ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Marítima y Ciencias del Mar

Diseño Conceptual de Embarcaciones Planeadoras Incluyendo Flaps para Servicio Interislas en Galápagos Utilizando el Algoritmo NSGA-II

PROYECTO INTEGRADOR

Previo la obtención del Título de:

INGENIERO NAVAL

Presentado por:

Janeth Leslie Meléndez Rivera

GUAYAQUIL - ECUADOR

Año: 2021

DEDICATORIA

Este trabajo se lo dedico a mi pequeña familia, a mi futuro esposo José Eduardo, a mi madre y hermana. A mis mejores amigos Steven y Samanta que fueron mi motivación durante tantos años y nunca dejaron que me rindiera. Y a mi amigo Edgar que me dio ánimos cada día en el trabajo para que culminara mis estudios, hoy lo hace desde el cielo.

AGRADECIMIENTOS

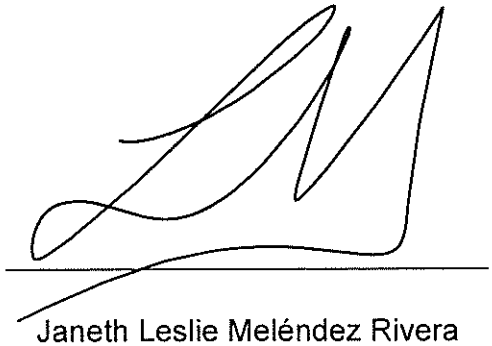
Al Dr. Esteban L. Castro Feliciano por su colaboración en este proyecto.

A mi tutor José R. Marín López por los conocimientos impartidos durante todos estos años de estudio y por su guía en el desarrollo de este trabajo.

A los profesores de la carrera de ingeniería naval, a cada uno por su esfuerzo y dedicación en formar profesionales para el día de mañana.

DECLARACIÓN EXPRESA

"Los derechos de titularidad y explotación, me corresponde conforme al reglamento de propiedad intelectual de la institución; *Janeth Leslie Meléndez Rivera* y doy mi consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"



Janeth Leslie Meléndez Rivera

EVALUADORES

Firmado electrónicamente por
NADIA ROSENDA
MUNOZ AGILA

Nadia Rosenda Muñoz Agila, MSc

PROFESOR DE LA MATERIA

JOSE ROLANDO MARIN LOPEZ

Digitally signed by JOSE ROLANDO MARIN LOPEZ
Date: 2021.12.15 14:13:58 -05'00'

José Rolando Marín López, PhD

PROFESOR TUTOR

RESUMEN

A principios del año 2020 se realizaron pruebas de mar en lanchas de transporte interislas en Galápagos registrándose mediciones de la aceleración en tiempo. Se encontró que muchos pasajeros llegaron a puerto con síntomas de mareo y algunos inclusive vomitaron. Este proyecto pretende dimensionar a nivel conceptual una embarcación interislas para operar en Galápagos, tratando de reducir la aceleración vertical, que es relacionada con el malestar en los pasajeros, considerando simultáneamente la resistencia al avance. Se planea utilizar el algoritmo multiobjetivo de optimización NSGA-II en Python y considerando la inclusión de apéndices tipo flap para reducir el ángulo de planeo de la lancha. Se incluyeron como variables de diseño: eslora, L, manga, B, posición longitudinal del centro de gravedad, lcg, ángulo de astilla muerta, β, ángulo de deflexión, δ, y cuerda del flap. Las restricciones consideradas son: ángulo de trimado dinámico, relación eslora-manga, criterios de estabilidad, francobordo y área para pasajeros. Los resultados muestran que el desempeño en el mar de la lancha es altamente afectado por los parámetros, lcg, β y δ. Las combinaciones de estos parámetros que reducen la función objetivo aceleración se producen principalmente cuando $\delta > 10°$, $\beta = 30°$ y con una ubicación del $lcg > 35\%$ de la eslora. Finalmente, el algoritmo NSGA-II produce un conjunto de soluciones en un espacio de diseño llamado frente de Pareto, que muestra que la aceleración podría reducirse hasta un 19% y la resistencia al avance de la lancha en un 5% con el incremento de las variables de diseño.

Palabras Clave: flaps, optimización, aceleración vertical, lcg

ABSTRACT

Early 2020, se trials were carried out in the interisland high-speed boats in Galapagos, it was recorded measurements for impact acceleration in time. It was found that many passengers arrived at port with motion sickness symptoms and some of them even throwing up. This work pretends to size in a conceptual level a high-speed boat to operate in Galapagos, trying to reduce the vertical acceleration, mainly cause of the motion sickness, while considering the resistance to advance. It is planned to use the multiobjective optimization method NSGA-II in python language and considering the inclusion of flaps which are a type of stern appendages to reduce the planing angle of the boat, and as a final result a reduction in the acceleration of its movement. The design variables included were: length, L, beam, B, longitudinal position of the center of gravity, lcg, deadrise angle of the section, β, flap angle of deflection, δ, and flap chord. The constrains considered were: dynamic trim angle, length-beam ratio, stability criteria, freeboard and required area for passengers. The results shows that the sea performance of the planing boat is highly affected by the parameters, lcg , β, and δ. The combinations of these parameters that helps to reduce the acceleration objective function mainly occur when $\delta > 10$ °, $\beta = 30$ ° and with a location of the $lcg > 35\%$ of the length of the ship. It was also observed that while the length of the flap is reduced, it needs a greater operating angle to reduce the vertical acceleration of the boat. Finally, the Genetic NSGA-II algorithm produces a set of solutions in a design space called the Pareto front, which shows that the acceleration could be reduced by up to 19% and the resistance to advance of the boat by 5% with the increase of the design variables.

Keywords: *flaps, optimization, vertical acceleration, lcg*

ÍNDICE GENERAL

ABREVIATURAS

GA	Genetic Algorithm
ISO	International Organization for Standardization
MSI	Motion Sickness Index
NSGA-II	Non-Sorting Genetic Algorithm
R_{olas}	Resistencia en olas
$R_{friccional}$	Resistencia Friccional
$R_{hidrodinamica}$	Resistencia Hidrodinámica
$R_{fuerza\ del\ flap}$	Resistencia por acción del flap
R_{aire}	Resistencia por Aire
$R_{whisper\ spray}$	Resistencia por el whisper spray

SIMBOLOGÍA

a_{CG}	Aceleración del centro de gravedad, m/s^2
$H_{\frac{1}{3}}$	Altura significativa de la ola
β	Ángulo de astilla muerta
τ	Ángulo de trimado
$\tau_{propising}$	Ángulo de trimado dinámico límite
$LoadArea$	Área mínima requerida para acomodación de pasajeros
T_{DWL}	Calado a la línea de agua
lcg	Centro de gravedad longitudinal
KG	Centro de gravedad vertical
C_Δ	Coeficiente de carga $\frac{\Delta}{wb^3}$
L_F	Cuerda del flap
δ	Deflección del flap
LOA	Eslora total
α_n	Factor de amontonamiento
km	Factor que depende de la población, 1/3 para población mixta
$FreeBoard$	Franco bordo
Δ_F	Incremento del levantamiento del flap
D_F	Incremento en la resistencia del flap
$KG_{DynMoment}$	KG máximo para que el momento dinámico se estabilice
L_k	Longitud de la quilla mojada
b	Manga en la estación 5
B_{max}	Manga maxima
M_F	Momento debido al flap
T	Periodo total de exposición
D	Puntal
σ	Radio de espaciamento/manga (Span-Beam Ratio)
R_T	Resistencia total
V_k	Velocidad en nudos
V	Velocidad

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE PLANOS

CAPÍTULO 1

1. INTRODUCCIÓN

El Archipiélago de Galápagos representa uno de los destinos turísticos más importantes del Ecuador. En el año 2019, antes de la pandemia, por el Corona virus, se recibieron 271 mil turistas en las islas según el informe anual de visitantes (Ministerio del Ambiente y Agua, 2020). Para esta alta demanda de transporte de turistas entre islas existen dos tipos de transportes, aéreo y marítimo. El aéreo se realiza sólo una vez por día y en un determinado horario, con un elevado costo, trabajándose bajo reserva. El transporte marítimo con un costo mucho menor realiza aproximadamente 142 viajes al día usando 71 lanchas rápidas, es el tipo de transporte más frecuentemente utilizado por los turistas que visitan las islas.

Estos viajes interislas se realizan mediante lanchas planeadoras de compañías de transporte privadas que miden aproximadamente 12 metros de eslora, 3 metros de manga y 1,5 metros de puntal, con capacidad promedio de 28 pasajeros y construidas con fibra de vidrio. Son impulsadas por motores fueraborda de alta potencia y alcanzan velocidades de aproximadamente 28 nudos. Las rutas principales que recorren estas lanchas son Santa Cruz-Isabela y Santa Cruz-San Cristóbal en viajes de ida y vuelta con horarios de salida en la mañana y tarde. Para las dos rutas la distancia de recorrido es aproximadamente 80 kilómetros con un tiempo aproximado de viaje de 2 horas; ambas incluyen un tramo considerado de mar abierto.

En Galápagos el máximo valor de altura significativa ocurre en el mes de Julio con un estado de mar 4, y el mínimo en Diciembre, con un estado de mar de 3, según el Instituto Oceanográfico y Antárctico de la Armada del Ecuador. Las altas velocidades y el largo trayecto de viaje en ambas rutas ocasionan gran malestar en los pasajeros de estas lanchas. A principios del año 2020 los ingenieros Mendoza y Vázquez (Mendoza & Vásquez, 2020) reportaron que muchos de los pasajeros llegaron con síntomas de mareo y algunos inclusive vomitaron al terminar el trayecto desde Puerto Ayora tanto a Santa Cruz como a Isabela. Por lo tanto, hay necesidad de

mejorar la comodidad de los usuarios de estas embarcaciones que permita brindar un mejor servicio para los pasajeros.

1.1 Descripción del problema

Cinetosis o también conocido como mareo es un malestar producido en los pasajeros por el movimiento de las embarcaciones cuando navegan en olas. Se atribuye este malestar principalmente a las aceleraciones verticales elevadas que exceden valores límites por ejemplo presentados en la norma ISO (ISO 2631-1, Reviewed 2021, p. 33). En mar abierto y a las elevadas velocidades con que operan, las embarcaciones planeadoras experimentan grandes amplitudes en los movimientos de cabeceo y levantamiento, lo que puede causar que este tipo de embarcaciones incluso salgan del agua. Estos elevados movimientos causados por las olas resultan en altos valores de aceleración, lo que sumado al prolongado tiempo de exposición resulta en vértigo y mareos en los pasajeros.

El valor de la aceleración vertical expresada en porcentajes de aceleración gravitacional determina el grado de incomodidad en pasajeros. La norma ISO 2631 expone que cuando los valores de la aceleración del centro de gravedad sobrepasan los 0.2g, los pasajeros experimentarán un extremo grado de incomodidad. Actualmente en las lanchas planeadoras interislas los pasajeros experimentan desde 0.15g hasta 0.4g dependiendo de la ruta del planeador (Villamarín, 2020). Por lo tanto, es importante que el diseñador reduzca al máximo el movimiento de la embarcación para asegurar la comodidad de los pasajeros durante el trayecto de un viaje.

El diseño de embarcaciones rápidas es un proceso complejo. Localmente, el diseño y construcción de estas embarcaciones se realiza de manera empírica, en el que no se incluyen la respuesta de los planeadores en olas ni el cumplimiento de normas al respecto, por ejemplo ISO (ISO 2631-1, Reviewed 2021) o similares. La respuesta de una embarcación en olas significa altas amplitudes lo que hace al problema de tipo no lineal como lo menciona Savitsky en su estudio de embarcaciones planeadoras (Savitsky & Koelbel , Seakeeping of Hard Chine Hulls,

1993), haciendo el problema muy complicado de resolver analíticamente. Entonces para su diseño es conveniente incluir procesos de optimización para seleccionar adecuadamente los parámetros que definen la nave, basándose en relaciones empíricas a partir de experimentación, para evaluar su desempeño. De esta manera se podrían determinar las nuevas dimensiones considerando el mareo que es causado por la aceleración vertical y sin dejar de lado la resistencia al avance del planeador.

1.2 Justificación del problema

El malestar de mareo y vómito representa un evidente problema para los usuarios y operadores de las lanchas que proveen transporte interislas en Galápagos como lo reportaron Mendoza y Vásquez en 2020. Mediante la instalación de dispositivos de control de trimado se podría conseguir una mejora en la dinámica de la embarcación la cual ayudará a aminorar la cinetosis de los pasajeros. El beneficio se centra en la comodidad de los usuarios y operadores en el trayecto del viaje en Galápagos, siendo este uno de los lugares más turísticos del Ecuador y que atrae a numerosos turistas de diferentes partes del mundo.

En el trabajo realizado anteriormente de lanchas interislas de Galápagos (Villamarín, 2020) se concluye que se logra una disminución de la aceleración del centro de gravedad mediante el movimiento hacia proa del centro de gravedad de la lancha. Se recomienda además investigar si con la instalación de los flaps se lograría un equivalente movimiento hacia proa del CG. Por tal razón, en este proyecto se pretende seleccionar de manera óptima las dimensiones principales de las lanchas interislas de Galápagos incluyendo flaps o trim tabs para determinar si con su instalación se mejora el problema de cinetosis de pasajeros mediante la reducción de la aceleración vertical del planeador. Finalmente, se podrá hacer una evaluación de los modelos desarrollados y analizar las potenciales mejoras al instalar los flaps en los planeadores.

1.3 Objetivos

1.3.1 Objetivo General

Mediante la ejecución de este proyecto se pretende reducir la incomodidad que experimentan los pasajeros en el trayecto de los viajes interislas, fijando de manera general el siguiente objetivo.

Dimensionar a nivel conceptual una embarcación interislas con flaps que opera en Galápagos, utilizando un método multiobjetivo de optimización para una disminución de la aceleración vertical consiguiendo mejoras en la comodidad de los pasajeros.

1.3.2 Objetivos Específicos

Se determinan las metas a alcanzar al finalizar cada una de las etapas en el proceso de ejecución de este proyecto, fijando de manera específica los siguientes objetivos.

1. Implementar un esquema de optimización estableciendo las dimensiones básicas del flap, consiguiendo una reducción en la aceleración vertical de una embarcación que presta servicio interislas en Galápagos.
2. Aplicar el esquema de optimización seleccionado para el dimensionamiento de la lancha con los flaps, obteniendo una reducción en la aceleración vertical, considerando la resistencia al avance.
3. Preparar un diseño conceptual de la embarcación planeadora considerando los resultados obtenidos en el proceso de optimización y para la comparación geométrica con el modelo original.

1.4 Marco Teórico

1.4.1 Dinámica de las Embarcaciones Rápidas Planeadoras

La dinámica de las embarcaciones rápidas es un fenómeno complejo que no puede resolverse analíticamente. Cuando se alcanzan velocidades de planeo, todas las

características de las embarcaciones rápidas tienen influencia compleja en los movimientos, aceleraciones y resistencia al avance en olas (Savitsky & Koelbel , Seakeeping of Hard Chine Hulls, 1993, p. 19). Ante la ausencia de un método analítico para resolver el problema, el ingeniero diseñador debe emplear fórmulas empíricas resultantes de experimentación en tanques de pruebas. Por ejemplo las desarrolladas por el profesor Savitsky (Savitsky & Brown, Procedures For Hydrodynamic Evaluation of Planing Hulls in Smooth and Rough Water, 1976) que permiten estimar la respuesta hidrodinámica de los planeadores dentro del rango de aplicabilidad de las variables en los experimentos.

Algunas variables de diseño que pueden ser controladas por el diseñador proveen un mejor desempeño del planeador en el olas y en la comodidad de los pasajeros. Mejoras tales como el movimiento vertical, reducción en resistencia y disminución en la aceleración de impacto se pueden conseguir con la variación apropiada de los parámetros L/B, ángulo de astilla muerta, movimiento del lcg, desplazamiento y altura metacéntrica (Savitsky & Koelbel , Seakeeping of Hard Chine Hulls, 1993, p. 88). Debido a que el mareo o cinetosis depende principalmente de la aceleración vertical que cualquier otro movimiento del planeador. Los parámetros ángulo de astilla muerta y lcg cumplen un rol muy importante en el diseño de la nave. Ambos parámetros influyen directamente a la reducción de la aceleración de impacto.

Es bastante bien establecido que el movimiento del lcg hacia proa reduce la aceleración de impacto de lanchas planeadoras. En diferentes estudios realizados por Savitsky (Savitsky & Koelbel , Seakeeping of Hard Chine Hulls, 1993) se determina que un movimiento hacia proa influye en la reducción del trimado. A altas velocidades y con un ángulo de trimado estático se consigue una reducción de la aceleración de impacto. Existen diferentes formas de lograr un movimiento del lcg, una de ellas es mediante el arreglo general de pesos. Pero cuando el diseño ha sido desarrollado y no es posible realizar cambios se puede implementar dispositivos de control de trimado o flaps.

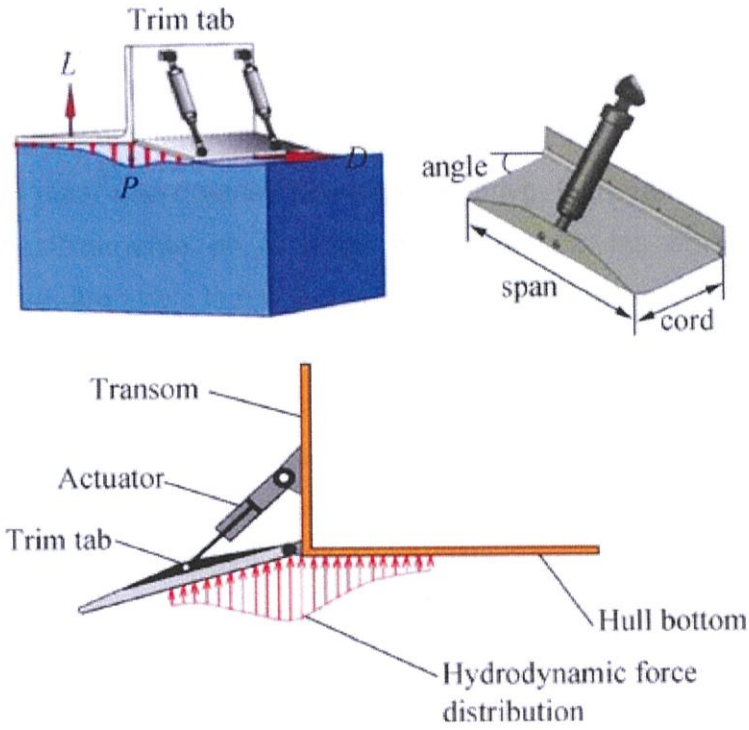


Figura 1.1 Características de los flaps

(Sakaki & Ghassemi, 2019)

Los flaps o también conocidos como trim tabs se instalan en la popa de un bote planeador para mejorar su respuesta dinámica. Estos apéndices producen un momento opuesto al levantamiento dinámico consiguiendo un movimiento equivalente de lcg y como consecuencia reducción del trimado (Savitsky & Koelbel , Seakeeping of Hard Chine Hulls, 1993). Posteriormente se puede conseguir una reducción en la aceleración vertical modificando los parámetros longitud de la cuerda, ancho y ángulo de deflexión, ver Figura 1.1. Para determinar las dimensiones de los flaps se han llevado a cabo estudios en pruebas de tanques (Savitsky & Brown, Procedures For Hydrodynamic Evaluation of Planing Hulls in Smooth and Rough Water, 1976). Pero un proceso que tome en consideración la aceleración de impacto y resistencia al avance requiere de construcción de modelos y numerosas pruebas de tanques. Una opción para determinar las dimensiones del bote planeador incluyendo flaps es mediante procesos multiobjetivo de optimización.

1.4.2 Optimización de Embarcaciones Rápidas

La optimización de embarcaciones de planeo es un proceso clave en la etapa del diseño que permite considerar su respuesta en olas. Este proceso se utiliza para maximizar o minimizar una o varias funciones objetivo. Generalmente en el diseño de buques específicamente en la parte hidrodinámica, las embarcaciones de desplazamiento se rigen bajo la función de resistencia al avance R_T. En las lanchas planeadoras debido a su comportamiento dinámico complejo existen varios componentes además de la resistencia que deben ser considerados (Savitsky & Koelbel , Seakeeping of Hard Chine Hulls, 1993). Un proceso de optimización que permite emplear más de una función objetivo simultáneamente se conoce como optimización multiobjetivo (Kochenderfer & Wheeler, 2019). Estos procesos permiten determinar las dimensiones óptimas de las variables del diseño que cumplan con las funciones objetivos sujetas a las restricciones.

Los procesos de optimización pueden estar agrupados por ciertas características del problema. Cuando la función objetivo es de tipo lineal se pueden aplicar métodos dependientes de las derivadas o gradientes como por ejemplo el método de Newton (Kochenderfer & Wheeler, 2019, p. 87). En la optimización de lanchas planeadoras en donde el problema es de tipo no lineal con variables de decisión no lineales, se pueden emplear métodos de gradientes como el de direcciones factibles (Vanderplaats, 2007), o a su vez métodos libres de gradientes. Recientemente, autores han desarrollado procesos de optimización multiobjetivo de lanchas rápidas utilizando algoritmos basados en población (Mohamad Ayob A. F., 2012), evolución y partículas de enjambre (Knight, 2014). Estos métodos son de tipo estocásticos y son muy sofisticados. El lenguaje de programación y el software para utilizar deben ser los apropiados para desarrollar el problema. Una opción a estos métodos son los algoritmos multiobjetivo basados en métodos exactos, como por ejemplo la optimización restringida multiobjetivo.

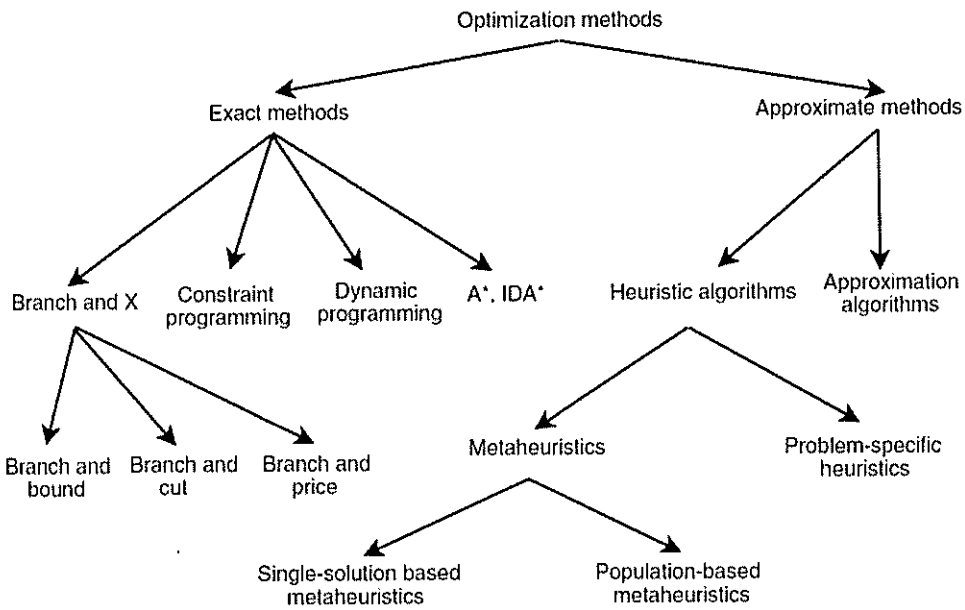


Figura 1.2 Esquema de los procesos de optimización

En este proyecto se aplica optimización tipo multiobjetivo para el diseño de una lancha planeadora, tratando de minimizar la aceleración vertical pero considerando también la resistencia al avance de la embarcación. Este proceso está sujeto a restricciones de tipo no lineal con múltiples variables de diseño. De esta manera se van a obtener las variables que definen el mejor diseño a nivel conceptual de la embarcación.

CAPÍTULO 2

2. METODOLOGÍA

En este trabajo, el dimensionamiento incluyendo apéndices tipo flaps de embarcaciones rápidas se realiza mediante la implementación de un proceso de optimización implementado en el programa "FlapOptResisandSeakee.py" desarrollado en lenguaje Python. En este capítulo se describen las formulaciones empíricas utilizadas para el cálculo de la respuesta hidrodinámica que sirven para evaluar las funciones objetivo en el proceso de optimización. Primero, se realizó una investigación acerca de proyectos relacionados a optimización de lanchas rápidas con la finalidad de determinar las variables de diseño, restricciones y función objetivo consideradas. Se escogió el esquema de optimización de acuerdo con el tipo de problema, no lineal, y se desarrolló un sistema concatenando módulos de evaluación de restricciones y función objetivo con un software de tipo código abierto. De esta forma se determinan las dimensiones principales de la lancha incluyendo los parámetro de los flaps, y finalmente se desarrolló un esquema del diseño conceptual resultante de la embarcación.

2.1 Flaps

El estudio de embarcaciones planeadoras con flaps tomó importancia en el año de 1976 cuando los autores Savitsky & Brown (Savitsky & Brown, Procedures For Hydrodynamic Evaluation of Planing Hulls in Smooth and Rough Water, 1976) estudiaron los efectos de los flaps añadiendo la respuesta en olas irregulares. Previamente, se desarrollaron experimentos en tanques de pruebas para determinar las dimensiones efectivas de los flaps y encontrar sus ecuaciones empíricas gobernantes (Brown, 1971). Con el tiempo, el uso de los flaps se volvió muy popular en embarcaciones rápidas. Autores como (Mohamad Ayob A. F., 2012) y (Sakaki & Ghassemi, 2019) han realizado estudios acerca del dimensionamiento de planeadores con flaps considerando procesos de optimización en el que la función objetivo es la resistencia al avance. En estos estudios modernos, no se lleva a cabo el diseño conceptual de la embarcación, ni tampoco se incluye la respuesta en olas y su influencia en los pasajeros.

2.1.1 Ecuaciones Empíricas de los Flaps

Se describen los flaps como dispositivos de control de trimado que sirven para mejorar el desempeño del planeador a altas velocidades. Sus parámetros representativos son la longitud de la cuerda, el ancho y el ángulo de deflexión, Figura 1.1. Las ecuaciones para estimar la fuerza de levamiento, arrastre o resistencia, y, momento causados por los flaps sobre una embarcación planeadora fueron determinadas por Savitsky & Brown (Savitsky & Brown, Procedures For Hydrodynamic Evaluation of Planing Hulls in Smooth and Rough Water, 1976). A continuación se las va a resumir, de acuerdo con las necesidades en el proyecto actual.

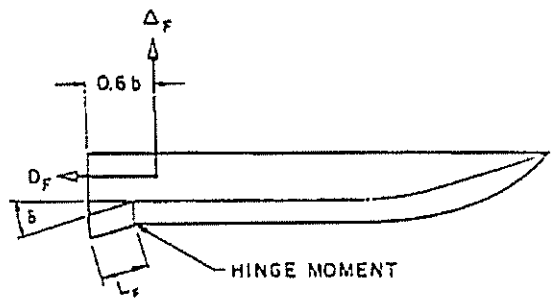


Figura 2.1 Equilibrio de fuerzas del sistema incluyendo los flaps

(Savitsky & Brown, Procedures For Hydrodynamic Evaluation of Planing Hulls in Smooth and Rough Water, 1976)

La fuerza normal al flap generada por el flap está determinada por la

(2.1:

$$\Delta_F = 0.046 L_F \, \delta \sigma b \left[\frac{\rho}{2} V^2\right] \qquad (2.1)$$

Debido a que el flap produce una fuerza perpendicular a su superficie, hay un incremento en la resistencia dada por su componente horizontal, $\Delta \tan \tau$, $D_F = 0.0052 \, \Delta_F \, (\tau + \delta)$ (2.2:

$$D_F = 0.0052 \, \Delta_F \, (\tau + \delta) \qquad (2.2)$$

donde τ es el ángulo de trimado de la lancha.

El momento hidrodinámico del flap respecto de un eje transversal, está dado por la $M_F = \Delta_F[0.6b + L_F(1\text{-}\sigma)]$ (2.3:

$$\mathbf{M_F = \Delta_F[0.6b + L_F(1 - \sigma)]} \quad \textbf{(2.3)}$$

De manera que el peso efectivo se determina mediante la $\Delta_e = \Delta\text{-}\Delta_F$ (2.4 :

$$\mathbf{\Delta_e = \Delta - \Delta_F} \quad \textbf{(2.4)}$$

Y la nueva posición del centro de gravedad, lcg, o posición efectiva es calculada mediante la $lcg_e = [\Delta(lcg)\text{-}0.6b\Delta_F]/\Delta_e$ (2.5:

$$\mathbf{lcg_e = [\Delta(lcg) - 0.6b\Delta_F]/\Delta_e} \quad \textbf{(2.5)}$$

En donde,

- Δ_F Incremento del levantamiento, lb
- D_F Incremento en la resistencia, lb
- M_F Momento debido al flap, ft-lb
- L_F Cuerda del flap, ft
- σ Razón longitud-cuerda del flap
- δ Deflección del flap, deg
- τ Ángulo de trimado , deg
- V Velocidad, ft per sec

2.2 Optimización de Embarcaciones Rápidas

Como se explicó anteriormente, la dinámica de un bote planeador es compleja y deben considerarse varios componentes, en este caso la resistencia al avance y la respuesta en olas. El diseño óptimo de un sistema de control trimado puede considerarse como un problema multiobjetivo. Existen diferentes tipos de algoritmos y tipos de optimización de las cuales se habla en el siguiente apartado.

2.2.1 Introducción a la Optimización de Embarcaciones Rápidas

Los problemas de optimización en general sirven para maximizar o minimizar una función.

$$\begin{aligned} &\text{Minimizar} && f(x) \\ &\quad x \\ &\text{Sujeto a } g_i(x) && x \in X \end{aligned}$$

donde, $\bar{x}$ representa un punto de diseño y puede ser representado como un vector de valores correspondiente a diferentes variables de diseño. Los elementos de este vector pueden ser ajustadas para minimizar la función f. Y, todos estos valores que se encuentran dentro de la zona factible y minimizan la función objetivo son llamadas soluciones o mínimos (Kochenderfer & Wheeler, 2019). La zona factible está definida como el conjunto de posibles soluciones que está contenida en el límite de una o varias restricciones.

$$\begin{aligned} &\text{Minimizar} && f(x_1, x_2) \\ &\quad x_1, x_2 \\ &\text{Sujeto a} && x_1 \geq 0 \\ &&& x_2 \geq 0 \\ &&& x_1, x_2 \leq 1 \end{aligned}$$

Modelar un problema con formulaciones matemáticas es un proceso complejo sobre todo cuando se trata de evitar los mínimos locales de la función o funciones. La modelación del proceso de optimización puede resultar en que el proceso de encontrar la solución sea fácil o complejo. Por tanto, es conveniente la correcta elección del proceso que se ajuste a la complejidad del problema.

Una manera de seleccionar el proceso es realizando una búsqueda bibliográfica de optimizaciones de embarcaciones rápidas para conocer de las experiencias en esquemas similares. Esto con el fin de determinar las variables de diseño,

restricciones y funciones objetivo para el problema actual. A continuación se resume los procesos de optimización que varios autores han realizado respectivo al tema.

Tabla 2.1 Referencias de trabajos de optimización a embarcaciones rápidas

	Autor	Titulo	Conferencia
1	Mohamad Ahmad F.	The Design of HSC Using an Optimization Framework	
2	Castro-Feliciano Esteban L.	Co-Design of Planing Craft and Active Control Systems	Tesis Doctoral
3	Knight Joshua T	Multiobjective Particle Swarm Optimization of a Planing Craft with Uncertainty	Journal of Ship Production and Design
4	Bréfort Dorian	Managing Epistemic Uncertainty in Design Models through Type-2 Fuzzy Logic Multidisciplinary Optimization	Tesis Doctoral
5	Sakaki Abdollah, Ghassemi Hassan	Evaluation of the Hydrodynamic Performance of Planing Boat with Trim Tab and Interceptor and Its Optimization Using, Genetic Algorithm	Journal Of Marine Science And Application
6	Villamarín Edgar G	Propuesta de mejora del comportamiento dinámico de lanchas interislas de Galápagos orientado al confort de pasajeros	Tesis de Grado

Tabla 2.2. Funciones Objetivo, Variables de decisión y Restricciones

	Función Objetivo	Variables de decisión	Restricciones
1	f (1) = Rt	L, B, T	g(1)= Disp, g(2)= GM, g(3)= L/V^1/3, g(4)= le, g(5)= L/B, g(6)= B/T, g(7)= l
2	f(1) = Rt + Seakeeping	LCG, L, β, B	g(1) = ángulo trimado, g(2)= β
3	f(1) = Drag, f(2)= aCG	L, LCG, B, β	g(1)= L/B , g(2)= ángulo trimado , g(3) = T
4	f(1) = Seakeeping, f(2) = Rt	LCG, Disp, Vk, h1/3	g(1)= L/B, g(2)= Cv, g(3)= ángulo trimado, g(4)=L, g(5)= B, g(6)= β
5	f(1) = Rt, f(2) = trim angle	LCG, V, β	
6	f(1)= Rt, f(2)=aCG	L, Cb, B, LCG, β	g(1)=GMT/GMTx , g(2)= Acargo/Acargox, g(3)= Fcbo/Fcbox, g(4)=L/B, g(5)= ángulo trimado

Tabla 2.3. Tipos de optimización empleados por los autores

	Método de Optimización	Tipo de Optimización	Agrupación	Ejemplo Utilizado
1	NSGA-II, IDEA, EASDS	Optimización de Objetivo Único	Evolutivo, población/genético	Optimización de una embarcación planeadora costera
2	MAWS	Optimización de Objetivo Único	Vectores	Optimización de una embarcación con 5 variables de decisión
3	PSO	Optimización Multiobjetivo	Población/Partículas	Optimización de una embarcación rápida
4	2-Fuzzy Logic	Optimización Multidisciplinaria	Incertidumbre	Optimización de una embarcación en diseño conceptual
5	GA	Optimización Multiobjetivo	Población/Genético	Optimización de una embarcación planeadora con flaps

	Método de Optimización	Tipo de Optimización	Agrupación	Ejemplo Utilizado
6	Direcciones Factibles	Optimización Multiobjetivo	Derivada/ Gradiente	Optimización de una embarcación rápida que navega en un estado de mar

De acuerdo con los trabajos revisados, últimamente se están utilizando métodos basados en población. Su uso se debe a que estos algoritmos son más sofisticados y elitistas cuando se requiere encontrar la solución o el conjunto de soluciones. Por lo cual, en este proyecto se realizó el proceso de optimización basado en algoritmos basados en población, específicamente en algoritmos genéticos.

2.2.2 Métodos de Población: Algoritmos Genéticos

Los métodos de optimización basados en población reducen la posibilidad de que la función se estanque en un mínimo local. Generalmente, estos métodos utilizan una población inicial los cuales son una colección de las variables de diseño. La población inicial se esparce por todo el espacio del diseño incrementando las posibilidades de encontrar un óptimo global sin estancarse en los mínimos locales (Kochenderfer & Wheeler, 2019).

El algoritmo genético (GA) (Goldberg, 1989) es un método estocástico inspirado en la evolución genética y el ADN. En el que el conjunto de soluciones se encuentra mediante la selección y mutación de los individuos mediante la evolución de los cromosomas que se realiza a través de las iteraciones. Los cromosomas poseen la información de la solución de cada individuo y es codificada de manera binaria, para finalmente asignar la función de aptitud (Función Objetivo). De manera que, los cromosomas con mayor puntuación representan una solución eficaz para el problema.

La presencia de múltiples objetivos en un problema conlleva a que existan múltiples soluciones o también conocido como "Pareto Front", debido a la cantidad de posibles soluciones. En estos casos, el usuario se ve en la obligación de encontrar

el mayor número de "Pareto optimal Solutions" dado que no hay manera de evaluar cual conjunto es mejor frente a otro.

2.2.3 Non-Dominated Sorting Genetic Algorithm (NSGA-II)

El proceso de optimización a ser aplicado en este proyecto es desarrollado a través de los paquetes *"OpenPlaning"* y *"Pymoo"* desarrollados en lenguaje de programación Python. *"OpenPlaning"* es un software de código abierto desarrollado por (Castro Feliciano, Octubre 2021) bajo la licencia MIT en el que se implementa el método de Savitsky (Savitsky , Hydrodynamic Design of Planing Hulls, 1964). *"Pymoo"* es un paquete de Python de código abierto desarrollado por (Blank & Deb, 2020) para desarrollar problemas de optimización de único y múltiple objetivo. El algoritmo utilizado para la optimización multiobjetivo es de tipo evolutivo, especificamente hablando se trata del algoritmo Non-dominated Sorting Genetic Algorithm (NSGA-II).

La habilidad de encontrar múltiples soluciones para un problema multiobjetivo puede ser implementado como una modificación del algoritmo genético. Dado que en los algoritmos de evolución (EA) se emplea una población inicial para obtener el conjunto de soluciones, y sucesivamente para obtener un conjunto del conjunto de soluciones, un EA puede ser modificado para obtener el Pareto Front óptimo en una solo simulación.

El algoritmo NSGA-II fue propuesto por (Deb, Pratap, Agarwal, & Meyarivan, April 2002) en el que se introduce el concepto de solución dominada y no dominada. Y se define de la siguiente manera: La solución x^1 domina a la solución x^2 si se cumple:

1. La solución x1 no es peor que la solución x2 en todos los objetivos.
2. La solución x1 es estrictamente mejor que la solución x2 en al menos un objetivo.

Si una de las dos condiciones no se cumple, entonces la solución x^1 no domina a la solución x^2.

Cuando se realiza este análisis de soluciones, y se encuentra que la primera condición no se cumple para ninguna de las soluciones (x^1 y x^2) no se puede concluir acerca de la dominancia de la una con respecto de la otra. Cuando esto sucede, se conoce a las soluciones como no-dominadas.

El algoritmo NSGA-II funciona a partir de una población $(P_t)_N$ combinada con una descendiente de esta misma $(Q_t)_N$ para formar $(R_t)_{2N}$. La población R_t de dimensión 2N es clasificada mediante el ordenamiento no-dominado, y se determinan los Pareto fronts de este nuevo conjunto (F_1, F_2, ... , etc.) siendo F_1 la mejor solución de la combinación y en comparación con las otros Paretos.

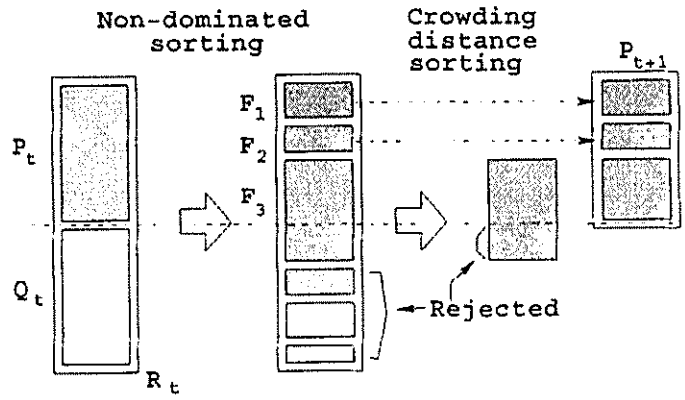


Figura 2.2 Algoritmo NSGA-II procedimiento

Si el conjunto F_1 es menor que N, todas las soluciones entran al nuevo conjunto P_{t+1}, caso contrario para completar el conjunto N se utilizan los conjuntos F_2, F_3, ...F_i, etc. Debido a que el nuevo conjunto P_{t+1} es de dimensión N, la manera en que se ordenan las soluciones del conjunto F_i mediante el operador de comparación α_n. Finalmente el conjunto P_{t+1} es utilizado para selección, combinación y mutación para crear una nueva población, Figura 2.2.

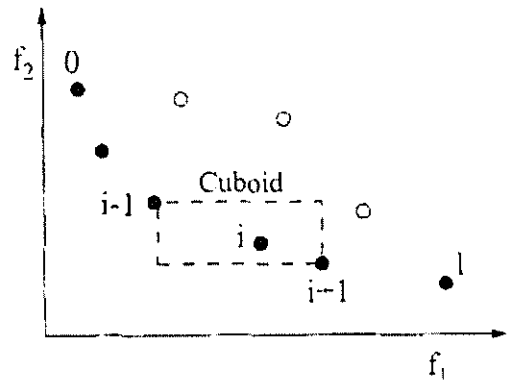


Figura 2.3 Cálculo de la distancia de amontonamiento.

En el algoritmo NSGA-II se mantiene un buen espaciamiento entre soluciones. A diferencia de otros algoritmos en donde se tiene que ingresar una función de espaciamiento, en NSGA-II dicha función es reemplazada por el "Crowed-Comparison Operator" (α_n), de manera que, si dos soluciones pertenecientes a diferentes "fronts" se escogerá la que tenga menor (mejor) rango. Y si, las soluciones pertenecen al mismo "front" se escogerá la que este en la solución menos amontonada, Figura 2.3. p

Entonces, el algoritmo de ordenamiento no dominado se escribe como:

$R_t = P_t \cup Q_t$	Combina la población de padres con la población de descendencia
$F = fast\ non - dominated\ sort\ (R_t)$	$F = (F_1, F_2, \dots)$, Todas las soluciones no dominadas del conjunto R_t
$P_{t+1} = \emptyset\ and\ i = 1$	
$Until\ \lvert P_{t+1}\rvert + \lvert F_i\rvert \leq N$	Hasta que la población es llena
$Crowing\ distance\ assigment\ (F_1)$	Se Calcula la distancia de amontonamiento en F_1
$P_{t+1} = P_{t+1} \cup F_i$	E incluye i enésimo no dominado en el nuevo Pareto
$i = i + 1$	

$Sort\ (F_1, \alpha_n)$	Ordena en forma descendiente usando el operador α_n
$P_{t+1} = P_{t+1} \cup F_i[1:(N - \lvert P_{t+1}\rvert)]$	Escoge el primer $(N - \lvert P_{t+1}\rvert)$ elementos de F_i
$Q_{t+1} = make\ new\ pop\ (P_{t+1})$	Usa selección, combinación y mutación para crear la nueva población Q_{t+1}
$t = t + 1$	Incremento en el contador de generaciones

En el programa "FlapOptResisandSeakee.py" se trabajó con un número de población de 200, y criterio de terminación basado en la convergencia de los valores de los puntos del frente de Pareto. A continuación se describen los parámetros utilizados para el proceso de optimización.

2.2.4 Funciones Objetivo, Variables de Diseño, Restricciones

Para las funciones objetivos, se considera de manera simultánea la resistencia al avance y la aceleración vertical del centro de gravedad de manera normalizada.

$$\underset{\boldsymbol{x}}{Minimizar} \quad \frac{f_1(\boldsymbol{x})}{f_{1_0}}; \quad \frac{f_2(\boldsymbol{x})}{f_{2_0}}$$

En donde $f_1(\boldsymbol{x})$, $\mathrm{f_1(x)} = \mathrm{R_{hidrodinamica}} + \mathrm{R_{friccional}} + \mathrm{R_{aire}} + \mathrm{R_{fuerza\ del\ flap}} + \mathrm{R_{whisper\ spray}} + \mathrm{R_{olas}}$

(2.6, es la resistencia al avance (R_t) que se calcula siguiendo el método de Savitsky (Savitsky , Hydrodynamic Design of Planing Hulls, 1964). A ella se añade la resistencia friccional, resistencia por aire, resistencia por los flaps (Savitsky & Brown, Procedures For Hydrodynamic Evaluation of Planing Hulls in

Smooth and Rough Water, 1976), la componente del whisper spray (Savitsky , DeLorme, & Datla, Inclusion of Whisper Spray in Drag in Preformance Prediction Method for High-Speed Planing Hulls, January 2007), y finalmente la componente al navegar en olas (Savitsky & Brown, Procedures For Hydrodynamic Evaluation of Planing Hulls in Smooth and Rough Water, 1976).

$$f_1(x) = R_{hidrodinamica} + R_{friccional} + R_{aire} + R_{fuerza\,del\,flap} + R_{whisper\,spray} + R_{olas} \quad \textbf{(2.6)}$$

La función $f_2(x)$ representa la aceleración del centro de gravedad (acg) (Savitsky & Brown, Procedures For Hydrodynamic Evaluation of Planing Hulls in Smooth and Rough Water, 1976), $f_2(x) = 0.0104\left(\frac{H_{\frac{1}{3}}}{b} + 0.084\right)\left(\frac{\frac{L}{b}}{C_\Delta}\right)\frac{\tau}{4}\left(\frac{5}{3} - \frac{\beta}{30}\right)\left(\frac{V_k}{\sqrt{L}}\right)^2$

(2.7 :

$$f_2(x) = \mathbf{0.0104}\left(\frac{H_{\frac{1}{3}}}{b} + \mathbf{0.084}\right)\left(\frac{\frac{L}{b}}{C_\Delta}\right)\frac{\tau}{4}\left(\frac{5}{3} - \frac{\beta}{30}\right)\left(\frac{V_k}{\sqrt{L}}\right)^2 \quad \textbf{(2.7)}$$

En donde,

$H_{\frac{1}{3}}$	Altura significativa de la ola, ft
b	Manga en la china, en la estación 5, ft
L	Eslora Total, ft
C_Δ	$\frac{\Delta}{wb^3}$
τ	Ángulo de trimado, grados
β	Ángulo de astilla muerta, grados
V_k	Velocidad en nudos, nudos

Las variables de diseño están representadas por x, conjunto compuesto por:

$$x = [LOA, b, lcg(\%L), \beta, \delta, L_F\,]$$

En donde,

L Eslora total, m

b Manga, m

lcg Posición longitudinal del centro de gravedad, medido desde el espejo, m

β Ángulo de astilla muerta, deg

δ Ángulo de deflección del flap, deg

L_F Cuerda del flap, m

Se consideran los siguientes rangos para estas variables:

$$9.5 \leq LOA \leq 16$$
$$3 \leq b \leq 5$$
$$0.3 \leq lcg(\%L) \leq 0.5$$
$$10 \leq \beta \leq 30$$
$$0 \leq \delta \leq 15$$
$$0.15 \leq L_F \leq 0.211$$

Las restricciones son de tipo desigualdad ($g(x) \leq 0$) y se consideran las siguientes:

$$Sujeto\ a: \quad g_1(\boldsymbol{x}) = 2 \leq \tau(x) \leq \tau_{porpoising}$$
$$g_2(\boldsymbol{x}) = L_k(x) \leq 0.9 * LOA$$
$$g_3(\boldsymbol{x}) = 3 \leq \frac{L}{b} \leq 5$$
$$g_4(\boldsymbol{x}) = 0.4 \leq KG \leq KG_{DynMoment}$$
$$g_5(\boldsymbol{x}) = FreeBoard \geq 0.68$$
$$g_6(\boldsymbol{x}) = PassArea \geq 21.11$$

En donde,

L_k Longitud de la quilla mojada, m

$\tau_{propising}$ Ángulo de trimado dinámico límite

KG Centro de Gravedad vertical

$KG_{DynMoment}$ KG máximo para que el momento dinámico se estabilice

$FreeBoard$ Franco bordo

$PassArea$ Área mínima requerida para acomodación de pasajeros

.3 Análisis de Confort en Pasajeros

La norma (ISO 2631-1, Reviewed 2021) es una regulación internacional que evalúa la influencia del movimiento de una embarcación sobre el confort de pasajeros. La probabilidad de ocurrencia de síntomas de mareo incrementa si el tiempo de exposición alcanza horas de viaje elevadas. Se determina el parámetro de incidencia al movimiento (MSI) como la relación entre la aceleración vertical y el tiempo de exposición, y que sirve para determinar el porcentaje de pasajeros mareados en el trayecto de un viaje. De manera que la normativa ISO 2631, propone la siguiente expresión para el cálculo de esta misma, $\%\mathrm{MSI} = \mathrm{km}.\mathrm{a_{CG}}.\sqrt{\mathrm{T}}$ (2.8 :

$$\%MSI = km.a_{CG}.\sqrt{T} \tag{2.8}$$

En donde,

km Factor que depende de la población, 1/3 para población mixta

a_{CG} Aceleración del centro de gravedad, m/s^2

T Periodo total de exposición, segundos

Mientras que, el MSI es un parámetro adimensional, su porcentaje dependerá de los individuos expuestos al movimiento. Ciertos individuos serán más propensos a marearse que otros, siendo el sexo femenino el más afectado por el movimiento.

2.4 Diseño Conceptual de Embarcaciones Rápidas

En este subcapítulo se describen los modelos conceptuales realizados anteriormente por Villamarín (2020) de tal manera que la comparación con el nuevo modelo generado pueda realizarse en el siguiente capítulo.

2.4.1 Modelo Original

El modelo original sirve como lancha base para el proceso de optimización, recordando del capítulo 1 que estas lanchas sirven para el transporte interislas Santa Cruz – Isabela y entre Santa Cruz – San Cristóbal, a una velocidad superior a los 28 nudos. En la Tabla 2.4 se presentan sus dimensiones principales.

Tabla 2.4 Dimensiones Principales de Lancha Interislas

	Lancha Angy
LOA , m	12.02
B_{max}, m	3.45
D, m	1.51
T_{DWL}, m	0.51
lcg, m	4.80
$Pasajeros$	24
BHP, HP	850
$Desplazamiento$, ton	8.26
β, deg	14

La sección transversal de la lancha es de tipo "V" típico de las embarcaciones planeadoras (Savitsky & Brown, Procedures For Hydrodynamic Evaluation of Planing Hulls in Smooth and Rough Water, 1976). Posee doble china para evitar el ascenso el agua hacia las cubiertas.

2.4.2 Modelo Optimizado sin Flaps

El modelo de lancha optimizado en el proyecto de embarcaciones rápidas fue realizado por el ingeniero (Villamarín, 2020).Dicho modelo presenta cambios significativos en las características principales, siendo el movimiento del lcg el parámetro más importante en el desarrollo de este nuevo diseño conceptual. A continuación se describen las características principales del modelo desarrollado previamente.

Tabla 2.5 Dimensiones Principales de Lancha Optimizada

(Villamarín, 2020)

	Lancha Opt
L_{wl}, m	13.66
B_{max}, m	3.44
D , m	1.55
$Pasajeros$	24
Δ , ton	8.52
β , deg	22
lcg , m	5.58
R_T , N	16205
$RMS\ a_{CG}$, g	0.98

A diferencia del modelo original, el diseño conceptual de esta embarcación realizada por el ingeniero Villamarín solo posee una china. Mientras que la sección se sigue manteniendo en forma de "V", continuando con las recomendaciones realizadas por Savitsky y Brown, (Savitsky & Brown, Procedures For Hydrodynamic Evaluation of Planing Hulls in Smooth and Rough Water, 1976)

CAPÍTULO 3

3. RESULTADOS Y ANÁLISIS

Mediante la ejecución del programa *"FlapOptResisandSeakee.py"* se obtuvieron las variables de diseño optimizadas. En este capítulo se emplean las formulaciones descritas en el capítulo 2 para el cálculo de la resistencia al avance y el comportamiento el olas mediante un software empleado en el lenguaje Python. Se obtuvo el espacio de las variables de diseño y las mejores soluciones contenidas en el frente de Pareto. La determinación de las variables de diseño adecuadas para el problema fue desarrollada considerando reducción en la función objetivo de aceleración vertical y resistencia al avance. Finalmente, se desarrolló el modelo conceptual con los resultados obtenidos en el proceso de optimización y se procedió con la comparación entre la lancha original y el diseño realizado en el proyecto interislas anterior.

3.1 Pareto Óptimo

En un proceso de optimización multiobjetivo no existe una sola solución que satisfaga ambas funciones objetivo. A su vez, existe un conjunto de soluciones que representan la solución para las funciones objetivo. Este conjunto de soluciones es llamado *"Pareto Front"*. Un Pareto óptimo es el conjunto de soluciones en donde es imposible mejorar un objetivo sin empeorar al menos algún otro objetivo. Como resultado del algoritmo NSGA-II se estimó el verdadero Pareto óptimo del problema.

El algoritmo implementado en este trabajo se detiene luego de 95 generaciones (un total de 19000 evaluaciones a las funciones objetivo), con un tiempo reportado de 553 segundos. Antes de iniciar con el proceso de optimización el algoritmo calcula la respuesta hidrodinámica de la lancha y el comportamiento en olas, estos resultados son impresos en la pantalla del ordenador, Tabla 3.1. Mientras que, las variables obtenidas en la optimización son salvadas en el ordenador en un documento con formato Excel "Out.xlsx". Se presentan los resultados de la ejecución del programa considerando una altura de ola de 1 metro.

Tabla 3.1 Resistencia Total y Aceleración Vertical Antes de la Optimización

H_{sig}, m	1
Resistencia al Avance Total, N	17227.62
Aceleración de Impacto Promedio, g's	1.14

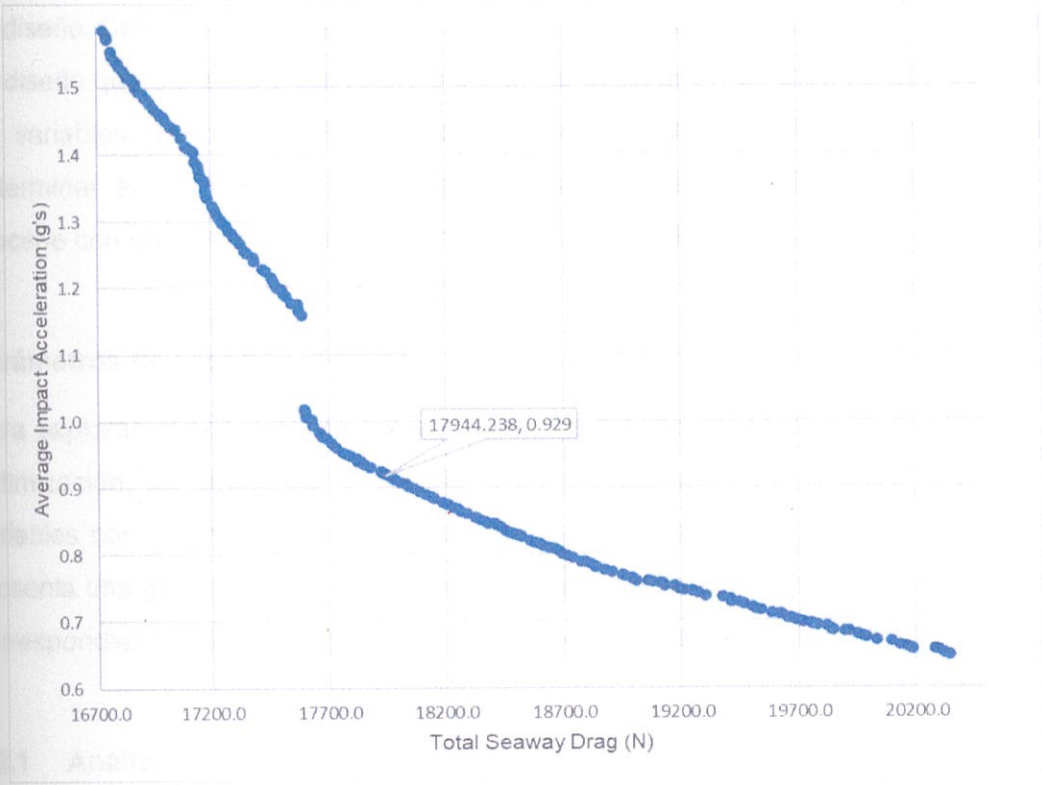


Figura 3.1 Pareto Front Estimado en la Optimización de Lanchas Interislas

3.1.1 Análisis del Pareto Front Estimado

En el presente trabajo, el Pareto front estimado es de tipo discontinuo, Figura 3.1. Desde el valor de resistencia reportado en la Tabla 3.1 hasta el valor más bajo en la gráfica se consiguió una mejora del 5%. Mientras que para la aceleración, del valor reportado hasta el valor más bajo se consiguió una mejora del 44%. Se puede observar que la lancha de la Tabla 3.1 no se encuentra en el espacio de soluciones óptimas ni el Pareto estimado, esto se debe a que esta embarcación no cumple las restricciones 5 y 6 (Francobordo y área de carga para pasajeros).

La gráfica muestra que es imposible mejorar la aceleración del centro de gravedad sin eventualmente aumentar la resistencia al avance. Por lo tanto, para un diseño conceptual lo más razonable es escoger la parte más baja del Pareto de manera que la reducción en ambas funciones objetivos sea equivalente.

La disminución en las funciones objetivos fue dada por el cambio en los parámetros de diseño. Cada solución del Pareto está representada por un conjunto de variables de diseño que satisfacen las restricciones. Se obtuvieron un total de 200 conjuntos de variables de diseño, cada conjunto conformado por 6 parámetros. Para determinar el set de variables que se utilizaron para el diseño conceptual se procede con un análisis de estos conjuntos presentado en el siguiente subcapítulo.

2 Parámetros de Diseño Óptimos

Para explorar el conjunto de las variables de diseño obtenidas en el proceso de optimización, se presenta el Pareto front correspondiente a cada una de las variables con un mapa de color diferente, ver Figura 3.2, Figura 3.3. Además se presenta una gráfica de las variables frente a otras variables con un mapa de color correspondiente a la aceleración vertical del centro de gravedad, ver Figura 3.4.

3.2.1 Análisis de las variables de diseño

Se observó que el desempeño en el mar del bote planeador es altamente afectado por los parámetros lcg y ángulo de astilla muerta β. Una ubicación del lcg hacia proa reduce significativamente la aceleración del centro del gravedad, aCG cuando se ubica (%L) > 0.34, Figura 3.2, Figura 3.4. Con respecto al ángulo de astilla muerta β , se observó que el conjunto de variables se sitúa radicalmente en 10° o en 30° sin ningún resultado intermedio, Figura 3.3 y Figura 3.4, consiguiendo la máxima reducción en aceleración cuando $\beta = 30°$. También se observó una reducción de aCG con el ángulo de astilla muerta $\beta = 10°$, pero esto sucede únicamente cuando los valores de lcg son cercanos al 40% de la eslora.

Significando que, se consigue una máxima reducción de la función objetivo aceleración del centro de gravedad con la variación del lcg, β, y la combinación de ambos. Los resultados llevan concordancia con lo expuesto por (Savitsky & Koelbel , Seakeeping of Hard Chine Hulls, 1993) en el capítulo 8 de su informe.

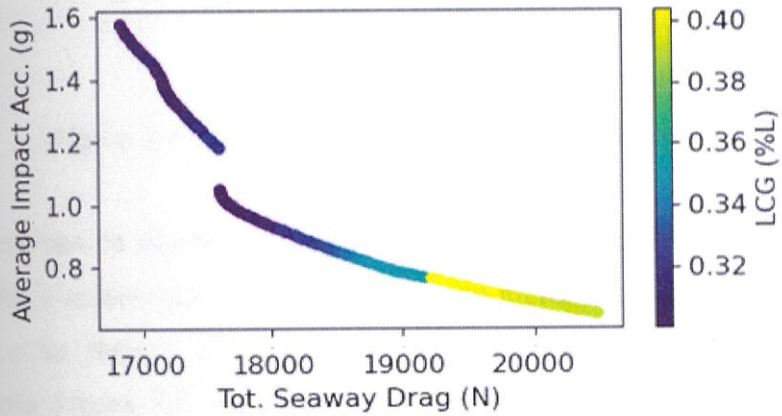


Figura 3.2 Pareto Front Estimado de la variable lcg (%L)

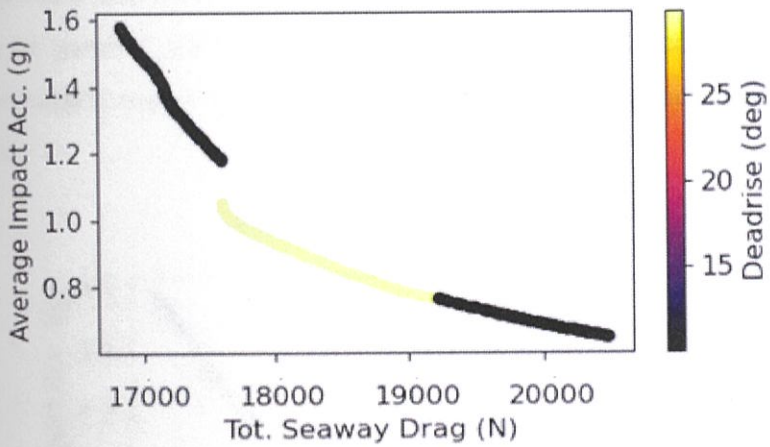


Figura 3.3 Pareto Front Estimado de la variable Ángulo de Astilla Muerta β

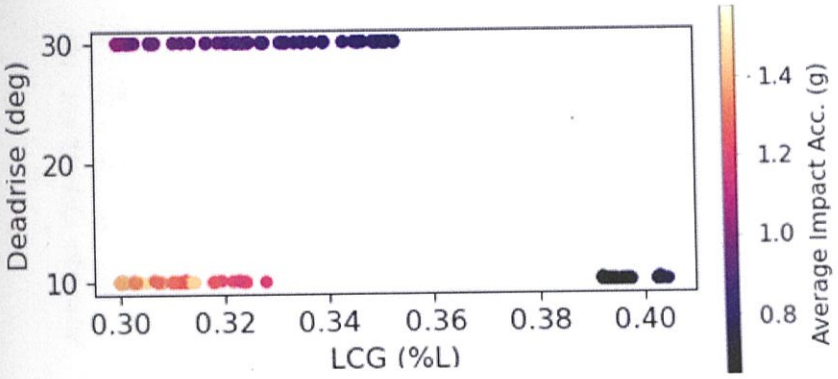


Figura 3.4 Variables Estimadas del Pareto Front

Para las variables de diseño eslora y manga se observa que se obtiene una mayor reducción de la aceleración cuando la eslora se incrementa $L > 13.5$ m, Figura 3.5, y la manga se reduce $B < 3.6$ m, Figura 3.6, es decir cuando la relación L/B aumenta, Figura 3.7, resultado que concuerda con lo reportado por Savitsky (Savitsky & Brown, Procedures For Hydrodynamic Evaluation of Planing Hulls in Smooth and Rough Water, 1976) en el capítulo 8 en el que se expone, una manga más estrecha reduce la aceleración y un aumento en la eslora mejora el movimiento de cabeceo en general. En la Figura 3.7 se observa que la combinación de estas variables produce la mayor reducción de la función objetivo aceleración del centro de gravedad.

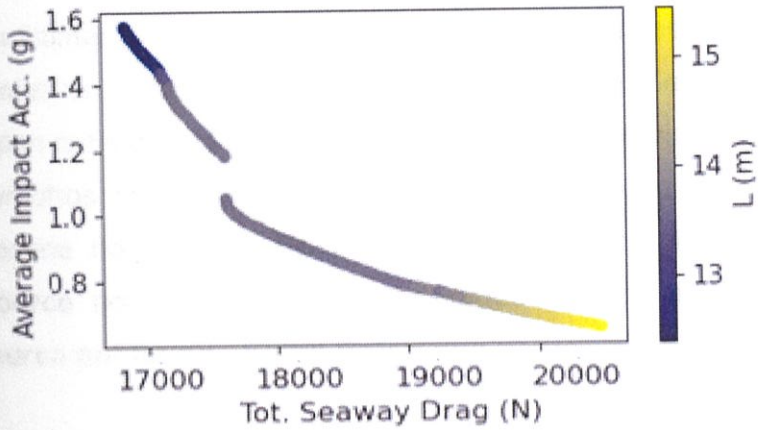


Figura 3.5 Pareto Front Estimado de la variable Eslora

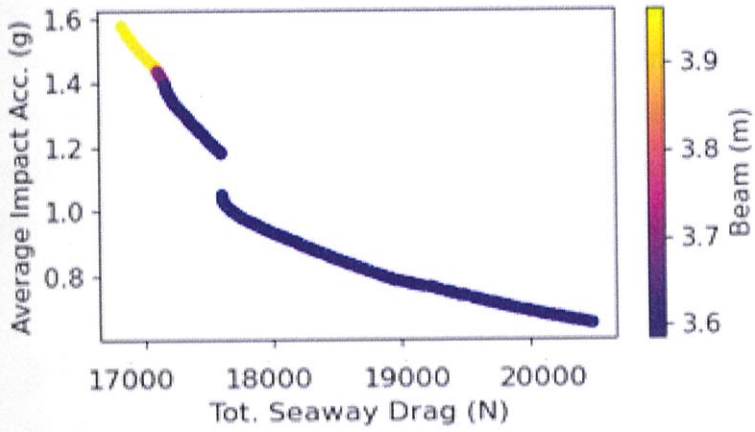


Figura 3.6 Pareto Front Estimado de la Manga

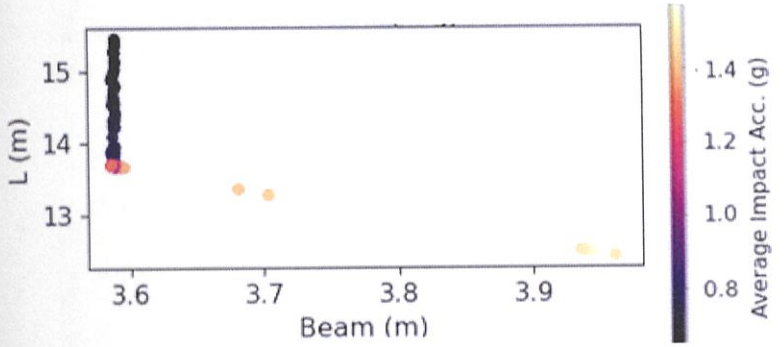


Figura 3.7 Variables Estimadas del Pareto Front

3.3 Efecto del Flap en las Embarcaciones Rápidas

Finalmente, las últimas dos variables de diseño correspondientes a las dimensiones del flap se analizaron obteniendo los siguientes resultados. Se consideró un flap con relación span radio σ fijo del 0.40, debido al espacio requerido por los motores. Resultados con otros valores de esta relación se presentan en el anexo A, **Error! Reference source not found.**, **Error! Reference source not found.**, **Error! Reference source not found.**, **Error! Reference source not found.**, **Error! Reference source not found.**, **Error! Reference source not found.**.

Se observó como el ángulo de deflexión del flap δ reduce considerablemente la función objetivo aceleración del centro de gravedad. Cuando $\delta > 8°$ se obtiene la menor reducción, Figura 3.8. Para la cuerda, se observa que puede existir una

variación de límite superior a inferior. Siendo el límite superior el más repetido a lo largo de la curva. Lo que significa que una mayor cuerda reduce considerablemente la función objetivo. Cuando el ángulo de deflección $\delta > 8°$ es cuando se conseguirá una menor aceleración, esto puede ser observado en la Figura 3.9 inferior.

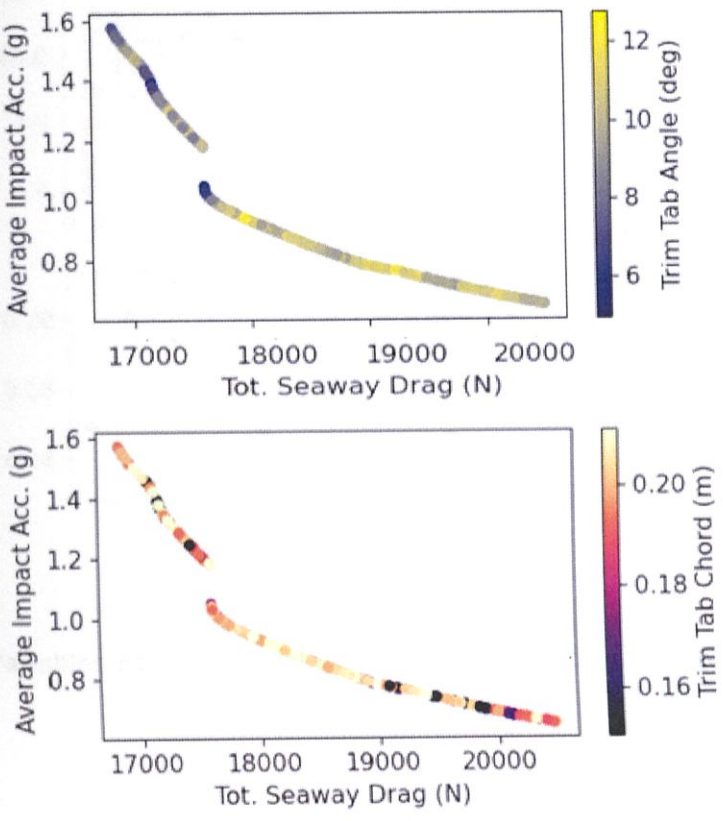


Figura 3.8 Pareto Front Estimado de las Variables, Superior, Ángulo de Deflección δ, Inferior, Variable Cuerda del Flap.

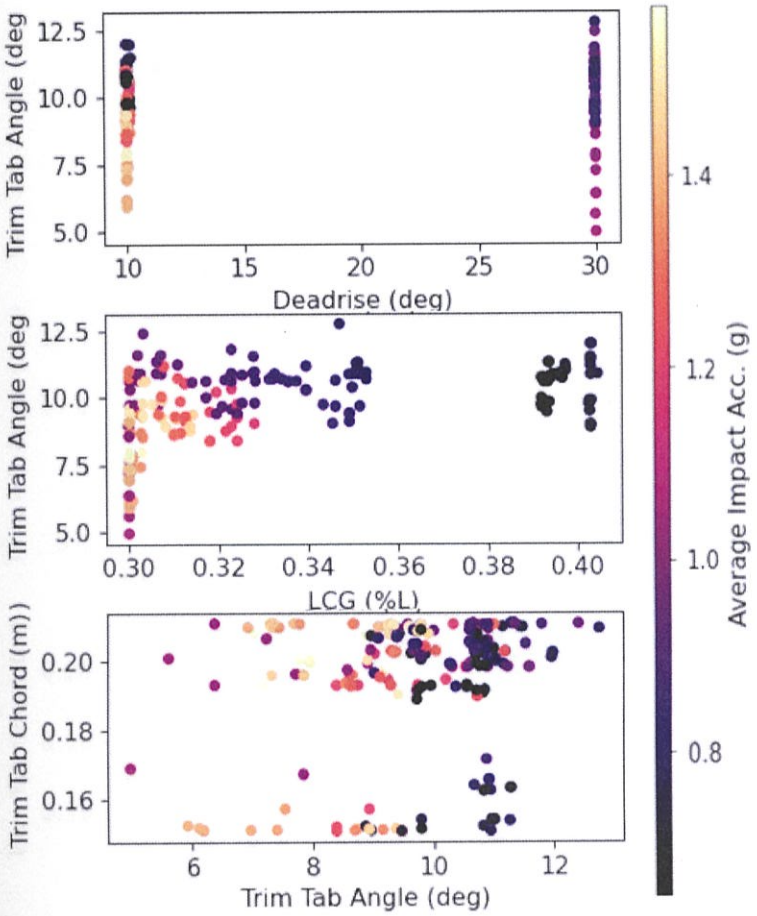


Figura 3.9 Variables Estimadas del Pareto Front Función Objetivo a_{CG}

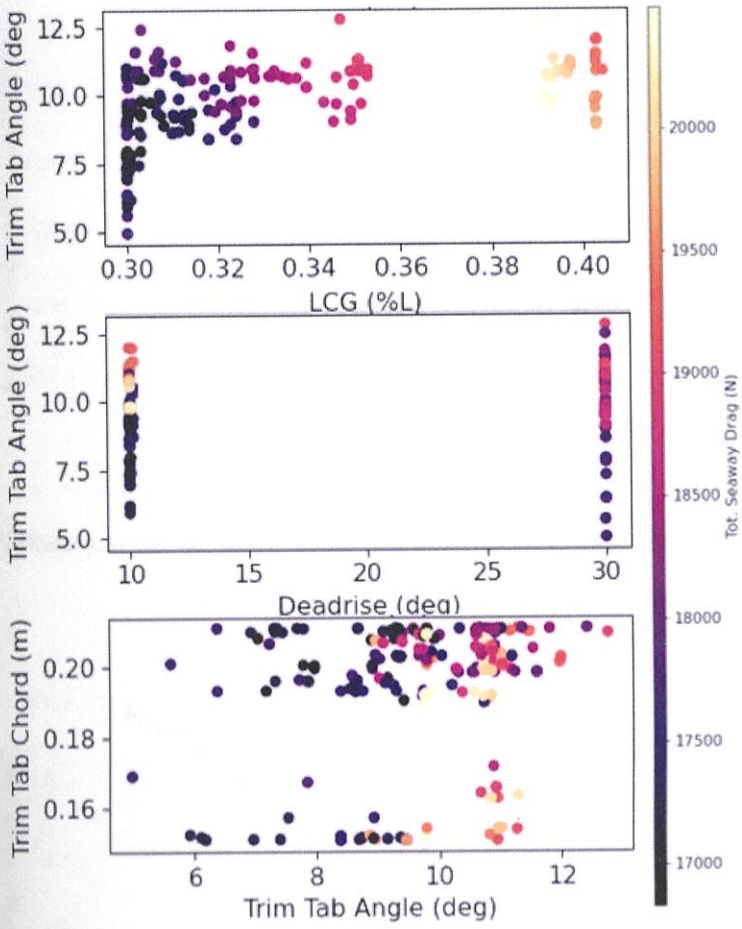


Figura 3.10 Variables Estimadas del Pareto Front Función Objetivo R_T

La razón longitud/cuerda del flap es otra variable que influye en la reducción de la aceleración. Se observó que mientras el span se reduce, el flap necesita mayor ángulo de deflección para reducir la aceleración Figura 3.12. Esto se explica debido a que el efecto del flap es reducir el ángulo de trimado τ mediante el levantamiento producido por el flap, un mayor "*flap lift*" se consigue aumentando el área del flap o aumentando el ángulo de deflección δ . Entonces, si el área se está reduciendo se puede esperar que el proceso de optimización aumente la deflexión para conseguir un mayor "*flap lift*". Esto lleva a una reducción del ángulo de trimado τ y finalmente de la aceleración del CG, Figura 3.11.

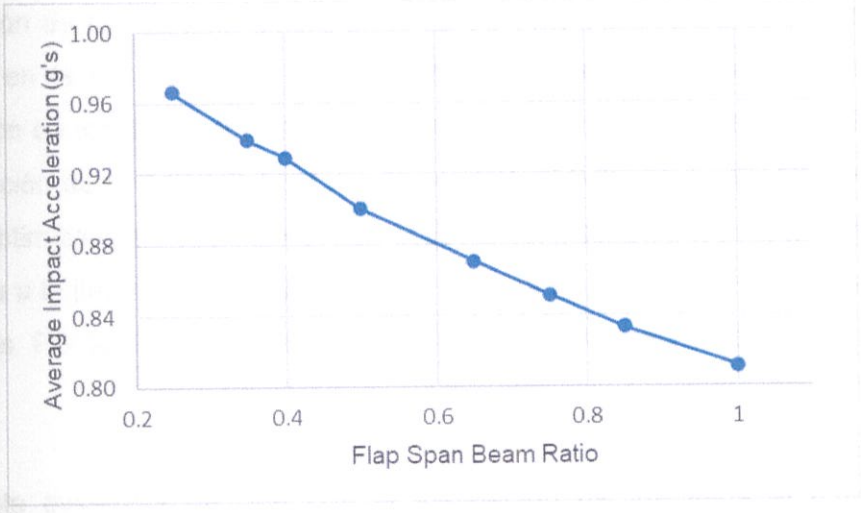


Figura 3.11 Influencia de la razón longitud/cuerda del Flap y la Aceleración

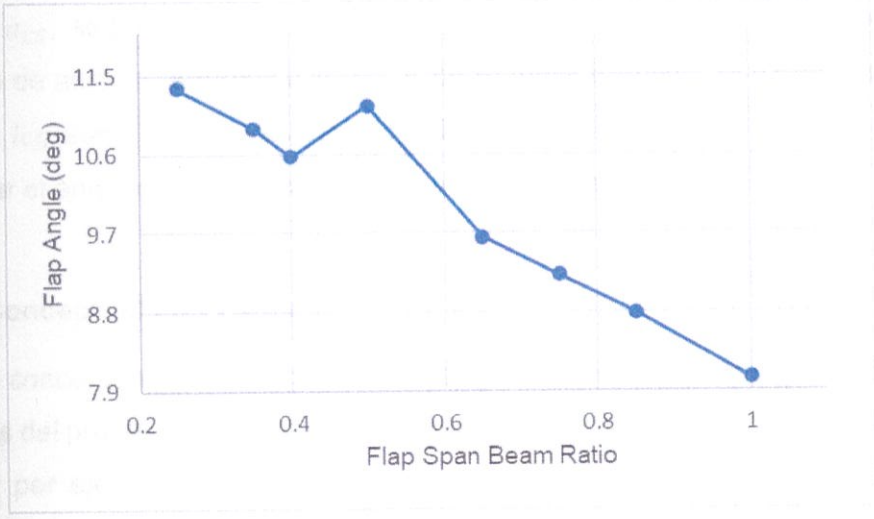


Figura 3.12 Influencia del Flap Span Beam Ratio y el Ángulo de Deflexión

El flap cumple un rol muy importante en la optimización de las embarcaciones rápidas. Las combinaciones de los parámetros principales lcg, β y δ que reducen la función objetivo aceleración, principalmente se dan cuando δ > 10°, β= 30° y una locación del lcg > 35% L. Mientras que, las combinaciones que reducen la función objetivo resistencia total se dan cuando δ < 10°, β= 10° y una ubicación del lcg < 30% L. Esto es consistente con el problema de optimización multiobjetivo y el método de optimización.

La relación de las características del flap con las otras variables de diseño se observa en la Figura 3.9 y Figura 3.10. Inicialmente se planteó que con la instalación de los flaps se conseguiría un movimiento equivalente del lcg: En la embarcación de la Tabla 2.4, el lcg se reportó al 37% de L, mientras que en la lancha optimizada sin flaps al 40% de L. En ambos casos L representa la longitud de eslora a la línea de agua, y difieren una de la otra por ser dos embarcaciones diferentes. Por tanto, una comparación debe considerarse en el mismo rango de longitud.

No existe evidencia de que con la instalación de los flaps el lcg sufra un movimiento hacia proa, pero si se observó que para cualquier rango de lcg (30% al 50% de L), una mayor deflección δ reduce considerablemente la función objetivo a_{CG}, llegando a su valor mínimo cuando hay una combinación en el aumento de ambas variables δ y lcg. Y de manera opuesta, el movimiento hacia proa del lcg aumenta la resistencia total. Por tanto, para este estudio es mejor aumentar el ángulo de deflección que mover al extremo el lcg.

4 Diseño Conceptual con Flaps

El diseño conceptual de la lancha de transporte interislas se realizó a partir de los resultados del proceso de optimización. Hay ciertas consideraciones que deben ser tomadas, por ejemplo el área de pasajeros, criterios de estabilidad, etc. Estos parámetros fueron implementados en el programa *"FlapOptResisandSeakee.py"* como parte de las restricciones y que sirven al momento de desarrollar el modelo conceptual. Para comparar y elegir el modelo óptimo se calculó el índice de mareo por movimiento expuesto en la ISO (ISO 2631-1, Reviewed 2021), $\%MSI = \mathrm{km}.\mathrm{a_{CG}}.\sqrt{\mathrm{T}}$ (2.8, usando el valor de la aceleración del centro de gravedad y el tiempo de exposición de viaje entre islas.

5 Comparación y Selección del Modelo Óptimo

Se calculó el %MSI para un tiempo de viaje promedio de 2 horas, y se presentan los resultados en la Tabla 3.2. La tabla muestra que se obtuvo una mejora del

18.5% en el MSI con la instalación del flap con respecto a la lancha original. Inicialmente con el modelo optimizado sin flaps se obtuvo una mejora del 14%, por lo que se concluye que, con la instalación de los flaps se puede obtener más del 18% en la reducción del índice de mareo por movimiento con respecto a una embarcación sin trim tabs. El valor de a_{CG} ha sido escogido tomando en cuenta la resistencia total, de manera que, si la segunda función objetivo no es impedimento, el valor de a_{CG} podría reducirse hasta un 40% de la aceleración de la lancha original, Figura 3.1.

Tabla 3.2 Cálculo del Porcentaje de Índice de Mareo por Movimiento (MSI)

Aceleración RMS [m/s²]		MSDVz [m/s½]	%MSI
Lancha Angy	1.14	96.73	32.24
Modelo sin flaps (Villamarín, 2020)	0.98	83.16	27.72
Modelo optimizado con flaps	0.93	78.91	26.30

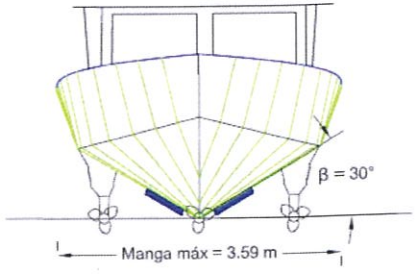
A continuación se presentan las dimensiones principales del nuevo modelo y su comparación con las dimensiones de los diseños anteriores, Tabla 3.3, PLANO 1 Distribución General de Lancha con Flaps.

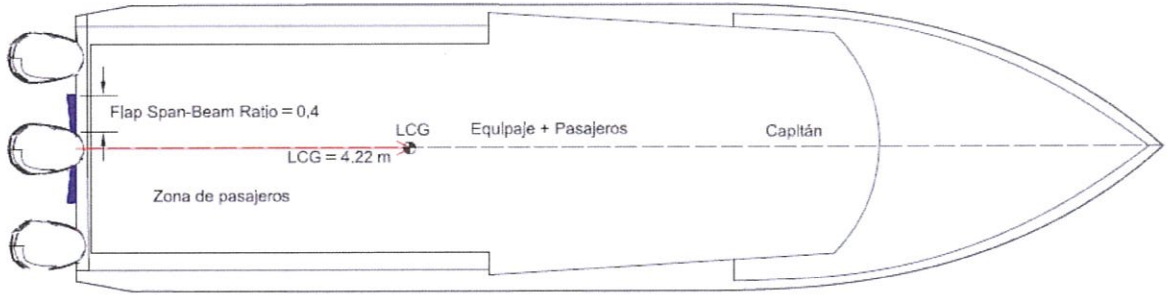
Tabla 3.3. Características de los Modelos de Lanchas Interislas

	Lancha Angy	**Modelo optimizado** (Villamarín, 2020)	**Modelo con Flaps**
Lwl , m	11.01	13.66	13.69
B_{max}, m	3.45	3.44	3.58
D, m	1.51	1.55	1.59
lcg, m	4.80	5.58	4.211
$Desplazamiento$, ton	8.26	8.52	9.01
β, grad	14	22	30
$Cuerda\ del\ flap$, m	-	-	0.21
$Ancho\ del\ flap$, manga	-	-	0.40
$Ángulo\ del\ flap$, grad	-	-	10.57
R_T , N	17952	16205	17944
$RMS\ a_{CG}$, g	1.14	0.98	0.93

El modelo optimizado por el ingeniero (Villamarín, 2020) fue desarrollado mediante el proceso de direcciones factibles utilizando el programa Copes-Conmin (Vanderplaats, 2007), y se requiere numerosas pruebas para evitar caer en los mínimos locales de las funciones objetivo. A través de la optimización tipo genética un modelo con características similares al modelo de (Villamarín, 2020) fue encontrado en una de las 200 generaciones. Lo que significa que, el algoritmo genético de ordenamiento no dominado es más efectivo al proceso que se estaba utilizando debido a la cantidad de generaciones producidas en una sola ejecución del programa.

Distribución General Lancha con Flaps





Especificaciones	
Eslora Total	13.69 [m]
Manga máx	3.59 [m]
Puntal	1,59 [m]
LCG (desde popa)	4.211 [m]
Ángulo de Astilla Muerta	30 [°]
Cuerda del flap	0.21[m]
Ancho del flap	0.4 [manga]
Ángulo del flap	10,57 [°]
Resistencia Total	17944 [N]
acg	0.93 [g]
Pasajeros	24

PLANO 1 Distribución General de Lancha con Flaps

CAPÍTULO 4

4. CONCLUSIONES Y RECOMENDACIONES

El archipiélago de Galápagos es un destino turístico que atrae a muchos visitantes locales e internacionales. El medio de transporte principal interislas son las lanchas por el moderado costo respecto del aéreo. Se ha reportado que estos viajes son muy incómodos por el elevado movimiento de estas embarcaciones planeadoras, llegando a ocasionar mareos e incluso vómitos en los pasajeros. Por tanto, el estudio de una solución que ayude a aminorar el problema de malestar en usuarios es de alta importancia para brindar un mejor servicio. El presente trabajo se realizó como continuidad del proyecto de investigación del ingeniero Villamarín (Villamarín, 2020), en el que se logró una disminución de la aceleración del centro de gravedad mediante un movimiento del centro de gravedad lcg, que se pudo implementar mediante el movimiento de pesos en la lancha.

El presente proyecto analizó la reducción de la misma función objetivo aceleración vertical mediante la colocación de flaps o dispositivos de control en la popa de la embarcación. Se implementó un algoritmo de optimización de tipo genético y múltiple objetivo, junto con restricciones de geometría del casco, francobordo y estabilidad. Finalmente, se realizó una comparación con el diseño desarrollado por (Villamarín, 2020) y la influencia en el malestar de pasajeros utilizando la norma internacional (ISO 2631-1, Reviewed 2021) para obtener un diseño conceptual que sirva en investigaciones futuras y en el diseño preliminar.

4.1 Conclusiones

Mediante el proceso que se ha seguido en el desarrollo de este proyecto finalmente se concluye que:

1. La dinámica de una embarcación planeadora es muy compleja, según lo describen numerosos estudios realizados por (Savitsky & Brown, Procedures For Hydrodynamic Evaluation of Planing Hulls in Smooth and Rough Water,

1976) entre algunos otros; allí se concluye que se puede conseguir una mejor respuesta del planeador mediante la instalación de los flaps. Para determinar las dimensiones del dispositivo se deben considerar la resistencia al avance y el desempeño en el mar. Un proceso que permita implementar dos funciones objetivo al mismo tiempo es conocido como optimización multiobjetivo. Algunos procesos son más efectivos que otros cuando se tienen que manejar un alto número de variables y restricciones. El algoritmo NSGA-II produce un conjunto de soluciones en un espacio de diseño llamado frente de Pareto, los resultados muestran que la aceleración del centro de gravedad a_{CG} podría reducirse hasta un 40% de la lancha original. Tomando en consideración ambas funciones objetivo, se podría conseguir hasta un 19% de mejora en la aceleración y 5% en la resistencia al avance.

2. Mediante la aplicación del programa implementado en este trabajo se obtuvo que las combinaciones de los parámetros principales lcg, β y δ que reducen la función objetivo aceleración principalmente se dan cuando $\delta > 10°$, $\beta = 30°$ y una ubicación del lcg > 35% L. Mientras que para las variables geométricas se observa que se produce una mayor reducción de la aceleración vertical del CG cuando la relación L/B aumenta, observación consistente con lo reportado por Savitsky (Savitsky & Brown, Procedures For Hydrodynamic Evaluation of Planing Hulls in Smooth and Rough Water, 1976) en el capítulo 8 de su reporte. Los flaps cumplen un rol importante en la optimización, encontrándose que con un mayor ángulo δ y una mayor cuerda L_F se logra reducir la función objetivo a_{CG}. Cuando el span es mayor al 50% de la manga, se requiere un menor ángulo δ y la reducción en a_{CG} será mayor en comparación a un flap con span restringido.

3. En un proceso de optimización multiobjetivo especificamente hablando del algoritmo NSGA-II el frente de Pareto muestra que es imposible mejorar un objetivo sin eventualmente empeorar otro. En la elección del nuevo diseño conceptual se trató de no incrementar la resistencia al avance, de manera que,

el armador puede reutilizar los motores en este nuevo diseño. De acuerdo con la norma internacional (ISO 2631-1, Reviewed 2021) con la instalación de los flaps se puede reducir más del 17% el índice de mareo por movimiento con respecto al modelo optimizado sin flaps con el que se obtuvo una mejora del 14%. Este nuevo modelo sitúa al lcg al 30% de la eslora, lo que significa que el armador puede mantener la distribución general de pesos de la lancha original.

4.2 Recomendaciones

1. Las formulaciones implementadas por (Savitsky & Brown, Procedures For Hydrodynamic Evaluation of Planing Hulls in Smooth and Rough Water, 1976) en el cálculo del comportamiento en olas son de tipo empírico. Esto influye en el cálculo de ciertas variables como por ejemplo el ángulo de astilla muerta. En el proceso de optimización esta variable busca siempre los límites extremos, superior o inferior, lo que hace complicado explorar un nuevo conjunto de variables entre estos límites. Situación que no sucede cuando no se considera el comportamiento en olas, proceso en el que si existe variación de esta variable de diseño. Por tanto, una revisión de la formulación de Savitsky o la implementación de un nuevo método para el cálculo del comportamiento en olas debe realizarse a manera de comparar los resultados que se obtienen actualmente.

2. El paquete de programación OpenPlaning sigue en desarrollo por su autor, los cambios que se realizaron para su uso en este proyecto pueden ser implementados en la actual y futura versión.

BIBLIOGRAFÍA

Blank, J., & Deb, K. (2020). Multi-Objective Optimization in Python. *IEEE Access*, 89497-89509.

Brown, P. W. (1971). *An Experimental And Theoretical Study Of Planing Surfaces With Trim Flaps.* Stevens Institute of Technology , Davidson Laboratory , Hoboken, New Jersey.

Castro Feliciano, E. L. (Octubre 2021). *OpenPlaning: Open-Source Framework for the Hydrodynamic Design of Planing Hulls.* Providence, RI: SNAME FAST 21.

Castro-Feliciano, E. (2016). Co-Design of Planing Craft and Active Control Systems. *(Doctoral Thesis).* University of Michigan.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (April 2002). *A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II.* Transactions on Evolutionary Computation, Vol 6, No. 2.

Faltinsen, O. M. (2005). *Hydrodynamics of High-Speed Marine Vehicles.* Cambridge, New York, USA: Cambridge University Press 2005.

Fridsma, G. (1971). *A Systematic Study of the Rough-Water Performance of Planing Boats (Irregular Waves- Part 2).* Naval Ship Research and Development Center, Davidson Laboratory. Stevens Institute of Technology.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley.

ISO 2631-1. (Reviewed 2021). Mechanical Vibration and Shock - Evaluation of Human Exposure to Whole Body Vibration. En T. i. Standardization, *International Standard* (pág. 33).

Knight, J. T. (2014). Multi-Objective Particle Swarm Optimization of a Planing Craft under Uncertainty. *Journal of Ship Production and Design*, 8.

Kochenderfer, M. J., & Wheeler, T. A. (2019). *Algorithms For Optimization.* Massachusetts: The MIT Press.

Lewandowski, E. M. (2004). *The Dynamics of Marine Craft.* WSPC.

Mendoza, I., & Vásquez, E. (2020). *Pruebas de Mar de Lanchas Interislas de Galápagos.* Guayaquil.

Ministerio del Ambiente y Agua. (2020). *Parque Nacional Galapagos Ecuador.* Obtenido de http://www.galapagos.gob.ec/wp-content/uploads/2021/02/Informe_anual_visitantes_2020_V_final_DEAPs.pdf

Mohamad Ayob, A. F. (2011). *Development of an optimization framework for the design of high speed planing craft.* Australia.

Mohamad Ayob, A. F. (2012). The Design Of High Speed Planing Craft Using An Optimization Framework. *Proceedings of the ASME 2012 International Mechanical Engineering Congress & Exposition*, 11.

Sakaki, A., & Ghassemi, H. (2019). Evaluation of the Hydrodynamic Performance of Planing Boat with Trim Tab and Interceptor and Its Optimization Using Genetic Algorithm. *Journal of Marine Science and Application*, 1-11.

Savitsky , D. (1964). Hydrodynamic Design of Planing Hulls. En The Society of Naval Architects and Marine Enginee (Ed.), *Marine Technology and SNAME News, 1-04*, págs. 71-95.

Savitsky , D. (February de 1985). Chapter IV: Planing Craft. (A. S. Engineers, Ed.) *Naval Engineers Journal, 97*(2), 113 - 135.

Savitsky , D., & Brown, P. W. (1976). Procedures For Hydrodynamic Evaluation of Planing Hulls in Smooth and Rough Water. *Marine Technology, 13*, 1-20.

Savitsky , D., & Koelbel , J. G. (1993). Seakeeping of Hard Chine Hulls. *Technical and Research Panel SC-1 Society of Naval Architects and Marine Engineers .*

Savitsky , D., DeLorme, M. F., & Datla, R. (January 2007). *Inclusion of Whisper Spray in Drag in Preformance Prediction Method for High-Speed Planing Hulls.* Marine Technology.

Vanderplaats, G. N. (2007). *Multidiscipline Design Optimization.* VR&D.

Villamarín, E. G. (2020). Propuesta de Mejora del Comportamiento Dinámico de Lanchas Interislas de Galapagos Orientado al Confort de Pasajeros. *(Tesis de grado).* Escuela Superior Politécnica del Litoral, Guayaquil.

APÉNDICES

APÉNDICE A

Resultados de las pruebas para diferentes Span-Beam Ratio

Span-Beam Ratio = 0.25

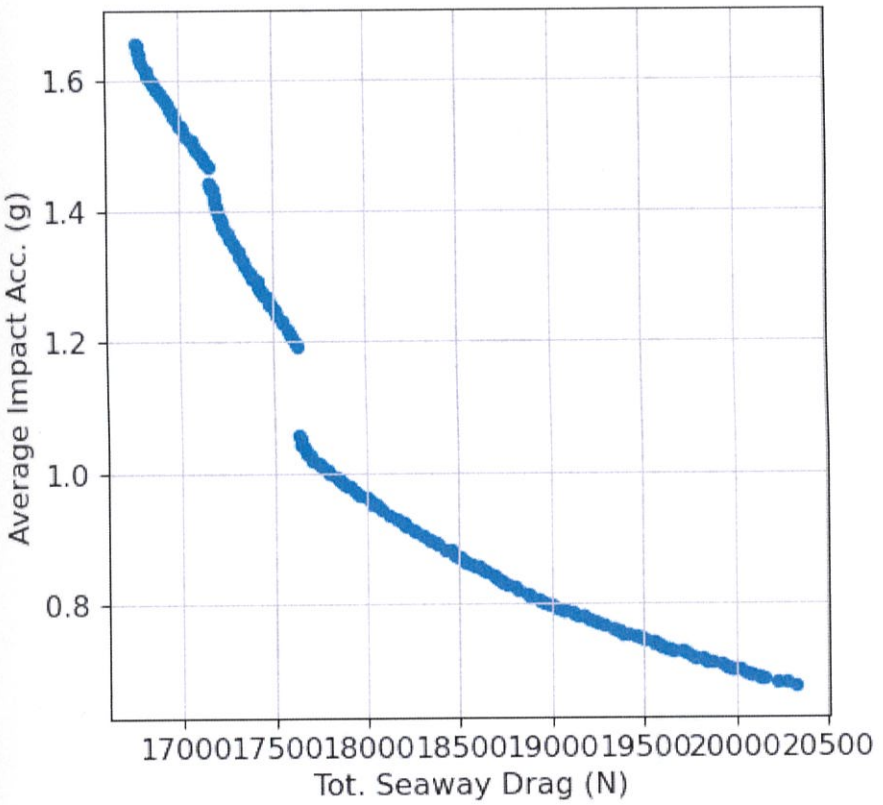


Figura A. 1 Pareto Front spam-beam ratio 0.25

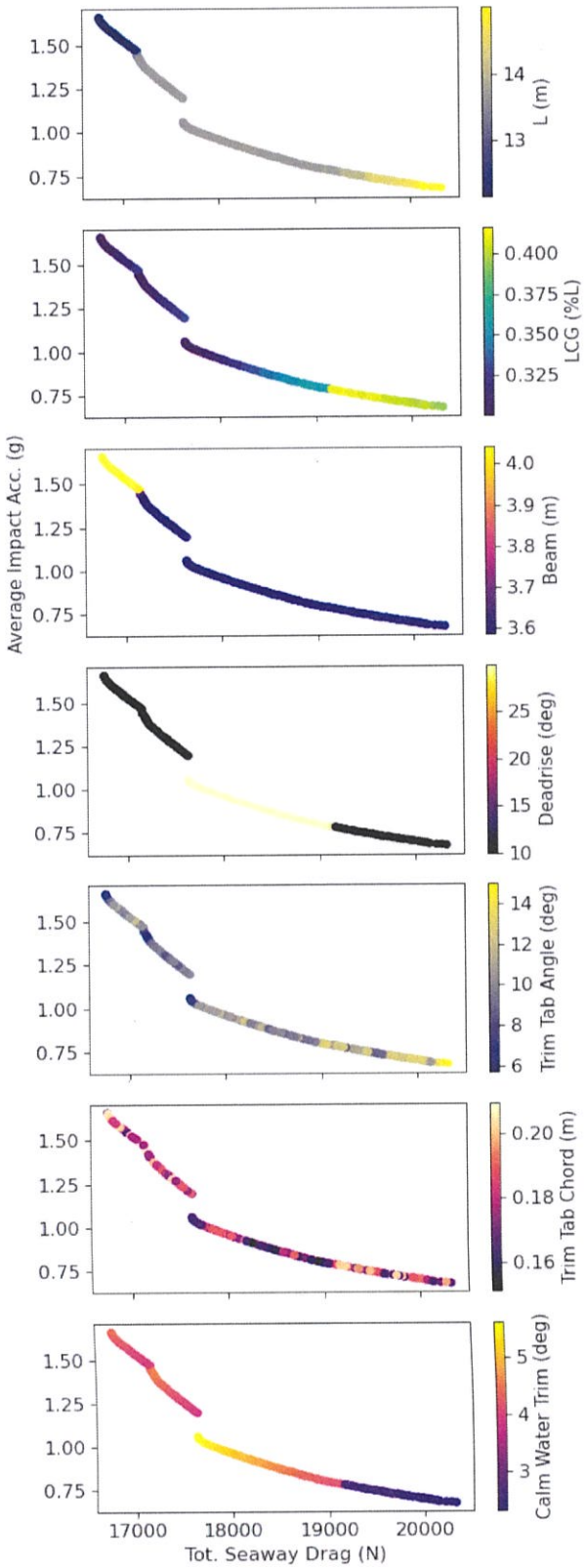


Figura A. 2 Pareto Front de las variables de diseño spam- beam ratio 0.25

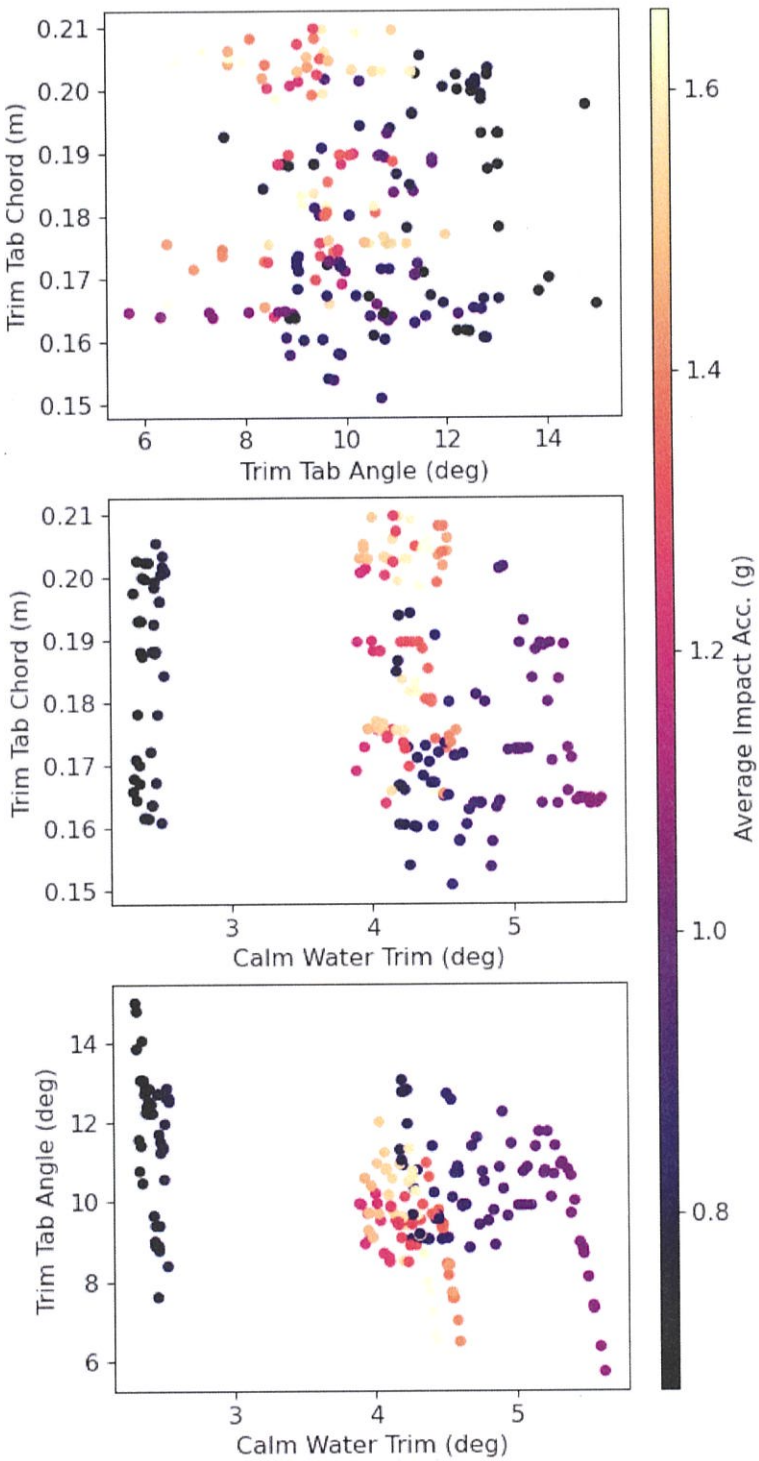


Figura A. 3 **Variables Estimadas del Pareto Front Función Objetivo a_{CG} Spam- beam ratio 0.25**

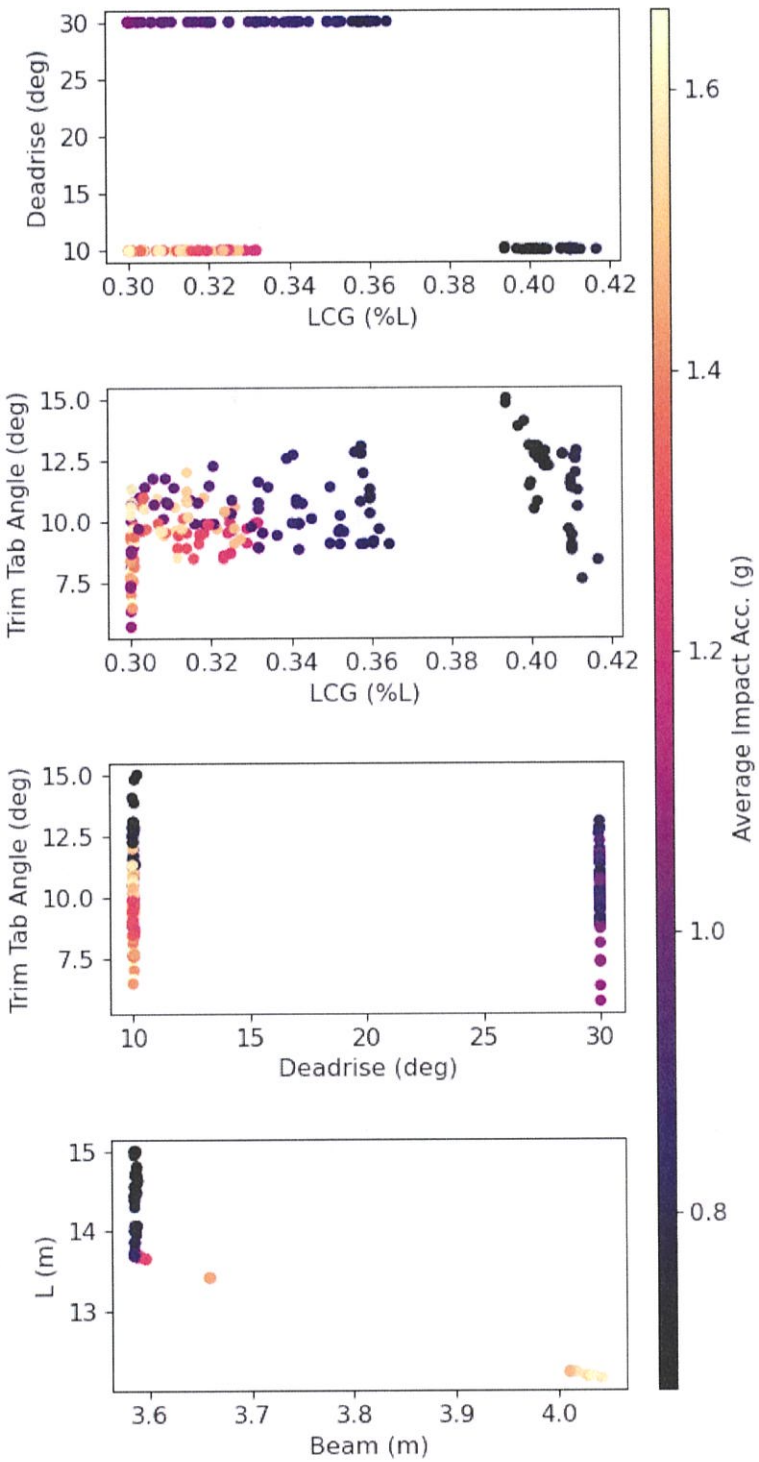


Figura A. 4 Variables Estimadas del Pareto Front Función Objetivo a_{CG} Spam- beam ratio 0.25

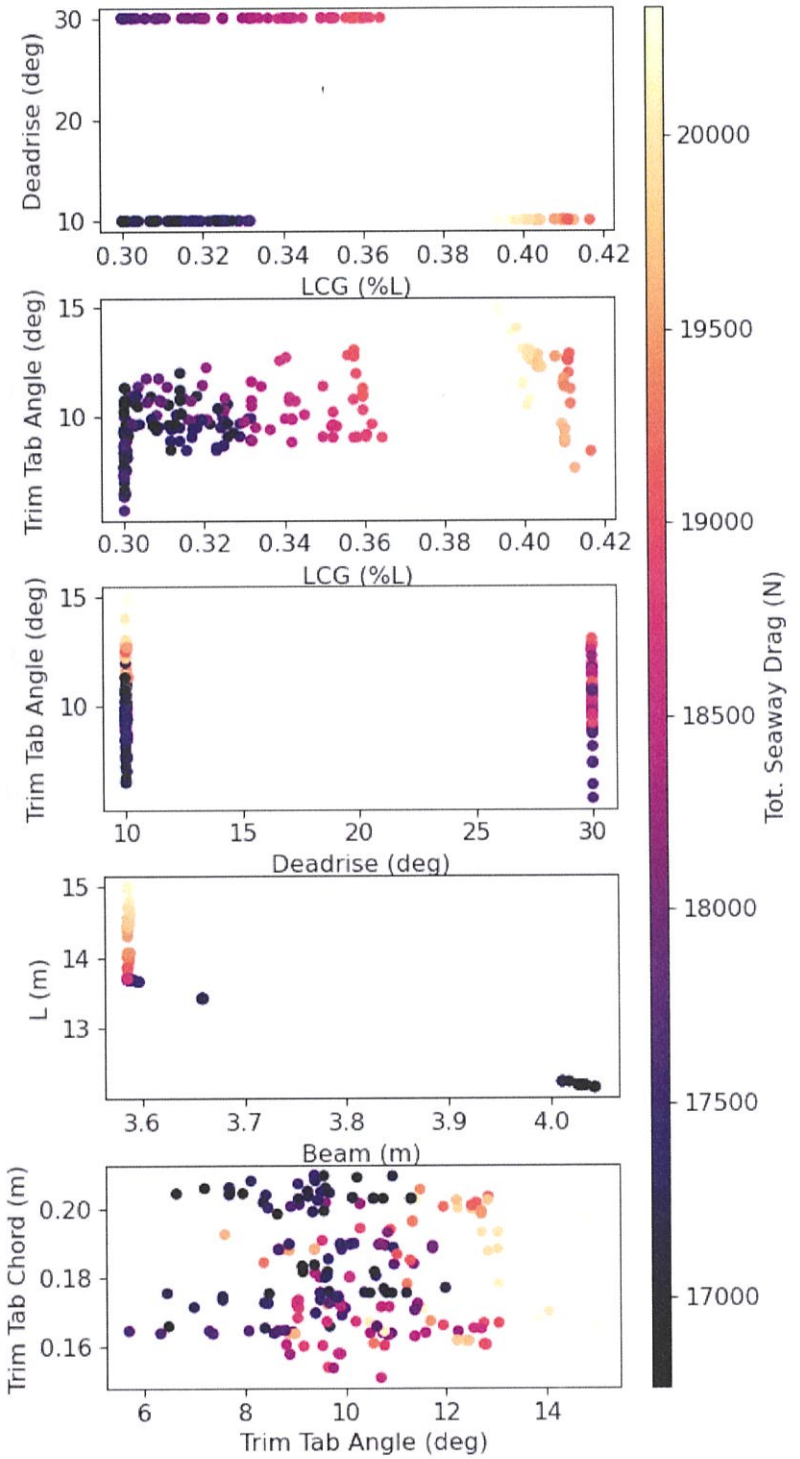


Figura A. 5 Variables Estimadas del Pareto Front Función Objetivo R_T Spam- beam ratio 0.25

Span- Beam Ratio = 0.35

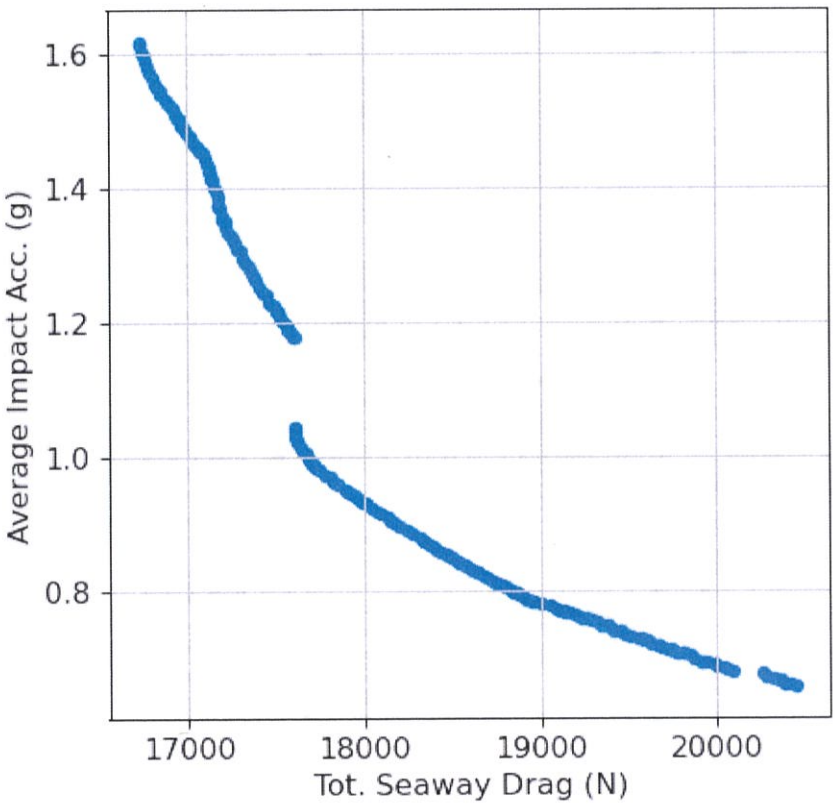


Figura A. 6 Pareto Front spam-beam ratio 0.35

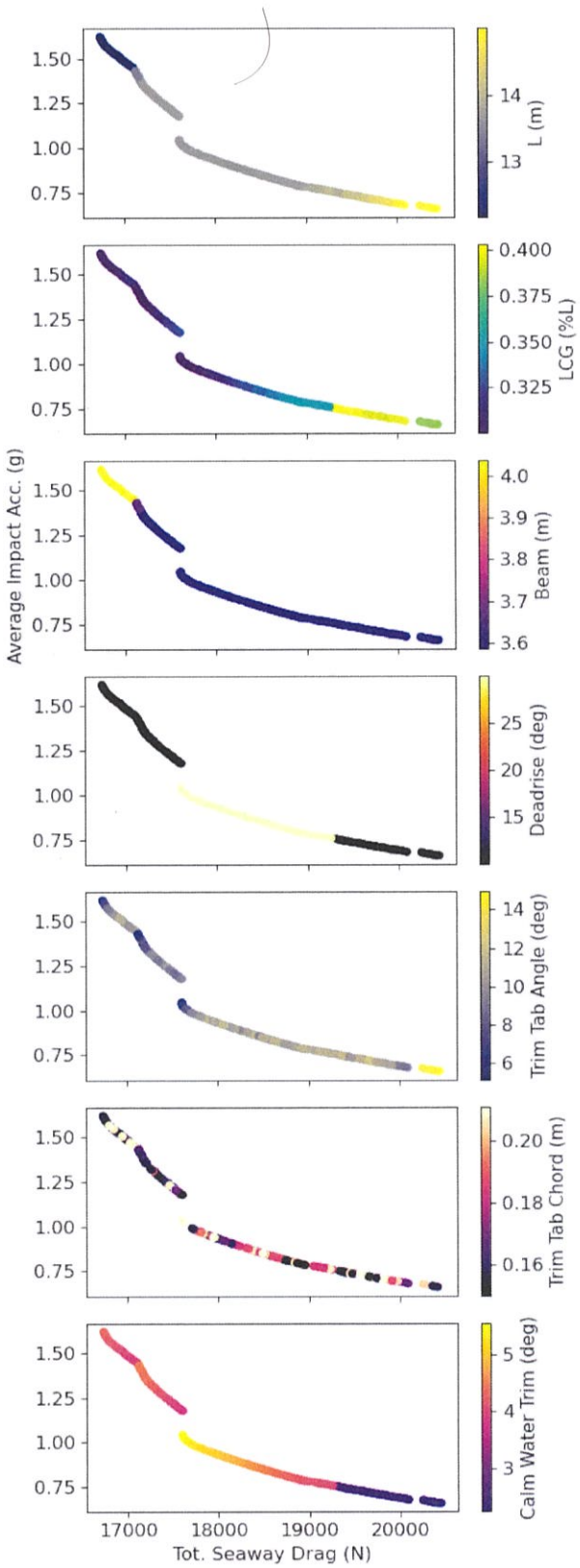


Figura A. 7 Pareto Front de las variables de diseño spam- beam ratio 0.35

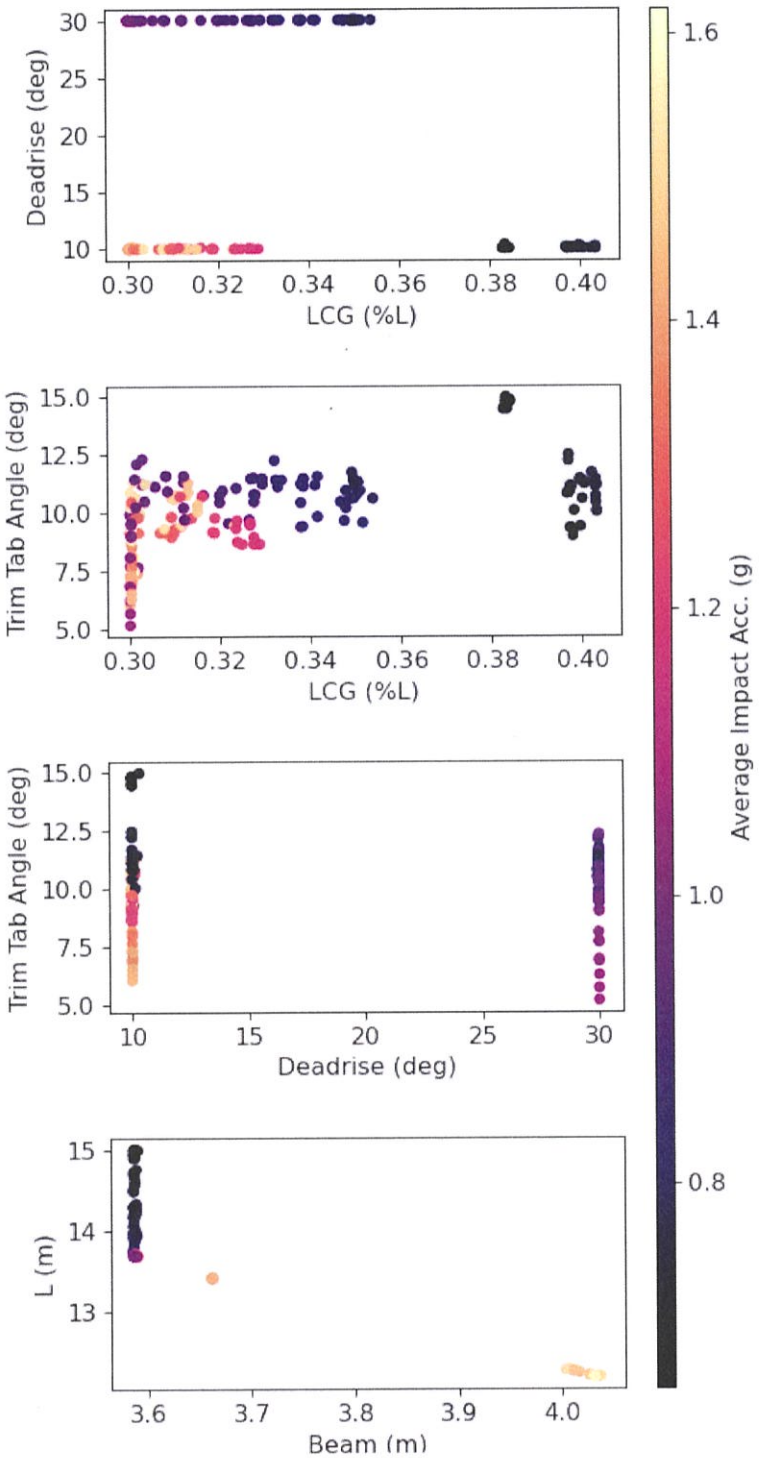


Figura A. 8 Variables Estimadas del Pareto Front Función Objetivo aCG Spam- beam ratio 0.35

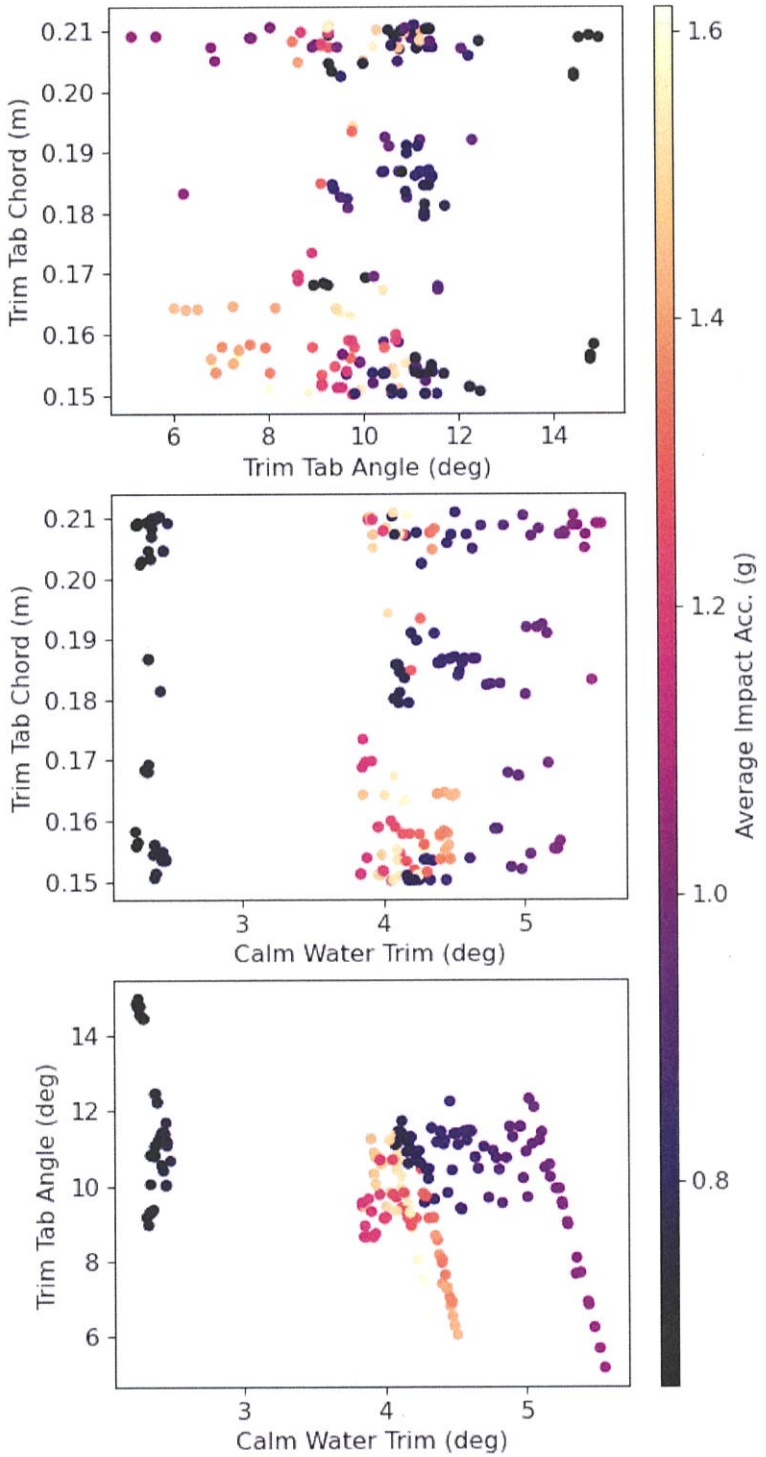


Figura A. 9 Variables Estimadas del Pareto Front Función Objetivo aCG Spam- beam ratio 0.35

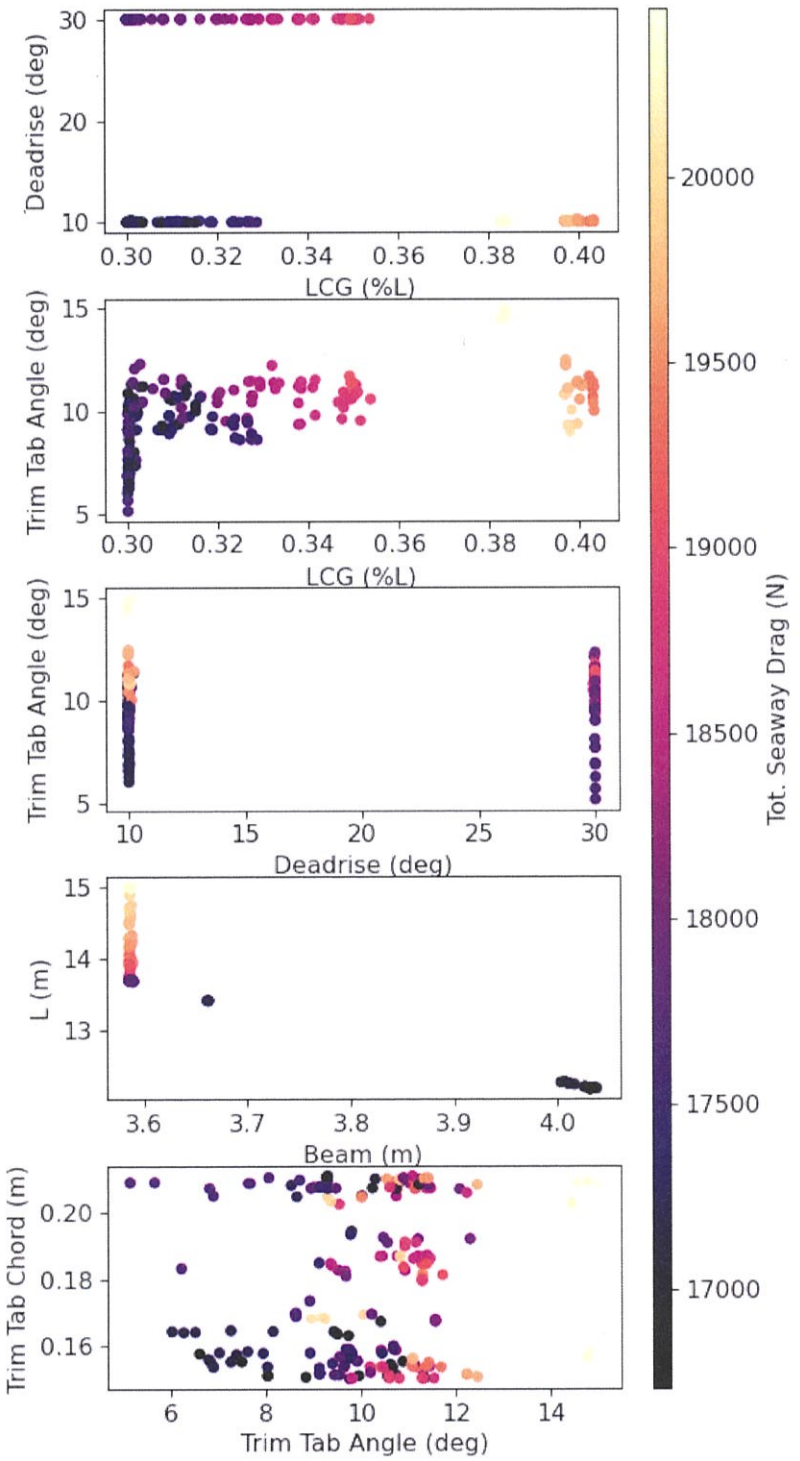


Figura A. 10 Variables Estimadas del Pareto Front Función Objetivo RT Spam- beam ratio 0.35

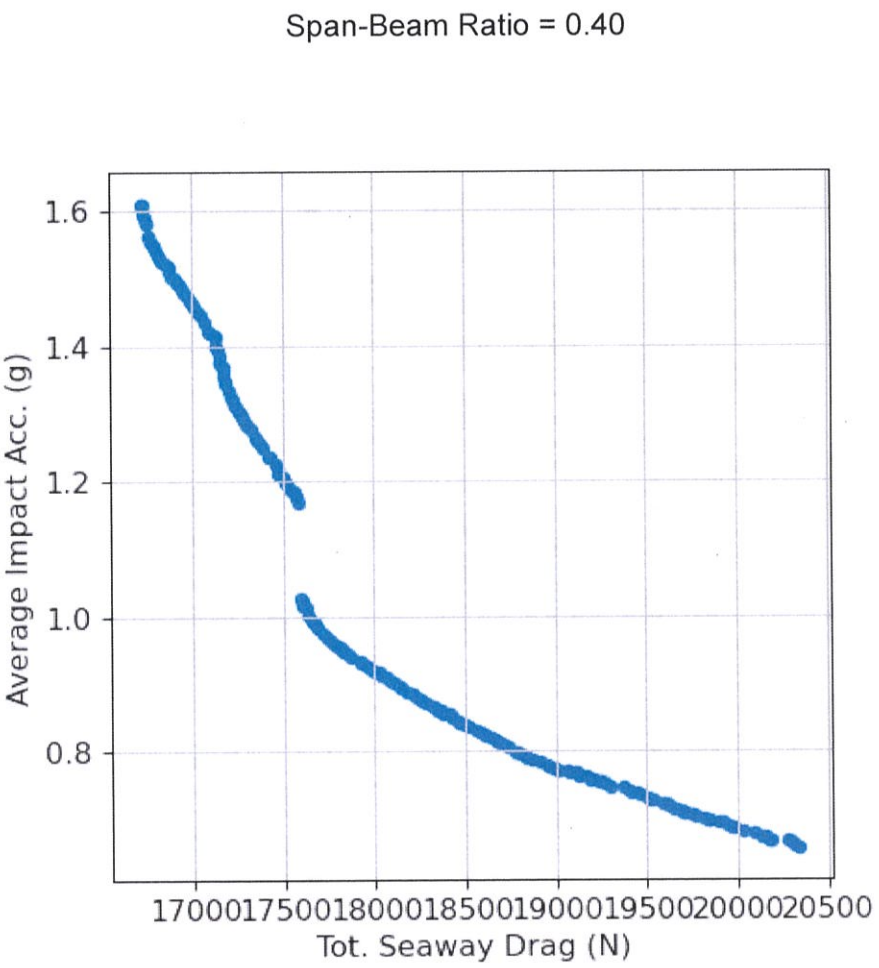


Figura A. 11 Pareto Front spam-beam ratio 0.40

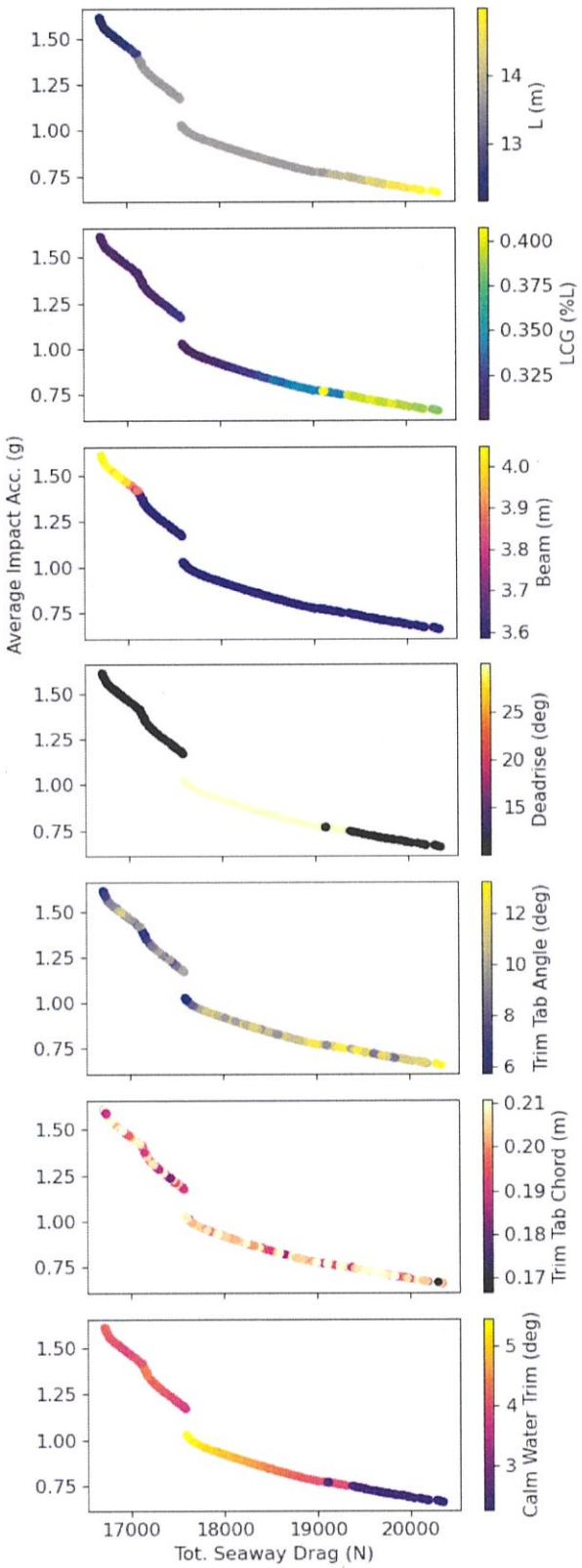


Figura A. 12 Pareto Front de las variables de diseño spam- beam ratio 0.40

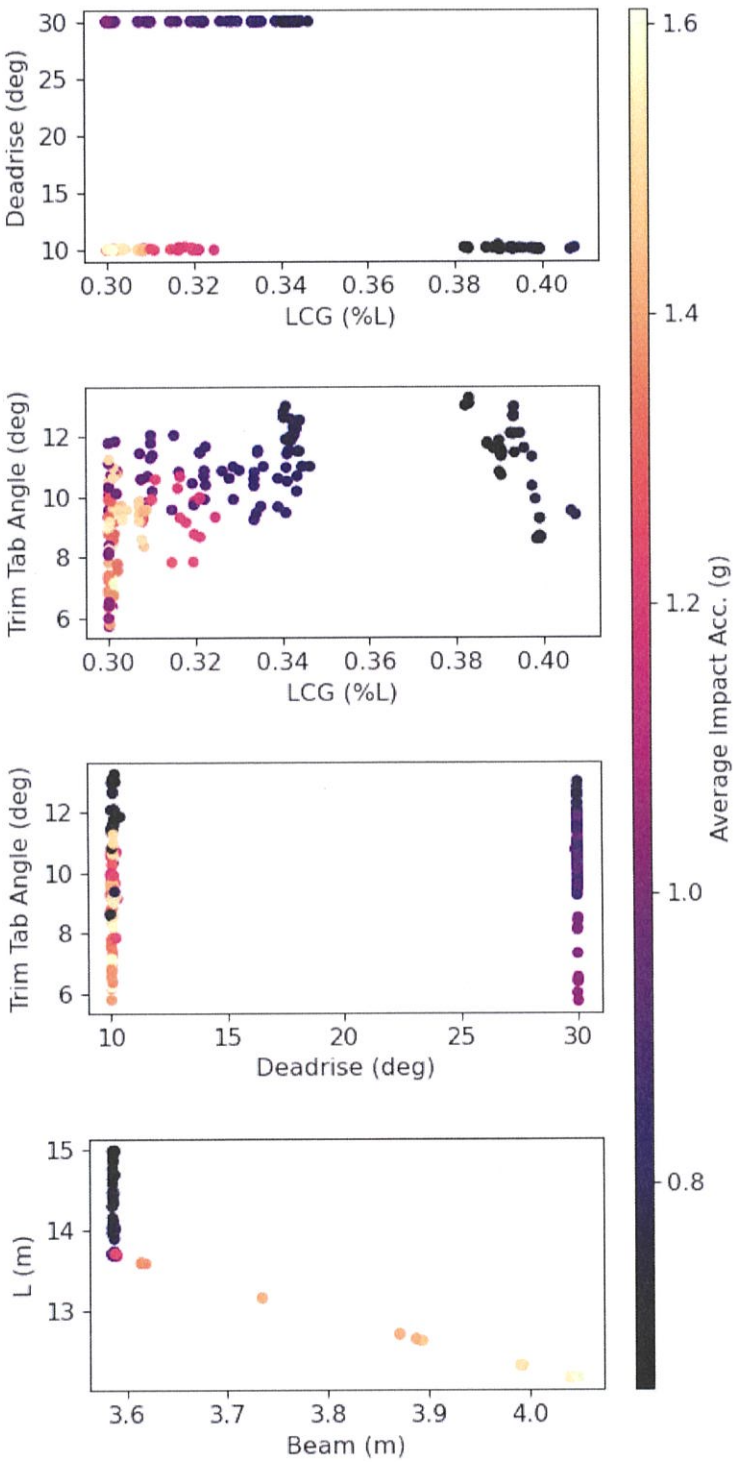


Figura A. 13 Variables Estimadas del Pareto Front Función Objetivo aCG Spam- beam ratio 0.40

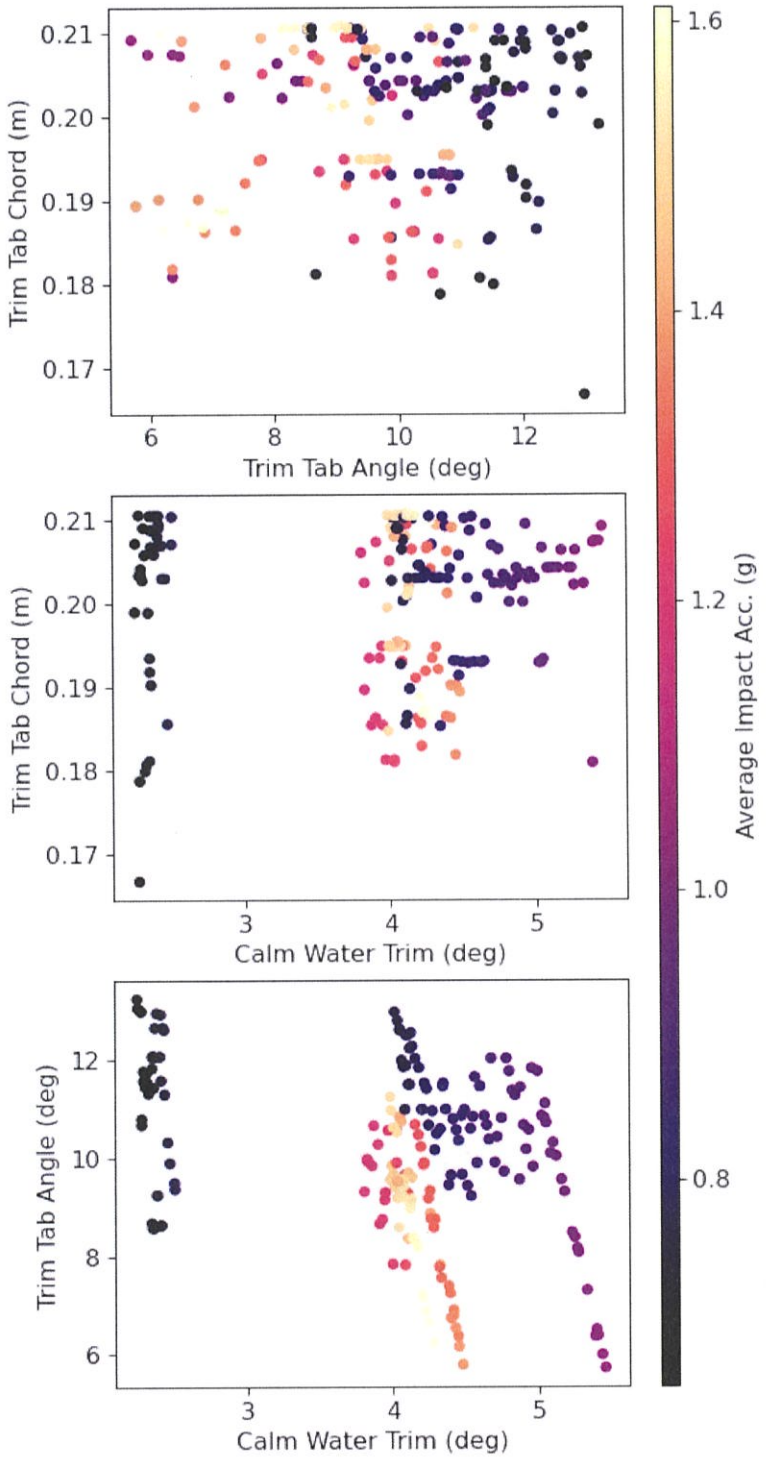


Figura A. 14 Variables Estimadas del Pareto Front Función Objetivo aCG Spam- beam ratio 0.40

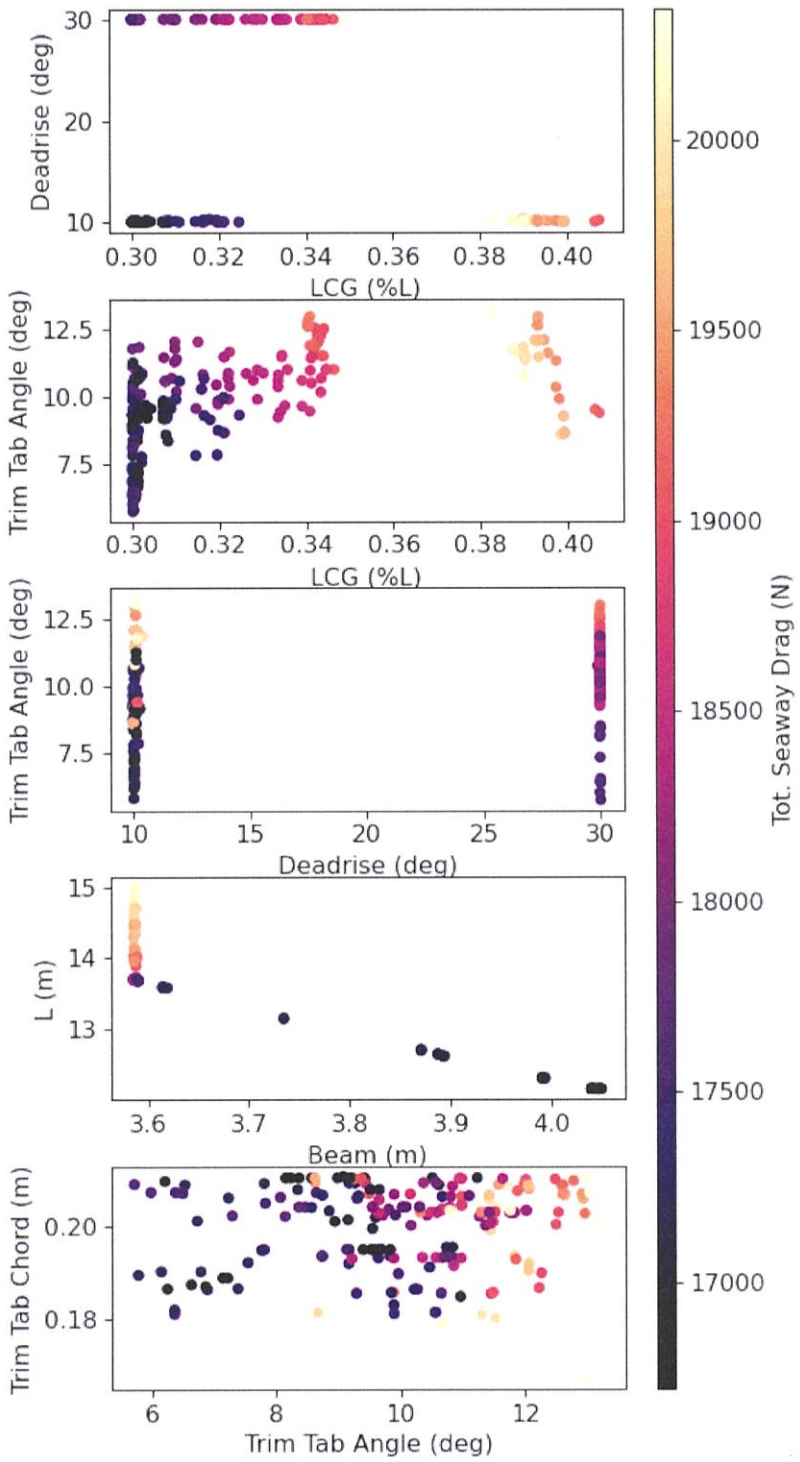


Figura A. 15 Variables Estimadas del Pareto Front Función Objetivo RT Spam- beam ratio 0.40

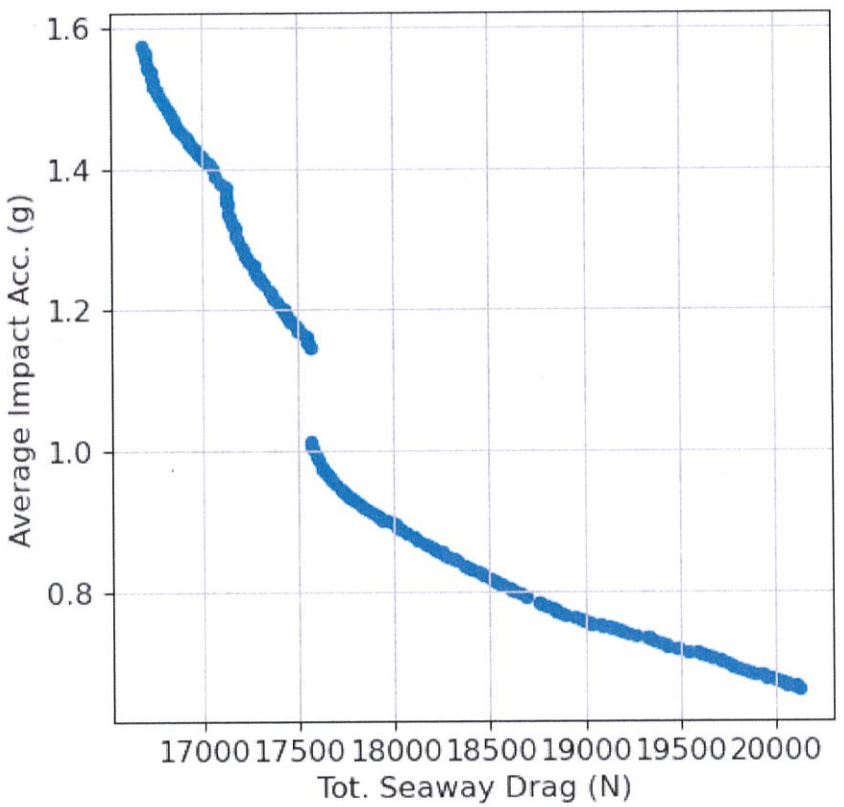


Figura A. 16 Pareto Front spam-beam ratio 0.50

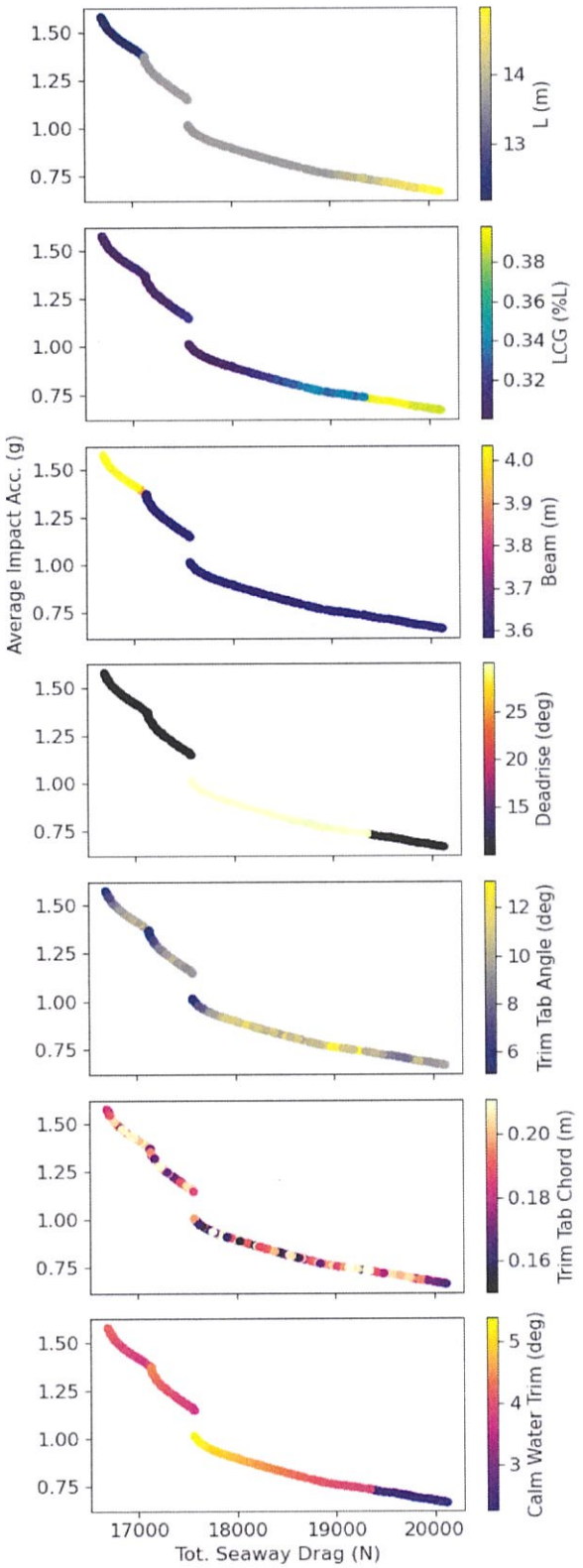


Figura A. 17 Pareto Front de las variables de diseño spam- beam ratio 0.50

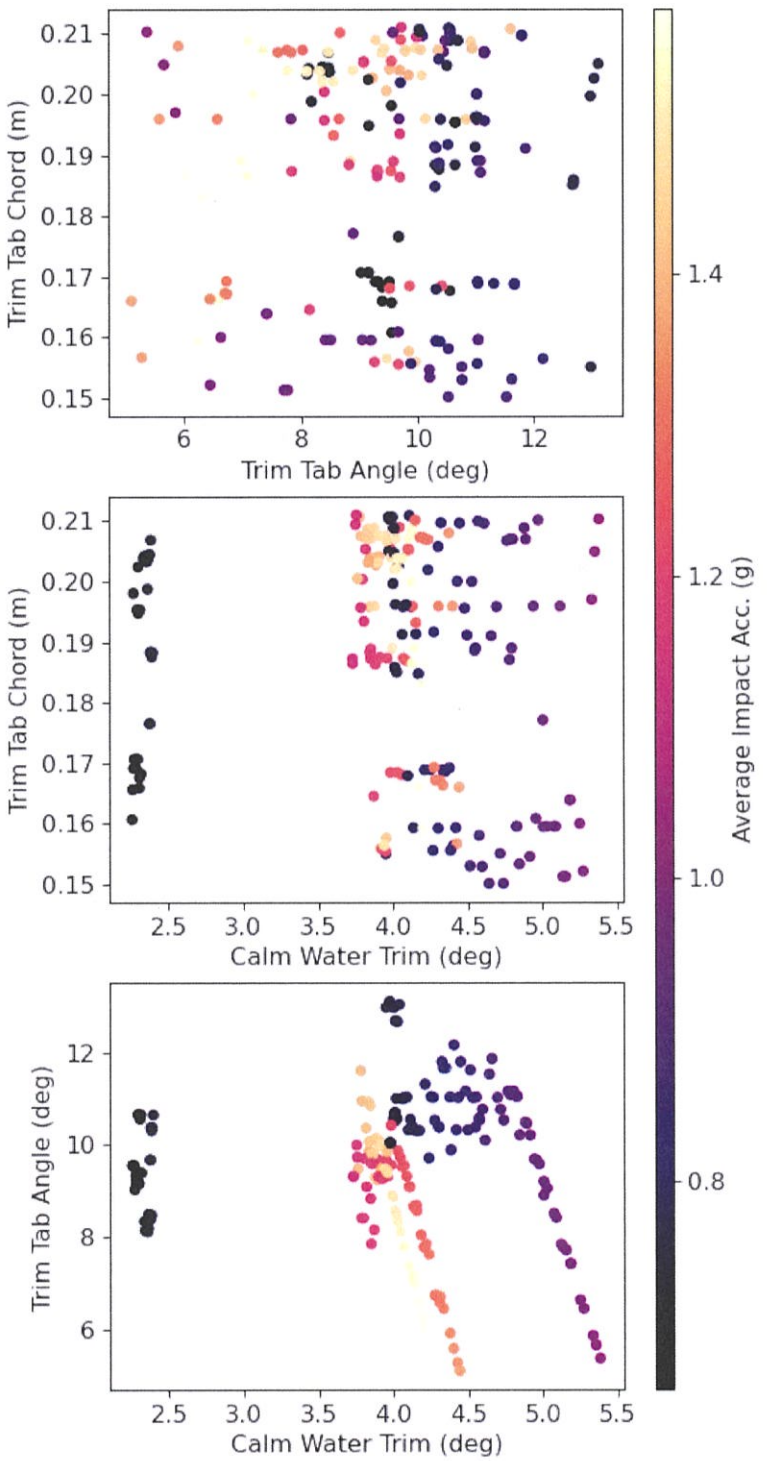


Figura A. 18 Variables Estimadas del Pareto Front Función Objetivo aCG Spam- beam ratio 0.50

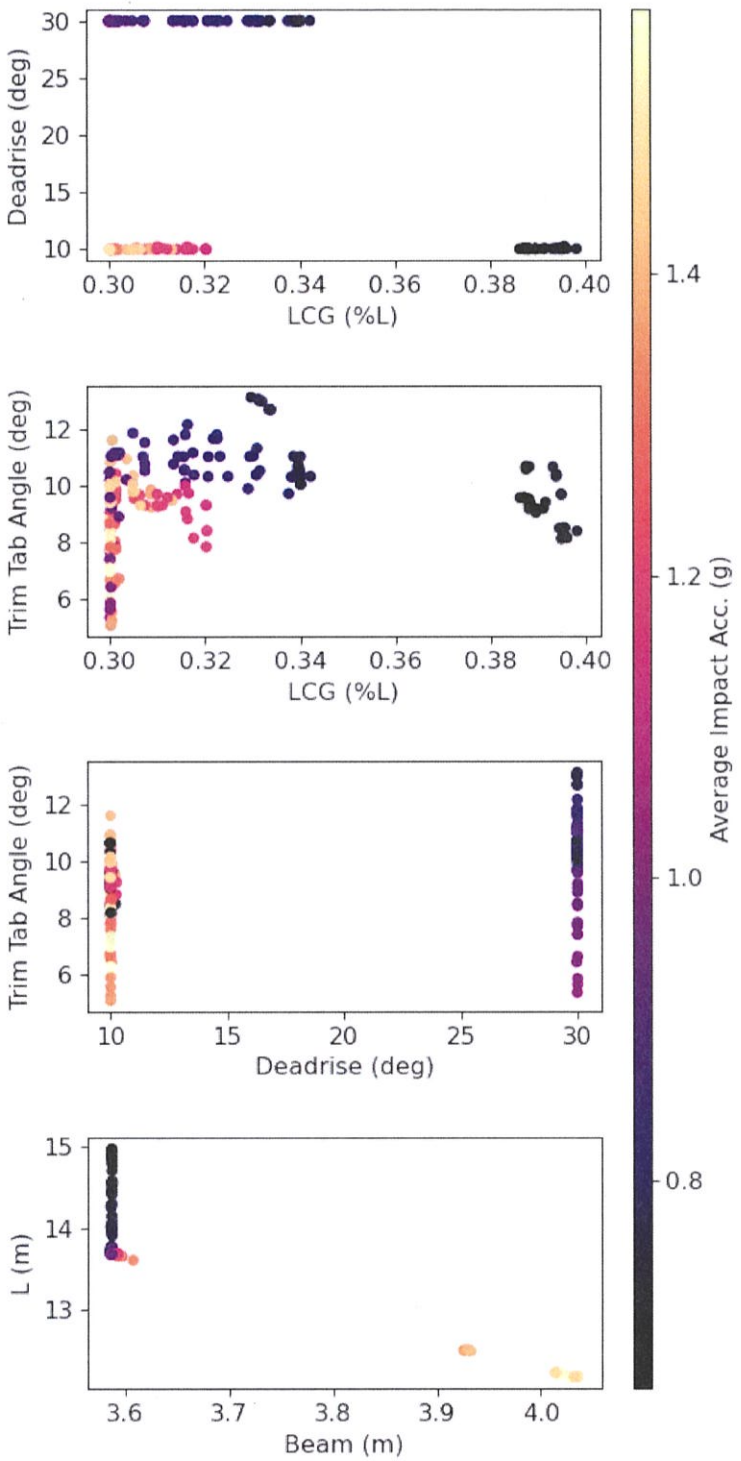


Figura A. 19 Variables Estimadas del Pareto Front Función Objetivo aCG Spam- beam ratio 0.50

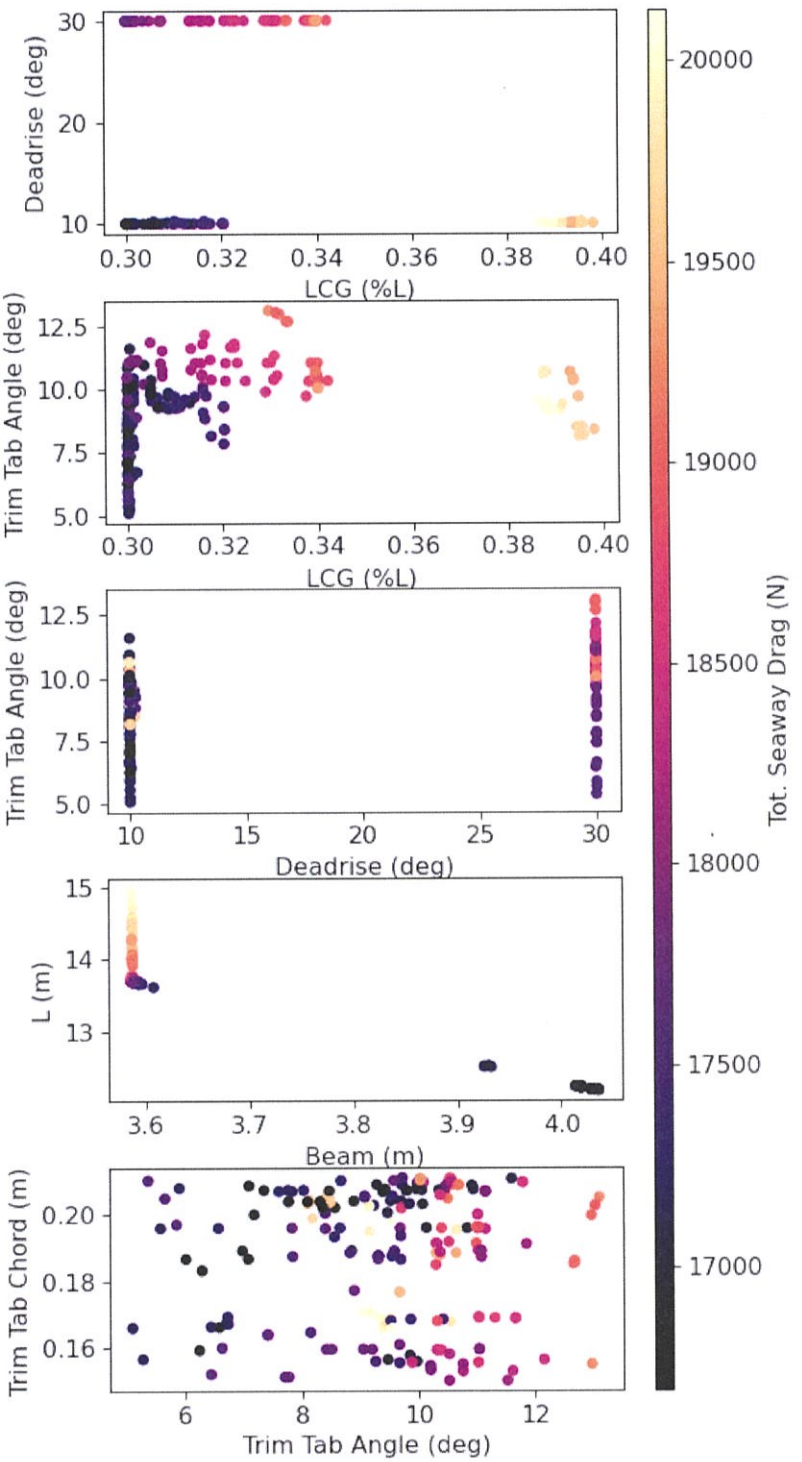


Figura A. 20 Variables Estimadas del Pareto Front Función Objetivo RT Spam- beam ratio 0.50

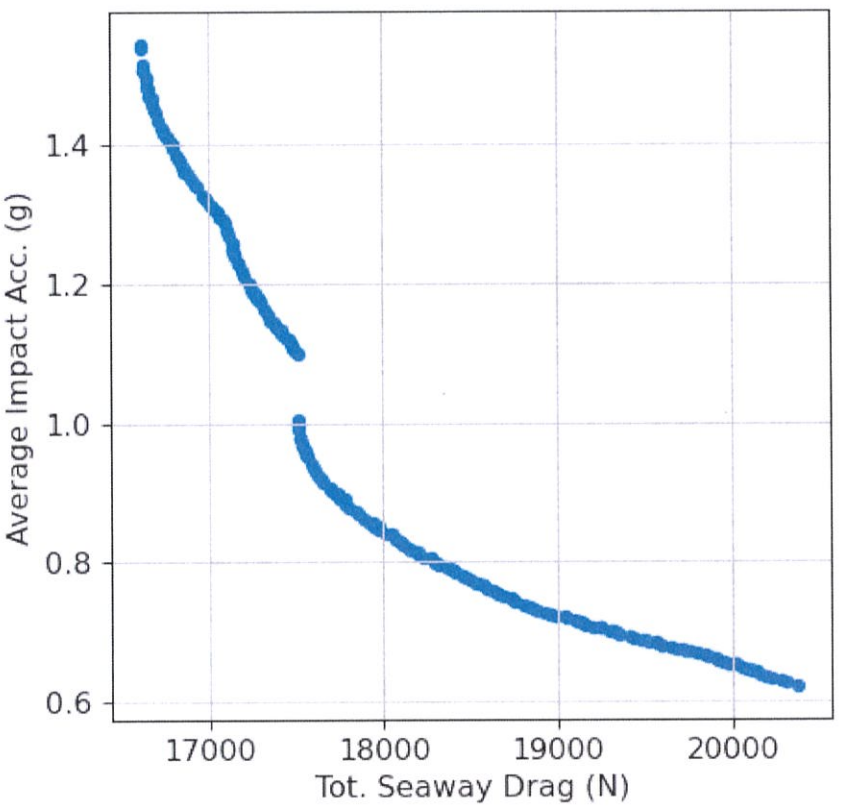


Figura A. 21 Pareto Front spam-beam ratio 0.75

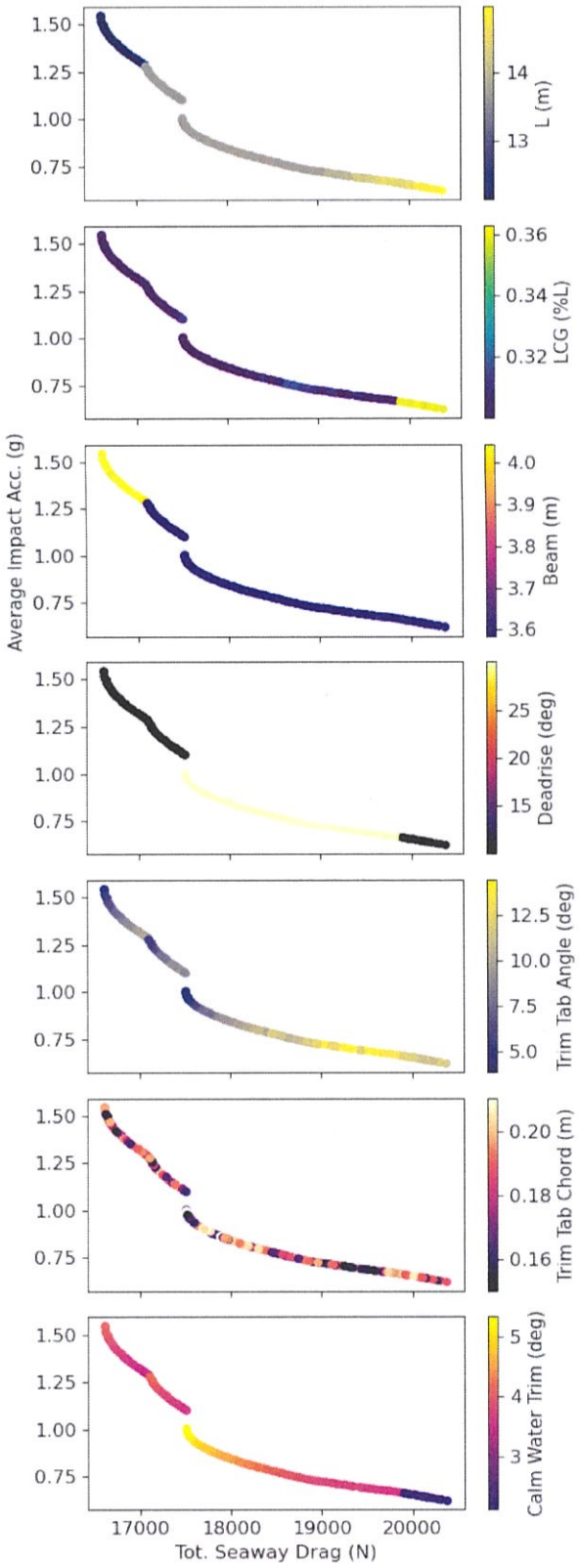


Figura A. 22 Pareto Front de las variables de diseño spam- beam ratio 0.75

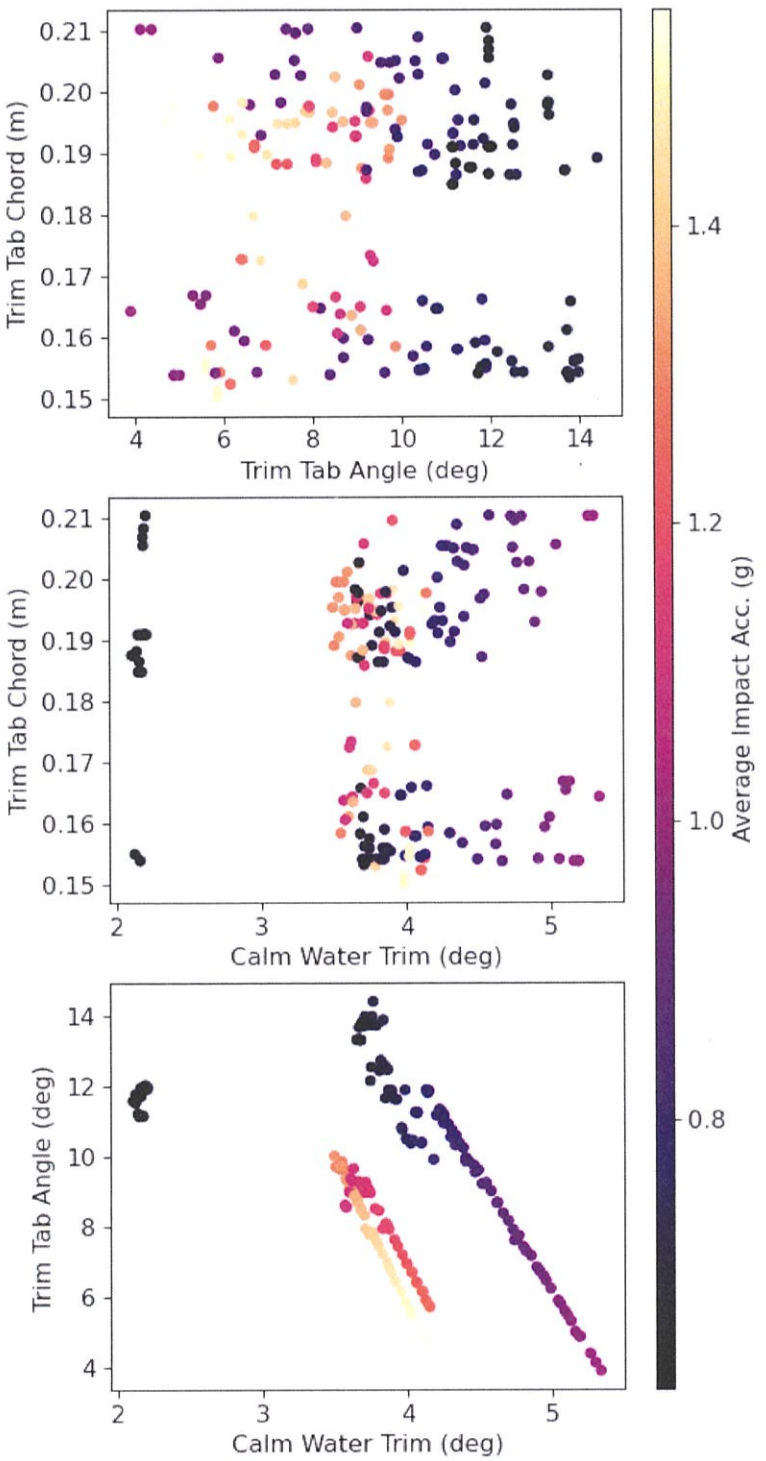


Figura A. 23 Variables Estimadas del Pareto Front Función Objetivo aCG Spam- beam ratio 0.75

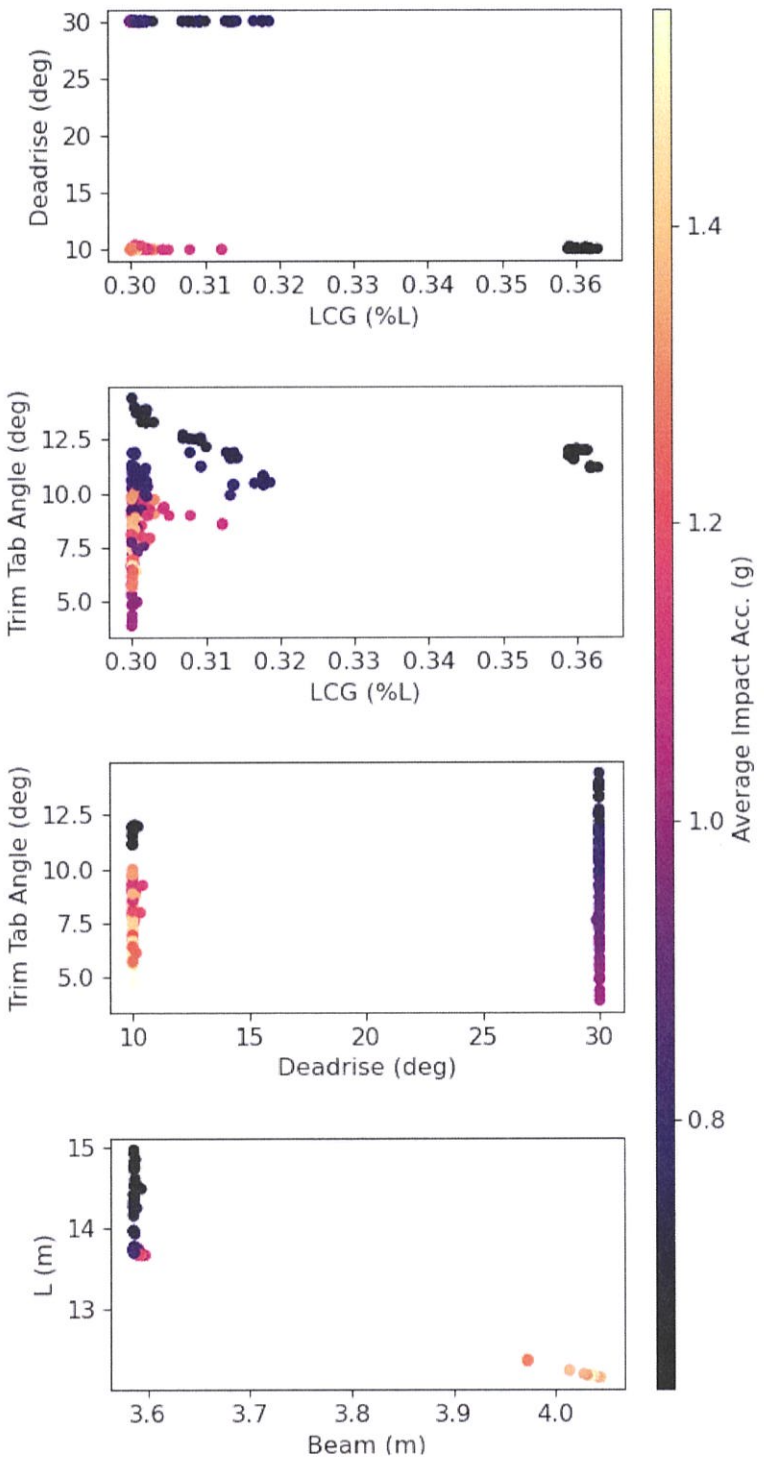


Figura A. 24 Variables Estimadas del Pareto Front Función Objetivo aCG Spam- beam ratio 0.75

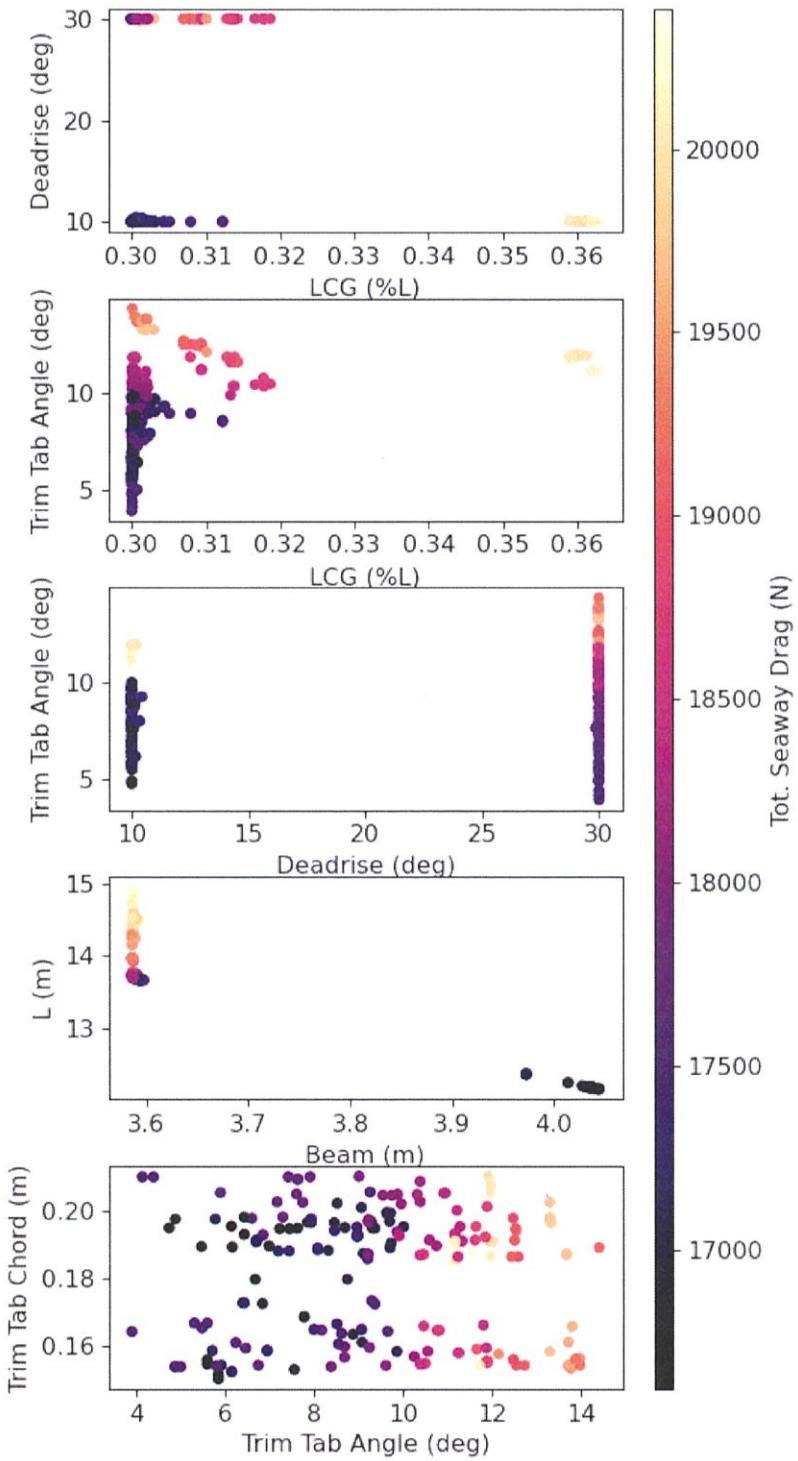


Figura A. 25 Variables Estimadas del Pareto Front Función Objetivo RT Spam- beam ratio 0.75

Span-Beam Ratio = 1.0

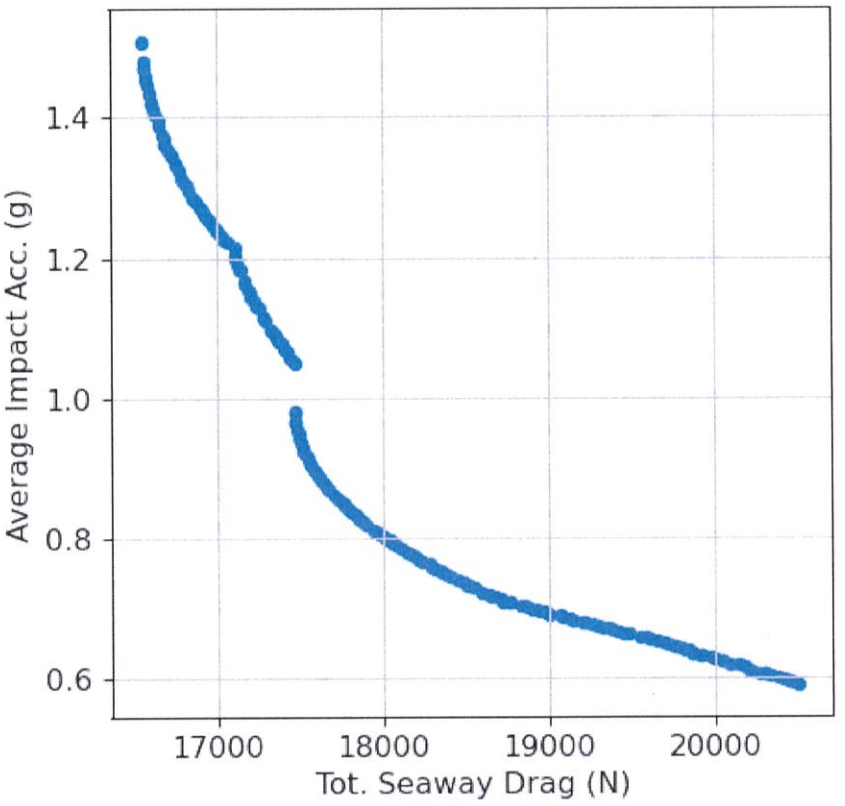


Figura A. 26 Pareto Front spam-beam ratio 1.0

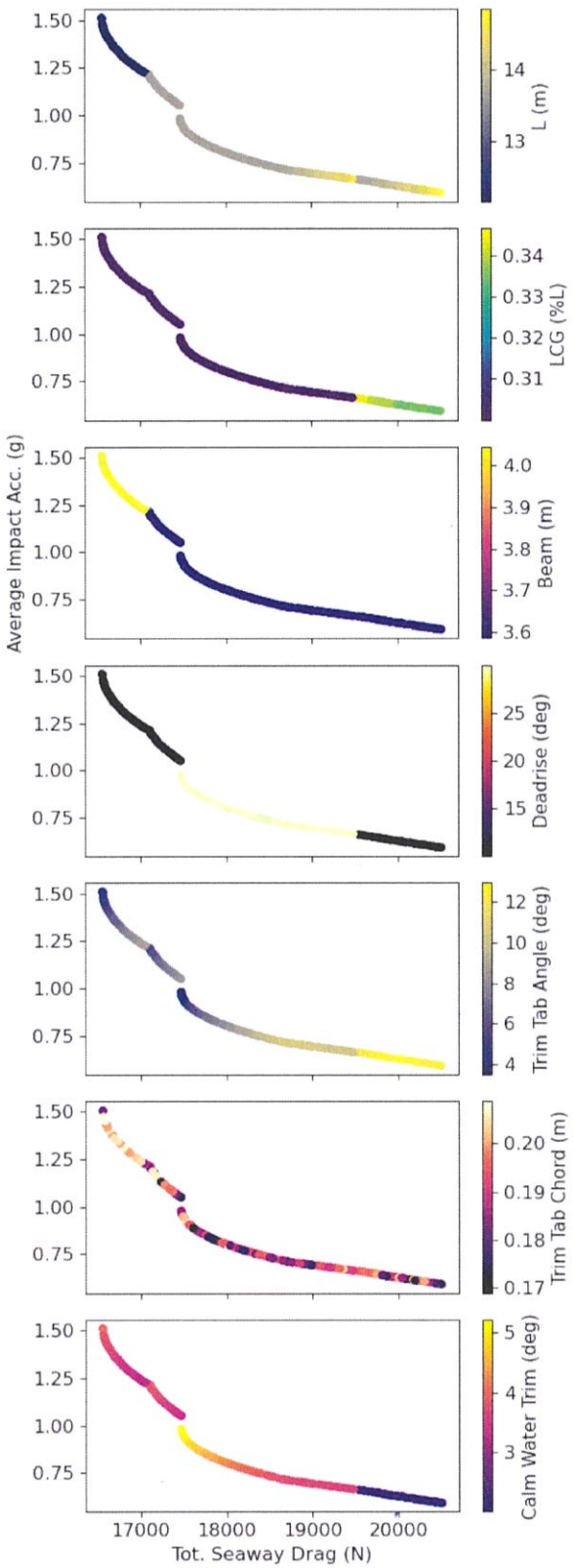


Figura A. 27 Pareto Front de las variables de diseño spam- beam ratio 1.0

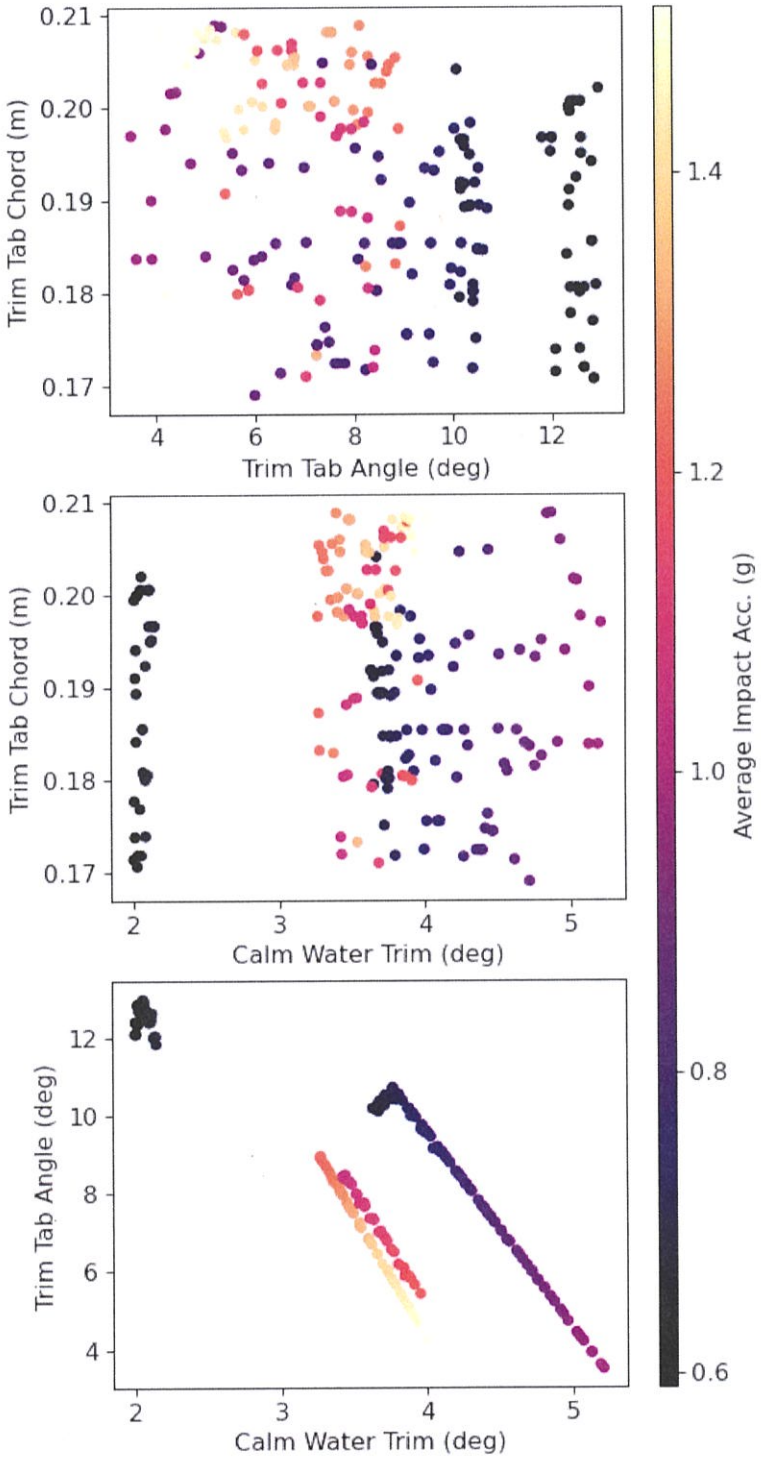


Figura A. 28 Variables Estimadas del Pareto Front Función Objetivo aCG Spam- beam ratio 1.0

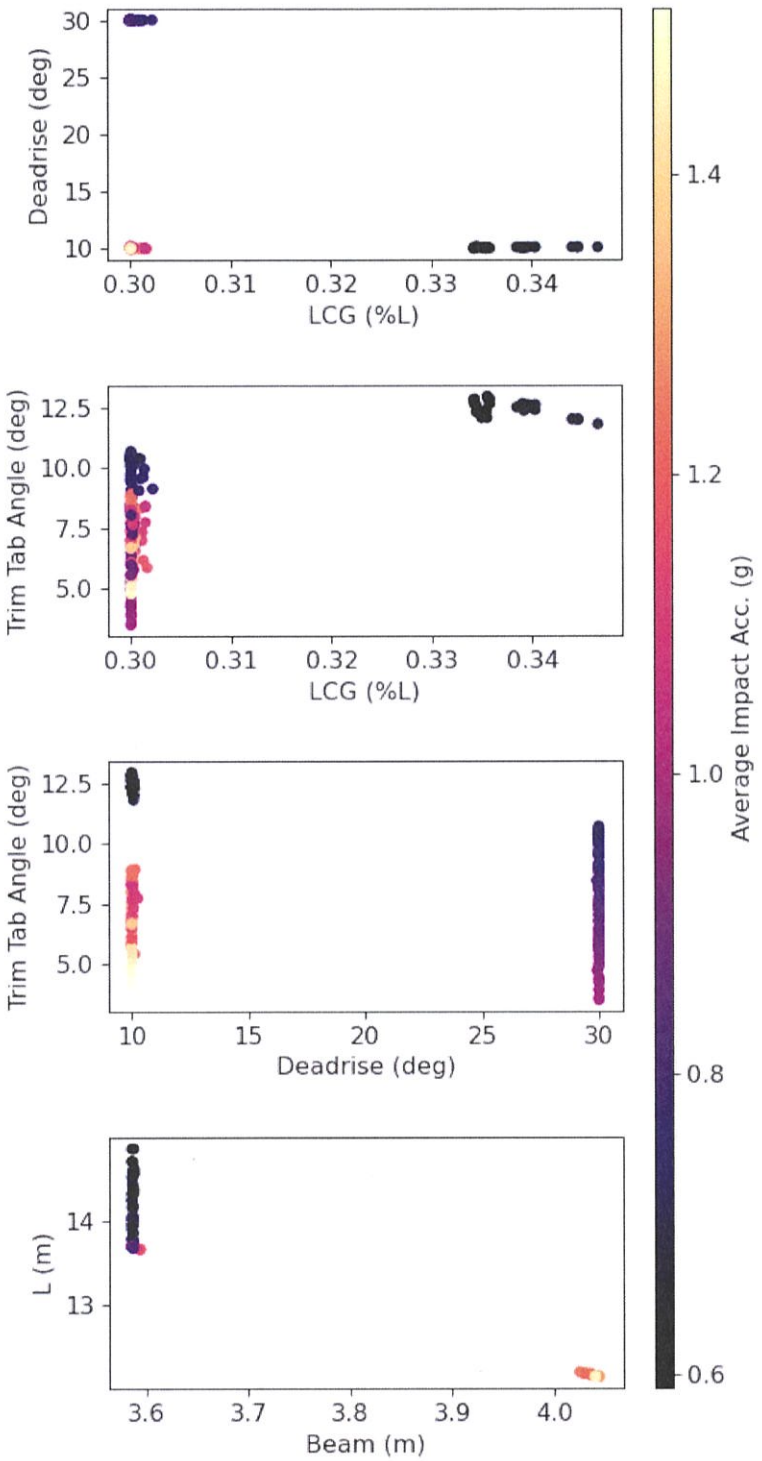


Figura A. 29 Variables Estimadas del Pareto Front Función Objetivo aCG Spam- beam ratio 1.0

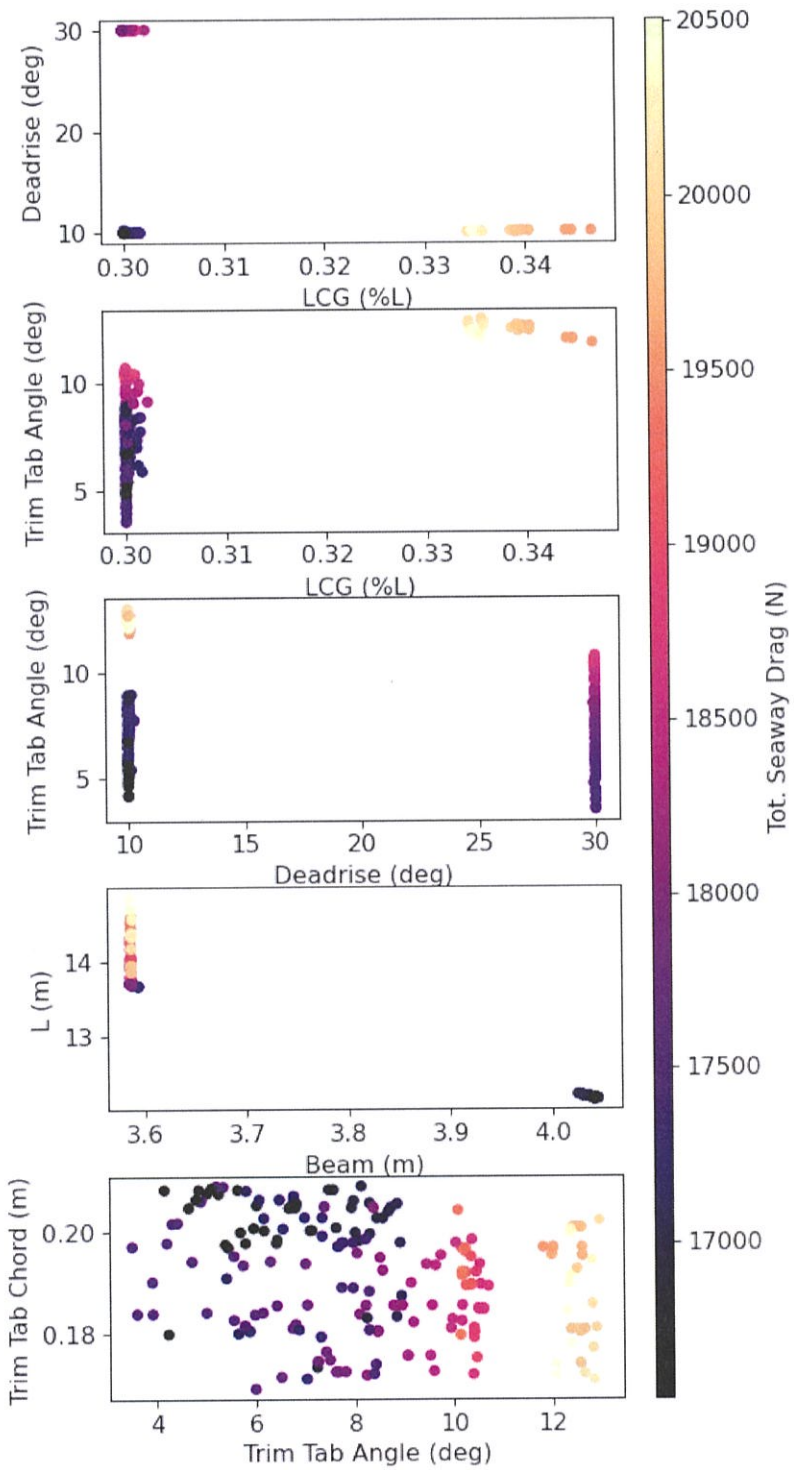


Figura A. 30 Variables Estimadas del Pareto Front Función Objetivo RT Spam- beam ratio 1.0

APÉNDICE B

Diagrama de Flujo del programa

"FlapOptResisandSeakee.py"

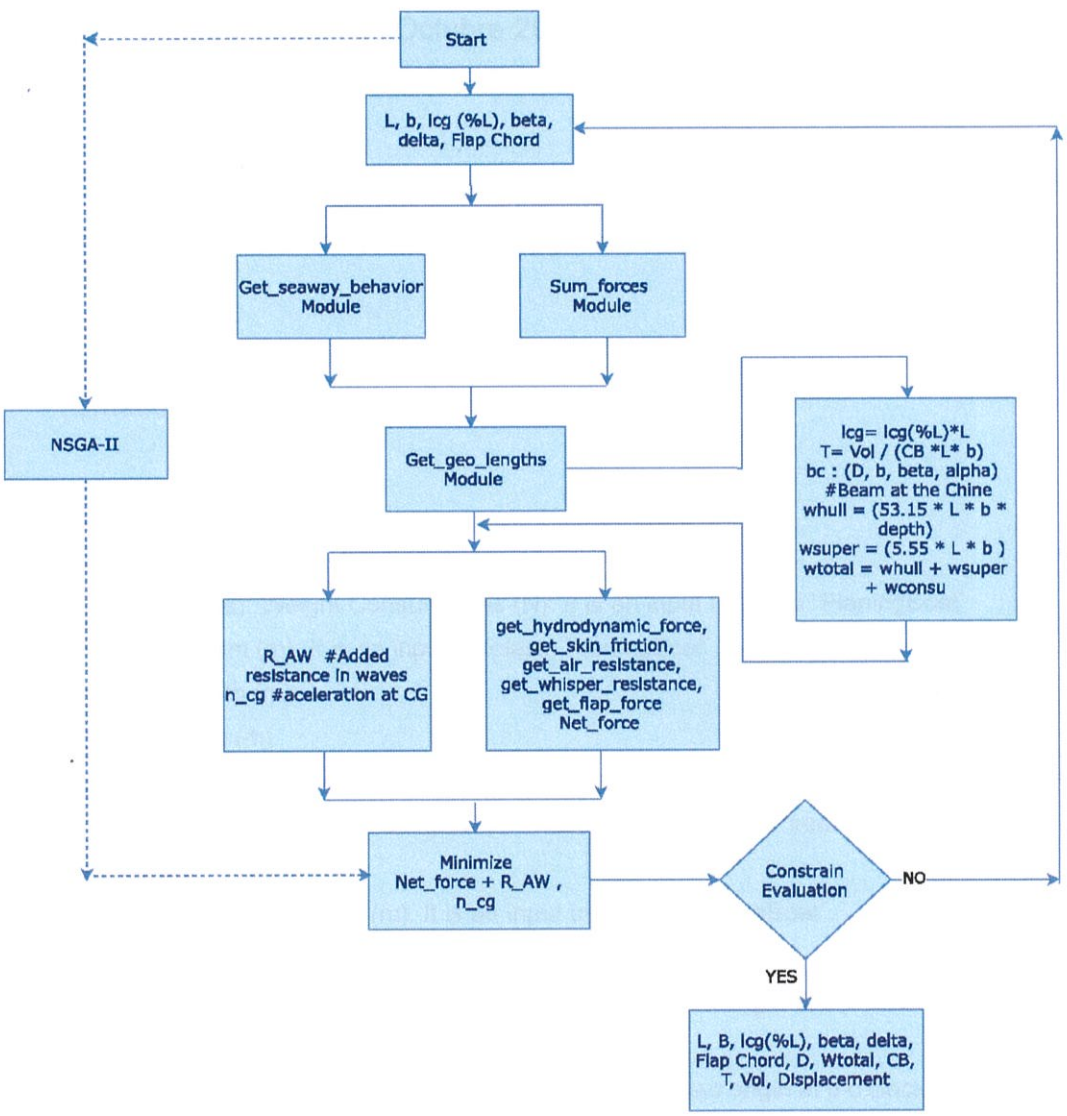


Figura B. 1 Diagrama de flujo del programa utilizado

APÉNDICE C

Código del programa

FlapOptResisandSeakee

Es un software desarrollado usando los paquetes Pymoo (Blank & Deb, 2020) y OpenPlaning (Castro Feliciano, Octubre 2021) en el lenguaje de programación Python, en que se implementan los cálculos de resistencia y comportamiento en olas de las lanchas planeadoras.

Librerías requeridas, descargar en command prompt como: pip install - (nombre de la librería) Numpy, ndmath, spicy, warnings, pymoo.

```
import numpy as np
class PlaningBoat():
    """Prismatic planing craft

    Attributes:
        speed (float): Speed (m/s). It is an input to :class:`PlaningBoat`.
        weightconsu (float): Weight Consumables (N). It is an input to :class:`PlaningBoat`.
        beam (float): Beam (m). It is an input to :class:`PlaningBoat`.
        depth (float): depth (m)
        draft (float) : draft (m)
        lcgpercent (float): Percentage of Longitudinal center of gravity. It is an input to :class:`PlaningBoat`.
        vcg (float): Vertical center of gravity, measured from the keel (m). It is an input to :class:`PlaningBoat`.
        r_g (float): Radius of gyration (m). It is an input to :class:`PlaningBoat`.
        beta (float): Deadrise (deg). It is an input to :class:`PlaningBoat`.
        epsilon (float): Thrust angle w.r.t. keel, CCW with body-fixed origin at 9 o'clock (deg). It is an input to :class:`PlaningBoat`.
        epsilon (float): Thrust angle w.r.t. keel, CCW with body-fixed origin at 9 o'clock (deg). It is an input to :class:`PlaningBoat`.
        vT (float): Thrust vertical distance, measured from keel, and positive up (m). It is an input to :class:`PlaningBoat`.
        lT (float): Thrust horizontal distance, measured from stern, and positive forward (m). It is an input to :class:`PlaningBoat`.
```

length (float): Vessel LOA for seaway behavior estimates (m). Defaults to None. It is an input to :class:`PlaningBoat`.

H_sig (float): Significant wave height in an irregular sea state (m). Defaults to None. It is an input to :class:`PlaningBoat`.

ahr (float): Average hull roughness (m). Defaults to 150*10**-6. It is an input to :class:`PlaningBoat`.

Lf (float): Flap chord (m). Defaults to 0. It is an input to :class:`PlaningBoat`.

sigma (float): Flap span-beam ratio (dimensionless). Defaults to 0. It is an input to :class:`PlaningBoat`.

delta (float): Flap deflection (deg). Defaults to 0. It is an input to :class:`PlaningBoat`.

l_air (float): Distance from stern to center of air pressure (m). Defaults to 0. It is an input to :class:`PlaningBoat`.

h_air (float): Height from keel to top of square which bounds the air-drag-inducing area (m). Defaults to 0. It is an input to :class:`PlaningBoat`.

b_air (float): Transverse width of square which bounds the air-drag-inducing area (m). Defaults to 0. It is an input to :class:`PlaningBoat`.

C_shape (float): Area coefficient for air-drag-inducing area (dimensionless). C_shape = 1 means the air drag reference area is h_air*b_air. Defaults to 0. It is an input to :class:`PlaningBoat`.

C_D (float): Air drag coefficient (dimensionless). Defaults to 0.7. It is an input to :class:`PlaningBoat`.

rho (float): Water density (kg/m^3). Defaults to 1025.87. It is an input to :class:`PlaningBoat`.

nu (float): Water kinematic viscosity (m^2/s). Defaults to 1.19*10**-6. It is an input to :class:`PlaningBoat`.

rho_air (float): Air density (kg/m^3). Defaults to 1.225. It is an input to :class:`PlaningBoat`.

g (float): Gravitational acceleration (m/s^2). Defaults to 9.8066. It is an input to :class:`PlaningBoat`.

z_wl (float): Vertical distance of center of gravity to the calm water line (m). Defaults to 0. It is an input to :class:`PlaningBoat`, but modified when running :meth:`get_steady_trim`.

tau (float): Trim angle (deg). Defaults to 5. It is an input to :class:`PlaningBoat`, but modified when running :meth:`get_steady_trim`.

eta_3 (float): Additional heave (m). Initiates to 0.

eta_5 (float): Additional trim (deg). Initiates to zero.

wetted_lengths_type (int): 1 = Use Faltinsen 2005 wave rise approximation, 2 = Use Savitsky's '64 approach, 3 = Use Savitsky's '76 approach. Defaults to 1. It is an input to :class:`PlaningBoat`.

z_max_type (int): 1 = Uses 3rd order polynomial fit, 2 = Uses cubic interpolation from table. This is only used if wetted_lenghts_type == 1. Defaults to 1. It is an input to :class:`PlaningBoat`.

L_K (float): Keel wetted length (m). It is updated when running :meth:`get_geo_lengths`.

L_C (float): Chine wetted length (m). It is updated when running :meth:`get_geo_lengths`.

lambda_W (float): Mean wetted-length to beam ratio, (L_K+L_C)/(2*beam) (dimensionless). It is updated when running :meth:`get_geo_lengths`.

x_s (float): Distance from keel/water-line intersection to start of wetted chine (m). It is updated when running :meth:`get_geo_lengths`.

z_max (float): Maximum pressure coordinate coefficient, z_max/Ut (dimensionless). It is updated when running :meth:`get_geo_lengths`.

hydrodynamic_force ((3,) ndarray): Hydrodynamic force (N, N, N*m). [F_x, F_z, M_cg] with x, y, rot directions in intertial coordinates. It is updated when running :meth:`get_forces`.

skin_friction ((3,) ndarray): Skin friction force (N, N, N*m). [F_x, F_z, M_cg]. It is updated when running :meth:`get_forces`.

air_resistance ((3,) ndarray): Air resistance force (N, N, N*m). [F_x, F_z, M_cg]. It is updated when running :meth:`get_forces`.

flap_force ((3,) ndarray): Flap resultant force (N, N, N*m). [F_x, F_z, M_cg]. It is updated when running :meth:`get_forces`.

thrust_force ((3,) ndarray): Thrust resultant force (N, N, N*m). [F_x, F_z, M_cg]. It is updated when running :meth:`get_forces`.

net_force ((3,) ndarray): Net force (N, N, N*m). [F_x, F_z, M_cg]. It is updated when running :meth:`get_forces`.

mass_matrix ((2, 2) ndarray): Mass coefficients matrix. [[A_33 (kg), A_35 (kg*m/rad)], [A_53 (kg*m), A_55 (kg*m^2/rad)]]. It is updated when running :meth:`get_eom_matrices`.

damping_matrix ((2, 2) ndarray): Damping coefficients matrix. [[B_33 (kg/s), B_35 (kg*m/(s*rad))], [B_53 (kg*m/s), B_55 (kg*m**2/(s*rad))]]. It is updated when running :meth:`get_eom_matrices`.

restoring_matrix ((2, 2) ndarray): Restoring coefficients matrix. [[C_33 (N/m), C_35 (N/rad)], [C_53 (N), C_55 (N*m/rad)]]. It is updated when running :meth:`get_eom_matrices`.

porpoising (list): [[eigenvalue result (bool), est. pitch settling time (s)], [Savitsky chart result (bool), critical trim angle (deg)]]. It is updated when running :meth:`check_porpoising`.

avg_impact_acc ((2,) ndarray): Average impact acceleration at center of gravity and bow (g's). [n_cg, n_bow]. It is updated when running :meth:`get_seaway_behavior`.

R_AW (float): Added resistance in waves (N). It is updated when running :meth:`get_seaway_behavior`.

"""

Los parámetros que inician el programa son descritos a continuación:

```
def __init__(self, speed, weightconsu, beam, depth, draft, lcgpercent, vcg, r_g, beta, epsilon, vT, lT, length=None, H_sig=None, ahr=150e-6, Lf=0, sigma=0, delta=0, l_air=0, h_air=0, b_air=0, C_shape=0, C_D=0.7, z_wl=0, tau=5, rho=1025.87, nu=1.19e-6, rho_air=1.225, g=9.8066, wetted_lengths_type=1, z_max_type=1):
```

"""Initialize attributes for PlaningBoat

Args:
speed (float): Speed (m/s).
weightconsu (float): Weidght (N).
beam (float): Beam (m).
depth (float): Depth (m)
draft (float) : draft (m)
lcgpercent (float): Percentage of Longitudinal center of gravity.
vcg (float): Vertical center of gravity, measured from the keel (m).
r_g (float): Radius of gyration (m).
beta (float): Deadrise (deg).
epsilon (float): Thrust angle w.r.t. keel, CCW with body-fixed origin at 9 o'clock (deg).
vT (float): Thrust vertical distance, measured from keel, and positive up (m).
lT (float): Thrust horizontal distance, measured from stern, and positive forward (m).
length (float, optional): Vessel LOA for seaway behavior estimates (m). Defaults to None.
H_sig (float, optional): Significant wave heigth in an irregular sea state (m). Defaults to None.
ahr (float, optional): Average hull roughness (m). Defaults to 150*10**-6.
Lf (float, optional): Flap chord (m). Defaults to 0.
sigma (float, optional): Flap span-beam ratio (dimensionless). Defaults to 0.
delta (float, optional): Flap deflection (deg). Defaults to 0.
l_air (float, optional): Distance from stern to center of air pressure (m). Defaults to 0.
h_air (float, optional): Height from keel to top of square which bounds the air-drag-inducing area (m). Defaults to 0.
b_air (float, optional): Transverse width of square which bounds the air-drag-inducing area (m). Defaults to 0.
C_shape (float, optional): Area coefficient for air-drag-inducing area (dimensionless). C_shape = 1 means the air drag reference area is h_air*b_air. Defaults to 0.
C_D (float, optional): Air drag coefficient (dimensionless). Defaults to 0.7.
z_wl (float, optional): Vertical distance of center of gravity to the calm water line (m). Defaults to 0.
tau (float, optional): Trim angle (deg). Defaults to 5.
rho (float, optional): Water density (kg/m^3). Defaults to 1025.87.
nu (float, optional): Water kinematic viscosity (m^2/s). Defaults to 1.19*10**-6.
rho_air (float, optional): Air density (kg/m^3). Defaults to 1.225.
g (float, optional): Gravitational acceleration (m/s^2). Defaults to 9.8066.

wetted_lengths_type (int, optional): 1 = Use Faltinsen 2005 wave rise approximation, 2 = Use Savitsky's '64 approach, 3 = Use Savitsky's '76 approach. Defaults to 1.

z_max_type (int, optional): 1 = Uses 3rd order polynomial fit, 2 = Uses cubic interpolation from table. This is only used if wetted_lenghts_type == 1. Defaults to 1.

"""

```
self.speed = speed
self.weightconsu = weightconsu
self.beam = beam
self.depth = depth
self.draft = draft
self.lcgpercent = lcgpercent
self.vcg = vcg
self.r_g = r_g
self.beta = beta
self.epsilon = epsilon
self.vT = vT
self.lT = lT
self.length = length
self.H_sig = H_sig
self.ahr = ahr
self.Lf = Lf
self.sigma = sigma
self.delta = delta
self.l_air = l_air
self.h_air = h_air
self.b_air= b_air
self.C_shape = C_shape
self.z_wl = z_wl
self.tau = tau
self.eta_3 = 0
self.eta_5 = 0
self.rho = rho
```

```
        self.rho_air = rho_air
        self.C_D = C_D
        self.g = g

        #self.gravity_force = np.array([0, -self.wtotal, 0])

        self.wetted_lengths_type = wetted_lengths_type
        self.z_max_type = z_max_type
```

La Función Print_Description, imprime la descripción de los resultados en pantalla.

```
    def print_description(self, sigFigs=7, runAllFunctions=True):
        """Returns a formatted description of the vessel.

        Args:
          sigFigs (int, optional): Number of significant figures to display. Defaults to 7.
          runAllFunctions (bool, optional): Runs all functions with default values before printing results.
    Defaults to True.
        """
        if runAllFunctions:
            self.get_geo_lengths()
            self.get_forces(runGeoLengths=False)
            self.get_eom_matrices(runGeoLengths=False)
            self.get_seaway_behavior()
            self.check_porpoising()

        wtotal = (self.weightconsu + self.whull + self.wsuper)
        volume = (self.weightconsu + self.whull + self.wsuper) /(self.g*self.rho)

        table = [
            ['---VESSEL---'],
            ['Speed', self.speed, 'm/s'],
```

```
        ['V_k', self.speed*1.944, 'knot'],
        ['Fn (beam)', self.speed/np.sqrt(self.g*self.beam), ''],
        ['Fn (volume)', self.speed/np.sqrt(self.g*((self.weightconsu + self.whull +
self.wsuper)/(self.g*self.rho))**(1/3)), ''],
        [''],
        ['---Weights---'],
        ['Weight Consumables', self.weightconsu, 'N'],
        ['Hull Weight',self.whull,'N'],
        ['SuperStructure Weight',self.wsuper,'N'],
        ['Total Weight', (self.weightconsu + self.whull + self.wsuper) , 'N'],
        ['Mass', (self.weightconsu + self.whull + self.wsuper) /self.g, 'kg'],
        ['Volume', (self.weightconsu + self.whull + self.wsuper) /(self.g*self.rho),
'm\u00B3'],
        [''],
        ['---GEOMETRICS---'],
        ['LOA', self.length, 'm'],
        ['Beam', self.beam, 'm'],
        ['Beam at the Chine', self.bc, 'm'],
        ['Depth', self.depth, 'm'],
        ['Draft', self.draft, 'm'],
        ['Vol', self.Vol, 'm\u00B3' ],
        ['CB', self.CB, '-'],
        ['CAp', self.CAp, 'm'],
        [''],
        ['LCG', self.lcg, 'm from stern'],
        ['VCG', self.vcg, 'm from keel'],
        ['R_g', self.r_g, 'm'],
        ['Deadrise', self.beta, 'deg'], #'\N{greek small letter beta}'
        ['trim Angle',self.tau, 'deg'],
        [''],
        ['AHR', self.ahr, 'm, average hull roughness'],
        [''],
```

['---ATTITUDE---'],

['z_wl', self.z_wl, 'm, vertical distance of center of gravity to the calm water line'],

['tau', self.tau, 'deg, trim angle'],

['\u03B7\u2083', self.eta_3, 'deg, additional heave'],

['\u03B7\u2085', self.eta_5, 'deg, additional trim'],

['Transom draft', self.L_K*np.sin((self.tau+self.eta_5)*np.pi/180), 'm, draft of keel at transom'],

[''],

['---PROPULSION---'],

['Thrust angle', self.epsilon, 'deg w.r.t. keel (CCW with body-fixed origin at 9 o\'clock)'],

['LCT', self.lT, 'm from stern, positive forward'],

['VCT', self.vT, 'm from keel, positive up'],

[''],

['---FLAP---'],

['Chord', self.Lf, 'm'],

['Span/Beam', self.sigma, ''],

['Angle', self.delta, 'deg w.r.t. keel (CCW with body-fixed origin at 9 o\'clock)'],

[''],

['---AIR DRAG---'],

['l_air', self.l_air, 'm, distance from stern to center of air pressure'],

['h_air', self.h_air, 'm, height from keel to top of square which bounds the air-drag-inducing shape'],

['b_air', self.b_air, 'm, transverse width of square which bounds the air-drag-inducing shape'],

['C_shape', self.C_shape, 'area coefficient for air-drag-inducing shape. C_shape = 1 means the air drag reference area is h_air*b_air'],

['C_D', self.C_D, 'air drag coefficient'],

[''],

['---ENVIRONMENT---'],

```
['\u03C1', self.rho, 'kg/m\u00B3, water density'],
['\u03BD', self.nu, 'm\u00B2/s, water kinematic viscosity'],
['\u03C1_air', self.rho_air, 'kg/m\u00B3, air density'],
['g', self.g, 'm/s\u00B2, gravitational acceleration'],
[''],
['---WETTED LENGTH OPTIONS---'],
['wetted_lengths_type', self.wetted_lengths_type, '(1 = Use Faltinsen 2005 wave rise approximation, 2 = Use Savitsky\'s \'64 approach, 3 = Use Savitsky\'s \'76 approach)'],
['z_max_type', self.z_max_type, '(1 = Uses 3rd order polynomial fit (faster, recommended), 2 = Use cubic interpolation)'],
[''],
['---WETTED LENGTHS---'],
['L_K', self.L_K, 'm, keel wetted length'],
['L_C', self.L_C, 'm, chine wetted length'],
['\u03BB', self.lambda_W, 'mean wetted-length to beam ratio (L_K+L_C)/(2*beam)'],
['x_s', self.x_s, 'm, distance from keel/water-line intersection to start of wetted chine'],
['z_max', self.z_max, 'maximum pressure coordinate coefficient (z_max/Ut)'],
[''],
['---FORCES [F_x (N, +aft), F_z (N, +up), M_cg (N*m, +pitch up)]---'],
['Hydrodynamic Force', self.hydrodynamic_force, ''],
['Skin Friction', self.skin_friction, ''],
['Air Resistance', self.air_resistance, ''],
['Whisper Spray Resistance', self.whisper_resistance, ''],
['Flap Force', self.flap_force, ''],
['Net Force', self.net_force, ''],
['Resultant Thrust', self.thrust_force, ''],
[''],
```

```
        ['---THURST & POWER---'],
        ['Thrust Magnitude', np.sqrt(self.thrust_force[0]**2+self.thrust_force[1]**2),
'N'],
        ['Effective Thrust', -self.thrust_force[0], 'N'],
        ['Eff. Power', -self.thrust_force[0]*self.speed/1000, 'kW'],
        ['Eff. Horsepower', -self.thrust_force[0]*self.speed/1000/0.7457, 'hp'],
        [''],
        ['---EOM MATRICES---'],
        ['Mass matrix, [kg, kg*m/rad; kg*m, kg*m\u00B2/rad]', self.mass_matrix, ''],
        ['Damping matrix, [kg/s, kg*m/(s*rad); kg*m/s, kg*m\u00B2/(s*rad)]',
self.damping_matrix, ''],
        ['Restoring matrix, [N/m, N/rad; N, N*m/rad]', self.restoring_matrix, ''],
        [''],
        ['---PORPOISING---'],
        ['[[Eigenvalue check result, Est. pitch settling time (s)],\n [Savitsky chart result,
Critical trim angle (deg)]]', np.array(self.porpoising), ''],
        [''],
        ['---BEHAVIOR IN WAVES---'],
        ['H_sig', self.H_sig, 'm, significant wave heigth'],
        ['R_AW', self.R_AW, 'N, added resistance in waves'],
        ['Average impact acceleration [n_cg, n_bow] (g\'s)', self.avg_impact_acc, ''],
        [''],
        ['---Optimization Process---'],
        ['Calculated Initial Objetive Funtion Values'],
        ['Total Seaway DRAG', self.net_force[0] + self.R_AW , 'N'],
        ['Average Impact Acceleration', self.avg_impact_acc[0],'(g\'s)'],
    ]
```

```
cLens=[16,0,0] #Min spacing for columns
    for row in table:
        if len(row)==3:
            if row[1] is None:
                print('{desc:<{cL0}}  {val:<{cL1}}  {unit:<{cL2}}'.format(desc=row[0],
val=row[1], unit='None', cL0='', cL1=cLens[1], cL2=cLens[2]))
            elif isinstance(row[1], (list,np.ndarray)):
                print(row[0]+' =')
                with np.printoptions(formatter={'float': f'{{:.{sigFigs}g}}'.format}):
                    print(row[1])
                print(row[2])
            else:
                print('{desc:<{cL0}}  {val:<{cL1}.{sNum}g}
{unit:<{cL2}}'.format(desc=row[0],  val=row[1],  unit=row[2],  cL0=cLens[0],
cL1=cLens[1], cL2=cLens[2], sNum=sigFigs))
        else:
            print(row[0])
```

La función Geo_Lengths alimenta al programa del cálculo de las variables, acepta nuevos cálculos que pueden ser implementados en todo el programa, se deben actualizar los valores usando el comando "self"

```
    def get_geo_lengths(self):
        """This function outputs the geometric lengths.

        Adds/updates the following attributes:

        - :attr:`L_K`
        - :attr:`L_C`
        - :attr:`lambda_W`
        - :attr:`x_s`
        - :attr:`z_max`
        - :attr:`lcg`
```

- :attr:`Draft`
- :attr:`bc`
-
"""

```
b = self.beam
weightconsu = self.weightconsu
length = self.length
depth= self.depth
draft = self.draft
lcgpercent = self.lcgpercent
vcg = self.vcg
z_wl = self.z_wl
tau = self.tau
beta = self.beta
eta_3 = self.eta_3
eta_5 = self.eta_5
pi = np.pi
wetted_lengths_type = self.wetted_lengths_type
z_max_type = self.z_max_type

#----------------------LMR-------------------------
#Now update values for calculations in the program
lcg = lcgpercent * self.length

#----------------------LMR-------------------------
#-----------HULL WEIGHT ESTIMATE--------------------
whull  = ( 53.15 * length * b * depth ) * 9.8 #Weigth of W/M Newtons
wsuper = ( 5.55 * length * b  ) * 9.8 # Newtons

wtotal = weightconsu + whull + wsuper  #Newtons
```

```
    #--------------------LMR--------------------------
    Vol = wtotal / (9.8 * 1025)
    CB = Vol / (length * b * draft)
    draft =  Vol / (CB * length * b)
    #--------------------LMR--------------------------
    #This subroutine calculates the Beam of the planing surface if the sides are not straight
    #Slope of the hull side wrt vertical line
    alphaslpe = 20

    #slope of hull wrt horizontal line
    alphac = (pi/2) - (alphaslpe*pi/180)

    #Beam at the chine
    bc = 2* ( ( depth - ( b / 2*np.tan(alphac)) ) / ( np.tan(pi/180*(beta)) - np.tan(alphac) ) )
    #--------------------LMR--------------------------
    #Keel wetted length, Eq. 9.50 of Faltinsen 2005, page 367
    L_K = lcg + vcg / np.tan(pi/180*(tau + eta_5)) - (z_wl + eta_3) / np.sin(pi/180*(tau + eta_5))
    if L_K < 0:
        L_K = 0

    if wetted_lengths_type == 1:
        #z_max/Vt coefficient, Table 8.3 of Faltinsen 2005, page 303----------
        beta_table = [4, 7.5, 10, 15, 20, 25, 30, 40]
        z_max_table = [0.5695, 0.5623, 0.5556, 0.5361, 0.5087, 0.4709, 0.4243, 0.2866]
    #Extrapolation warning
        if beta < beta_table[0] or beta > beta_table[-1]:
            warnings.warn('Deadrise ({0:.3f}) outside the interpolation range of 4-40 deg (Table 8.3 of Faltinsen 2005).
```

```
        if z_max_type == 1:
            z_max = np.polyval([-2.100644618790201e-006, -6.815747611588763e-
005, -1.130563334939335e-003, 5.754510457848798e-001], beta)
        elif z_max_type == 2:
            z_max_func = interpolate.interp1d(beta_table, z_max_table, kind='cubic',
fill_value='extrapolate') #Interpolation of the table
            z_max = z_max_func(beta)
        #------------------------------------------------------------------

        #Distance from keel/water-line intersection to start of wetted chine (Eq. 9.10 of
Faltinsen)
        x_s = 0.5 * bc * np.tan(pi/180*beta) / ((1 + z_max) * (pi/180)*(tau + eta_5))
        if x_s < 0:
            x_s = 0

        #Chine wetted length, Eq. 9.51 of Faltinsen 2005
        L_C = L_K - x_s
        if L_C < 0:
            L_C = 0
            x_s = L_K
            warnings.warn('Vessel operating with dry chines (L_C = 0).', stacklevel=2)

        #Mean wetted length-to-beam ratio
        lambda_W = (L_K + L_C) / (2 * bc)

    elif wetted_lengths_type == 2:
        #Eq. 3 of Savitsky '64
        x_s = bc /pi*np.tan(pi/180*beta)/np.tan(pi/180*(tau + eta_5))

        #Chine wetted length
        L_C = L_K - x_s
        if L_C < 0:
```

```
        L_C = 0
        x_s = L_K
        warnings.warn('Vessel operating with dry chines (L_C = 0).', stacklevel=2)

    #Mean wetted length-to-beam ratio
    lambda_W = (L_K + L_C)/(2*bc)

    #z_max/Vt coefficient (E. 9.10 of Faltinsen 2005 rearranged)
    z_max = 0.5 * bc * np.tan(pi/180*beta) / (x_s * (pi/180)*(tau + eta_5)) - 1

elif wetted_lengths_type == 3:
    #Eq. 12 of Savitsky '76
    w = (0.57 + beta/1000)*(np.tan(pi/180*beta)/(2*np.tan(pi/180*(tau+eta_5)))-
beta/167)

    lambda_K = L_K/bc

    #Eq. 14 of Savitsky '76
    lambda_C = (lambda_K-w)-0.2*np.exp(-(lambda_K-w)/0.3)
    if lambda_C < 0:
        lambda_C = 0
    L_C = lambda_C*bc

    #Mean wetted length-to-beam ratio, Eq. 15 of Savitsky '76
    lambda_W = (lambda_K + lambda_C)/2+0.03

    x_s = L_K-L_C

    #z_max/Vt coefficient (E. 9.10 of Faltinsen 2005 rearranged)
    z_max = 0.5 * bc * np.tan(pi/180*beta) / (x_s * (pi/180)*(tau + eta_5)) - 1
```

```
    if self.length is not None:
        if L_K > self.length:
            warnings.warn('The estimated wetted chine length ({0:.3f}) is larger than the
vessel length ({1:.3f}).'.format(L_K, self.length), stacklevel=2)

    #Estimation of the Centroid of the projected area
    y_c_1 = 0.2 * (0.95 * length) / 3
    y_c_2 = 0.8 * (0.95 * length) / 2
    A_1 = 0.2 * (0.95 * length * bc) /2
    A_2 = 0.8 * (0.95 * length * bc)
    Atotal = A_1 + A_2
    CAp = ( y_c_1 * A_1 + y_c_2 * A_2 ) / Atotal

    #----------Update values----------
    self.L_K = L_K
    self.L_C = L_C
    self.lambda_W = lambda_W
    self.x_s = x_s
    self.z_max = z_max

    self.lcg = lcg
    self.wtotal = wtotal
    self.whull = whull
    self.wsuper = wsuper
    self.gravity_force = np.array([0, -self.wtotal, 0])
    self.CB = CB
    self.draft = draft
    self.Vol = Vol
    self.bc = bc
    self.CAp = CAp
```

La función Get_Forces efectuará el cálculo de resistencia utilizando las componentes: hidrodinámica, aire, friccional, whisper spray y flap. La suma de las componentes es almacenada en la función net_force.

```
def get_forces(self, runGeoLengths=True):
    """This function calls all the force functions to update the respective object attributes.

    Adds/updates the following attributes:

    - :attr:`hydrodynamic_force`
    - :attr:`skin_friction`
    - :attr:`air_resistance`
    - :attr:`whisper_resistance`
    - :attr:`flap_force`
    - :attr:`thrust_force`
    - :attr:`net_force`

    Args:
        runGeoLengths (boolean, optional): Calculate the wetted lengths before calculating the forces. Defaults to True.

    Methods:
        get_hydrodynamic_force(): This function follows Savitsky 1964 and Faltinsen 2005 in calculating the vessel's hydrodynamic forces and moment.
        get_skin_friction(): This function outputs the frictional force of the vessel using ITTC 1957 and the Bowden and Davison 1974 roughness coefficient.
        get_air_resistance(): This function estimates the air drag. It assumes a square shape projected area with a shape coefficient.
        get_whisper_resistance(): This function estimates the whisper spray component using ATTC formula. (Savitsky, Delorme & Datla, 2007)
        get_flap_force(): This function outputs the flap forces w.r.t. global coordinates (Savitsky & Brown 1976). Horz: Positive Aft, Vert: Positive Up, Moment: Positive CCW.
        sum_forces(): This function gets the sum of forces and moments, and consequently the required net thrust. The coordinates are positive aft, positive up, and positive counterclockwise.
    """
    if runGeoLengths:
        self.get_geo_lengths() #Calculated wetted lengths in get_forces()
```

```
g = self.g
rho_air = self.rho_air
C_D = self.C_D
rho = self.rho
nu = self.nu
AHR = self.ahr
W = self.wtotal
epsilon = self.epsilon
vT = self.vT
lT = self.lT
U = self.speed
b = self.bc
lcg = self.lcg
vcg = self.vcg

Lf = self.Lf
sigma = self.sigma
delta = self.delta
beam = self.beam

l_air = self.l_air
h_air = self.h_air
b_air = self.b_air
C_shape = self.C_shape

z_wl = self.z_wl
tau = self.tau
beta = self.beta
eta_3 = self.eta_3
eta_5 = self.eta_5
```

```
        L_K = self.L_K
        L_C = self.L_C
        lambda_W = self.lambda_W
        x_s = self.x_s
        z_max = self.z_max

        pi = np.pi
#----

        def get_hydrodynamic_force():
            """This function follows Savitsky 1964 and Faltinsen 2005 in calculating the vessel's hydrodynamic forces and moment.
            """
            #Beam Froude number
            Fn_B = U/np.sqrt(g*b)

            #Warnings
            if Fn_B < 0.6 or Fn_B > 13:
                warnings.warn('Beam Froude number = {0:.3f}, outside of range of applicability (0.60 <= U/sqrt(g*b) <= 13.00) for planing lift equation. Results are extrapolations.'.format(Fn_B), stacklevel=2)
            if lambda_W > 4:
                warnings.warn('Mean wetted length-beam ratio = {0:.3f}, outside of range of applicability (lambda <= 4) for planing lift equation. Results are extrapolations.'.format(lambda_W), stacklevel=2)
            if tau < 2 or tau > 15:
                warnings.warn('Vessel trim = {0:.3f}, outside of range of applicability (2 deg <= tau <= 15 deg) for planing lift equation. Results are extrapolations.'.format(tau), stacklevel=2)

            #0-Deadrise lift coefficient
            C_L0 = (tau + eta_5)**1.1 * (0.012 * lambda_W**0.5 + 0.0055 * lambda_W**2.5 / Fn_B**2)
```

```
        #Lift coefficient with deadrise, C_Lbeta
        C_Lbeta = C_L0 - 0.0065 * beta * C_L0**0.6

        #Vertical force (lift)
        F_z = C_Lbeta * 0.5 * rho * U**2 * b**2

        #Horizontal force
        F_x = F_z*np.tan(pi/180*(tau + eta_5))

        #Lift's Normal force w.r.t. keel
        F_N = F_z / np.cos(pi/180*(tau + eta_5))

        #Longitudinal position of the center of pressure, l_p (Eq. 4.41, Doctors 1985)
        l_p = lambda_W * b * (0.75 - 1 / (5.21 * (Fn_B / lambda_W)**2 + 2.39)) #Limits
for this is (0.60 < Fn_B < 13.0, lambda < 4.0)

        #Moment about CG (Axis consistent with Fig. 9.24 of Faltinsen (P. 366)
        M_cg = - F_N * (lcg - l_p)

        #Update values
        self.hydrodynamic_force = np.array([F_x, F_z, M_cg])

    def get_skin_friction():
        """This function outputs the frictional force of the vessel using ITTC 1957 and the Bowden and
Davison 1974 roughness coefficient.
        """
        #Surface area of the dry-chine region
        S1 = x_s * b / (2 * np.cos(pi/180*beta))
        if L_K < x_s:
            S1 = S1 * (L_K / x_s)**2

        #Surface area of the wetted-chine region
        S2 = b * L_C / np.cos(pi/180*beta)
```

```
        #Total surface area
        S = S1 + S2
        if S == 0:
            F_x = 0
            F_z = 0
            M_cg = 0
        else:
            #Mean bottom fluid velocity, Savitsky 1964 - Hadler's empirical formula
            V_m = U * np.sqrt(1 - (0.012 * tau**1.1 * np.sqrt(lambda_W) - 0.0065 * beta
* (0.012 * np.sqrt(lambda_W) * tau**1.1)**0.6) / (lambda_W * np.cos(tau * pi/180)))

            #Reynolds number (with bottom fluid velocity)
            Rn = V_m * lambda_W * b / nu

            #'Friction coefficient' ITTC 1957
            C_f = 0.075/(np.log10(Rn) - 2)**2

            #Additional 'friction coefficient' due to skin friction, Bowden and Davison
(1974)
            deltaC_f = (44*((AHR/(lambda_W*b))**(1/3) - 10*Rn**(-1/3)) + 0.125)/10**3

            #Frictional force
            R_f = 0.5 * rho * (C_f + deltaC_f) * S * U**2

            #Geometric vertical distance from keel
            l_f = (b / 4 * np.tan(pi/180*beta) * S2 + b / 6 * np.tan(pi/180*beta) * S1) / (S1
+ S2)

            #Horizontal force
            F_x = R_f * np.cos(pi/180*(tau + eta_5))

            #Vertical force
```

```
            F_z = - R_f * np.sin(pi/180*(tau + eta_5))

            #Moment about CG (Axis consistent with Fig. 9.24 of Faltinsen (P. 366))
            M_cg = R_f * (l_f - vcg)

        #Update values
        self.skin_friction = np.array([F_x, F_z, M_cg])

    def get_air_resistance():
        """This function estimates the air drag. It assumes a square shape projected area with a shape
coefficient.
        """
        if C_shape == 0 or b_air == 0:
            self.air_resistance = np.array([0, 0, 0])
            return

        #Vertical distance from calm water line to keel at LOA
        a_dist = np.sin(pi/180*(tau + eta_5))*(l_air-L_K)

        #Vertical distance from keel to horizontal line level with boat's height
        b_dist = np.cos(pi/180*(tau + eta_5))*h_air

        #Vertical distance from CG to center of square (moment arm, positive is CG
above)
        momArm = z_wl - (a_dist + b_dist)/2

        #Square projected area
        Area = (a_dist+b_dist)*b_air*C_shape
        if Area < 0:
            Area = 0
```

```
        #Horizontal force (Positive aft)
        F_x = 0.5*rho_air*C_D*Area*U**2

        #Vertical force (Positive up)
        F_z = 0

        #Moment (positve CCW)
        M_cg = -F_x*momArm

        #Update values
        self.air_resistance = np.array([F_x, F_x, M_cg])

#----------*LMR*----------

    def get_CF_ATTC(R_ws,CF,C_fattc):
        """This function output cf using ATTC formulation. ITTC Recommended Procedures and Guidelines.
        Resistance and Propulsion Test and Performance Prediction with Skin Frictional Drag Reduction Techniques, 2017."""

        if (0.242/np.sqrt(CF) - np.log10(R_ws*CF))*(0.242/np.sqrt(CF+1) - np.log10(R_ws* CF+1)) < 0 :
            p = CF
            q =  CF + 1
        while np.absolute ((q - p)/ q) > 1e-10 :
            C_fattc = (p + q)/ 2
            if (0.242/np.sqrt(p) - np.log10(R_ws*p))*(0.242/np.sqrt(C_fattc) - np.log10(R_ws*C_fattc)) < 0 :
                q = C_fattc
            else :
                p = C_fattc
        return C_fattc
```

```
def get_whisper_resistance():
    """This function estimates the whisper spray resistance. (Savitsky, Delorme & Datla, 2007)
    """
    #Whisper spray angle
    alfa = (np.arctan((pi*np.tan(pi/180*(tau + eta_5))) /
((2*np.tan(pi/180*beta)))))*180/pi

    #Whisper spray length
    L_ws = b / (4*np.sin(pi/180 * (2 * alfa))*np.cos(pi/180*beta))

    #Reynols of whisper spray
    R_ws = U * L_ws / nu

    #Call CF ATTC funcion
    C_fattc = 0
    CF = 1e-4
    C_fattc = get_CF_ATTC(R_ws,CF,C_fattc)

    #Whisper Spray Force
    F_ws = 0.5 * rho * U**2 * b**2 * C_fattc / (4* np.sin(pi/180*(2*alfa)) *
np.cos(pi/180*(beta)))
    THBG = 2 * alfa / np.cos(pi/180*(beta))

    #Horizontal Component
    F_x = F_ws * np.cos(pi/180*THBG)

    #Vertical component
    F_z = F_ws * np.sin(pi/180*THBG)
    M_cg = 0

    #Update values
    self.whisper_resistance = np.array([F_x, F_z, M_cg])
```

```
    def get_flap_force():
        """This function outputs the flap forces w.r.t. global coordinates (Savitsky & Brown 1976). Horz: Positive Aft, Vert: Positive Up, Moment: Positive CCW.
        """
        if Lf == 0:
            self.flap_force = np.array([0, 0, 0])
            return

        #Warnings
        if Lf > 0.10*(L_K + L_C)/2 or Lf < 0:
            warnings.warn('Flap chord = {0:.3f} outside of bounds (0-10% of mean wetted length) for flap forces estimates with Savitsky & Brown 1976'.format(Lf), stacklevel=2)
        if delta < 0 or delta > 15:
            warnings.warn('Flap deflection angle = {0:.3f} out of bounds (0-15 deg) for flap forces estimates with Savitsky & Brown 1976'.format(delta), stacklevel=2)
        Fn_B = U/np.sqrt(g*b)
        if Fn_B < 2 or Fn_B > 7:
            warnings.warn('Beam-based Froude number Fn_B = {0:.3f} out of bounds (2-7) for flap forces estimates with Savitsky & Brown 1976'.format(Fn_B), stacklevel=2)

        F_z = 0.046*(Lf*3.28084)*delta*sigma*(b*3.28084)*(rho/515.379)/2*(U*3.28084)**2*4.4482

        F_x = 0.0052*F_z*(tau+eta_5+delta)

        l_flap = 0.6*b+Lf*(1-sigma)

        M_cg = -F_z*(lcg-l_flap)

        #Update values
        self.flap_force = np.array([F_x, F_z, M_cg])
```

```
    def sum_forces():
        """This function gets the sum of forces and moments, and consequently the required net thrust.
The coordinates are positive aft, positive up, and positive counterclockwise.
        """
        #Call all force functions-------
        get_hydrodynamic_force()
        get_skin_friction()
        get_air_resistance()
        get_whisper_resistance()
        get_flap_force()
        #----------------------------
        forcesMatrix = np.column_stack((self.gravity_force, self.hydrodynamic_force,
self.skin_friction, self.air_resistance,self.whisper_resistance, self.flap_force)) #Forces
and moments
        F_sum = np.sum(forcesMatrix, axis=1) #F[0] is x-dir, F[1] is z-dir, and F[2] is
moment

        #Required thrust and resultant forces
        T = F_sum[0]/np.cos(pi/180*(epsilon+tau+eta_5)); #Magnitude
        T_z = T*np.sin(pi/180*(epsilon+tau+eta_5)); #Vertical
        T_cg = T*np.cos(pi/180*epsilon)*(vcg - vT) - T*np.sin(pi/180*epsilon)*(lcg - lT);
#Moment about cg

        #Update resultant thurst values
        self.thrust_force = np.array([-F_sum[0], T_z, T_cg])

        #Include resultant thrust forces in sum
        F_sum[1] = F_sum[1]+T_z
        F_sum[2] = F_sum[2]+T_cg

        #Update values
        self.net_force = F_sum
```

```
        #Call functions
        sum_forces()
```

El cálculo del trimado estático se realiza en la función Steady_trim, itera el número de veces necesarias para obtener el trimado estático.

```
    def get_steady_trim(self, x0=[0, 3], tauLims=[0.5, 35], tolF=10**-6, maxiter=50):
        """This function finds and sets the equilibrium point when the vessel is steadily running in calm
    water.
        Updates the following attributes:
        - :attr:`z_wl`
        - :attr:`tau`

        Args:
            x0 (list of float): Initial guess for equilibirum point [z_wl (m), tau (deg)]. Defaults to [0, 3].
            tauLims (list of float): Limits for equilibrium trim search. Defaults to [0.5, 35].
            tolF (float): Tolerance for convergence to zero. Defaults to 10**-6.
            maxiter (float): Maximum iterations. Defaults to 50.
        """
        def _boatForces(x):
            self.z_wl = x[0]/10 #the division is for normalization of the variables
            self.tau = x[1]
            self.get_forces()
            return self.net_force[1:3]

        def _boatForcesPrime(x):
            return ndmath.complexGrad(_boatForces, x)

        def _L_K(x):
            # self.z_wl = x[0]/10
            # self.tau = x[1]
            # self.get_geo_lengths() #No need to call, because ndmath's nDimNewton
    allways calls the obj function before calling this "constraint"
            return [-self.L_K]
```

```
        xlims = np.array([[-np.Inf, np.Inf], tauLims])
        warnings.filterwarnings("ignore", category=UserWarning)
        [self.z_wl, self.tau] = ndmath.nDimNewton(_boatForces, x0, _boatForcesPrime,
tolF, maxiter, xlims, hehcon=_L_K)/[10, 1]
        warnings.filterwarnings("default", category=UserWarning)

    def get_eom_matrices(self, runGeoLengths=True):
        """This function returns the mass, damping, and stiffness matrices following Faltinsen 2005.

        Adds/updates the following parameters:
         - :attr:`mass_matrix`
        - :attr:`damping_matrix`
        - :attr:`restoring_matrix`

        Args:
            runGeoLengths (boolean, optional): Calculate the wetted lengths before calculating the EOM
matrices. Defaults to True.

        Methods:
            get_mass_matrix(): This function returns the added mass coefficients following Sec. 9.4.1 of
Faltinsen 2005, including weight and moment of inertia.
            get_damping_matrix(): This function returns the damping coefficients following Sec. 9.4.1 of
Faltinsen 2005.
            get_restoring_matrix(diffType=1, step=10**-6.6): This function returns the restoring coefficients
following the approach in Sec. 9.4.1 of Faltinsen 2005.
        """
        if runGeoLengths:
            self.get_geo_lengths() #Calculated wetted lengths in get_eom_matrices()

        W = self.wtotal
        U = self.speed
        rho = self.rho
        b = self.bc
        lcg = self.lcg
        tau = self.tau
```

```
beta = self.beta
g = self.g
r_g = self.r_g
eta_5 = self.eta_5

L_K = self.L_K
L_C = self.L_C
lambda_W = self.lambda_W
x_s = self.x_s
z_max = self.z_max
pi = np.pi

def get_mass_matrix():
    """This function returns the added mass coefficients following Sec. 9.4.1 of Faltinsen 2005, including weight and moment of inertia
    """
    #Distance of CG from keel-WL intersection
    x_G = L_K - lcg

    #K constant (Eq. 9.63 of Faltinsen 2005)
    K = (pi / np.sin(pi/180*beta) * gamma(1.5 - beta/180) / (gamma(1 - beta/180)**2 * gamma(0.5 + beta/180)) - 1) / np.tan(pi/180*beta)

    kappa = (1 + z_max) * (pi/180)*(tau + eta_5) #User defined constant

    #Based on Faltinsen's
    A1_33 = rho * kappa**2 * K * x_s**3 / 3
    A1_35 = A1_33 * (x_G - x_s * 3/4)
    A1_53 = A1_35
    A1_55 = A1_33 * (x_G**2 - 3/2 * x_G * x_s + 3/5 * x_s**2)

    #Contribution from wet-chine region
```

```
        if L_C > 0:
            C_1 = 2 * np.tan(pi/180*beta)**2 / pi * K

            A2_33 = (rho * b**3) * C_1 * pi / 8 * L_C / b
            A2_35 = (rho * b**4) * (- C_1 * pi / 16 * ((L_K / b)**2 - (x_s / b)**2) + x_G / b
* A2_33 / (rho * b**3))
            A2_53 = A2_35
            A2_55 = (rho * b**5) * (C_1 * pi / 24 * ((L_K / b)**3 - (x_s / b)**3) - C_1 / 8 *
pi * (x_G / b) * ((L_K / b)**2 - (x_s / b)**2) + (x_G / b)**2 * A2_33 / (rho * b**3))
        else:
            A2_33 = 0
            A2_35 = 0
            A2_53 = 0
            A2_55 = 0

        #Total added mass & update values
        A_33 = A1_33 + A2_33 + W/g # kg, A_33
        A_35 = A1_35 + A2_35 # kg*m/rad, A_35
        A_53 = A1_53 + A2_53 # kg*m, A_53
        A_55 = A1_55 + A2_55 + W/g*r_g**2 # kg*m^2/rad, A_55
        self.mass_matrix = np.array([[A_33, A_35], [A_53, A_55]])

    def get_damping_matrix():
        """This function returns the damping coefficients following Sec. 9.4.1 of Faltinsen 2005
        """
        #Heave-heave added mass (need to substract W/g since it was added)
        A_33 = self.mass_matrix[0,0] - W/g

        if L_C > 0:
            d = 0.5 * b * np.tan(pi/180*beta)
        else:
            d = (1 + z_max) * (pi/180)*(tau + eta_5) * L_K
```

```
        #K constant (Eq. 9.63 of Faltinsen 2005, P. 369)
        K = (pi / np.sin(pi/180*beta) * gamma(1.5 - beta/180) / (gamma(1 - beta/180)**2
* gamma(0.5 + beta/180)) - 1) / np.tan(pi/180*beta)

        #2D Added mass coefficient in heave
        a_33 = rho * d**2 * K

        #Infinite Fn lift coefficient
        C_L0 = (tau + eta_5)**1.1 * 0.012 * lambda_W**0.5

        #Derivative w.r.t. tau (rad) of inf. Fn C_L0
        dC_L0 = (180 / pi)**1.1 * 0.0132 * (pi/180*(tau + eta_5))**0.1 * lambda_W**0.5

        #Derivative w.r.t. tau (rad) of inf. Fn C_Lbeta
        dC_Lbeta = dC_L0 * (1 - 0.0039 * beta * C_L0**-0.4)

        #Damping coefficients & update values
        B_33 = rho / 2 * U * b**2 * dC_Lbeta # kg/s, B_33, Savitsky based
        B_35 = - U * (A_33 + lcg * a_33) # kg*m/(s*rad), B_35, Infinite frequency based
        B_53 = B_33 * (0.75 * lambda_W * b - lcg) # kg*m/s, B_53, Savitsky based
        B_55 = U * lcg**2 * a_33 # kg*m**2/(s*rad), B_55, Infinite frequency based
        self.damping_matrix = np.array([[B_33, B_35], [B_53, B_55]])

    def get_restoring_matrix(diffType=1, step=10**-6.6):
        """This function returns the restoring coefficients following the approach in Sec. 9.4.1 of Faltinsen
2005
        Args:
            diffType (int, optional): 1 (recommended) = Complex step method, 2 = Foward step difference.
Defaults to 1.
            step (float, optional): Step size if using diffType == 2. Defaults to 10**-6.
        """
        def _func(eta):
            self.eta_3 = eta[0]
```

```
        self.eta_5 = eta[1]
        self.get_forces()
        return self.net_force[1:3]

    temp_eta_3 = self.eta_3
    temp_eta_5 = self.eta_5

    if diffType == 1:
        C_full = -ndmath.complexGrad(_func, [temp_eta_3, temp_eta_5])
    elif diffType == 2:
        C_full = -ndmath.finiteGrad(_func, [temp_eta_3, temp_eta_5], 10**-6.6)

    #Reset values
    self.eta_3 = temp_eta_3
    self.eta_5 = temp_eta_5
    self.get_forces()

    #Conversion deg to rad (degree in denominator)
    C_full[0,1] = C_full[0,1] / (pi/180) # N/rad, C_35
    C_full[1,1] = C_full[1,1] / (pi/180) # N*m/rad, C_55

    #Update values
    self.restoring_matrix = C_full

#Call functions
get_mass_matrix()
get_damping_matrix()
get_restoring_matrix()
```

La función Check_porposing se usa para calcular el ángulo de trimado dinámico.

```
def check_porpoising(self, stepEstimateType=1, runGeoLengths=True):
    """This function checks for porpoising.
    Adds/updates the following parameters:
    - :attr:`porpoising` (list):

    Args:
      stepEstimateType (int, optional): Pitch step response settling time estimate type, 1 = -3/np.real(eigVals[0])], 2 = Time-domain simulation estimate. Defaults to 1.
    """

    if runGeoLengths:
      self.get_geo_lengths() #Calculated beam at the chine

    #Eigenvalue analysis
    try:
      self.mass_matrix
    except AttributeError:
      warnings.warn('No Equation Of Motion (EOM) matrices found. Running get_eom_matrices().', stacklevel=2)
      self.get_eom_matrices()

    M = self.mass_matrix
    C = self.damping_matrix
    K = self.restoring_matrix

    nDim = len(M)
    A_ss = np.concatenate((np.concatenate((np.zeros((nDim,nDim)), np.identity(nDim)), axis=1), np.concatenate((-np.linalg.solve(M,K), -np.linalg.solve(M,C)), axis=1))) #State space reprecentation

    eigVals = np.linalg.eigvals(A_ss)
    eig_porpoise = any(eigVal >= 0 for eigVal in eigVals)
```

```
    if stepEstimateType == 1:
        settling_time = -3/np.real(eigVals[0])
    elif stepEstimateType == 2:
        B_ss = np.array([[1],[0],[0],[0]]) #Pitch only
        C_ss = np.array([[1,0,0,0]]) #Pitch only
        D_ss = np.array([[0]])

        system = (A_ss,B_ss,C_ss,D_ss)
        t, y = signal.step(system)
        settling_time = (t[next(len(y)-i for i in range(2,len(y)-1) if abs(y[-i]/y[-1])>1.02)]-
t[0])

    #Savitsky '64 chart method
    C_L = (self.weightconsu + self.whull
+self.wsuper)/(1/2*self.rho*self.speed**2*self.bc**2)
    x = np.sqrt(C_L/2)

    #Warnings
    if x > 0.3 or x < 0.13:
        warnings.warn('Lift Coefficient = {0:.3f} outside of bounds (0.0338-0.18) for
porpoising estimates with Savitsky 1964. Results are extrapolations.'.format(C_L),
stacklevel=2)
    if self.beta > 20:
        warnings.warn('Deadrise = {0:.3f} outside of bounds (0-20 deg) for porpoising
estimates with Savitsky 1964. Results are extrapolations.'.format(self.beta),
stacklevel=2)

    tau_crit_0 = -376.37*x**3 + 329.74*x**2 - 38.485*x + 1.3415
    tau_crit_10 = -356.05*x**3 + 314.36*x**2 - 41.674*x + 3.5786
    tau_crit_20 = -254.51*x**3 + 239.65*x**2 - 23.936*x + 3.0195
```

```
    tau_crit_func = interpolate.interp1d([0, 10, 20], [tau_crit_0, tau_crit_10,
tau_crit_20], kind='quadratic', fill_value='extrapolate')
    tau_crit = tau_crit_func(self.beta)

    if self.tau > tau_crit:
        chart_porpoise = True
    else:
        chart_porpoise = False

    #Update values
    self.porpoising = [[eig_porpoise, settling_time], [chart_porpoise, float(tau_crit)]]
    self.tau_crit = float(tau_crit)

  def out_tau_crit(self):
    """This function just get out Tau_critico calculated previously"""

    self.taucritic = self.tau_crit
    return self.taucritic
```

a función Seakeeping es utilizada para calcular el comportamiento del planeador en el
nar, se calcula la resistencia debido al comportamiento en olas y la aceleración del
entro de gravedad y en la proa

```
  def get_seaway_behavior(self, runGeoLengths=True):
    """This function calculates the seaway behavior as stated in Savitsky & Brown '76.

    Adds/updates the following parameters:
    - :attr:`avg_impact_acc`
    - :attr:`R_AW`
    """
#*********************************LMR*****************************************
    if runGeoLengths:
        self.get_geo_lengths() #Calculated beam at the chine
```

```
        if self.H_sig is None:
            self.H_sig = self.beam*0.5 #Arbitrary wave height if no user-defined wave
height
            warnings.warn('Significant wave height has not been specified. Using
beam*0.5 = {0:.3f} m.'.format(self.H_sig), stacklevel=2)
        if self.length is None:
            self.length = self.beam*3
            warnings.warn('Vessel length has not been specified. Using beam*3 = {0:.3f}
m.'.format(self.length), stacklevel=2)
        H_sig = self.H_sig

        W = self.weightconsu + self.whull + self.wsuper
        beta = self.beta
        tau = self.tau
        pi = np.pi

        Delta_LT = W/9964 #Displacement in long tons
        Delta = Delta_LT*2240 #Displacement in lbf
        L = self.length*3.281 #Length in ft
        b = self.bc*3.281 #Beam of The Planing Surface in ft
        V_K = self.speed*1.944 #Speed in knots
        H_sig = H_sig*3.281 #Significant wave height in ft

        w = self.rho*self.g/(4.448*35.315) #Specific weight in lbf/ft^3
        C_Delta = Delta/(w*b**3) #Static beam-loading coefficient

        #Check that variables are inside range of applicability (P. 395 of Savitsky & Brown
'76)
        P1 = Delta_LT/(0.01*L)**3
        P2 = L/b
        P5 = H_sig/b
        P6 = V_K/np.sqrt(L)
```

```
    if P1 < 100 or P1 > 250:
        warnings.warn('Vessel displacement coefficient = {0:.3f}, outside of range of applicability (100 <= Delta_LT/(0.01*L)^3 <= 250, with units LT/ft^3). Results are extrapolations.'.format(P1), stacklevel=2)
    if P2 < 3 or P2 > 5:
        warnings.warn('Vessel length/beam = {0:.3f}, outside of range of applicability (3 <= L/b <= 5). Results are extrapolations.'.format(P2), stacklevel=2)
    if tau < 3 or tau > 7:
        warnings.warn('Vessel trim = {0:.3f}, outside of range of applicability (3 deg <= tau <= 7 deg). Results are extrapolations.'.format(tau), stacklevel=2)
    if beta < 10 or beta > 30:
        warnings.warn('Vessel deadrise = {0:.3f}, outside of range of applicability (10 deg <= beta <= 30 deg). Results are extrapolations.'.format(beta), stacklevel=2)
    if P5 < 0.2 or P5 > 0.7:
        warnings.warn('Significant wave height / beam = {0:.3f}, outside of range of applicability (0.2 <= H_sig/b <= 0.7). Results are extrapolations.'.format(P5), stacklevel=2)
    if P6 < 2 or P6 > 6:
        warnings.warn('Speed coefficient = {0:.3f}, outside of range of applicability (2 <= V_K/sqrt(L) <= 6, with units knots/ft^0.5). Results are extrapolations.'.format(P6), stacklevel=2)

    R_AW_2 = (w*b**3)*66*10**-6*(H_sig/b+0.5)*(L/b)**3/C_Delta+0.0043*(tau-4) #Added resistance at V_K/sqrt(L) = 2
    R_AW_4 = (Delta)*(0.3*H_sig/b)/(1+2*H_sig/b)*(1.76-tau/6-2*np.tan(beta*pi/180)**3) #V_K/sqrt(L) = 4
    R_AW_6 = (w*b**3)*(0.158*H_sig/b)/(1+(H_sig/b)*(0.12*beta-21*C_Delta*(5.6-L/b)+7.5*(6-L/b))) #V_K/sqrt(L) = 6
    R_AWs = np.array([R_AW_2, R_AW_4, R_AW_6])*4.448 #lbf to N conversion
```

```
    R_AWs_interp = interpolate.interp1d([2,4,6], R_AWs, kind='quadratic',
fill_value='extrapolate')
    R_AW = R_AWs_interp([V_K/np.sqrt(L)])

    n_cg = 0.0104*(H_sig/b+0.084)*tau/4*(5/3-
beta/30)*(V_K/np.sqrt(L))**2*L/b/C_Delta #g, at CG
    n_bow = n_cg*(1+3.8*(L/b-2.25)/(V_K/np.sqrt(L))) #g, at bow
    avg_impact_acc = np.array([n_cg, n_bow])

    #Update values
    self.avg_impact_acc = avg_impact_acc
    self.R_AW = R_AW[0]
```

Para ejecutar el programa y el proceso de optimización se deben ingresar las características generales del planeador.

```
#Vessel Particulars #Angy
speed = 14.4 #m/s
weightconsu = 45630.0 #N
beam = 3.45 #m
depth = beam/2.25 #m
draft = 0.51 #m
length = 12.02 #m  (LOA)
lcgpercent = 0.4 #(0.3 - 0.5 % LOA)
vcg = beam/7 #m
r_g = 0.25*length #m
beta = 14 #deg (estimate) (10-30deg)

# Propulsion
epsilon = 0 #deg
vT = -0.6 #m
lT = 0.7 #m
```

```
# Seaway
H_sig = 1 #m, significant wave height (H_sig/b = 0.2-0.7)

# Trim tab
Lf = 0 #m, flap chord
sigma = 0.35 #flap span / vessel beam ratio
delta = 0 #deg, flap deflection (estimate) (0-15deg)

#Create boat object
boat_2 = PlaningBoat(speed, weightconsu, beam, depth, draft, lcgpercent, vcg, r_g,
beta, epsilon, vT, lT, length, H_sig, Lf=Lf, sigma=sigma, delta=delta)

#Calculates the equilibrium trim and heave for the initial vessel
boat_2.get_steady_trim()
boat_2.print_description()
```

Si se decide usar el programa solo para cálculos de resistencia, no se debe ejecutar el siguiente código.
La siguiente parte del código incluye el proceso de optimización multiobjetivo usando las funciones resistencia al avance y aceleración del centro de gravedad.

```
###.....OPTIMIZATION PROBLEM.....###

import numpy as np
from pymoo.model.problem import Problem

from pymoo.algorithms.nsga2 import NSGA2
from pymoo.util.termination.f_tol import MultiObjectiveSpaceToleranceTermination
#https://pymoo.org/interface/termination.html
from pymoo.optimize import minimize
import logging
```

```
#Normalization values for obj functions
fl = np.array([12000, 0.80])
fu = np.array([25000, 3.5])
#fl = np.array([0, 0]) #No normalization
#fu = np.array([1, 1])
```

Se describen los límites de las restricciones. Las restricciones son de tipo desigualdad <=.

```
#----------LMR----------
#  Trim angle constraints using porpoising angle as limit
#  Next function calls porpoising angle
tau_crit = boat_2.out_tau_crit()
tau_l = 2
tau_u = tau_crit

#    L/B constrains (Savitsky & Koelbel, 1993) Increasing L/b produces better seakeeping. Narrower beam reduces acg
LB_l = 3.0
LB_u = 5.0

#    vcg (KG) constrain following (Lewandowski, 1998) The maximum KG for the dynamic moment to be stabilizing eq (31)
KG_l = 0.4
KG_u = 3.63

#  Freeboard constrain using as reference the design draft of the parent boat
fbd_l = 0.68
fbd_u = 0.78

#  Load Area constrain following (Marin, 2007) min area required per passenger 0.56 m^2
```

```
aload_l = 21.11 #29 passengers
aload_u = 30.0

#  Dynamic Transverse Instabilities
dti_lcg_l = 0.03
dti_lcg_u = 1.0

dti_hull_l = 5.8
dti_hull_u = 9.0
```

Se debe describir el número de variables, objetivos y número de restricciones, las variables deben seguir el orden especificado por el arreglo X, y son delimitadas entre el límite inferior y superior.

```
class boatOptProblem(Problem):

    def __init__(self, boat_object):

        self.boat = boat_object

        super().__init__(n_var=6,
                         n_obj=2,
                         n_constr=12,
                         xl=np.array([9.5, 3.2, 0.3, 10.0, 0.0, 0.15]), #Range of Applicability
(Savitsky & Brown, 1976) [ 3 < L/b < 5 ], [ 10 < Deadrise < 30]
                         xu=np.array([15.0, 5.0, 0.5, 30.0, 15.0, 0.211]), #Range of Applicability
(Savitsky & Brown, 1976) [ 0 < delta < 15] , [ 0 < chord < 10% lamda]
                         elementwise_evaluation=True)

    def _evaluate(self, x, out, *args, **kwards):
```

```
#Optimization variables
[self.boat.length, self.boat.beam, self.boat.lcgpercent, self.boat.beta,
self.boat.delta, self.boat.lf] = x

#Attempt to get steady trim, and fail inequality constraint if not successful
g4 = 0
try:
   self.boat.get_steady_trim()
except RuntimeError:
   g4 = 1

#Run the function needed for seakeeping
self.boat.get_seaway_behavior()
#Objective functions and normalization
f1 = ((self.boat.net_force[0] + self.boat.R_AW) - fl[0])/(fu[0]-fl[0])
f2 = (self.boat.avg_impact_acc[0] - fl[1])/(fu[1]-fl[1])

#Inequality constraints (g(x)<=0)
g1 = self.boat.tau - tau_u
g2 = tau_l - self.boat.tau
g3 = self.boat.L_K - self.boat.length*.9
g5 = (self.boat.length/self.boat.beam) - LB_u
g6 = LB_l - (self.boat.length/self.boat.beam)
g7 = self.boat.beam/7 - KG_u
g8 = KG_l - self.boat.beam/7
g9 = (1 - (2.25 * self.boat.draft /self.boat.beam)) - fbd_u
g10 = fbd_l - (1 - (2.25 * self.boat.draft /self.boat.beam))
g11 = (0.43*self.boat.length*self.boat.beam) - aload_u
g12 = aload_l - (0.43*self.boat.length*self.boat.beam)
#g13 = dti_lcg_l - ( (self.boat.CAp - self.boat.lcg) / 0.95*self.boat.length)
#g15 = (self.boat.CAp / self.boat.Vol**(2/3)) - dti_hull_u
#g16 = dti_hull_l - (self.boat.CAp / self.boat.Vol**(2/3))
```

```
        out['F'] = [f1, f2]
        out['G'] = [g1, g2, g3, g4, g5, g6, g7, g8, g9, g10, g11, g12]

algorithm = NSGA2(pop_size=200)
termination = MultiObjectiveSpaceToleranceTermination(tol=0.0025,
                                                      n_last=20,
                                                      nth_gen=5,
                                                      n_max_gen=500,
                                                      n_max_evals=None)

#Run optimization
logging.captureWarnings(True) #Silence warnings
start_time = time.time()
result = minimize(boat_problem,
                  algorithm,
                  termination,
                  seed=1,
                  verbose=True)
elapsed_time = time.time() - start_time
logging.captureWarnings(False)
print(f"The elapsed wall time was {elapsed_time:.0f} s")
```

Los resultados serán impresos en la misma carpeta que contenga el programa, se encontrará como out.xlsx junto con las imágenes del pareto y las variables estimadas.

```
#----------Results------#
import pandas as pd
import matplotlib.pyplot as plt

#Creating a dataframe for convenient data manipulation
```

```
= pd.DataFrame(result.X, columns = ['L (m)', 'Beam (m)','LCG (%L)', 'Deadrise
eg)', 'Trim Tab Angle (deg)','Trim Tab Chord (m)'])
[['Tot. Seaway Drag (N)','Average Impact Acc. (g)']] = pd.DataFrame(result.F*(fu - fl)
fl)
[['tau - '+str(tau_u), str(tau_l)+' - tau','L_K - 0.9*length', 'Steady Trim Fail','L/B - '
str(LB_u), str(LB_l)+' - L/B', 'KG - ' +str(KG_u), str(KG_l)+' - KG','fbd - ' +str(fbd_u),
r(fbd_l)+' - fbd', 'Aload - ' +str(aload_u), str(aload_l)+' - Aload', =
d.DataFrame(result.G)
f['Weight (N)'] = (df['L (m)'] * df[ 'Beam (m)' ] * df[ 'Beam (m)' ]/2.25 * 53.15 * 9.8 ) +
df['L (m)'] * df[ 'Beam (m)' ] * 5.55 * 9.8 ) + (weightconsu)
f['Tot. Seaway Drag/Weight'] = df['Tot. Seaway Drag (N)']/df['Weight (N)']
f['LCG (m)'] = df['LCG (%L)'] * df['L (m)']
f['Calm Water Trim (deg)'] = df['tau - '+str(tau_u)] + tau_u
lf.to_excel("out.xlsx")
lf
rint (df)

#Plot font size
FONT_SIZE = 11.5
plt.rc('font', size=FONT_SIZE) # controls default text sizes
plt.rc('axes', titlesize=FONT_SIZE) # fontsize of the axes title
plt.rc('axes', labelsize=FONT_SIZE) # fontsize of the x and y labels
plt.rc('xtick', labelsize=FONT_SIZE) # fontsize of the tick labels
plt.rc('ytick', labelsize=FONT_SIZE) # fontsize of the tick labels
plt.rc('legend', fontsize=FONT_SIZE) # legend fontsize
plt.rc('figure', titlesize=FONT_SIZE) # fontsize of the figure title
fig_1, axs_1 = plt.subplots(figsize=[5,5])
df.plot.scatter('Tot. Seaway Drag (N)', 'Average Impact Acc. (g)', sharex=False,
ax=axs_1);
axs_1.grid(True)
fig_1.savefig('pareto.png')
plt.show()
```

```
fig_2, axs_2 = plt.subplots(7, 1, figsize=[5,14])
df.plot.scatter('Tot. Seaway Drag (N)', 'Average Impact Acc. (g)', c='L (m)',
colormap='cividis', sharex=True, ax=axs_2[0]);
df.plot.scatter('Tot. Seaway Drag (N)', 'Average Impact Acc. (g)', c='LCG (%L)',
colormap='viridis', sharex=True, ax=axs_2[1]);
df.plot.scatter('Tot. Seaway Drag (N)', 'Average Impact Acc. (g)', c='Beam (m)',
colormap='plasma', ax=axs_2[2]);
df.plot.scatter('Tot. Seaway Drag (N)', 'Average Impact Acc. (g)', c='Deadrise (deg)',
colormap='inferno', ax=axs_2[3]);
df.plot.scatter('Tot. Seaway Drag (N)', 'Average Impact Acc. (g)', c='Trim Tab Angle
(deg)', colormap='cividis', ax=axs_2[4]);
df.plot.scatter('Tot. Seaway Drag (N)', 'Average Impact Acc. (g)', c='Trim Tab Chord
(m)', colormap='magma', ax=axs_2[5]);
df.plot.scatter('Tot. Seaway Drag (N)', 'Average Impact Acc. (g)', c='Calm Water Trim
(deg)', colormap='plasma', ax=axs_2[6]);
axs_2[0].set_ylabel(None);
axs_2[1].set_ylabel(None);
axs_2[3].set_ylabel(None);
axs_2[4].set_ylabel(None);
axs_2[5].set_ylabel(None);
axs_2[6].set_ylabel(None);
fig_2.tight_layout()
fig_2.savefig('pareto_vars.png')
plt.show()
```

```
fig_3, axs_3 = plt.subplots(5, 1, figsize=[5.5,12])
color_values = 'Tot. Seaway Drag (N)'
df.plot.scatter('LCG (%L)', 'Deadrise (deg)', c=color_values, colormap='magma',
sharex=False, ax=axs_3[0], colorbar=False);
df.plot.scatter('LCG (%L)', 'Trim Tab Angle (deg)', c=color_values, colormap='magma',
ax=axs_3[1], colorbar=False);
df.plot.scatter('Deadrise (deg)', 'Trim Tab Angle (deg)', c=color_values,
colormap='magma', ax=axs_3[2], colorbar=False);
df.plot.scatter('Beam (m)', 'L (m)', c=color_values, colormap='magma', ax=axs_3[3],
colorbar=False );
df.plot.scatter( 'Trim Tab Angle (deg)','Trim Tab Chord (m)', c=color_values,
colormap='magma', ax=axs_3[4], colorbar=False);
plt.subplots_adjust( hspace = 0.3 )
im = plt.gca().get_children()[0]
cbar = fig_3.colorbar(im, ax=axs_3.ravel().tolist(), aspect=80)
cbar.set_label(color_values)
fig_3.savefig('vars.png')
plt.show()

fig_4, axs_4 = plt.subplots(4, 1, figsize=[5.5,12])
color_values = 'Average Impact Acc. (g)'
df.plot.scatter('LCG (%L)', 'Deadrise (deg)', c=color_values, colormap='magma',
sharex=False, ax=axs_4[0], colorbar=False);
df.plot.scatter('LCG (%L)', 'Trim Tab Angle (deg)', c=color_values, colormap='magma',
ax=axs_4[1], colorbar=False);
df.plot.scatter('Deadrise (deg)', 'Trim Tab Angle (deg)', c=color_values,
colormap='magma', ax=axs_4[2], colorbar=False);
df.plot.scatter('Beam (m)', 'L (m)', c=color_values, colormap='magma', ax=axs_4[3],
colorbar=False );
plt.subplots_adjust( hspace = 0.5 )
im = plt.gca().get_children()[0]
cbar = fig_4.colorbar(im, ax=axs_4.ravel().tolist(), aspect=80)
```

```
cbar.set_label(color_values)
fig_4.savefig('bars.png')
plt.show()

fig_5,axs_5 = plt.subplots(3, 1, figsize=[5.5,12])
color_values = 'Average Impact Acc. (g)'
df.plot.scatter( 'Trim Tab Angle (deg)','Trim Tab Chord (m)', c=color_values,
colormap='magma', ax=axs_5[0], colorbar=False);
df.plot.scatter( 'Calm Water Trim (deg)','Trim Tab Chord (m)', c=color_values,
colormap='magma', ax=axs_5[1], colorbar=False);
df.plot.scatter( 'Calm Water Trim (deg)','Trim Tab Angle (deg)', c=color_values,
colormap='magma', ax=axs_5[2], colorbar=False);
im = plt.gca().get_children()[0]
cbar = fig_5.colorbar(im, ax=axs_5.ravel().tolist(), aspect=80)
cbar.set_label(color_values)
fig_5.savefig('bars2.png')
plt.show()
```