicial y MLW la nueva línea después de que la boya haya modificado su ángulo de inclinación.
- **G:** es el centro de gravedad de la boya, se mantiene estable en cualquier posición de la boya.
- **M:** es el metacentro de la boya, es el punto de intersección entre la línea de acción de la fuerza de flotación cuando la boya se encuentra inclinada y la línea vertical que pasa por el centro de flotación en equilibrio.
- **GM:** es la altura metacéntrica de la boya, esta es la distancia entre el centro de gravedad y el metacentro del objeto, y esta se calcula a partir de la distancia KG y la distancia KM, de forma que la altura metacéntrica se calcula como:

$$GM = KM - KG, \begin{cases} GM > 0 & \text{la boya tiende a estabilizarse} \\ GM < 0 & \text{la boya será inestable} \end{cases}$$

Ecuación 2.12: Altura metacéntrica

- **GZ:** indica como se realiza la fuerza de estabilidad estática de la boya, para ángulos pequeños de inclinación de la boya se calcula como:

$$GZ = GM \cdot \sin(\phi)$$

Ecuación 2.13: Fuerza de estabilidad estática para ángulos pequeños

En cambio cuando el ángulo de inclinación es mayor, se ha de recalcular la nueva posición del centro de flotación. Para llegar a la ecuación que permite calcular la fuerza de estabilidad cuando se supera un ángulo de inclinación grande, se han de seguir los siguientes pasos, tal y como se especifica en el documento de Zheng et al. (2024):

1. Cálculo del momento restaurador:

$$M_R = M \times GZ \qquad \text{donde M es el desplazamiento}$$

2. Cálculo de la distancia (L_s) desde la línea de flotación hasta el nuevo centro de gravedad (S) siguiendo como referencia la figura 2.3:

$$L_s = \bar{OE} + \bar{OO'} + \bar{SQ} = l_\phi + c\cos(\phi) + (d_0 - \bar{KS})\sin(\phi)$$

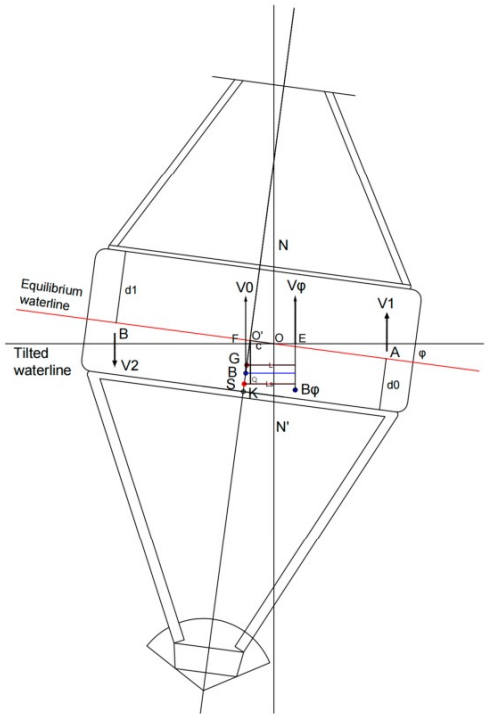

Figura 2.3: Diagrama modificado de puntos de la boya (Zheng et al., 2024)

3. Cálculo de la palanca de estabilidad estática con el nuevo centro de gravedad ($GZ(\phi)$):

$$\begin{aligned} GZ(\phi) &= L_s - SG\sin\phi \\ &= L_s - (\bar{KG} - \bar{KS})\sin\phi \\ &= l_\phi + c\cos(\phi) + (d_0 - \bar{KS})\sin(\phi) - (\bar{KG} - \bar{KS})\sin\phi \end{aligned}$$

Ecuación 2.14: Palanca de estabilidad estática

2.2.2.2. Estabilidad dinámica

Como mencionan Zheng et al. (2024) el momento de inercia es igual al momento adrizante en el sentido opuesto, por lo que:

$$I_v\ddot{\phi} + M \times \bar{GM} \times sin(\phi) = 0$$

Aunque para oscilaciones pequeñas de cabeceo de la boya ($\sin(\phi) \approx \phi$) se queda como:

$$I_v\ddot{\phi} + M \times \bar{GM} \times \phi = 0$$

Por lo que se obtiene la ecuación diferencial de la oscilación armónica simple no amortiguada:

$$\ddot{\phi} + \frac{M \times \bar{GM} \times \phi}{I_v} = 0$$

Ecuación 2.15: Ecuación diferencial de la oscilación armónica simple no amortiguada

Y el periodo natural de balanceo queda descrito por:

$$T_0 = 2\pi\sqrt{\frac{I_v}{M \times \bar{GM}}}$$

Ecuación 2.16: Periodo natural de balanceo

2.2.2.3. Curva de estabilidad dinámica

En el estudio, Zheng et al. (2024), describen la función de densidad de respuesta de amplitud sobre el movimiento rotacional de la boya en función del espectro del oleaje, la relación entre las cuales aparece como:

$$S_{\theta\theta}(\omega) = RAO^2 \times S_{XX}(\omega)$$

Y donde el operador de respuesta de la amplitud (RAO) se calcula como:

$$RAO^2 = \frac{\omega^4}{g^2} \times \frac{1}{\left(1 - \left(\frac{\omega}{\omega_0}\right)^2\right)^2 + 4 \times \mu_0 \times \left(\frac{\omega}{\omega_0}\right)^2}$$

Ecuación 2.17: Operador de respuesta de la amplitud

Donde:

- μ_0 es el coeficiente de amortiguamiento.
- ω_0 es la frecuencia natural angular del movimiento de rotación.

2.2.3. Técnicas de estabilización mecánica

En el trabajo de Malek Azari et al. (2020), se propone utilizar un *Seesaw-like Tuned Mass Damper (STMD)*, el cual es un sistema que permite estabilizar la boya de forma automática, usando un ala sumergida y conectada a la estructura de la boya. Para lograrlo se ha realizado un estudio mediante un dispositivo que se usa como un amortiguador sobre las oscilaciones de la boya, moviéndose en contra-fase del movimiento oscilatorio con el objetivo de atenuar la amplitud del movimiento. El diseño del sistema se puede observar en la figura 2.4.

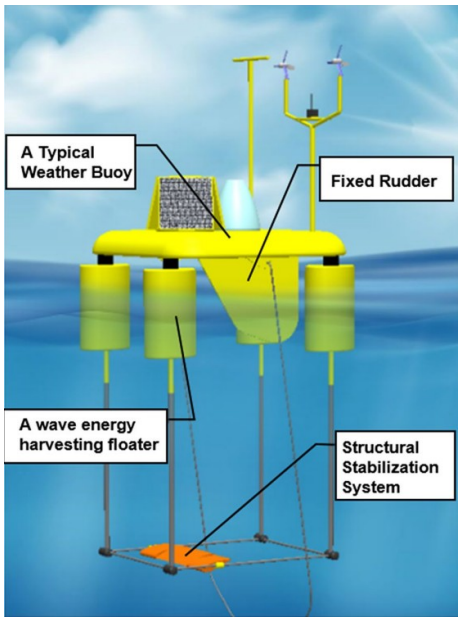

Figura 2.4: Dispositivo STMD (Malek Azari et al., 2020)

Para realizar la simulación de este dispositivo, Malek Azari et al. (2020) usaron dos tipos de STMD, uno pasivo y otro activo.

- **Modo pasivo del STMD:** en este modo no exiten actuadores para modificar el estado del ala sumergida, esta modifica su estado a partir de las propias fuerzas hidrodinámicas generadas sobre el ala. En este caso, el dispositivo se compone de cuatro cilindros que contienen una varilla vertical en el interior que permiten modificar la altura de los mismos frente al movimiento del oleaje.

  En este modo, usando un ala con un peso del 8 % de la masa total de la boya se logró obtener un 81 % de atenuación en resonancia sobre el movimiento de la boya.

  Este modo obtuvo unos buenos resultados frente a un oleaje irregular y ofreció una mejor estabilidad cuando la boya no estaba en resonancia.

- **Modo activo del STMD:** en este modo se añadieron una serie de sensores para controlar los ángulos de rotación de la boya y se conectó a un controlador PID que se encargaba de ajustar el ángulo de actuación del ala sumergida. El controlador sigue el proceso indicado en el diagrama de la figura 2.5.

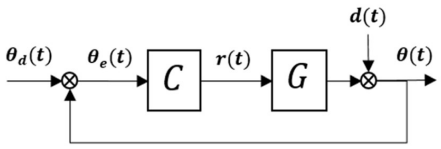

Figura 2.5: Diagrama del control activo PID (Malek Azari et al., 2020)

En este modo, y con la masa del ala del 8 % de la masa total, se obtuvieron resultados llegando a reducir un 79 % el cabeceo de la boya. Y se comprobó que usando este modo se mejoró la respuesta del sistema al poder adaptar la respuesta del ala en tiempo real de forma activa.

2.3. Estimación del oleaje con una IMU

Una solución efectiva y económica para poder estimar el oleaje desde una boya, es con el uso de una IMU, que, a parte de poder instalarla de una forma sencilla en prácticamente cualquier dispositivo, permite recoger datos con una alta frecuencia de muestreo, lo que puede facilitar la estimación del estado marítimo en tiempo real (Gwatiringa, 2018).

2.3.1. Características y limitaciones de una IMU

2.3.1.1. Componentes de la IMU:

Existen dos tipos de IMU, las que tienen 6-DOF, teniendo un acelerómetro y un giroscopio y las de 9-DOF, las cuales también incluyen un magnetómetro.

Cada uno de los dispositivos se encarga de medir unos datos concretos:

- **acelerómetro:** este dispositivo se encarga de medir la aceleración a la que se somete la IMU y que si se resta sobre la aceleración de la gravedad se obtiene la aceleración absoluta que se ha aplicado sobre el dispositivo.

$$a_{abs} = a - g$$

- **giroscopio:** este dispositivo se encarga de obtener los datos relativos a la velocidad angular del dispositivo, y ayuda a conocer la orientación del objeto sobre el que se sitúa, permitiendo conocer los ángulos (ϕ, θ, ψ).
- **magnetómetro:** se encarga de medir el campo magnético terrestre.

Este tipo de dispositivos tienen el inconveniente de que puede aparecer tanto ruido en baja frecuencia como una deriva acumulativa durante el proceso de medición, lo que provoca que se tenga que usar un proceso de filtrado para poder corregir estas desviaciones en los datos y también se ha de tener en cuenta un uso computacional extra para realizar el paso de los datos del eje-cuerpo al eje-mundo (Zhang et al., 2019).

2.3.1.2. Calibración y corrección de sesgos

Como cualquier sensor, para obtener unos datos fiables de los datos obtenidos se ha de realizar un proceso de calibración para los diferentes dispositivos de medida:

- **Calibración del acelerómetro:** de forma general, la calibración del acelerómetro se realiza colocando la IMU en seis posiciones conocidas $(+x, -x, +y, -y, +z, -z)$ siendo x, y, z los tres ejes de coordenadas. Para cada posición conocida se ha de obtener una medición de $\pm 1g \approx \pm 9{,}806 \frac{m}{s^2}$, si no es así haría falta calcular el offset necesario

  $$offset = a - g$$

  para obtener el valor de $\pm 1g$, de forma que para cada medición hecha posteriormente se sume el offset calculado previamente. En un entorno ideal si la IMU esta situada de forma estable y sobre un eje, los otros dos ejes deberían dar una medición de 0.

- **Calibración del giroscopio:** se deberán registrar valores por un tiempo para posteriormente obtener el promedio de los datos calculados para poder restarlo de los datos calculados durante la medición real.

2.3.1.3. Transformación cuerpo → Tierra

Las mediciones de la IMU se pueden realizar desde dos sistemas de referencia conocidos, desde el eje-cuerpo o desde el eje-Tierra. Para el eje-cuerpo, las mediciones varían a la par con el movimiento de la boya, en cambio, para el eje-Tierra, se tienen en cuenta las mediciones respecto al centro de la tierra, teniendo que realizar un procesamiento de los datos recogidos en el eje-cuerpo para pasarlos al eje-Tierra.

Para poder realizar el traspaso entre ejes se utiliza la matriz de rotación $(R(\phi, \theta, \psi))$, la cual se obtiene de la siguiente manera, siguiendo el trabajo de Zhang et al. (2019):

$$R(\phi, \theta, \psi) = R_z(\psi) \cdot R_y(\theta) \cdot R_x(\phi)$$

Donde las matrices de rotación son las siguientes:

- Rotación alrededor del eje x (roll):

  $$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

- Rotación alrededor del eje y (pitch):

  $$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

- Rotación alrededor del eje z (yaw):

  $$R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2.3.2. Algoritmos de fusión

Para poder determinar los ángulos de rotación de la boya, es necesario usar una serie de algoritmos que permiten combinar y filtrar las señales medidas por el acelerómetro, giroscopio y magnetómetro. Estos algoritmos se encargan de compensar la deriva de los giroscopios, el ruido de los acelerómetros y la variabilidad del campo magnético con la finalidad de obtener una estimación precisa y fiable de la orientación real de la IMU (Gwatiringa et al., 2017).

2.3.2.1. Filtro complementario

Un filtro complementario combina de forma sencilla la señal de alta frecuencia del giroscopio con la señal de baja frecuencia del acelerómetro mediante ponderaciones, altas y bajas respectivamente. El filtro complementario se expresa como (Jouybari et al., 2019):

$$angle = \alpha \times (angle + gyroData \times d_t) + (1 - \alpha)) \times (accData)$$

Ecuación 2.18: Filtro complementario

El coeficiente α, definido por Jouybari et al. (2019) como $\alpha = 0{,}98$ permite crear un filtro paso alto para el giroscopio y otro paso bajo para el acelerómetro. En la figura 2.6 se puede observar el diagrama de bloques del filtro complementario.

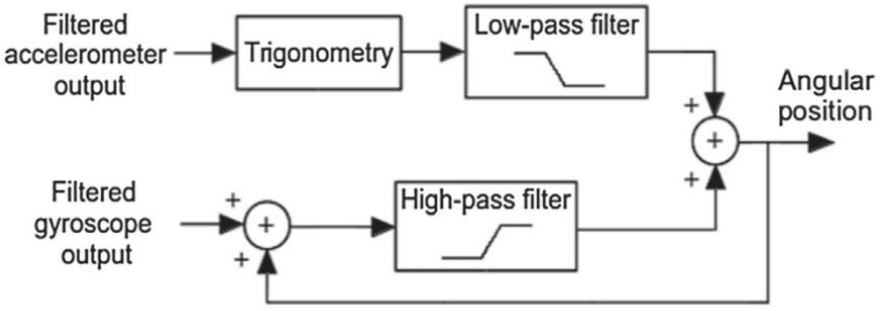

Figura 2.6: Diagrama de bloques del filtro complementario (Jouybari et al., 2019)

2.3.2.2. Filtro de Mahony

El filtro de Mahony es una variante no lineal del filtro complementario. Este introduce términos proporcionales e integrales para corregir los errores de orientación, logrando una compensación de los sesgos de los sensores sin necesidad de linealizarlos. Tal y como se detalla en el trabajo de Jouybari et al. (2019) la actualización del cuaternión $\hat{q}$ se hace con la señal de corrección proporcional.

$$\hat{q}_t = \frac{1}{2}\hat{q}_{t-1} \otimes^S \hat{\omega}_{r,q,t}$$

Ecuación 2.19: Filtro de Mahony

2.3.2.3. Filtro de Madgwick

Este algoritmo busca minimizar el error entre el acelerómetro y la predicción. El algoritmo de integración se formula de la siguiente manera (Jouybari et al., 2019):

$$^{E}\hat{q}_{est,t} = \frac{b\Delta}{l_t||r_f||} + (^{E}\hat{q}_{est,t} + ^{E}\dot{q}_{\omega,t}\Delta t)$$

Ecuación 2.20: Filtro de Madgwick

2.3.3. Estimación de los parámetros de ola

2.3.3.1. Estimación de la altura significativa

En el trabajo realizado por Huang et al. (2016) se presenta la estimación de la altura del oleaje como la siguiente:

$$H_s = 4\sqrt{\sum_{f_l}^{f_u}(S(f) \times d(f))}$$

Ecuación 2.21: Estimación de la altura significativa del oleaje

Como se comenta en el estudio de Huang et al. (2016), la altura significativa del oleaje es aproximadamente igual a la media del primer tercio de las olas más altas.

2.3.3.2. Estimación del periodo pico

El periodo estimado se extrae a partir de la frecuencia pico, expuesta por la FFT, quedando el periodo pico definido como (Huang et al., 2016):

$$T_p = \frac{1}{f_p}$$

Ecuación 2.22: Estimación del periodo pico del oleaje

2.4. Metodología escogida para la implementación

Dados los requerimientos de este proyecto y después de un estudio sobre las diferentes opciones a escoger, en las implementaciones realizadas, se han utilizado las siguientes metodologías:

- **Modelado del oleaje:** después de haber analizado los diferentes modelos para generar un oleaje, finalmente se escogió usar el espectro JONSWAP dado que este permite simular un oleaje en crecimiento, el cual es el que terminará afectando a las medidas de la boya cuando esta se encuentre en mar abierto, a diferencia del modelo de Phillips que genera un mar completamente desarrollado. Por otro lado el

espectro JONSWAP permite, mediante sus parámetros empíricos, poder modificar las características del oleaje de forma precisa.

- **Modelado del movimiento 6-DOF de la boya:** en cuanto al modelado de la boya, se ha utilizado la matriz de rotacion cuerpo → Tierra calculada por Zhang et al. (2019).
- **Modelado y simulación de la IMU:** en cuanto a los datos simulados de la IMU, se han seguido una serie de modelos matemáticos sencillos para lograr modelar los datos del acelerómetro, giroscopio y magnetómetro de la forma más sencilla posible.
- **Estimación de la altura y periodo del oleaje:** para realizar la estimación de la altura significativa y el periodo pico, se han seguido las ecuaciones de Huang et al. (2016) habiendo tratado los datos de la IMU préviamente con una serie de filtros y procesos de integración sencillos.

Capítulo 3

Diseño basado en modelo

En este capítulo se trata la metodología de diseño aplicada al sistema de monitorización del mar mediante boyas de deriva. Como ya se ha comentado, el MBD permite desarrollar un entorno virtual, sin tener que disponer del hardware y entornos físicos complejos. De esta forma, el entorno virtual a generar se compone de un modelado del oleaje, el modelado de la dinámica de la boya y la simulación de las mediciones de una IMU a partir del movimiento de la boya, permitiendo relaizar las pruebas necesarias para verificar el funcionamiento del algoritmo de estimación. A continuación se entra más en detalle en estos puntos comentados anteriormente sobre el modelado, simulación y desarrollo del algoritmo.

3.1. Modelado y simulación del entorno

En esta sección se presentará el proceso de modelado del oleaje seguido por la simulación del estado de una boya, la cual será afectada directamente por el oleaje generado. Para realizar las tareas mencionadas anteriormente se ha usado la herramienta MATLAB, dado que mediante sus características matemáticas se ha podido obtener una implementación precisa y eficaz sobre la estimación de la ola, también se han obtenido una serie de gráficas que han permitido valorar, de forma gráfica, los resultados obtenidos.

En la figura 3.1 se puede observar un diagrama con los diferentes bloques necesarios junto con los datos de salida de cada uno de ellos que permiten lograr la estimación de la ola.

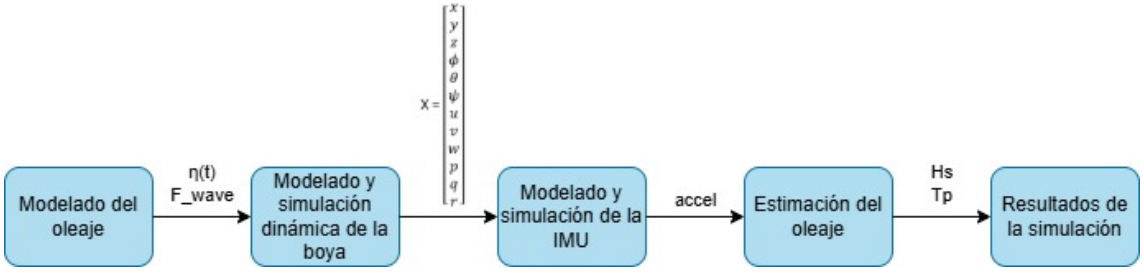

Figura 3.1: Diagrama de los modelos

3.1.1. Modelado del oleaje

Con el modelado del oleaje se busca obtener un comportamiento espacio-temporal de la superficie marítima, usándolo para poder simular las dinámicas de la boya. En este apartado se describen los diferentes modelos utilizados para lograr la generación de la ola, logrando una simulación más simple con una dirección de ola única y una simulación más compleja, en la cual se incluyen algunos desplazamientos laterales.

Los siguientes modelos se implementan mediante la superposición lineal de **N** componentes senoidales, lo que permite ajustar el ancho espectral, la resolución temporal y la banda de frecuencias de interés de la simulación.

3.1.1.1. Generación del espectro JONSWAP

El espectro **JONSWAP** permite distribuir la energía de la ola en diferentes frecuencias, logrando obtener una señal resultante con semejanza al comportamiento real de una ola.

Para modelar la energía de la ola en el dominio de frecuencias se ha usado el espectro **JONSWAP** con **N** componentes.

Primero se han generado **N** frecuencias angulares espaciadas de manera uniforme entre $\omega_{\mathbf{min}}$ y $\omega_{\mathbf{max}}$.

$$\omega_{\mathbf{min}} = 0{,}3 \cdot \left[\frac{2 \cdot \pi}{T_p}\right]$$

$$\omega_{\mathbf{max}} = 3 \cdot \left[\frac{2 \cdot \pi}{T_p}\right]$$

El espectro **JONSWAP** se define como:

$$S_\eta(\omega) = \alpha \cdot g^2 \cdot \omega^{-5} \cdot e^{\left[-1{,}25 \cdot (\frac{\omega_p}{\omega})^4\right]} \cdot \gamma^{e^{\left[-\frac{(\omega-\omega_p)^2}{2 \cdot \sigma^2 \cdot \omega_p^2}\right]}}$$

Ecuación 3.1: Espectro JONSWAP

Donde:

- ω**:** es la frecuencia angular.

$$\omega = 2 \cdot \pi \cdot f$$

- ω_p**:** es la frecuencia pico.

$$\omega_{\mathbf{p}} = \frac{2 \cdot \pi}{T_p}$$

- α**:** es el factor de escala, el cual se encarga de normalizar la altura significativa de la ola.

$$\alpha = 0{,}076 \cdot {H_s}^2 \cdot {T_p}^{-2}$$

  Donde H_s y T_p son la altura significativa y el periodo pico.

- γ**:** es el factor *peak enhancement*, el cual se encarga de controlar el refuerzo del pico de energía espectral alrededor de la frecuencia pico ω_p.

$$\gamma = 3{,}3$$

- σ**:** es el parámetro de ancho de banda que aparece en la parte de refuerzo de pico del espectro **JONSWAP**.

$$\sigma(\omega) = \begin{cases} \sigma_{low}, \omega \leq \omega_p, \\ \sigma_{high}, \omega > \omega_p, \end{cases}$$

3.1.1.2. Generación de la elevación temporal de la ola

Para la generación de la elevación temporal de la ola, se define la amplitud para cada componente de frecuencia como:

$$amp(i) = \sqrt{2 \cdot S_\eta(\omega(i)) \cdot \Delta\omega}$$

Ecuación 3.2: Amplitud de la ola

Y para calcular la elevación temporal $\eta(t)$ se asigna una fase uniforme a cada componente, seleccionada aleatoriamente entre $[0, 2\pi)$ y la elevación se obtiene a partir de la suma de las senoidales:

$$\eta(t) = \sum_{i=1}^{N} amp(i) \cdot cos(\omega_i \cdot t_k + \phi(i))$$

Ecuación 3.3: Ecuación de la altura de ola

3.1.1.3. Cálculo de las fuerzas de excitación de la ola

La fuerza vertical de la ola, se calcula de forma proporcional a la altura de la misma y se expresa como:

$$\mathbf{F_z} = \rho \cdot g \cdot A \cdot \eta(t)$$

Ecuación 3.4: Ecuación de la fuerza vertical de la ola

Donde:

- ρ: es la densidad del agua de mar.
- g: es la aceleración de la gravedad.
- A: es el área de la boya en contacto con la superficie marítima.
- $\eta(i)$: es la elevación de la ola en i.

Para el caso del modelado de olas con desplazamientos laterales también se calculan las fuerzas en las componentes laterales:

$$\mathbf{F_x} = \sum_{i=1}^{N} C_h \cdot \rho \cdot g \cdot \eta(i) \cdot cos(\theta(i))$$

Ecuación 3.5: Ecuación de la fuerza horizontal en x de la ola

$$\mathbf{F_y} = \sum_{i=1}^{N} C_h \cdot \rho \cdot g \cdot \eta(i) \cdot sin(\theta(i))$$

Ecuación 3.6: Ecuación de la fuerza horizontal en y de la ola

Donde:

- C_h : es un coeficiente que permite ajustar la escala de la fuerza hidrodinámica frente a la elevación de la ola.
- $\theta(i)$**:** indica el ángulo con el que actuarán las fuerzas horizontales de la ola.

En cuanto a los momentos generados por el oleaje, solo se generan momentos laterales, los cuales actuarán sobre el grado de inclinación de la boya. Estos se calculan de la siguiente forma:

$$\mathbf{M_x} = \ell_\phi \cdot F_z \cdot k_\phi$$

Ecuación 3.7: Ecuación del momento horizontal en x de la ola

$$\mathbf{M_y} = \ell_\theta \cdot F_z \cdot k_\theta$$

Ecuación 3.8: Ecuación del momento horizontal en y de la ola

Donde:

- ℓ_x**:** indica la distancia entre el centro de gravedad de la boya y el punto de aplicación de la fuerza, tanto para el ángulo de ϕ y de θ.
- F_z: es la fuerza vertical ejercida por la ola.
- k_x**:** es un factor de escala para el momento en ϕ y θ.

3.1.1.4. Funciones de la implementación MATLAB asociadas

Los modelos explicados anteriormente se han implementado en el código MATLAB en las siguientes funciones:

- ***generateRandomWave, generateRandomWaveDirectional:*** en estas funciones se realiza el modelado del oleaje, tanto para el simple como el direccional, empezando por la generación del espectro JONSWAP, seguido del cálculo de la elevación temporal y finalmente se calculan las fuerzas y momentos generados por el oleaje.

- ***waveExcitationForce, waveExcitationForceDirectional***: dentro de las funciones anteriores se usan estas funciones para calcular las fuerzas y momentos del oleaje.

3.1.1.5. Gráficas generadas

En estas gráficas se pueden observar las fuerzas generadas por la ola respecto al tiempo. En el caso de las olas simples solo se muestra la fuerza vertical F_z y para el caso de las olas direccionales también se muestran las fuerzas laterales F_x y F_y.

Como se puede observar en las figuras 3.2 y 3.3, las fuerzas de las olas generadas son idénticas, exceptuando las fuerzas laterales, las cuales sí que tienen valor en el caso de las olas direccionales. Las fuerzas laterales modificarán el comportamiento de la boya durante la simulación del movimiento de esta.

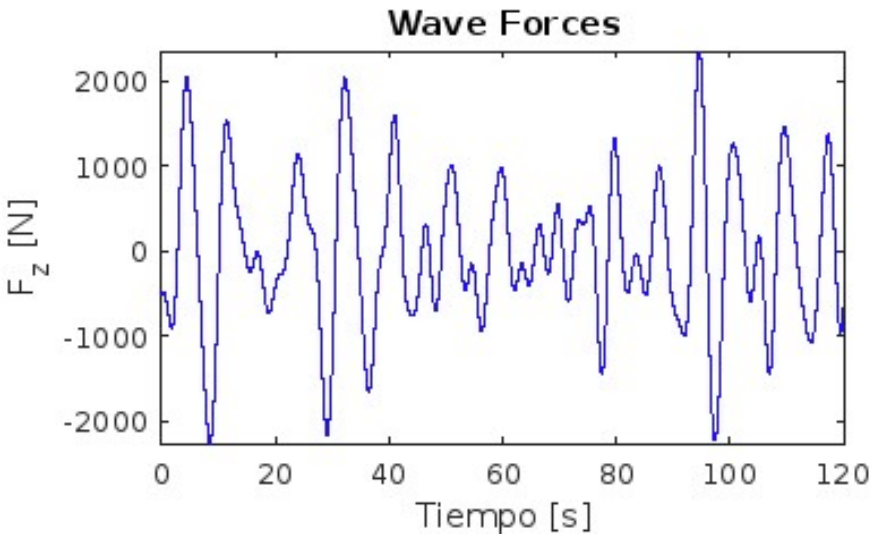

Figura 3.2: Gráfica de fuerzas de la ola simple

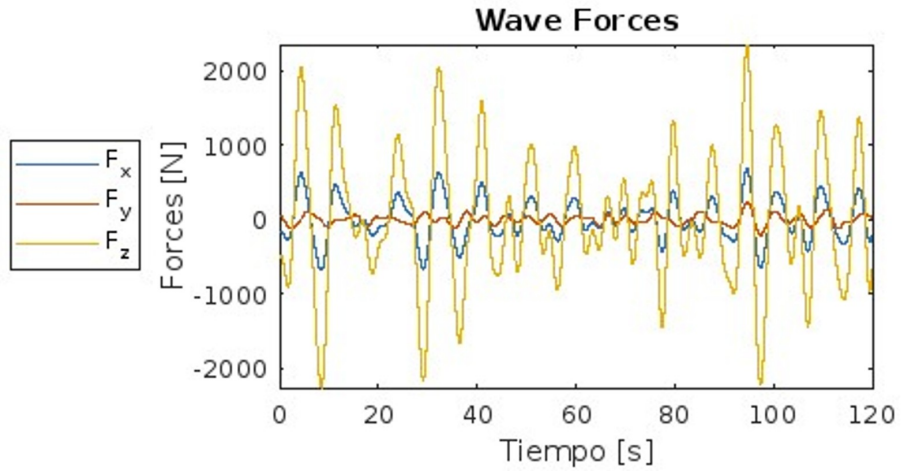

Figura 3.3: Gráfica de fuerzas de la ola direccional

3.1.2. Modelado y simulación de la dinámica 6-DOF de la boya

En esta sección se trata el modelado del movimiento de la boya teniendo en cuenta sus seis grados de libertad, las traslaciones (*surge, sway, heave*) y las rotaciones (*roll, pitch, yaw*). Para lograr el modelado correcto de la boya se definen matrices de masa-inercia (M_{mat}), de amortiguamiento dinámico (C_{lin}) y de rigidez hidrostática (K_{lin}), las cuales junto al acoplamiento de las fuerzas de excitación existentes en la ola generada, permiten obtener los movimientos de la boya de forma realista.

Primero se han definido los diferentes factores que se tendrán en cuenta para poder realizar una simulación correcta de la boya; estos se pueden separar en tres grupos: los de posición, los de orientación y los de velocidad. Estos últimos se dividen en velocidades lineales y angulares.

$$\mathbf{P} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \text{Posición } \textit{de la boya} \qquad \nu = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad \textit{Velocidades lineales de la boya}$$

$$\Theta = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \quad \text{Ángulos } \textit{de Euler} \qquad \omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad \textit{Velocidades angulares de la boya}$$

Después se realiza un bucle de integración para poder calcular el estado en el siguiente instante temporal de la boya. A partir de la ecuación diferencial se usa el método de Euler hacia adelante para aproximar la evolución continua del sistema mediante pasos discretos.

$$M\ddot{\eta}(t) + C_{lin}\dot{\eta}(t) + K_{hydro}\eta(t) = F_{wave}(t)$$

Ecuación 3.9: Ecuación diferencial 6-DOF

$$x_{k+1} = x_k + \Delta t \cdot f(x_k, t_k)$$

Ecuación 3.10: Aproximación de Euler

3.1.2.1. Cinemática

Se ha definido la matriz de rotación cuerpo-Tierra, la cual permite modificar el sistema de referencia del centro de gravedad de la boya a los ejes de la Tierra. Se convierten los ángulos de Euler $[\phi, \theta, \psi]^T$ (*roll, pitch, yaw*) en una matriz 3x3.

$$R(\phi, \theta, \psi) = R_z(\psi) \cdot R_y(\theta) \cdot R_x(\phi)$$

Donde las matrices de rotación son las siguientes:

- Rotación alrededor del eje x (roll):

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

- Rotación alrededor del eje y (pitch):

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

- Rotación alrededor del eje z (yaw):

$$R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Una vez obtenida la matriz de rotación cuerpo-Tierra se han calculado las siguientes 2 ecuaciones:

- **Ecuación de velocidad de traslación:** calcula el desplazamiento de la boya a partir de su orientación actual (ϕ, θ, ψ) y de la velocidad lineal.

$$\dot{\mathbf{p}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R(\phi, \theta, \psi) \cdot \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

Ecuación 3.11: Ecuación de velocidad de traslación

- **Ecuación de velocidad orientación:** calcula la nueva orientación de la boya a partir de su orientación actual (ϕ, θ) y de la velocidad angular.

$$\dot{\mathbf{\Theta}} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = T_{eul}(\phi, \theta) \cdot \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

Ecuación 3.12: Ecuación de velocidad de orientación

Donde:

$$\mathbf{T_{eul}} = \begin{bmatrix} 1 & \sin\phi \cdot \tan\theta & \cos\phi \cdot \tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi \cdot \sec\theta & \cos\phi \cdot \sec\theta \end{bmatrix}$$

3.1.2.2. Dinámica

La posición y velocidades de la boya se definen de la siguiente manera:

$$pos = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix}$$

$$vel = \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix}$$

Las fuerzas y momentos de la ola se definen como:

$$F_{wave} = \begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix}$$

Se pueden calcular la fuerza total (F_{net}) que se ejerce sobre la boya, sumando la fuerza hidrostática (F_{hydro}) junto a la de la ola (F_{wave}).

$$F_{hydro} = -(F_a - F_r)$$

Donde:

- F_a es la fuerza de amortiguamiento de la boya.

$$F_a = -C_{lin} * vel$$

- F_h es la fuerza de restauración de la boya.

$$F_h = -k_{lin} * pos$$

Y la fuerza total es:

$$F_{net} = F_{hydro} + F_{wave}$$

La suma de las fuerzas que ejercen sobre la boya permiten calcular las aceleraciones que sufre la boya (acc).

$$acc = {M_{mat}}^{-1} \cdot F_{net}$$

3.1.2.3. Funciones de la implementación MATLAB asociadas

Los modelos mencionados anteriormente se han utilizado en la implementación MATLAB en las siguientes partes del código:

- **6-DOF Buoy State Simulation, Simulate a complete IMU**: en estas dos partes del código se han utilizado los fundamentos explicados anteriormente.
- ***eul2rotm***: en esta función se genera la matriz R para poder transformar el sistema de referencia del cuerpo a la Tierra.

3.1.2.4. Gráficas generadas

En la simulación del movimiento de la Boya se han generado las siguientes gráficas, en las cuales se pueden observar los movimientos dinámicos de la boya frente a las fuerzas generadas por las olas.

Analizando la primera gráfica de la figura 3.4 se puede observar como la boya sigue el movimiento vertical de la ola junto con algunos desplazamientos laterales de aproximadamente 20 metros, tanto en el eje x como en el eje y. En cuanto a la segunda gráfica, se observan los diferentes ángulos de la boya (roll, pitch, yaw) a lo largo de la simulación, llegando a inclinarse con ángulos de hasta 50^{o} aproximadamente sobre el eje x, y en cuanto al yaw se pueden observar una serie de ángulos que representan el cabeceo de la boya siguiendo las subidas y bajadas provocadas por el oleaje modelado.

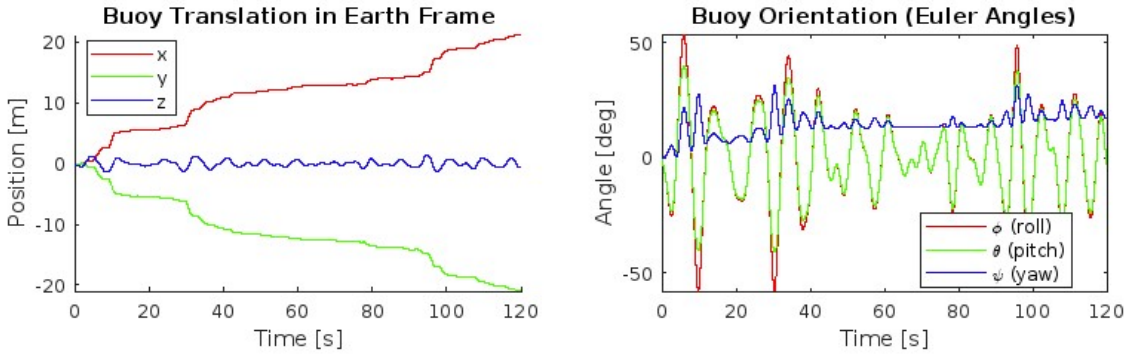

Figura 3.4: Gráficas de la dinámica 6-DOF con la ola simple

En cuanto a las gráficas de la figura 3.5, en la primera gráfica se puede observar un movimiento más realista de la boya siguiendo las corrientes de la ola, generadas por las fuerzas laterales de la misma, lo que produce ese movimiento armónico, llegando a desplazar

la boya 20 metros por cada eje lateral. Por otro lado, en la segunda gráfica se puede observar una salida idéntica a la de la simulación con olas simples, lo cual tiene sentido dado que la ola generada en ambos casos es la misma a excepción de las fuerzas laterales para la simulación direccional.

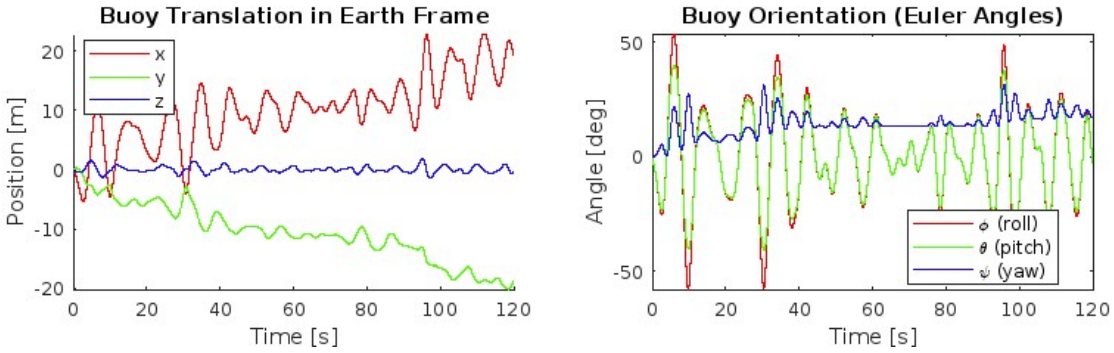

Figura 3.5: Gráficas de la dinámica 6-DOF con la ola direccional

En conclusión, como se pueden observar en las gráficas de la figura 3.4 y de la figura 3.5, la diferencia que hay las dinámicas de la boya frente a una ola simple y otra direccional, es que en la direccional se genera un movimiento más realista, llegando a simular las corrientes generadas por las olas, como se puede observar en el movimiento de la boya respecto a los ejes laterales x e y, la boya tiene un movimiento que va y viene, teniendo un desplazamiento aproximado de 10 metros para el eje x y un desplazamiento menor, de unos 3 metros para el eje y. Esto se debe a la aparición de fuerzas laterales en la generación de la ola.

3.1.3. Modelado y simulación de la IMU

Una vez se ha modelado el movimiento de la boya, se modelan los datos de la IMU, en este caso se simula una IMU de 9-DOF con un acelerómetro, un giroscopio y un magnetómetro. Para simular estos datos se calcula la aceleración, la velocidad angular y el campo magnético en el marco de la boya, añadiendo valores de ruido y sesgo para obtener un mayor realismo en el resultado de la simulación.

3.1.3.1. Parámetros de ruido y sesgo

Se han definido los siguientes valores de ruido y sesgo para los resultados de la simulación de la IMU:

- **Desviaciones estándar del ruido gaussiano:** indica el valor de ruido que se añade a cada componente, se aplica al acelerómetro, giroscopio y magnetómetro.

$$accelNoiseStd = 0{,}05\frac{m}{s^2}$$

$$gyroNoiseStd = 0{,}001\frac{rad}{s}$$

$$magNoiseStd = 0{,}002G$$

- **Vectores constantes de sesgo:** simulan los errores sistemáticos de cada sensor.

$$accelBias = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \left(\frac{m}{s^2}\right)$$

$$gyroBias = \begin{bmatrix} 0{,}01 \\ -0{,}01 \\ 0{,}01 \end{bmatrix} \quad \left(\frac{rad}{s}\right)$$

$$gyroBias = \begin{bmatrix} 0{,}01 \\ -0{,}01 \\ 0{,}01 \end{bmatrix} \quad (Gauss)$$

- **Campo magnético de la Tierra en coordenadas ENU:** se usa para generar la señal del magnetómetro.

$$EarthMagField\_ENU = \begin{bmatrix} 0{,}3 \\ 0 \\ 0{,}4 \end{bmatrix} \quad (Gauss)$$

3.1.3.2. Simulación del acelerómetro

Primero se calcula la aceleración de la boya a lo largo de toda la simulación de dos formas diferentes dependiendo del instante actual de la simulación.

- Para $k = 1$: se usa diferencia hacia delante

$$a = \frac{vel(:, k+1) - vel(:, k)}{\Delta t}$$

- Para $k \neq 1$: se usa diferencia centrada

$$a = \frac{vel(:, k+1) - vel(:, k-1)}{2 \cdot \Delta t}$$

Luego se calcula la aceleración de la gravedad, cogiendo como referencia el eje de la boya. Esto se consigue usando la matriz (R) calculada previamente en el modelado del movimiento de la boya.

$$g_{body} = R \cdot g_{earth}$$

Después se calcula el valor del acelerómetro mediante la diferencia de la aceleración de la boya contra la aceleración de la gravedad en el eje referencial de la boya y se suman los valores de ruido y sesgo para obtener un resultado más realista..

$$accel = a - g_{body} + accelBias + accelNoiseStd$$

Finalmente se calcula el valor del acelerómetro respecto al eje-Tierra, usando la matriz R traspuesta (R^T) y sumando el valor de la gravedad respecto al mismo eje de referencia.

$$accel_w = R^T \cdot accel + g_{earth}$$

3.1.3.3. Simulación del giroscopio

Para calcular el valor del giroscopio se usa el valor de la velocidad angular y se suman los valores de ruido y sesgo para dar realismo al resultado.

$$gyro = \omega + gyroBias + gyroNoiseStd$$

3.1.3.4. Simulación del magnetómetro

Para calcular el valor del magnetómetro se ha transformado el campo magnético de la tierra ($EarthMagField\_ENU$) al eje-cuerpo y se le han sumado los valores de sesgo y ruido.

$$mag = R \cdot EarthMagField\_ENU + magBias + magNoiseStd$$

3.1.3.5. Gráficas generadas

En estas gráficas se puede observar el comportamiento simulado de la IMU situada en la boya frente a las olas generadas. En estas se puede observar la salida de los tres parámetros de la IMU, el acelerómetro, el giroscopio y el magnetómetro.

En las figuras 3.6 y 3.7 se pueden observar unos valores idénticos en el giroscopio, dado que el cálculo de este depende de los valores de los momentos generados por la ola, y estos son idénticos para ambos tipos de olas. Para el magnetómetro también se pueden apreciar dos gráficas idénticas, esto se debe a que los ángulos de Euler también son iguales para ambos oleajes. Por otro lado, el valor de las aceleraciones se ve mínimamente modificado en el caso de las olas direccionales, dado que aparecen las fuerzas laterales del oleaje, lo que produce un cambio en las mediciones de las aceleraciones en los ejes laterales.

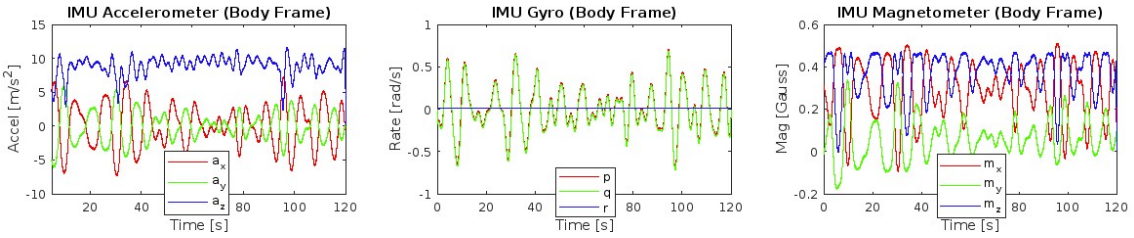

Figura 3.6: Datos de la IMU de la boya con la ola simple

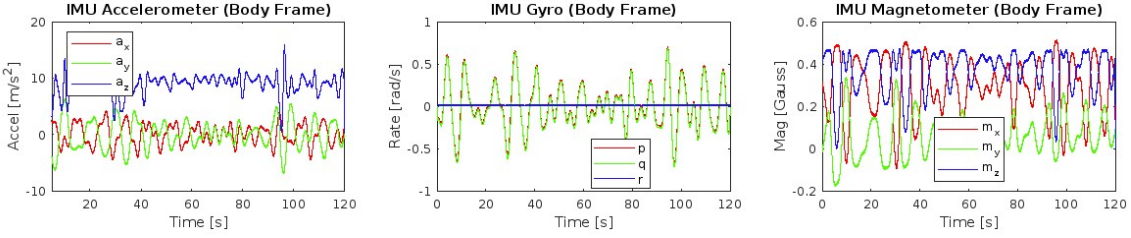

Figura 3.7: Datos de la IMU de la boya con la ola direccional

3.2. Estimación del oleaje

Para poder estimar el oleaje, se ha usado como base la simulación del acelerómetro, a partir de la cual se calcula el desplazamiento vertical de la boya. El movimiento se estima tanto en el 'eje-Tierra', en el cual se tiene en cuenta el eje vertical según el eje de la tierra, como en el 'eje-cuerpo', donde el movimiento vertical se calcula dependiendo de la orientación del cuerpo, en este caso la boya definida. Con el desplazamiento vertical obtenido, se calculan las estimaciones de la ola generada, para calcular la altura significativa se usa un análisis pico-valle y el periodo se estima mediante una FFT.

En la figura 3.8 se puede observar un diagrama de bloques con los diferentes pasos seguidos para realizar la estimación de la altura y periodo del oleaje.

Figura 3.8: Diagrama de bloques para estimar Hs y Tp

3.2.1. Filtrado

Para realizar la estimación se han usado dos tipos de filtros, dependiendo del tipo de ola sobre la que se realice la estimación.

Mediante este proceso de filtrado se consiguen aislar los componentes oscilatorios que más convienen para realizar la estimación y también se logra eliminar la deriva en baja frecuencia.

Se ha utilizado un filtro Butterworth, el cual es un filtro IIR (Infinite Impulse Response) que ofrece una atenuación suave y monótona hacia las bandas eliminadas. Este filtro se basa en las siguientes funciones y ecuaciones:

- **Función de transferencia:**

$$\mid H(jw) \mid = \frac{1}{\sqrt{1 + \left(\frac{\omega}{\omega_c}\right)^{2N}}}$$

Ecuación 3.13: Función de transferencia del filtro Butterworth

Donde:

- N : es el orden del filtro y determina la pendiente de atenuación fuera de la banda aceptada.
- $\omega_c(rad/s)$: define el punto hasta el cual la ganancia del filtro es prácticamente constante.

Los polos se calculan resolviendo la siguiente ecuación:

$$1 + \left(j \cdot \left(\frac{\omega}{\omega_c}\right)\right)^{2N} = 0$$

- **Ecuación de diferencias:**

$$y[n] = -\sum_{k=1}^{N} a_k \cdot y[n-k] + \sum_{k=0}^{N} b_k \cdot x[n-k]$$

Ecuación 3.14: Ecuación de diferencias del filtro Butterworth

Los dos tipos de filtro utilizados han sido los siguientes:

- **Filtro paso alto:** este filtro se ha usado para filtrar las aceleraciones generadas sobre las olas simples. Se usa este tipo de filtrado porque el objetivo es simplemente eliminar la deriva de baja frecuencia, sin llegar a modificar el resto del oleaje. Se ha utilizado un filtro Butterworth de segundo orden con la frecuencia de corte en 0,1 Hz, se ha seleccionado esta frecuencia de corte para eliminar las componentes de baja frecuencia, como podría ser la deriva, que se introducen durante el proceso de integración que permite obtener el desplazamiento a partir de la aceleración medida por la IMU. Este corte también permite el paso de la banda de frecuencias en las que se genera el oleaje, las cual viene definida por el periodo.
- **Filtro paso banda:** este filtro se ha usado para el filtrado de las aceleraciones generadas sobre las olas direccionales. Con este oleaje más complejo, aparte de la deriva de baja frecuencia, también puede aparecer ruido en alta frecuencia, por lo que es necesario realizar un filtrado para eliminar estos dos tipos de señales indeseadas, quedándonos con la banda de interés que contiene la energía de la ola. En este caso también se aplica un filtro Butterworth de segundo orden pero entre las frecuencias de 0,1 a 1 Hz, se ha escogido usar un filtro paso banda con estas frecuencias, dado que al incluir las componentes laterales del oleaje aparecen señales de alta frecuencia sobre el movimiento de la boya, y de esta forma se consiguen limitar estas señales.

El proceso de filtrado se realiza en un inicio para filtrar la aceleración y una segunda vez sobre el desplazamiento vertical obtenido. Posterior al primer filtrado se recortan los primeros 25 segundos de la señal para que se trabaje únicamente con una estimación estable de la señal de aceleración. Después de esta eliminación se procede al proceso de doble integración, que se explicará en el próximo apartado y una vez finalizado este, se eliminara la deriva lineal del desplazamiento vertical y se eliminarán los transitorios tanto iniciales y finales que aparecen durante las fases de adaptación y atenuación del filtrado.

3.2.2. Integración

El objetivo de este paso es obtener el desplazamiento vertical de la boya a partir de la aceleración en los diferentes instantes de tiempo. Para lograr esto, es necesario integrar dos veces la aceleración, una primera para obtener la velocidad y una segunda integración sobre la velocidad para obtener el desplazamiento vertical $z(t)$.

1. Obtención de la velocidad:

$$v[n+1] = v[n] + a[n] \cdot \Delta t$$

Ecuación 3.15: Integración de la aceleración

2. Obtención del desplazamiento vertical:

$$z[n+1] = z[n] + v[n+1] \cdot \Delta t$$

Ecuación 3.16: Integración de la velocidad

Durante el proceso de integración puede haber una acumulación de error debido a la aparición de pequeños offsets en la aceleración o del propio ruido de la señal. Una vez realizada la doble integración, se extrae el offset del desplazamiento vertical por mínimos cuadrados para poder eliminar la deriva generada.

3.2.3. Estimación de Hs

Para estimar la altura significativa de la ola se usa la señal del desplazamiento vertical previamente calculada siguiendo los siguientes pasos:

1. **Detección de los picos y valles de la señal:** se buscan los máximos y mínimos de la señal.
2. **Cálculo de las alturas de la ola:** se calcula la altura midiendo la diferencia entre el pico y el valle siguiente.
3. **Cálculo de la altura significativa de la ola:** se ordenan las alturas calculadas en el paso anterior de mayor a menor y se realiza el promedio del primer tercio de alturas, tal y como se comenta en el estudio de Huang et al. (2016), donde indican que la altura significativa del oleaje es aproximadamente igual a la media del primer tercio de las olas más altas.

3.2.4. Estima de Tp

Para calcular el periodo dominante de la ola estimada se usa un análisis espectral mediante FFT.

1. **Cálculo de la transformada discreta de Fourier** $X(k)$
2. **Normalizar la amplitud**

$$P_1(f_k) = \begin{cases} \frac{|X(k)|}{L} & , \quad k = 0 \quad o \quad k = L/2 \\ \frac{|X(k)|}{L} & , \quad en\ otro\ caso \end{cases}$$

   donde:

$$f_k = \frac{k}{L \cdot \Delta t}$$

3. **Obtener la frecuencia máxima de P1 y calcular el periodo a partir de la misma:**

$$T_p = \frac{1}{f_{max}}$$

3.2.5. Funciones de la implementación MATLAB asociadas

Los pasos mencionados anteriormente para realizar la estimación del oleaje se han realizado en la parte del código **Wave Height & Period Estimation From IMU** y para realizar la estimación de la altura y periodo se han implementado las funciones ***estimateWaveHeightPeriod*** y ***simpleFFT*** para realizar la transformada discreta de Fourier.

3.2.6. Resultados de la estimación

En esta sección se realizará un análisis de las gráficas comparativas que se generan al finalizar la estimación del oleaje.

3.2.6.1. Comparativa de oleajes estimado vs real

En esta gráfica se compara el oleaje generado con la estimación respecto al eje-Tierra, esta comparación permite observar la altura que estima la boya a lo largo del tiempo de simulación.

En la figura 3.9 se puede observar una estimación de alturas bastante precisa, aunque en algunos casos se estima por debajo de la altura real de la ola. En cambio, en la figura 3.10 la estimación llega a sobrepasar mínimamente la altura real del oleaje y la altura estimada parece ser algo menos fiel a la ola generada con respecto a la simulación con la ola simple.

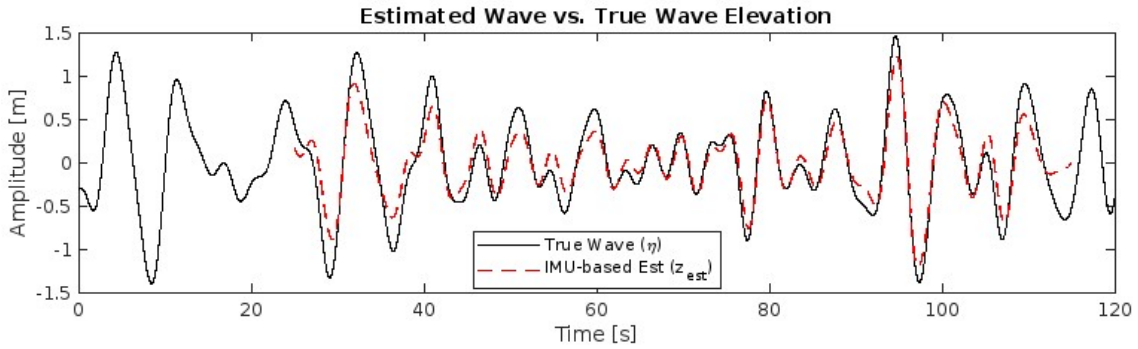

Figura 3.9: Estimación de la altura con la ola simple

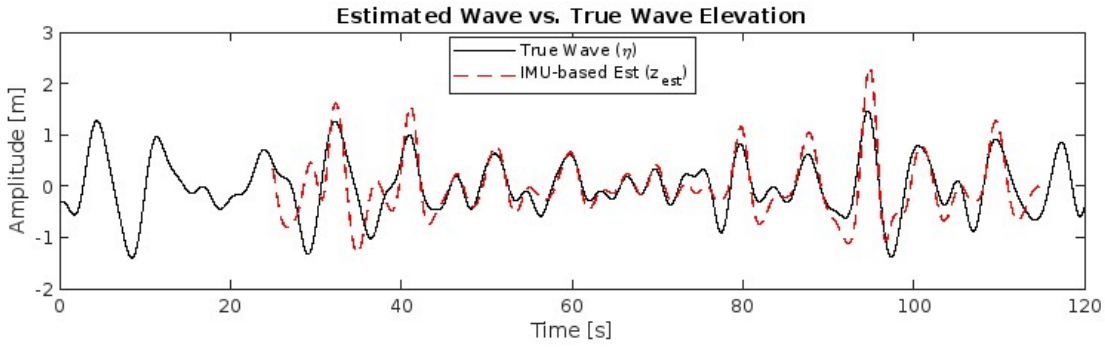

Figura 3.10: Estimación de la altura con la ola direccional

3.2.6.2. Gráfica comparativa del espectro de frecuencias (FFT)

En la siguiente gráfica se observa la comparativa del espectro de frecuencias, el de la ola generada contra el de las frecuencias de la ola estimada por la boya.

Como ya se ha observado anteriormente, para ambas estimaciones, tanto para las olas simples como direccionales, en esta simulación se ha estimado el mismo periodo. Observando las gráficas de las figuras 3.11 y 3.12, se puede apreciar un pico en la frecuencia $F = \frac{1}{T} = \frac{1}{7{,}69s} = 0{,}13\ Hz$ para el espectro de la ola generada y en el espectro de la estimación, usando el periodo estimado, se obtiene una frecuencia de $F = \frac{1}{T} = \frac{1}{6{,}92s} = 0{,}145\ Hz$. Comparando el espectro de frecuencias de la estimación entre la ola simple o la direccional se pueden observar algunas diferencias entorno a frecuencias de aproximadamente 0,25 Hz.

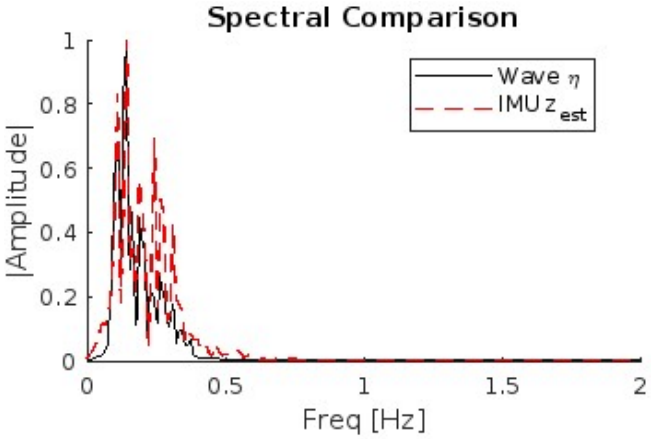

Figura 3.11: Espectro de frecuencias con la ola simple

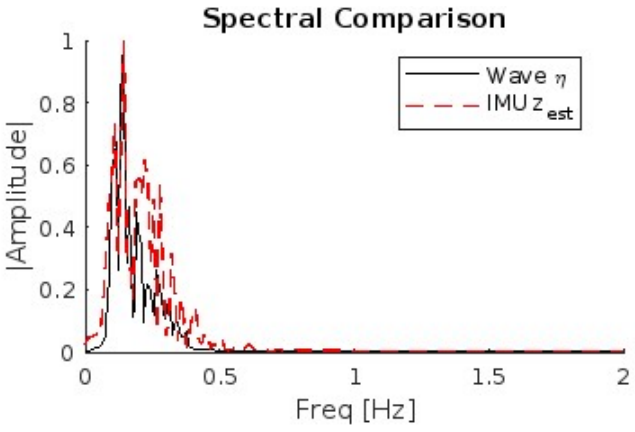

Figura 3.12: Espectro de frecuencias con la ola direccional

3.2.6.3. Comparativa del oleaje estimado

En la figura 3.13 se presenta la gráfica en la que se puede observar la comparativa entre la ola generada y las estimaciones respecto al eje-Tierra tanto para la ola simple como la ola direccional. Las estimaciones para los dos tipos de olas siguen, en su mayoría, el movimiento de la ola.

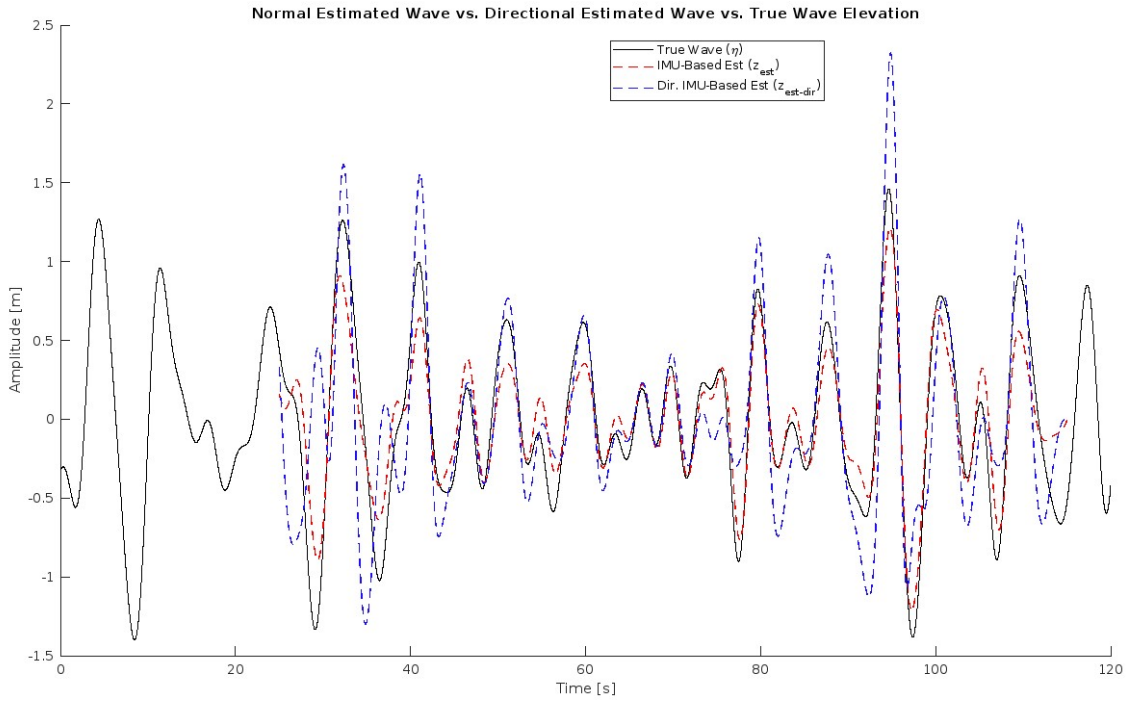

Figura 3.13: Comparativa estimaciones vs ola real

3.2.6.4. Comparativa eje-cuerpo vs eje-Tierra

En las figuras 3.14 y 3.15 se muestran dos gráficas, en las cuales se puede ver la diferencia de la estimación de la ola entre el eje-cuerpo y el eje-Tierra.

En estas figuras se puede observar bastante similitud entre las estimaciones entre ambos ejes de referencia, aunque con la estimación respecto al eje-Tierra se realiza un mejor seguimiento de la ola real generada.

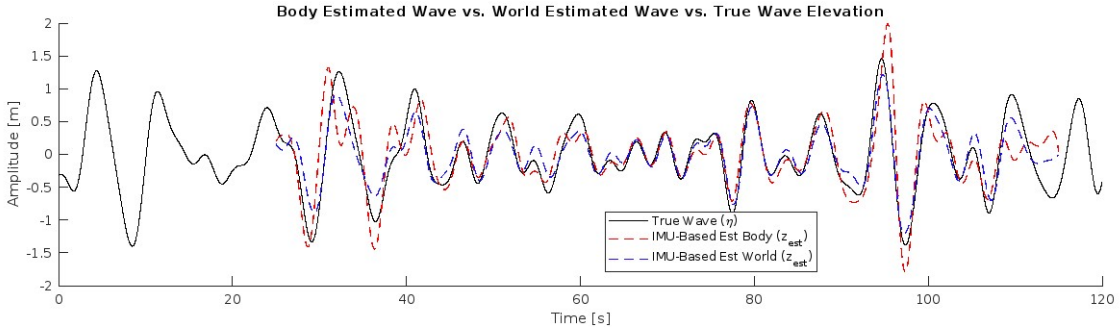

Figura 3.14: Comparativa eje-cuerpo vs eje-Tierra con la ola simple

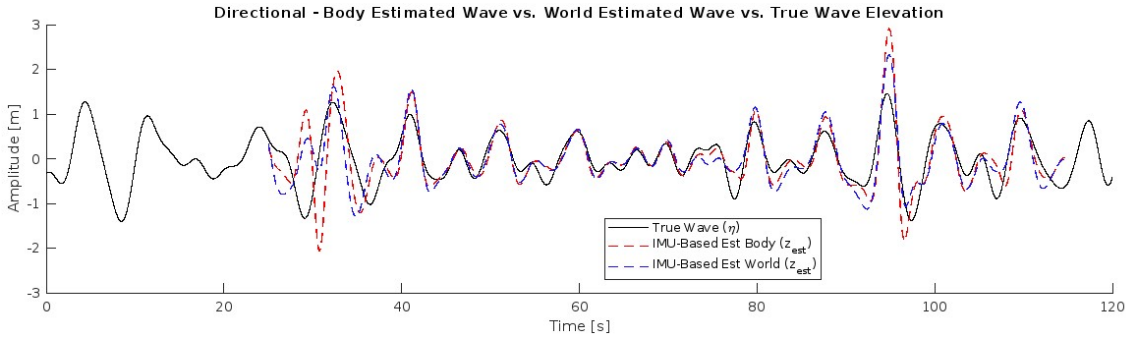

Figura 3.15: Comparativa eje-cuerpo vs eje-Tierra con la ola direccional

3.2.6.5. RMSE

En esta figura se puede visualizar un gráfico de barras, el cual muestra el error cuadrático medio entre las 4 estimaciones posibles (respecto al cuerpo y al mundo para los dos tipos de olas).

En la figura 3.16 se puede observar el resultado del error cuadrático medio calculado para cada tipo de ola tanto para el eje-cuerpo como para el eje-Tierra respecto a la ola generada. Como se puede observar en el gráfico, para esta simulación, en el caso de las olas simples, se puede apreciar una disminución del rmse en la estimación realizada respecto al eje-Tierra, en cambio, para las olas direccionales, se obtiene un menor valor del error en el eje-cuerpo. Esto se debe a que para el caso de las olas simples, al existir únicamente el movimiento vertical de la ola, para extraer el desplazamiento respecto al eje-Tierra, se ha de basar únicamente en la medición de la aceleración en un eje, ele vertical. En cambio, para el caso de olas direccionales, se ha de transformar tres componentes de aceleración en el eje-cuerpo respecto al eje-Tierra, lo que provoca la aparición de más ruido y errores durante el proceso de transformación.

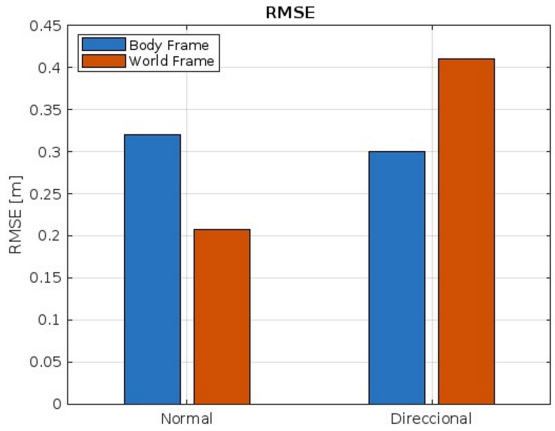

Figura 3.16: RMSE

3.2.6.6. Estimación del oleaje

En este apartado se presenta la salida generada a partir de las simulaciones explicadas anteriormente. Esta salida se compone tanto por la estimación general de la simulación, compuesta por la altura significativa como el periodo pico, tanto para la medición respecto

eje-cuerpo como eje-Tierra, como para cada caso de ola, tanto simples como direccionales, un ejemplo de estimación se puede observar en la figura 3.17.

```
------------------------------------------------
Wave Height & Period (True vs. IMU-based Estimate)
True Wave Height: 1.84 m
True Wave Period:  7.69 s
Est Wave Height (body frame):   1.73 m
Est Wave Period (body frame):   6.92 s
Est Wave Height (world frame):   1.39 m
Est Wave Period (world frame):   6.92 s
------------------------------------------------
------------------------------------------------
Directional Wave Height & Period (True vs. IMU-based Estimate)
True Wave Height: 1.84 m
True Wave Period:  7.69 s
Est Wave Height (body frame):   2.19 m
Est Wave Period (body frame):   6.92 s
Est Wave Height (world frame):   1.88 m
Est Wave Period (world frame):   6.92 s
------------------------------------------------
```

Figura 3.17: Estimaciones realizadas

En las estimaciones de la figura 3.17 se puede observar una mejor estimación en la simulación con la ola direccional, llegando a estimar una altura de 2.19 metros respecto al eje-cuerpo y una altura de 1.88 metros respecto al eje-Tierra, siendo la altura real de la ola generada de 1.84 metros. Por otro lado, la estimación para la ola simple ha tenido una mejor estimación de la altura teniendo como referencia el eje-mundo con 1.79 metros, en cambio, para el eje-Tierra ha estimado una altura de 1.39 metros.

Respecto al periodo estimado, se ha estimado para todos los casos un periodo de 6.92 segundos cuando el periodo real de la ola ha sido de 7.69 segundos.

3.3. Parámetros y condiciones

La inicialización de los diferentes parámetros y matrices mencionados durante este capítulo se puede ver reflejada en las tablas del anexo A. Para la elección de estos se ha tenido en cuenta una boya pequeña, aunque son escalables a cualquier tipo de boya, así como los parámetros de simulación, también son modificables para poder simular durante más o menos tiempo dependiendo del modelado objetivo que se desea.

Por otro lado, la implementación en MATLAB puede ejecutarse con un oleaje modelado de máximo 6 metros pico a pico y con un periodo de 10 segundos. También se debería implementar el sistema con otro modelo de oleaje relativo, para que este sea capaz de calcular fuertes excitaciones del oleaje.

3.4. Conclusiones

El uso de la metodología MBD ha facilitado en gran medida el proceso tanto de desarrollo como experimental de esta parte del trabajo. Gracias a los modelos implementados se ha podido comprobar el funcionamiento del algoritmo desarrollado de una forma sencilla y eficaz.

Por otro lado la posibilidad que ha ofrecido MATLAB para visualizar los modelos generados y los resultados del algoritmo, ha facilitado mucho el análisis de todo el sistema, pudiendo observar aquellos puntos donde no se obtenía una salida esperada del sistema y de esta forma, poder mejorar la implementación.

Finalmente, analizando los resultados obtenidos, se puede concretar que se ha obtenido un algoritmo de caracterización del oleaje preciso y eficaz frente a un amplio rango de oleajes.

Capítulo 4

Implementación

Este capítulo se centrará en la explicación de las diferentes implementaciones realizadas para el entorno físico donde se realizarán las pruebas del algoritmo. Estas implementaciones están disponibles en la carpeta de Google Drive (Gómez, 2025) y se distribuyen bajo la licencia GPL v3.

En la primera sección se introducirá el entorno de pruebas diseñado, el cual se compone de un sistema mecánico que permite la simulación de un movimiento vertical y de la placa utilizada, que es de Espressif, para obtener las mediciones de la aceleración, utilizando una IMU de Invensense.

Seguidamente se comentarán los diferentes desarrollos hechos, entre los que se encuentra un proyecto implementado en el entorno de Zephyr Project, para ejecutarlo en la placa de Espressif y que permitirá obtener los datos de la aceleración. También se ha implementado en Python el algoritmo de estimación desarrollado en MATLAB para poder verificar la eficiencia del sistema físico. Y, finalmente se ha diseñado y desarrollado una librería en C que permite la estimación del oleaje en sistemas embebidos.

4.1. Sistema de pruebas y placa utilizada

En esta sección se comentará el sistema mecánico diseñado por Javier, colaborador de este trabajo, para poder obtener un movimiento sinusoidal, con la finalidad de poder realizar las pruebas necesarias de las implementaciones que se comentarán a lo largo de este capítulo. El sistema utilizado ha sido el que se puede observar en la figura 4.1 , el cual ha sido programado para realizar un movimiento vertical de 30 cm pico a pico y con un periodo de 5 segundos, haciendo un total de 22 periodos pudiendo simular el movimiento durante 110 segundos.

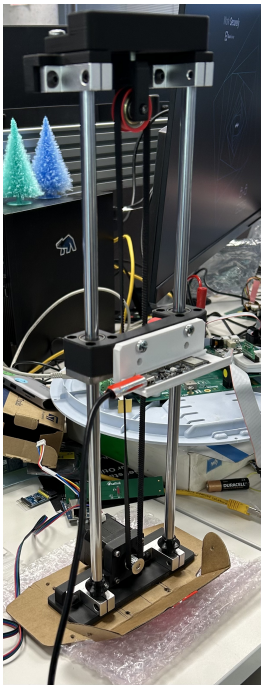

Figura 4.1: Estructura mecánica

El cual está formado por dos varillas verticales, un motor paso a paso, una correa, una polea y una base fabricada con impresión 3D para que la placa con la IMU esté lo más estable posible.

Por otro lado, la placa utilizada para realizar las pruebas ha sido una esp32c3-devkit-rust-1 (figura 4.2), la cual contiene una IMU ICM-42670-P de InvenSense (figura 4.3).

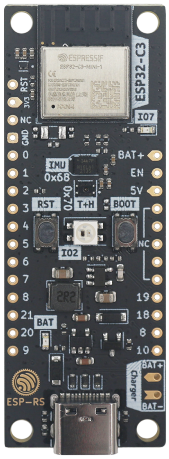

Figura 4.2: esp32c3-devkit-rust-1 (Espressif, 2020)

Figura 4.3: ICM-42670-P (TDK_InvenSense, 2024)

Algunas características de la IMU utilizada son las siguientes:

- Contiene seis ejes de medida, juntando un acelerómetro y un giroscópio.
- Tiene un bajo consumo energético de 0,55 mA
- Disponibilidad de 12,5 Mhz $I3C^{SM}$, 1 MHz I^2C y 24 MHz SPI

4.2. Recogida de datos IMU usando Zephyr RTOS

4.2.1. Introducción a Zephyr RTOS

Zephyr Project es un sistema operativo en tiempo real (RTOS), está orientado a tareas en el ámbito del IoT, enfocado a aquellos dispositivos con recursos limitados. Está mantenido y desarrollado por la propia comunidad, bajo el amparo de la *Linux Foundation*, por lo que está siendo actualizado continuamente. También ofrece soporte nativo para gran cantidad de placas y periféricos gracias a la utilización del *device tree*.

Este sistema operativo ha sido el seleccionado para este proyecto, dada su alta portabilidad entre plataformas distintas y la facilidad de integrar los sensores necesarios.

4.2.2. Estructuración del proyecto

La estructura de una aplicación básica en Zephyr RTOS es la siguiente:

```
├── CMakeLists.txt
├── boards
├── build
├── prj.conf
└── src
```

El proyecto se estructura de la siguiente manera:

- **CMakeLists.txt:** este fichero sirve como punto de partida para el generador del sistema de construcción y permite personalizar el comportamiento del mismo.

- **boards:** esta carpeta, que es opcional, permite añadir ficheros de configuración para diferentes placas. Al escoger una placa para compilar el proyecto, Zephyr aplicará automáticamente los parámetros definidos en el archivo "<nombre_placa>.overla", asegurando que se compile y ejecute correctamente en el hardware seleccionado. Estos ficheros permiten modificar los periféricos o pines a usar por la aplicación.
- **build:** es el directorio donde se guarda toda la compilación de la aplicación.
- **prj.conf:** este fichero de configuración permite la activación y configuración de los diferentes módulos, componentes y librerías de las que depende la aplicación.
- **src:** este directorio incluye los ficheros fuente de la aplicación, en este caso solo existe uno llamado "main.c". Este directorio podría llamarse con cualquier otro nombre, dado que el directorio principal se ha de configurar en el fichero "CMakeLists.tx".

4.2.3. Implementación de la aplicación

En una primera fase, se ha diseñado un sistema de pruebas sencillo, donde la captura de los datos de la IMU se ha tratado en el propio nodo y la implementación del algoritmo de estimación se ha realizado en un computador externo, mediante una implementación en Python. Esto ha simplificado el desarrollo del trabajo en el entorno físico y ha permitido realizar una comparación de los resultados obtenidos contra los del código en MATLAB.

Como ya se ha comentado, en esta implementación se ha buscado obtener los datos de la aceleración de la IMU, para poder mandarlos en *raw* a través del puerto serie del ordenador y poder procesarlos en una aplicación externa al nodo para estimar la altura y periodo a partir de las mediciones de la IMU.

El código se divide en las siguientes partes:

4.2.3.1. Librerías utilizadas

Se han utilizado las siguientes librerías:

- **zephyr/drivers/sensor.h:** este driver de Zephyr ha permitido utilizar la API genérica de sensores, que facilita el uso de los sensores con cualquier placa.
- **zephyr/kernel.h:** esta librería ha permitido realizar esperas en el código a partir del reloj del sistema.
- **zephyr/sys/byteorder.h:** esta librería se ha utilizado para convertir los datos obtenidos del acelerómetro a un formato correcto, mediante la llamada a la función *sys_get_be16()*, la cual permite convertir una entero de 16 bits en formato *Big Endian* a la arquitectura local del procesador.
- **icm42670.h**, **imu/inv_imu_driver.h** y **imu/inv_imu_regmap_rev_a.h**: permiten acceder al driver específico de la IMU utilizada, para poder recoger los datos del acelerómetro correctamente.

4.2.3.2. Obtención del sensor

En esta segunda parte del desarrollo se ha utilizado el *Device Tree aliases* que permite obtener los datos del sensor definido en el *overlay* dentro del directorio *boards*, en este caso

el acelerómetro de la IMU. Mediante la macro *DEVICE_DT_GET* se crea un puntero al sensor y los datos de este se guardan en la variable *sensor*.

4.2.3.3. Función *print_accels()*

Esta función se encarga de recoger los datos de las aceleraciones en x, y, z del acelerómetro y enviarlas junto a un *timestamp (ms)* por el puerto serie para que puedan ser tratadas posteriormente.

Para realizar esto se utilizan las propias funciones, definidas en los drivers de la IMU utilizada para poder hacer el *fetch* de los datos y una vez disponibles, se hace el *get* de los mismos y se convierten en big endian mediante la función *sys_get_be16* para enviarlos por el puerto serie con el formato $timestamp, accel_x, accel_y, accel_z$.

4.2.3.4. Función *set_sampling_freq*

Esta función permite configurar la frecuencia de muestreo del sensor, para hacerlo, se hace un intento de lectura de la Output Data Rate (ODR), y en caso de que no sea posible, establece la frecuencia a 100 Hz.

4.2.3.5. Rutina principal *main*

En esta función se organiza la lógica de la implementación a partir de las funciones descritas anteriormente. La lógica seguida en esta función es la siguiente:

Algorithm 1 Rutina principal

```
1: if not device_is_ready(sensor) then
2:     print("sensor: device is not ready.")
3:     return 0
4: end if
5: set_sampling_freq(sensor)
6: while true do
7:     ret ← print_accels(sensor)
8:     if ret < 0 then
9:         return 0
10:    end if
11:    sleep(20 ms)
12: end while
13: return 0
```

Como se puede observar, se configura la frecuencia de muestreo del sensor y seguidamente se crea un bucle infinito donde se recogen muestras del acelerómetro cada 20 ms. Se ha utilizado *sleep* para realizar las esperas entre la recogida de muestras, dado que el entorno experimental no es muy complejo, ya que solo recogemos las muestras de un sensor, es una implementación suficiente. Aunque pensando en un futuro, en el caso de que se implemente un sistema más complejo donde aparezcan más sensores se debería realizar la espera mediante interrupciones.

4.3. Implementación Python para estimar Hs y Tp

En este código se ha buscado recoger los datos del acelerómetro enviados por el puerto serie para posteriormente procesarlos. El flujo del programa es el siguiente:

4.3.1. Calibración del sensor

Inicialmente, se realizan 5 segundos de calibración del sensor, en los cuales este ha de estar en su posición de reposo sobre el eje z, estos 5 segundos de calibración equivalen a 250 muestras del acelerómetro. Esta calibración permite obtener la referencia del valor de la aceleración en reposo ($bias_z$) para restarla posteriormente y obtener así la aceleración real medida por el sensor.

4.3.2. Guardar datos para realizar la estimación

En esta segunda parte del código se recogen datos del acelerómetro durante un total de 112 segundos, este tiempo se ha establecido así porque el movimiento vertical producido por la estructura mecánica está programado en un total de 22 periodos, lo que equivale a 110 segundos. Se recogen los datos de la aceleración en el eje z y del *timestamp* durante el tiempo mencionado y se guardan en dos búfers para poder tratar los datos posteriormente.

Para el caso de los datos de la aceleración que llegan en función de LSB, se han seguido los pasos necesarios mencionados en el datasheet de la IMU ICM-42670-P (TDK_InvenSense, 2021), donde se comenta que usando un *Full-Scale Range* de $\pm 16g$, se ha de aplicar un *Sensitivity Scale Factor* de $2048\frac{LSB}{g}$, por lo que en la implementación se ha aplicado la siguiente fórmula para obtener la aceleración en $\frac{m}{s^2}$:

$$accel_z = \frac{(accel_{zraw} - bias_z) \cdot g}{2048 LSB}, \qquad g = 9{,}81$$

Al obtener el *timestamp*, se coge como referencia el primero que se lee, para poder referenciarlos todos respecto a 0, para que, al momento de graficar la estimación y aceleración, esté bien la referencia de tiempo.

4.3.3. Estimación de Hs y Tp

Una vez obtenidos todos los datos de la aceleración, se ha mantenido el proceso de estimación comentado en el capítulo anterior con la simulación MATLAB. En la función *design_ bandpass()* se ha diseñado un filtro butterworth pasa banda para poder realizar el filtrado de las señales. Y en la función *estimate_ displacement()* se ha obtenido el desplazamiento vertical a partir de la aceleración, para ello, se ha realizado una doble integración filtrando la salida después de cada proceso de integración. Una vez obtenido el desplazamiento vertical se llama a la función *estimate_ wave_ properties* donde se han buscado los picos y alturas del desplazamiento para estimar H_s y T_p.

4.3.4. Gráficas y resultados

Finalmente, después de haber estimado las características del movimiento, se han graficado, tanto la aceleración medida (figura 4.4) como el desplazamiento vertical medido por la IMU (figura 4.5).

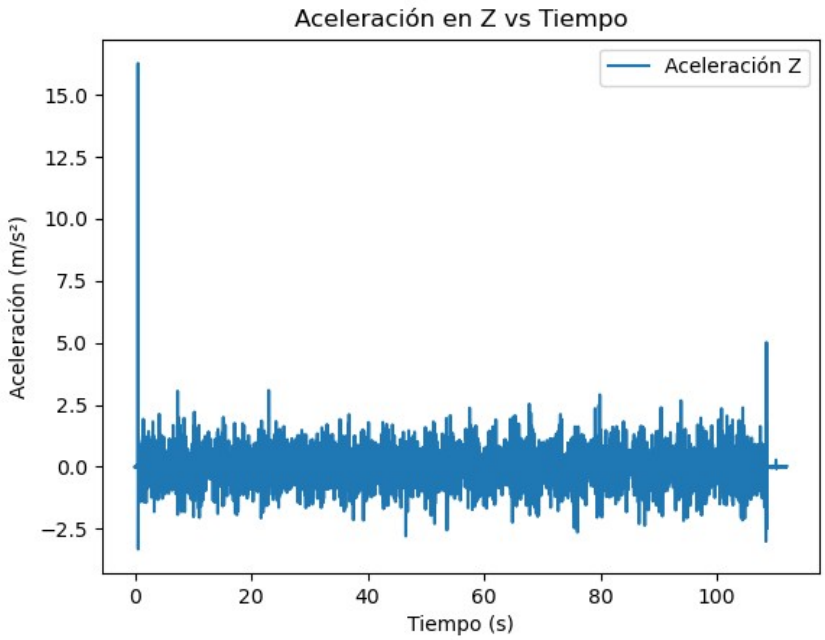

Figura 4.4: Gráfica de la aceleración

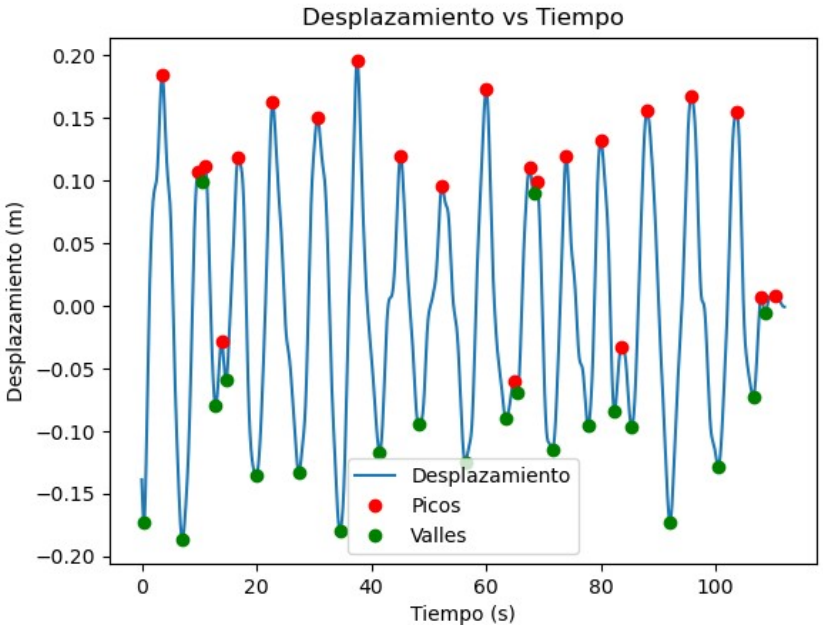

Figura 4.5: Gráfica del desplazamiento vertical

Como se pueden observar en las figuras anteriores, los resultados obtenidos tanto de aceleración como de desplazamiento no son del todo estables, aunque se haya usado como base el dispositivo mecánico con el movimiento sinusoidal. Estos se deben a pequeños errores de medida de la IMU, la cual, aunque se realice el mismo movimiento, mide aceleraciones diferentes. Aunque la salida de la estimación sí que ha tenido una precisión elevada, tal y como se puede observar en la figura 4.6.

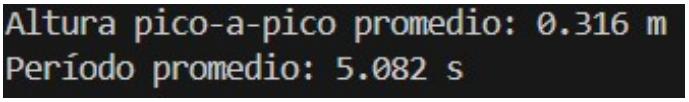

Figura 4.6: Estimación Hs y Tp

4.4. Librería de estimación en C

Se ha diseñado e implementado una librería en C que permite estimar la altura y periodo del oleaje, esta se ha diseñado de forma genérica pensando su integración en Zephyr RTOS en un futuro, se han tenido las siguientes consideraciones de diseño.

- **Macros:** se han definido una serie de macros para facilitar su configuración desde el menú de configuración del proyecto que permite Zephyr.
- **Funciones propias:** dentro de lo posible, se han implementado nuevas funciones para no tener dependencia sobre ninguna librería externa, las cuales podrían no estar soportadas por Zephyr.

Para definir los nombres de los métodos usados en la librería se ha utilizado el prefijo 'we', indicando la librería 'wave estimate'. De esta forma, cuando en un futuro se utilicen las funciones, se podrán vincular rápidamente a la librería de estimación del oleaje.

Para desarrollarla, se ha cogido como base la implementación en Python, comentada anteriormente, que permite hacer la estimación correctamente.

La librería está compuesta por los siguientes módulos:

4.4.1. Módulo de lectura de un fichero

Este módulo se ha diseñado principalmente para realizar las pruebas experimentales de la librería. Se espera que esta librería sea utilizada directamente en el nodo que realice la estimación del oleaje, por lo que esta funcionalidad no sería necesaria. Aunque se ha decidido dejarlo implementado para dar la opción a los usuarios de poder obtener los datos tanto desde un fichero en formato csv como desde los propios sensores de la placa.

Este módulo está formado por los ficheros "csv.c" y "csv.h", y en estos se incluye la lógica que permite leer un archivo con datos en formato csv y guardar estos datos, en este caso el *timestamp* y la aceleración vertical medida por el acelerómetro, para que puedan ser tratados posteriormente.

El método implementado en el fichero .c es *we_csv_read*, este recibe los siguientes parámetros de entrada:

- **filename**: es un tipo char* donde se indica la ruta del fichero con los datos en formato csv.
- **time**: es un tipo double* donde se guardarán los diferentes tiempos de medida de cada dato.
- **acc_z**: es un tipo double* donde se guardaran todas las aceleraciones leídas del fichero.

- **max_samples**: es una variable del tipo entero que indica el tamaño máximo de los búfers de time y acc_z.
- **N**: es de tipo int* y devolverá el número de muestras leídas.

4.4.2. Módulo de estimación

En este módulo se encuentra la lógica para poder caracterizar el oleaje a partir del desplazamiento vertical. Este contiene los ficheros "estimate.c" y "estimate.h" en los cuales se ha implementado el método de estima *we_wave_estimate*, al cual se le pasan los siguientes parámetros:

- **sig**: es un tipo double* que contiene el desplazamiento vertical realizado.
- **t**: es un tipo double* que contiene los *timestamps* de cada dato de desplazamiento.
- **N**: es un tipo entero que indica la cantidad de muestras que hay en los búfers de desplazamiento y *timestamps*.
- **dt**: es un tipo double que indica el intervalo de tiempo entre muestras Δt.
- **Hs**: es de tipo double* y devolverá la estimación de la altura significativa de la ola.
- **Tp**: es de tipo double* y devolverá la estimación del periodo pico.

4.4.3. Módulo de filtrado

En este módulo se ha desarrollado la lógica para realizar el filtrado de las señales, esta se encuentra en los ficheros "filter.c" y "filter.h", y está implementada en los siguientes métodos:

- *we_filter*: este método permite filtrar una señal mediante un filtro pasa banda. Los parámetros de entrada son los siguientes:
  - **in**: es un tipo double* donde se indica la señal a filtrar.
  - **out**: es un tipo double* donde se retornará la señal filtrada.
  - **N**: es un tipo entero que indica la cantidad de muestras que hay en los búfer de entrada.
  - **b**: es un tipo double* que indica los coeficientes 'b'del filtro a utilizar.
  - **a**: es de tipo double* que indica los coeficientes 'a'del filtro a utilizar.
  - **order**: es de tipo entero e indica el orden del filtro que se aplica.
  - **tmp**: es del tipo double* de tamaño N que sirve como búfer temporal.
- *we_detrend*: este método se encarga de eliminar la tendencia lineal de la señal. Los parámetros de entrada son los siguientes:
  - **data:** es del tipo double* y son los datos para eliminar la tendencia lineal.
  - **N:** es un tipo entero que indica la cantidad de muestras que hay en los búfer de entrada.

4.4.4. Rutina principal

Esta rutina está implementada en el fichero "main.c" y es donde se está desarrolla toda la lógica del programa, donde se llaman a los métodos nombrados previamente en esta sección para terminar estimando la altura y periodo de la ola. Se sigue el mismo procedimiento explicado tanto en la implementación en MATLAB como en la de Python.

4.4.5. Integración con Zephyr

La integración de esta librería en el entorno de desarrollo de Zephyr es necesaria dada la facilidad que ofrecerá este añadido al momento de querer ejecutar la aplicación sobre una placa con una IMU integrada, de forma que sea posible realizar toda la lógica a medida que se van tomando muestras.

En este caso no ha sido posible realizar la integración con el RTOS dada la complejidad que tiene el proceso y el tiempo limitado para hacerlo. Esta librería se ha diseñado de forma genérica, para que no sea necesario depender de librerías externas complejas, las cuales podrían no estar implementadas en el entorno de Zephyr.

Para poder integrar la librería en Zephyr harían falta seguir una serie de consideraciones:

- Creación de un módulo en Zephyr de la librería y conseguir su funcionamiento correcto, inicializando el fichero "CMakeLists.txt" con las macros necesarias de Zephyr.
- Modificación del código para poder portarlo al amplio repertorio de placas disponibles en Zephyr, esto se puede conseguir gracias al uso del Device Tree.
- Se deberían incluir los diferentes parámetros de configuración en el fichero de configuración del proyecto de Zephyr para facilitar la modularidad del código y de esta manera aumentar la capacidad de uso de la librería.

4.5. Resultados experimentales

Para poder estudiar la precisión y eficacia del estimador, se han generado dos códigos:

1. **data_to_csv.py:** esta implementación se encarga de guardar los datos de la aceleración del puerto serie en un fichero con formato csv, de la misma forma que se ha comentado anteriormente para la implementación del estimador, aunque en lugar de guardar los datos en un búfer, se guardan en un archivo en formato csv.
2. **wave_estimation_statistics.py**: este código se encarga de realizar la estimación de altura y periodo a partir de los datos recogidos durante las pruebas y una vez obtenidos los resultados realizar un estudio estadístico mediante el método *describe()* y mostrando dos gráficas con la altura y periodo estimados para cada uno de los archivos.

Para realizar estas pruebas se han usado datos de 33 archivos en formato csv, y los resultados han sido los siguientes:

Para la estimación de la altura, se han obtenido los resultados de la tabla 4.1.

En estos se puede observar un error medio del 10 %, siendo la media calculada de todas las pruebas 33 cm, frente al recorrido real realizado de 30 cm.

	Valor
media (cm)	30.11
Desviación estándar (cm)	4.96
Mínimo(cm)	18.92
Percentil 25 % (cm)	27.44
Percentil 50 % (cm)	30.70
Percentil 75 % (cm)	32.39
Máximo (cm)	41.37

Tabla 4.1: Tabla estadísticas altura estimada (Hs)

Según los resultados observados, podemos observar las siguientes características:

- **Valor medio:** Se ha obtenido una altura media de 30.11 cm, lo que supone una sobrestimación del 0.36 % sobre la altura real de 30 cm.

$$\frac{30{,}11 - 30}{30} \cdot 100 = 0{,}36\,\%$$

- **Desviación estándar:** tiene un valor de 4.9 cm, el 16.33 % de la altura total, lo que indica que las medidas tiene una variabilidad alta.
- **Rango intercuartílico:** la mitad central de las alturas estimadas se encuentra entre el -8.52 % y el 8 % de la altura real.
- **Valor máximo y mínimo:** el valor mínimo de la altura estimada obtenido en las pruebas ha sido de 18.92 cm y el máximo ha sido de 41.38 cm.

Para la estimación de la altura, se han obtenido los resultados de la tabla 4.2. En los resultados se puede observar un error de estimación contando la media medida de un 4.8 %, variando 0.26 segundos sobre el periodo real de 5 segundos.

	Valor
media (s)	4.77
Desviación estándar (s)	0.535
Mínimo(s)	3.559
Percentil 25 % (s)	4.429
Percentil 50 % (s)	4.805
Percentil 75 % (s)	5.085
Máximo (s)	5.938

Tabla 4.2: Tabla estadísticas altura estimada (Hs)

Analizando los resultados obtenidos, podemos observar las siguientes características:

- **Valor medio:** Se ha obtenido un periodo medio de 4.766 segundos, lo que supone una subestimación del 4.7 % sobre el periodo real de 5 segundos.

$$\frac{4{,}766 - 5}{5} \cdot 100 = -4{,}59\,\%$$

- **Desviación estándar:** tiene un valor de 0.535 segundos, aproximadamente el 10.7 % del periodo real, lo que indica que las medidas tiene una variabilidad alta.
- **Rango intercuartílico:** la mitad central de los periodos estimados se encuentra entre el -11.4 % y el 1.7 % de los 5 segundos reales.
- **Valor máximo y mínimo:** el valor mínimo de la altura estimada obtenido en las pruebas ha sido de 3.559 segundos y el máximo ha sido de 5.938 segundos.

Las gráficas siguientes muestran el valor de altura (figura 4.7) y periodo (figura 4.8) estimados para cada una de las pruebas realizadas. Se puede observar, tal y como se ha comentado en el análisis estadístico realizado, como la estimación de los valores tiene una variación elevada, lo que indica que se debería ajustar el algoritmo de caracterización.

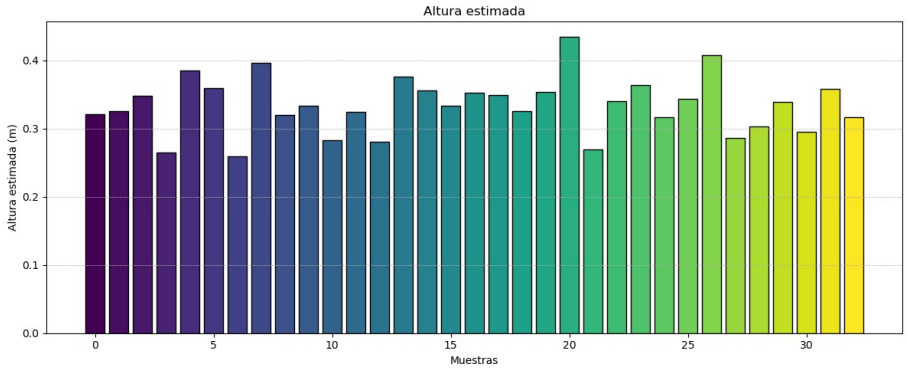

Figura 4.7: Gráfica estadística Hs

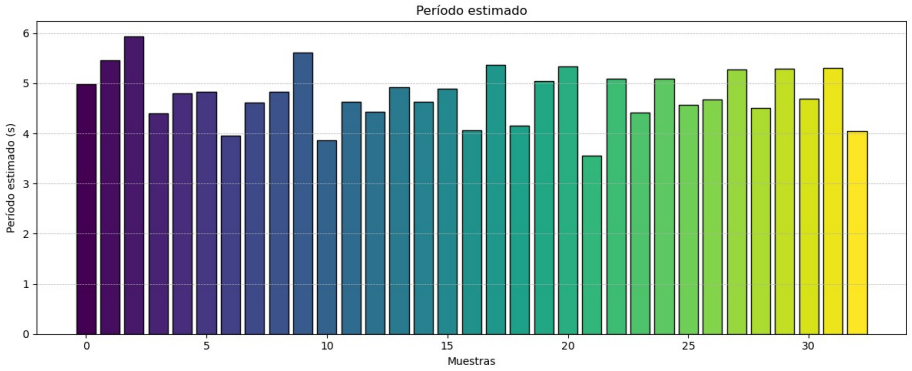

Figura 4.8: Gráfica estadística Tp

4.6. Conclusiones

Mediante las diferentes implementaciones realizadas en este capítulo, se ha podido observar el funcionamiento del algoritmo en un entorno físico. Consiguiendo un entorno experimental, mediante el sistema mecánico, que permitía obtener una entrada controlada sobre el movimiento vertical y que ha permitido realizar diversas pruebas de medición con la IMU, facilitando el análisis de los resultados.

Capítulo 5

Conclusiones y Trabajo Futuro

En la realización de este trabajo, se ha logrado diseñar e implementar un algoritmo para caracterizar oleajes de forma precisa y eficiente, esto ha sido posible gracias al uso de la metodología MBD, que ha facilitado la implementación del mismo. Este algoritmo se ejecutará en una boya de deriva, la cual formará parte de un sistema de monitorización del mar mediante estos dispositivos. Se han conseguido abordar con éxito cada objetivo planteado inicialmente, los éxitos de los cuales se presentan a continuación:

- **Modelado del oleaje:** se ha desarrollado el modelado de un oleaje basado en el espectro JONSWAP, con el cual se pueden generar oleajes con alturas y periodos realistas.
- **Modelado de la dinámica de una boya:** en este segundo modelado de la implementación se ha logrado simular el movimiento de la boya frente al oleaje generado previamente. El movimiento simulado se basa en la posición, orientación y velocidades, tanto lineales como angulares de la boya, de forma que se obtiene un movimiento realista y preciso.
- **Simulación de la IMU:** a partir del modelado de la dinámica de la boya, se han logrado obtener las mediciones del acelerómetro, giroscopio y magnetómetro para la IMU simulada, generándolos a partir del estado de la boya en cada instante de tiempo y añadiendo sesgos y ruido a la señal generada para añadir un factor de realismo a la simulación.
- **Algoritmo de estimación:** se ha logrado implementar un algoritmo de estimación capaz de caracterizar oleajes de hasta seis metros y diez segundos de periodo.
- **Implementación en python:** se ha logrado implementar el algoritmo de estima en python, lo que ha facilitado la realización de pruebas frente a una entrada real, observando los diferentes ruidos y sesgos que existen en la realidad.
- **Implementación de la biblioteca en c:** en esta implementación se han estudiado las formas para lograr la generalización de los diferentes modelos matemáticos utilizados durante el proceso de estimación. Obteniendo, finalmente, una librería genérica para poder portarla a cualquier dispositivo.
- **Pruebas realizadas:** se han podido realizar una serie de pruebas utilizando el dispositivo mecánico que simula el movimiento vertical de la ola, en las cuales se han

podido evaluar los resultados. La estimación calculada para cada una de las pruebas hechas ha tenido una desviación estándar del 16.33 % para la altura y del 10.7 % para el periodo.

En conjunto, con los logros obtenidos se puede plantear la posibilidad de implementar el algoritmo de estimación en los dispositivos de la capa *Edge* en una distribución masiva de boyas. El uso de MBD ha facilitado el diseño y desarrollo del trabajo, permitiendo avanzar de forma progresiva desde los primeros pasos de la simulación hasta la validación física.

Por otro lado, se han registrado los tiempos dedicados a cada una de las tareas en el diagrama de Gantt de la figura 7.1. Como se puede observar, las tareas han llevado más tiempo del comentado en el apartado del plan de trabajo del capítulo 1.

Figura 5.1: Diagrama de Gantt con los tiempos de realización de cada tarea

Aunque se ha logrado un buen rendimiento del algoritmo desarrollado, en un futuro se deberían tener en cuenta las siguientes consideraciones:

- Incluir en Zephyr RTOS la librería de estimación desarrollada, lo que, gracias a las facilidades de portabilidad que ofrece Zephyr, permitirá poder ejecutar el algoritmo en una gran cantidad de placas y sensores fácilmente. Esto permitirá estimar el oleaje en tiempo real a partir de los datos obtenidos desde la propia placa.
- Lograr una mejor precisión en la estimación de la altura y periodo del oleaje, se debería obtener una desviación máxima de los valores estimados aproximada al 5 % del valor real. También se debería limitar la variación de estimación entre el valor máximo y mínimo en un rango del 10 %, logrando que cuadre con la desviación máxima comentada anteriormente.
- Realizar una serie de pruebas físicas sobre un oleaje más realista, con diferente composición de fuerzas, dado que actualmente solo se simula a partir de un movimiento vertical sin llegar a modificar el ángulo de rotación de la IMU, con esta simulación más realista se podrá mejorar el algoritmo para obtener la aceleración vertical de la IMU respecto al eje-Tierra, teniendo en cuanta la dinámica de la boya.
- Realizar una validación cruzada de las pruebas, usando los datos reales de la IMU para estimar el oleaje con el algoritmo implementado en MATLAB y comparar los

resultados con el algoritmo implementado en C. Y viceversa, probar la estimación en la implementación en C con datos de la IMU simulada.

- Implementar las partes restantes del sistema propuesto, se deberán diseñar y desarrollar las boyas de deriva que formarán el despliegue en la superficie marítima. Para lograr el buen funcionamiento de esto se deberá trabajar sobre las posibles métodos de comunicación posibles, como por ejemplo comunicación satelital o mediante cobertura móvil, y también se deberá diseñar la infraestructura tecnológica donde se reciban y procesen los datos enviados por las boyas.

Chapter 6

Introduction

Alongside the terrestrial expansion of the IoT, there has also been great progress over maritime surfaces, where satellite connectivity or mobile coverage has enabled the establishment of large networks of smart buoys in the oceans. These buoys have installed sensors that allow measurements of temperature, water salinity, or to characterize wave fields, providing precise and reliable data in hard-to-reach areas. In this way, digital models of the ocean can be constructed to facilitate its study and to predict how the maritime surface will change over time.

In this work, we will design and implement a drifting buoy system capable of monitoring sea conditions through the estimation of wave height and period. The main objective of this deployment is to improve safety in the area where the buoy system is located, providing wave information both to personnel working at sea and to issue alerts about adverse phenomena.

The deployment of buoy networks would enable the creation of an Early Warning System (EWS), which can detect critical events in real time and, if a pre-established threshold is exceeded, distribute an alert to the responsible parties so that they can respond to these events and minimize damage. The factor to be analyzed in these systems can vary; for example, in the work of Chacón et al. (2024), a system is mentioned that controls water quality by detecting harmful algal and cyanobacterial blooms (HACBs). One could also analyze a factor concerning wave conditions, as in our case, and there are studies that sought to alert nearby populations about possible tsunamis or coastal flooding, such as those in Haq et al. (2019) and Doong et al. (2012). Our system's goal is to implement, alongside wave estimation, an EWS capable of detecting critical events in real time, issuing warnings about anomalies in wave conditions that could be dangerous both for vessels near the area and for nearby coastal zones where those waves might have an impact.

In the system designed in Andrade et al. (2024), a three-tier IoT architecture (Fog–Cloud–Edge) has been implemented, which efficiently handles the large volume of data and meets stringent latency, reliability, and scalability requirements.

For our system, we will also implement a Fog–Cloud–Edge architecture, in which the layers will be defined as follows:

- **Edge Level:** At this level are the buoys distributed across the ocean, each powered by a rechargeable battery via a solar panel, equipped with a satellite module to send the necessary data to control centers, and also fitted with an IMU to capture

the buoy's motions caused by waves. These measurements will be processed by an estimation algorithm to characterize the wave field.

At this layer, data measured by the IMU are collected and undergo initial processing. Filtering processes are implemented to reduce signal noise, thereby decreasing the total volume of data by eliminating erroneous or invalid readings. Once unwanted data are filtered out, a second processing stage calculates the buoy's vertical displacement to characterize the wave field in real time, obtaining its height and period. By performing wave estimation directly on the buoy, the amount of data sent for subsequent large-scale processing is greatly reduced, resulting in lower latency, less bandwidth usage, reduced battery consumption, and lower transmission costs—an important advantage for satellite communications.

Finally, at this level the EWS logic is implemented, enabling the detection of critical events in real time and the issuance of warnings when dangerous wave anomalies are detected, either for vessels in the vicinity or for nearby coastal areas that might be affected.

- **Fog Level:** This is the intermediate layer. In the current system design this layer will not be used, although it may be implemented in the future depending on the network deployment architecture.

- **Cloud Level:** This is the highest layer of the architecture, consisting of more advanced data centers with greater storage and computing capacity, located farther from the buoys.

  At this layer, the large volume of data sent by the Edge-level buoys is processed, centralizing all received data from the various buoys. This dataset is used to map ocean conditions, leveraging the higher computational resources to run advanced algorithms and models that can simulate complex wave fields and, in some cases, predict how sea state will vary over a forthcoming time interval. The processed data can then be provided both to maritime personnel—such as buoy maintenance operators or fishing vessels that use real-time wave estimates to plan daily routes—and to authorities for EWS alerts if there is a hazard to the public, or internally if further review and action are required.

Using a Fog–Cloud–Edge architecture offers great efficiency, with the most time-sensitive and critical computations performed close to the node (Edge level) and more complex calculations are done in the Cloud level, where infrastructure supports higher computational loads.

6.1. Motivation

Real-time monitoring of ocean conditions is a cornerstone for the safety and efficiency of a wide range of maritime activities. From commercial navigation and fishing—which depend on safe routes—to the protection of coastal infrastructure and the development of offshore renewable energy, having accurate data on wave conditions is crucial. Early Warning Systems (EWS) powered by networks of smart buoys have proven vital for mitigating risks associated with extreme events such as dangerous wave surges or tsunamis, enabling rapid response that can save lives and reduce economic losses.

Although there are very precise commercial solutions, they often rely on specialized—and expensive—sensors. An example is moored buoys, which are anchored to the seafloor and remain nearly stationary at the surface. These buoys are robust but costly both in initial deployment and maintenance, and their placement is constrained by seafloor conditions, limiting possible installation sites. These factors have driven research toward more affordable, lower-power alternatives, facilitating larger-scale deployments. The result is drifting buoys, which are released to drift at sea, as their name implies. Advantages of these buoys include low production and maintenance costs, ease of repair in case of failure, and the ability to deploy virtually anywhere without reliance on the seafloor, granting access to remote ocean regions requiring periodic monitoring. It is true that moored buoys yield cleaner signals—being affected only by vertical and lateral wave motion—although mooring can sometimes restrict measurements. In contrast, drifting buoys are influenced by additional factors such as currents, but these factors provide more realistic data on the maritime surface.

In this context, we face the technological challenge of characterizing a wave field—estimating its height and period—using only an IMU, a cost-effective sensor integrable with most microcontrollers and featuring low power consumption, which is a major advantage when deployed in hard-to-access, battery-powered devices.

The central motivation of this master's thesis is the challenge outlined above. Given the scarce availability of studies focused exclusively on wave characterization via IMUs, there is a clear need to investigate and validate this approach both in a simulated environment—where variables can be controlled—and through a physical implementation to demonstrate viability under real conditions.

As described in the three-tier architecture (Edge–Fog–Cloud), parameter estimation and alert detection tasks must be executed with the lowest possible latency. Therefore, this work will focus specifically on the design, development, and validation of the computation software for the Edge layer—that is, the algorithm that will run directly on the buoy. This focus is critical because the buoy software must be computationally lightweight, robust, and energy-efficient to ensure autonomy.

To systematically and reliably develop this software, the Model-Based Design (MBD) methodology will be adopted. MBD is a development paradigm in which an executable model of the system is the central artifact throughout the project lifecycle—from initial concept and requirements analysis to implementation and testing (Andrade et al., 2024).

MBD is highly effective for this work because it allows system design, development, and testing without any hardware or physical environment. This enables the development and implementation of the estimation algorithm through models and simulations. To achieve this, both the wave field and the buoy dynamics under generated waves must be modeled, and then simulated IMU data are extracted from the buoy's modeled motion over a time window, allowing testing of the algorithm's functionality.

This greatly facilitates the progress of the work, as the wave state is known at all times during the simulation without relying on, for example, a physical test tank. Additionally, it reduces error factors present in physical implementations, making development and debugging simpler. Of course, bias and noise can be added to the models to produce more realistic outputs.

6.2. Objectives

The primary objective of this work is to design and implement an application capable of characterizing the wave field using IMU measurements installed on a buoy, estimating wave height and period. The following specific objectives have been defined to achieve this:

- Implement an application capable of characterizing a wave field, for which the following specific objectives are required:
  1. Model the wave field using the JONSWAP spectrum to generate realistic wave characteristics. This wave field will form the basis for simulating the estimation algorithm, seeking maximum realism to ensure effective real-device testing.
  2. Model the buoy's motion under the generated wave field, obtaining realistic motion that accounts for the buoy's degrees of freedom.
  3. Simulate IMU measurements realistically by incorporating noise and bias typical of physical devices.
  4. Develop an accurate and efficient estimation algorithm for various wave types.
- Implement in Python the wave-characterization application that receives acceleration data from an IMU, calibrates and processes it, and estimates the wave field.
- Develop a generic C library capable of estimating wave parameters given time-stamped acceleration input.
- Conduct tests to demonstrate the efficiency of the previous implementations. Tests will be carried out using a mechanical setup driven by a stepper motor to simulate vertical wave motion on a physical IMU.

6.3. Work Plan

The work plan is divided into the following parts:

1. **Preliminary Study and Literature Review:** Conduct a comprehensive study of projects related to wave, buoy, or IMU modeling and simulation, as well as works focused on wave characterization and existing techniques. This initial study will provide a knowledge base on modeling and estimation methods and the mathematical principles required for our project.

   Estimated duration: 3 weeks.

2. **MATLAB Implementation:** Develop the different models and simulations: first the wave field, then buoy dynamics, and finally simulate the IMU. Once these are implemented, design the characterization algorithm and implement the necessary mathematical models to obtain accurate wave height and period estimates. MATLAB is chosen for its ease of handling equations, models, and plotting.

   Estimated duration: 3 weeks.

3. **Python Estimation Implementation:** After validating the estimation algorithm, test its operation with real accelerometer data, which will be pre-processed before

input to the algorithm. Python is chosen for its similarity to MATLAB in math functions and its ease of plotting results.

Estimated duration: 2 weeks.

4. **Test System Development:** Design a mechanical device capable of generating sinusoidal vertical motion to simulate wave action on a physical IMU.

   Estimated duration: 1 week.

5. **C Estimation Library Implementation:** Finally, develop a generic C library to perform wave estimation from accelerometer data read from a CSV file that has been pre-processed for precise estimation. This will involve implementing functions for the necessary characterization processes.

   Estimated duration: 3 weeks.

6. **Testing:** For each implementation, conduct a series of tests to analyze estimation results and verify the algorithm's precision and efficiency.

   Estimated duration: 2 weeks.

6.4. Thesis Outline

This thesis is composed of the following chapters:

1. **Introduction:** Contextualizes the aspects leading to the work's motivation, states the objectives, and presents the work plan and tasks.
2. **State of the Art:** Reviews the literature on relevant works in the areas addressed.
3. **Model-Based Design:** Explains the mathematical models and foundations followed to achieve correct results, based on the studies in the previous chapter, including wave modeling, buoy dynamics, and IMU measurements, culminating in the characterization algorithm.
4. **Implementation:** Describes and explains the implementations developed to validate the estimation algorithm.
5. **Conclusions and Future Work:** Analyzes the objectives and results of the tests, presents a timeline diagram of completed tasks, and proposes future work for a more comprehensive algorithm.

Chapter 7

Conclusions and Future Work

During the realization of this work, a precise and efficient algorithm for characterizing waves has been successfully designed and implemented, this has been possible thanks to the use of the MBD metodology, which facilitated its implementation. This algorithm will run on a drifting buoy, which will form part of a sea-monitoring system using these devices. Each of the initially proposed objectives has been successfully addressed, the achievements of which are presented below:

- **Wave modeling:** A wave model based on the JONSWAP spectrum has been developed, enabling the generation of waves with realistic heights and periods.
- **Buoy dynamics modeling:** In this second implementation stage, the motion of the buoy in response to the previously generated waves has been successfully simulated. The simulated motion is based on the buoy's position, orientation, and both linear and angular velocities, yielding a realistic and accurate movement.
- **IMU simulation:** From the buoy dynamics model, accelerometer, gyroscope, and magnetometer measurements for the simulated IMU have been obtained by deriving them from the buoy's state at each time step and adding biases and noise to the signals for added realism.
- **Estimation algorithm:** An estimation algorithm capable of characterizing waves up to six meters in height and ten seconds in period has been implemented.
- **Python implementation:** The estimation algorithm has been implemented in Python, facilitating tests against real input data and allowing observation of the various noises and biases present in practice.
- **C library implementation:** In this implementation, methods for generalizing the different mathematical models used during estimation have been studied, resulting in a generic library that can be ported to any device.
- **Tests performed:** A series of tests using a mechanical device that simulates vertical wave motion were conducted, and the results evaluated. The estimated values in these tests showed a standard deviation of 16.33% for height and 10.7% for period.

Overall, these achievements demonstrate the feasibility of implementing the estimation algorithm on *Edge*-layer devices in a large-scale deployment of buoys. The use of Model-

Based Design (MBD) has facilitated the design and development process, allowing for a progressive progression from initial simulation steps to physical validation.

On the other hand, the time dedicated to each task has been recorded in the Gantt chart shown in Figure 7.1. As can be observed, the tasks took longer than indicated in the work plan section of chapter 6.

Figure 7.1: Gantt chart showing the duration of each task

Although the developed algorithm achieved good performance, the following considerations should be taken into account in the future:

- Include the developed estimation library in Zephyr RTOS, which—thanks to Zephyr's portability features—will allow the algorithm to run easily on a wide range of boards and sensors. This will enable real-time wave estimation directly from onboard data.
- Achieve better accuracy in the estimation of wave height and period: the maximum deviation of estimated values should be approximately 5% of the real value. Moreover, the variation between the maximum and minimum estimated values should be limited to within a 10% range, in line with the maximum deviation target.
- Conduct a series of physical tests on more realistic waves with varied force compositions; currently, only vertical motion is simulated without changing the IMU's rotation angle. A more realistic simulation will improve the algorithm's ability to derive the IMU's vertical acceleration relative to the Earth-fixed frame, taking buoy dynamics into account.
- Perform cross-validation of tests by using real IMU data to estimate waves with the MATLAB implementation and comparing the results to those of the C implementation. Conversely, test the C implementation's estimation using simulated IMU data.
- Implement the remaining parts of the proposed system by designing and developing the drifting buoys for deployment on the sea surface. To ensure proper operation, potential communication methods—such as satellite links or mobile networks—should be evaluated, and the technological infrastructure for receiving and processing data from the buoys must be designed.

Bibliografía

Y así, del mucho leer y del poco dormir, se le secó el celebro de manera que vino a perder el juicio.

Miguel de Cervantes Saavedra

Andrade, G. A., Esteban, S., Risco-Martín, J. L., Chacón, J. y Besada-Portas, E. Modeling- and simulation-driven methodology for the deployment of an inland water monitoring system. *Information*, vol. 15(5), página 267, 2024.

Badulin, S. I. y Zakharov, V. E. The phillips spectrum and a model of wind-wave dissipation. *Theoretical and Mathematical Physics*, vol. 202(3), página 309–318, 2020.

Chacón, J., Andrade, G. A., Risco-Martín, J. L. y Esteban, S. A bleeding edge web application for early detection of cyanobacterial blooms. *Electronics*, vol. 13(5), página 942, 2024.

Doong, D.-J., Chuang, L. Z.-H., Wu, L.-C., Fan, Y.-M., Kao, C. C. y Wang, J.-H. Development of an operational coastal flooding early warning system. *Natural Hazards and Earth System Sciences*, vol. 12(2), página 379–390, 2012.

Espressif. Esp32-c3-devkit-rust-1 development board. Disponible en `https://www.espressif.com/en/dev-board/esp32-c3-devkit-rust-1-en`.

Fournier, A. y Reeves, W. T. A simple model of ocean waves. *ACM SIGGRAPH Computer Graphics*, vol. 20(4), página 75–84, 1986.

Guo, Q. y Xu, Z. Simulation of deep-water waves based on jonswap spectrum and realization by matlab. *2011 19th International Conference on Geoinformatics*, página 1–4, 2011.

Gwatiringa, T. G. Sea state estimation from inertial platform data for real-time ocean wave prediction. 2018.

Gwatiringa, T. G., Verrinder, R. A. y Boje, E. Phase-locked loop and kalman filter strategy to track ocean waves from mobile platform imu data ∗ ∗this work has been supported by a joint south african department of science and technology (dst) and council for scientific and industrial research (csir) robotics strategy of south africa (rossa) grant. *IFAC-PapersOnLine*, vol. 50(2), página 277–282, 2017.

Gómez, C. Carpeta de google drive con las implementaciones realizadas. Disponible en `https://drive.google.com/drive/folders/1uXeVyhZz8bSPXQSuVJ3bB-ajENq7gvJD?usp=sharing`.

Haq, E. S., Suwardiyanto, D., Ayatullah, M. D., Rini, E. M., Sutiksno, H. y Setyati, E. The coastal early warning system based on buoy sensor measurement. En *2019 2nd International Conference of Computer and Informatics Engineering (IC2IE)*, páginas 39–43. 2019.

Huang, Y.-L., Kuo, C.-Y., Shih, C.-H., Lin, L.-C., Chiang, K.-w. y Cheng, K.-C. Monitoring high-frequency ocean signals using low-cost gnss/imu buoys. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLI-B8, página 1127–1134, 2016.

Johnson, B. y Cotilla-Sanchez, E. Estimating the impact of ocean wave energy on power system reliability with a well-being approach. 2020.

Jouybari, A., Amiri, H., Ardalan, A. A. y Zahraee, N. K. Methods comparison for attitude determination of a lightweight buoy by raw data of imu. *Measurement*, vol. 135, página 348–354, 2019.

Karimirad, M. Aerodynamic and hydrodynamic loads. *Offshore Energy Structures*, página 187–221, 2014.

Li, X. y Bian, Y. Modeling and prediction for the buoy motion characteristics. *Ocean Engineering*, vol. 239, página 109880, 2021.

Malek Azari, M., Salazar Luces, J. V. y Hirata, Y. Design, assessment and evaluation of structural stabilization system for weather buoys using a moving foil. *ROBOMECH Journal*, vol. 7(1), 2020.

Pierson, W. J. y Moskowitz, L. A proposed spectral form for fully developed wind seas based on the similarity theory of s. a. kitaigorodskii. 1963.

TDK_InvenSense. Icm-42670-p 6-axis imu datasheet. Disponible en `https://invensense.tdk.com/wp-content/uploads/2021/07/DS-000451-ICM-42670-P-v1.0.pdf`. Datasheet ICM-42670-P.

TDK_InvenSense. Icm-42670 motion-tracking sensor. Disponible en `https://invensense.tdk.com/products/motion-tracking/6-axis/icm-42670-p/`. Página oficial del sensor ICM-42670-P.

Tessendorf, J. Simulating ocean water. 2001.

Zhang, Y., Qi, L., Dong, J., Wen, Q. y Lv, M. Data processing based on low-precision imu equipment to predict wave height and wave period. *2019 2nd International Conference on Data Intelligence and Security (ICDIS)*, página 103–107, 2019.

Zheng, H., Chen, Y., Liu, Q., Zhang, Z., Li, Y. y Li, M. Theoretical and numerical analysis of ocean buoy stability using simplified stability parameters. *Journal of Marine Science and Engineering*, vol. 12(6), página 966, 2024.

Apéndice A

Parámetros de la implementación matlab

	Símbolo	**Valor**
Resolución temporal de la simulación	$\Delta t\ (s)$	0.005
Duración total de la simulación	$T_{sim}\ (s)$	120
Vector de tiempo uniforme	$tVec$	$[0 : \Delta t : T_{sim}]$
Aceleración de la gravedad	$g\ \frac{m}{s^2}$	9,81
Densidad del agua de mar	$\rho\ (\frac{kg}{m^3})$	1025

Tabla A.1: Parámetros de simulación

	Símbolo	**Valor**
Masa	$m_{buoy}\ (kg)$	7
Momentos de inercia	$I_{xx},\ I_{yy},\ I_{zz}\ (kg \cdot m^2)$	0,13, 0,13, 0,185
Matriz diagonal de inercia	I_{buoy}	$\begin{bmatrix} 0{,}13 & 0 & 0 \\ 0 & 0{,}13 & 0 \\ 0 & 0 & 0{,}185 \end{bmatrix}$
Coeficientes de traslacion	$C_{surge},\ C_{sway},\ C_{heave}\ (\frac{N \cdot s}{m})$	100, 100, 300
Coeficientes de rotación	$C_{roll},\ C_{pitch},\ C_{yaw}\ (\frac{N \cdot m \cdot s}{rad})$	20, 20, 30
Matriz de amortiguamiento lineal	C_{lin}	$\begin{bmatrix} C_{surge} & 0 & 0 & 0 & 0 & 0 \\ 0 & C_{sway} & 0 & 0 & 0 & 0 \\ 0 & 0 & C_{yaw} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{roll} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{pitch} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{yaw} \end{bmatrix}$
Superficie en contacto con el agua	$A\ (m^2)$	0.16
Altura metacéntric en roll	$M_{roll}\ (m)$ 0.05	
Altura metacéntric en pitch	$M_{roll}\ (m)$ 0.05	
Coeficientes de traslacion hidrostáticos	$k_{surge},\ k_{sway},\ k_{heave}\ (\frac{N \cdot s}{m})$	0; 0; $\rho \cdot g \cdot A = 1608{,}84$
Coeficientes de rotación hidrostáticos	$k_{roll},\ k_{pitch},\ k_{yaw}\ (\frac{N \cdot m \cdot s}{rad})$	$m_{buoy} \cdot g \cdot M_{roll} = 3{,}433;\ m_{buoy} \cdot g \cdot M_{roll} = 3{,}433;\ 0$
Matriz de rigidez hidrostática	K_{hydro}	$\begin{bmatrix} k_{surge} & 0 & 0 & 0 & 0 & 0 \\ 0 & k_{sway} & 0 & 0 & 0 & 0 \\ 0 & 0 & k_{yaw} & 0 & 0 & 0 \\ 0 & 0 & 0 & k_{roll} & 0 & 0 \\ 0 & 0 & 0 & 0 & k_{pitch} & 0 \\ 0 & 0 & 0 & 0 & 0 & k_{yaw} \end{bmatrix}$

Tabla A.2: Parámetros de la boya