UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Naval y Oceánica

Estudio del barco viga de un buque mediante diversos métodos numéricos. Aplicación a un Ro-Ro.

GRADO EN ARQUITECTURA NAVAL E INGENIERÍA DE SISTEMAS MARINOS

Autor: Clara Salmerón Bermúdez
Directora del trabajo: Sonia Busquier Sáez
Codirector del trabajo: Juan Ruiz Álvarez

Universidad Politécnica de Cartagena

AGRADECIMIENTOS

Primero de todo, me gustaría agradecer a mis tutores de TFG, por la ayuda prestada para poder llevar a cabo este proyecto. He aprendido mucho, y admiro la pasión y la entrega que realizáis día a día por vuestro trabajo y por vuestros alumnos.

A Marta, Yesenia, Borja, Victoria, Germán, Samuel y Nico, por haberme acompañado y apoyado desde el principio y durante todos estos años para seguir adelante. Sé que este no será el fin de nuestras historias. Y a las nuevas amistades, por permitirme entrar en vuestras vidas y descubrir nuevos mundos.

Por último, a mi familia, por el apoyo incondicional sin el que no hubiese podido realizar estos estudios.

OBJETIVOS

Este trabajo se plantea con el objetivo de simplificar y generalizar el cálculo del Barco Viga estudiado. La idea surge a partir de la motivación por aprender a emplear el lenguaje de MATLAB en mayor profundidad y ser capaz de diseñar una interfaz gráfica útil y simple de uso para el usuario, de forma que se puedan analizar los resultados obtenidos.

En el cálculo del Barco Viga, se emplean frecuentemente integrales definidas de funciones que pueden llegar a ser complejas, es por esto por lo que se emplean métodos de cuadratura que nos ayudan a obtener soluciones próximas a la real. Para buques de gran eslora y formas complejas, esto puede llegar a ser costoso desde el punto de vista computacional. Es por esto por lo que se pretende simplificar estos cálculos mediante la interfaz propuesta.

Otro objetivo propuesto es la capacidad de aplicar esta programación al buque seleccionado para el proyecto y obtener los resultados después de realizar un alargamiento del buque.

Este alargamiento se plantea con el objetivo de aumentar la capacidad de carga, y la posibilidad de aplicar un método estadístico de cálculo de potencia que nos permite determinar si el motor instalado es adecuado, también realizaremos el estudio del buque viga para esta situación del buque ya alargado.

El método estadístico empleado para el cálculo de la resistencia al avance del buque también tendrá una interfaz gráfica propia, ya que resulta interesante ver de forma gráfica los resultados obtenidos, de esta forma se permite simplificar los cálculos como se ha comentado anteriormente.

Índice

Índice de Figuras

Índice de Tablas

1 Motivación

La razón por la cual este proyecto se basa en el cálculo estructural de un buque es, principalmente, debido a una de las asignaturas más importantes cursadas durante el grado, *Diseño y Cálculo de Estructuras Navales*. Ésta da a conocer profundamente los conceptos básicos de la Arquitectura Naval y es la razón por la que resulta interesante trabajar sobre ella.

Otra de las motivaciones por la que el trabajo se realiza en *Matlab*, tiene como origen una asignatura de primer curso, *Fundamentos de Informática*, donde se aprende a desenvolverse con el programa. Para alguien primerizo sorprende el funcionamiento de lo que programas tan comunes en esta rama de la tecnología naval como *Maxsurf*, implementan dentro de su código para realizar cálculos tan complejos. Es por esto por lo que se escoge la herramienta *Matlab*, pudiendo aprender de forma mucho más amplia su lenguaje y aplicaciones.

El objeto de trabajo seleccionado para comprobar los distintos cálculos aplicados es un buque de tipo "*Roll-on Roll-off*" (posteriormente denominado como *Ro-Ro*). Este tipo de buques se dedican al transporte de carga rodada como automóviles, camiones o trenes. Son interesantes de analizar debido al pequeño número de mamparos que contiene, y que estructuralmente debe ser bien estudiado. Un ejemplo de este tipo de buques se puede observar en la siguiente figura.

Figura 1.1 – Buque Ro-Ro Amadeo Matacena.[1]

[1] «File:Ro-Ro ship Amedeo Matacena (IMO 8213940).jpg ».

2 Introducción

Los orígenes de la navegación se remontan a muchos siglos atrás. El ser humano ha necesitado adentrarse en el mar tanto con fines exploratorios como para obtener alimento. En sus inicios, se sirvieron de balsas fabricadas con troncos de madera atados, pasando por las grandes embarcaciones de vela de madera, hasta llegar a las nuevas tecnologías que conocemos hoy en día. Aquellos que salen a navegar deben enfrentarse a todo tipo de adversidades que en ocasiones pueden llegar a dañar el barco.

El principio de Arquímedes afirma que todo cuerpo sumergido en un fluido experimenta un empuje vertical y con sentido hacia arriba igual al peso de fluido desalojado. [2] A partir de éste, se basan todas las teorías para los cálculos de distintos parámetros hidrostáticos, necesarios para determinar la estabilidad del buque entre otras características de las que en este proyecto no se profundizará.

El buque seleccionado, como anteriormente se ha comentado, se trata de un buque diseñado y construido para el transporte de carga rodada, que accede por sus propios medios a las bodegas mediante la configuración de rampas, que se pueden hacer fijas a tierra, para hacer práctica y segura la carga y descarga.

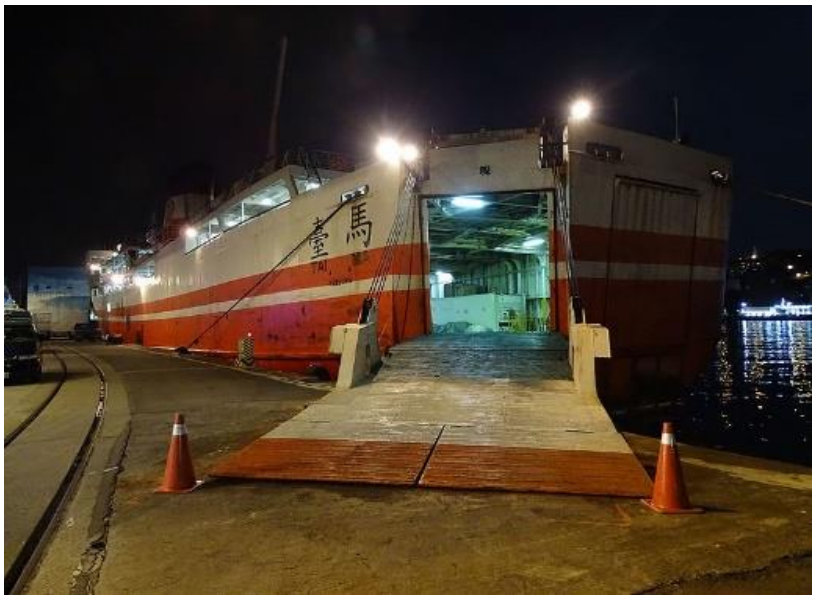


Figura 2.1 - Rampa de popa del buque Tai Ma de Tønsberg.[3]

[2] Terán, «PRINCIPIO DE ARQUÍMEDES».

[3] «File:Tai Ma Ship in Port of Keelung 20140909 night.jpg|Tai Ma Ship in Port of Keelung 20140909 night» Chang, *中文（繁體）*.

Figura 2.2 - Rampa de proa del ferry Nils Holgersson.[4]

En esta categoría de buques Ro-Ro, cabe distinguir los *"car-carriers"*, caracterizados por su forma paralepipédica y porque transportan exclusivamente coches, mientras que los más tradicionales pueden llegar a transportar hasta tráilers y/o vehículos industriales. Además, existen buques capaces de transportar de forma adicional al cargamento, a personas, cuando se superan los doce pasajeros, se denominan utilizando el acrónimo *"Ro-PAX"* o como comúnmente se les denomina, *Ferries*. También existe una combinación entre un Ro-Ro y un buque portacontenedores, el buque *"Con-Ro"* cuando se transportan contenedores sobre la cubierta principal.

Un rasgo característico que distingue a los buques de cargas rodadas de los demás, es la cubierta o cubiertas abiertas que permiten que se desplacen a lo largo de toda la eslora del buque hasta su posición de estiba. Esto nos lleva al problema principal de estos buques: no disponen de mamparos de subdivisión transversal, que suelen incorporarse para mantener la estabilidad con avería o la integridad estanca del buque en caso de inundación de cualquiera de los compartimentos. Ésto no sólo hace que el barco pierda su flotabilidad inherente, sino que también afecta de forma negativa a su estabilidad debido al aumento del efecto de superficie libre. Todos los buques de transporte rodado tienen un francobordo considerable, lo que significa que funcionan con un bajo calado.

El principal requisito que se debe comprobar es que el buque sea capaz de navegar de forma estable estructuralmente, esto es, que las tensiones generadas por el peso y el empuje sean admisibles según el acero utilizado. Para ello, se aplica la Teoría del Buque como Viga que supone que el buque se encuentra en una situación de mar donde se le obliga a una flexión más severa, generalmente es causada por una longitud de onda igual a la eslora. En función de la posición de la cresta, el buque se encontrará en la situación de quebranto o arrufo. También se supondrá que el buque está momentáneamente en reposo sobre la ola, es decir, un equilibrio estático.

[4] «File:Roro faehre.jpg|Roro faehre» Hägele, *English*.

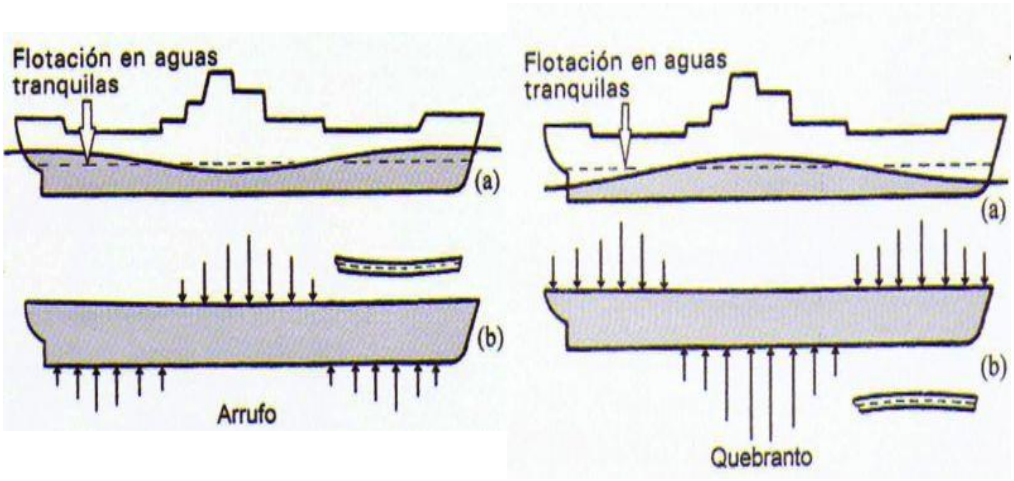


Figura 2.3 - Representación gráfica de las situaciones del buque sobre la ola.

El trabajo se basa en el planteamiento de un buque sobre una ola de tipo senoidal, ya sea sobre una cresta o sobre un seno, que generarán distintos momentos flectores. También será posible analizar el caso del buque en mar en calma. Las formas del buque se aproximarán, siempre que se pueda, a trapecios para obtener una distribución de pesos aproximada en la medida de lo posible a la realidad, y teniendo en cuenta las posibles discontinuidades o pesos puntuales como pueden ser grúas para carga y descarga.

La aplicación de las reglas de Simpson nos permite calcular de forma aproximada las integrales complejas que se obtienen a partir de la curva de cargas, parametrizada a partir de la distribución de pesos y empujes. Dentro de esta curva, existirán máximos y mínimos relativos, y cortes con el eje, que nos permiten conocer de forma aproximada las formas que tendrán las curvas de fuerzas cortantes y momentos flectores.

2.1 Historia del arte

Los buques Roll On – Roll Off.

El primer ferry se construyó en 1861, en Escocia, con el nombre *Forfarshire*. Su principal función era el transporte de vagones de tren entre los márgenes de los ríos que eran demasiado anchos para los puentes.

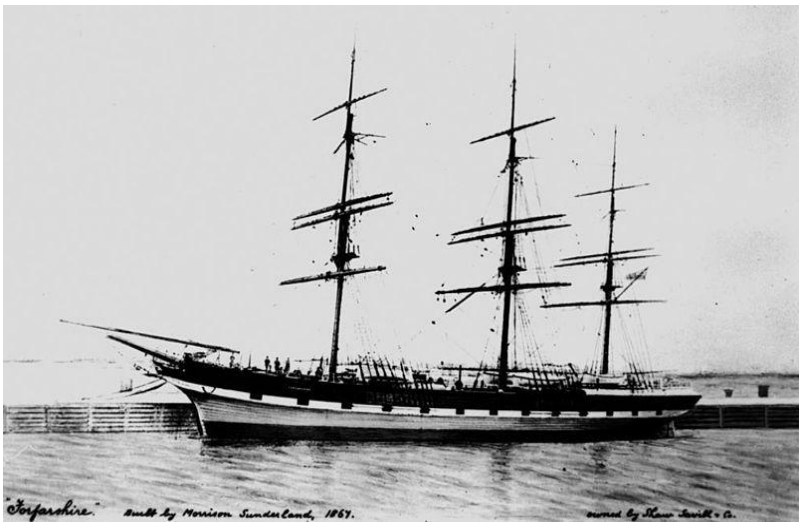


Figura 2.4 - Buque Forfarshire.[5]

[5] «File:*Forfarshire* (Ship)».

El desarrollo de este tipo de buques se motivó en la Segunda Guerra Mundial para el desembarco de equipos militares de transporte por tierra, aunque posteriormente muchos de éstos fueron convertidos para operar como barcos mercantes. El uso del concepto *Ro-Ro* en los buques mercantes comenzó a finales de los años cuarenta y principio de los cincuenta (1940-1950), principalmente en las rutas marítimas de corta distancia.

Su flexibilidad, capacidad de integración con otros sistemas de transporte y velocidad de operación han hecho que este tipo de buques sea cada vez más popular en muchas rutas marítimas. El sistema de trabajo consiste en el transporte desde la fábrica, donde se cargan los camiones, hasta el almacén terminal donde se descargan, de esta manera, el radio de acción de los operadores terrestre aumenta. El *Ro-Ro* nos permite desplazar una mayor carga de volumen, aprovechando al máximo el espacio disponible mediante puntales de bodega ajustables. Como causa directa de este servicio, disminuyen considerablemente los gastos en gasolina, vehículos y conductores.

A pesar de su éxito comercial, se han producido accidentes preocupantes. Después del *Titanic*, el hundimiento repentino y catastrófico del *M/S Herald of Free Enterprise* un viernes 6 de marzo de 1987 se considera como el peor accidente británico, junto con la pérdida todavía más trágica del *M/S Estonia* en septiembre de 1994. A consecuencia de estos incidentes, la Organización Marítima Internacional (OMI)[6] aprobó una serie de modificaciones al Convenio Internacional para la Seguridad de la Vida Humana en el Mar (SOLAS)[7] que tienen como objeto garantizar que no se produzcan este tipo de accidentes.

Robert Fulton.

Figura 2.5 - Robert Fulton.[8]

Nació el 14 de noviembre de 1765 cerca de Lancaster, Pensilvania. Desde muy temprana edad, se interesó en la construcción de dispositivos mecánicos, aunque en su edad adolescente se centró más en el arte. A sus 18 años, se trasladó a vivir a Filadelfia donde pudo mantenerse económicamente gracias a la venta de retratos y dibujos.

[6] «Organización Marítima Internacional».
[7] «Convenio internacional para la seguridad de la vida humana en el mar, 1974 (Convenio SOLAS)».
[8] «Robert Fulton».

Vivió un tiempo en Inglaterra donde su arte comenzó a ser menos importante y dado que Inglaterra se encontraba en plena revolución industrial, Fulton se sorprendió por los grandes avances en las áreas de construcción de canales, puentes, fábricas, etc. Su trabajo sobre las vías navegables interiores ultimó en el *Tratado sobre el Mejoramiento de la Navegación por Canales*, en 1796.

Más tarde en Francia, se interesó por la idea de un *"barco hundido"*, donde trató de interesar al gobierno francés, que se encontraba en guerra con Inglaterra, bajo la idea de que este submarino pudiera ser utilizado para colocar minas bajo los barcos de guerra enemigos. Los franceses estuvieron de acuerdo, y se lanzó el primer submarino por encargo de Napoleón Bonaparte, con el nombre de *Nautilus*, aunque su rendimiento no fue el esperado debido a la baja velocidad durante las pruebas y que los buques ingleses eran capaces de evitarlo, fue desechado.

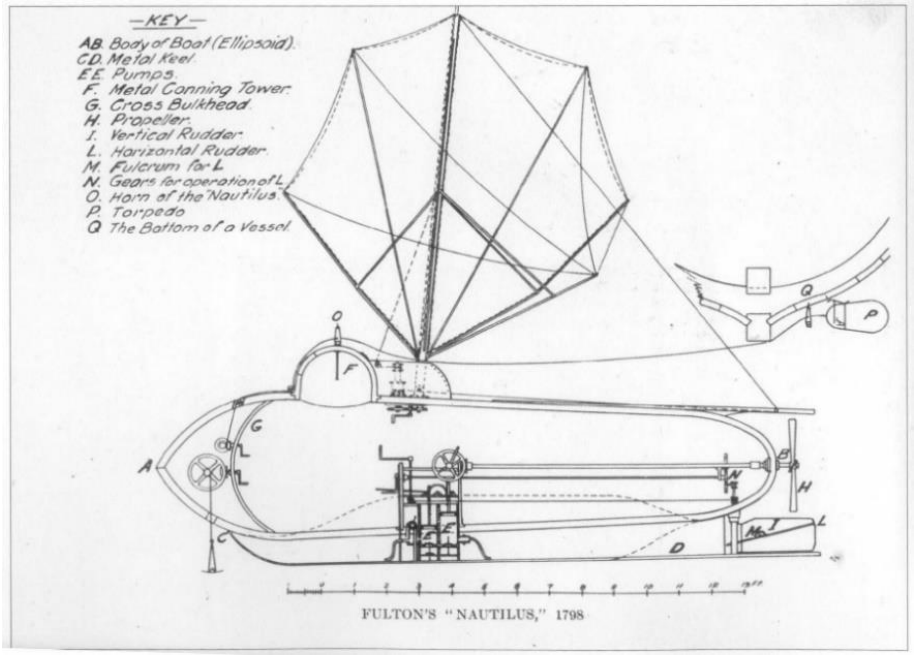


Figura 2.6 - Nautilus, el primer submarino de Fulton.[9]

Tras los sucesivos fracasos, Fulton decidió volver a investigar sobre el uso de la energía de vapor. En París conoció a Robert Livingston, Ministro de Asuntos Exteriores de Estados Unidos en Francia, que lo ayudó a conseguir los recursos necesarios para construir su barco. Botaron un modesto barco a vapor sobre el río Sena, por lo que demostró que la tecnología podía funcionar con ciertas modificaciones.

En 1803, regresó a Estados unidos para mejorar sus diseños de buques de vapor, y tras cuatro años de avances, Fulton lanzó con gran éxito el buque *Clermont*, que hizo el viaje por el río Hudson desde Nueva York hasta Albany a una velocidad de 5 millas por hora, reduciendo el tiempo habitual de navegación de 64 horas a 32 horas. Posteriormente se construirían 13 buques más, incluyendo el *Demologus*, primer barco de guerra a vapor.

[9] «File:FultonNautilus.jpg - Wikipedia, la enciclopedia libre».

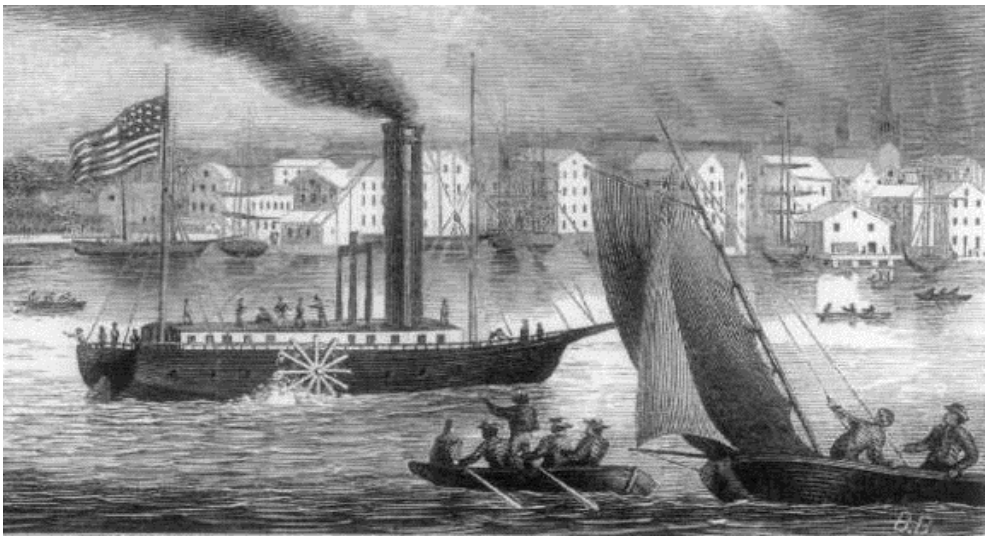


Figura 2.7 - El buque Clermont.[10]

Fulton murió de neumonía a sus 49 años, tras haberse casado con Harriet Livingston y tener cuatro hijos, tres de los cuales eran mujeres. En todas sus relaciones sociales fue un hombre altruista, cortés y amistoso. Su extraordinaria dedicación a la caridad, hospitalidad y promoción de la ciencia se vio reflejada en las grandes donaciones realizadas a favor de éstas.

Thomas Simpson.

Figura 2.8 - Thomas Simpson.[11]

Nació el 22 de agosto de 1710. Recibió muy poca educación, aunque acudió a un centro escolar durante un tiempo antes de comenzar a trabajar como tejedor, el mismo oficio que ejerció su padre. Su interés por las ciencias matemáticas comenzó al observar el eclipse solar que tuvo lugar en 1724. Aprendió matemáticas de forma autodidacta, una particularidad en aquella época debido al trabajo que desempeñaba.

[10] File:Andibrunt, *Clermont illustration - Robert Fulton from Project Gutenberg eText 15161.jpg*.
[11] «Thomas Simpson (1710-1761) ».

Se trasladó a Nuneaton, Warwickshire en 1725 donde ocupó un puesto como profesor de matemáticas hasta el año 1733, en el mismo lugar donde estudió. Fue una de las personas más ilustres de un grupo de conferenciantes que enseñaban en los cafés de Londres. Esto puede parecer extraño, pero en este período, las cafeterías se denominaban en ocasiones como las *Universidades Penny*, debido a la educación barata que ofrecían, donde cobraban la entrada a un centavo y luego los clientes podían escuchar las conferencias mientras tomaban café.

En 1743 fue elegido jefe de matemáticas de la Real Academia Militar de Woolwich (RMA), su nombramiento tuvo un gran impacto sobre los contenidos matemáticos que se investigaron, particularmente los problemas de ingeniería y los relacionados con las fortificaciones. También fue elegido miembro de la 'Royal Society' (Real Sociedad de Londres para el Avance de la Ciencia Natural) y de la Real Academia Sueca de Ciencias.

Simpson es recordado fundamentalmente por su trabajo sobre interpolación y métodos numéricos de integración. Trabajó en la teoría de la probabilidad donde se basó en trabajos anteriores de De Moivre; en la teoría de los errores, donde trató de demostrar que la media aritmética era mejor que una sola observación. Fueron sus obvias habilidades matemáticas las que llamaron la atención de otros matemáticos de la época.

Thomas Bouch.

Figura 2.9 - Thomas Bouch.[12]

Nació el 25 de febrero de 1822, comenzó a estudiar en Carlisle, Cumbria y posteriormente a la muerte de su padre, inició sus estudios de ingeniería en Liverpool. Allí, encontró un empleo como ingeniero en una empresa ferroviaria, cuyas redes se estaban progresando vertiginosamente.

Thomas Bouch fue principalmente conocido como el diseñador del desafortunado puente Tay Rail que se derrumbó con la pérdida de 75 vidas. Cuando el puente abrió sus puertas, era el más largo del mundo y redujo el tiempo de viaje entre Edimburgo y Dundee en una hora. La reina Victoria, presente en la inauguración, otorgó a Bouch un título de caballero en reconocimiento a sus logros.

[12] «Thomas Bouch: Biography on Undiscovered Scotland».

Figura 2.10 - Vigas caídas del puente Tay Rail.[13]

A la edad de 26 años, fue nombrado ingeniero de los Ferrocarriles de Edimburgo y del Norte, donde se hizo un nombre desarrollando la tecnología necesaria para los transbordadores ferroviarios de carga rodada. También ayudó a diseñar la estación de Waverley de Edimburgo.

[13] «File: Fallen girders, Tay Bridge.jpg».

3 Nociones Previas

En este capítulo, se procede a introducir los conceptos esenciales para definir las características de un buque, entre otros conceptos. Todas estas definiciones se obtienen del libro *"De proa a popa. Conceptos básicos."* de Luis Delgado y de los apuntes de clase de la asignatura *"Construcción Naval"*.

- **Proa (Pr):** Es la zona frontal del buque en sentido longitudinal, la forma más habitual es de cuña a fin de presentar menor resistencia al medio en el que se desplaza. Existen formas muy variadas como las representadas en la siguiente figura:

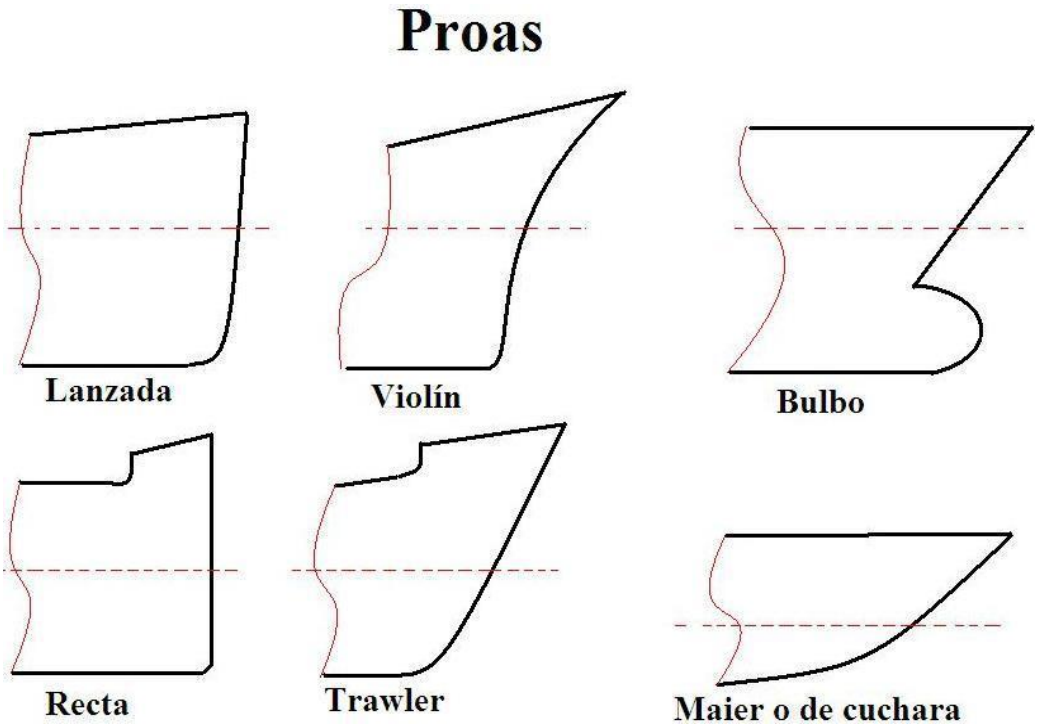


Figura 3.1 - Distintas formas de Proa.[14]

- **Popa (Pp):** Es la zona posterior del buque en sentido longitudinal. Las diferentes formas que se realizan para las popas tienen como objetivo principal evitar remolinos, y que el flujo expulsado por la hélice pueda incidir sobre el timón.
- **Estribor (Er):** Es la zona a la derecha del buque en sentido longitudinal.
- **Babor (Br):** Es la zona izquierda del buque en sentido longitudinal.
- **Plano de flotación:** Es el compuesto por la intersección de la superficie del agua con el volumen del buque.
- **Flotación:** La intersección del plano de flotación con la superficie fuera forros del buque, define la línea de flotación o simplemente flotación. La parte del barco situada por debajo del plano de flotación se llama *carena*.
- **Perpendicular de popa (P_{pp}):** Es la perpendicular a la flotación de proyecto que pasa por el centro del eje del timón o mecha del timón.

[14] «File:Formas de proas.jpg|Formas de proas».

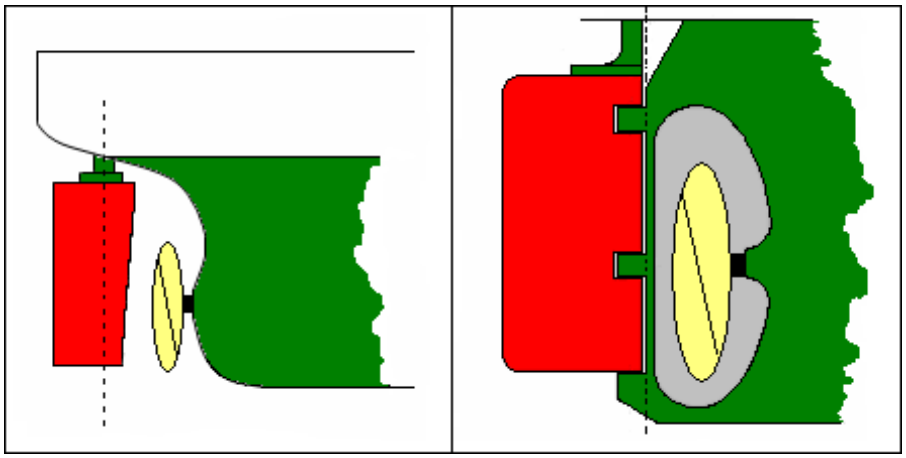


Figura 3.2 - Perpendicular de Popa.[15]

- **Perpendicular de proa (P_{pr}):** Es la perpendicular a la flotación de proyecto que pasa por la cara exterior o interior de la roda, según sea su estructura de acero fundido o armada respectivamente.

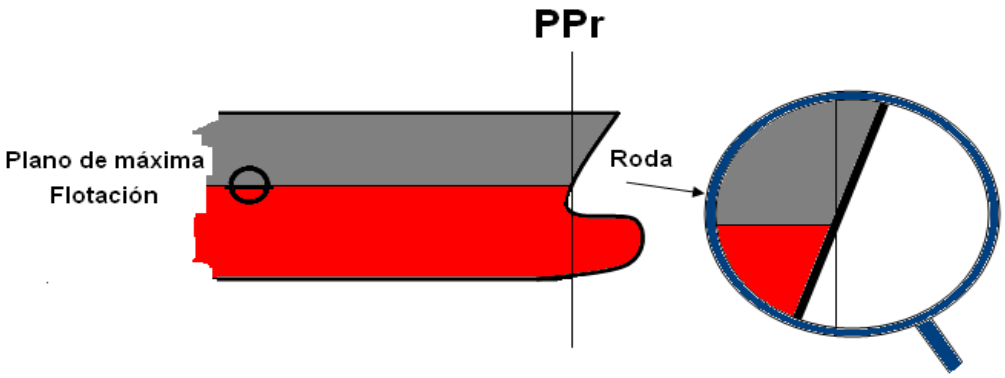


Figura 3.3 - Perpendicular de Proa.[16]

- **Línea de base:** Es la paralela a la flotación de proyecto trazada por la cara interna de la plancha de quilla.
- **Plano diametral:** Es el compuesto por la intersección de planos longitudinales con el volumen del buque.

3.1 Dimensiones principales del buque

- **Eslora total o máxima (L_T):** Es la longitud medida horizontalmente y paralela a la flotación de proyecto, entre las perpendiculares a la misma que pasan por los puntos más sobresalientes de proa y de popa.
- **Eslora entre perpendiculares (L_{pp}):** Es la longitud medida horizontalmente y paralela a la flotación máxima entre las perpendiculares de proa y de popa.
- **Eslora en la flotación (L_F):** Es la longitud a fuera forros del buque tomada en el plano de la flotación para cada situación del buque.

[15] « File:Perpendiculardepopa.PNG ».
[16] « File:Perpendiculardeproa.PNG ».

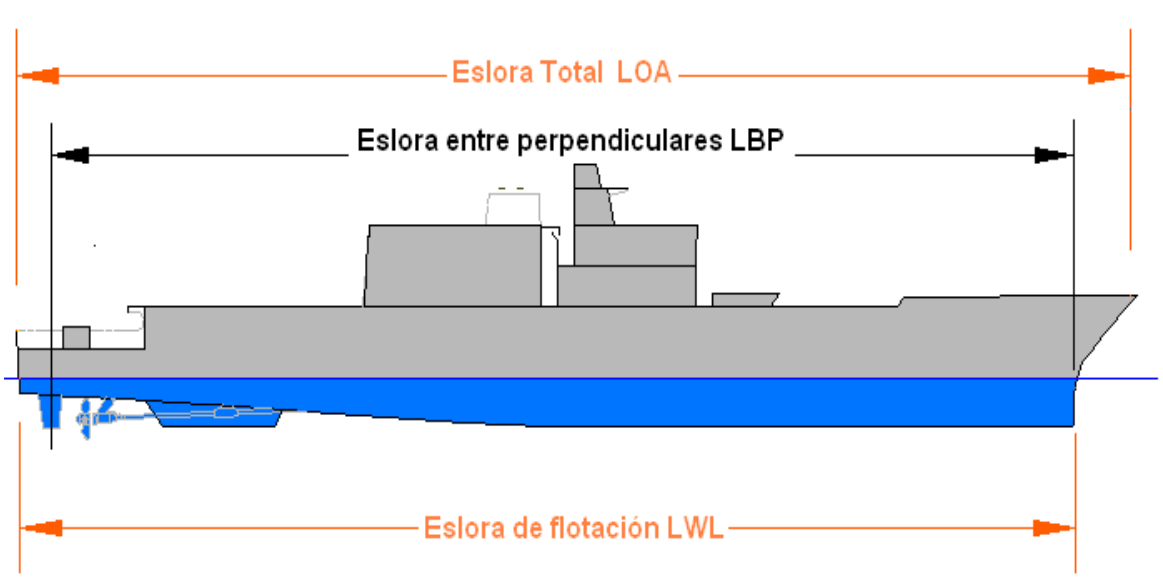


Figura 3.4 – Diferentes esloras.[17]

- **Manga de trazado (B):** Es la mayor distancia en sentido transversal del buque medida horizontalmente entre las perpendiculares a la flotación de proyecto y sin considerar las caras del forro exterior, también suele llamarse *manga de forros*. Está referida a la *Sección o Cuaderna Maestra*, que es la sección transversal de mayor área, y es la que se utiliza en proyecto y la que viene en los planos.
- **Manga máxima:** Se trata de la misma distancia que la manga de trazado, pero tomada por la parte exterior del forro o casco. También se denomina *manga fuera de forros*.
- **Manga de flotación:** Es la anchura del buque medida en el plano de flotación de la cuaderna maestra. Es variable, ya que cambia el plano de flotación con el calado. Cuando un buque tiene costados rectos, la manga de flotación y la de trazado coinciden.
- **Puntal de trazado (D):** Es la altura fuera forros medida en la línea central del buque en la sección transversal correspondiente a la Cuaderna Maestra.
- **Puntal de construcción o de francobordo:** Se mide en la sección maestra y es la altura fuera forros en el plano de crujía desde la quilla plana a la cubierta principal.
- **Calado máximo (T_{max}):** Es la altura o profundidad de la carena, medida desde el punto más bajo del buque, hasta los planos de las flotaciones. El calado varía a lo largo de la eslora del buque, motivo por el cual se mide y se marca en lugares perfectamente determinados a popa, a proa y centro del buque.
- **Asiento o trimado (t):** Es la diferencia entre el calado de proa y el de popa. En función de si el trimado se produce por proa o por popa, el valor será positivo o negativo.

$$t = T_{pp} - T_{pr} \ \ (m) \qquad (3.1)$$

[17] « File:Esloras.PNG ».

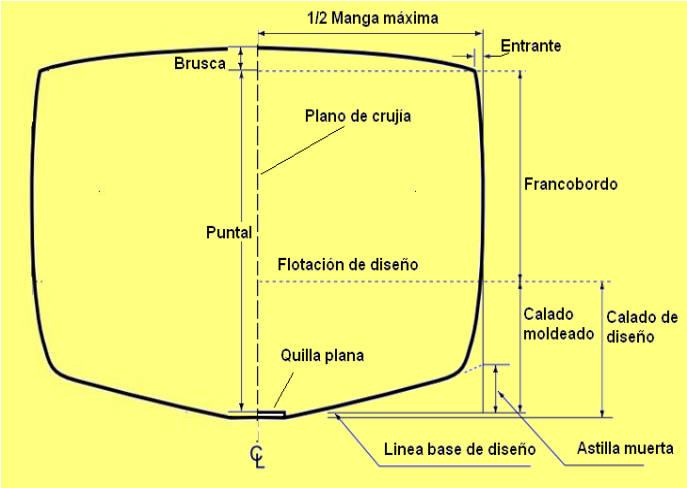


Figura 3.5 - Dimensiones transversales.[18]

- **Francobordo (f):** Es la distancia medida en el costado del buque, entre la flotación y la cubierta más alta con medios permanentes de cierre (cubierta de francobordo). De su valor depende la seguridad del buque. En todo buque existe una línea de máxima carga a partir de la cual no está permitido cargar. La distancia entre dicha línea y la cubierta de francobordo, es el francobordo mínimo. Las líneas empleadas son las siguientes:
    - TD: Línea de carga tropical en agua dulce.
    - D: Línea de carga para agua dulce en verano.
    - T: Línea de carga tropical.
    - V: Línea de carga para verano. Esta línea está a la misma altura que el centro del disco.
    - I: Línea de carga para invierno.
    - ANI: Línea de carga para invierno en el Atlántico Norte.

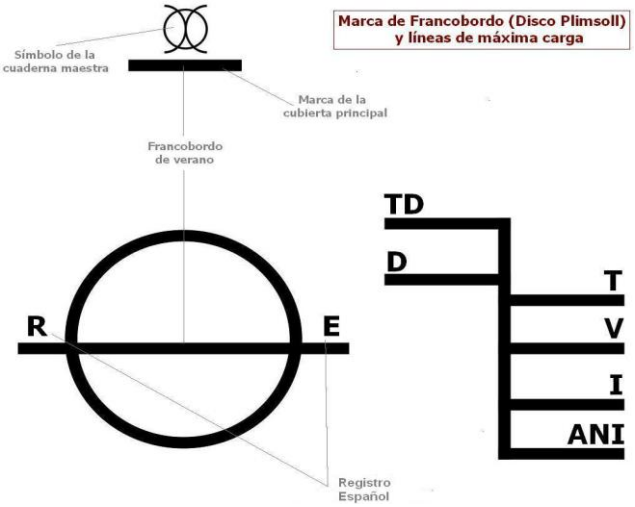


Figura 3.6 - Línea de francobordo.

[18] « File:Dimensiones.PNG ».

3.2 Otras definiciones de características del buque

- **Astilla muerta:** Es la elevación sobre la horizontal de las planchas del fondo. Se llama *ángulo de astilla muerta* al ángulo que forman las planchas del fondo a partir de la quilla con la horizontal (α).
- **Pantoque:** Es la superficie curva que une los costados y el fondo del buque. Su sección, generalmente, es un arco de circunferencia.
- **Aleta:** Es la zona del costado donde la manga disminuye para cerrar y formar la popa del buque. Nos encontramos con la aleta de estribor, y la de babor.
- **Amura:** Es la zona del costado donde el casco forma la proa del buque. Nos encontramos con la amura de estribor, y la de babor.
- **Través:** Son los costados del buque en la zona intermedia de la eslora. Se suele emplear este término para determinar la dirección de una ola transversal.

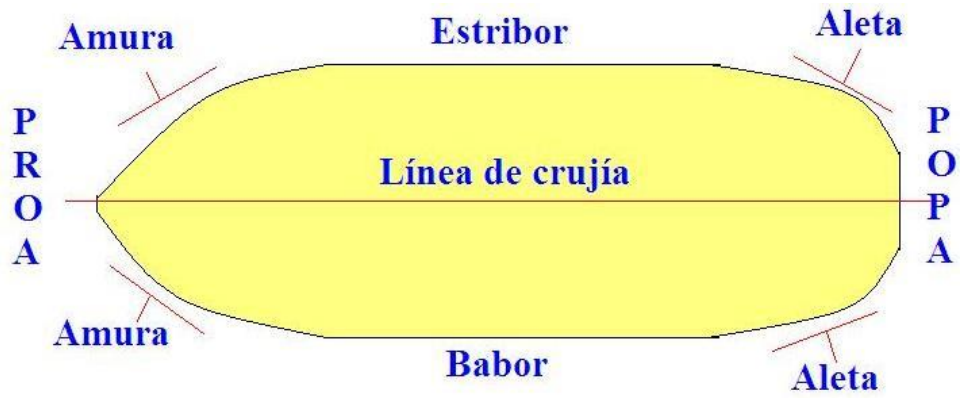


Figura 3.7 - Nomenclatura básica del buque.

- **Codaste:** Es la zona más a popa del casco, donde se unen los costados por debajo de la flotación. Puede tener distintas configuraciones dependiendo del tipo de timón que se elija para el buque. En la figura 13 podemos ver el codaste para un timón suspendido.
- **Roda:** Es la zona más a proa del casco, donde se unen los costados. Puede tener distintos perfiles, la forma más usual es la proa lanzada. Se puede apreciar esta forma en la figura 14.

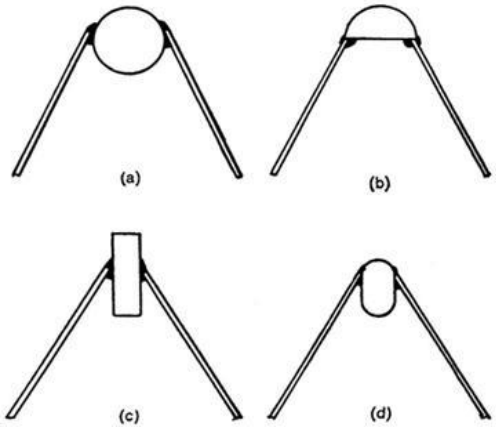


Figura 3.8 - Tipos de rodas.[19]

[19] «Calculo Estructural del Buque: Capítulo 4. Quilla, Roda y Codaste.».

- **Quilla:** Es una pieza longitudinal formada por un material robusto, usualmente se utiliza la madera o el acero. Recorre la parte inferior central del barco, y en sus extremos se une a la roda y al codaste. Forma la base de la estructura del barco proporcionando rigidez y resistencia. Existen diversas formas de esta pieza como quilla vertical, quilla horizontal, o quilla de barra como se muestra en la siguiente figura:

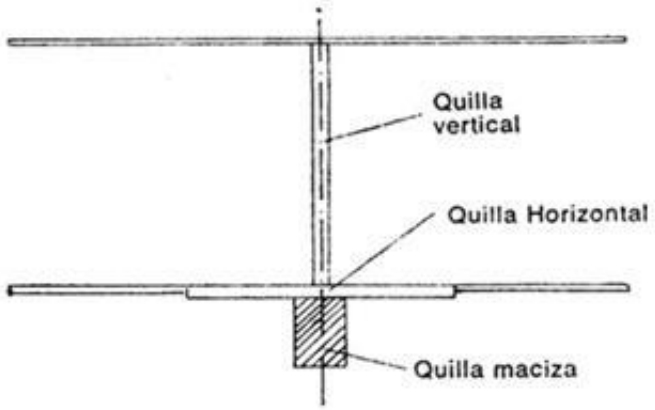


Figura 3.9 - Formas de la quilla.[20]

- **Superestructuras:** Se refiere a aquellas estructuras por encima de la cubierta principal, pueden ocupar o no la totalidad de la manga dependiendo de la posición donde se encuentren. El lugar más alto habitable suele ser donde se instalen los equipos necesarios para la navegación y control, este local recibe el nombre de Puente. Podemos distinguir entre:
    - Castillo, si va en proa.
    - Ciudadela, si va en el centro del buque.
    - Toldilla, si va a popa.
- **Bulbo de proa:** Consiste en una distribución protuberante del volumen de carena situado en la proa y bajo la línea de flotación. Su objetivo principal es generar una perturbación en el fluido que produzca un sistema de olas adicional de forma que interfiera con el producido por la carena para que el sistema resultante sea de menor amplitud. Así se consigue reducir la resistencia por formación de olas del buque. Podemos encontrar dos tipos de bulbo (bulbo explícito y bulbo implícito) y tres tipos si distinguimos según las distintas secciones que existen (tipo gota de agua, tipo elíptico y tipo peonza).

3.3 Características hidrostáticas del buque

- **Desplazamiento en rosca (Δ_{Rosca}):** Es el peso mínimo del buque capaz de empezar a navegar. Comprende el peso del acero del casco, el peso de la maquinaria instalada, el peso del equipo y habilitación, el peso de la dotación, así como del peso de los fluidos en circuitos de máquinas y calderas.
- **Desplazamiento total (Δ):** Es el peso total del buque, cargado de tal forma que su calado sea el correspondiente al francobordo mínimo de verano. Se llama también *desplazamiento a máxima carga*.

[20] « Calculo Estructural del Buque: Capítulo 4. Quilla, Roda y Codaste.».

- **Arqueo total:** También recibe el nombre de tonelaje de registro bruto (GT), y corresponde a la capacidad del buque desde la línea base hasta la cubierta estanca más alta, excluyendo los tanques de lastre.
- **Arqueo neto:** Otra forma de definirlo sería el tonelaje de registro neto, que se obtiene a partir del arqueo total previamente definido, en este caso, sin tener en cuenta los espacios necesarios para la operatividad del buque, como los pañoles, calderas, cámara de máquinas o alojamientos de la dotación.

- **Peso muerto (PM):** Es la diferencia entre el desplazamiento del buque en agua salada correspondiente al calado con francobordo de verano y el peso del buque vacío o en rosca. Está constituido por:
    - Peso de la carga que puede transportar el buque.
    - Combustible.
    - Agua de reserva para alimentación y aceite de reserva.
    - Víveres.
    - Peso del pasaje con sus equipajes.
- **Coeficiente de bloque (C_B):** Es la relación del volumen desplazado por el casco y el volumen del paralelepípedo circunscrito al mismo, de lados: la eslora entre perpendiculares, la manga y el calado medio del buque.

$$C_B = \frac{\nabla}{L_{pp} \cdot B \cdot T_m} \qquad (3.2)$$

- **Coeficiente de la Maestra (C_M):** Es la relación del área sumergida de la cuaderna maestra (A_M) normalmente referida en carga máxima con la del rectángulo circunscrito a la misma.

$$C_M = \frac{A_M}{B \cdot T_m} \qquad (3.3)$$

- **Coeficiente de la Flotación (C_F):** Es la relación entre el área de la flotación (A_F) y la superficie del rectángulo circunscrito, la manga y la eslora en la flotación.

$$C_F = \frac{A_F}{B \cdot L_F} \qquad (3.4)$$

- **Criterios de estabilidad:** Se trata de un conjunto de reglas que debe cumplir un buque para que su estabilidad alcance valores mínimos que garanticen su seguridad. Suelen venir definidos por distintas normativas o sociedades de clasificación.

- **Escora:** Se refiere a la inclinación transversal que toma un buque cuando éste se aparta de la vertical. Las causas pueden ser diversas, debido a golpes de mar, corrimiento de la carga, varada u otros motivos. También existe la escora permanente, debida a la configuración de pesos del buque, ya que su centro de gravedad no se encuentra en la vertical y el buque debe encontrar el equilibrio. Existe un ángulo a partir del cual el buque puede comenzar a inundarse, ángulo de inundación, si éste se supera es posible que el buque no sea capaz de recuperarse y comience a hundirse.

- **Autonomía:** Se refiere a la distancia, en unidades del sistema anglosajón (millas), que el buque es capaz de navegar a una velocidad determinada, con el máximo consumo de combustible.

3.4 Definiciones sobre resistencia de materiales

- **Carga:** De forma general, es el peso que actúa sobre un cuerpo, sometiendo a la estructura a una situación que generalmente provoca deformaciones sobre ésta. Se suele utilizar para definir la cantidad de mercancía que transporta.
- **Esfuerzo:** Es el resultado directo de la carga sobre el cuerpo. Dependiendo de la magnitud de ese esfuerzo, el material puede sufrir deformaciones o incluso llegar a la fractura, donde la carga a la que se somete es más elevada que la carga de rotura del material, para evitar esto se deben seleccionar los materiales adecuados.
    - Esfuerzo de tracción. Son aquellas fuerzas o cargas que lo estiren, esto es, con dirección perpendicular al cuerpo y hacia fuera.
    - Esfuerzo de compresión. Son aquellas fuerzas o cargas que lo comprimen, esto es, con dirección perpendicular al cuerpo y hacia dentro.
    - Esfuerzo de flexión. Son aquellas fuerzas o cargas que lo doblen.
    - Esfuerzo de cortante: Son aquellas fuerzas actuando en sentido paralelo y direcciones opuestas.
    - Esfuerzo de torsión. Se producen al aplicar un momento sobre el eje longitudinal del cuerpo.

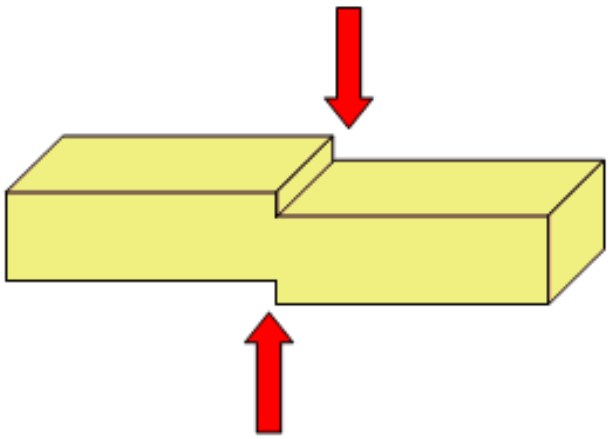


Figura 3.10 – Representación de los esfuerzos cortantes.[21]

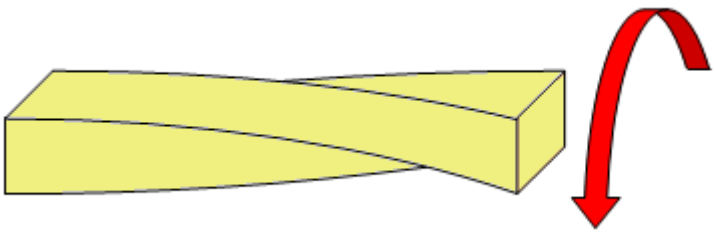


Figura 3.11 - Representación del esfuerzo de torsión.[22]

[21] «File:Esfuerzo-cortante.png|Esfuerzo-cortante»
[22] «File:Esfuerzo-torsion.png|Esfuerzo-torsion»

- **Deformación:** Se produce como consecuencia del esfuerzo producido, y es la medida de la alteración de las formas. En función de la zona en la que nos encontremos, podemos definir:
    - Deformación elástica: El material es sometido a una fuerza y cuando ésta deja de actuar sobre él, es capaz de recuperar su forma original.
    - Deformación plástica: En este caso, el material se deforma permanentemente, esto quiere decir que, si es sometido de nuevo a la misma fuerza o superior, es probable que la pieza termine rompiéndose.
- **Módulo de elasticidad o de Young:** Es el valor del esfuerzo dividido por la deformación, dentro del límite elástico del material. Es una constante dependiente del tipo de material.

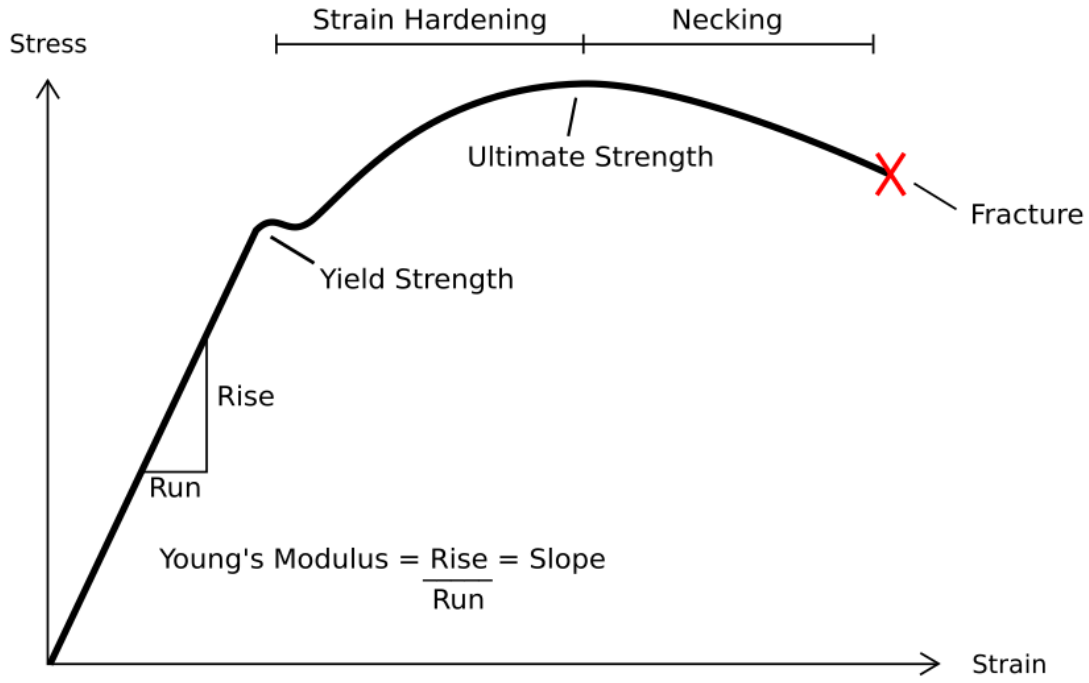


Figura 3.12 – Diagrama tensión-deformación.[23]

3.5 Características de las olas

- **Cresta y seno:** Definimos la cresta de ola como su parte más alta, mientras que el seno corresponde a la parte más baja, este último puede recibir diversos nombres, como vano o valle.
- **Longitud (λ):** Es la distancia horizontal que existe entre dos crestas o dos senos consecutivos.
- **Altura (H):** Es la distancia vertical medida entre la cresta y el seno de la ola.
- **Amplitud (A):** Distancia vertical entre una cresta o un seno, y el nivel de la superficie libre. La altura de ola se puede estimar como dos veces la amplitud si la amplitud es constante en el tiempo, estableciéndose una relación entre estas dos características.

[23] «File:Stress Strain Ductile Material.png»

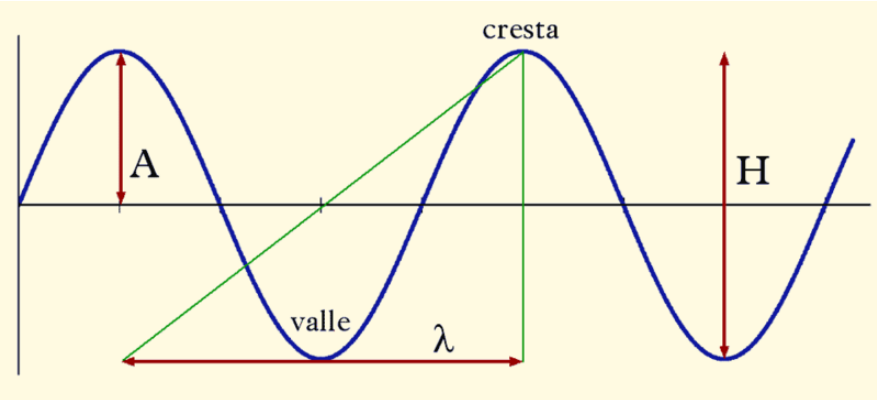


Figura 3.13 – Características de la ola.[24]

- **Frecuencia (ω):** Es la inversa del periodo de ola, esto quiere decir que nos proporciona el número de veces que se repite una cresta o un seno en un intervalo de tiempo.
- **Generación de olas por el viento:** El viento puede suscitar una perturbación sobre la superficie libre del mar, dando origen a olas. Esta perturbación dependerá de la intensidad y duración del viento, así como de la extensión de la superficie afectada, denominada *fetch*. Una vez formadas las olas, se desplazarán fuera de la zona donde se generaron, hasta que se agote su energía. Cada uno de los fenómenos, viento y mar, se mide con las siguientes escalas:

ESCALA DE BEAUFORT			**ESCALA DE DOUGLAS**		
Grado	**Nombre**	**Velocidad viento [m/s]**	**Grado**	**Nombre**	**Altura significativa de ola, $H_{1/3}$ [m]**
0	Calma	0.0 – 0.2	0	Calma	0
1	Ventolina	0.3 – 1.5	1	Llana	0.1 – 0.2
2	Flojito	1.6 – 3.3	2	Rizada	0.3 – 0.5
3	Flojo	3.4 – 5.4			0.6 – 1.0
4	Bonancible	5.5 – 7.9	3	Marejadilla	1.5
5	Fresquito	8.0 – 10.7	4	Marejada	2.0
6	Fresco	10.8 – 13.8	5	Gruesa	3.5
7	Frescachón	13.9 – 17.1	6	Muy Gruesa	5.0
8	Duro	17.2 – 20.7			7.5
9	Muy Duro	20.8 – 24.4	7	Arbolada	9.5
10	Temporal	24.5 – 28.4	8	Montañosa	12.0
11	Borrasca	28.5 – 32.7	9	Confusa	15.0
12	Huracán	> 32.7			> 15

Tabla 3.1 – Escalas de Beaufort y Douglas.[25]

[24] «File:Olas parametros.PNG».
[25] «File:escalas_de_viento_y_oleaje.pdf».

- **Esfuerzos sobre el casco:** Se debe tener en cuenta que existen diferentes tipos de fuerzas que actúan. Para el estudio del barco viga solo se plantean los esfuerzos longitudinales y de torsión, debido a su grado de importancia. Pueden ser: fuerzas estáticas, y fuerzas dinámicas. En este proyecto sólo se tendrán en cuenta las fuerzas estáticas, aquellas para la condición del buque parado en aguas tranquilas, a pesar de que se plantean las fuerzas dinámicas y se estudian durante todo el trabajo.

3.6 Componentes de la resistencia al avance

- **Resistencia viscosa:** Es la componente asociada a la viscosidad del fluido en el que se encuentra el buque, puede dividirse en otras dos componentes:
  - Resistencia de fricción: Se puede calcular asumiendo la semejanza del buque a una placa plana con la misma longitud y superficie mojada.
  - Resistencia de forma: A su vez podemos distinguir entre la debida a la curvatura del casco, transversal y longitudinal, y a la debida por la presión de origen viscoso, producido principalmente por la capa límite.
- **Resistencia por formación de olas:** Debida a la energía empleada para generar un sistema de olas en la superficie del agua provocado por el avance del buque.

Posteriormente, se detallará el método empleado en mayor profundidad para la estimación de la resistencia al avance del buque para obtener la potencia a instalar del motor.

4 Introducción a MATLAB

MATLAB es la abreviatura de Matrix Laboratory, fue creado en 1984 y es un software de cálculo con un lenguaje de programación propio y un entorno de desarrollo integrado (IDE) interactivo utilizado por muchos ingenieros y científicos de todo el mundo. Con este software se pueden realizar multitud de tareas dado que proporciona al usuario una herramienta de cálculo potente y agradable de utilizar mediante el uso de sencillas instrucciones en forma de comandos, que permite la construcción de auténticos programas.

La ventana principal que aparece al ejecutar MATLAB en un ordenador por primera vez será la siguiente:

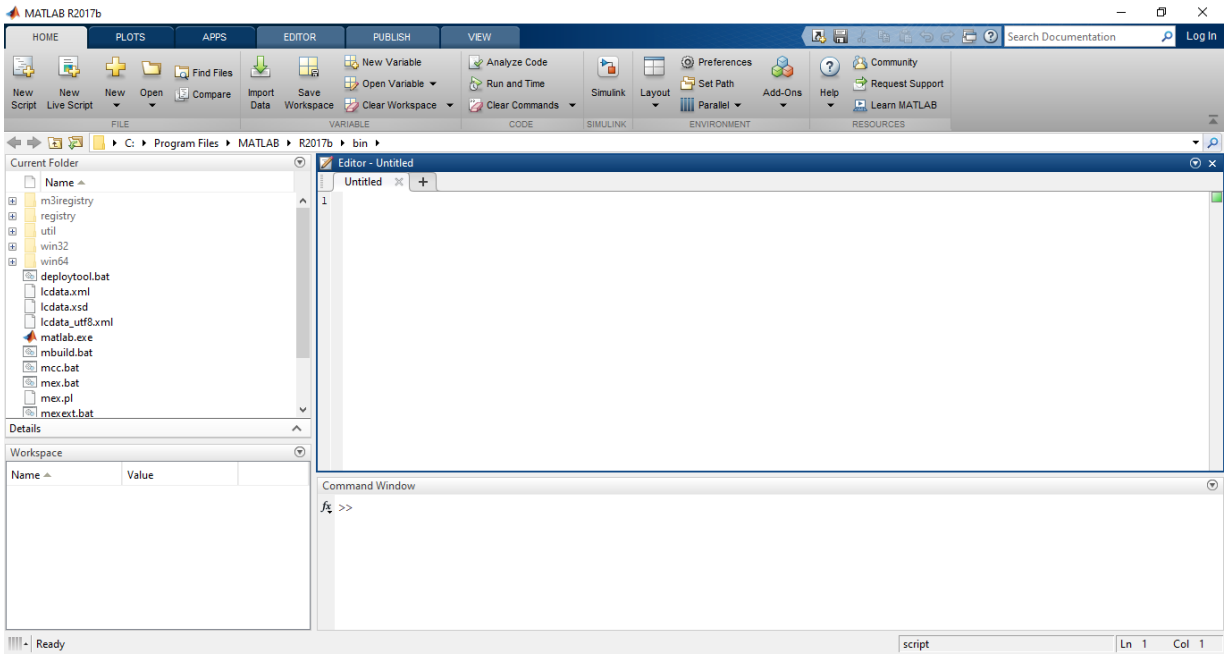


Figura 4.1 - Entorno principal de trabajo de MATLAB.

En esta imagen podemos visualizar los elementos básicos del escritorio de trabajo, seguidamente desarrollaremos una descripción más avanzada a continuación.

- **Command Window:** Se trata de la ventana desde la que se ejecutan las instrucciones y programas. Se trabaja de forma directa con el programa, es decir, insertamos la instrucción o el nombre del programa o función que deseemos, y al presionar el botón de Enter en el teclado, se ejecutará. Es posible interrumpir la ejecución de un programa con 'Control + C', en los casos que el programa entre en bucle, también podemos limpiar la ventana con el comando 'clc'.
- **Command History:** Aparece cuando nos situamos en la ventana anterior, y pulsamos a la flecha que señala hacia arriba en el teclado, entonces nos aparecen todos los comandos ejecutados recientemente, aunque es posible obtener una ventana extra dentro del escritorio anterior donde podamos ver los comandos de forma directa. Por comodidad a la hora de trabajar en este proyecto, se realiza de la primera forma.
- **Current Directory:** Localiza la carpeta de trabajo en la que nos encontramos, y es posible modificarla.

- **Workspace:** Nos da información sobre las variables que se han usado, generalmente su valor y su dimensión, en el caso de vectores y matrices. En el caso de querer eliminar las variables de trabajo, podemos ejecutar el comando 'clear' con la variable que queramos, o directamente eliminar todas con el comando 'clear all'.
- **Editor:** Desde esta ventana, podemos trabajar para crear distintos programas o funciones que posteriormente necesitemos utilizar, podemos realizar comentarios para explicar el trabajo que se realiza mediante la inserción del símbolo '%' delante de lo que queramos comentar. Todos los ficheros de comandos tienen como extensión .m y deben tener el mismo nombre que la función, para que posteriormente MATLAB pueda localizarlo de forma correcta.
- **Help:** Puede utilizarse de forma directa desde la barra de herramientas, o ejecutando el comando 'help' de forma directa en el Command Window. Es posible obtener información de una función ya integrada en el programa MATLAB si ejecutamos '*help función'*, donde conseguiremos una breve descripción de la función y posibles ejemplos. A continuación, se muestra un ejemplo sobre su uso en el caso de que queramos conocer más sobre la función 'length':

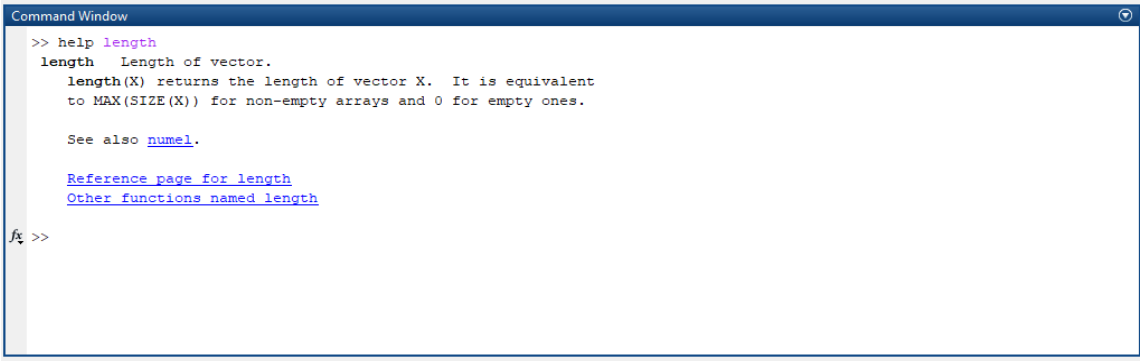


Figura 4.2 - Muestra del uso de la función 'help' en MATLAB.

También podemos crear programas con interfaces gráficas de usuario, conocidas como GUI, que proporcionan un control sencillo de las aplicaciones sin tener que preocuparse por los comandos necesarios para ejecutar un programa. Las aplicaciones de MATLAB son programas independientes con un frontal gráfico de usuario GUI que automatizan una tarea o un cálculo. Uno de los propósitos de este trabajo es crear una de estas aplicaciones, con una interfaz de usuario personalizada, para que otras personas la utilicen.

Para esto, se escribe el comando 'guide' o se pulsa sobre él en la barra de herramientas. Una vez ejecutado, aparece una ventana donde podemos seleccionar configuraciones predeterminadas de MATLAB, o crear la nuestra propia a partir de una interfaz en blanco.

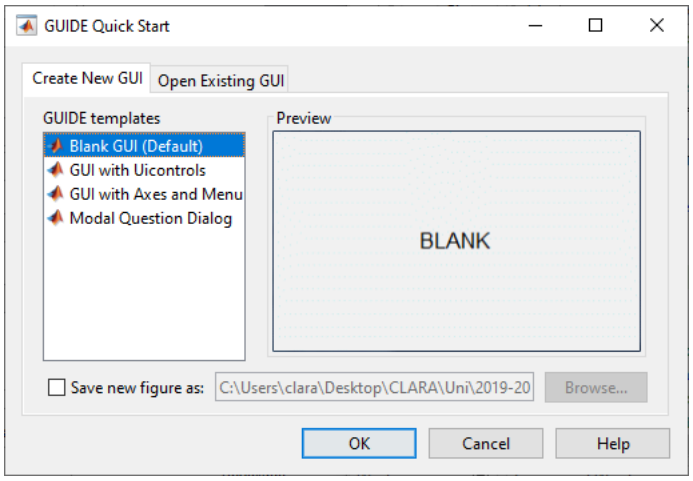


Figura 4.3 – Selección para la creación de una interfaz en MATLAB.

Una vez que tenemos la interfaz en blanco, podemos insertar distintos tipos de elementos, como botones, representaciones gráficas, figuras, texto, etc. MATLAB generará de forma automática el código correspondiente a los elementos que insertemos, que podemos modificar para que realicen distintas acciones. Este archivo generado se guardará como un fichero .fig y con otro fichero con el mismo nombre que corresponde al programa que ejecuta la interfaz con un fichero .m como se ha comentado anteriormente.

5 Estudio del barco-viga

El objetivo de este trabajo se centra en el estudio de la resistencia longitudinal de un buque, para ello, la principal hipótesis que se debe plantear es que el casco puede ser proyectado como una viga, con la singularidad de que el buque es soportado por el agua en toda su eslora.

Figura 5.1 - Imagen de un buque portacontenedores en aguas tranquilas.[26]

Podemos suponer distintos casos en la superficie de apoyo de la viga. Con mar en calma obtendríamos una recta que puede cambiar de pendiente en función de la condición de carga del buque, aunque ésta no se asemeja con la realidad, dado que lo habitual es que el buque navegue sobre olas. Estas olas suelen estar definidas de dos formas, como ola trocoidal, la más habitual pero más compleja de trabajar, o como ola sinusoidal, más habitual para realizar los cálculos que se suelen simplificar al considerar este tipo de ola.

En este trabajo se plantea la posibilidad de realizar el estudio del buque sobre olas de tipo sinusoidal, pero principalmente nos centraremos sobre la condición de aguas tranquilas, donde el buque tendrá un empuje lineal.

Para realizar el estudio de las fuerzas que actúan sobre el buque, se divide el buque en distintas secciones para poder trabajar de una forma más sencilla. Así, es posible distinguir entre el peso de la sección, y el empuje debido a su forma sumergida. Por el Principio de Arquímedes, el peso será igual al empuje para flotar en equilibrio. Cada sección no podrá establecer su equilibrio consigo mismo y quedará sometido a un esfuerzo por carga en sentido vertical, que será la diferencia entre el peso y el empuje en dicho módulo, por metro de estructura. Estos esfuerzos producen tensiones internas que se pueden calcular.

[26] «File:Espacio de carga libre».

Figura 5.2 - Imagen de un buque sometido a esfuerzos de arrufo.[27]

La teoría de la flexión de la viga se basa principalmente en un sistema de fuerzas aplicado sobre ésta, donde las secciones giran de forma que la cara convexa se encuentra sometida a esfuerzos de tracción y la cara cóncava a esfuerzos de compresión, aquella sección que no sufre ningún tipo de deformación se le denomina línea neutra.

Una vez desarrollada la resistencia transversal de forma general, se realiza el estudio de los esfuerzos longitudinales sobre el buque en aguas tranquilas, ya que es la condición esencial del buque.

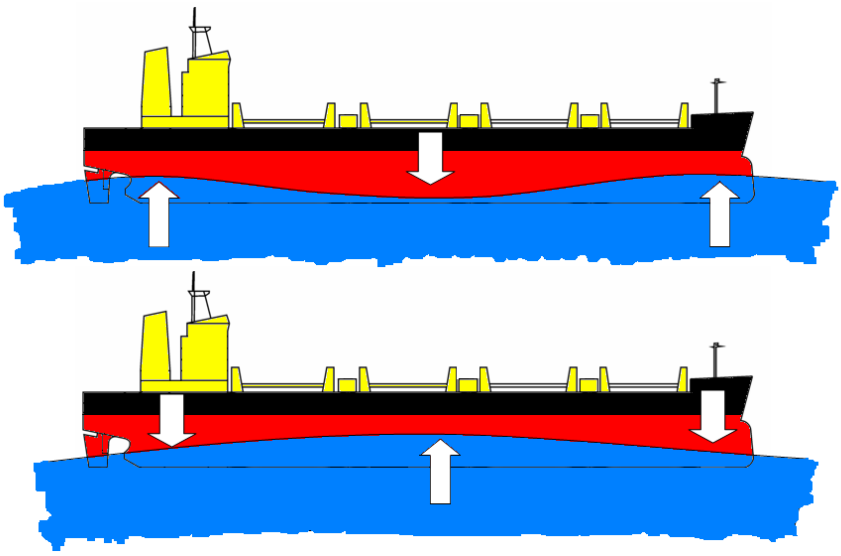


Figura 5.3 - Situaciones de equilibrio del buque.[28]

Como se ha comentado previamente, el buque se encuentra en equilibrio en toda su eslora, pero al dividir en secciones, podemos obtener diferencias de pesos y empujes. Si las rebanadas fuesen independientes, cada una encontraría su propio equilibrio variando su calado. La carga en cada rebanada se obtiene a partir de la diferencia de pesos y empujes, obteniendo la curva de cargas resultante en toda la eslora.

[27] «File:Colapso del MOL Comfort».
[28] «File:*Esfuerzo de arrufo y quebranto*».

Esta distribución de cargas implica que se produzcan esfuerzos cortantes y momentos flectores entre las rebanadas. Como consecuencia se produce la deformación de la viga-casco. Las relaciones entre las fuerzas anteriores, esfuerzos cortantes y momento flector son:

- Carga de una rebanada:

$$c = p - b \qquad (5.1)$$

Donde c es la carga, p es el peso y b es el empuje de la rebanada transversal. Para el buque, si consideramos el peso p como el desplazamiento del buque, y b como el empuje total del buque, la carga c es nula $(c = 0)$.

- Relación entre la carga y el esfuerzo cortante:

$$c = \frac{d(EC)}{dx} \qquad (5.2)$$

- Relación entre el esfuerzo cortante y el momento flector:

$$EC = dMFdx \qquad (5.3)$$

- Relación entre la carga, el esfuerzo cortante y el momento flector:

$$c = \frac{d(EC)}{dx} = \frac{d^2(MF)}{dx^2} \qquad (5.4)$$

A partir de estas expresiones, se deduce que para la obtención de los esfuerzos cortantes (EC) y momentos flectores (MF) únicamente es necesario integrar la expresión anterior, de la siguiente forma:

$$EC = \int_0^L c \cdot dx \qquad (5.5)$$

$$MF = \int_0^L EC \cdot dx = \int_0^L \int_0^L c \cdot dx \cdot dx \qquad (5.6)$$

En el capítulo siguiente, se explicarán los distintos métodos de cuadratura que se pueden aplicar a las integrales obtenidas.

Para el caso del buque entre olas, se debe tener en cuenta que la distribución del empuje varía a lo largo de la eslora, esto supone que se produzcan modificaciones en los esfuerzos cortantes y momentos flectores obtenidos para el buque en aguas tranquilas. Las dos peores condiciones se dan para el buque en arrufo o quebranto en su zona central, como se ha visto anteriormente.

5.1 Curva de pesos

Esta curva representa el conjunto de pesos del buque, teniendo en cuenta el peso del buque en rosca, el peso de los tanques de servicio, la carga en bodegas y cubierta, entre otros. Para obtener esta curva, el buque se divide por secciones de forma longitudinal, situadas a intervalos constantes para simplificar la distribución, aunque para definir mejor las formas de proa y popa, el espaciado puede ser menor para obtener una mejor aproximación.

Dividiendo el peso total de cada sección por su eslora, se obtiene el peso medio por unidad de eslora, generalmente en toneladas por metro. La forma habitual de esta

distribución es rectangular, aunque se pueden estudiar otros casos como la distribución triangular o la distribución trapezoidal.

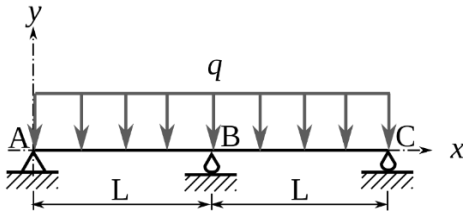


Figura 5.4 - Distribución uniforme del peso.[29]

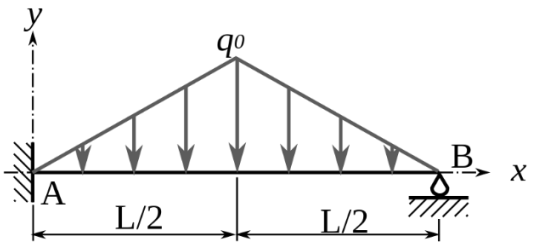


Figura 5.5 - Distribución triangular cerrada del peso.[30]

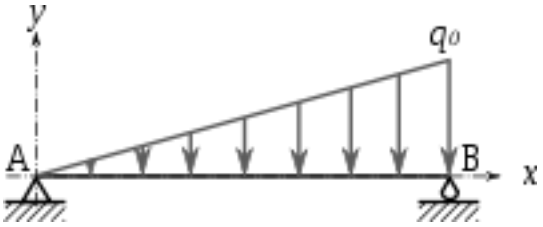


Figura 5.6 - Distribución triangular del peso.[31]

A lo largo de la eslora, podemos obtener una forma escalonada de la curva de pesos final.

5.2 Curva de empujes

Como se ha visto anteriormente, para que el buque se encuentre en equilibrio, el desplazamiento es igual al empuje, y la posición del centro de gravedad del buque y el centro de carena estarán alineados en la misma línea vertical, si esta posición se encuentra desplazada respecto a la sección media, el buque encontrará la posición de equilibrio trimando por proa o popa según sea esa distancia positiva o negativa respectivamente. Esta curva dependerá principalmente de la condición del agua de mar. El número de secciones empleados coincidirá con las secciones empleadas para la curva de pesos, obteniendo los empujes correspondientes para cada sección.

5.3 Curva de cargas

La carga en un punto cualquiera de la eslora del buque es igual a:

$$q(carga) = p(peso) - e(empuje) \qquad (5.7)$$

Por lo tanto, para obtener esta curva a lo largo de toda la eslora, es necesario realizar este cálculo para cada sección del buque a estudio.

[29] « File:Poutre appuis trois appuis charge uniforme deux travees stat.svg|Poutre appuis trois appuis charge uniforme deux travees stat».

[30] « File:Poutre appuis console appuyee charge triangle stat.svg».

[31] « File:Poutre appuis charge lineaire stat.svg|Poutre appuis charge lineaire stat».

6 Resistencia al avance

Para el estudio de la resistencia al avance se emplea el método de Holtrop, que subdivide la resistencia total en diversas componentes de diferente origen. Es un método estadístico desarrollado a partir de regresiones matemáticas de los resultados conseguidos en ensayos en el canal de Wageningen y de resultados de pruebas de mar de buques construidos.

Figura 6.1 – Buque Ro-Ro navegando en aguas tranquilas.[32]

Es un método que proporciona estimaciones de la resistencia bastante satisfactorias. La muestra de buques para este método abarca desde petroleros, pasando por portacontenedores y pesqueros, hasta remolcadores y fragatas, entre otros. Es un método válido tanto para buques de una como de dos líneas de ejes.

El método tiene un rango de aplicación distinto para cada tipo de buque. Como en este caso el buque a estudio se trata de un buque Ro-Ro, se definen los siguientes límites que asegurarán una buena aproximación de la potencia estimada:

$Fn_{máx}$	**C_P**		**L/B**		**B/T**	
	Min.	Max.	Min.	Max.	Min.	Max.
0.35	0.55	0.67	5.3	8.0	3.2	4.0

Tabla 6.1 - Rango de aplicación del método de Holtrop para buques Ro-Ro.

Para este método, se emplea la formulación de Hughes, utilizando la línea básica de fricción de la ITTC-57. Por tanto, la resistencia al avance se divide en:

$$R_T = R_V + R_{AP} + R_W + R_B + R_{TR} + R_A \qquad (6.1)$$

Donde:

$R_T = Resistencia\ total.$
$R_V = Resistencia\ viscosa.$
$R_{AP} = Resistencia\ de\ los\ apéndices.$
$R_W = Resistencia\ por\ formación\ de\ olas.$

[32] «Thermopylae, el primer buque de la clase HERO. | VA DE BARCOS».

$R_B = Resistencia\ de\ presión\ producida\ por\ el\ bulbo.$
$R_{TR} = Resistencia\ de\ presión\ por\ las\ popas\ de\ estampa\ cuando\ están\ sumergidas.$
$R_A = Resistencia\ debida\ al\ coeficiente\ de\ correlación\ modelo - buque, C_A.$

Una vez definida la división de la resistencia total, obtenemos las distintas ecuaciones que nos permiten estimar las distintas resistencias previamente mencionadas.

6.1 Resistencia viscosa, RV

$$R_V = \frac{1}{2} \cdot \rho \cdot S_m \cdot v^2 \cdot C_F \cdot (1 + k_1) \quad (6.2)$$

Donde S_m es la superficie mojada del buque, v es la velocidad del buque, ρ es la densidad del fluido en el que se encuentra, C_F es el coeficiente de resistencia de fricción, que se puede calcular según la fórmula de la ITTC-57, y el valor de $(1+k_1)$ se obtuvo de forma estadística, aplicando las siguientes expresiones:

$$(1 + k_1) = 0.93 + 0.487118 * C_{14} * \left(\frac{B}{L_F}\right)^{1.06806} * \left(\frac{T}{L_F}\right)^{0.46106} * \left(\frac{L_F}{L_R}\right)^{0.121563} * \left(\frac{L_F^3}{\nabla}\right)^{0.36486} * (1 - C_P)^{-0.604247} \quad (6.3)$$

$$\frac{L_R}{L_F} = 1 - C_P + \frac{0.06 * C_P * LCB}{4 * C_P - 1} \quad (6.4)$$

$$C_F = \frac{0.075}{(\log_{10} Rn - 2)^2} \quad (6.5)$$

Para la estimación de $(1+k_1)$ el coeficiente prismático (C_P) se obtiene considerando la eslora en la flotación (L_F), LCB es la posición longitudinal del centro de carena desde la sección media (positivo a proa y negativo a popa) en forma de porcentaje respecto a la eslora en la flotación. En caso de no conocer la superficie mojada del buque, este método nos proporciona la siguiente fórmula para su estimación, donde A_{BT} es el área transversal del bulbo:

$$S_m = L \cdot (2 \cdot T + B) \cdot C_M^{0.5} \cdot \left(0.453 + 0.4425 \cdot C_B - 0.2862 \cdot C_M - 0.003467 \cdot \frac{B}{T} + 0.3696 \cdot C_F\right) + 2.38 \cdot \frac{A_{BT}}{C_B} \quad (6.6)$$

El coeficiente C_{14} busca introducir la forma de popa en el factor $(1+k_1)$:

$C_{stern} = -25$	Popas tipo góndola
$C_{stern} = -10$	Cuadernas en V
$C_{stern} = 0$	Cuadernas normales
$C_{stern} = +10$	Cuadernas en U con popa Hogner

Tabla 6.2 - Valores del coeficiente Cstern en función de las formas del buque.

$$C_{14} = 1 + 0.011 \cdot C_{stern} \quad (6.7)$$

6.2 Resistencia de los apéndices, R_{AP}

$$R_{AP} = \frac{1}{2} \cdot \rho \cdot S_{AP} \cdot v^2 \cdot C_F \cdot (1 + k_2)_{eq} \quad (6.8)$$

$$(1 + k_2)_{eq} = \frac{\sum S_i \cdot (1 + k_2)_i}{\Sigma S_i} \quad (6.9)$$

Para los distintos tipos de apéndices, se dan unos valores aproximados de $(1+k_2)$ a partir de los cuales se obtiene $(1+k_2)_{eq}$.

Tipo de apéndice	$(1 + k_2)$
Timón buque 1 hélice	1.3 a 1.5
Timón buque 2 hélices	2.8
Timón y quillote	1.5 a 2.0
Quillote solo	1.5 a 2.0
Arbotantes	3.0
Tipo de apéndice	$(1 + k_2)$
Henchimientos protectores	3.0
Henchimientos integrados	2.0
Ejes	2.0 a 4.0
Aletas estabilizadoras	2.8
Domo	2.7
Quillas de balance	1.4

Tabla 6.3 - Valor aproximado de $(1+k_2)$ según el tipo de apéndice.

6.3 Resistencia por formación de olas, R_W

Al tratarse de un buque Ro-Ro, una de las restricciones del rango de aplicación es que el número de Froude no sea superior a 0.35, por lo que siempre emplearemos para este apartado de la resistencia por formación de olas el caso dado para valores inferiores o iguales a 0.4 que encontramos en este método. Una vez conocida esta condición, podemos definir esta resistencia como:

$$R_W = \rho \cdot g \cdot \nabla \cdot C_1 \cdot C_2 \cdot C_5 \cdot e^{(m_1 \cdot Fn^d + m_2 \cdot \cos(\lambda \cdot Fn^{-2}))} \quad (6.10)$$

$$C_1 = 2223105 \cdot C_7^{3.78613} \cdot \left(\frac{T}{B}\right)^{1.07961} \cdot (90 - i_E)^{-1.37565} \quad (6.11)$$

Donde i_E es el semi–ángulo de entrada en la flotación en grados, si no se conoce se puede estimar mediante la siguiente fórmula:

$$i_E = 1 + 89 \cdot e^{\left(-\left(\frac{L_F}{B}\right)^{0.80856} \cdot (1-C_F)^{0.30484} \cdot (1-C_P-0.0225 \cdot LCB)^{0.6367} \cdot \left(\frac{L_R}{B}\right)^{0.34574} \cdot \left(\frac{100 \cdot \nabla}{L_F^3}\right)^{0.16302}\right)} \quad (6.12)$$

$C_7 = 0.229577 \cdot \left(\frac{B}{L_F}\right)^{0.33333}$	$para \frac{B}{L_F} \leq 0.11$
$C_7 = \frac{B}{L_F}$	$para\ 0.11 \leq \frac{B}{L_F} \leq 0.25$

$C_7 = 0.5 - 0.0625 \cdot \frac{B}{L_F}$	$para \frac{B}{L_F} > 0.25$

Tabla 6.4 - Valores del coeficiente C_7 en función de la relación B/L_F.

$$d = -0.9 \quad (6.13)$$

$$C_2 = e^{-1.89 \cdot \sqrt{C_3}} \quad (6.14)$$

$$C_3 = \frac{0.56 \cdot A_{BT}^{1.5}}{B \cdot T \cdot \left(0.31 \cdot \sqrt{A_{BT}} + T_{pr} - h_B\right)} \quad (6.15)$$

Donde h_B es la distancia vertical desde la línea base hasta el centro de gravedad de la sección del bulbo de proa.

$$C_5 = 1 - \frac{(0.8 \cdot A_{TR})}{B \cdot T_m \cdot C_M} \quad (6.16)$$

$$m_1 = 0.014047 \cdot \frac{L_F}{T} - 1.75254 \cdot \frac{\nabla^{\frac{1}{3}}}{L_F} - 4.79323 \cdot \frac{B}{L_F} - C_{16} \quad (6.17)$$

De la misma forma que se ha comentado anteriormente, para este tipo de buques, el valor de C_P siempre será menor a 0.8, por tanto, aplicaremos la siguiente expresión:

$$C_{16} = 8.07981 \cdot C_P - 13.8673 \cdot C_P^2 + 6.984388 \cdot C_P^3 \quad (6.18)$$

$$m_2 = C_{15} \cdot C_P^2 \cdot 0.4 \cdot e^{-0.1 \cdot Fn^{-2}} \quad (6.19)$$

$C_{15} = -1.69385$	$Si \frac{L_F^3}{\nabla} \leq 512$
$C_{15} = -1.69385 + \frac{\frac{L_F}{\nabla^{\frac{1}{3}}} - 8.0}{2.36}$	$Si\ 512 \leq \frac{L_F^3}{\nabla} \leq 1727$
$C_{15} = 0$	$Si\ \frac{L_F^3}{\nabla} > 1727$
$\lambda = 1.446 \cdot C_P - 0.03 \cdot \frac{L_F}{B}$	$Si \frac{L_F}{B} \leq 12$
$\lambda = 1.446 \cdot C_P$	$Si \frac{L_F}{B} > 12$

Tabla 6.5 - Valores de los coeficientes C_{15} y λ.

6.4 Resistencia de presión producida por el bulbo, R_B

$$R_B = 0.11 \cdot e^{-P_B^{-2}} \cdot \frac{Fn_i^3 \cdot A_{BT}^{1.5} \cdot \rho \cdot g}{1 + Fn_i^2} \quad (6.20)$$

Donde P_B es una medida para el bulbo de proa, y Fn_i es el número de Froude referido a la inmersión. Se definen de la siguiente forma:

$$P_B = \frac{0.56 \cdot \sqrt{A_{BT}}}{T_{pr} - 1.5 \cdot h_B} \; F \qquad (6.21)$$

$$n_i = \frac{v}{\sqrt{g \cdot \left(T_{pr} - h_B - 0.25 \cdot \sqrt{A_{BT}}\right) + 0.15 \cdot v^2}} \qquad (6.22)$$

6.5 Resistencia adicional debida a la inmersión del espejo, R_{TR}

$$R_{TR} = \frac{1}{2} \cdot \rho \cdot v^2 \cdot A_{TR} \cdot C_6 \qquad (6.23)$$

El coeficiente C_6 está relacionado con el número de Froude referido al área sumergida de la estampa (Fn_{NT}):

$C_6 = 0.2 \cdot (1 - 0.2 \cdot Fn_{NT})$	$Si\ Fn_{NT} < 5$
$C_6 = 0$	$Si\ Fn_{NT} \geq 5$

Tabla 6.6 - Valores del coeficiente C_6 en función del número de Froude Fn_{NT}.

$$Fn_{NT} = \frac{v}{\sqrt{\frac{2 \cdot g \cdot A_{TR}}{B + B \cdot C_F}}} \qquad (6.24)$$

6.6 Resistencia debida a la correlación modelo-buque, R_A

$$R_A = \frac{1}{2} \cdot \rho \cdot S_m \cdot v^2 \cdot C_A \qquad (6.25)$$

Siendo C_A el coeficiente de correlación. Este coeficiente también tiene en cuenta la rugosidad del casco, y la resistencia aerodinámica del buque (obra muerta). Considerando una rugosidad estándar de 150 μm, se obtienen las siguientes expresiones:

$$C_A = 0.0006 \cdot (L_F + 100)^{-0.16} - 0.00205 + 0.003 \cdot \left(\frac{L_F}{7.5}\right)^{0.5} \cdot C_B^4 \cdot C_2 \cdot (0.04 - C_4) \qquad (6.26)$$

$$C_2 = e^{-1.89 \cdot \sqrt{C_3}} \qquad (6.27)$$

$$C_3 = 1 - \frac{(0.8 \cdot A_{TR})}{B \cdot T_m \cdot C_M} \qquad (6.28)$$

$C_4 = \frac{T_{pr}}{L_F}$	$Si\ \frac{F_{pr}}{L_F} \leq 0.04$
$C_4 = 0.04$	$Si\ \frac{F_{pr}}{L_F} > 0.04$

Tabla 6.7 - Valores para el coeficiente C_4.

Si se desea predecir el valor de C_A para una rugosidad distinta a la supuesta anteriormente, podemos emplear la siguiente fórmula:

$$\Delta C_A = \frac{\left(0.105 \cdot (k_s)^{\frac{1}{3}} - 0.05579\right)}{L_F^{\frac{1}{3}}} \qquad (6.29)$$

7 Fundamentos Matemáticos

Primeramente, para desarrollar los fundamentos matemáticos relacionados con el presente trabajo, cabe mencionar que los distintos teoremas se obtienen principalmente del libro referencia *"Análisis numérico 6ª Edición"* de los autores Richard L. Burden y J. Douglas Faires.

La integración numérica está compuesta por una amplia diversidad de algoritmos para computar el valor numérico de una integral definida. En ocasiones, es indispensable evaluar la integral definida de una función que no tiene una función primitiva o la cual no es fácil de obtener. El término cuadratura numérica es más o menos sinónimo de integral numérica, especialmente si se atribuye a integrales de una dimensión.

El problema elemental estudiado por la integración numérica es calcular una solución aproximada a una integral definida del tipo:

$$\int_a^b f(x) \cdot dx \qquad (7.1)$$

El método de la cuadratura se basa en los polinomios interpolantes. En primer lugar, se seleccionan un conjunto de nodos diferentes $\{x_{0,\dots}, x_n\}$ dentro de un intervalo $[a, b]$, seguidamente integramos el polinomio interpolante, como el de Lagrange, que tiene la siguiente forma:

$$P_n(x) = \sum_{i=0}^{n} f(x_i) \cdot L_i(x) \qquad (7.2)$$

Y su término de error de truncamiento en $[a, b]$ se obtiene:

$$\int_a^b f(x) \cdot dx = \int_a^b \sum_{i=0}^{n} f(x_i) \cdot L_i(x) \cdot dx + \int_a^b \prod_{i=0}^{n} (x - x_i) \cdot \frac{f^{(n+1)}(\xi(x))}{(n+1)!} \cdot dx$$
$$= \sum_{i=0}^{n} a_i \cdot f(x_i) + \frac{1}{(n+1)!} \int_a^b \prod_{i=0}^{n} (x - x_i) \cdot f^{(n+1)}(\xi(x)) \cdot dx \qquad (7.3)$$

Donde $\xi(x)$ se encuentra en el intervalo para cada x, y $a_i = \int_a^b L_i(x) \cdot dx$ para cada $i = 0,1, \dots, n$. Por lo que la fórmula de cuadratura y el error vienen dados por:

$$\int_a^b f(x) \cdot dx \approx \sum_{i=0}^{n} a_i \cdot f(x_i) \qquad (7.4)$$

$$E(f) = \frac{1}{(n+1)!} \int_a^b \prod_{i=0}^{n} (x - x_i) \cdot f^{(n+1)}(\xi(x)) \cdot dx \qquad (7.5)$$

La regla del trapecio y la regla de Simpson se elaboran utilizando los polinomios de Lagrange, de primer y segundo grado respectivamente, con nodos equiespaciados.

7.1 Regla del trapecio

Para deducir esta regla con el objetivo de aproximar la solución de la operación $\int_a^b f(x) \cdot dx$ usaremos el polinomio lineal de Lagrange, siendo $x_0 = a, x_1 = b,\ h = b - a$.

$$P_1(x) = \frac{(x - x_1)}{(x_0 - x_1)} \cdot f(x_0) + \frac{(x - x_0)}{(x_1 - x_0)} \cdot f(x_1) \qquad (7.6)$$

Por consiguiente,

$$\int_a^b f(x) \cdot dx = \int_{x_0}^{x_1} \left[\frac{(x - x_1)}{(x_0 - x_1)} \cdot f(x_0) + \frac{(x - x_0)}{(x_1 - x_0)} \cdot f(x_1)\right] \cdot dx + \frac{1}{2} \int_{x_0}^{x_1} f''\big(\xi(x)\big) \cdot (x - x_0) \cdot (x - x_1) \cdot dx \qquad (7.7)$$

Dado que $(x - x_0) \cdot (x - x_1)$ no cambia de signo en $[x_0, x_1]$, podemos aplicar el teorema del valor medio del cálculo integral al término de error para obtener:

$$\int_{x_0}^{x_1} f''\big(\xi(x)\big) \cdot (x - x_0) \cdot (x - x_1) \cdot dx = f''(\xi) \cdot \int_{x_0}^{x_1} (x - x_0) \cdot (x - x_1) \cdot dx = f''(\xi) \cdot \left[\frac{x^3}{3} - \frac{(x_1 - x_0)}{2} \cdot x^2 + x_0 \cdot x_1 \cdot x\right]_{x_0}^{x_1} = -\frac{h^3}{6} \cdot f''(\xi) \qquad (7.8)$$

En consecuencia, la ecuación (7.3) implica que:

$$\int_a^b f(x) \cdot dx = \left[\frac{(x - x_1)^2}{2(x_0 - x_1)} \cdot f(x_0) + \frac{(x - x_0)^2}{2(x_1 - x_0)} \cdot f(x_1)\right]_{x_0}^{x_1} - \frac{h^3}{12} \cdot f''(\xi) = \frac{(x_1 - x_0)}{2} \cdot [f(x_0) + f(x_1)] - \frac{h^3}{12} \cdot f''(\xi) \qquad (7.9)$$

Puesto que $h = x_1 - x_0$, tenemos la regla del trapecio.

La regla del trapecio es uno de los métodos más utilizados para calcular aproximaciones numéricas de integrales definidas. El nombre *"Regla del trapecio"* se debe a la interpretación geométrica que se hace de la fórmula, dado que cuando f es una función con valores positivos, aproximamos $\int_a^b f(x)\, dx$ por el área de un trapecio.

$$\int_a^b f(x)\, dx = \frac{h}{2}[f(x_0) + f(x_1)] - \frac{h^3}{12} f''(\xi) \qquad (7.10)$$

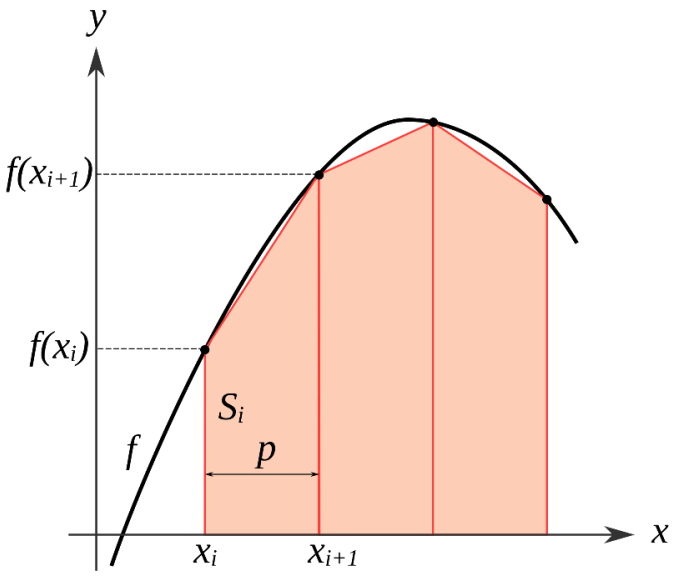


Figura 7.1 – Representación gráfica de la regla de los trapecios.

Se observa que el término de error contiene f'', esto significa que el resultado será exacto cuando se aplica a una función cuya segunda derivada sea cero, es decir, cualquier polinomio de grado 1 o inferior.

7.2 Regla de Simpson

Se procede de forma similar que anteriormente, en esta ocasión aplicamos el polinomio de Lagrange de grado 2 con los nodos $x_0 = a, x_2 = b,\ x_1 = a + h,\ h = \frac{b-a}{2}$. Por tanto:

$$\int_a^b f(x)\cdot dx = \int_{x_0}^{x_2} \left[\frac{(x-x_1)\cdot(x-x_2)}{(x_0-x_1)\cdot(x_0-x_2)} f(x_0) + \frac{(x-x_0)\cdot(x-x_2)}{(x_1-x_0)\cdot(x_1-x_2)} f(x_1) + \frac{(x-x_0)\cdot(x-x_1)}{(x_2-x_0)\cdot(x_2-x_1)}\cdot f(x_2)\right] dx + \int_{x_0}^{x_2} \frac{(x-x_0)\cdot(x-x_1)\cdot(x-x_2)}{6}\cdot f^{(3)}\big(\xi(x)\big)\cdot dx \qquad (7.11)$$

Sin embargo, al derivar la regla de Simpson de esta manera, únicamente se obtiene un término de error $O(h^4)$ que contiene $f^{(3)}$. Si planteamos el problema de otra forma, podemos derivar un término de orden superior que incluya $f^{(4)}$.

Para explicar con un ejemplo esta fórmula alternativa, supongamos que f está expandida usando un polinomio de Taylor de grado 3 alrededor de x_1. Entonces, para cada x en $[x_0, x_2]$, existe un número $\xi(x)$ en (x_0, x_2) con

$$f(x) = f(x_1) + f'(x_1)\cdot(x-x_2) + \frac{f''(x_1)}{2}\cdot(x-x_1)^2 + \frac{f'''(x_1)}{6}\cdot(x-x_1)^3 + \frac{f^{(4)}\big(\xi(x)\big)}{24}\cdot(x-x_1)^4 \qquad (7.12)$$

Dado que $(x - x_1)^4$ nunca es negativo en $[x_0, x_2]$, el teorema del valor medio del cálculo integral implica que:

$$\frac{1}{24} \cdot \int_{x_0}^{x_2} f^{(4)}\big(\xi(x)\big) \cdot (x - x_1)^4 \cdot dx = \frac{f^{(4)}(\xi_1)}{24} \cdot \int_{x_0}^{x_2} (x - x_1)^4 \cdot dx = \left[\frac{f^{(4)}(\xi_1)}{120} \cdot (x - x_1)^5\right]_{x_0}^{x_2} \quad (7.13)$$

Para cualquier número ξ_1 en (x_0, x_2).

Sin embargo, $h = x_2 - x_1 = x_1 - x_0$,

$$(x_2 - x_1)^2 - (x_0 - x_1)^2 = (x_2 - x_1)^4 - (x_0 - x_1)^4 = 0 \quad (7.14)$$

Y en cambio,

$$(x_2 - x_1)^3 - (x_0 - x_1)^3 = 2h^3 \;\; y \; (x_2 - x_1)^5 - (x_0 - x_1)^5 = 2h^5 \quad (7.15)$$

En consecuencia, podemos reescribir la ecuación (13) como:

$$\int_{x_0}^{x_2} f(x) \cdot dx = 2 \cdot h \cdot f(x_1) + \frac{h^3}{3} \cdot f''(x_1) + \frac{f^{(4)}(\xi_1)}{60} \cdot h^5 \quad (7.16)$$

Si ahora reemplazamos $f''(x_1)$ por la aproximación de la ecuación:

$$f''(x_0) = \frac{1}{h^2} \cdot [f(x_0 - h) - 2 \cdot f(x_0) + f(x_0 + h)] - \frac{h^2}{12} \cdot f^{(4)}(\xi) \quad (7.17)$$

Tendremos:

$$\int_{x_0}^{x_2} f(x) \cdot dx = 2 \cdot h \cdot f(x_1) + \frac{h^3}{3} \cdot \left\{\frac{1}{h^2} \cdot [f(x_0) - 2 \cdot f(x_1) + f(x_2)] - \frac{h^2}{12} \cdot f^{(4)}(\xi_2)\right\} + \frac{f^{(4)}(\xi_1)}{60} \cdot h^5 = \frac{h}{3} \cdot [f(x_0) + 4 \cdot f(x_1) + f(x_2)] - \frac{h^5}{12} \cdot \left[\frac{1}{3} \cdot f^{(4)}(\xi_2) - \frac{1}{5} \cdot f^{(4)}(\xi_1)\right] \quad (7.18)$$

Esto nos da la regla de Simpson, que se suele expresar de la siguiente forma:

$$\int_{x_0}^{x_2} f(x) \cdot dx = \frac{h}{3} \cdot [f(x_0) + 4 \cdot f(x_1) + f(x_2)] - \frac{h^5}{90} \cdot f^{(4)}(\xi) \quad (7.19)$$

Dado que el término de error contiene la cuarta derivada de f, esto significa que proporciona soluciones exactas al aplicarla a un polinomio cualquiera de grado tres o inferior. Por tanto, en todos los casos la regla de Simpson es mucho mejor que el método de los trapecios.

La derivación normal de las fórmulas del error de cuadratura se basa en determinar la clase de polinomios con los cuales estas fórmulas producen resultados exactos. El grado de exactitud o precisión de una fórmula de cuadratura es el entero positivo más grande n, tal que la fórmula sea exacta para x^k, cuando k = 0, 1, 2, …, n. Esto implica que las reglas del trapecio y de Simpson tienen un grado de precisión de uno y tres, respectivamente.

7.3 Fórmulas de Newton-Cotes

Las reglas ya mencionadas, son ejemplos de una clase de métodos denominados fórmulas de Newton-Cotes. Se puede distinguir entre: abiertas y cerradas.

Se le denomina cerrada a aquellas donde los extremos del intervalo se incluyen como nodos. La fórmula cerrada para $(n+1)$ puntos utiliza los nodos $x_i = x_0 + i \cdot h$, para i = 0, 1, 2, …, n, donde $x_0 = a, x_n = b$ y $h = (b-a)/n$. La fórmula adquiere la siguiente forma:

$$\int_a^b f(x) \cdot dx \approx \sum_{i=0}^{n} a_i \cdot f(x_i) \qquad (7.20)$$

Donde:

$$a_i = \int_a^b L_{i\,(x)} \cdot dx = \int_{x_0}^{x_n} \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{(x - x_j)}{(x_i - x_j)} \cdot dx \qquad (7.21)$$

A continuación, se puntualiza el análisis del error asociado a las fórmulas cerradas.

Suponiendo que $\sum_{i=0}^{n} a_i \cdot f(x_i)$ denota la fórmula cerrada de *(n+1)* puntos de Newton-Cotes, con $x_0 = a$, $x_n = b$ y $h = (b-a)/n$. Existe una $\xi \in (a, b)$ para la cual:

$$\sum_{i=0}^{n} a_i \cdot f(x_i) + \frac{h^{n+3} \cdot f^{(n+2)}(\xi)}{(n+2)!} \int_0^n t^2 \cdot (t-1) \ldots (t-n) \cdot dt \qquad (7.22)$$

- Si n es par, si $f \in C^{n+2}[a, b]$, y

$$\int_a^b f(x) \cdot dx = \sum_{i=0}^{n} a_i \cdot f(x_i) + \frac{h^{n+2} \cdot f^{(n+1)}(\xi)}{(1+2)!} \int_0^n (t-1) \ldots (t-n) \cdot dt \qquad (7.23)$$

- Si n es impar, si $f \in C^{n+1}[a, b]$.

Se observa que, cuando n es par, el grado de exactitud es $n+1$ aunque el polinomio de interpolación es, como máximo, de grado n. En el caso de que n sea impar, el grado de exactitud será n.

A continuación, se incluyen algunas de las fórmulas cerradas comunes que aplicaremos en este trabajo:

Regla del trapecio $(n = 1)$:

$$\int_{x_0}^{x_1} f(x) \cdot dx = \frac{h}{2} \cdot [f(x_0) + f(x_1)] - \frac{h^3}{12} \cdot f''(\xi), \text{donde } x_0 < \xi < x_1 \qquad (7.24)$$

Regla de Simpson ($n = 2$):

$$\int_{x_0}^{x_2} f(x) \cdot dx = \frac{h}{3} \cdot [f(x_0) + 4 \cdot f(x_1) + f(x_2)] - \frac{h^5}{90} \cdot f^{(4)}(\xi), donde\ x_0 < \xi < x_2 \qquad (7.25)$$

Regla de tres octavos de Simpson ($n = 3$):

$$\int_{x_0}^{x_3} f(x) \cdot dx = \frac{3 \cdot h}{8} \cdot [f(x_0) + 3 \cdot f(x_1) + 3 \cdot f(x_2) + f(x_3)] - \frac{3 \cdot h^5}{80} \cdot f^{(4)}(\xi), \quad donde\ x_0 < \xi < x_3 \qquad (7.26)$$

Para el caso de $n = 4$:

$$\int_{x_0}^{x_4} f(x) \cdot dx = \frac{2 \cdot h}{45} \cdot [7 \cdot f(x_0) + 32 \cdot f(x_1) + 12 \cdot f(x_2) + 32 \cdot f(x_3) + 7 \cdot f(x_4)] - \frac{8 \cdot h^7}{945} \cdot f^{(6)}(\xi), \qquad donde\ x_0 < \xi < x_4 \qquad (7.27)$$

Por lo general, estas fórmulas no son adecuadas para su uso en grandes intervalos de integración. Para estos casos se requieren fórmulas de mayor grado, y los valores de sus coeficientes son complejos de obtener. Además, las fórmulas de Newton-Cotes se basan en la interpolación de polinomios utilizando nodos equiespaciados, procedimiento que resulta inexacto en grandes intervalos debido a la naturaleza de los polinomios de mayor grado.

El *método fragmentario* se usa para realizar la integración numérica, en la que se aplican las fórmulas de Newton-Cotes de bajo orden. *Estos son los métodos más utilizados*. Si queremos aplicar un método fragmentario, dividimos el intervalo $[a, b]$ en n subintervalos equiespaciados y se aplica la regla de Simpson a cada par consecutivo de subintervalos.

Con $h = (b - a)/n$ y $x_j = a + j \cdot h$ para cada $j = 0,1, \dots, n$, tenemos:

$$\int_a^b f(x) \cdot dx = \sum_{j=1}^{n/2} \int_{x_{2j-2}}^{x_{2j}} f(x) \cdot dx = \sum_{j=1}^{n/2} \left\{ \frac{h}{3} \cdot \left[f\left(x_{2j-2}\right) + 4 \cdot f\left(x_{2j-1}\right) + f\left(x_{2hj}\right) \right] - \frac{h^5}{90} \cdot f^{(4)}(\xi_j) \right\} \qquad (7.28)$$

Para alguna ξ_j con $x_{2j-2} < \xi_j < x_{2j}$, siempre que $f \in C^4[a, b]$. Al aplicar el hecho de que, para cada $j = 1,2, \dots, (n/2) - 1$, tenemos que $f(x_{2j})$ aparece en el término correspondiente al intervalo $[x_{2j-2}, x_{2j}]$ y también en el término correspondiente al intervalo $[x_{2j}, x_{2j+2}]$, podemos reducir esta suma a:

$$\int_a^b f(x)\cdot dx = \frac{h}{3}\cdot\left[f(x_0)+2\cdot\sum_{j=l}^{(n/2)-1} f\left(x_{2j}\right)+4\cdot\sum_{j=1}^{(n,2)} f\left(x_{2j-1}\right)+f(x_n)\right] - \frac{h^5}{90}\cdot\sum_{j=1}^{n/2} f^{(4)}\left(\xi_j\right) \quad (7.29)$$

El error asociado con esta aproximación es:

$$E(f) = -\frac{h^5}{90}\cdot\sum_{j=1}^{n/2} f^{(4)}\left(\xi_j\right) \quad (7.30)$$

Donde $x_{2j-2} < \xi_j < x_{2j}$ para cada $j = 1,2,\dots,(n/2)$. Si $f \in C^4[a,b]$, el teorema de los valores extremos implica que $f^{(4)}$ asume su máximo y mínimo en $[a,b]$. Puesto que:

$$\min_{x\in[a,b]} f^{(4)}(x) \le f^{(4)}\left(\xi_j\right) \le \max_{x\in[a,b]} f^{(4)}(x) \quad (7.31)$$

Tenemos,

$$\frac{n}{2}\cdot\min_{x\in[a,b]} f^{(4)}(x) \le \sum_{j=1}^{\frac{n}{2}} f^{(4)}\left(\xi_j\right) \le \frac{n}{2}\cdot\max_{x\in[a,b]} f^{(4)}(x) \quad (7.32)$$

Y

$$\min_{x\in[a,b]} f^{(4)}(x) \le \frac{2}{n}\cdot\sum_{j=1}^{n/2} f^{(4)}\left(\xi_j\right) \le \max_{x\in[a,b]} f^{(4)}(x) \quad (7.33)$$

De acuerdo con el teorema del valor intermedio, existe $\mu \in (a,b)$ tal que

$$f^{(4)}(\mu) = \frac{2}{n}\cdot\sum_{j=1}^{n/2} f^{(4)}\left(\xi_j\right) \quad (7.34)$$

Por tanto,

$$E(f) = -\frac{h^5}{180}\cdot n\cdot f^{(4)}(\mu) \quad (7.35)$$

O, como $n = (b-a)/h$,

$$E(f) = -\frac{b-a}{180}\cdot h^4\cdot f^{(4)}(\mu) \quad (7.36)$$

Las observaciones anteriores producen el siguiente resultado.

Sean $f \in C^4[a,b]$, n par, $h = (b-a)/n$ y $x_j = a+jh$ para cada $j = 0,1,\dots,n$. Existe $\mu \in (a,b)$ tal que la regla compuesta de Simpson para n subintervalos puede escribirse con su término de error como:

$$\int_a^b f(x)\cdot dx = \frac{h}{3}\cdot\left[f(a)+2\cdot\sum_{j=1}^{(n/2)-1} f\left(x_{2j}\right)+4\cdot\sum_{j=1}^{n/2} f\left(x_{2j-1}\right)+f(b)\right] - \frac{b-a}{180}\cdot h^4\cdot f^{(4)}(\mu) \qquad (7.37)$$

El método de subdivisión puede emplearse en cualquiera de las fórmulas de orden inferior. La regla del trapecio requiere sólo un intervalo en cada aplicación, por lo cual el entero n puede ser par o impar.

Sean $f \in C^2[a,b]$, $h=(b-a)/n$ y $x_j = a+jh$ para cada $j = 0,1,\dots,n$. Existe $\mu \in (a,b)$ tan que la regla compuesta del trapecio para n subintervalos puede escribirse con su término de error como:

$$\int_a^b f(x)\cdot dx = \frac{h}{2}\cdot\left[f(a)+2\cdot\sum_{j=1}^{n-1} f\left(x_j\right)+f^{\beta}\right] - \frac{b-a}{12}\cdot h^2\cdot f''(\mu) \qquad (7.38)$$

Una propiedad importante que comparten todos los métodos de integración compuesta de Newton-Cotes es la estabilidad respecto al error de redondeo. Para demostrarla, supongamos que aplicamos la regla compuesta de Simpson con n subintervalos a una función f en $[a,b]$ y que determinamos la cota máxima de dicho error. Supongamos que aproximamos $f(x_i)$ mediante$\tilde{f}(x_i)$ y que

$$f(x_i) = \tilde{f}(x_i) + e_i, por\ cada\ i = 0,1,\dots,n \qquad (7.39)$$

Donde e_i denota el error de redondeo que implica usar $\tilde{f}(x_i)$ para aproximar $f(x_i)$. Entonces, el error acumulado, $e(h)$, en la regla compuesta de Simpson es

$$e(h) = \left|\frac{h}{3}\cdot\left[e_0+2\cdot\sum_{j=1}^{(n/2)-1} e_{2j}+4\cdot\sum_{j=1}^{n/2} e_{2j-1}+e_n\right]\right| \leq \frac{h}{3}\cdot\left[|e_0|+2\cdot\sum_{j=1}^{(n/2)-1}\left|e_{2j}\right|+4\cdot\sum_{j=1}^{n/2}\left|e_{2j-1}\right|+|e_n|\right] \qquad (7.40)$$

Si los errores de redondeo están uniformemente acotados por ε, entonces

$$e(h) \leq \frac{h}{3}\cdot\left[\varepsilon+2\cdot\left(\frac{n}{2}-1\right)\cdot\varepsilon+4\cdot\left(\frac{n}{2}\right)\cdot\varepsilon+\varepsilon\right] = \frac{h}{3}\cdot 3\cdot n\cdot\varepsilon = n\cdot h\cdot\varepsilon \qquad (7.41)$$

Pero $nh = b-a$, de modo que,

$$e(h) \leq (b-a)\cdot\varepsilon \qquad (7.42)$$

Es una cota independiente de h. Este resultado indica que el procedimiento es estable al aproximarse h a cero.

8 Funcionalidad y Caso práctico

Una vez explicados los métodos que se deben aplicar y la forma de trabajo del programa, comenzamos a estructurar la realización del programa. Se comienza mediante la creación de distintas funciones para los diversos pasos que necesitaremos, como la obtención de la curva de pesos, la curva de empujes, etc.

Como anteriormente se ha comentado, utilizamos la herramienta GUI de MATLAB para la creación de nuestra interfaz. Inicialmente seleccionamos la opción de *"Create New GUI"* para crear una nueva interfaz, posteriormente si queremos modificarla, podremos acceder a ella desde la opción *"Open Existing GUI"*.

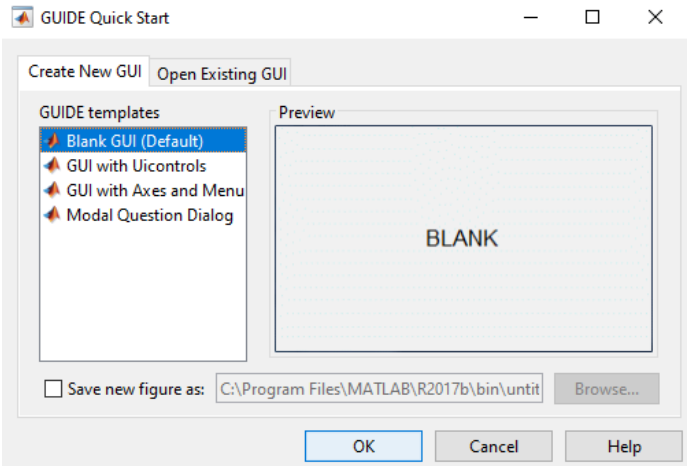


Figura 8.1 - Ventana para la creación de una interfaz en blanco o selección de una existente.

Una vez creada la interfaz, obtenemos las siguientes herramientas para crear los diferentes botones, gráficos y textos:

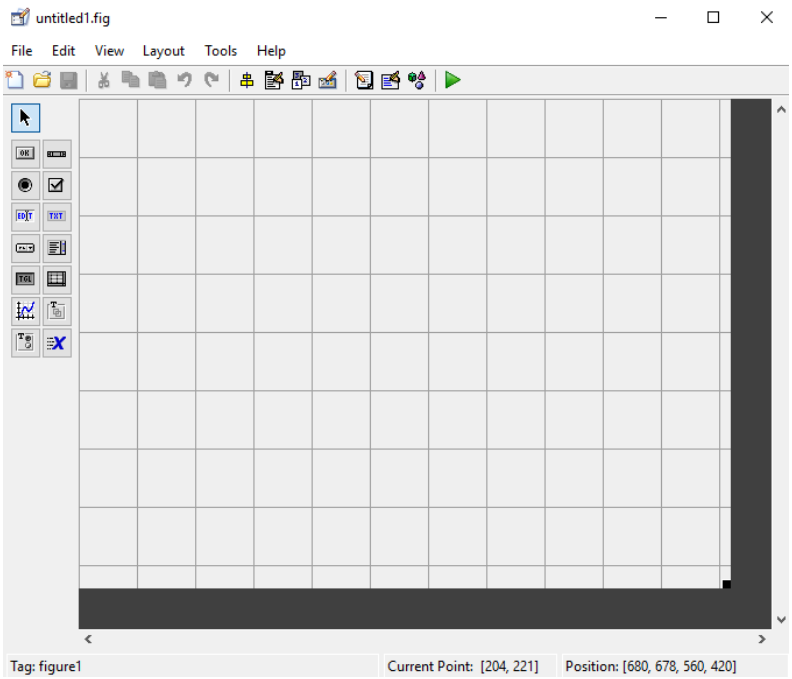


Figura 8.2 - Ventana para la personalización de la interfaz.

Sabiendo esto, se procede a la creación de la interfaz de inicio para la ejecución del programa, obteniendo la siguiente estructura:

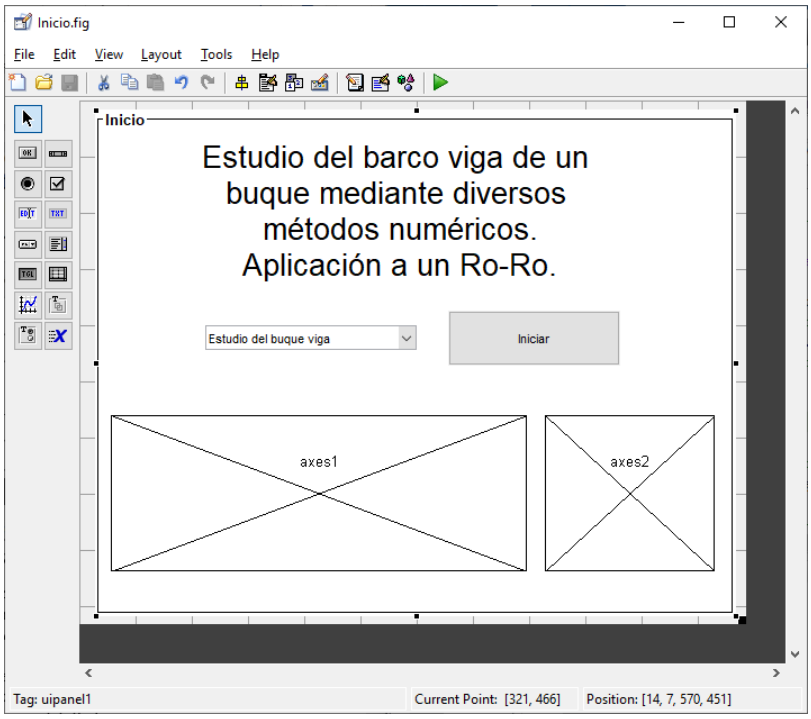


Figura 8.3 - Creación del programa Inicio.

Podemos ver que se colocan dos funciones *axes* que se emplean, en este caso, para insertar imágenes, esta parte del código debe insertarse en el inicio del programa para que se muestren al arrancar. Así mismo, una parte del código ejecutado inicialmente, se destina para centrar la ventana dentro de la pantalla del ordenador, de forma que sea más sencillo su uso, El botón *"Iniciar"* es del estilo *"pushbutton"* de forma que podemos pulsar varias veces y siempre se ejecutará la función asignada, que consiste en ejecutar un nuevo programa en función de la opción seleccionada en el menú desplegable, estas pueden ser tanto la aplicación objetivo de este proyecto, el cálculo del barco viga, como la aplicación del método de Holtrop a partir del cual podemos obtener una predicción de potencia para el buque.

Figura 8.4 - Estilo final de la interfaz creada para el programa Inicio.

El programa se puede ejecutar directamente desde la consola, llamando al programa como *Inicio*, desde la propia ventana de la creación de la interfaz mediante el botón con forma triangular verde (*"Run Figure"*), o desde la ventana del editor de Matlab (*"Run"*).

8.1 Método de Holtrop y selección del motor

Para comenzar la aplicación de este método, se debe especificar el tipo de agua, salada o dulce, para determinar la densidad y la propia temperatura, para aproximar la viscosidad cinemática mediante las siguientes expresiones:

$$v_{ad} = \left((0.585 * 10^{-3} * (T - 12.0) - 0.03361) * (T - 12.0) + 1.2350\right) * 10^{-6} \quad (8.1)$$

$$v_{as} = \left((0.659 * 10^{-3} * (T - 1.0) - 0.05076) * (T - 1.0) + 1.7688\right) * 10^{-6} \quad (8.2)$$

Siendo T la temperatura del agua en grados centígrados.

Los parámetros necesarios que se deben conocer se introducen de forma manual a través de la interfaz, que el programa recopilará para trabajar con ellos:

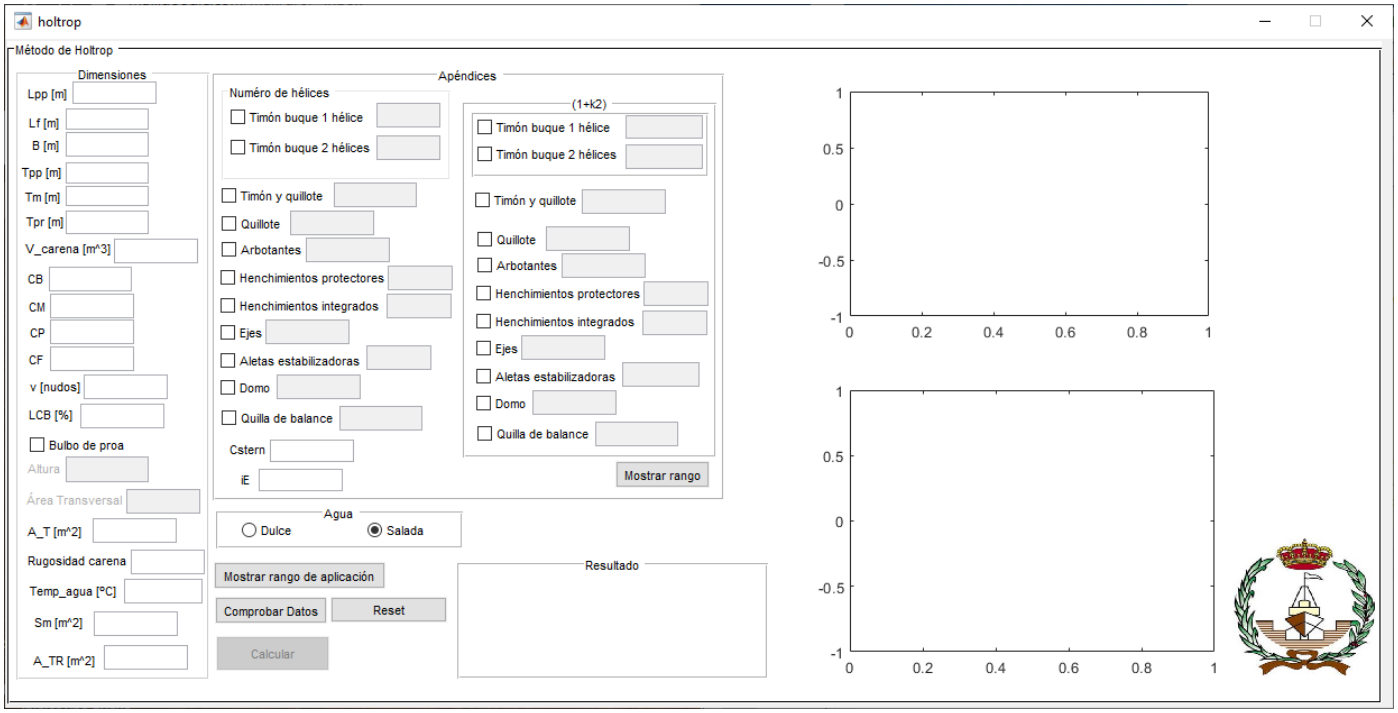


Figura 8.5 – Interfaz gráfica para la aplicación del método de Holtrop.

Los datos de nuestro buque conocido serán los siguientes:

Características del buque		Valor	
Eslora en la flotación	L_F	95	m
Eslora entre perpendiculares	L_{pp}	90	m
Velocidad de proyecto	V	7.099	m/s
Velocidad de proyecto	V	13.8	nudos
Manga máxima	B	16.5	m
Calado máximo de verano	T	6.1	m
Volumen de carena	∇	5734.03	m^3
Coeficiente de bloque	C_B	0.633	

Características del buque		Valor
Coeficiente de la maestra	C_M	0.955
Coeficiente de la flotación	C_F	0.919
Coeficiente prismático	C_P	0.659
Semi-ángulo de entrada en la flotación	i_E	27 °
Posición longitudinal del centro de carena	LCB	-1.34 %
Coeficiente de formas de popa	C_{STERN}	0
Área transversal del bulbo de proa	A_{BT}	6.96 m^2
Altura del bulbo de proa	h_B	2.198 m
Área transversal de la popa de espejo	A_{TR}	3.238 m^2
Rugosidad de la carena	k_s	150 µm
Temperatura del agua	T	15 °C

Tabla 8.1 – Datos característicos del buque necesarios.

Características de los apéndices		Valor
Superficie mojada del timón y 1 hélice	A_{Tim}	10 m^2
$(1+k_2)_{timón}$		1.4
Superficie mojada del eje	A_{eje}	3.14 m^2
$(1+k_2)_{eje}$		3

Tabla 8.2 – Datos característicos de los apéndices del buque.

Podemos ver que disponemos de dos botones que nos muestran los distintos rangos de aplicación descritos en las tablas 6.1 y 6.3.

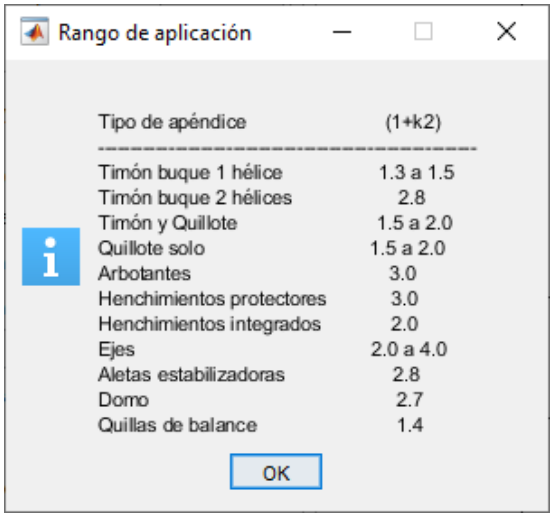


Figura 8.6 – Rango válido para los valores de $(1+k_2)$ de los apéndices.

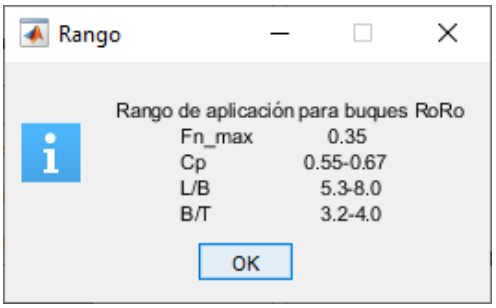


Figura 8.7 – Rango válido para los valores característicos del buque.

Estos datos se validan mediante el botón *"Comprobar Datos"*, donde también se comprueba que todas las casillas contengan la información necesaria, es decir, sin tener valores negativos o dejar casillas vacías, para los distintos casos, se obtienen los siguientes mensajes:

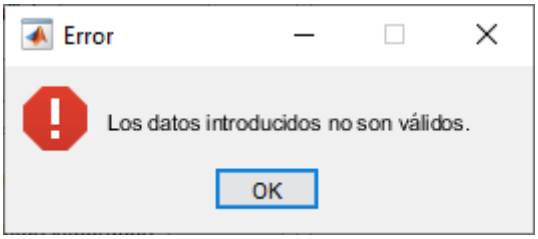


Figura 8.9 – Error al comprobar los datos introducidos.

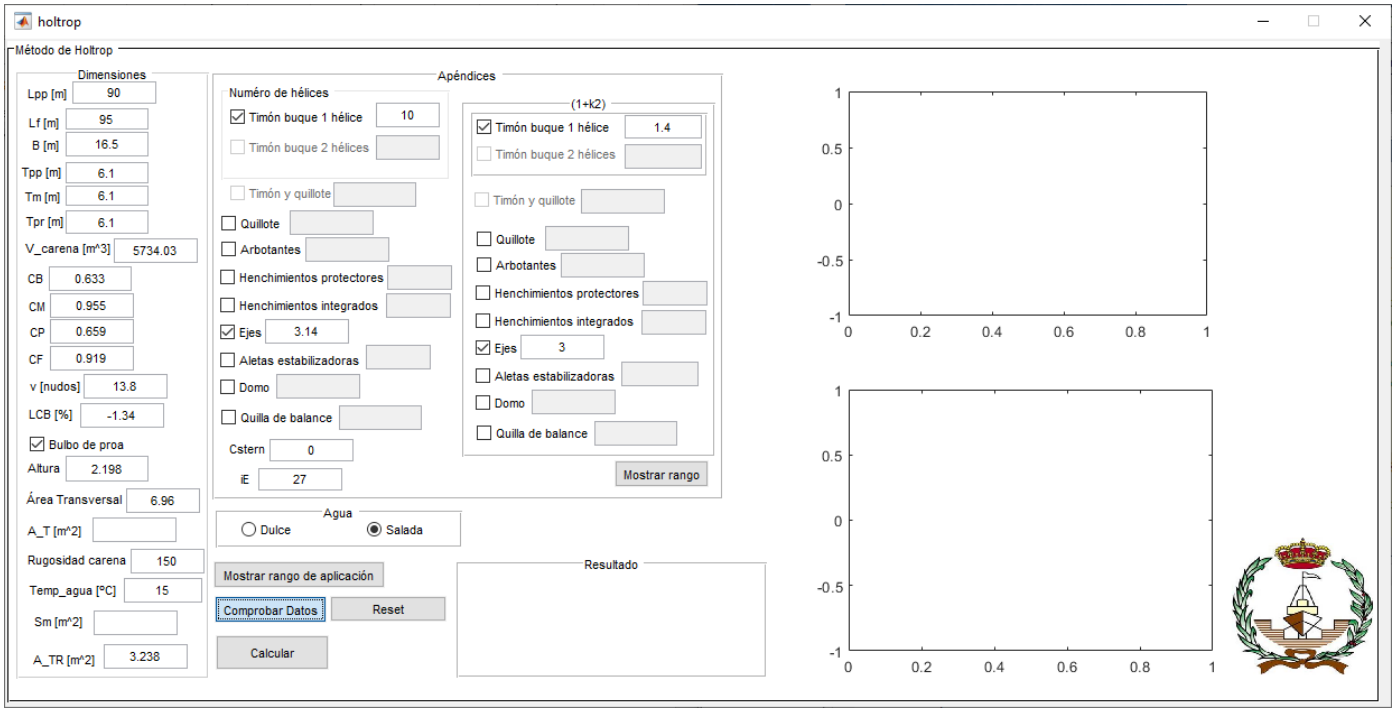


Figura 8.8 - Datos introducidos en el programa.

Para poder introducir las características correspondientes al bulbo de proa, es necesario elegir la opción de *"Bulbo de Proa"* que nos proporciona. Podemos ver que cuando seleccionamos una de las tres opciones dentro de las opciones para el apéndice del timón, el resto se bloquean, para evitar la selección de dos opciones no compatibles. También se observa que, al seleccionar cualquier opción para introducir el área, automáticamente se selecciona la casilla correspondiente para introducir el valor de $(1+k_2)$, y viceversa.

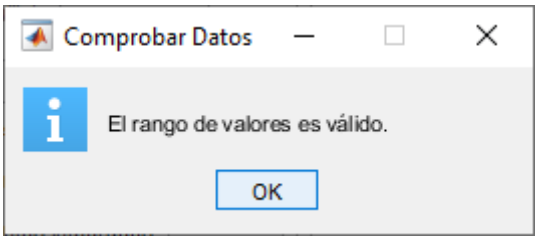


Figura 8.10 – Validación al comprobar los datos introducidos.

Una vez comprobados los datos, podemos ver que el programa nos desbloquea el botón *"Calcular"*. Para producir los resultados y mostrarlos por pantalla, simplemente es necesario pulsar sobre éste, de forma que obtenemos lo siguiente:

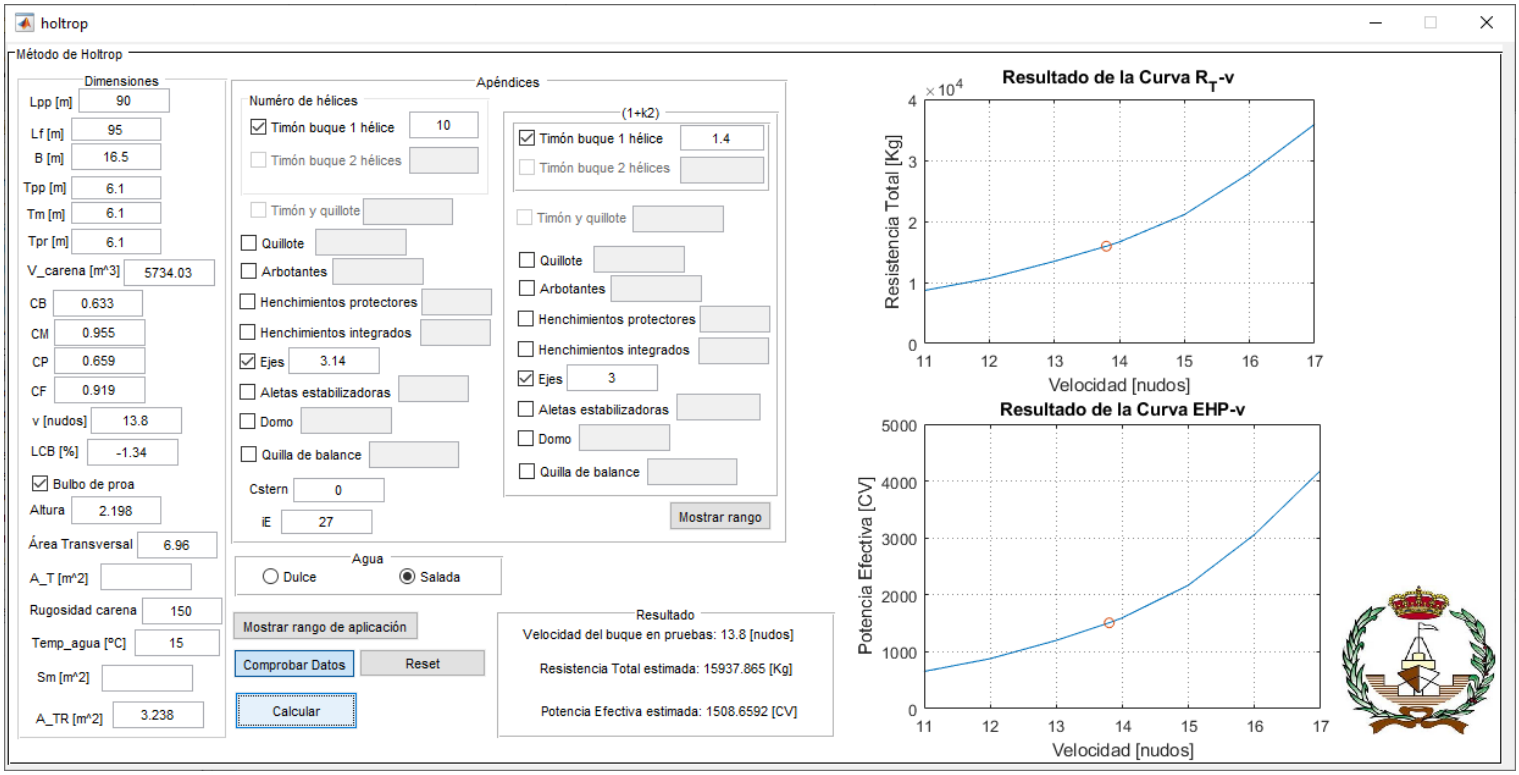


Figura 8.11 – Resultados obtenidos en el método de Holtrop para el buque a estudio.

Donde podemos observar en la casilla *"Resultado"* los valores para la velocidad correspondiente de resistencia total y potencia efectiva. . Con esta primera estimación de la potencia efectiva, aplicando los rendimientos correspondientes, que posteriormente se comentarán de forma más precisa, y aplicando un coeficiente de seguridad para sobredimensionar la potencia necesaria, es posible seleccionar el motor que incorporará el buque. Otra opción para analizar el resultado es que podemos pinchar sobre cualquiera de las gráficas obtenidas, y se nos mostrará en una nueva figura, de forma ampliada:

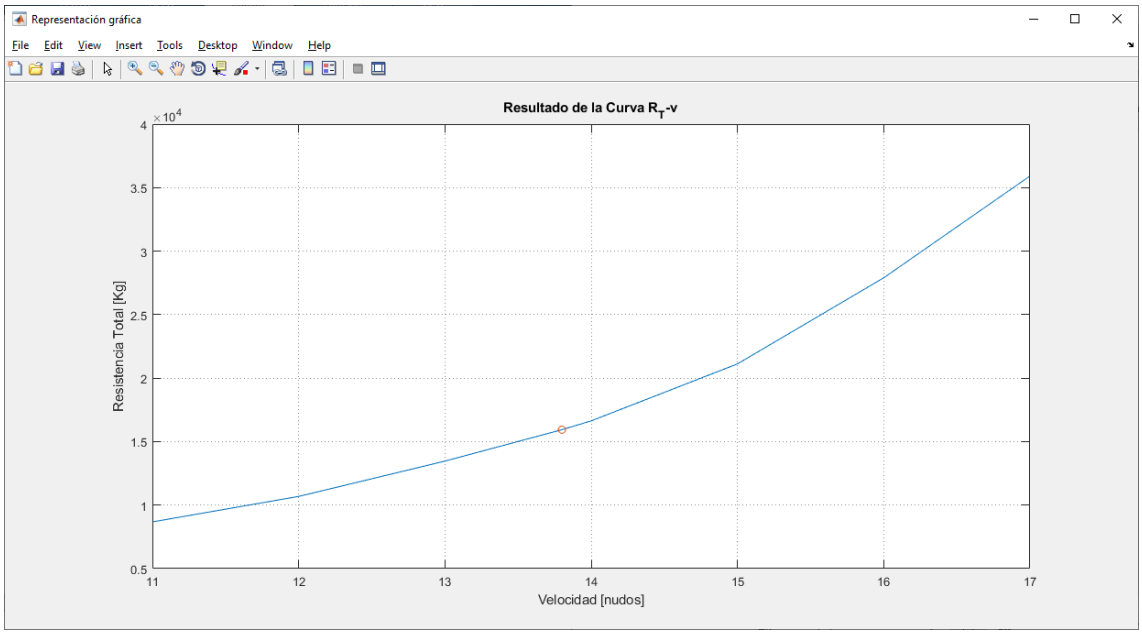


Figura 8.12 – Resultado de la curva Resistencia Total - Velocidad.

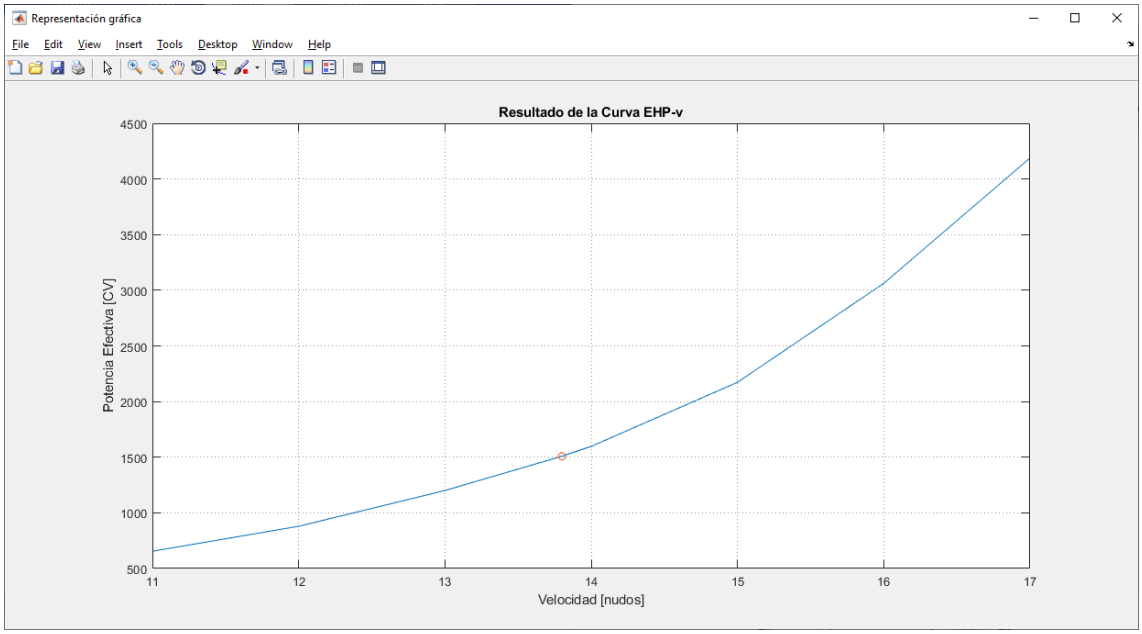


Figura 8.13 – Resultado de la curva Potencia Efectiva - Velocidad.

Los datos que construyen estas gráficas se almacenarán en un archivo Excel, con nombre *"Resultados_Holtrop.xls"*, para poder trabajar con ellos en cualquier momento.

Por otro lado, el botón *"Reset"* nos permite reiniciar el programa, de forma que se vacían las casillas, se reestablecen los resultados obtenidos y las gráficas, entre otras configuraciones, como se muestra a continuación:

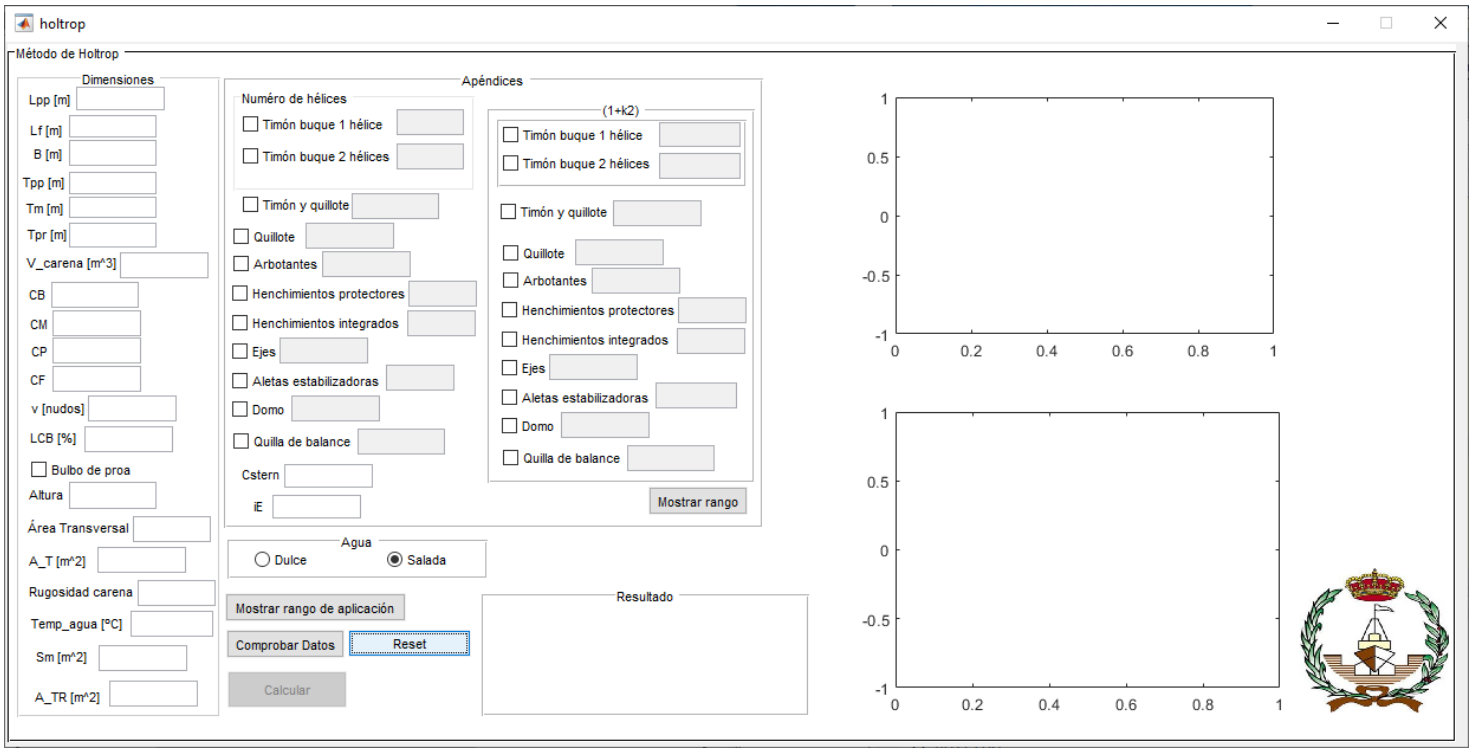


Figura 8.14 – Programa al pulsar el botón Reset.

Una vez conocidos los valores obtenidos que son los siguientes:

Velocidad del buque, v	13.8 Nudos
Resistencia total, R_T	15937.865 Kg
Potencia Efectiva, EHP	1508.66 CV

Tabla 8.3 – Resultados obtenidos en el método de Holtrop.

Se conoce que la relación entre la potencia efectiva (EHP) y la potencia al freno (BHP), que definirá el criterio de selección de nuestro motor, es la siguiente:

$$BHP = \frac{EHP}{\eta_D * \eta_M * K_P} \qquad (8.3)$$

Sabiendo que η_D es el rendimiento cuasi – propulsivo, obtenido a partir de distintas fórmulas que nos ayudan a estimar su valor, η_M es el rendimiento mecánico, y K_P un coeficiente considerado para el punto de funcionamiento del propulsor.

Buques lentos y llenos	0.85
Buques finos y rápidos	0.9

Tabla 8.4 – Valores de Kp en función del buque a estudio.

Fórmula de Lap

$$\eta_D = 0.885 - 0.00012 * N * \sqrt{L_{pp}} \qquad (8.4)$$

Fórmula de Parga

$$\eta_D = 0.84 - \frac{N * \sqrt{L_{pp}}}{18000} + \frac{\left(\frac{V}{C_B}\right)^2}{24000} \qquad (8.5)$$

Fórmula de Parga modificada

$$\eta_D = K * \left[0.84 - \frac{N * \sqrt{L_{pp}}}{18000} + \frac{\left(\frac{V}{C_B}\right)^2}{24000}\right] \quad (8.6)$$

L_{pp} (m)	200	210	220	230	240	250	260	270
K	1.03	1.02	1.01	1.00	1.00	1.00	0.99	0.99
L_{pp} (m)	280	290	300	310	320	330	340	350
K	0.99	0.98	0.98	0.98	0.98	0.98	0.98	0.98

Tabla 8.5 – Valores de la constante K en función de la eslora entre perpendiculares.

Debido a las dimensiones del buque seleccionado, esta fórmula no es posible aplicarla debido a que la eslora entre perpendiculares no se encuentra considerada dentro de los valores para la constante K.

Fórmula del Canal de El Pardo

$$\eta_D = 0.943 - 0.000187 * N * \sqrt{L_{pp}} + 0.023 * \frac{B}{T} - 0.2 * C_B + 0.00013 * N * C_B * \sqrt{L_{pp}} \quad (8.7)$$

En caso de desconocer las revoluciones a las que gira la hélice del buque, para una primera aproximación, como es este caso, podemos estimarlas a partir de la siguiente tabla:

Desplazamiento (t)	Revoluciones (rpm)
Menos de 1000	500
De 1000 a 2000	400
De 2000 a 3000	300
De 3000 a 5000	200
De 5000 a 7500	150
De 7500 a 12000	125
De 12000 a 25000	115
De 25000 a 50000	110
Más de 50000	100

Tabla 8.6 – Estimación de las revoluciones de la hélice a partir del desplazamiento del buque.

El desplazamiento de nuestro buque es de 5883.12 t, por lo que se encuentra dentro del rango de 5000 a 7500 t, esto significará que las revoluciones del propulsor inicialmente se considerarán iguales a 150 rpm. Una vez determinada esta característica, aplicamos las distintas fórmulas obteniendo los siguientes resultados:

Formula de Lap	0.7142
Fórmula de Parga	0.7807
Fórmula del Canal de El Pardo	0.7296

Tabla 8.7 – Valores del rendimiento cuasi-propulsivo estimado.

El rendimiento mecánico depende principalmente de la potencia instalada y de la existencia o no de una reductora. Por tanto:

Reducción	Potencia instalada	η_M
No	BHP < 1000 CV	0.97
No	1000 < BHP < 10000 CV	0.98
No	BHP > 10000 CV	entre 0.985 y 0.99
Si	BHP < 1000 CV	entre 0.94 y 0.95
Si	1000 < BHP < 10000 CV	entre 0.94 y 0.96
Si	BHP > 10000 CV	entre 0.96 y 0.97

Tabla 8.8 – Valores aproximados para el rendimiento mecánico.

A partir de estos valores y de la potencia efectiva obtenida, podemos suponer que la BHP obtenida se encontrará dentro del rango de valores de 1000 < BHP < 10000 CV, y dado que el motor que instalaremos será un motor rápido de cuatro tiempos (4T), necesitaremos instalar un reductor.

Por tanto, obtenemos los siguientes resultados para los distintos valores del rendimiento cuasi – propulsivo estimado anteriormente:

Formula de Lap	
K_P	0.850
η_D	0.714
η_M	0.950
EHP	1508.659 CV
BHP	2615.810 CV
Fórmula de Parga	
K_P	0.850
η_D	0.781
η_M	0.950
EHP	1508.659 CV
BHP	2392.977 CV
Fórmula del Canal de El Pardo	
K_P	0.850
η_D	0.730
η_M	0.950
EHP	1508.659 CV
BHP	2560.701 CV

Tabla 8.9 – Resultados de la potencia necesaria a instalar en función de la fórmula empleada para la estimación del coeficiente cuasi – propulsivo.

Si seleccionamos la situación más desfavorable, aquella en la que el buque necesite más potencia, y sobredimensionando este valor con un factor de 1.2 (para que el motor trabaje en su punto óptimo de funcionamiento que suele ser al 80% del MCR), obtenemos que la **potencia necesaria a instalar es de 3138.973 CV**, o lo que es lo mismo **2340.732 kW.** Una vez conocido esta potencia, seleccionamos a partir de un catálogo de motores aquellos que cumplan las características, de forma que tendremos los siguientes:

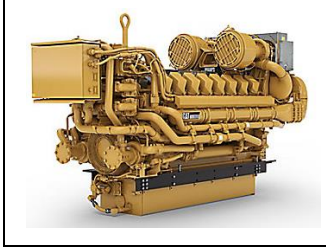
Motor de Propulsión Caterpillar C175-16	
	Gama de potencia: 2683 – 3420 BHP Gama de velocidad: 1600 – 1800 rpm

Tabla 8.10 – Características del motor C175-16.[33]

Motor de Propulsión Caterpillar 3516E TIER 4 FINAL	
	Gama de potencia: 3003 – 3150 BHP Gama de velocidad: 1800 rpm

Tabla 8.11 – Características del motor 3516E TIER 4 FINAL.[34]

Motor de Propulsión Caterpillar 3516C IMO III	
	Gama de potencia: 1650 – 3386 BHP Gama de velocidad: 1200 – 1800 rpm

Tabla 8.12 – Características del motor 3516C IMO III.[35]

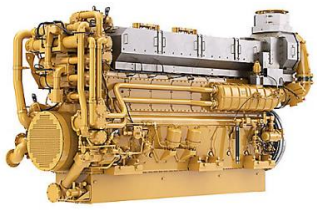
Motor de Propulsión Caterpillar C280-8	
	Gama de potencia: 1650 – 3386 BHP Gama de velocidad: 1200 – 1800 rpm

Tabla 8.13 – Características del motor C280-8.[36]

Motor de Propulsión Wärtsilä 8L26	
	Gama de potencia: 3648 BHP Gama de velocidad: 1000 rpm

Tabla 8.14 – Características del motor Wärtsilä 8L26.[37]

[33] «File:Cat | C175-16 | Caterpillar».
[34] «File:Cat | 3516E Tier 4 Final | Caterpillar».
[35] «File:Cat | 3516C IMO II | Caterpillar».
[36] «File:Cat | C280-8 | Caterpillar».
[37] «File:Wärtsilä 26 - diesel engine».

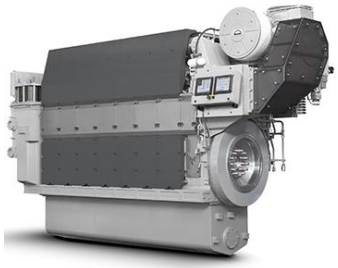
Motor de Propulsión MAN 7L27/38	
	Gama de potencia: 3192 BHP Gama de velocidad: 800 rpm

Tabla 8.15 – Características del motor MAN 7L27/38.[38]

A partir de estos motores, el motor seleccionado finalmente y que **incorporará** el buque será el **motor Caterpillar 3516E TIER 4 FINAL**, ya que cumple las especificaciones de emisiones descritas en la normativa internacional, y dentro de las diferentes características, es el motor que mejor se adapta a la potencia necesaria, esto también significará que su peso y volumen será menor que el resto de los motores, por lo que es otra ventaja que se debe tener en cuenta siempre que sea posible.

8.2 Alargamiento del buque

Una vez seleccionado el motor para estas condiciones, se puede realizar un alargamiento del buque para que la pérdida de velocidad no sea excesivamente grande, esto es aproximadamente 0.5 nudos.

Para ello, es necesario estimar la eslora y el volumen de carena que se debe añadir al buque, sabiendo que el alargamiento se produce en la zona central, o cuerpo cilíndrico, del buque, esto es, donde la sección se mantiene constante. Esto modificará las características del buque inicial.

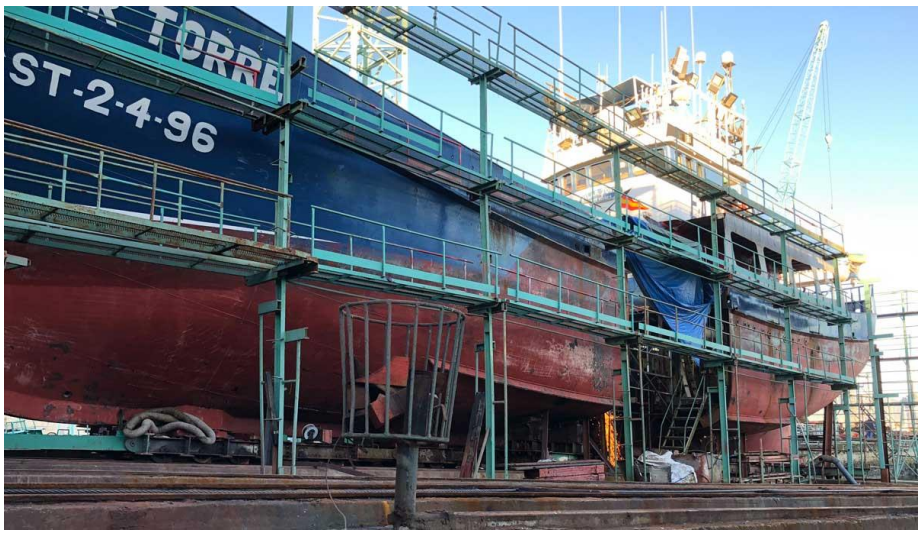


Figura 8.15 - Alargamiento de un buque en astillero.[39]

Inicialmente, el aumento en el desplazamiento del buque será conocido, ya que depende principalmente de la eslora. Podemos dividir el desplazamiento en: Peso Muerto (PM) y Peso en Rosca (PR), de forma que:

$$\delta\Delta = \delta PR + \delta PM \qquad (8.8)$$

[38] «File:Four-Stroke L27/38 - Profile|MAN».
[39] «File:Alargamiento buque Pilar Torre.png».

$$\Delta' = \Delta + \delta\Delta = \Delta + \delta PR + \delta PM = PR' + PM' \qquad (8.9)$$

La modificación en el peso en rosca se origina principalmente por el peso por metro lineal de la estructura del cuerpo cilíndrico. En este proyecto se empleará la fórmula empleada en el libro *"El proyecto básico del buque mercante"* que nos permite aproximar el nuevo peso en rosca de un buque tipo Ro – Ro.

$$PR'[t] = 0.03371 * L_{pp}^{1.5} * B * D^{0.5} + 0.59996 * MCO^{0.69965} + 9.5 * \left(\frac{MCO}{N}\right)^{0.84999} + 0.03791 * L_{pp} * B * D \qquad (8.10)$$

Donde MCO es la potencia máxima continua del motor propulsor en kW y N las revoluciones por minuto.

De esta forma, podemos calcular la variación del Peso en Rosca por metro lineal sabiendo que el valor de la manga es de 16.5 m y el puntal es de 11.0 m:

$$\delta PR = PR'(L_{pp} = 91m) - PR'(L_{pp} = 90m) \qquad (8.11)$$

$$\delta PR = 33204.79\ [kg] * \frac{\delta L\ [m]}{1\ [m]} \qquad (8.12)$$

En el caso de que esta estimación sobreestime de forma excesiva el peso por metro lineal real de la estructura, se podrá aprovechar para aumentar el Peso Muerto del buque.

Para la estimación del Peso Muerto, se comprueba la condición de carga más desfavorable, es decir, aquella que suponga que el peso por metro de carril sea mayor, para ello, se debe tener en cuenta el margen entre la carga para que pueda ser anclada de forma correcta. Estimando el peso medio de cada tipo de carga que transportará nuestro buque, podemos definir las siguientes distribuciones:

Tipo de Carga	Longitud + Margen [m]	Peso medio [kg]	Peso/L. Carril [kg/m]
Roll Tráiler 20'	6.40	21000.0	3281.250
Roll Tráiler 40'	12.40	30800.0	2483.871
Camión Rígido	14.00	20000.0	1428.571

Tabla 8.16 – Distribución del peso medio por longitud de carril de la carga.

Por tanto, la carga más desfavorable se dará con los Roll Tráiler de 20 pies, para el caso de la carga en bodega y entrepuentes. Sabiendo que en bodega se disponen de 5 carriles de carga, el peso final por metro de eslora será de:

$$\delta PM_{Roll\ Trailer\ 20'} = 3281.250\ \left[\frac{kg}{m}\right] * 5\ [carriles] = 16406.250\ \left[\frac{kg}{m}\right] \qquad (8.13)$$

A continuación, debemos realizar un estudio sobre la cantidad de contenedores sobre cubierta que se podrán añadir, manteniendo una distancia como se ha comentado anteriormente para el anclaje cada dos filas columnas de contenedores. Las distancias de seguridad vienen determinadas generalmente por las Sociedades de Clasificación, que para este caso recomiendan un margen de 0.5 m.

Las hileras de contenedores que podemos tener en manga son 6, por lo que podemos aproximar el Peso Muerto añadido por metro a partir de la siguiente expresión:

$$\delta PM_{TEU} = \frac{(20320.0\ [kg] * 0.7 * 6\ [filas] * 2\ [columnas] * 2\ [alturas])}{6\ [m] * 2 + 0.5\ [m]} = 27310.080\ \left[\frac{kg}{m}\right] \qquad (8.14)$$

De esta forma, podemos aproximar el Peso Muerto añadido total por metro de eslora, sumando las dos últimas expresiones, obteniendo la siguiente fórmula:

$$\delta PM = [16406.250 * 2 + 27310.080\ ] * \delta L = 60122.580 * \delta L \qquad (8.15)$$

Por tanto, si estimamos distintos alargamientos del buque, podremos obtener la potencia necesaria y ver si el motor seleccionado es compatible con el alargamiento para cumplir con la condición de pérdida de velocidad, ya que el objetivo es mantener el rendimiento máximo del motor. Obteniendo las nuevas características hidrostáticas (calado, coeficientes hidrostáticos, posición longitudinal del centro de carena, etc.) a partir del volumen de carena de las curvas correspondientes del buque e introduciéndolas en el programa, obtenemos los siguientes resultados para una velocidad del buque de 13.3 nudos:

Alargamiento [m]	Δ [t]	Resistencia [Kg]	EHP [CV]	Revoluciones [rpm]	Formula de Lap	BHP [CV]
3.2	6176.029	14730.1397	1343.8179	150	0.7068	2825.5590
6.4	6474.676	15020.9558	1370.3488	150	0.7039	2893.1322
9.6	6773.324	15280.3232	1394.0106	150	0.7011	2954.9846
12.8	7071.972	15505.892	1414.5891	150	0.6983	3010.6025
16.0	7370.619	15748.969	1436.7648	150	0.6955	3069.9022
19.2	7669.267	16038.0403	1463.1365	125	0.7248	2999.7816
22.4	7967.914	16384.9052	1494.7807	125	0.7226	3074.1173
25.6	**8266.562**	**16775.5159**	**1530.4158**	**125**	**0.7204**	**3157.0109**
28.8	8565.209	17126.2508	1562.413	125	0.7182	3232.7602

Tabla 8.17 – Obtención de la nueva potencia necesaria para distintos alargamientos.

Por tanto, se llega a la conclusión de que el alargamiento más grande permitido sin necesidad de modificar el motor instalado se da para 25.6 m, teniendo en cuenta que las BHP mostradas en la tabla anterior se sobredimensionan con el mismo factor que anteriormente se ha comentado, para no sobrecargar el motor seleccionado.

8.3 Cálculo del barco viga

Para este caso, tendremos la siguiente estructura en la interfaz gráfica, donde podemos ver los datos necesarios que se deben introducir para realizar el cálculo preliminar de esfuerzos cortantes y momentos flectores.

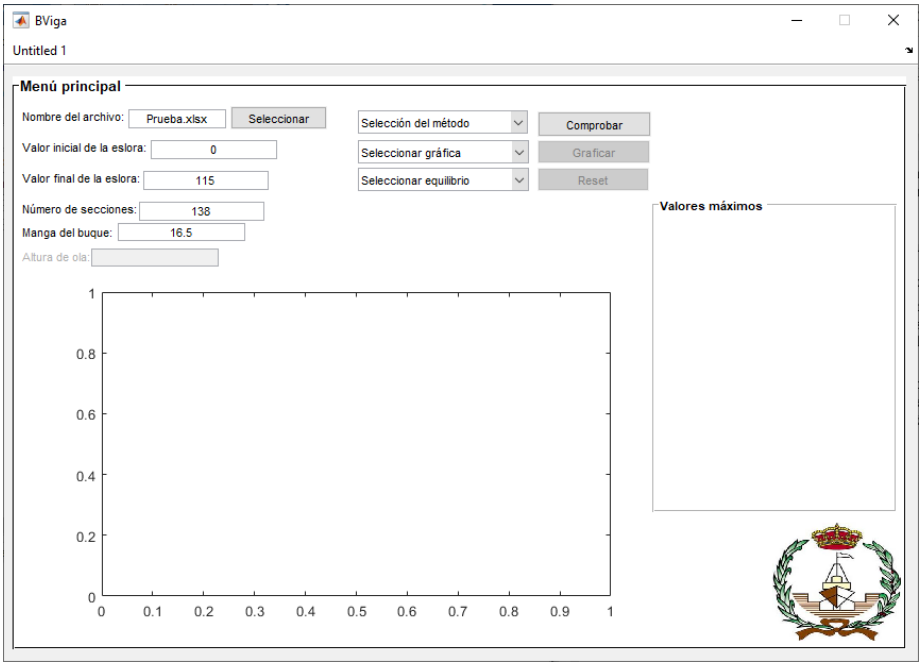


Figura 8.16 – Interfaz gráfica para el cálculo de Buque-Viga.

En este proyecto, se ha decidido obtener de forma directa la curva de cargas a partir de una hoja Excel donde se localice la curva de pesos del buque, el programa realizado trabajará con aquellos datos que se sitúen en la pestaña del archivo con nombre *"Peso"* y que ocupen las dos primeras filas, correspondientes a las secciones y pesos. Estos datos se pueden modificar de forma sencilla en el código en el caso de que sea necesario.

En caso de conocer la curva de empuje del buque, en equilibrio, es posible introducirla también si modificamos el código de nuevo, ya que de esa forma es posible obtener una mejor aproximación a la solución real.

La programación se ha basado en realizar funciones que podamos emplear en cualquier ocasión, para facilitar su uso y comprensión, así como reducir las líneas de código. De esta forma también se facilita la corrección al detectar errores durante la realización de este proyecto de forma escalonada.

Como se puede observar en la figura anterior, este programa nos permite realizar distintas configuraciones con las siguientes opciones:

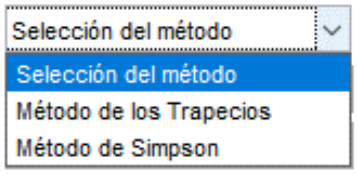


Figura 8.17 – Selección del método a aplicar.

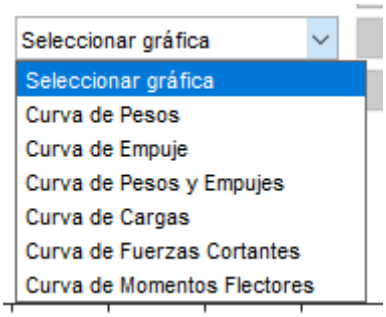


Figura 8.18 – Selección del cálculo y gráfica a realizar.

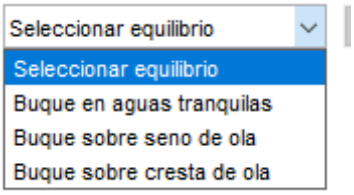


Figura 8.19 – Selección del equilibrio a estudio.

Otra de las utilidades del programa es que nos permite seleccionar de forma sencilla entre los archivos Excel que se encuentran en la carpeta del programa, como se muestra en la siguiente figura:

Figura 8.20 – Ventana de selección entre archivos Excel presentes en la carpeta del programa.

8.3.1 Aplicación a un ejemplo como base del programa

Inicialmente, para el planteamiento de los diversos programas, se propuso como ejercicio base del proyecto, un ejercicio correspondiente a la asignatura de *"Diseño y Cálculo de Estructuras Navales"* que trata de una barcaza paralepipédica, con una eslora de 80 m, una manga de 10 m, y un puntal de 8 m, a partir del peso en rosca que es igual a 1600 t distribuidas de forma uniforme a lo largo de la eslora, y el peso muerto, igual a 1200 t distribuido uniformemente en las bodegas más a popa y proa de la barcaza, sabiendo que se divide en cuatro compartimentos equiespaciados. De esta forma obtenemos:

L [m]	0	5	10	15	20	20	25	30	35	40	45	50	55	60	60	65	70	75	80
Peso/m	-50	-50	-50	-50	-50	-20	-20	-20	-20	-20	-20	-20	-20	-20	-50	-50	-50	-50	-50

Tabla 8.18 – Distribución del peso en una barcaza.

El problema se estudia para el caso de encontrarse en equilibrio sobre el seno de una ola senoidal de 3 m de altura y 80 m de longitud, de forma que coincide con la eslora del buque, que como se ha comentado en apartados anteriores, es la situación más desfavorable.

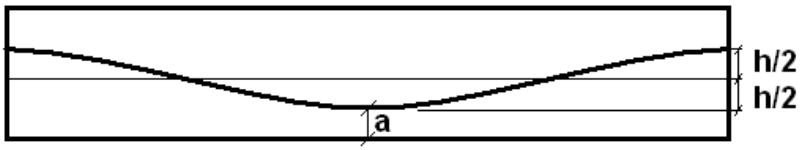


Tabla 8.19 – Representación gráfica del desplazamiento.

Para calcular el empuje por metro del buque, simplemente es necesario plantear la ecuación de una onda. Para este caso obtenemos la siguiente ecuación:

$$Ec.\,ola\ senoidal\ con\ seno\ en\ el\ centro = a + \frac{h}{2} + \frac{h}{2} * sen\left(\frac{2 * \pi * x}{L} + \frac{\pi}{2}\right) \qquad (8.16)$$

Sabiendo que el empuje es igual al peso, podemos obtener el valor de a, correspondiente al calado mínimo del buque, despejando de forma sencilla:

$$Peso = Empuje \tag{8.17}$$

$$Peso\ Rosca + Peso\ Muerto = \left(a + \frac{h}{2}\right) * L * B \tag{8.18}$$

$$a = \frac{PR + PM - \frac{h}{2} * L * B}{L * B} \tag{8.19}$$

Conocido este valor, podemos sustituirlo en la expresión (8.16) que multiplicamos por el valor de la manga para tener la carga por metro, y no una presión por metro cuadrado, de forma que la ecuación para el empuje será:

$$Empuje/metro = \left(\frac{PR + PM - \frac{h}{2} * L * B}{L * B} + \frac{h}{2} + \frac{h}{2} * sen\left(\frac{2 * \pi * x}{L} + \frac{\pi}{2}\right)\right) * B \tag{8.20}$$

Una vez que tenemos la curva de empujes y pesos, como se ha visto anteriormente, calculamos la curva de cargas total del buque con la que se trabajará para obtener las fuerzas cortantes y momentos flectores.

Inicialmente el problema se resuelve mediante una hoja Excel, obteniendo las siguientes gráficas de resultados mediante la integración por Simpson:

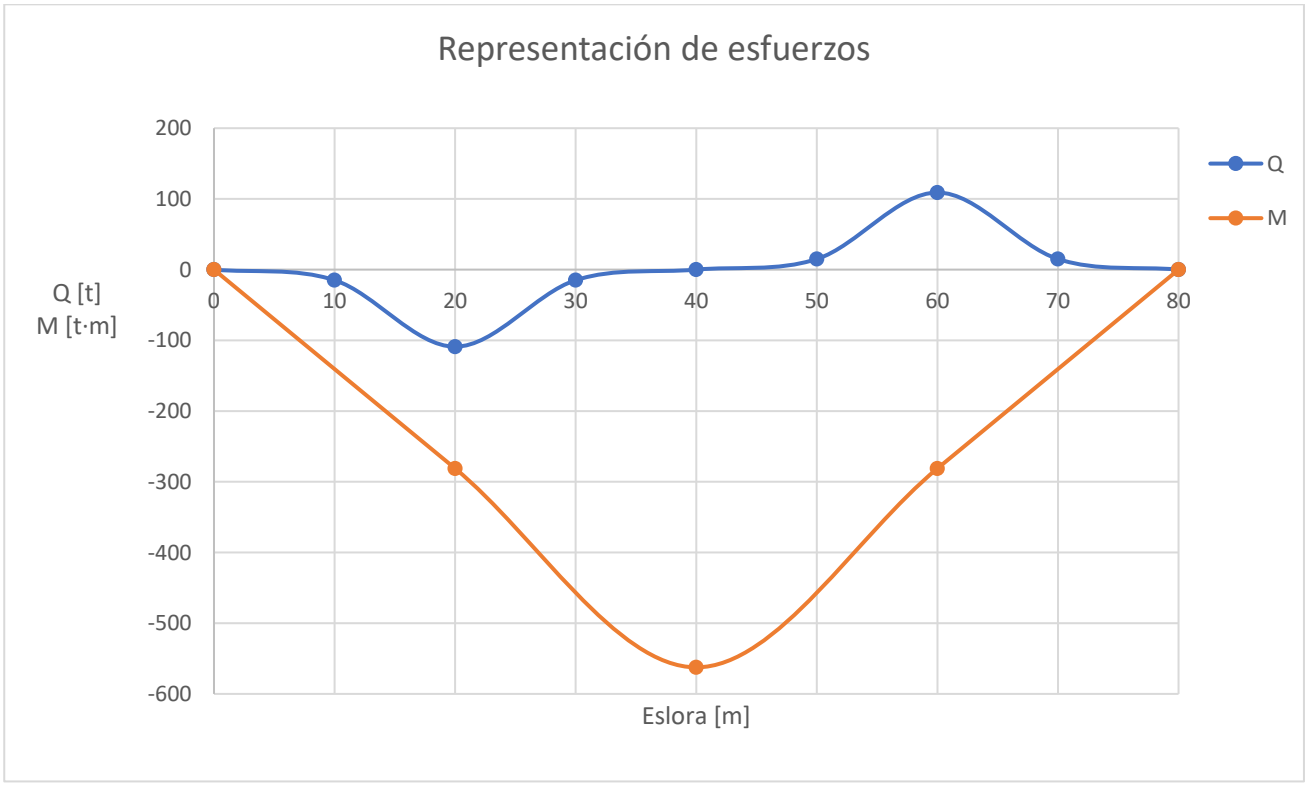


Figura 8.21 – Representación de la curva de Fuerzas Cortantes para el ejemplo.

Sección	Q [t]	Sección	M [t·m]
0	0	0	0
10	-14.934	20	-281.210
20	-108.988	40	-562.420
30	-14.934	60	-281.210
40	0	80	0
50	14.934		
60	108.988		
70	14.934		
80	0		

Tabla 8.20 – Resultados obtenidos para la integración por Simpson del ejemplo.

Por tanto, introduciendo los datos correspondientes para este ejemplo, podemos obtener las siguientes gráficas:

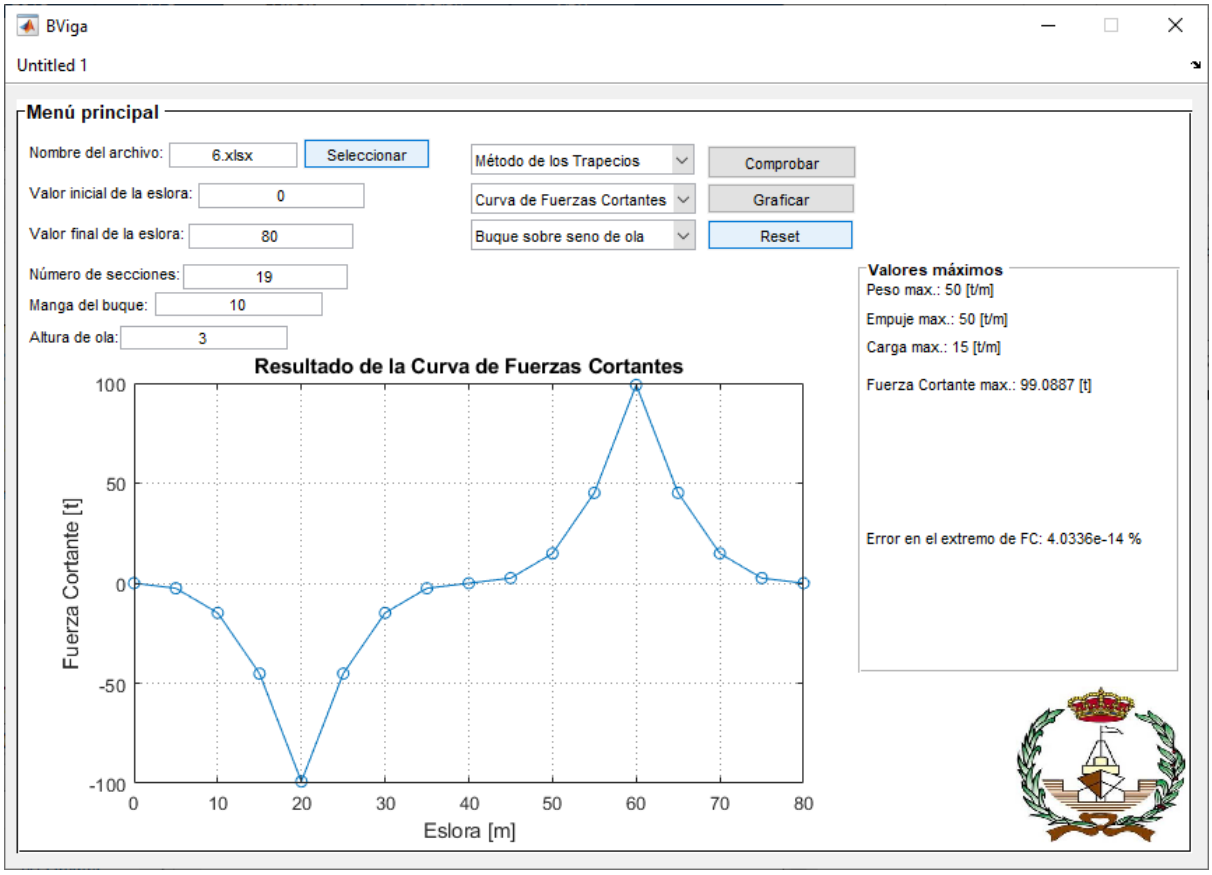


Figura 8.22 – Resultados obtenidos mediante el método de los trapecios para las Fuerzas Cortantes del ejemplo.

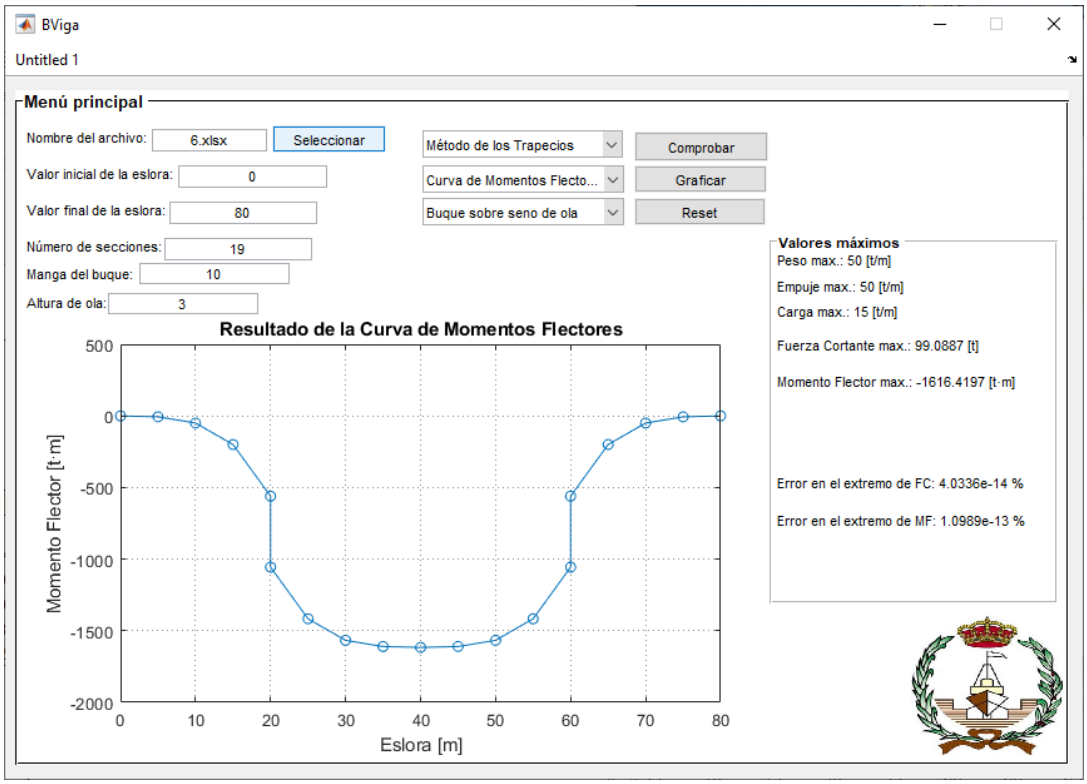


Figura 8.23 – Resultados obtenidos mediante el método de los trapecios para los Momentos Flectores del ejemplo.

Para el caso de aplicar el método de Simpson, para este ejemplo se puede aplicar únicamente la primera regla de 1/3, ya que el número de secciones que con las que trabajamos son pares, como veremos posteriormente, esto se ve condicionado al número de secciones con las que trabajemos, para obtener el menor error posible, realizando una combinación de las dos reglas de Simpson. Por tanto, tenemos:

Figura 8.24 – Resultados obtenidos mediante el método de Simpson para las Fuerzas Cortantes del ejemplo.

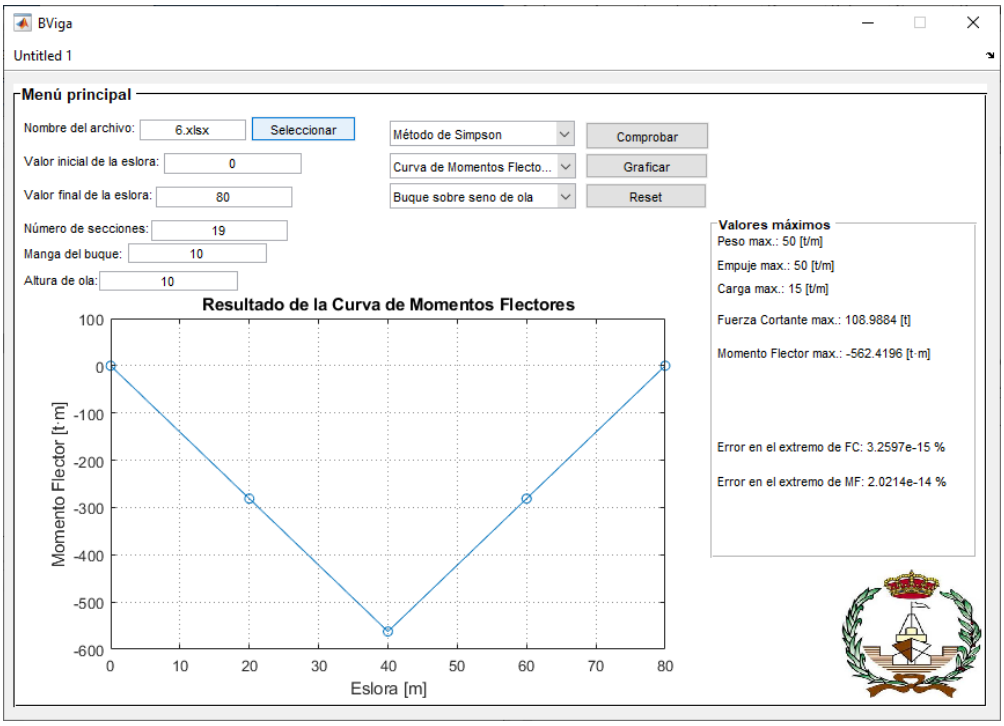


Figura 8.25 – Resultados obtenidos mediante el método de Simpson para los Momentos Flectores del ejemplo.

Podemos ver que, a pesar de representar menos puntos de forma gráfica, el error que obtenemos mediante este método frente al de los trapecios es menor, y que el error cometido se puede considerar como el error generado por la propia máquina al realizar las aproximaciones:

Curva	Método de los Trapecios	Método de Simpson
Fuerzas Cortantes	4.0336E-14	3.2597E-15
Momentos Flectores	1.0989E-13	2.0214E-14

Figura 8.26 – Errores cometidos en los distintos métodos para el ejemplo considerado.

Para calcular este error cometido, comparamos el valor obtenido en el extremo con respecto del valor máximo de la curva, ya que este tipo de errores pueden llegar a ser aceptables dentro de un margen de aproximadamente un 10% debido a errores como puede ser el de truncamiento.

8.3.2 Aplicación al buque de estudio

El objetivo principal de este trabajo se basa en la aplicación del cálculo de buque viga a cualquier tipo de buque considerado, únicamente a partir de una curva de cargas. Para dar una aplicación práctica y evaluar la viabilidad del programa, se proporciona un buque Ro-Ro con los siguientes valores característicos:

Datos	Valor	Unidad
Eslora Total	127.1	m
Eslora entre Perpendiculares	115.6	m
Manga	16.5	m
Calado de proyecto	5.45	m
Puntal	11	m

Tabla 8.21 - Características principales del buque.

Elemento	Cant.	Peso (t)	L_G (m)	T_G (m)	V_G (m)
Acero continuo	1	2633.911	53.9	0	7.292
Carga Bodeguín	1	1170	64.05	0	2.75
Carga Entrepuente	1	1944	55.05	0	8.2
Carga Cub. Superior	1	1140	52.4	-0.165	12.9
Nº 1 Rasel de Proa	1	96.57	112.837	0	4.112
Nº 2 Er. Trimado	1	135.942	103.86	1.943	4.38
Nº 2 Br. Trimado	0	0	101.03	0	0.009
Nº 3 Tanque Lastre	0	0	96.839	0	0.007
Nº 4 Er. Tanque Lastre	0	0	72.226	2.802	0.017
Nº 4 Br. Tanque Lastre	0	0	72.226	-2.802	0.017
Nº 5 Er. Tanque Lastre	0	0	72.771	0.016	0
Nº 5 Br. Tanque Lastre	0	0	72.771	-0.016	0
Nº 4A Er. Tanque Lastre	0	0	48.231	4204	0.01
Nº 4A Br. Tanque Lastre	0	0	48.231	-4204	0.01
Nº 5A Er. Tanque Fuel-Oil	1	134.484	59.318	2.098	0.727
Nº 5A Br. Tanque Fuel-Oil	1	134.484	59.318	-2.098	0.727
Nº 6 Er. Tanque Lastre	0	0	45.655	4.205	0.01
Nº 6 Br. Tanque Lastre	0	0	45.655	-4.205	0.01
Nº 7 Er. Tanque Fuel-Oil	1	47.472	38.282	2.072	0.389
Nº 7 Br. Tanque Fuel-Oil	1	87.995	38.198	-2.086	0.737
Nº 8 Tanque Diesel-Oil	1	49.484	26.429	0	0.817
Nº 9 Er. Tanque Adrizamiento	0	0	27.501	5.123	1450
Nº 9 Br. Tanque Adrizamiento	0	0	27.501	-5.123	1450
Nº 10 Derrames	0	0	22.712	-1.4	0.16
Nº 11 Residuos	0	0	22.712	1.4	0.16
Nº 12 Er. Agua Dulce	1	16.542	18.051	2.406	1.73
Nº 12 Br. Agua Dulce	1	16.542	18.051	-2.406	1.73
Nº 13 Aceite Retorno	1	5.852	18.86	0	1.066
Nº 14 Aceite Sucio	1	5.491	18.906	0	0.378
Nº 15 Aceite Bocina	1	1.976	12.932	0	1.495
Nº 16 Tanque Lastre	0	0	6.307	0	0.018
Nº 17 Er. Tanque Lastre	0	0	5.828	0.643	3.43
Nº 17 Br. Tanque Lastre	0	0	5.828	-0.643	3.43
Nº 18 Er. Tanque Lastre	0	0	0	0	5.155
Nº 18 Br. Tanque Lastre	0	0	0	0	5.155
Nº 19 Er. Agua Dulce	1	13.489	104.03	6.116	8336
Nº 19 Br. Agua Dulce	1	13.489	104.03	-6.116	8.336
Nº 20 Servicio Diario	1	26.485	28.05	0	3.5
Nº 21 Sedimentación	1	35.65	28.58	0	3.9
Provisiones	1	5	104.6	0	10.5
Cargos-Fonda	1	3	102.22	0	13
Cargos-Maquinas	1	8	109.6	0	8
Cargos y equipo	1	14	78.01	0	8.94
Tripulantes-Efectos	1	3	98.02	0	13.5
Total Loadcase		**7742.56**	**56.776**	**-0.001**	**7.131**
FS correction					0
VCG fluid					7.131

Tabla 8.22 - Distribución de pesos en el buque.

De esta forma, tenemos definidas todas las cargas del buque, con sus posiciones en eslora gracias a los planos de forma y planos de distribución general que se anexan a este documento.

Debido principalmente al problema general que supone plantear un tipo de problema como éste, se encuentra la dificultad añadida de plantear los distintos casos que consideren las formas del buque con mayor o menor precisión para definir una distribución que proporcione mayor precisión en los resultados obtenidos. Como se ha visto anteriormente, la distribución puede ser lineal, triangular o trapezoidal, nos centraremos en la primera.

A continuación, el siguiente paso que se realiza es definir el espaciado que se desea realizar entre las secciones consideradas, que no coincidirán con las secciones del buque en el plano de formas. Como se ha visto en el ejemplo anterior, para un espaciado de 5 m se obtiene una curva muy poco definida, por tanto, se plantea la realización de un espaciado de 1 m, de esta forma podemos obtener una curva de cargas por metro de eslora mucho más definida.

Generalmente, cada 5 secciones, esta última se repite para poder definir la carga a un lado y al otro de dicha sección, debido a que podemos encontrar distintos tanques llenos o vacíos, es por esto por lo que al determinar el número de secciones consideradas en el buque se especificarán 138 en lugar de 115.

8.3.3 Condición de equilibrio para aguas tranquilas

Para las hipótesis de una distribución lineal podemos obtener la siguiente curva de pesos en la condición de equilibrio en aguas tranquilas:

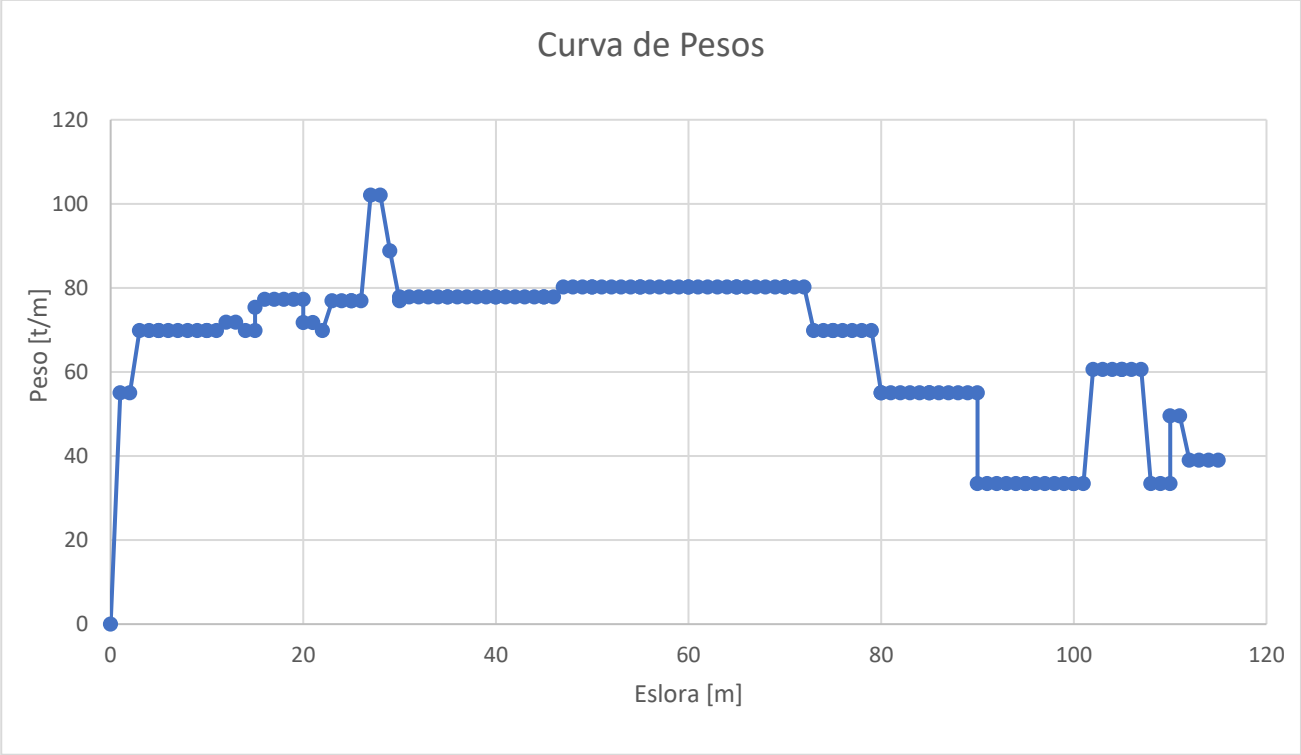


Figura 8.27 – Representación gráfica de los pesos del buque para una distribución lineal.

Ésta será la configuración con la que trabajará el programa para realizar los cálculos que se les especifiquen. Se selecciona esta condición de forma inicial, ya que al plantear

las soluciones mediante una hoja de cálculo, se obtienen mejores resultados frente a la hipótesis de una distribución triangular y trapezoidal.

Inicialmente calculamos mediante el método de los trapecios, la curva de fuerzas cortantes y la curva de momentos flectores en la condición de aguas tranquilas, donde obtenemos los siguientes resultados:

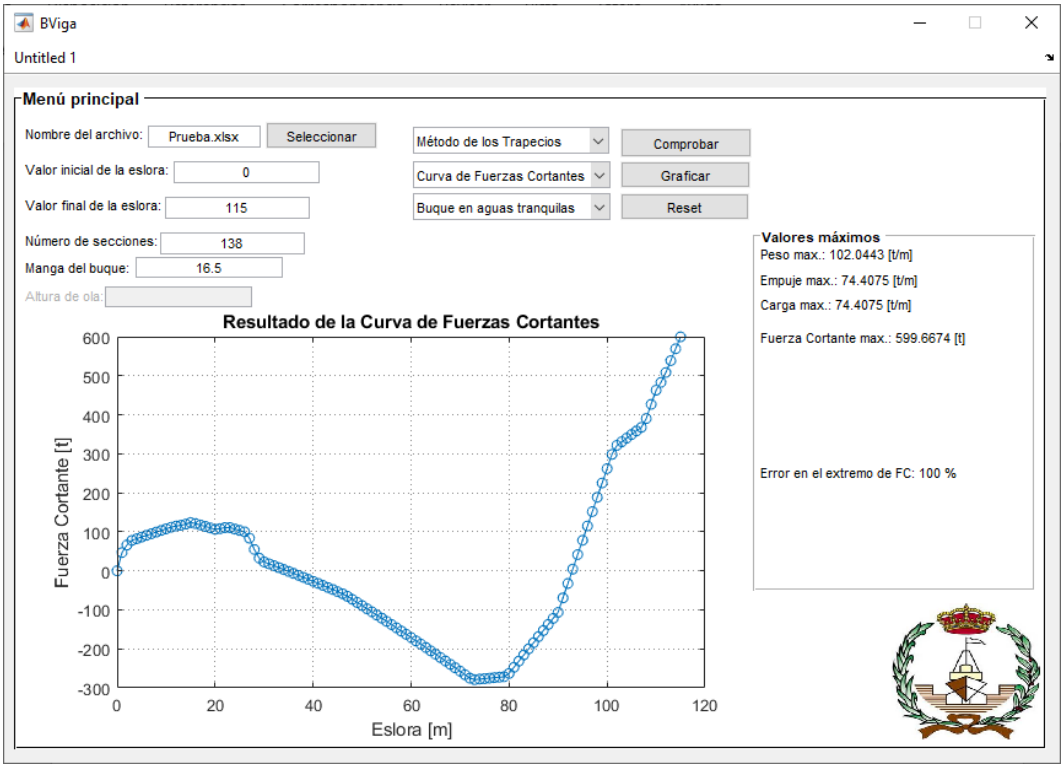


Figura 8.28 – Resultados para la curva de Fuerzas Cortantes mediante el método de los trapecios.

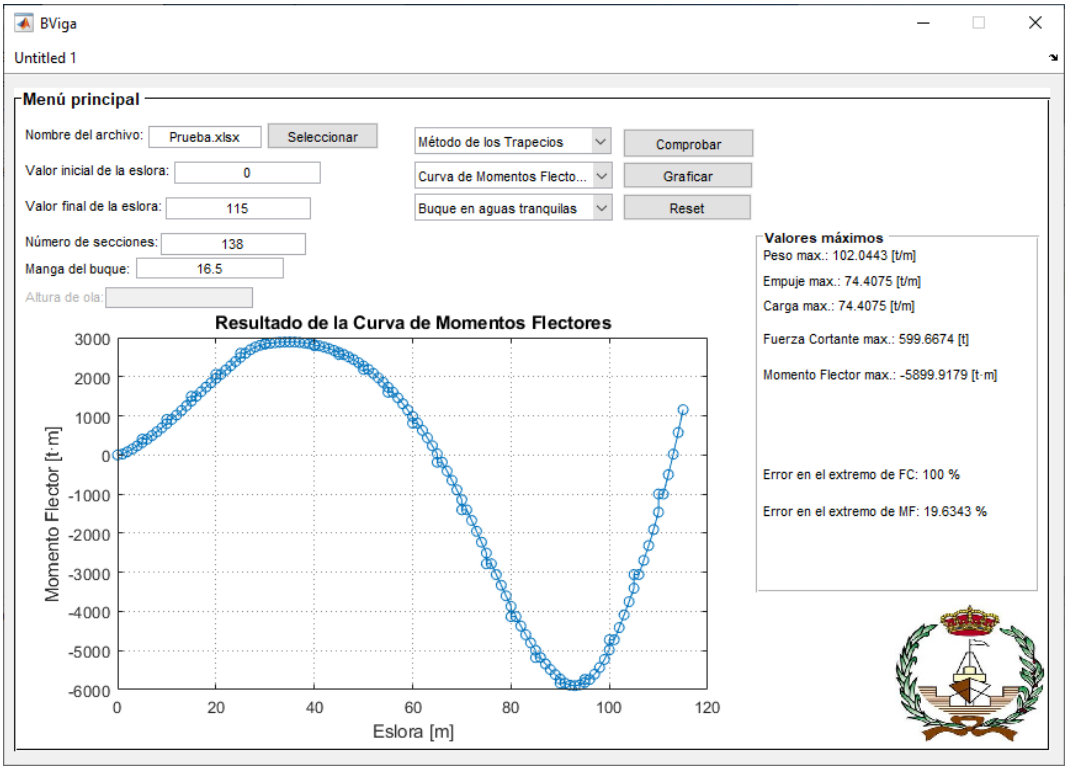


Figura 8.29 – Resultados para la curva de Momentos Flectores mediante el método de los trapecios.

Se puede ver de forma clara las distintas formas de las gráficas, cuando la fuerza cortante se hace cero, se observa que el momento flector tiene un máximo o mínimo relativo, y que cuando la fuerza cortante se hace máxima, el momento flector se hace cero.

El estudio más crítico del buque se basa en el estudio de los momentos flectores a los que se encuentra sometido, ya que éste define el espesor de la plancha que se deberá de instalar en el buque, por tanto, se ha priorizado la optimización de la curva de momentos flectores frente a la de esfuerzos cortantes.

Si comparamos los errores con los obtenidos al realizar el mismo procedimiento mediante una hoja de Excel, obtenemos los siguientes errores:

Errores		
Curva	**Excel**	**Matlab**
Fuerzas Cortantes	100%	100%
Momentos Flectores	20.6656%	19.6343%

Tabla 8.23 – Comparación de errores entre programas para el método de los trapecios.

Podemos ver una diferencia con los resultados obtenidos mediante la hoja de Excel, esto puede deberse principalmente a que el error de truncamiento es mayor que en MATLAB.

Si realizamos el mismo cálculo mediante el método de Simpson, obtenemos:

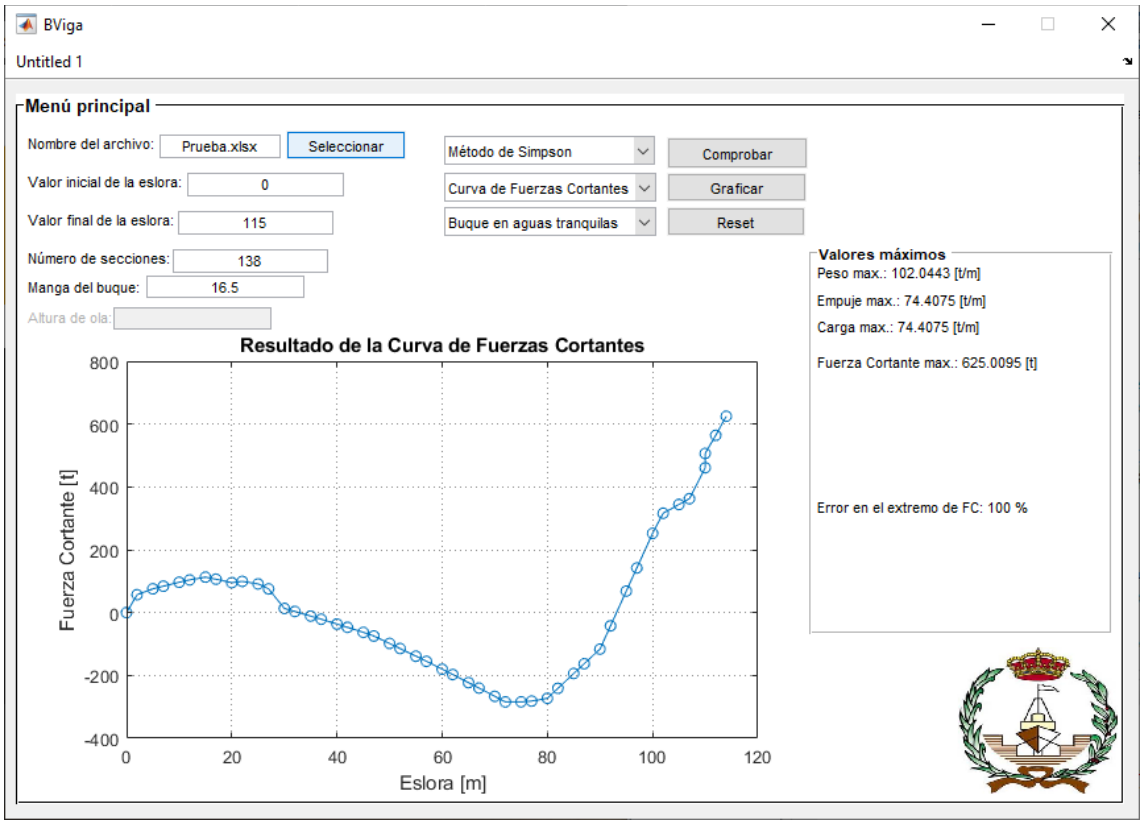


Figura 8.30 – Resultados para la curva de Fuerzas Cortantes mediante el método de Simpson.

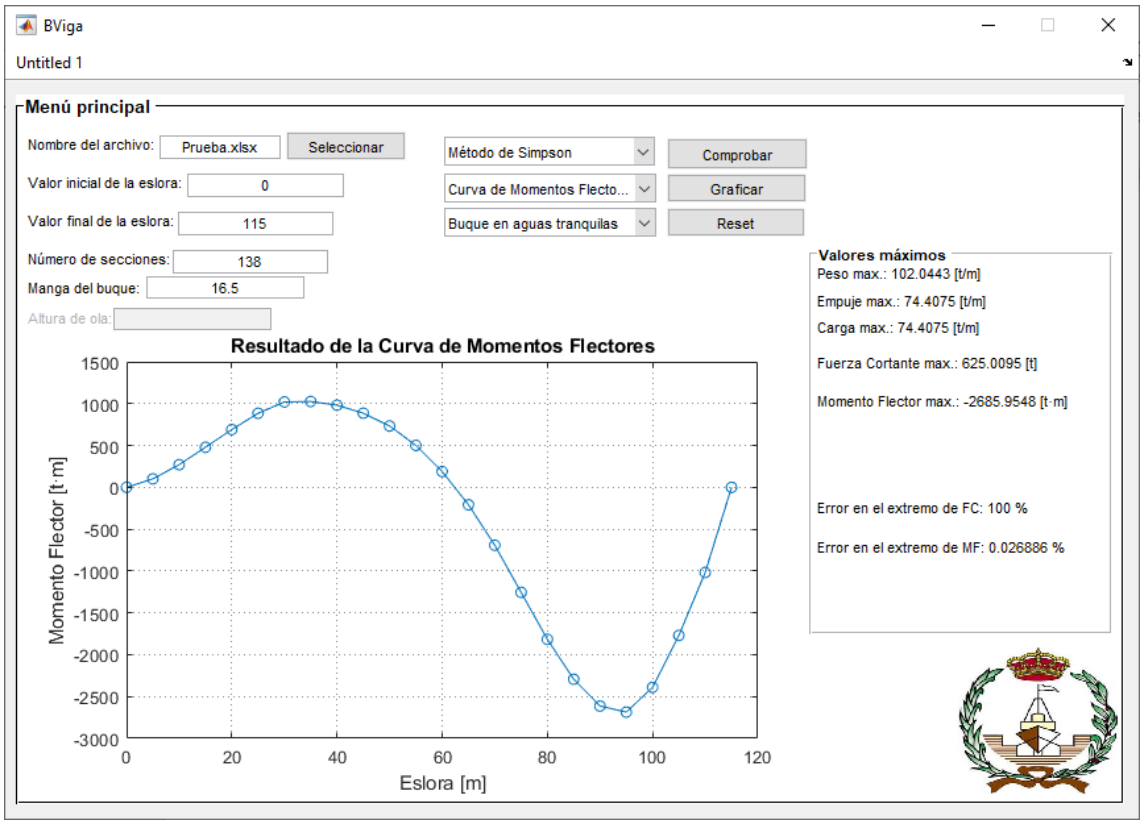


Figura 8.31 – Resultados para la curva de Momentos Flectores mediante el método de Simpson.

De nuevo, si se comparan con los obtenidos mediante hoja de cálculo, se puede apreciar de forma más clara la diferencia en la precisión del método empleado:

Errores		
Curva	**Excel**	**Matlab**
Fuerzas Cortantes	100%	100%
Momentos Flectores	0.2732%	0.0269%

Tabla 8.24 – Comparación de errores entre programas para el método de Simpson.

Si comparamos a continuación los resultados entre el error obtenido mediante el programa con los distintos métodos, se puede llegar a la conclusión de que el método de Simpson, a pesar de ser un recurso matemático más complejo, para casos como este en el que se desea definir el área bajo la curva de forma precisa, es mucho mejor.

Errores		
Curva	**Trapecios**	**Simpson**
Fuerzas Cortantes	100%	100%
Momentos Flectores	19.6343%	0.0269%

Tabla 8.25 – Comparación de errores entre métodos aplicados.

8.3.4 Condición de equilibrio en arrufo

Uno de los propósitos de este proyecto es evaluar este tipo de estudio a un buque con distintas situaciones de equilibrios en el mar, es por esto por lo que, en este apartado, nos centramos en el caso del buque en la situación de arrufo.

Para esta situación, se realiza un programa que obtenga de forma automática la curva de empuje para la parametrización de una ola senoidal con un seno en el centro del buque, dando un valor de 2 metros para la altura de la ola, se obtienen:

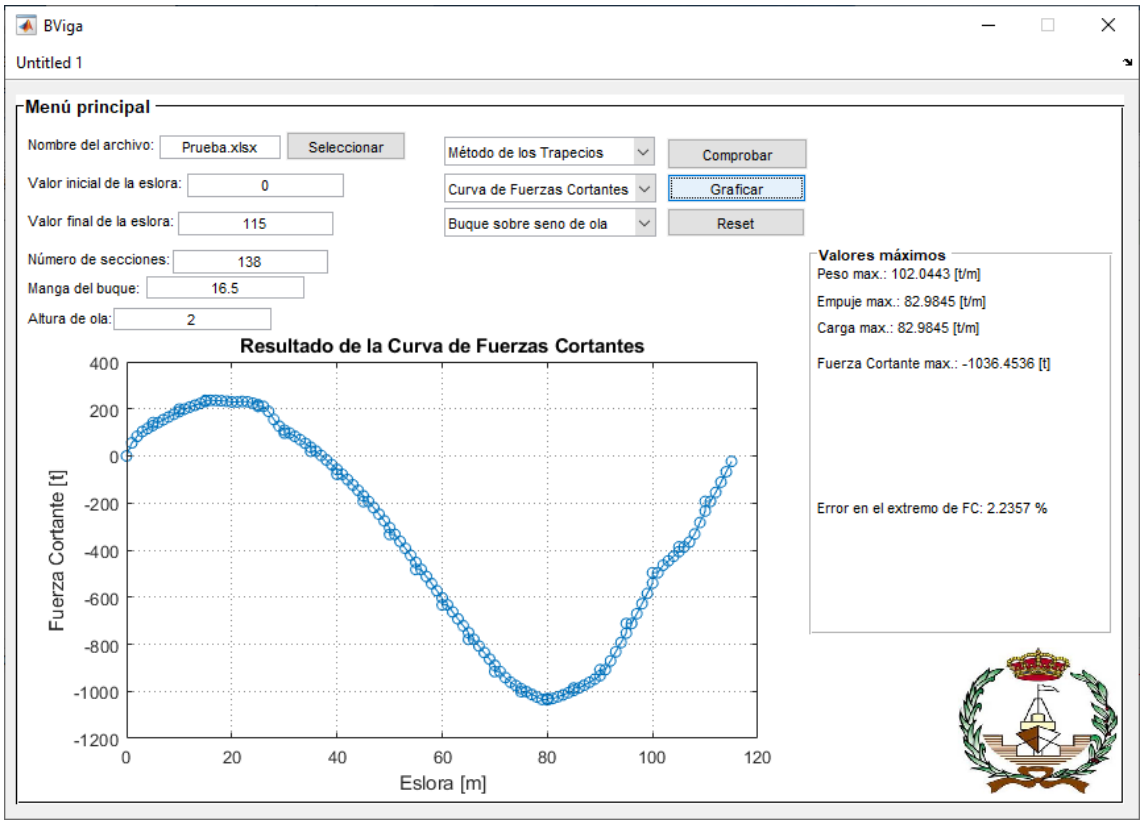


Figura 8.32 – Resultados para la curva de Fuerzas Cortantes mediante el método de los trapecios.

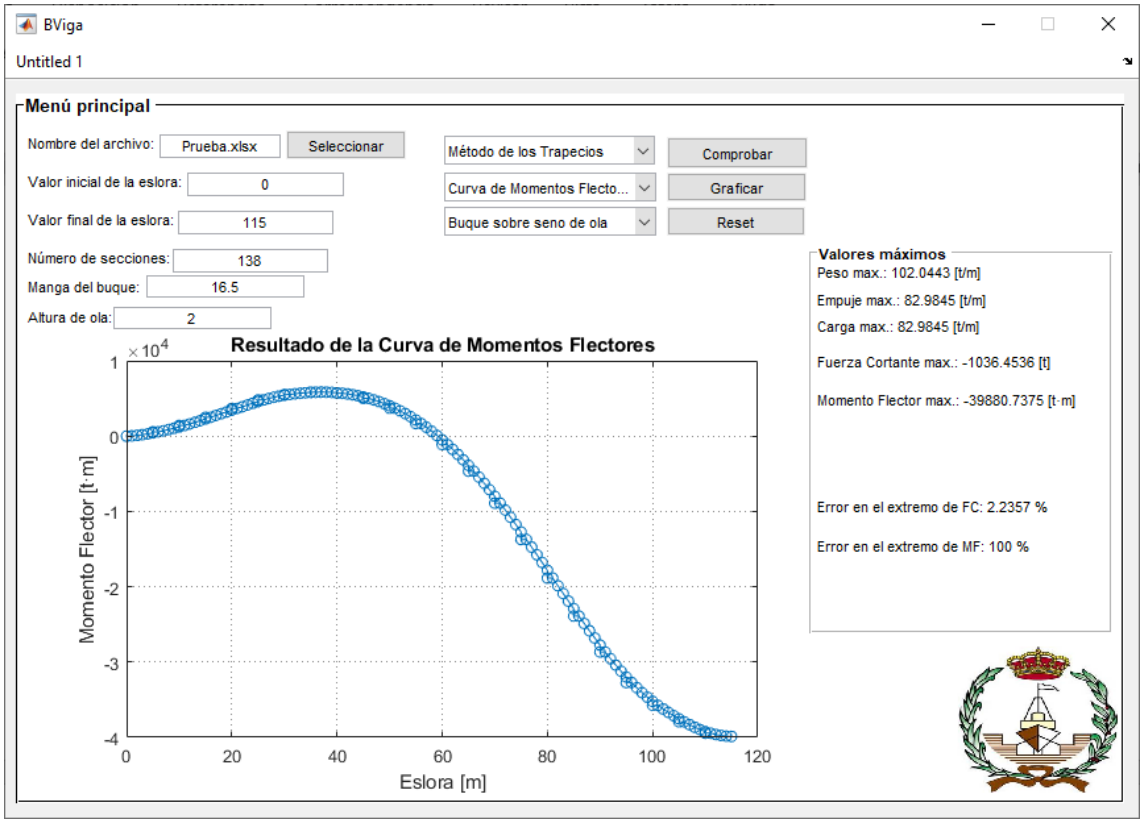


Figura 8.33 – Resultados para la curva de Momentos Cortantes mediante el método de los trapecios.

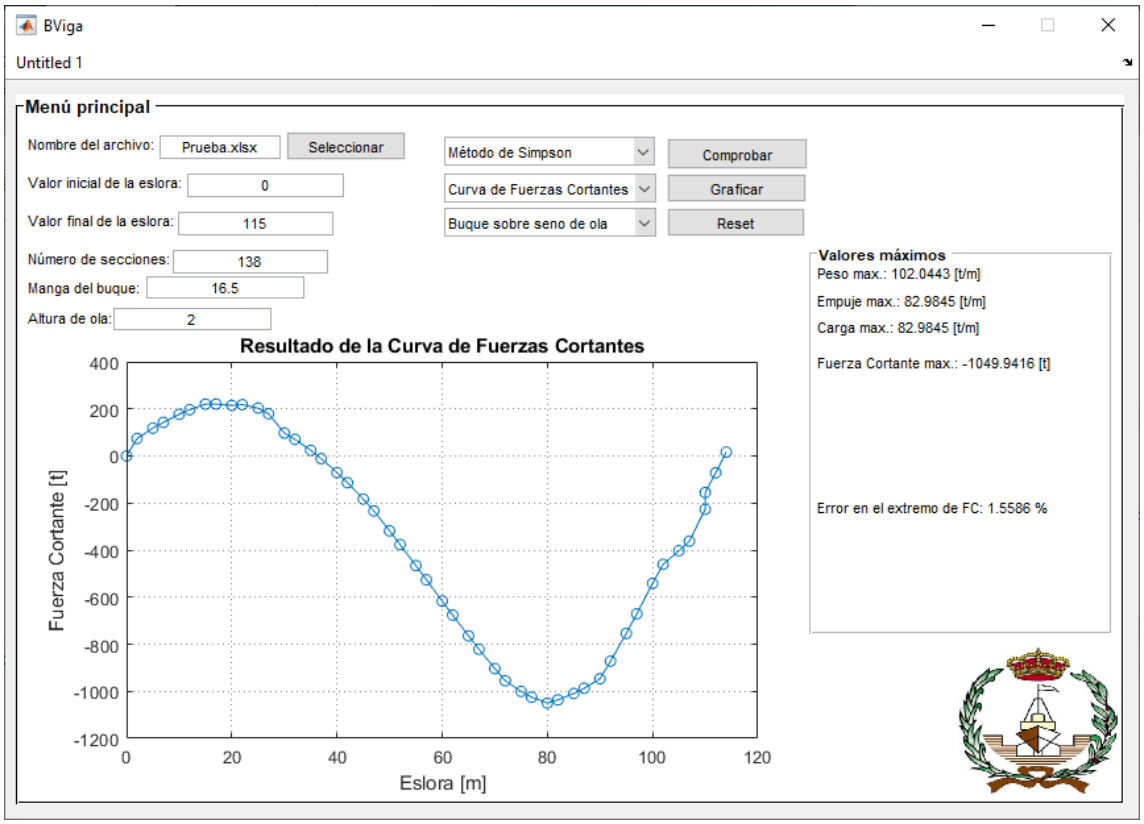


Figura 8.34 – Resultados para la curva de Fuerzas Cortantes mediante el método de Simpson.

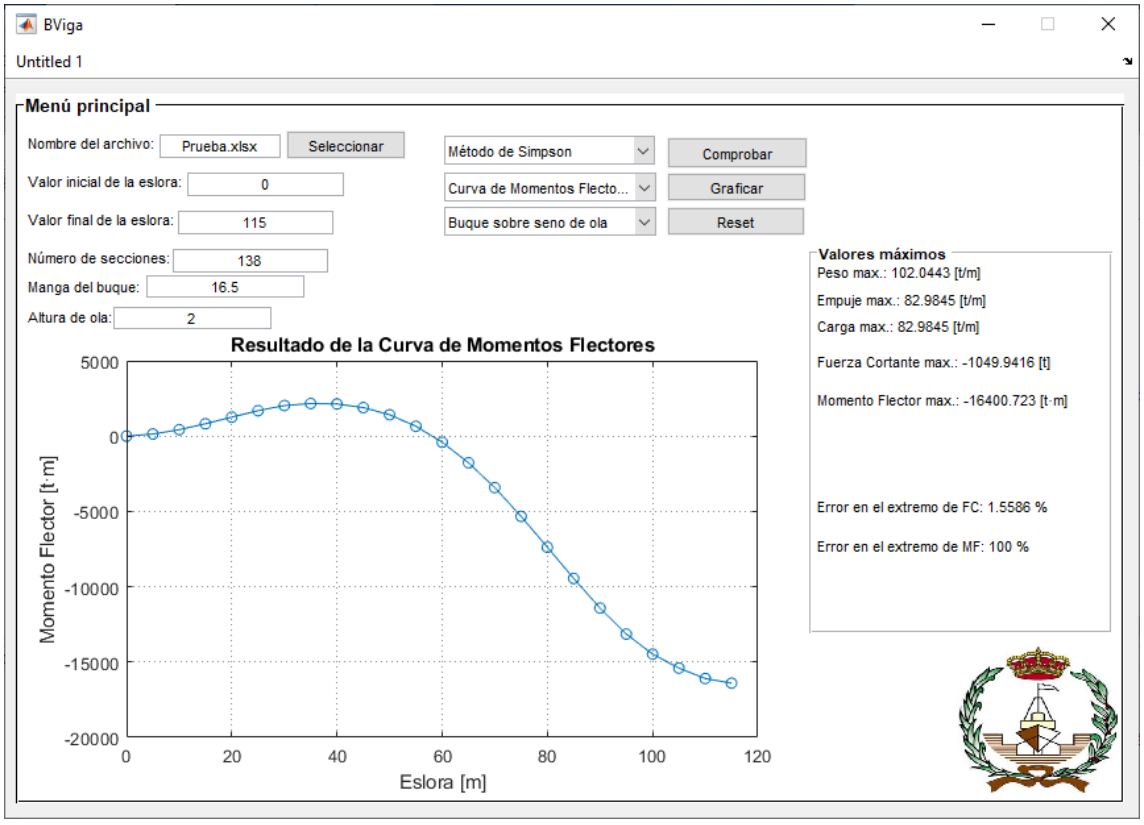


Figura 8.35 – Resultados para la curva de Momentos Cortantes mediante el método de Simpson.

Para este caso, se aprecia que estas curvas no están optimizadas para la curva de momentos flectores, dado que debido a los diversos problemas encontrados durante este proyecto no ha sido posible terminar esta, y la condición de equilibrio en quebranto.

A pesar de esto, podemos ver que el error obtenido para la curva de fuerzas cortantes es bastante aceptable.

Errores		
Curva	**Trapecios**	**Simpson**
Fuerzas Cortantes	2.2357%	1.5586%
Momentos Flectores	100%	100%

Figura 8.36 – Comparación de errores entre métodos aplicados.

8.3.5 Condición de equilibrio en quebranto

De la misma forma que en el apartado anterior, se calcula el desfase necesario con respecto de la ola anterior para obtener la situación de cresta en la sección media. Aplicando los programas correspondientes que se encuentran anexados, se vuelven a obtener las curvas representadas gráficamente con el siguiente resultado:

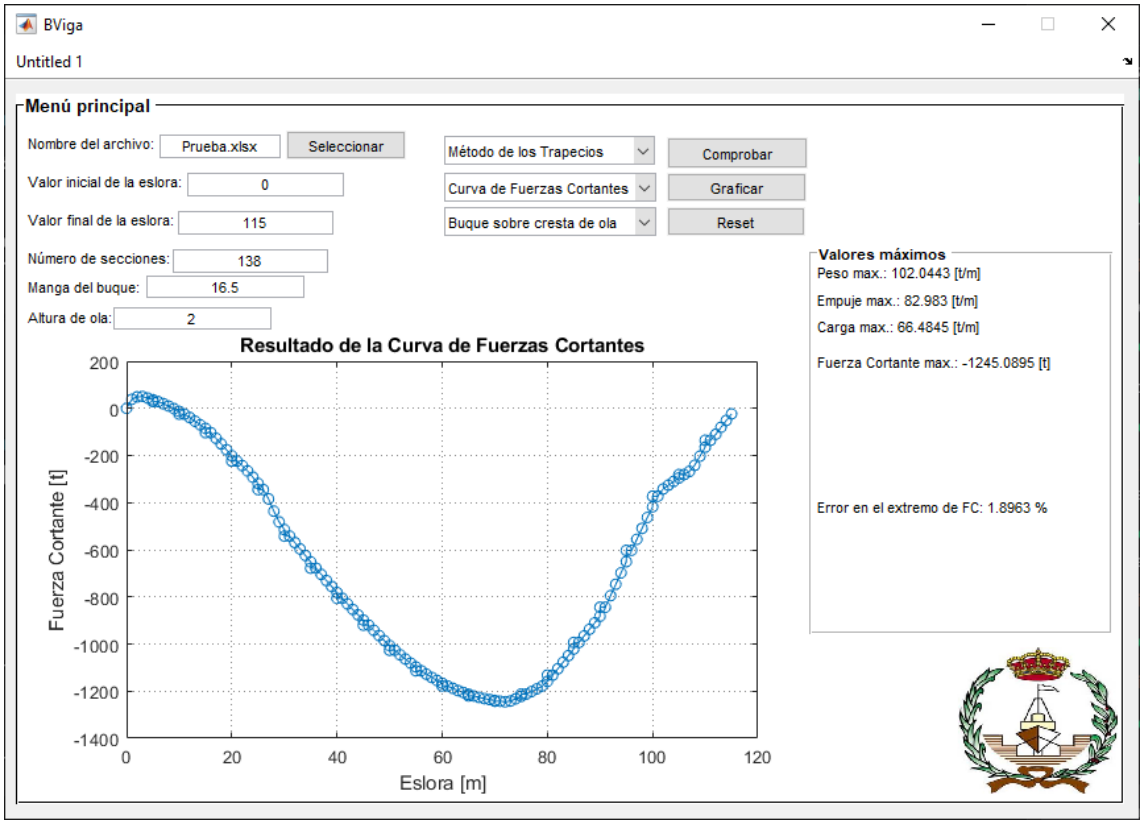


Figura 8.37 – Resultados para la curva de Fuerzas Cortantes mediante el método de los trapecios.

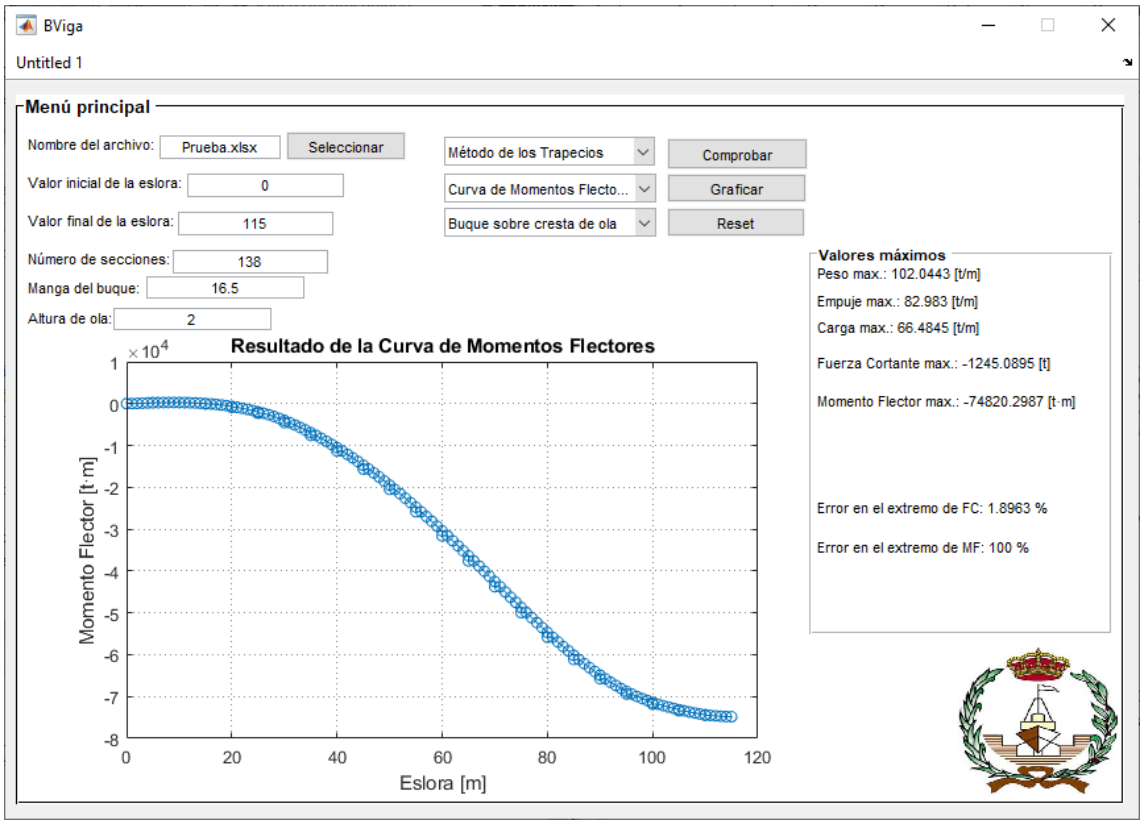


Figura 8.38 – Resultados para la curva de Momentos Cortantes mediante el método de los trapecios.

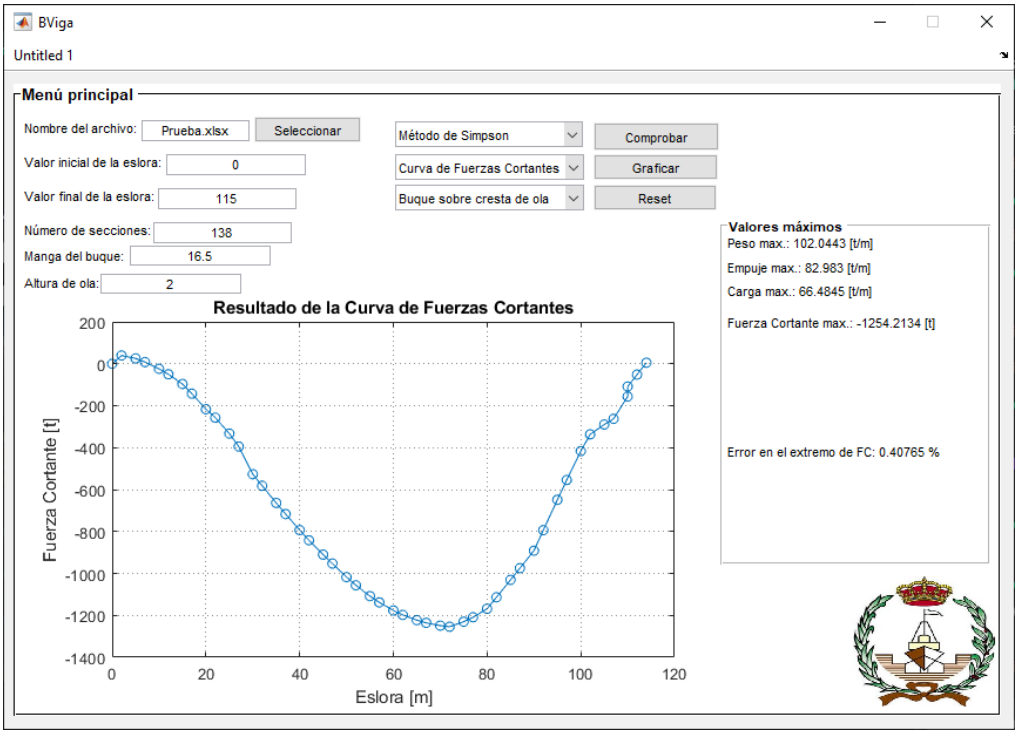


Figura 8.39 – Resultados para la curva de Fuerzas Cortantes mediante el método de Simpson.

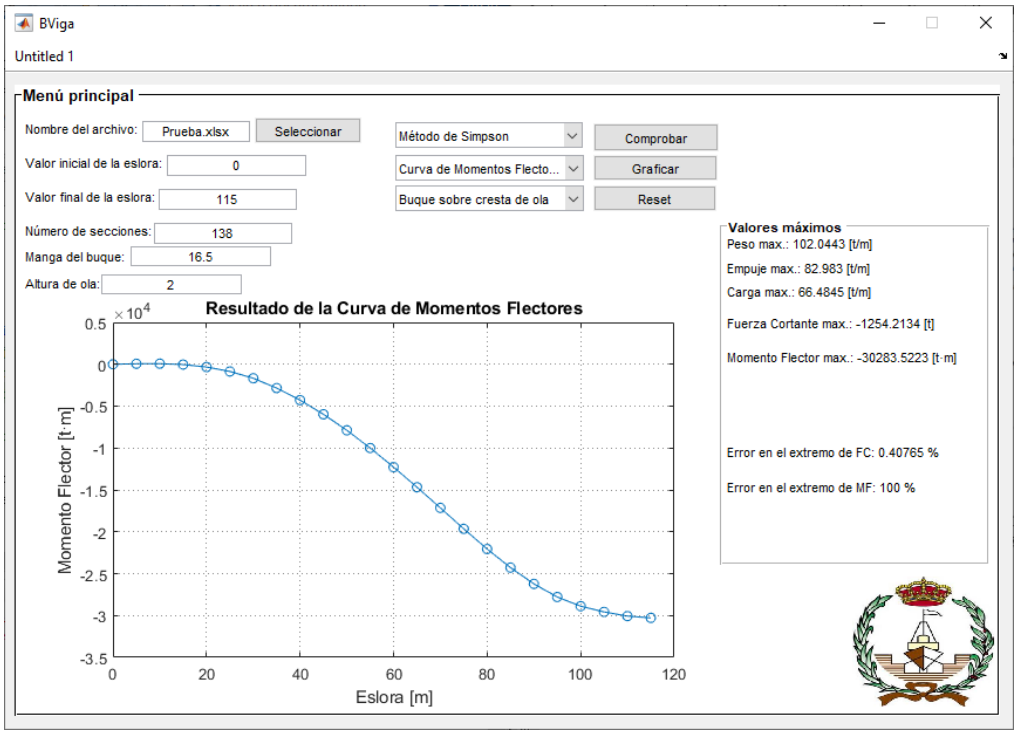


Figura 8.40 – Resultados para la curva de Momentos Cortantes mediante el método de Simpson.

Si comparamos de nuevo los resultados entre los métodos empleados, podemos ver la precisión característica de cada método para este caso.

Errores		
Curva	**Trapecios**	**Simpson**
Fuerzas Cortantes	1.8963%	0.40765%
Momentos Flectores	100%	100%

Figura 8.41 – Comparación de errores entre métodos aplicados.

9 Conclusiones

El buque, a lo largo de su vida útil, se encontrará con diversas situaciones de carga y estados de la mar, por lo que es importante conocer las rutas en las que navegará para poder parametrizar de una forma más precisa los posibles estados más críticos en los que se pueda encontrar, generalmente la situación de arrufo es la más restrictiva.

Tras realizar el alargamiento, se consigue aumentar la capacidad de carga del buque sin necesidad de reducir de forma considerable su velocidad de proyecto, por lo que dicho objetivo se ha completado de forma exitosa. La estimación de pesos siempre se supone como una distribución constante, a pesar de que la carga a lo largo de la eslora pueda ser variable en las diversas situaciones de navegación. Para este estudio, se considera la peor situación de carga para la estructura, esto es, que el buque navegue a plena carga.

Por otro lado, el buque tiene un buen comportamiento para la situación de equilibrio en aguas tranquilas, como hemos visto anteriormente, y a partir del momento flector máximo obtenido, y la inercia de la cuaderna maestra, es posible determinar el espesor de las planchas de cubierta y fondo del buque, así como entrepuentes.

Analizando los resultados obtenidos, y comparando los distintos métodos de cuadratura numérica, se comprueba que los métodos de Simpson son los más adecuados a la hora de obtener mejores aproximaciones a las soluciones reales.

Para este buque, todavía se pueden realizar los estudios de estabilidad para determinar que el buque no zozobre en determinadas situaciones, o en casos de inundación de un local, así como conocer la escora permanente del buque. Este tipo de estudios se pueden proponer como una ampliación al presente trabajo.

Por último, podemos determinar que los objetivos se cumplen para la realización de las distintas interfaces que simplifican y ayudan al usuario a realizar los distintos cálculos comentados anteriormente, mientras que para la aplicación práctica se llega a la conclusión de que es posible que para este tipo de buque se necesite una parametrización más exhausta no estudiada durante el grado.

10 Bibliografía

1. *Diseño y Cálculo de Estructuras Navales*, apuntes de la asignatura, Prof. José Alfonso Martínez García, curso 2016-2017.
2. *Construcción Naval*, apuntes de la asignatura, Prof. Leandro Ruiz Peñalver, curso 2015-2016.
3. *Proyectos*, apuntes de la asignatura, Prof. José Enrique Gutiérrez Romero y Prof. Leandro Ruiz Peñalver, curso 2018-2019.
4. *Hidrodinámica, Resistencia y Propulsión*, apuntes de la asignatura, Prof. Domingo García López, curso 2018-2019.
5. BAQUERO MAYOR, ANTONIO, *"Introducción a la Propulsión de Buques"*. Sección de Publicaciones de la E.T.S.I.N. Madrid.
6. Ship Structural Analysis and Design. Owen Hughes. SNAME.
7. Bonilla de la Corte, Antonio, "Construcción naval y servicios".
8. Etter, Delores M. "Soluciones de problemas de ingeniería con Matlab".
9. Burden y Faires. *"Análisis Numérico"*. International Thomson editores.
10. Stoer and Bulirsch, "Introduction to Numerical Analysis". Springer
11. *El proyecto básico del buque mercante*, Ricardo Alvariño, Juan José Azpíroz y Manuel Meizso.
12. HOLTROP, J., MENNEN, G.G.J. *"An approximate power prediction method"*. Internacional Shipbuilding Progress. Vol. 29, July 1982.
13. «1_DIMENSIONES_Y_CARACTERISTICAS_DEL_BUQUE.pdf»
14. «2_DESCRIPCION_DE_LA_ESTRUCTURA_GENERAL_DEL_BUQUE.pdf»
15. «3_CLASIFICACIÓN DE BUQUES0.pdf»
16. «03_Curvas_Hidrostaticas.pdf». Accedido 14 de enero de 2020. https://wiki.ead.pucv.cl/images/b/ba/03_Curvas_Hidrostaticas.pdf.
17. «4510_0018_04.pdf». Accedido 16 de marzo de 2020. https://marine.man-es.com/docs/librariesprovider6/marine-engine-programmes/4510_0018_04.pdf?sfvrsn=f4e9fda2_56.
18. Adre-es. *Español: Esfuerzo de cizalladura o cortante*. 2 de enero de 2017. Own work. https://commons.wikimedia.org/wiki/File:Esfuerzo-cortante.png.
19. ———. *Español: Esfuerzo de torsión*. 2 de enero de 2017. Own work. https://commons.wikimedia.org/wiki/File:Esfuerzo-torsion.png.
20. ASTILLEROS ARMADA. «Alargamiento buque Pilar Torre», 25 de febrero de 2019. https://astillerosarmada.com/alargamiento-buque-pilar-torre/.
21. Alejandro Díez Fernández. «ISM - Curso Buques RO-RO & Pasaje - 5 estabilidad». Educación, 12:27:30 UTC. https://es.slideshare.net/adiezfernandez/5-estabilidad-17205426.
22. Alvarez, Jorge. «Las insólitas barcazas de hormigón que reposan en el Támesis y se usaron en el Desembarco de Normandía». *La Brújula Verde* (blog), 1 de julio de 2017. https://www.labrujulaverde.com/2017/07/las-insolitas-barcazas-de-hormigon-que-reposan-en-el-tamesis-y-se-usaron-en-el-desembarco-de-normandia.
23. «Archivo:Coeficientes de bloque.svg». En *Wikipedia, la enciclopedia libre*. Accedido 29 de octubre de 2019. https://es.wikipedia.org/wiki/Archivo:Coeficientes_de_bloque.svg#/media/Archivo:Coeficientes_de_bloque.svg.
24. «Archivo:Dimensiones.PNG». En *Wikipedia, la enciclopedia libre*. Accedido 28 de septiembre de 2019. https://es.wikipedia.org/wiki/Archivo:Dimensiones.PNG.

25. «Archivo:Esloras.PNG - Wikipedia, la enciclopedia libre». Accedido 28 de septiembre de 2019. https://commons.wikimedia.org/wiki/File:Esloras.PNG.
26. «Archivo:FultonNautilus.jpg - Wikipedia, la enciclopedia libre». Accedido 21 de octubre de 2019. https://es.m.wikipedia.org/wiki/Archivo:FultonNautilus.jpg.
27. «Archivo:Olas parametros.png». En *Wikipedia, la enciclopedia libre*. Accedido 6 de febrero de 2020. https://es.wikipedia.org/wiki/Archivo:Olas_parametros.png.
28. «Archivo:Perpendiculardepopa.PNG». En *Wikipedia, la enciclopedia libre*. Accedido 28 de septiembre de 2019. https://es.wikipedia.org/wiki/Archivo:Perpendiculardepopa.PNG.
29. «Archivo:Perpendiculardeproa.PNG». En *Wikipedia, la enciclopedia libre*. Accedido 28 de septiembre de 2019. https://es.wikipedia.org/wiki/Archivo:Perpendiculardeproa.PNG.
30. «Archivo:Poutre appuis charge lineaire stat.svg - Wikipedia, la enciclopedia libre». Accedido 4 de marzo de 2020. https://commons.wikimedia.org/wiki/File:Poutre_appuis_charge_lineaire_stat.svg.
31. «Archivo:Poutre appuis console appuyee charge triangle stat.svg». En *Wikipedia, la enciclopedia libre*. Accedido 4 de marzo de 2020. https://es.wikipedia.org/wiki/Archivo:Poutre_appuis_console_appuyee_charge_triangle_stat.svg.
32. «Archivo:Poutre appuis trois appuis charge uniforme deux travees stat.svg - Wikipedia, la enciclopedia libre». Accedido 4 de marzo de 2020. https://commons.wikimedia.org/wiki/File:Poutre_appuis_trois_appuis_charge_uniforme_deux_travees_stat.svg.
33. «Archivo:Runge phenomenon.svg». En *Wikipedia, la enciclopedia libre*. Accedido 23 de noviembre de 2019. https://es.wikipedia.org/wiki/Archivo:Runge_phenomenon.svg.
34. Bilogistik. «Embarcaciones Ro-Ro, Cargas Rodadas Por Vía Marítima». *Bilogistik* (blog), 10 de agosto de 2018. https://www.bilogistik.com/blog/embarcaciones-ro-ro/.
35. Breakdown. *Typical Stress vs. Strain diagram for a ductile material (e.g. Steel).* 12 de marzo de 2008. Own work. https://commons.wikimedia.org/wiki/File:Stress_Strain_Ductile_Material.png.
36. VA DE BARCOS. «Buques de carga rodada (Ro-Ro). El MV Tonsberg.», 14 de septiembre de 2014. https://vadebarcos.net/2014/09/14/buques-carga-rodada-ro-ro-mv-tonsberg/.
37. Scribd. «BUQUE-VIGA». Accedido 27 de septiembre de 2019. https://es.scribd.com/document/131909725/BUQUE-VIGA.
38. Scribd. «BUQUE-VIGA | Inclinarse | Elasticidad (Física)». Accedido 11 de octubre de 2019. https://es.scribd.com/document/131909725/BUQUE-VIGA.
39. «Buscar índices y valores de elementos no nulos - MATLAB find - MathWorks España». Accedido 25 de febrero de 2020. https://es.mathworks.com/help/matlab/ref/find.html.
40. CalmX, some as, Was an Experimental Artist, Film Director, producer, Video Game Content Creator, freelance writer for some 18 years She specialized in writing about inventors, y inventions. «Who Was Robert Fulton, Developer of a Steamboat Called Clermont?» ThoughtCo. Accedido 22 de octubre de 2019. https://www.thoughtco.com/robert-fulton-steamboat-4075444.
41. «Características de las olas. Longitud de onda, altura, amplitud, dirección. / Oceanografía / Apuntes Náuticos / Portada - masmar». Accedido 16 de octubre de 2019. http://www.masmar.net/index.php/esl/Apuntes-

N%C3%A1uticos/Oceanograf%C3%ADa/Caracter%C3%ADsticas-de-las-olas.-Longitud-de-onda,-altura,-amplitud,-direcci%C3%B3n.
42. «Características de las olas. Longitud de onda, altura, amplitud, dirección. / Oceanografía / Apuntes Náuticos / Portada - masmar». Accedido 20 de marzo de 2020. http://www.masmar.net/index.php/esl/Apuntes-N%C3%A1uticos/Oceanograf%C3%ADa/Caracter%C3%ADsticas-de-las-olas.-Longitud-de-onda,-altura,-amplitud,-direcci%C3%B3n.
43. «Cargar datos desde un archivo - MATLAB importdata - MathWorks España». Accedido 25 de noviembre de 2019. https://es.mathworks.com/help/matlab/ref/importdata.html.
44. https://www.cat.com/es_ES/products/new/power-systems/marine-power-systems/commercial-propulsion-engines/18404461.html. «Cat | 3512C IMO II | Caterpillar». Accedido 16 de marzo de 2020. https://www.cat.com/es_ES/products/new/power-systems/marine-power-systems/commercial-propulsion-engines/18404461.html.
45. https://www.cat.com/es_ES/products/new/power-systems/marine-power-systems/commercial-propulsion-engines/1000031003.html. «Cat | 3512E Tier 4 Final de la EPA | Caterpillar». Accedido 17 de marzo de 2020. https://www.cat.com/es_ES/products/new/power-systems/marine-power-systems/commercial-propulsion-engines/1000031003.html.
46. https://www.cat.com/es_ES/products/new/power-systems/marine-power-systems/commercial-propulsion-engines/18408926.html. «Cat | 3516C IMO II | Caterpillar». Accedido 20 de marzo de 2020. https://www.cat.com/es_ES/products/new/power-systems/marine-power-systems/commercial-propulsion-engines/18408926.html.
47. https://www.cat.com/es_ES/products/new/power-systems/marine-power-systems/commercial-propulsion-engines/1000031000.html. «Cat | 3516E Tier 4 Final | Caterpillar». Accedido 20 de marzo de 2020. https://www.cat.com/es_ES/products/new/power-systems/marine-power-systems/commercial-propulsion-engines/1000031000.html.
48. https://www.cat.com/es_ES/products/new/power-systems/marine-power-systems/commercial-propulsion-engines/18385954.html. «Cat | C175-16 | Caterpillar». Accedido 16 de marzo de 2020. https://www.cat.com/es_ES/products/new/power-systems/marine-power-systems/commercial-propulsion-engines/18385954.html.
49. https://www.cat.com/es_ES/products/new/power-systems/marine-power-systems/commercial-propulsion-engines/18385954.html. «Cat | C175-16 | Caterpillar». Accedido 20 de marzo de 2020. https://www.cat.com/es_ES/products/new/power-systems/marine-power-systems/commercial-propulsion-engines/18385954.html.
50. https://www.cat.com/es_ES/products/new/power-systems/marine-power-systems/commercial-propulsion-engines/18393912.html. «Cat | C280-8 | Caterpillar». Accedido 20 de marzo de 2020. https://www.cat.com/es_ES/products/new/power-systems/marine-power-systems/commercial-propulsion-engines/18393912.html.
51. https://www.cat.com/es_ES/products/new/power-systems/marine-power-systems/commercial-propulsion-engines.html. «Cat | Motores de propulsión comerciales | Caterpillar». Accedido 16 de marzo de 2020. https://www.cat.com/es_ES/products/new/power-systems/marine-power-systems/commercial-propulsion-engines.html.

52. Chang, Bunkichi. *中文（繁體）：臺馬輪靠泊基隆港。*. 9 de septiembre de 2014. https://www.flickr.com/photos/120532743@N04/15030255877. https://commons.wikimedia.org/wiki/File:Tai_Ma_Ship_in_Port_of_Keelung_20140909_night.jpg?uselang=es.
53. «Cleve Moler | Computer History Museum». Accedido 29 de octubre de 2019. https://www.computerhistory.org/fellowawards/hall/cleve-moler/.
54. «Concepto de carga - Definición en DeConceptos.com». Accedido 31 de enero de 2020. https://deconceptos.com/ciencias-naturales/carga.
55. «CONCEPTOS BASICOS - Fundamentos Navales». Accedido 28 de septiembre de 2019. https://sites.google.com/site/fundamentosnavales/conceptos-basicos.
56. «Construcción del Buque I: Representación de las formas de un buque». Accedido 14 de marzo de 2020. http://arquitecturabuque.blogspot.com/2011/07/representacion-de-las-formas-de-un.html.
57. «Convenio internacional para la seguridad de la vida humana en el mar, 1974 (Convenio SOLAS)». Accedido 13 de marzo de 2020. http://www.imo.org/es/about/conventions/listofconventions/paginas/international-convention-for-the-safety-of-life-at-sea-(solas),-1974.aspx.
58. Cook, John. *English: Thomas Simpson (1808-40) led an expedition which explored the North American arctic coast, 1836-40*. 1845. https://www.torontopubliclibrary.ca/detail.jsp?Entt=RDMDC-JRR7&R=DC-JRR7. https://commons.wikimedia.org/wiki/File:Thomas_Simpson_(explorer).jpg.
59. «Crear y ejecutar una aplicación sencilla mediante el diseñador de aplicaciones - MATLAB & Simulink - MathWorks España». Accedido 2 de febrero de 2020. https://es.mathworks.com/help/matlab/creating_guis/create-a-simple-app-or-gui-using-app-designer.html.
60. «Create question dialog box - MATLAB questdlg - MathWorks España». Accedido 2 de febrero de 2020. https://es.mathworks.com/help/matlab/ref/questdlg.html.
61. «cresta - Definición - WordReference.com». Accedido 6 de febrero de 2020. https://www.wordreference.com/definicion/cresta.
62. «Cresta (ola)». En *Wikipedia, la enciclopedia libre*, 17 de septiembre de 2019. https://es.wikipedia.org/w/index.php?title=Cresta_(ola)&oldid=119435559.
63. La voz del muro. «Cuando Los Barcos Se Hacían de Hormigón y Su Uso Ayudó a Cambiar La Historia Del Mundo», 16 de enero de 2016. https://lavozdelmuro.net/cuando-los-barcos-se-hacian-de-hormigon-y-su-uso-ayudo-a-cambiar-la-historia-del-mundo/.
64. Davidhv22. *English: Different prow of ship*. [object HTMLTableCellElement]. Elaboración propia. https://commons.wikimedia.org/wiki/File:Formas_de_proas.jpg.
65. ———. *English: elementary structure of ship or boat.* Elaboración propia. Accedido 14 de enero de 2020. https://commons.wikimedia.org/wiki/File:Nomenclatura_del_buque.jpg.
66. «Definición de carga - Qué es, Significado y Concepto». Accedido 31 de enero de 2020. https://definicion.de/carga/.
67. «Deformacion Elastica Elasticidad y Materiales Elasticos». Accedido 31 de enero de 2020. https://www.areatecnologia.com/materiales/deformacion-elastica.html.
68. «delores-m-etter.pdf». Accedido 19 de octubre de 2019. https://cristiancastrop.files.wordpress.com/2010/09/delores-m-etter.pdf.

69. «Descripción del producto MATLAB - MATLAB & Simulink - MathWorks España». Accedido 25 de noviembre de 2019. https://es.mathworks.com/help/matlab/learn_matlab/product-description.html.
70. de.wikipedia, KMJ at. *Deutsch: Der Hafenschlepper Michel dreht den Ro-Ro-Frachter Tamesis aus Tønsberg, von der Reederei Wallenius Wilhelmsen, auf der Norderelbe, um ihn in den Moldauhafen zu bugsieren.* 16 de junio de 2004. Secondary source: http://de.wikipedia.org/wiki/Bild:Hafenschlepper_01_KMJ.jpg. https://commons.wikimedia.org/wiki/File:Hafenschlepper_01_KMJ.jpg.
71. «Downloads». Accedido 16 de marzo de 2020. https://marine.man-es.com/four-stroke/brochures.
72. IT History Society. «Dr. Cleve Barry Moler», 21 de diciembre de 2015. https://www.ithistory.org/honor-roll/dr-cleve-barry-moler.
73. Elias, Claudio. *Esfuerzo de arrufo y quebranto. (spanish)*. 29 de diciembre de 2006. Own work. https://commons.wikimedia.org/wiki/File:Arrufoquebranto.PNG.
74. «Encontrar elementos de matriz que cumplen una condición - MATLAB & Simulink - MathWorks España». Accedido 28 de febrero de 2020. https://es.mathworks.com/help/matlab/matlab_prog/find-array-elements-that-meet-a-condition.html.
75. Wartsila.com. «Engines and generating sets». Accedido 16 de marzo de 2020. https://www.wartsila.com/marine/build/engines-and-generating-sets.
76. *English: Forfarshire (ship)*. Item is held by John Oxley Library, State Library of Queensland. Accedido 6 de marzo de 2020. https://commons.wikimedia.org/wiki/File:StateLibQld_1_145355_Forfarshire_(ship).jpg.
77. «Escala Beaufort y Douglas. Los vientos y la mar / Meteorología / Apuntes Náuticos / Portada - masmar». Accedido 21 de marzo de 2020. http://www.masmar.net/index.php/esl/Apuntes-N%C3%A1uticos/Meteorolog%C3%ADa/Escala-Beaufort-y-Douglas.-Los-vientos-y-la-mar2.
78. «escalas_de_viento_y_oleaje.pdf». Accedido 21 de marzo de 2020. https://www.aemet.es/documentos/es/conocermas/maritima/escalas_de_viento_y_oleaje.pdf.
79. «Esfuerzo de Torsión». Accedido 21 de marzo de 2020. https://www.physicstutorials.org/pt/es/53-Esfuerzo_de_Torsi%C3%B3n.
80. «Estabilidad+de+buques.pdf». Accedido 14 de enero de 2020. https://riull.ull.es/xmlui/bitstream/handle/915/335/Estabilidad+de+buques.pdf;jsessionid=EEAC7D2C3566871B77BC14217FDECDD8?sequence=1.
81. Farelly, Elly. «Concrete Ships Turned Out To Be A Surprisingly Good Idea». *WAR HISTORY ONLINE* (blog), 25 de octubre de 2016. https://www.warhistoryonline.com/military-vehicle-news/concrete-ships-surprisingly-good-idea.html.
82. Cuaderno de Cultura Científica. «Fase y ecuación de onda», 20 de noviembre de 2018. https://culturacientifica.com/2018/11/20/fase-y-ecuacion-de-onda/.
83. «File:Fallen Girders, Tay Bridge.Jpg». En *Wikipedia*. Accedido 6 de marzo de 2020. https://en.wikipedia.org/wiki/File:Fallen_girders,_Tay_Bridge.jpg.
84. «File:Fultondesign7.Jpg». En *Wikipedia*, 26 de enero de 2009. https://en.wikipedia.org/w/index.php?title=File:Fultondesign7.jpg&oldid=266551942.
85. «File:Ro-Ro ship Amedeo Matacena (IMO 8213940) - Harbour of Reggio Calabria - Italy - 3 June 2013.jpg - Wikimedia Commons». Accedido 27 de septiembre de

2019. https://commons.wikimedia.org/wiki/File:Ro-Ro_ship_Amedeo_Matacena_(IMO_8213940)_-_Harbour_of_Reggio_Calabria_-_Italy_-_3_June_2013.jpg.
86. «File:Thomas Bouch ILN.Jpg - Wikimedia Commons». Accedido 27 de septiembre de 2019. https://commons.wikimedia.org/wiki/File:Thomas_Bouch_ILN.jpg.
87. «Fix:Composite trapezoidal rule illustration small.png - Vicipedia». Accedido 15 de noviembre de 2019. https://commons.wikimedia.org/wiki/File:Composite_trapezoidal_rule_illustration_small.png.
88. «*Forfarshire* (Ship)». En *Wikipedia*, 7 de julio de 2019. https://en.wikipedia.org/w/index.php?title=Forfarshire_(ship)&oldid=905227392.
89. «Fotos espacio de carga libres de regalías | Pxfuel». Accedido 28 de abril de 2020. https://www.pxfuel.com/es/search?q=espacio+de+carga.
90. «Four Stroke 28/32A - Profile». Accedido 19 de marzo de 2020. https://marine.man-es.com/four-stroke/engines/l28-32a-draft/profile.
91. «Four-Stroke L27/38 - Profile». Accedido 20 de marzo de 2020. https://marine.man-es.com/four-stroke/engines/l27-38/profile.
92. «Four-Stroke MAN 175D - Profile». Accedido 19 de marzo de 2020. https://marine.man-es.com/four-stroke/engines/175d/profile.
93. Franklin Gualán. «Metodo de simpsons y de los trapecios». Educación, 14:12:44 UTC. https://es.slideshare.net/franklingualan75/metodo-de-simpsons-y-de-los-trapecios.
94. GILFAN. *Español: Líneas de máxima carga representada en el Registro Español.* [object HTMLTableCellElement]. Own work. https://commons.wikimedia.org/wiki/File:Registro_Espa%C3%B1ol.JPG.
95. «Glosario: Frecuencia». Accedido 21 de marzo de 2020. https://ec.europa.eu/health/scientific_committees/opinions_layman/es/campos-electromagneticos/glosario/def/frecuencia.htm.
96. Grasselli, Matheus, Matheus Grasselli, y Dmitry Pelinovsky. «Numerical mathematics». En *6.6 Newton-Cotes integration rules*, 1ª., pp. 328. Massachusetts (USA): Jones & Bartlett Learning, 2008.
97. «grupo.pdf». Accedido 14 de enero de 2020. https://bibliotecadigital.jcyl.es/es/catalogo_imagenes/grupo.cmd?path=10124147.
98. «GUI de MATLAB». Accedido 25 de noviembre de 2019. https://es.mathworks.com/discovery/matlab-gui.html.
99. «GUI_Matlab.pdf». Accedido 25 de noviembre de 2019. http://www.utm.mx/~vero0304/HCPM/GUI_Matlab.pdf.
100. Hägele, Wolfgang. *English: RoRo ferry Nils Holgersson*. [object HTMLTableCellElement]. Transferred from de.wikipedia to Commons. (de:Datei:Roro faehre.jpg). https://commons.wikimedia.org/wiki/File:Roro_faehre.jpg.
101. «Historia de los buques de concreto – Blog de Exordio». Accedido 11 de octubre de 2019. https://www.exordio.com/blog/otros-temas/historia-de-los-buques-de-concreto.html.
102. Holtrop, J., y G. G. J. Mennen. «A STATISTICAL POWER PREDICTION METHOD». *International Shipbuilding Progress* 25, n.º 290 (octubre de 1978). https://trid.trb.org/view/86706.
103. «Identificar con MATLAB valores repetidos en un vector | Silvia Alonso Pérez». Accedido 28 de febrero de 2020.

http://www.silviaalonsoperez.com/2014/07/identificar-con-matlab-valores-repetidos-en-un-vector/.
104. «Imagen gratis en Pixabay - Carguero, Roll-On-Roll-Off». Accedido 15 de febrero de 2020. https://pixabay.com/es/photos/carguero-roll-on-roll-off-contenedor-4764609/.
105. «inicio». Accedido 29 de septiembre de 2019. https://www.fundacionaquae.org/wiki-explora/23_navegantes/index.html.
106. «Integración numérica». En *Wikipedia, la enciclopedia libre*, 22 de octubre de 2019. https://es.wikipedia.org/w/index.php?title=Integraci%C3%B3n_num%C3%A9rica&oldid=120647217.
107. «Integración numérica». Accedido 14 de noviembre de 2019. http://www.sc.ehu.es/sbweb/fisica3/numerico/integral/integral.html.
108. «Ironclad». En *Wikipedia, la enciclopedia libre*, 11 de octubre de 2019. https://es.wikipedia.org/w/index.php?title=Ironclad&oldid=120179368.
109. Jose, Rafael Morilla San. «Los barcos de hormigón», 6 de agosto de 2019. https://quevuelenaltolosdados.com/2019/08/06/barco-de-cemento/.
110. Kepler, Johannes, y Johannes Kepler. *Neue Stereometrie der Fässer*. Leipzig, 1908.
111. Kupras, L. K. «COMPUTER PROGRAM FOR DETERMINATION OF REQUIRED PROPULSIVE POWER AT INITIAL STAGE OF SHIP DESIGN: METHOD OF HOLTROP AND MENNEN», 1984. https://trid.trb.org/view/393696.
112. «La base de datos de OMI-Vega». Accedido 13 de marzo de 2020. http://www.imo.org/es/Publications/Paginas/IMO-Vega.aspx.
113. «Leer datos de archivo de texto - MATLAB fscanf - MathWorks España». Accedido 25 de noviembre de 2019. https://es.mathworks.com/help/matlab/ref/fscanf.html.
114. Logistic, Stock. «EL SISTEMA DE TRANSPORTE RO-RO». *Stock Logistic* (blog), 12 de febrero de 2018. https://www.stocklogistic.com/el-sistema-de-transporte-ro-ro/.
115. ———. «¿Qué es Roll On-Roll Off?» *Stock Logistic* (blog), 21 de junio de 2016. https://www.stocklogistic.com/que-es-roll-roll/.
116. Luis Delgado Lallemand. *De proa a popa. Conceptos básicos. Tomo 1*. Vol. Tomo 1, s. f. Accedido 28 de septiembre de 2019.
117. «MAN 175D Profile». Accedido 20 de marzo de 2020. https://marine.man-es.com/four-stroke/engines/175d/profile.
118. «Maniobra de buques - Maniobras de alijo y completado (Lightering and Top-Off)». Accedido 28 de abril de 2020. http://www.maniobradebuques.com/articulosDeInteres/ainteres21.html.
119. «man-l27-38.pdf». Accedido 20 de marzo de 2020. https://marine.man-es.com/docs/default-source/shopwaredocumentsarchive/man-l27-38.pdf?sfvrsn=d58a5e86_4.
120. «man-l28-32df.pdf». Accedido 17 de marzo de 2020. https://marine.man-es.com/docs/default-source/shopwaredocuments/man-l28-32df.pdf?sfvrsn=1b527d13_5.
121. «MathWorks - Founders - Cleve Moler». Accedido 29 de septiembre de 2019. https://es.mathworks.com/company/aboutus/founders/clevemoler.html.
122. «MATLAB | Universidad de Burgos». Accedido 29 de octubre de 2019. https://www.ubu.es/servicio-de-informatica-y-comunicaciones/catalogo-de-

servicios/software-tu-disposicion/software-disposicion-de-la-comunidad-universitaria/matlab.
123. «MATLAB Documentation - MathWorks España». Accedido 1 de marzo de 2020. https://es.mathworks.com/help/index.html.
124. Mayor, Harry Gonzalez. «Calculo Estructural del Buque: Capítulo 4. Quilla Roda y Codaste.» *Calculo Estructural del Buque* (blog), 4 de febrero de 2012. http://calculoestructuraldelbuque.blogspot.com/2012/01/capitulo-4-quilla-roda-y-codaste.html.
125. mediterráneo, En el fondo del mar. «Los barcos y la navegación en la antigüedad (parte II)». *En el fondo del mar Mediterráneo* (blog), 1 de abril de 2015. https://arqueomediterraneo.wordpress.com/2015/04/01/los-barcos-y-la-navegacion-en-la-antiguedad-parte-ii/.
126. «método de Holtrop | Real Academia de Ingeniería». Accedido 3 de marzo de 2020. http://diccionario.raing.es/es/lema/m%C3%A9todo-de-holtrop.
127. «Métodos Numéricos». Accedido 15 de noviembre de 2019. http://repositorio.uned.ac.cr/multimedias/metodos_numericos_ensenanza/modulo4/descripcion.html.
128. Monografias.com, Pablo Turmero. «Introducción a MATLAB - Monografias.com». Accedido 25 de noviembre de 2019. https://www.monografias.com/trabajos102/introduccion-al-matlab/introduccion-al-matlab.shtml.
129. «Motor para buque - Todos los fabricantes del nautismo y del marítimo». Accedido 16 de marzo de 2020. https://www.nauticexpo.es/fabricante-barco/motor-buque-35923.html.
130. Nikolopoulos, Lampros, y Evangelos Boulougouris. «A Study on the Statistical Calibration of the Holtrop and Mennen Approximate Power Prediction Method for Full Hull Form, Low Froude Number Vessels». Text, febrero de 2019. https://doi.org/info:doi/10.5957/JSPD.170034.
131. «Offshore Supply & Service Vessels». Accedido 16 de marzo de 2020. https://www.mtu-solutions.com/eu/en/applications/commercial-marine/commercial-marine-solutions/offshore-supply-and-service-vessels.html.
132. Ola, La. «DISTRIBUCIÓN DEL PESO.- Una primera aproximación muy útil de la distribución del peso estructural es la de suponer que los 2/3 del peso sigue la curva del empuje en aguas tranquilas y que el el centro de gravedad 1/3 restante se distribuye en forma de trapecio, de manera que finalmente el centro de gravedad de ese», s. f., 12.
133. Olivella Puig, Joan. *Teoría del buque. Ola trocoidal, movimientos y esfuerzos.* Edicions UPC, 1998.
134. Olsson, Kristian Alarcon. «UNIVERISDAD POLITÉCNICA DE CARTAGENA», s. f., 101.
135. Omi, La. «2.- Normativa de la OMI referente a buques de carga rodada3», s. f., 51.
136. «Organización Marítima Internacional». Accedido 13 de marzo de 2020. http://www.imo.org/ES/Paginas/Default.aspx.
137. «Oxford DNB article: Simpson». Accedido 27 de septiembre de 2019. https://www-history.mcs.st-andrews.ac.uk/DNB/Simpson.html.
138. «PARTES DE UN BARCO, PROA POPA AMURA ALETA OBRA VIVA OBRA MUERTA BABOR». Accedido 14 de enero de 2020. https://sailandtrip.com/partes-de-un-barco/.
139. Pedersen, Benjamin Pjedsted, y Jan Larsen. «Prediction of Full-Scale Propulsion Power Using Artificial Neural Networks», s. f., 14.

140. Peregrina Quintela Estévez. *Introducción a Matlab y sus aplicaciones.* II. Servicio de Publicacións da Universidade de Santiago de Compostela Campus universitario sur, 1997.
141. «PFC: Estudio de los sistemas de seguridad para un buque Ro-Pax», s. f., 150.
142. «PG_M-Mk3-III_L2330H.pdf». Accedido 17 de marzo de 2020. https://marine.man-es.com/applications/projectguides/4stroke/manualcontent/PG_M-Mk3-III_L2330H.pdf.
143. «Primeros Barcos de Acero Historia de la Construcción y Evolución». Accedido 11 de octubre de 2019. https://historiaybiografias.com/barcos2/.
144. «Principio de Arquímedes». Accedido 13 de marzo de 2020. http://www.sc.ehu.es/sbweb/fisica/fluidos/estatica/arquimedes/arquimedes.htm.
145. «Profile». Accedido 20 de marzo de 2020. https://marine.man-es.com/four-stroke/engines/175d/profile.
146. «proy-int-num.pdf». Accedido 14 de noviembre de 2019. https://euler.us.es/~renato/clases/am1/proy-int-num.pdf.
147. «Quiénes somos». Accedido 13 de marzo de 2020. http://www.imo.org/es/About/Paginas/Default.aspx.
148. «Quiénes somos OMI». Accedido 16 de octubre de 2019. http://www.imo.org/es/About/Paginas/Default.aspx.
149. Rao, Sankara, y Sankara Rao. «Numerical Methods For Scientists And Engineers». En *7.6 Newton-Cotes integration formulae*, 3ª., pp. & nbsp;151-159. New Delhi (India): Prentice-Hall of India Learning Private, 2007.
150. «References for Thomas Simpson». Accedido 27 de septiembre de 2019. https://www-history.mcs.st-andrews.ac.uk/References/Simpson.html.
151. Richard L. Burden, J. Douglas Daires. *Análisis Numérico*. 6ª Edición. International Thomson Editores, S.A. de C.V., s. f.
152. «Robert Fulton». Accedido 21 de octubre de 2019. /topics-resources/content/Robert-Fulton.
153. Encyclopedia Britannica. «Robert Fulton | Biography, Inventions, & Facts». Accedido 27 de septiembre de 2019. https://www.britannica.com/biography/Robert-Fulton-American-inventor.
154. «Roll Trailers - transport solution for shipping industry - Novatech DK». Accedido 17 de marzo de 2020. http://www.novatech.dk/products,roll-trailers,90.html.
155. «RO-RO Ferries». Accedido 16 de octubre de 2019. http://www.imo.org/es/OurWork/Safety/Regulations/Paginas/RO-ROFerries.aspx.
156. Rus, Cobo, y Juan María. «Cálculo de la potencia y propulsor de un petrolero de productos limpios de 50.000m3 de bodegas», 2015. https://ruc.udc.es/dspace/handle/2183/16580.
157. «Simpson.pdf». Accedido 27 de septiembre de 2019. https://www-history.mcs.st-andrews.ac.uk/DSB/Simpson.pdf.
158. «Sinusoide». En *Wikipedia, la enciclopedia libre*, 21 de febrero de 2020. https://es.wikipedia.org/w/index.php?title=Sinusoide&oldid=123730412.
159. Scribd. «Solucion de Problemas de Ingenieria con Matlab». Accedido 19 de octubre de 2019. https://es.scribd.com/document/48851538/Solucion-de-Problemas-de-Ingenieria-con-Matlab.
160. Tejero, Ignacio Moleres, y Jordi Moncunill Marimón. «ANÁLISIS DE LA OPERATIVA EN BUQUES DE CARGA RODADA», s. f., 130.
161. «Tema7.pdf». Accedido 14 de noviembre de 2019. http://www.dma.uvigo.es/~lino/Tema7.pdf.

162. Terán, Leonardo Vite. «PRINCIPIO DE ARQUÍMEDES», s. f., 9.
163. Marine Insight. «The Dangers of Ro-Ro Ship Design: A Naval Architect's Perspective», 11 de abril de 2019. https://www.marineinsight.com/naval-architecture/ro-ro-ship-design-dangers/.
164. «Thermopylae, el primer buque de la clase HERO. | VA DE BARCOS». Accedido 28 de abril de 2020. https://vadebarcos.net/2017/02/04/thermopylae-primer-buque-clase-hero/.
165. «Thomas Bouch: Biography on Undiscovered Scotland». Accedido 27 de septiembre de 2019. https://www.undiscoveredscotland.co.uk/usbiography/b/thomasbouch.html.
166. «Thomas Simpson (1710-1761)». Accedido 22 de octubre de 2019. http://www-history.mcs.st-andrews.ac.uk/Biographies/Simpson.html.
167. «Tipos de Buques de Carga Transporte Marítimo | DSV». Accedido 17 de marzo de 2020. https://www.es.dsv.com/sea-freight/tipos-buques-carga-transporte-maritimo.
168. «Tipos de camión tráiler y dimensiones | DSV». Accedido 17 de marzo de 2020. https://www.es.dsv.com/road-transport/tipos-de-trailer-y-dimensiones.
169. «Torsión mecánica». En *Wikipedia, la enciclopedia libre*, 7 de enero de 2020. https://es.wikipedia.org/w/index.php?title=Torsi%C3%B3n_mec%C3%A1nica&oldid=122564102.
170. «Transporte RO-RO». Accedido 11 de octubre de 2019. https://www.transeop.com/blog/Transporte-RO-RO/420/.
171. «Trapezoidal Rule». En *Wikipedia*, 6 de noviembre de 2019. https://en.wikipedia.org/w/index.php?title=Trapezoidal_rule&oldid=924897659.
172. Turnbull, S. R., y D. Dawson. «THE SECURING OF VEHICLES ON ROLL-ON/ROLL-OFF SHIPS», 1995. https://trid.trb.org/view/449799.
173. VA DE BARCOS. «Una breve historia de los barcos de hormigón armado», 22 de enero de 2019. https://vadebarcos.net/2019/01/22/una-breve-historia-de-los-barcos-de-hormigon-armado/.
174. User:Andibrunt. *Image cropped Clermont illustration - Robert Fulton from Project Gutenberg eText 15161.jpg*. 1870. cropped image of Image:Clermont illustration - Robert Fulton - Project Gutenberg eText 15161.jpg. https://commons.wikimedia.org/wiki/File:Robert_Fulton_Clermont_cropped.jpg.
175. «Valores únicos en array - MATLAB unique - MathWorks España». Accedido 11 de abril de 2020. https://es.mathworks.com/help/matlab/ref/unique.html.
176. Wartsila.com. «Wärtsilä - Enabling sustainable societies with smart technology». Accedido 16 de marzo de 2020. https://www.wartsila.com.
177. Wartsila.com. «Wärtsilä 14 high-speed engine». Accedido 16 de marzo de 2020. https://www.wartsila.com/marine/build/engines-and-generating-sets/diesel-engines/wartsila-14.
178. Wartsila.com. «Wärtsilä 20 - Diesel engine». Accedido 16 de marzo de 2020. https://www.wartsila.com/marine/build/engines-and-generating-sets/diesel-engines/wartsila-20.
179. Wartsila.com. «Wärtsilä 26 - diesel engine». Accedido 20 de marzo de 2020. https://www.wartsila.com/marine/build/engines-and-generating-sets/diesel-engines/wartsila-26.
180. Marine Insight. «What Are Ro-Ro Ships?», 10 de mayo de 2019. https://www.marineinsight.com/types-of-ships/what-are-ro-ro-ships/.
181. «When was the first ship with a metal hull built? - Quora». Accedido 11 de octubre de 2019. https://www.quora.com/When-was-the-first-ship-with-a-metal-hull-built.

182. Wussing, Hans, y Hans Wussing. *Lecciones de historia de las matemáticas*. Siglo XXI de España Editores, 1998. http://books.google.es/books?id=IG3_b5Xm8PMC&lpg=PA141&dq=Kepler%20barril&pg=PA142#v=onepage&q&f=false.
183. Accedido 11 de octubre de 2019. https://www.histarmar.com.ar/nomenclatura/TeoriadelBuque.htm.
184. Accedido 11 de octubre de 2019. https://www.histarmar.com.ar/InfGral/BarcosConcretoBase.htm.

11 Anexo I

A continuación, se presentan los códigos realizados para la realización de este proyecto:

11.1 Programa para la Interfaz de Inicio

```
function varargout = Inicio(varargin)
% INICIO MATLAB code for Inicio.fig
%      INICIO, by itself, creates a new INICIO or raises the existing
%      singleton*.
%
%      H = INICIO returns the handle to a new INICIO or the handle to
%      the existing singleton*.
%
%      INICIO('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in INICIO.M with the given input
arguments.
%
%      INICIO('Property','Value',...) creates a new INICIO or raises
the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before Inicio_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Inicio_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Inicio

% Last Modified by GUIDE v2.5 12-Mar-2020 18:39:05

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Inicio_OpeningFcn, ...
                   'gui_OutputFcn',  @Inicio_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before Inicio is made visible.
function Inicio_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Inicio (see VARARGIN)

% Choose default command line output for Inicio
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Inicio wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% Código para que al iniciar el programa se coloque a la mitad de la
% pantalla
set(handles.output,'Units','pixels');
screenSize=get(0,'ScreenSize');
position=get(handles.output,'Position');
position(1)=(screenSize(3)-position(3))/2;
position(2)=(screenSize(4)-position(4))/2;
set(handles.output,'Position',position);
% Imagen Logo UPCT
axes(handles.axes1);
Logo1=imread('ExcelenciaInternacional.jpg');
image(Logo1);
axis off;
% Imagen Logo ETSINO
axes(handles.axes2);
Logo2=imread('etsino_nuevo.jpg');
image(Logo2);
axis off;

% --- Outputs from this function are returned to the command line.
function varargout = Inicio_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Iniciar=get(hObject,'Value'); %Obtenemos el valor del botón para saber
si se ha clicado sobre él
if Iniciar==1
    if get(handles.popupmenu1,'Value')==1
        Prueba1 %Ejecutamos la interfaz correspondiente al cálculo de
barco viga
    elseif get(handles.popupmenu1,'Value')==2
```

```
        holtrop %Ejecutamos la interfaz correspondiente a la
aplicación del método de holtrop
    end
end


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu1


% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

11.2 Programa para la Interfaz del Método de Holtrop

```
function varargout = holtrop(varargin)
% HOLTROP MATLAB code for holtrop.fig
%      HOLTROP, by itself, creates a new HOLTROP or raises the
existing
%      singleton*.
%
%      H = HOLTROP returns the handle to a new HOLTROP or the handle
to
%      the existing singleton*.
%
%      HOLTROP('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in HOLTROP.M with the given input
arguments.
%
%      HOLTROP('Property','Value',...) creates a new HOLTROP or raises
the
```

```
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before holtrop_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to holtrop_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help holtrop

% Last Modified by GUIDE v2.5 15-Mar-2020 19:15:09

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @holtrop_OpeningFcn, ...
                   'gui_OutputFcn',  @holtrop_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before holtrop is made visible.
function holtrop_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to holtrop (see VARARGIN)

% Choose default command line output for holtrop
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes holtrop wait for user response (see UIRESUME)
% uiwait(handles.figure1);
set(handles.output,'Units','pixels');
screenSize=get(0,'ScreenSize');
position=get(handles.output,'Position');
position(1)=(screenSize(3)-position(3))/2;
position(2)=(screenSize(4)-position(4))/2;
set(handles.output,'Position',position);
```

```
axes(handles.axes1);
Logo1=imread('etsino_nuevo.jpg');
image(Logo1);
axis off;
set(handles.radiobutton2,'Value',0);
set(handles.radiobutton6,'Value',0);
set(handles.edit51,'Enable','off');
set(handles.edit50,'Enable','off');
set(handles.edit66,'Enable','off');
set(handles.edit67,'Enable','off');
axes(handles.axes2);
plot(handles.axes2,(0:1),0);
axes(handles.axes3);
plot(handles.axes3,(0:1),0);
% --- Outputs from this function are returned to the command line.
function varargout = holtrop_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8
as a double


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9
as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text
%        str2double(get(hObject,'String')) returns contents of edit12
as a double


% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit13_Callback(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit13 as text
%        str2double(get(hObject,'String')) returns contents of edit13
as a double


% --- Executes during object creation, after setting all properties.
function edit13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2
as a double



% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3
as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4
as a double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5
as a double



% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6
as a double


% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7
as a double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10
as a double


% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit14_Callback(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit14 as text
%        str2double(get(hObject,'String')) returns contents of edit14
as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11
as a double


% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in checkbox2.
function checkbox2_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox2
TyQ=get(hObject,'Value');
if TyQ==1
    set(handles.edit19,'Enable','on');
    set(handles.checkbox13,'Value',1);
    set(handles.edit57,'Enable','on');
    set(handles.radiobutton1,'Enable','off');
    set(handles.radiobutton2,'Enable','off');
    set(handles.radiobutton5,'Enable','off');
    set(handles.radiobutton6,'Enable','off');
elseif TyQ==0
    set(handles.edit19,'Enable','off');
```

```
    set(handles.edit19,'String','');
    set(handles.checkbox13,'Value',0);
    set(handles.edit57,'Enable','off');
    set(handles.edit57,'String','');
    set(handles.radiobutton1,'Enable','on');
    set(handles.radiobutton2,'Enable','on');
    set(handles.radiobutton5,'Enable','on');
    set(handles.radiobutton6,'Enable','on');
end


function edit19_Callback(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit19 as text
%        str2double(get(hObject,'String')) returns contents of edit19
as a double


% --- Executes during object creation, after setting all properties.
function edit19_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in checkbox3.
function checkbox3_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox3

Quillote=get(hObject,'Value');
if Quillote==1
    set(handles.edit20,'Enable','on');
    set(handles.checkbox14,'Value',1);
    set(handles.edit58,'Enable','on');
elseif Quillote==0
    set(handles.edit20,'Enable','off');
    set(handles.edit20,'String','');
    set(handles.checkbox14,'Value',0);
    set(handles.edit58,'Enable','off');
    set(handles.edit58,'String','');
end
```

```
function edit20_Callback(hObject, eventdata, handles)
% hObject    handle to edit20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit20 as text
%        str2double(get(hObject,'String')) returns contents of edit20
as a double


% --- Executes during object creation, after setting all properties.
function edit20_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in checkbox4.
function checkbox4_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox4

Arbotante=get(hObject,'Value');
if Arbotante==1
    set(handles.edit21,'Enable','on');
    set(handles.checkbox15,'Value',1);
    set(handles.edit59,'Enable','on');
elseif Arbotante==0
    set(handles.edit21,'Enable','off');
    set(handles.edit21,'String','');
    set(handles.checkbox15,'Value',0);
    set(handles.edit59,'Enable','off');
    set(handles.edit59,'String','');
end



function edit21_Callback(hObject, eventdata, handles)
% hObject    handle to edit21 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit21 as text
%        str2double(get(hObject,'String')) returns contents of edit21
as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit21_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit21 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in checkbox5.
function checkbox5_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox5
Hench_prot=get(hObject,'Value');
if Hench_prot==1
    set(handles.edit22,'Enable','on');
    set(handles.checkbox16,'Value',1);
    set(handles.edit60,'Enable','on');
elseif Hench_prot==0
    set(handles.edit22,'Enable','off');
    set(handles.edit22,'String','');
    set(handles.checkbox16,'Value',0);
    set(handles.edit60,'Enable','off');
    set(handles.edit60,'String','');
end



function edit22_Callback(hObject, eventdata, handles)
% hObject    handle to edit22 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit22 as text
%        str2double(get(hObject,'String')) returns contents of edit22
as a double



% --- Executes during object creation, after setting all properties.
function edit22_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit22 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end


% --- Executes on button press in checkbox6.
function checkbox6_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox6
Hench_int=get(hObject,'Value');
if Hench_int==1
    set(handles.edit23,'Enable','on');
    set(handles.checkbox17,'Value',1);
    set(handles.edit61,'Enable','on');
elseif Hench_int==0
    set(handles.edit23,'Enable','off');
    set(handles.edit23,'String','');
    set(handles.checkbox17,'Value',0);
    set(handles.edit61,'Enable','off');
    set(handles.edit61,'String','');
end



function edit23_Callback(hObject, eventdata, handles)
% hObject    handle to edit23 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit23 as text
%        str2double(get(hObject,'String')) returns contents of edit23
as a double

% --- Executes during object creation, after setting all properties.
function edit23_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit23 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in checkbox7.
function checkbox7_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox7

Ejes=get(hObject,'Value');
if Ejes==1
```

```
    set(handles.edit24,'Enable','on');
    set(handles.checkbox18,'Value',1);
    set(handles.edit62,'Enable','on');
elseif Ejes==0
    set(handles.edit24,'Enable','off');
    set(handles.edit24,'String','');
    set(handles.checkbox18,'Value',0);
    set(handles.edit62,'Enable','off');
    set(handles.edit62,'String','');
end


function edit24_Callback(hObject, eventdata, handles)
% hObject    handle to edit24 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit24 as text
%        str2double(get(hObject,'String')) returns contents of edit24
as a double


% --- Executes during object creation, after setting all properties.
function edit24_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit24 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox1

Bulbo=get(hObject,'Value');
if Bulbo==1
    set(handles.text15,'Enable','on');
    set(handles.text16,'Enable','on');
    set(handles.edit14,'Enable','on');
    set(handles.edit15,'Enable','on');
elseif Bulbo==0
    set(handles.text15,'Enable','off');
    set(handles.text16,'Enable','off');
    set(handles.edit14,'Enable','off');
    set(handles.edit15,'Enable','off');
    set(handles.edit14,'String','');
    set(handles.edit15,'String','');
end
```

```
function edit15_Callback(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit15 as text
%        str2double(get(hObject,'String')) returns contents of edit15
as a double


% --- Executes during object creation, after setting all properties.
function edit15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit16_Callback(hObject, eventdata, handles)
% hObject    handle to edit16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit16 as text
%        str2double(get(hObject,'String')) returns contents of edit16
as a double


% --- Executes during object creation, after setting all properties.
function edit16_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit17_Callback(hObject, eventdata, handles)
% hObject    handle to edit17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit17 as text
%        str2double(get(hObject,'String')) returns contents of edit17
as a double


% --- Executes during object creation, after setting all properties.
function edit17_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit18_Callback(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit18 as text
%        str2double(get(hObject,'String')) returns contents of edit18
as a double


% --- Executes during object creation, after setting all properties.
function edit18_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in togglebutton2.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of togglebutton2

Mostrar_rango=get(hObject,'Value');
if Mostrar_rango==1
    Rango=msgbox({'Tipo de apéndice
(1+k2)';'-------------------------------------------------------------
';'Timón buque 1 hélice                  1.3 a 1.5';'Timón buque 2
hélices                    2.8';'Timón y Quillote
1.5 a 2.0';'Quillote solo                                  1.5 a
2.0';'Arbotantes
```

```
3.0';'Henchimientos protectores            3.0';'Henchimientos
integrados             2.0';'Ejes
2.0 a 4.0';'Aletas estabilizadoras                    2.8';'Domo
2.7';'Quillas de balance                          1.4'},'Rango de
aplicación','help');
end

% --- Executes on button press in checkbox8.
function checkbox8_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox8

Aletas=get(hObject,'Value');
if Aletas==1
    set(handles.edit25,'Enable','on');
    set(handles.checkbox19,'Value',1);
    set(handles.edit63,'Enable','on');
elseif Aletas==0
    set(handles.edit25,'Enable','off');
    set(handles.edit25,'String','');
    set(handles.checkbox19,'Value',0);
    set(handles.edit63,'Enable','off');
    set(handles.edit63,'String','');
end



function edit25_Callback(hObject, eventdata, handles)
% hObject    handle to edit25 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit25 as text
%        str2double(get(hObject,'String')) returns contents of edit25
as a double


% --- Executes during object creation, after setting all properties.
function edit25_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit25 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in checkbox9.
function checkbox9_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hint: get(hObject,'Value') returns toggle state of checkbox9
Domo=get(hObject,'Value');
if Domo==1
    set(handles.edit26,'Enable','on');
    set(handles.checkbox20,'Value',1);
    set(handles.edit64,'Enable','on');
elseif Domo==0
    set(handles.edit26,'Enable','off');
    set(handles.edit26,'String','');
    set(handles.checkbox20,'Value',0);
    set(handles.edit64,'Enable','off');
    set(handles.edit64,'String','');
end


function edit26_Callback(hObject, eventdata, handles)
% hObject    handle to edit26 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit26 as text
%        str2double(get(hObject,'String')) returns contents of edit26
as a double


% --- Executes during object creation, after setting all properties.
function edit26_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit26 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in checkbox12.
function checkbox12_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox12
Quilla_Balance=get(hObject,'Value');
if Quilla_Balance==1
    set(handles.edit48,'Enable','on');
    set(handles.checkbox21,'Value',1);
    set(handles.edit65,'Enable','on');
elseif Quilla_Balance==0
    set(handles.edit48,'Enable','off');
    set(handles.edit48,'String','');
    set(handles.checkbox21,'Value',0);
    set(handles.edit65,'Enable','off');
    set(handles.edit65,'String','');
```

```
end



function edit48_Callback(hObject, eventdata, handles)
% hObject    handle to edit48 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit48 as text
%        str2double(get(hObject,'String')) returns contents of edit48
as a double


% --- Executes during object creation, after setting all properties.
function edit48_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit48 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit49_Callback(hObject, eventdata, handles)
% hObject    handle to edit49 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit49 as text
%        str2double(get(hObject,'String')) returns contents of edit49
as a double

% --- Executes during object creation, after setting all properties.
function edit49_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit49 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit50_Callback(hObject, eventdata, handles)
% hObject    handle to edit50 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit50 as text
%        str2double(get(hObject,'String')) returns contents of edit50
as a double



% --- Executes during object creation, after setting all properties.
function edit50_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit50 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit51_Callback(hObject, eventdata, handles)
% hObject    handle to edit51 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit51 as text
%        str2double(get(hObject,'String')) returns contents of edit51
as a double


% --- Executes during object creation, after setting all properties.
function edit51_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit51 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all properties.
function togglebutton2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to togglebutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called


% --- Executes during object creation, after setting all properties.
```

```
function pushbutton6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called


function edit52_Callback(hObject, eventdata, handles)
% hObject    handle to edit52 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit52 as text
%        str2double(get(hObject,'String')) returns contents of edit52
as a double


% --- Executes during object creation, after setting all properties.
function edit52_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit52 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit53_Callback(hObject, eventdata, handles)
% hObject    handle to edit53 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit53 as text
%        str2double(get(hObject,'String')) returns contents of edit53
as a double


% --- Executes during object creation, after setting all properties.
function edit53_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit53 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton1

Timon_1helice=get(hObject,'Value');
if Timon_1helice==1
    set(handles.edit50,'Enable','on');
    set(handles.edit66,'Enable','on');
    set(handles.radiobutton6,'Enable','off');
    set(handles.radiobutton2,'Enable','off');
    set(handles.checkbox2,'Enable','off');
    set(handles.checkbox13,'Enable','off');
    set(handles.radiobutton5,'Value',1);
else
    set(handles.edit50,'Enable','off');
    set(handles.edit66,'Enable','off');
    set(handles.edit50,'String','');
    set(handles.edit66,'String','');
    set(handles.radiobutton2,'Enable','on');
    set(handles.radiobutton6,'Enable','on');
    set(handles.checkbox2,'Enable','on');
    set(handles.checkbox13,'Enable','on');
    set(handles.radiobutton5,'Value',0);
end

% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton2

Timon_2helices=get(hObject,'Value');
if Timon_2helices==1
    set(handles.edit51,'Enable','on');
    set(handles.checkbox2,'Enable','off');
    set(handles.checkbox13,'Enable','off');
    set(handles.radiobutton1,'Enable','off');
    set(handles.radiobutton5,'Enable','off');
    set(handles.edit67,'Enable','on');
    set(handles.radiobutton6,'Value',1);
else
    set(handles.edit50,'Enable','off');
    set(handles.edit67,'Enable','off');
    set(handles.edit50,'String','');
    set(handles.edit67,'String','');
    set(handles.radiobutton1,'Enable','on');
    set(handles.radiobutton5,'Enable','on');
    set(handles.checkbox2,'Enable','on');
    set(handles.checkbox13,'Enable','on');
    set(handles.radiobutton6,'Value',0);
end


% --- Executes on button press in togglebutton2.
```

```
function togglebutton2_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of togglebutton2
global LCB
LCB1=get(handles.edit13,'String');
LCB=str2num(LCB1);
global CP
CP1=get(handles.edit10,'String');
CP=str2num(CP1);
global CB
CB1=get(handles.edit8,'String');
CB=str2num(CB1);
global CM
CM1=get(handles.edit9,'String');
CM=str2num(CM1);
global CF
CF1=get(handles.edit11,'String');
CF=str2num(CF1);
global Tpp
Tpp1=get(handles.edit4,'String');
Tpp=str2num(Tpp1);
global Tpr
Tpr1=get(handles.edit6,'String');
Tpr=str2num(Tpr1);
global Tm
Tm1=get(handles.edit5,'String');
Tm=str2num(Tm1);
global Lf
Lf1=get(handles.edit2,'String');
Lf=str2num(Lf1);
global Lpp
Lpp1=get(handles.edit1,'String');
Lpp=str2num(Lpp1);
global B
B1=get(handles.edit3,'String');
B=str2num(B1);
global A_Quillote
if get(handles.checkbox3,'Value')==1
    S_Quillote1=get(handles.edit20,'String');
    A_Quillote=str2num(S_Quillote1);
else
    A_Quillote=0;
end
global A_Arbotante
if get(handles.checkbox4,'Value')==1
    S_Arbotante1=get(handles.edit21,'String');
    A_Arbotante=str2num(S_Arbotante1);
else
    A_Arbotante=0;
end
global A_Hprot
if get(handles.checkbox5,'Value')==1
    A_Hprot1=get(handles.edit22,'String');
    A_Hprot=str2num(A_Hprot1);
else
    A_Hprot=0;
end
global A_Hint
```

```
if get(handles.checkbox6,'Value')==1
    A_Hint1=get(handles.edit23,'String');
    A_Hint=str2num(A_Hint1);
else
    A_Hint=0;
end
global A_Ejes
if get(handles.checkbox7,'Value')==1
    A_Ejes1=get(handles.edit24,'String');
    A_Ejes=str2num(A_Ejes1);
else
    A_Ejes=0;
end
global A_BT %Área transversal del bulbo
if get(handles.checkbox1,'Value')==1
    A_BT1=get(handles.edit15,'String');
    A_BT=str2num(A_BT1);
else
    A_BT=0;
end
global Temp_agua
Temp_agua1=get(handles.edit16,'String');
Temp_agua=str2num(Temp_agua1);
global A_Domo
if get(handles.checkbox9,'Value')==1
    A_Domo1=get(handles.edit26,'String');
    A_Domo=str2num(A_Domo1);
else
    A_Domo=0;
end
global Fn
global Fn_v
global v
global velocidades
v1=get(handles.edit12,'String');
v=str2num(v1);
v_redondear=round(v);
if v>v_redondear
    velocidades=[v_redondear-3 v_redondear-2 v_redondear-1 v_redondear
v v_redondear+1 v_redondear+2 v_redondear+3];
elseif v<v_redondear
    velocidades=[v_redondear-3 v_redondear-2 v_redondear-1 v
v_redondear v_redondear+1 v_redondear+2 v_redondear+3];
elseif v==v_redondear
    velocidades=[v_redondear-3 v_redondear-2 v_redondear-1 v_redondear
v_redondear+1 v_redondear+2 v_redondear+3];
elseif isempty(v)==1
    msgbox('Los datos introducidos no son válidos.','Error','error');
end
global pos_v
pos_v=find(velocidades==v);
global h_B %Altura del bulbo de proa
if get(handles.checkbox1,'Value')==1
    h_B1=get(handles.edit14,'String');
    h_B=str2num(h_B1);
else
    h_B=0;
end
global g
global V_carena
V_carena1=get(handles.edit7,'String');
```

```
V_carena=str2num(V_carena1);
global A_T
A_T1=get(handles.edit52,'String');
A_T=str2num(A_T1);
global Rugosidad
Rugosidad1=get(handles.edit49,'String');
Rugosidad=str2num(Rugosidad1);
if isempty(Rugosidad)==1
    Rugosidad=150;%Valor general estimado para la rugosidad de las
carenas
end
global Sm
Sm1=get(handles.edit17,'String');
Sm=str2num(Sm1);
global A_TR
A_TR1=get(handles.edit18,'String');
A_TR=str2num(A_TR1);
global A_Aletas
if get(handles.checkbox8,'Value')==1
    A_Aletas1=get(handles.edit25,'String');
    A_Aletas=str2num(A_Aletas1);
else
    A_Aletas=0;
end
global A_Qbalance
if get(handles.checkbox12,'Value')==1
    A_Qbalance1=get(handles.edit48,'String');
    A_Qbalance=str2num(A_Qbalance1);
else
    A_Qbalance=0;
end
global A_Tim
if get(handles.radiobutton1,'Value')==1
    A_Tim1=get(handles.edit50,'String');
    A_Tim=str2num(A_Tim1);
    if isempty(A_Tim1)==1
        A_Tim=Lpp*Tm/100*(1.05+25*(B/Lpp)^2)*2;
    end
elseif get(handles.radiobutton2,'Value')==1
    A_Tim1=get(handles.edit51,'String');
    A_Tim=str2num(A_Timon_2helices1);
elseif get(handles.checkbox2,'Value')==1
    A_Tim1=get(handles.edit19,'String');
    A_Tim=str2num(A_TyQ1);
end
global L_B
global Cstern
Cstern1=get(handles.edit55,'String');
Cstern=str2num(Cstern1);
global i_E
iE=get(handles.edit56,'String');
i_E=str2num(iE);
global k2_tim
if get(handles.radiobutton1,'Value')==1
    k2_tim=get(handles.edit66,'String');
    k2_tim=str2num(k2_tim);
elseif get(handles.radiobutton2,'Value')==1
    k2_tim=get(handles.edit67,'String');
    k2_tim=str2num(k2_tim);
elseif get(handles.checkbox13,'Value')==1
    k2_tim=get(handles.edit57,'String');
```

```
    k2_tim=str2num(k2_tim);
end
global k2_quillote
if get(handles.checkbox14,'Value')==1
    k2_quillote=get(handles.edit58,'String');
    k2_quillote=str2num(k2_quillote);
else
    k2_quillote=0;
end
global k2_arbotante
if get(handles.checkbox15,'Value')==1
    k2_arbotante=get(handles.edit59,'String');
    k2_arbotante=str2num(k2_arbotante);
else
    k2_arbotante=0;
end
global k2_hench_prot
if get(handles.checkbox16,'Value')==1
    k2_hench_prot=get(handles.edit60,'String');
    k2_hench_prot=str2num(k2_hench_prot);
else
    k2_hench_prot=0;
end
global k2_hench_int
if get(handles.checkbox17,'Value')==1
    k2_hench_int=get(handles.edit61,'String');
    k2_hench_int=str2num(k2_hench_int);
else
    k2_hench_int=0;
end
global k2_ejes
if get(handles.checkbox18,'Value')==1
    k2_ejes=get(handles.edit62,'String');
    k2_ejes=str2num(k2_ejes);
else
    k2_ejes=0;
end
global k2_aletas
if get(handles.checkbox19,'Value')==1
    k2_aletas=get(handles.edit63,'String');
    k2_aletas=str2num(k2_aletas);
else
    k2_aletas=0;
end
global k2_domo
if get(handles.checkbox20,'Value')==1
    k2_domo=get(handles.edit64,'String');
    k2_domo=str2num(k2_domo);
else
    k2_domo=0;
end
global k2_quilla
if get(handles.checkbox21,'Value')==1
    k2_quilla=get(handles.edit65,'String');
    k2_quilla=str2num(k2_quilla);
else
    k2_quilla=0;
end
%% Procedemos a comprobar los datos introducidos
g=9.81;
for i=1:length(velocidades)
```

```
    Fn(i)=(velocidades(i)*0.51445)/sqrt(g*Lpp);
end
Fn_v=(v*0.51445)/sqrt(g*Lpp);
L_B=Lf/B;
Comprobar_Datos=get(hObject,'Value');
if Comprobar_Datos==1
%     if Fn_v<0.35 && 0.55<CP && CP<0.67 && 5.7<L_B && L_B<8
        msgbox('El rango de valores es válido.','Comprobar
Datos','help');
        set(handles.pushbutton5,'Enable','on');
%     else%if Fn_v>0.35 || 0.55>CP || CP>0.67 || 5.7>L_B || L_B>8 ||
CB<=0 || isnan(CB)==1|| CM<=0 || isnan(CM)==1|| v<=0 || isnan(v)==1 ||
isnan(LCB)==1 || isnan(Lpp)==1 || Lpp<=0 || Lf<=0 || isnan(Lf)==1 ||
B<=0 || isnan(B)==1 || Tpp<=0 || isnan(Tpp)==1 || Tpr<=0 ||
isnan(Tpr)==1 || Tm<=0 || isnan(Tm)==1 || V_carena<=0 ||
isnan(V_carena)==1 || A_T<=0 || isnan(A_T)==1 || Rugosidad<=0 ||
isnan(Rugosidad)==1 || isnan(Temp_agua)==1 || Sm<=0 || isnan(Sm)==1 ||
A_TR<=0 || isnan(A_TR)==1
%         errordlg('Los datos introducidos no son
válidos.','Error','error');
%         set(handles.pushbutton5,'Enable','off');
%     end
end


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%% Definimos las variables necesarias previas a los cálculos
global V_carena
global Lf
global B
global Tpr
global Tm
global CB
global CM
global CP
global CF
global LCB
global pos_v
global v
global velocidades
global h_B
global A_BT
global Sm
global A_TR
global A_Tim
global A_Quillote
global A_Arbotante
global A_Hprot
global A_Hint
global A_Ejes
global A_Aletas
global A_Domo
global A_Qbalance
global g
global Fn
global Cstern
global Rn
```

```
global viscosidad
global densidad
global i_E
global R_T
global S_Ap
global EHP
global Agua_d
global Agua_s
global k2_tim
global k2_quillote
global k2_arbotante
global k2_hench_prot
global k2_hench_int
global k2_ejes
global k2_aletas
global k2_quilla
global k2_domo
global Temp_agua
if Agua_d==1 && Agua_s==0
    densidad=1000;%Densidad del agua dulce a 15°C en Kg/m3
    viscosidad=((0.585*10^(-3)*(Temp_agua-12)-0.03361)*(Temp_agua-
12)+1.235)*10^(-6);
elseif Agua_s==1 && Agua_d==0
    densidad=1026;%Densidad del agua de mar a 15°C en Kg/m3
    viscosidad=((0.659*10^(-3)*(Temp_agua-1)-0.05076)*(Temp_agua-
1)+1.7688)*10^(-6); %Estimacion de la viscosidad a partir de la
temperatura del agua
end
if isempty(Sm)==1
    Sm=Lf*(2*Tm+B)*(CM^0.5)*(0.453+0.4425*CB-0.2862*CM-
0.003467*B/Tm+0.3696*CF)+2.38*A_BT/CB;
    %Esta ecuacion se utiliza para estimar la superficie mojada si no
se
    %conoce
end
%Comienzo a calcular las resistencias de las que se compone la
Resistencia
%Total R_T, empiezo por la resistencia viscosa R_v
%Realizo la iteración para el rango de velocidades para obtener una
curva
%R_T-V o EHP-v

for i=1:length(velocidades)
    %% Resistencia viscosa
    LR=Lf*(1-CP+((0.06*CP*LCB)/(4*CP-1)));
    C_14=1+0.011*Cstern;

factor_k1=0.93+0.487118*C_14*((B/Lf)^1.06806)*((Tm/Lf)^0.46106)*((Lf/L
R)^0.121563)*((Lf^3/V_carena)^0.36486)*(1-CP)^(-0.604247);
    Rn=(velocidades(i)*0.51445*Lf)/viscosidad;%Calculo el numero de
Reynolds
    C_f=0.075/(log10(Rn)-2)^2;%Calculo el coeficiente de resistencia
de fricción s/ITTC-57

R_v(i)=(1/2)*(densidad/g)*Sm*(velocidades(i)*0.51445)^2*C_f*factor_k1;

    %% Resistencia de los apéndices R_AP
    S_i=[A_Tim A_Quillote A_Arbotante A_Hprot A_Hint A_Ejes A_Aletas
A_Domo A_Qbalance];
    S_Ap=sum(S_i);
```

```
    factor_k2i=[k2_tim k2_quillote k2_arbotante k2_hench_prot
k2_hench_int k2_ejes k2_aletas k2_domo k2_quilla];
    S_i_factor=S_i.*factor_k2i;
    suma_Sifactor=sum(S_i_factor);
    factor_k2eq=suma_Sifactor/S_Ap;

R_AP(i)=(1/2)*(densidad/g)*S_Ap*(velocidades(i)*0.51445)^2*C_f*factor_
k2eq;

    %% Resistencia por formación de olas R_w
    %Como el numero de Froude siempre será menor que 0.4, aplicamos
las
    %siguientes fórmulas para la resistencia por formacion de olas R_w
    if isempty(i_E)==1
        i_E=1+89*(exp(-(Lf/B)^0.80856*(1-CF)^0.30484*(1-CP-
0.0225*LCB)^0.6367*(LR/B)^0.34574*(100*V_carena/Lf^3)^0.16302));
        %Esta ecuacion se utiliza para estimar la superficie mojada si
no se conoce
    end
    if B/Lf<=0.11
        C_7=0.229577*(B/Lf)^0.33333;
    elseif 0.11<B/Lf && B/Lf<=0.25
        C_7=B/Lf;
    elseif 0.25<B/Lf
        C_7=0.5-0.0625*(Lf/B);
    end
    d=-0.9;
    C_1=2223105*C_7^3.78613*(Tm/B)^1.07961*(90-i_E)^(-1.37565);
    C_3=(0.56*A_BT^1.5)/(B*Tm*(0.31*sqrt(A_BT)+Tpr-h_B));
    C_2=exp(-1.89*sqrt(C_3));
    C_5=1-(0.8*A_TR)/(B*Tm*CM);
    if CP<=0.8
        C_16=8.07981*CP-13.8673*CP^2+6.984388*CP^3;%Siempre tendremos
un valor de CP<0.8 para este tipo de buques
    elseif CP>0.8
        C_16=1.73014-0.7067*CP;
    end
    m_1=0.014047*(Lf/Tm)-1.75254*(V_carena^(1/3)/Lf)-4.79323*(B/Lf)-
C_16;
    if (Lf^3/V_carena)<=512
        C_15=-1.693851;
    elseif (Lf^3/V_carena)>512 && (Lf^3/V_carena)<=1727
        C_15=-1.69385+Lf^3/V_carena-0.8/2.36;
    elseif (Lf^3/V_carena)>1727
        C_15=0;
    end
    m_2=C_15*CP^2*0.4*exp(-0.1*Fn(i)^(-2));
    if Lf/B<=12
        lambda=1.446*CP-0.03*Lf/B;
    elseif Lf/B>12
        lambda=1.446*CP-0.36;
    end

R_w(i)=(densidad/g)*g*V_carena*C_1*C_2*C_5*exp(m_1*Fn(i)^d+m_2*cos(lam
bda*Fn(i)^(-2)));

    %% Resistencia de presión producida por el bulbo cerca de la
flotación R_B
    P_B=(0.56*sqrt(A_BT))/(Tpr-1.5*h_B);
    Fn_i=(velocidades(i)*0.51445)/sqrt(g*(Tpr-h_B-
0.25*sqrt(A_BT))+0.15*(velocidades(i)*0.51445)^2);
```

```
    R_B(i)=0.11*exp(-3*P_B^(-
2))*(Fn_i^3*A_BT^1.5*densidad*g)/(1+Fn_i^2);

    %% Resistencia adicional debida a la inmersión del espejo R_TR
    Fn_NT=(velocidades(i)*0.51445)/sqrt((2*g*A_TR)/(B+B*CF));
    if Fn_NT<5
        C_6=0.2*(1-0.2*Fn_NT);
    elseif Fn_NT>=5
        C_6=0;
    end
    R_TR(i)=(1/2)*(densidad/g)*(velocidades(i)*0.51445)^2*A_TR*C_6;

    %% Resistencia debida a la correlación modelo-buque R_A, tiene en
cuenta
    %también la rugosidad del casco y la resistencia del aire.
    C_3=1-(0.8*A_TR)/(B*Tm*CM);
    if Tpr/Lf<=0.04
        C_4=Tpr/Lf;
    elseif Tpr/Lf>0.04
        C_4=0.04;
    end
    C_2=exp(-1.89*sqrt(C_3));
    C_A=0.006*(Lf+100)^(-0.16)-
0.00205+0.003*(Lf/7.5)^0.5*CB^4*C_2*(0.04-C_4);
    R_A(i)=(1/2)*(densidad/g)*Sm*(velocidades(i)*0.51445)*C_A;

    %% La resistencia total R_T será la suma de todas las anteriores
    R_T(i)=R_v(i)+R_AP(i)+R_w(i)+R_B(i)+R_TR(i)+R_A(i);
    EHP(i)=(R_T(i)*(velocidades(i)*0.51445))/75;
end
%% Locacizo el valor exacto para la velocidad del buque en pruebas y
lo muestro en la interfaz
global R_T_v
global EHP_v
sol_v=['Velocidad del buque en pruebas: ' num2str(v) ' [nudos]'];
R_T_v=R_T(pos_v);
sol_RT=['Resistencia Total estimada: ' num2str(R_T_v) ' [Kg]'];
EHP_v=R_T_v*v*0.51445/75;
sol_EHP=['Potencia Efectiva estimada: ' num2str(EHP_v) ' [CV]'];
set(handles.text47,'String',sol_v);
set(handles.text48,'String',sol_RT);
set(handles.text49,'String',sol_EHP);
%% Represento gráficamente los resultados obtenidos
axes(handles.axes2);
handles.plot1=plot(handles.axes2,velocidades,R_T,'-'); hold
on;plot(handles.axes2,v,R_T_v,'o');hold off;
grid on;ax=gca;ax.GridLineStyle=':';ax.GridAlpha=0.5;ax.Layer='top';
xlabel('Velocidad [nudos]');ylabel('Resistencia Total
[Kg]');title('Resultado de la Curva R_T-v');
set(handles.plot1,'HitTest','off');
set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));

axes(handles.axes3);
handles.plot2=plot(handles.axes3,velocidades,EHP,'-'); hold
on;plot(handles.axes3,v,EHP_v,'o');hold off;
grid on;ax=gca;ax.GridLineStyle=':';ax.GridAlpha=0.5;ax.Layer='top';
xlabel('Velocidad [nudos]');ylabel('Potencia Efectiva
[CV]');title('Resultado de la Curva EHP-v');
set(handles.plot2,'HitTest','off');
```

```
set(handles.axes3,'ButtonDownFcn',@(s,e)axes3_ButtonDownFcn(s,e,handle
s));

%Almaceno los datos en una tabla para guardarla en excel
T=table(velocidades',R_T');
T.Properties.VariableNames={'Velocidad_buque','Resistencia_Total'};
% Guardo los resultados en un excel
filename='Resultados_Holtrop.xls';
writetable(T,filename,'Sheet','Resistencia_Total');
T1=table(velocidades',EHP');
T1.Properties.VariableNames={'Velocidad_buque','Potencia_Efectiva'};
% Guardo los resultados en un excel
filename='Resultados_Holtrop.xls';
writetable(T1,filename,'Sheet','EHP');

set(handles.pushbutton5,'Value',0);

function edit54_Callback(hObject, eventdata, handles)
% hObject    handle to edit54 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit54 as text
%        str2double(get(hObject,'String')) returns contents of edit54
as a double


% --- Executes during object creation, after setting all properties.
function edit54_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit54 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit55_Callback(hObject, eventdata, handles)
% hObject    handle to edit55 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit55 as text
%        str2double(get(hObject,'String')) returns contents of edit55
as a double


% --- Executes during object creation, after setting all properties.
function edit55_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit55 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit56_Callback(hObject, eventdata, handles)
% hObject    handle to edit56 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit56 as text
%        str2double(get(hObject,'String')) returns contents of edit56
as a double


% --- Executes during object creation, after setting all properties.
function edit56_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit56 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on mouse press over axes background.
function axes2_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to axes2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global velocidades
global R_T
global v
global R_T_v
handles.newfig=figure('Name','Representación
gráfica','Numbertitle','off');
plot(velocidades,R_T,'-'); hold on; plot(v,R_T_v,'o'); hold off;
grid on;ax=gca;ax.GridLineStyle=':';ax.GridAlpha=0.5;ax.Layer='top';
xlabel('Velocidad [nudos]');ylabel('Resistencia Total
[Kg]');title('Resultado de la Curva R_T-v');
set(handles.newfig,'Units','pixels');
screenSize=get(0,'ScreenSize');
position=get(handles.output,'Position');
position(1)=(screenSize(3)-position(3));
position(2)=(screenSize(4)-position(4));
set(handles.newfig,'Position',position);
movegui(handles.newfig,'center');
```

```
% --- Executes on mouse press over axes background.
function axes3_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to axes3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global velocidades
global EHP
global v
global EHP_v
handles.newfig=figure('Name','Representación
gráfica','Numbertitle','off');
plot(velocidades,EHP,'-'); hold on; plot(v,EHP_v,'o'); hold off;
grid on;ax=gca;ax.GridLineStyle=':';ax.GridAlpha=0.5;ax.Layer='top';
xlabel('Velocidad [nudos]');ylabel('Potencia Efectiva
[CV]');title('Resultado de la Curva EHP-v');
set(handles.newfig,'Units','pixels');
screenSize=get(0,'ScreenSize');
position=get(handles.output,'Position');
position(1)=(screenSize(3)-position(3));
position(2)=(screenSize(4)-position(4));
set(handles.newfig,'Position',position);
movegui(handles.newfig,'center');
% --- Executes on button press in pushbutton5.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

Reset=get(hObject,'Value');
if Reset==1
    set(handles.edit1,'String','');
    set(handles.edit2,'String','');
    set(handles.edit3,'String','');
    set(handles.edit4,'String','');
    set(handles.edit5,'String','');
    set(handles.edit6,'String','');
    set(handles.edit7,'String','');
    set(handles.edit8,'String','');
    set(handles.edit9,'String','');
    set(handles.edit10,'String','');
    set(handles.edit11,'String','');
    set(handles.edit12,'String','');
    set(handles.edit13,'String','');
    set(handles.edit14,'String','');
    set(handles.edit15,'String','');
    set(handles.edit52,'String','');
    set(handles.edit49,'String','');
    set(handles.edit16,'String','');
    set(handles.edit17,'String','');
    set(handles.edit18,'String','');
    set(handles.edit50,'String','');
    set(handles.edit51,'String','');
    set(handles.edit19,'String','');
    set(handles.edit20,'String','');
    set(handles.edit21,'String','');
    set(handles.edit22,'String','');
    set(handles.edit23,'String','');
    set(handles.edit24,'String','');
    set(handles.edit25,'String','');
    set(handles.edit26,'String','');
```

```
set(handles.edit48,'String','');
set(handles.edit55,'String','');
set(handles.edit56,'String','');
set(handles.edit57,'String','');
set(handles.edit58,'String','');
set(handles.edit59,'String','');
set(handles.edit60,'String','');
set(handles.edit61,'String','');
set(handles.edit62,'String','');
set(handles.edit63,'String','');
set(handles.edit64,'String','');
set(handles.edit65,'String','');
set(handles.edit66,'String','');
set(handles.edit67,'String','');
set(handles.pushbutton5,'Enable','off');
set(handles.pushbutton6,'Value',0);
set(handles.togglebutton2,'Value',0);
set(handles.checkbox1,'Value',0);
set(handles.checkbox2,'Value',0);
set(handles.checkbox2,'Enable','on');
set(handles.checkbox3,'Value',0);
set(handles.checkbox4,'Value',0);
set(handles.checkbox5,'Value',0);
set(handles.checkbox6,'Value',0);
set(handles.checkbox7,'Value',0);
set(handles.checkbox8,'Value',0);
set(handles.checkbox9,'Value',0);
set(handles.checkbox12,'Value',0);
set(handles.checkbox13,'Value',0);
set(handles.checkbox13,'Enable','on');
set(handles.checkbox14,'Value',0);
set(handles.checkbox15,'Value',0);
set(handles.checkbox16,'Value',0);
set(handles.checkbox17,'Value',0);
set(handles.checkbox18,'Value',0);
set(handles.checkbox19,'Value',0);
set(handles.checkbox20,'Value',0);
set(handles.checkbox21,'Value',0);
set(handles.text47,'String','');
set(handles.text48,'String','');
set(handles.text49,'String','');
set(handles.radiobutton2,'Value',0);
set(handles.radiobutton2,'Enable','on');
set(handles.radiobutton1,'Value',0);
set(handles.radiobutton1,'Enable','on');
set(handles.radiobutton3,'Value',0);
set(handles.radiobutton4,'Value',1);
set(handles.radiobutton6,'Value',0);
set(handles.radiobutton6,'Enable','on');
set(handles.radiobutton5,'Value',0);
set(handles.radiobutton5,'Enable','on');
set(handles.edit50,'Enable','off');
set(handles.edit51,'Enable','off');
set(handles.edit19,'Enable','off');
set(handles.edit66,'Enable','off');
set(handles.edit67,'Enable','off');
set(handles.edit57,'Enable','off');
axes(handles.axes2);
plot(handles.axes2,(0:1),0);
axes(handles.axes3);
plot(handles.axes3,(0:1),0);
```

```
end


% --- Executes on button press in radiobutton3.
function radiobutton3_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton3
global Agua_d
global Agua_s
Agua_d=get(hObject,'Value');
if Agua_d==1
    set(handles.radiobutton4,'Value',0);
    Agua_d=1;
    Agua_s=0;
else
    set(handles.radiobutton4,'Value',1);
    Agua_s=1;
    Agua_d=0;
end


% --- Executes on button press in radiobutton4.
function radiobutton4_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton4
global Agua_s
global Agua_d
Agua_s=get(hObject,'Value');
if Agua_s==1
    set(handles.radiobutton3,'Value',0);
    Agua_s=1;
    Agua_d=0;
else
    set(handles.radiobutton3,'Value',1);
    Agua_d=1;
    Agua_s=0;
end


% --- Executes on button press in checkbox13.
function checkbox13_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox13

TyQ=get(hObject,'Value');
if TyQ==1
    set(handles.edit19,'Enable','on');
    set(handles.checkbox2,'Value',1);
    set(handles.edit57,'Enable','on');
    set(handles.radiobutton1,'Enable','off');
```

```
    set(handles.radiobutton2,'Enable','off');
    set(handles.radiobutton5,'Enable','off');
    set(handles.radiobutton6,'Enable','off');
elseif TyQ==0
    set(handles.edit19,'Enable','off');
    set(handles.edit19,'String','');
    set(handles.checkbox2,'Value',0);
    set(handles.edit57,'Enable','off');
    set(handles.edit57,'String','');
    set(handles.radiobutton1,'Enable','on');
    set(handles.radiobutton2,'Enable','on');
    set(handles.radiobutton5,'Enable','on');
    set(handles.radiobutton6,'Enable','on');
end


function edit57_Callback(hObject, eventdata, handles)
% hObject    handle to edit57 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit57 as text
%        str2double(get(hObject,'String')) returns contents of edit57
as a double


% --- Executes during object creation, after setting all properties.
function edit57_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit57 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in checkbox14.
function checkbox14_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox14

Quillote=get(hObject,'Value');
if Quillote==1
    set(handles.edit20,'Enable','on');
    set(handles.checkbox14,'Value',1);
    set(handles.edit58,'Enable','on');
elseif Quillote==0
    set(handles.edit20,'Enable','off');
    set(handles.edit20,'String','');
    set(handles.checkbox14,'Value',0);
    set(handles.edit58,'Enable','off');
```

```
    set(handles.edit58,'String','');
end


function edit58_Callback(hObject, eventdata, handles)
% hObject    handle to edit58 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit58 as text
%        str2double(get(hObject,'String')) returns contents of edit58
as a double


% --- Executes during object creation, after setting all properties.
function edit58_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit58 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in checkbox15.
function checkbox15_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox15
Arbotante=get(hObject,'Value');
if Arbotante==1
    set(handles.edit21,'Enable','on');
    set(handles.checkbox4,'Value',1);
    set(handles.edit59,'Enable','on');
elseif Arbotante==0
    set(handles.edit21,'Enable','off');
    set(handles.edit21,'String','');
    set(handles.checkbox4,'Value',0);
    set(handles.edit59,'Enable','off');
    set(handles.edit59,'String','');
end


function edit59_Callback(hObject, eventdata, handles)
% hObject    handle to edit59 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit59 as text
%        str2double(get(hObject,'String')) returns contents of edit59
as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit59_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit59 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in checkbox16.
function checkbox16_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox16
Hench_prot=get(hObject,'Value');
if Hench_prot==1
    set(handles.edit22,'Enable','on');
    set(handles.checkbox5,'Value',1);
    set(handles.edit60,'Enable','on');
elseif Hench_prot==0
    set(handles.edit22,'Enable','off');
    set(handles.edit22,'String','');
    set(handles.checkbox5,'Value',0);
    set(handles.edit60,'Enable','off');
    set(handles.edit60,'String','');
end



function edit60_Callback(hObject, eventdata, handles)
% hObject    handle to edit60 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit60 as text
%        str2double(get(hObject,'String')) returns contents of edit60
as a double



% --- Executes during object creation, after setting all properties.
function edit60_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit60 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end


% --- Executes on button press in checkbox17.
function checkbox17_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox17
Hench_int=get(hObject,'Value');
if Hench_int==1
    set(handles.edit23,'Enable','on');
    set(handles.checkbox6,'Value',1);
    set(handles.edit61,'Enable','on');
elseif Hench_int==0
    set(handles.edit23,'Enable','off');
    set(handles.edit23,'String','');
    set(handles.checkbox6,'Value',0);
    set(handles.edit61,'Enable','off');
    set(handles.edit61,'String','');
end


function edit61_Callback(hObject, eventdata, handles)
% hObject    handle to edit61 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit61 as text
%        str2double(get(hObject,'String')) returns contents of edit61
as a double


% --- Executes during object creation, after setting all properties.
function edit61_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit61 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in checkbox18.
function checkbox18_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox18
Ejes=get(hObject,'Value');
if Ejes==1
    set(handles.edit24,'Enable','on');
```

```
    set(handles.checkbox7,'Value',1);
    set(handles.edit62,'Enable','on');
elseif Ejes==0
    set(handles.edit24,'Enable','off');
    set(handles.edit24,'String','');
    set(handles.checkbox7,'Value',0);
    set(handles.edit62,'Enable','off');
    set(handles.edit62,'String','');
end


function edit62_Callback(hObject, eventdata, handles)
% hObject    handle to edit62 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit62 as text
%        str2double(get(hObject,'String')) returns contents of edit62
as a double


% --- Executes during object creation, after setting all properties.
function edit62_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit62 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in checkbox19.
function checkbox19_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox19

Aletas=get(hObject,'Value');
if Aletas==1
    set(handles.edit25,'Enable','on');
    set(handles.checkbox8,'Value',1);
    set(handles.edit63,'Enable','on');
elseif Aletas==0
    set(handles.edit25,'Enable','off');
    set(handles.edit25,'String','');
    set(handles.checkbox8,'Value',0);
    set(handles.edit63,'Enable','off');
    set(handles.edit63,'String','');
end

function edit63_Callback(hObject, eventdata, handles)
% hObject    handle to edit63 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit63 as text
%        str2double(get(hObject,'String')) returns contents of edit63
as a double


% --- Executes during object creation, after setting all properties.
function edit63_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit63 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in checkbox20.
function checkbox20_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox20

Domo=get(hObject,'Value');
if Domo==1
    set(handles.edit26,'Enable','on');
    set(handles.checkbox9,'Value',1);
    set(handles.edit64,'Enable','on');
elseif Domo==0
    set(handles.edit26,'Enable','off');
    set(handles.edit26,'String','');
    set(handles.checkbox9,'Value',0);
    set(handles.edit64,'Enable','off');
    set(handles.edit64,'String','');
end


function edit64_Callback(hObject, eventdata, handles)
% hObject    handle to edit64 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit64 as text
%        str2double(get(hObject,'String')) returns contents of edit64
as a double


% --- Executes during object creation, after setting all properties.
function edit64_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit64 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in checkbox21.
function checkbox21_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox21 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox21

Quilla_Balance=get(hObject,'Value');
if Quilla_Balance==1
    set(handles.edit48,'Enable','on');
    set(handles.checkbox12,'Value',1);
    set(handles.edit65,'Enable','on');
elseif Quilla_Balance==0
    set(handles.edit48,'Enable','off');
    set(handles.edit48,'String','');
    set(handles.checkbox12,'Value',0);
    set(handles.edit65,'Enable','off');
    set(handles.edit65,'String','');
end


function edit65_Callback(hObject, eventdata, handles)
% hObject    handle to edit65 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit65 as text
%        str2double(get(hObject,'String')) returns contents of edit65
as a double


% --- Executes during object creation, after setting all properties.
function edit65_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit65 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in radiobutton5.
```

```
function radiobutton5_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton5
Timon_1helice=get(hObject,'Value');
if Timon_1helice==1
    set(handles.radiobutton2,'Value',0);
    set(handles.checkbox2,'Enable','off');
    set(handles.checkbox13,'Enable','off');
    set(handles.radiobutton2,'Enable','off');
    set(handles.radiobutton6,'Enable','off');
    set(handles.edit50,'Enable','on');
    set(handles.edit66,'Enable','on');
    set(handles.radiobutton1,'Value',1);
else
    set(handles.edit66,'Enable','off');
    set(handles.edit50,'Enable','off');
    set(handles.edit66,'String','');
    set(handles.checkbox2,'Enable','on');
    set(handles.checkbox13,'Enable','on');
    set(handles.radiobutton2,'Enable','on');
    set(handles.radiobutton6,'Enable','on');
    set(handles.edit50,'String','');
    set(handles.edit66,'String','');
    set(handles.radiobutton1,'Value',0);
end

% --- Executes on button press in radiobutton6.
function radiobutton6_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton6
Timon_2helices=get(hObject,'Value');
if Timon_2helices==1
    set(handles.edit67,'Enable','on');
    set(handles.checkbox2,'Enable','off');
    set(handles.checkbox13,'Enable','off');
    set(handles.radiobutton1,'Enable','off');
    set(handles.radiobutton5,'Enable','off');
    set(handles.edit51,'Enable','on');
    set(handles.radiobutton2,'Value',1);
else
    set(handles.edit67,'Enable','off');
    set(handles.edit67,'String','');
    set(handles.checkbox2,'Enable','on');
    set(handles.checkbox13,'Enable','on');
    set(handles.radiobutton1,'Enable','on');
    set(handles.edit51,'Enable','off');
    set(handles.radiobutton2,'Value',0);
end



function edit66_Callback(hObject, eventdata, handles)
% hObject    handle to edit66 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit66 as text
%        str2double(get(hObject,'String')) returns contents of edit66
as a double


% --- Executes during object creation, after setting all properties.
function edit66_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit66 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit67_Callback(hObject, eventdata, handles)
% hObject    handle to edit67 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit67 as text
%        str2double(get(hObject,'String')) returns contents of edit67
as a double


% --- Executes during object creation, after setting all properties.
function edit67_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit67 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Mostrar_rango=get(hObject,'Value');
if Mostrar_rango==1
    Rango=msgbox({'   Rango de aplicación para buques RoRo   ';'
Fn_max              0.35';'                Cp                   0.55-
0.67';'               L/B                    5.3-8.0';'
B/T                   3.2-4.0'},'Rango','help');
end
```

```
% --- Executes during object creation, after setting all properties.
function pushbutton7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

11.3 Programa para la Interfaz del Buque – Viga

```
function varargout = BViga(varargin)
% BVIGA MATLAB code for BViga.fig
%      BVIGA, by itself, creates a new BVIGA or raises the existing
%      singleton*.
%
%      H = BVIGA returns the handle to a new BVIGA or the handle to
%      the existing singleton*.
%
%      BVIGA('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in BVIGA.M with the given input
arguments.
%
%      BVIGA('Property','Value',...) creates a new BVIGA or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before BViga_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to BViga_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help BViga

% Last Modified by GUIDE v2.5 26-Mar-2020 16:48:59

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @BViga_OpeningFcn, ...
                   'gui_OutputFcn',  @BViga_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before BViga is made visible.
function BViga_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to BViga (see VARARGIN)

% Choose default command line output for BViga
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes BViga wait for user response (see UIRESUME)
% uiwait(handles.figure1);
set(handles.output,'Units','pixels');
screenSize=get(0,'ScreenSize');
position=get(handles.output,'Position');
position(1)=(screenSize(3)-position(3))/2;
position(2)=(screenSize(4)-position(4))/2;
set(handles.output,'Position',position);

axes(handles.axes1);
Logo1=imread('etsino_nuevo.jpg');
image(Logo1);
axis off;

% --- Outputs from this function are returned to the command line.
function varargout = BViga_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes when uipanel1 is resized.
function uipanel1_SizeChangedFcn(hObject, eventdata, handles)
% hObject    handle to uipanel1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2
as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3
as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit4_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4
as a double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --------------------------------------------------------------------
function Untitled_1_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate axes1

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu1


% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu2
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu2


% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'));
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function axes2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate axes2

% --- Executes on selection change in popupmenu3.
function popupmenu3_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu3
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu3

if get(hObject,'Value')==3||get(hObject,'Value')==4
    set(handles.edit5,'Enable','on');
    set(handles.text15,'Enable','on');
else
    set(handles.edit5,'Enable','off');
    set(handles.text15,'Enable','off');
end
```

```
% --- Executes during object creation, after setting all properties.
function popupmenu3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all properties.
function text8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called


% --- Executes during object creation, after setting all properties.
function text9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called


% --- Executes during object creation, after setting all properties.
function text10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called


% --- Executes during object creation, after setting all properties.
function text11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called


% --- Executes during object creation, after setting all properties.
function text12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
archivos=dir;
list={};
cnt=0;
for i=1:length(archivos)
    final=endsWith(archivos(i,1).name,'.xlsx');
    if final==1
        cnt=cnt+1;
        list{cnt,1}=archivos(i,1).name;
    else
    end
end
[index,tf]=listdlg('ListString',list);
for j=1:index
    if index==j
        set(handles.edit1,'String',list{j,1});
    elseif j==0
        f=msgbox('No se ha seleccionado la opción de forma
correcta','Error:','error');
    else
        continue
    end
end

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Reset=get(hObject,'Value');
if Reset==1
    r=questdlg('¿Está seguro de continuar y resetear todos datos
introducidos?','Alerta','Si','No','No');
    if strcmp(r,'No') %Esta función compara dos strings
        return;
    else
    set(handles.pushbutton3,'Enable','off');
    set(handles.pushbutton1,'Enable','off');
    set(handles.edit5,'Enable','off');
    set(handles.text15,'Enable','off');
    axes(handles.axes2);
    plot(handles.axes2,(0:1),0);
    set(handles.edit1,'String','');
    set(handles.edit2,'String','');
    set(handles.edit3,'String','');
    set(handles.edit4,'String','');
    set(handles.edit5,'String','');
    set(handles.edit6,'String','');
    set(handles.text8,'String','');
    set(handles.text9,'String','');
    set(handles.text10,'String','');
    set(handles.text11,'String','');
    set(handles.text12,'String','');
    set(handles.text13,'String','');
    set(handles.text14,'String','');
    set(handles.text16,'String','');
    set(handles.text17,'String','');
    set(handles.popupmenu1,'Value',1);
    set(handles.popupmenu2,'Value',1);
    set(handles.popupmenu3,'Value',1);
    set(handles.pushbutton4,'Value',0);
```

```
    set(handles.pushbutton3,'Value',0);
    set(handles.pushbutton1,'Value',0);
    f=msgbox('Se han reseteado todos los datos','Aviso:','warn');
    end
end

% --- Executes on button press in pushbutton4.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global metodo
global Grafica
global Comprobar
global Equilibrio
Comprobar=1;
metodo=get(handles.popupmenu1,'Value');
Grafica=get(handles.popupmenu2,'Value');
Equilibrio=get(handles.popupmenu3,'Value');
B1=get(handles.edit6,'String');
nombre=get(handles.edit1,'String');
xi=str2num(get(handles.edit2,'String'));
xf=str2num(get(handles.edit3,'String'));
n=str2num(get(handles.edit4,'String'));
if Equilibrio==3||Equilibrio==4
    A=str2num(get(handles.edit5,'String'));
    if A<=0 || isempty(A)==1
            f=msgbox('No se ha seleccionado las opciones de forma
correcta','Error:','error');
    end
end
if metodo==1 || Grafica==1 || Equilibrio==1 || isempty(nombre)==1 ||
isempty(xi)==1 || isempty(xf)==1 || isempty(n)==1 || xf<=0 || n<=0 ||
isempty(B1)==1
    f=msgbox('No se ha seleccionado las opciones de forma
correcta','Error:','error');
elseif metodo==2 && Grafica==2 && Equilibrio==2
    f=msgbox({'Regla de los trapecios';'Curva de Pesos';'Aguas
tranquilas'},'Opciones seleccionadas:','help');
elseif metodo==2 && Grafica==3 && Equilibrio==2
    f=msgbox({'Regla de los trapecios';'Curva de Empuje';'Aguas
tranquilas'},'Opciones seleccionadas:','help');
elseif metodo==2 && Grafica==4 && Equilibrio==2
    f=msgbox({'Regla de los trapecios';'Curva de Pesos y
Empujes';'Aguas tranquilas'},'Opciones seleccionadas:','help');
elseif metodo==2 && Grafica==5 && Equilibrio==2
    f=msgbox({'Regla de los trapecios';'Curva de Cargas';'Aguas
tranquilas'},'Opciones seleccionadas:','help');
elseif metodo==2 && Grafica==6 && Equilibrio==2
    f=msgbox({'Regla de los trapecios';'Curva de Fuerzas
Cortantes';'Aguas tranquilas'},'Opciones seleccionadas:','help');
elseif metodo==2 && Grafica==7 && Equilibrio==2
    f=msgbox({'Regla de los trapecios';'Curva de Momentos
Flectores';'Aguas tranquilas'},'Opciones seleccionadas:','help');
elseif metodo==3 && Grafica==2 && Equilibrio==2
    f=msgbox({'Primera Regla de Simpson';'Curva de Pesos';'Aguas
tranquilas'},'Opciones seleccionadas:','help');
elseif metodo==3 && Grafica==3 && Equilibrio==2
    f=msgbox({'Primera Regla de Simpson';'Curva de Empuje';'Aguas
tranquilas'},'Opciones seleccionadas:','help');
elseif metodo==3 && Grafica==4 && Equilibrio==2
```

```
    f=msgbox({'Primera Regla de Simpson';'Curva de Pesos y
Empujes';'Aguas tranquilas'},'Opciones seleccionadas:','help');
elseif metodo==3 && Grafica==5 && Equilibrio==2
    f=msgbox({'Primera Regla de Simpson';'Curva de Cargas';'Aguas
tranquilas'},'Opciones seleccionadas:','help');
elseif metodo==3 && Grafica==6 && Equilibrio==2
    f=msgbox({'Primera Regla de Simpson';'Curva de Fuerzas
Cortantes';'Aguas tranquilas'},'Opciones seleccionadas:','help');
elseif metodo==3 && Grafica==7 && Equilibrio==2
    f=msgbox({'Primera Regla de Simpson';'Curva de Momentos
Flectores';'Aguas tranquilas'},'Opciones seleccionadas:','help');
elseif metodo==2 && Grafica==2 && Equilibrio==3
    f=msgbox({'Regla de los trapecios';'Curva de Pesos';'Buque sobre
el seno de una ola'},'Opciones seleccionadas:','help');
elseif metodo==2 && Grafica==3 && Equilibrio==3
    f=msgbox({'Regla de los trapecios';'Curva de Empuje';'Buque sobre
el seno de una ola'},'Opciones seleccionadas:','help');
elseif metodo==2 && Grafica==4 && Equilibrio==3
    f=msgbox({'Regla de los trapecios';'Curva de Pesos y
Empujes';'Buque sobre el seno de una ola'},'Opciones
seleccionadas:','help');
elseif metodo==2 && Grafica==5 && Equilibrio==3
    f=msgbox({'Regla de los trapecios';'Curva de Cargas';'Buque sobre
el seno de una ola'},'Opciones seleccionadas:','help');
elseif metodo==2 && Grafica==6 && Equilibrio==3
    f=msgbox({'Regla de los trapecios';'Curva de Fuerzas
Cortantes';'Buque sobre el seno de una ola'},'Opciones
seleccionadas:','help');
elseif metodo==2 && Grafica==7 && Equilibrio==3
    f=msgbox({'Regla de los trapecios';'Curva de Momentos
Flectores';'Buque sobre el seno de una ola'},'Opciones
seleccionadas:','help');
elseif metodo==3 && Grafica==2 && Equilibrio==3
    f=msgbox({'Primera Regla de Simpson';'Curva de Pesos';'Buque sobre
el seno de una ola'},'Opciones seleccionadas:','help');
elseif metodo==3 && Grafica==3 && Equilibrio==3
    f=msgbox({'Primera Regla de Simpson';'Curva de Empuje';'Buque
sobre el seno de una ola'},'Opciones seleccionadas:','help');
elseif metodo==3 && Grafica==4 && Equilibrio==3
    f=msgbox({'Primera Regla de Simpson';'Curva de Pesos y
Empujes';'Buque sobre el seno de una ola'},'Opciones
seleccionadas:','help');
elseif metodo==3 && Grafica==5 && Equilibrio==3
    f=msgbox({'Primera Regla de Simpson';'Curva de Cargas';'Buque
sobre el seno de una ola'},'Opciones seleccionadas:','help');
elseif metodo==3 && Grafica==6 && Equilibrio==3
    f=msgbox({'Primera Regla de Simpson';'Curva de Fuerzas
Cortantes';'Buque sobre el seno de una ola'},'Opciones
seleccionadas:','help');
elseif metodo==3 && Grafica==7 && Equilibrio==3
    f=msgbox({'Primera Regla de Simpson';'Curva de Momentos
Flectores';'Buque sobre el seno de una ola'},'Opciones
seleccionadas:','help');
elseif metodo==2 && Grafica==2 && Equilibrio==4
    f=msgbox({'Regla de los trapecios';'Curva de Pesos';'Buque sobre
la cresta de una ola'},'Opciones seleccionadas:','help');
elseif metodo==2 && Grafica==3 && Equilibrio==4
    f=msgbox({'Regla de los trapecios';'Curva de Empuje';'Buque sobre
la cresta de una ola'},'Opciones seleccionadas:','help');
elseif metodo==2 && Grafica==4 && Equilibrio==4
```

```
    f=msgbox({'Regla de los trapecios';'Curva de Pesos y
Empujes';'Buque sobre la cresta de una ola'},'Opciones
seleccionadas:','help');
elseif metodo==2 && Grafica==5 && Equilibrio==4
    f=msgbox({'Regla de los trapecios';'Curva de Cargas';'Buque sobre
la cresta de una ola'},'Opciones seleccionadas:','help');
elseif metodo==2 && Grafica==6 && Equilibrio==4
    f=msgbox({'Regla de los trapecios';'Curva de Fuerzas
Cortantes';'Buque sobre la cresta de una ola'},'Opciones
seleccionadas:','help');
elseif metodo==2 && Grafica==7 && Equilibrio==4
    f=msgbox({'Regla de los trapecios';'Curva de Momentos
Flectores';'Buque sobre la cresta de una ola'},'Opciones
seleccionadas:','help');
elseif metodo==3 && Grafica==2 && Equilibrio==4
    f=msgbox({'Primera Regla de Simpson';'Curva de Pesos';'Buque sobre
la cresta de una ola'},'Opciones seleccionadas:','help');
elseif metodo==3 && Grafica==3 && Equilibrio==4
    f=msgbox({'Primera Regla de Simpson';'Curva de Empuje';'Buque
sobre la cresta de una ola'},'Opciones seleccionadas:','help');
elseif metodo==3 && Grafica==4 && Equilibrio==4
    f=msgbox({'Primera Regla de Simpson';'Curva de Pesos y
Empujes';'Buque sobre la cresta de una ola'},'Opciones
seleccionadas:','help');
elseif metodo==3 && Grafica==5 && Equilibrio==4
    f=msgbox({'Primera Regla de Simpson';'Curva de Cargas';'Buque
sobre la cresta de una ola'},'Opciones seleccionadas:','help');
elseif metodo==3 && Grafica==6 && Equilibrio==4
    f=msgbox({'Primera Regla de Simpson';'Curva de Fuerzas
Cortantes';'Buque sobre la cresta de una ola'},'Opciones
seleccionadas:','help');
elseif metodo==3 && Grafica==7 && Equilibrio==4
    f=msgbox({'Primera Regla de Simpson';'Curva de Momentos
Flectores';'Buque sobre la cresta de una ola'},'Opciones
seleccionadas:','help');
end
if metodo~=1 && Grafica~=1 && Equilibrio~=1
    set(handles.pushbutton3,'Enable','on');
    set(handles.pushbutton1,'Enable','on');
end


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Graficar
global Grafica
global metodo
global n
global nombrearch
global peso
global emp
global secc
global q
global ftrapro
global mtrapro
global xi
global xf
global Comprobar
```

```
global fsimpro
global msimpro
global secc_fuerza
global secc_mom
global Equilibrio
global B
global A
Graficar=get(hObject,'Value');
nombrearch=get(handles.edit1,'String');
xi_1=get(handles.edit2,'String');
xi=str2num(xi_1);
xf_1=get(handles.edit3,'String');
xf=str2num(xf_1);
n_1=get(handles.edit4,'String');
n=str2num(n_1);
B_1=get(handles.edit6,'String');
B=str2num(B_1);
if Equilibrio==3||Equilibrio==4
    A_ola=get(handles.edit5,'String');
    A=str2num(A_ola);
end

if Graficar==1 && Comprobar==0
    r=questdlg('¿Está seguro de continuar sin comprobar los datos
seleccionados?','Alerta','Si','No','No');
    if strcmp(r,'No')%Esta función compara dos strings
        return;
    end
end
if Grafica==2 && metodo==2 && Graficar==1 || Grafica==2 && metodo==3
&& Graficar==1
    [q,peso,emp,secc]=cargasro(xi,xf,n,nombrearch);
    axes(handles.axes2);
    handles.plot1=plot(handles.axes2,secc,-peso,'-o');
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Peso [t/m]');title('Resultado de la
Curva de Pesos');
    set(handles.plot1,'HitTest','off');

set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    if abs(min(peso))>abs(max(peso))
        max_peso=-min(peso);
    else
        max_peso=-max(peso);
    end
    Sol_peso=['Peso max.: ' num2str(max_peso) ' [t/m]'];
    set(handles.text8,'String',Sol_peso);
    return
elseif Grafica==3 && metodo==2 && Graficar==1 && Equilibrio==2 ||
Grafica==3 && metodo==3 && Graficar==1 && Equilibrio==2
    [emp,secc]=empuje(xi,xf,n,nombrearch);
    axes(handles.axes2);
    handles.plot1=plot(handles.axes2,secc,emp,'-o');
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Empuje [t/m]');title('Resultado de la
Curva de Empuje');
    set(handles.plot1,'HitTest','off');
```

```
set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    if abs(min(emp))>abs(max(emp))
        max_emp=min(emp);
    else
        max_emp=max(emp);
    end
    Sol_emp=['Empuje max.: ' num2str(max_emp) ' [t/m]'];
    set(handles.text9,'String',Sol_emp);
    return
elseif Grafica==3 && metodo==2 && Graficar==1 && Equilibrio==3 ||
Grafica==3 && metodo==3 && Graficar==1 && Equilibrio==3
    [emp,secc]=empuje_seno(A,xi,xf,n,nombrearch,B);
    axes(handles.axes2);
    handles.plot1=plot(handles.axes2,secc,emp,'-o');
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Empuje [t/m]');title('Resultado de la
Curva de Empuje');
    set(handles.plot1,'HitTest','off');

set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    if abs(min(emp))>abs(max(emp))
        max_emp=min(emp);
    else
        max_emp=max(emp);
    end
    Sol_emp=['Empuje max.: ' num2str(max_emp) ' [t/m]'];
    set(handles.text9,'String',Sol_emp);
    return
elseif Grafica==3 && metodo==2 && Graficar==1 && Equilibrio==4 ||
Grafica==3 && metodo==3 && Graficar==1 && Equilibrio==4
    [emp,secc]=empuje_cresta(A,xi,xf,n,nombrearch,B);
    axes(handles.axes2);
    handles.plot1=plot(handles.axes2,secc,emp,'-o');
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Empuje [t/m]');title('Resultado de la
Curva de Empuje');
    set(handles.plot1,'HitTest','off');

set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    if abs(min(emp))>abs(max(emp))
        max_emp=min(emp);
    else
        max_emp=max(emp);
    end
    Sol_emp=['Empuje max.: ' num2str(max_emp) ' [t/m]'];
    set(handles.text9,'String',Sol_emp);
    return
elseif Grafica==4 && metodo==2 && Graficar==1 && Equilibrio==2 ||
Grafica==4 && metodo==3 && Graficar==1 && Equilibrio==2
    [q,peso,emp,secc]=cargasro(xi,xf,n,nombrearch);
    axes(handles.axes2);
    handles.plot1=plot(handles.axes2,secc,-emp,'-o');
    hold on
    handles.plot2=plot(handles.axes2,secc,-peso,'-*r');
    hold off
```

```
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Empuje [t/m]');title('Resultado de la
Curva de Pesos y Empuje');
    set(handles.plot1,'HitTest','off');

set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    set(handles.plot2,'HitTest','off');

set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    if abs(min(peso))>abs(max(peso))
        max_peso=-min(peso);
    else
        max_peso=-max(peso);
    end
    if abs(min(emp))>abs(max(emp))
        max_emp=min(emp);
    else
        max_emp=max(emp);
    end
    Sol_peso=['Peso max.: ' num2str(max_peso) ' [t/m]'];
    Sol_emp=['Empuje max.: ' num2str(max_emp) ' [t/m]'];
    set(handles.text8,'String',Sol_peso);
    set(handles.text9,'String',Sol_emp);
    return
elseif Grafica==4 && metodo==2 && Graficar==1 && Equilibrio==3 ||
Grafica==4 && metodo==3 && Graficar==1 && Equilibrio==3
    [q,peso,emp,secc]=cargasro_seno(A,xi,xf,n,nombrearch,B);
    axes(handles.axes2);
    handles.plot1=plot(handles.axes2,secc,-emp,'-o');
    hold on
    handles.plot2=plot(handles.axes2,secc,-peso,'-*r');
    hold off
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Empuje [t/m]');title('Resultado de la
Curva de Pesos y Empuje');
    set(handles.plot1,'HitTest','off');

set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    set(handles.plot2,'HitTest','off');

set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    if abs(min(peso))>abs(max(peso))
        max_peso=-min(peso);
    else
        max_peso=-max(peso);
    end
    if abs(min(emp))>abs(max(emp))
        max_emp=min(emp);
    else
        max_emp=max(emp);
    end
    Sol_peso=['Peso max.: ' num2str(max_peso) ' [t/m]'];
    Sol_emp=['Empuje max.: ' num2str(max_emp) ' [t/m]'];
    set(handles.text8,'String',Sol_peso);
    set(handles.text9,'String',Sol_emp);
```

```
    return
elseif Grafica==4 && metodo==2 && Graficar==1 && Equilibrio==4 ||
Grafica==4 && metodo==3 && Graficar==1 && Equilibrio==4
    [q,peso,emp,secc]=cargasro_cresta(A,xi,xf,n,nombrearch,B);
    axes(handles.axes2);
    handles.plot1=plot(handles.axes2,secc,-emp,'-o');
    hold on
    handles.plot2=plot(handles.axes2,secc,-peso,'-*r');
    hold off
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Empuje [t/m]');title('Resultado de la
Curva de Pesos y Empuje');
    set(handles.plot1,'HitTest','off');

set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    set(handles.plot2,'HitTest','off');

set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    if abs(min(peso))>abs(max(peso))
        max_peso=-min(peso);
    else
        max_peso=-max(peso);
    end
    if abs(min(emp))>abs(max(emp))
        max_emp=min(emp);
    else
        max_emp=max(emp);
    end
    Sol_peso=['Peso max.: ' num2str(max_peso) ' [t/m]'];
    Sol_emp=['Empuje max.: ' num2str(max_emp) ' [t/m]'];
    set(handles.text8,'String',Sol_peso);
    set(handles.text9,'String',Sol_emp);
    return
elseif Grafica==5 && metodo==2 && Graficar==1 && Equilibrio==2 ||
Grafica==5 && metodo==3 && Graficar==1 && Equilibrio==2
    [q,peso,emp,secc]=cargasro(xi,xf,n,nombrearch);
    axes(handles.axes2);
    handles.plot1=plot(handles.axes2,secc,q,'-o');
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Carga [t/m]');title('Resultado de la
Curva de Cargas');
    set(handles.plot1,'HitTest','off');

set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    if abs(min(peso))>abs(max(peso))
        max_peso=-min(peso);
    else
        max_peso=-max(peso);
    end
    if abs(min(emp))>abs(max(emp))
        max_emp=min(emp);
    else
        max_emp=max(emp);
    end
    if abs(min(q))>abs(max(q))
        max_carga=min(q);
```

```
    else
        max_carga=max(q);
    end
    Sol_peso=['Peso max.: ' num2str(max_peso) ' [t/m]'];
    Sol_emp=['Empuje max.: ' num2str(max_emp) ' [t/m]'];
    Sol_carga=['Carga max.: ' num2str(max_carga) ' [t/m]'];
    set(handles.text8,'String',Sol_peso);
    set(handles.text9,'String',Sol_emp);
    set(handles.text10,'String',Sol_carga);
    return
elseif Grafica==5 && metodo==2 && Graficar==1 && Equilibrio==3 ||
Grafica==5 && metodo==3 && Graficar==1 && Equilibrio==3
    [q,peso,emp,secc]=cargasro_seno(A,xi,xf,n,nombrearch,B);
    axes(handles.axes2);
    handles.plot1=plot(handles.axes2,secc,q,'-o');
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Carga [t/m]');title('Resultado de la
Curva de Cargas');
    set(handles.plot1,'HitTest','off');

set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    if abs(min(peso))>abs(max(peso))
        max_peso=-min(peso);
    else
        max_peso=-max(peso);
    end
    if abs(min(emp))>abs(max(emp))
        max_emp=min(emp);
    else
        max_emp=max(emp);
    end
    if abs(min(q))>abs(max(q))
        max_carga=min(q);
    else
        max_carga=max(q);
    end
    Sol_peso=['Peso max.: ' num2str(max_peso) ' [t/m]'];
    Sol_emp=['Empuje max.: ' num2str(max_emp) ' [t/m]'];
    Sol_carga=['Carga max.: ' num2str(max_carga) ' [t/m]'];
    set(handles.text8,'String',Sol_peso);
    set(handles.text9,'String',Sol_emp);
    set(handles.text10,'String',Sol_carga);
    return
elseif Grafica==5 && metodo==2 && Graficar==1 && Equilibrio==4 ||
Grafica==5 && metodo==3 && Graficar==1 && Equilibrio==4
    [q,peso,emp,secc]=cargasro_cresta(A,xi,xf,n,nombrearch,B);
    axes(handles.axes2);
    handles.plot1=plot(handles.axes2,secc,q,'-o');
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Carga [t/m]');title('Resultado de la
Curva de Cargas');
    set(handles.plot1,'HitTest','off');

set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    if abs(min(peso))>abs(max(peso))
        max_peso=-min(peso);
    else
```

```
        max_peso=-max(peso);
    end
    if abs(min(emp))>abs(max(emp))
        max_emp=min(emp);
    else
        max_emp=max(emp);
    end
    if abs(min(q))>abs(max(q))
        max_carga=min(q);
    else
        max_carga=max(q);
    end
    Sol_peso=['Peso max.: ' num2str(max_peso) ' [t/m]'];
    Sol_emp=['Empuje max.: ' num2str(max_emp) ' [t/m]'];
    Sol_carga=['Carga max.: ' num2str(max_carga) ' [t/m]'];
    set(handles.text8,'String',Sol_peso);
    set(handles.text9,'String',Sol_emp);
    set(handles.text10,'String',Sol_carga);
    return
elseif Grafica==6 && metodo==2 && Graficar==1 && Equilibrio==2
    [ftrapro,q,peso,emp,secc]=trapeciosfro(xi,xf,n,nombrearch);
    axes(handles.axes2);
    handles.plot1=plot(handles.axes2,secc,ftrapro,'-o');
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Fuerza Cortante
[t]');title('Resultado de la Curva de Fuerzas Cortantes');
    set(handles.plot1,'HitTest','off');

set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    if abs(min(peso))>abs(max(peso))
        max_peso=-min(peso);
    else
        max_peso=-max(peso);
    end
    if abs(min(emp))>abs(max(emp))
        max_emp=min(emp);
    else
        max_emp=max(emp);
    end
    if abs(min(q))>abs(max(q))
        max_carga=min(q);
    else
        max_carga=max(q);
    end
    if abs(min(ftrapro))>abs(max(ftrapro))
        max_fuerza=min(ftrapro);
    else
        max_fuerza=max(ftrapro);
    end
    error_fuerza=abs(ftrapro(end)/max_fuerza)*100;
    Sol_peso=['Peso max.: ' num2str(max_peso) ' [t/m]'];
    Sol_emp=['Empuje max.: ' num2str(max_emp) ' [t/m]'];
    Sol_carga=['Carga max.: ' num2str(max_carga) ' [t/m]'];
    Sol_fuerza=['Fuerza Cortante max.: ' num2str(max_fuerza) ' [t]'];
    Sol_errorf=['Error en el extremo de FC: ' num2str(error_fuerza) '
%'];
    set(handles.text8,'String',Sol_peso);
    set(handles.text9,'String',Sol_emp);
    set(handles.text10,'String',Sol_carga);
```

```
    set(handles.text11,'String',Sol_fuerza);
    set(handles.text16,'String',Sol_errorf);
    return
elseif Grafica==6 && metodo==2 && Graficar==1 && Equilibrio==3

[ftrapro,q,peso,emp,secc]=trapeciosfro_seno(A,xi,xf,n,nombrearch,B);
    axes(handles.axes2);
    handles.plot1=plot(handles.axes2,secc,ftrapro,'-o');
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Fuerza Cortante
[t]');title('Resultado de la Curva de Fuerzas Cortantes');
    set(handles.plot1,'HitTest','off');

set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    if abs(min(peso))>abs(max(peso))
        max_peso=-min(peso);
    else
        max_peso=-max(peso);
    end
    if abs(min(emp))>abs(max(emp))
        max_emp=min(emp);
    else
        max_emp=max(emp);
    end
    if abs(min(q))>abs(max(q))
        max_carga=min(q);
    else
        max_carga=max(q);
    end
    if abs(min(ftrapro))>abs(max(ftrapro))
        max_fuerza=min(ftrapro);
    else
        max_fuerza=max(ftrapro);
    end
    error_fuerza=abs(ftrapro(end)/max_fuerza)*100;
    Sol_peso=['Peso max.: ' num2str(max_peso) ' [t/m]'];
    Sol_emp=['Empuje max.: ' num2str(max_emp) ' [t/m]'];
    Sol_carga=['Carga max.: ' num2str(max_carga) ' [t/m]'];
    Sol_fuerza=['Fuerza Cortante max.: ' num2str(max_fuerza) ' [t]'];
    Sol_errorf=['Error en el extremo de FC: ' num2str(error_fuerza) '
%'];
    set(handles.text8,'String',Sol_peso);
    set(handles.text9,'String',Sol_emp);
    set(handles.text10,'String',Sol_carga);
    set(handles.text11,'String',Sol_fuerza);
    set(handles.text16,'String',Sol_errorf);
    return
elseif Grafica==6 && metodo==2 && Graficar==1 && Equilibrio==4

[ftrapro,q,peso,emp,secc]=trapeciosfro_cresta(A,xi,xf,n,nombrearch,B);
    axes(handles.axes2);
    handles.plot1=plot(handles.axes2,secc,ftrapro,'-o');
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Fuerza Cortante
[t]');title('Resultado de la Curva de Fuerzas Cortantes');
    set(handles.plot1,'HitTest','off');
```

```
set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    if abs(min(peso))>abs(max(peso))
        max_peso=-min(peso);
    else
        max_peso=-max(peso);
    end
    if abs(min(emp))>abs(max(emp))
        max_emp=min(emp);
    else
        max_emp=max(emp);
    end
    if abs(min(q))>abs(max(q))
        max_carga=min(q);
    else
        max_carga=max(q);
    end
    if abs(min(ftrapro))>abs(max(ftrapro))
        max_fuerza=min(ftrapro);
    else
        max_fuerza=max(ftrapro);
    end
    error_fuerza=abs(ftrapro(end)/max_fuerza)*100;
    Sol_peso=['Peso max.: ' num2str(max_peso) ' [t/m]'];
    Sol_emp=['Empuje max.: ' num2str(max_emp) ' [t/m]'];
    Sol_carga=['Carga max.: ' num2str(max_carga) ' [t/m]'];
    Sol_fuerza=['Fuerza Cortante max.: ' num2str(max_fuerza) ' [t]'];
    Sol_errorf=['Error en el extremo de FC: ' num2str(error_fuerza) '
%'];
    set(handles.text8,'String',Sol_peso);
    set(handles.text9,'String',Sol_emp);
    set(handles.text10,'String',Sol_carga);
    set(handles.text11,'String',Sol_fuerza);
    set(handles.text16,'String',Sol_errorf);
    return
elseif Grafica==7 && metodo==2 && Graficar==1 && Equilibrio==2

[mtrapro,ftrapro,q,peso,emp,secc]=trapeciosmro(xi,xf,n,nombrearch);
    axes(handles.axes2);
    handles.plot1=plot(handles.axes2,secc,mtrapro,'-o');
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Momento Flector
[t·m]');title('Resultado de la Curva de Momentos Flectores');
    set(handles.plot1,'HitTest','off');

set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    if abs(min(peso))>abs(max(peso))
        max_peso=-min(peso);
    else
        max_peso=-max(peso);
    end
    if abs(min(emp))>abs(max(emp))
        max_emp=min(emp);
    else
        max_emp=max(emp);
    end
    if abs(min(q))>abs(max(q))
        max_carga=min(q);
```

```
    else
        max_carga=max(q);
    end
    if abs(min(ftrapro))>abs(max(ftrapro))
        max_fuerza=min(ftrapro);
    else
        max_fuerza=max(ftrapro);
    end
    if abs(min(mtrapro))>abs(max(mtrapro))
        max_mom=min(mtrapro);
    else
        max_mom=max(mtrapro);
    end
    error_fuerza=abs(ftrapro(end)/max_fuerza)*100;
    error_mom=abs(mtrapro(end)/max_mom)*100;
    Sol_peso=['Peso max.: ' num2str(max_peso) ' [t/m]'];
    Sol_emp=['Empuje max.: ' num2str(max_emp) ' [t/m]'];
    Sol_carga=['Carga max.: ' num2str(max_carga) ' [t/m]'];
    Sol_fuerza=['Fuerza Cortante max.: ' num2str(max_fuerza) ' [t]'];
    Sol_mom=['Momento Flector max.: ' num2str(max_mom) ' [t·m]'];
    Sol_errorf=['Error en el extremo de FC: ' num2str(error_fuerza) '
%'];
    Sol_errorm=['Error en el extremo de MF: ' num2str(error_mom) '
%'];
    set(handles.text8,'String',Sol_peso);
    set(handles.text9,'String',Sol_emp);
    set(handles.text10,'String',Sol_carga);
    set(handles.text11,'String',Sol_fuerza);
    set(handles.text12,'String',Sol_mom);
    set(handles.text16,'String',Sol_errorf);
    set(handles.text17,'String',Sol_errorm);
    return
elseif Grafica==7 && metodo==2 && Graficar==1 && Equilibrio==3

[mtrapro,ftrapro,q,peso,emp,secc]=trapeciosmro_seno(A,xi,xf,n,nombrear
ch,B);
    axes(handles.axes2);
    handles.plot1=plot(handles.axes2,secc,mtrapro,'-o');
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Momento Flector
[t·m]');title('Resultado de la Curva de Momentos Flectores');
    set(handles.plot1,'HitTest','off');

set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    if abs(min(peso))>abs(max(peso))
        max_peso=-min(peso);
    else
        max_peso=-max(peso);
    end
    if abs(min(emp))>abs(max(emp))
        max_emp=min(emp);
    else
        max_emp=max(emp);
    end
    if abs(min(q))>abs(max(q))
        max_carga=min(q);
    else
        max_carga=max(q);
    end
```

```
    if abs(min(ftrapro))>abs(max(ftrapro))
        max_fuerza=min(ftrapro);
    else
        max_fuerza=max(ftrapro);
    end
    if abs(min(mtrapro))>abs(max(mtrapro))
        max_mom=min(mtrapro);
    else
        max_mom=max(mtrapro);
    end
    error_fuerza=abs(ftrapro(end)/max_fuerza)*100;
    error_mom=abs(mtrapro(end)/max_mom)*100;
    Sol_peso=['Peso max.: ' num2str(max_peso) ' [t/m]'];
    Sol_emp=['Empuje max.: ' num2str(max_emp) ' [t/m]'];
    Sol_carga=['Carga max.: ' num2str(max_carga) ' [t/m]'];
    Sol_fuerza=['Fuerza Cortante max.: ' num2str(max_fuerza) ' [t]'];
    Sol_mom=['Momento Flector max.: ' num2str(max_mom) ' [t·m]'];
    Sol_errorf=['Error en el extremo de FC: ' num2str(error_fuerza) '
%'];
    Sol_errorm=['Error en el extremo de MF: ' num2str(error_mom) '
%'];
    set(handles.text8,'String',Sol_peso);
    set(handles.text9,'String',Sol_emp);
    set(handles.text10,'String',Sol_carga);
    set(handles.text11,'String',Sol_fuerza);
    set(handles.text12,'String',Sol_mom);
    set(handles.text16,'String',Sol_errorf);
    set(handles.text17,'String',Sol_errorm);
    return
elseif Grafica==7 && metodo==2 && Graficar==1 && Equilibrio==4

[mtrapro,ftrapro,q,peso,emp,secc]=trapeciosmro_cresta(A,xi,xf,n,nombre
arch,B);
    axes(handles.axes2);
    handles.plot1=plot(handles.axes2,secc,mtrapro,'-o');
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Momento Flector
[t·m]');title('Resultado de la Curva de Momentos Flectores');
    set(handles.plot1,'HitTest','off');

set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    if abs(min(peso))>abs(max(peso))
        max_peso=-min(peso);
    else
        max_peso=-max(peso);
    end
    if abs(min(emp))>abs(max(emp))
        max_emp=min(emp);
    else
        max_emp=max(emp);
    end
    if abs(min(q))>abs(max(q))
        max_carga=min(q);
    else
        max_carga=max(q);
    end
    if abs(min(ftrapro))>abs(max(ftrapro))
        max_fuerza=min(ftrapro);
    else
```

```
        max_fuerza=max(ftrapro);
    end
    if abs(min(mtrapro))>abs(max(mtrapro))
        max_mom=min(mtrapro);
    else
        max_mom=max(mtrapro);
    end
    error_fuerza=abs(ftrapro(end)/max_fuerza)*100;
    error_mom=abs(mtrapro(end)/max_mom)*100;
    Sol_peso=['Peso max.: ' num2str(max_peso) ' [t/m]'];
    Sol_emp=['Empuje max.: ' num2str(max_emp) ' [t/m]'];
    Sol_carga=['Carga max.: ' num2str(max_carga) ' [t/m]'];
    Sol_fuerza=['Fuerza Cortante max.: ' num2str(max_fuerza) ' [t]'];
    Sol_mom=['Momento Flector max.: ' num2str(max_mom) ' [t·m]'];
    Sol_errorf=['Error en el extremo de FC: ' num2str(error_fuerza) '
%'];
    Sol_errorm=['Error en el extremo de MF: ' num2str(error_mom) '
%'];
    set(handles.text8,'String',Sol_peso);
    set(handles.text9,'String',Sol_emp);
    set(handles.text10,'String',Sol_carga);
    set(handles.text11,'String',Sol_fuerza);
    set(handles.text12,'String',Sol_mom);
    set(handles.text16,'String',Sol_errorf);
    set(handles.text17,'String',Sol_errorm);
    return
elseif Grafica==6 && metodo==3 && Graficar==1 && Equilibrio==2

[fsimpro,q,peso,emp,secc,secc_fuerza]=segsimpsonfro(xi,xf,n,nombrearch
);
    axes(handles.axes2);
    handles.plot1=plot(handles.axes2,secc_fuerza,fsimpro,'-o');
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Fuerza Cortante
[t]');title('Resultado de la Curva de Fuerzas Cortantes');
    set(handles.plot1,'HitTest','off');

set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    if abs(min(peso))>abs(max(peso))
        max_peso=-min(peso);
    else
        max_peso=-max(peso);
    end
    if abs(min(emp))>abs(max(emp))
        max_emp=min(emp);
    else
        max_emp=max(emp);
    end
    if abs(min(q))>abs(max(q))
        max_carga=min(q);
    else
        max_carga=max(q);
    end
    if abs(min(fsimpro))>abs(max(fsimpro))
        max_fuerza=min(fsimpro);
    else
        max_fuerza=max(fsimpro);
    end
    error_fuerza=abs(fsimpro(end)/max_fuerza)*100;
```

```
    Sol_peso=['Peso max.: ' num2str(max_peso) ' [t/m]'];
    Sol_emp=['Empuje max.: ' num2str(max_emp) ' [t/m]'];
    Sol_carga=['Carga max.: ' num2str(max_carga) ' [t/m]'];
    Sol_fuerza=['Fuerza Cortante max.: ' num2str(max_fuerza) ' [t]'];
    Sol_errorf=['Error en el extremo de FC: ' num2str(error_fuerza) '
%'];
    set(handles.text8,'String',Sol_peso);
    set(handles.text9,'String',Sol_emp);
    set(handles.text10,'String',Sol_carga);
    set(handles.text11,'String',Sol_fuerza);
    set(handles.text16,'String',Sol_errorf);
    return
elseif Grafica==6 && metodo==3 && Graficar==1 && Equilibrio==3

[fsimpro,q,peso,emp,secc,secc_fuerza]=segsimpsonfro_seno(A,xi,xf,n,nom
brearch,B);
    axes(handles.axes2);
    handles.plot1=plot(handles.axes2,secc_fuerza,fsimpro,'-o');
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Fuerza Cortante
[t]');title('Resultado de la Curva de Fuerzas Cortantes');
    set(handles.plot1,'HitTest','off');

set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    if abs(min(peso))>abs(max(peso))
        max_peso=-min(peso);
    else
        max_peso=-max(peso);
    end
    if abs(min(emp))>abs(max(emp))
        max_emp=min(emp);
    else
        max_emp=max(emp);
    end
    if abs(min(q))>abs(max(q))
        max_carga=min(q);
    else
        max_carga=max(q);
    end
    if abs(min(ftrapro))>abs(max(fsimpro))
        max_fuerza=min(fsimpro);
    else
        max_fuerza=max(fsimpro);
    end
    error_fuerza=abs(fsimpro(end)/max_fuerza)*100;
    Sol_peso=['Peso max.: ' num2str(max_peso) ' [t/m]'];
    Sol_emp=['Empuje max.: ' num2str(max_emp) ' [t/m]'];
    Sol_carga=['Carga max.: ' num2str(max_carga) ' [t/m]'];
    Sol_fuerza=['Fuerza Cortante max.: ' num2str(max_fuerza) ' [t]'];
    Sol_errorf=['Error en el extremo de FC: ' num2str(error_fuerza) '
%'];
    set(handles.text8,'String',Sol_peso);
    set(handles.text9,'String',Sol_emp);
    set(handles.text10,'String',Sol_carga);
    set(handles.text11,'String',Sol_fuerza);
    set(handles.text16,'String',Sol_errorf);
    return
elseif Grafica==6 && metodo==3 && Graficar==1 && Equilibrio==4
```

```
[fsimpro,q,peso,emp,secc,secc_fuerza]=segsimpsonfro_cresta(A,xi,xf,n,n
ombrearch,B);
    axes(handles.axes2);
    handles.plot1=plot(handles.axes2,secc_fuerza,fsimpro,'-o');
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Fuerza Cortante
[t]');title('Resultado de la Curva de Fuerzas Cortantes');
    set(handles.plot1,'HitTest','off');

set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    if abs(min(peso))>abs(max(peso))
        max_peso=-min(peso);
    else
        max_peso=-max(peso);
    end
    if abs(min(emp))>abs(max(emp))
        max_emp=min(emp);
    else
        max_emp=max(emp);
    end
    if abs(min(q))>abs(max(q))
        max_carga=min(q);
    else
        max_carga=max(q);
    end
    if abs(min(fsimpro))>abs(max(fsimpro))
        max_fuerza=min(fsimpro);
    else
        max_fuerza=max(fsimpro);
    end
    error_fuerza=abs(fsimpro(end)/max_fuerza)*100;
    Sol_peso=['Peso max.: ' num2str(max_peso) ' [t/m]'];
    Sol_emp=['Empuje max.: ' num2str(max_emp) ' [t/m]'];
    Sol_carga=['Carga max.: ' num2str(max_carga) ' [t/m]'];
    Sol_fuerza=['Fuerza Cortante max.: ' num2str(max_fuerza) ' [t]'];
    Sol_errorf=['Error en el extremo de FC: ' num2str(error_fuerza) '
%'];
    set(handles.text8,'String',Sol_peso);
    set(handles.text9,'String',Sol_emp);
    set(handles.text10,'String',Sol_carga);
    set(handles.text11,'String',Sol_fuerza);
    set(handles.text16,'String',Sol_errorf);
    return
elseif Grafica==7 && metodo==3 && Graficar==1 && Equilibrio==2

[msimpro,fsimpro,q,peso,emp,secc,secc_fuerza,secc_mom]=segsimpsonmro(A
,xi,xf,n,nombrearch,B);
    axes(handles.axes2);
    handles.plot1=plot(handles.axes2,secc_mom,msimpro,'-o');
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Momento Flector
[t·m]');title('Resultado de la Curva de Momentos Flectores');
    set(handles.plot1,'HitTest','off');

set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    if abs(min(peso))>abs(max(peso))
```

```
        max_peso=-min(peso);
    else
        max_peso=-max(peso);
    end
    if abs(min(emp))>abs(max(emp))
        max_emp=min(emp);
    else
        max_emp=max(emp);
    end
    if abs(min(q))>abs(max(q))
        max_carga=min(q);
    else
        max_carga=max(q);
    end
    if abs(min(fsimpro))>abs(max(fsimpro))
        max_fuerza=min(fsimpro);
    else
        max_fuerza=max(fsimpro);
    end
    if abs(min(msimpro))>abs(max(msimpro))
        max_mom=min(msimpro);
    else
        max_mom=max(msimpro);
    end
    error_fuerza=abs(fsimpro(end)/max_fuerza)*100;
    error_mom=abs(msimpro(end)/max_mom)*100;
    Sol_peso=['Peso max.: ' num2str(max_peso) ' [t/m]'];
    Sol_emp=['Empuje max.: ' num2str(max_emp) ' [t/m]'];
    Sol_carga=['Carga max.: ' num2str(max_carga) ' [t/m]'];
    Sol_fuerza=['Fuerza Cortante max.: ' num2str(max_fuerza) ' [t]'];
    Sol_mom=['Momento Flector max.: ' num2str(max_mom) ' [t·m]'];
    Sol_errorf=['Error en el extremo de FC: ' num2str(error_fuerza) '
%'];
    Sol_errorm=['Error en el extremo de MF: ' num2str(error_mom) '
%'];
    set(handles.text8,'String',Sol_peso);
    set(handles.text9,'String',Sol_emp);
    set(handles.text10,'String',Sol_carga);
    set(handles.text11,'String',Sol_fuerza);
    set(handles.text12,'String',Sol_mom);
    set(handles.text16,'String',Sol_errorf);
    set(handles.text17,'String',Sol_errorm);
    return
elseif Grafica==7 && metodo==3 && Graficar==1 && Equilibrio==3

[msimpro,fsimpro,q,peso,emp,secc,secc_fuerza,secc_mom]=segsimpsonmro_s
eno(A,xi,xf,n,nombrearch,B);
    axes(handles.axes2);
    handles.plot1=plot(handles.axes2,secc_mom,msimpro,'-o');
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Momento Flector
[t·m]');title('Resultado de la Curva de Momentos Flectores');
    set(handles.plot1,'HitTest','off');

set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    if abs(min(peso))>abs(max(peso))
        max_peso=-min(peso);
    else
        max_peso=-max(peso);
```

```
    end
    if abs(min(emp))>abs(max(emp))
        max_emp=min(emp);
    else
        max_emp=max(emp);
    end
    if abs(min(q))>abs(max(q))
        max_carga=min(q);
    else
        max_carga=max(q);
    end
    if abs(min(fsimpro))>abs(max(fsimpro))
        max_fuerza=min(fsimpro);
    else
        max_fuerza=max(fsimpro);
    end
    if abs(min(msimpro))>abs(max(msimpro))
        max_mom=min(msimpro);
    else
        max_mom=max(msimpro);
    end
    error_fuerza=abs(fsimpro(end)/max_fuerza)*100;
    error_mom=abs(msimpro(end)/max_mom)*100;
    Sol_peso=['Peso max.: ' num2str(max_peso) ' [t/m]'];
    Sol_emp=['Empuje max.: ' num2str(max_emp) ' [t/m]'];
    Sol_carga=['Carga max.: ' num2str(max_carga) ' [t/m]'];
    Sol_fuerza=['Fuerza Cortante max.: ' num2str(max_fuerza) ' [t]'];
    Sol_mom=['Momento Flector max.: ' num2str(max_mom) ' [t·m]'];
    Sol_errorf=['Error en el extremo de FC: ' num2str(error_fuerza) '
%'];
    Sol_errorm=['Error en el extremo de MF: ' num2str(error_mom) '
%'];
    set(handles.text8,'String',Sol_peso);
    set(handles.text9,'String',Sol_emp);
    set(handles.text10,'String',Sol_carga);
    set(handles.text11,'String',Sol_fuerza);
    set(handles.text12,'String',Sol_mom);
    set(handles.text16,'String',Sol_errorf);
    set(handles.text17,'String',Sol_errorm);
    return
elseif Grafica==7 && metodo==3 && Graficar==1 && Equilibrio==4

[msimpro,fsimpro,q,peso,emp,secc,secc_fuerza,secc_mom]=segsimpsonmro_c
resta(A,xi,xf,n,nombrearch,B);
    axes(handles.axes2);
    handles.plot1=plot(handles.axes2,secc_mom,msimpro,'-o');
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Momento Flector
[t·m]');title('Resultado de la Curva de Momentos Flectores');
    set(handles.plot1,'HitTest','off');

set(handles.axes2,'ButtonDownFcn',@(s,e)axes2_ButtonDownFcn(s,e,handle
s));
    if abs(min(peso))>abs(max(peso))
        max_peso=-min(peso);
    else
        max_peso=-max(peso);
    end
    if abs(min(emp))>abs(max(emp))
        max_emp=min(emp);
```

```
    else
        max_emp=max(emp);
    end
    if abs(min(q))>abs(max(q))
        max_carga=min(q);
    else
        max_carga=max(q);
    end
    if abs(min(fsimpro))>abs(max(fsimpro))
        max_fuerza=min(fsimpro);
    else
        max_fuerza=max(fsimpro);
    end
    if abs(min(msimpro))>abs(max(msimpro))
        max_mom=min(msimpro);
    else
        max_mom=max(msimpro);
    end
    error_fuerza=abs(fsimpro(end)/max_fuerza)*100;
    error_mom=abs(msimpro(end)/max_mom)*100;
    Sol_peso=['Peso max.: ' num2str(max_peso) ' [t/m]'];
    Sol_emp=['Empuje max.: ' num2str(max_emp) ' [t/m]'];
    Sol_carga=['Carga max.: ' num2str(max_carga) ' [t/m]'];
    Sol_fuerza=['Fuerza Cortante max.: ' num2str(max_fuerza) ' [t]'];
    Sol_mom=['Momento Flector max.: ' num2str(max_mom) ' [t·m]'];
    Sol_errorf=['Error en el extremo de FC: ' num2str(error_fuerza) '
%'];
    Sol_errorm=['Error en el extremo de MF: ' num2str(error_mom) '
%'];
    set(handles.text8,'String',Sol_peso);
    set(handles.text9,'String',Sol_emp);
    set(handles.text10,'String',Sol_carga);
    set(handles.text11,'String',Sol_fuerza);
    set(handles.text12,'String',Sol_mom);
    set(handles.text16,'String',Sol_errorf);
    set(handles.text17,'String',Sol_errorm);
    return
elseif Grafica==1||metodo==1||Equilibrio==1
    f=msgbox('No se ha seleccionado las opciones de forma
correcta','Error:','error');
    return
end


% --- Executes on mouse press over axes background.
function axes2_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to axes2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global metodo
global Grafica
global peso
global emp
global secc
global q
global ftrapro
global mtrapro
global fsimpro
global msimpro
global secc_fuerza
global secc_mom
```

```
% global Mantener_grafica
handles.newfig=figure('Name','Representación
gráfica','NumberTitle','off');
if Grafica==2 && metodo==2 || Grafica==2 && metodo==3
    plot(secc(1:end),-peso,'-o');axis tight;
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Peso [t/m]');title('Resultado de la
Curva de Pesos');
    set(handles.newfig,'Units','pixels');
    screenSize=get(0,'ScreenSize');
    position=get(handles.output,'Position');
    position(1)=(screenSize(3)-position(3));
    position(2)=(screenSize(4)-position(4));
    set(handles.newfig,'Position',position);
    movegui(handles.newfig,'center');
elseif Grafica==3 && metodo==2 || Grafica==3 && metodo==3
    plot(secc(1:end),emp,'-o');axis tight;
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Empuje [t/m]');title('Resultado de la
Curva de Empuje');
    set(handles.newfig,'Units','pixels');
    screenSize=get(0,'ScreenSize');
    position=get(handles.output,'Position');
    position(1)=(screenSize(3)-position(3))/2;
    position(2)=(screenSize(4)-position(4))/2;
    set(handles.newfig,'Position',position);
    movegui(handles.newfig,'center');
elseif Grafica==4 && metodo==2 || Grafica==4 && metodo==3
    plot(secc,-emp,'-o');axis tight;
    hold on
    plot(secc,-peso,'-*r');axis tight;
    hold off
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Empuje [t/m]');title('Resultado de la
Curva de Pesos y Empuje');
    set(handles.newfig,'Units','pixels');
    screenSize=get(0,'ScreenSize');
    position=get(handles.output,'Position');
    position(1)=(screenSize(3)-position(3))/2;
    position(2)=(screenSize(4)-position(4))/2;
    set(handles.newfig,'Position',position);
    movegui(handles.newfig,'center');
elseif Grafica==5 && metodo==2 || Grafica==5 && metodo==3
    plot(secc,q,'-o');axis tight;
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Carga [t/m]');title('Resultado de la
Curva de Cargas');
    set(handles.newfig,'Units','pixels');
    screenSize=get(0,'ScreenSize');
    position=get(handles.output,'Position');
    position(1)=(screenSize(3)-position(3))/2;
    position(2)=(screenSize(4)-position(4))/2;
    set(handles.newfig,'Position',position);
    movegui(handles.newfig,'center');
elseif Grafica==6 && metodo==2
    plot(secc,ftrapro,'-o');axis tight;
```

```
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Fuerza Cortante
[t]');title('Resultado de la Curva de Fuerzas Cortantes');
    set(handles.newfig,'Units','pixels');
    screenSize=get(0,'ScreenSize');
    position=get(handles.output,'Position');
    position(1)=(screenSize(3)-position(3))/2;
    position(2)=(screenSize(4)-position(4))/2;
    set(handles.newfig,'Position',position);
    movegui(handles.newfig,'center');
elseif Grafica==7 && metodo==2
    plot(secc,mtrapro,'-o');
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Momento Flector
[t·m]');title('Resultado de la Curva de Momentos Flectores');
    set(handles.newfig,'Units','pixels');
    screenSize=get(0,'ScreenSize');
    position=get(handles.output,'Position');
    position(1)=(screenSize(3)-position(3))/2;
    position(2)=(screenSize(4)-position(4))/2;
    set(handles.newfig,'Position',position);
    movegui(handles.newfig,'center');
elseif Grafica==6 && metodo==3
    plot(secc_fuerza,fsimpro,'-o');
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Momento Flector
[t·m]');title('Resultado de la Curva de Momentos Flectores');
    set(handles.newfig,'Units','pixels');
    screenSize=get(0,'ScreenSize');
    position=get(handles.output,'Position');
    position(1)=(screenSize(3)-position(3))/2;
    position(2)=(screenSize(4)-position(4))/2;
    set(handles.newfig,'Position',position);
    movegui(handles.newfig,'center');
elseif Grafica==7 && metodo==3
    plot(secc_mom,msimpro,'-o');
    grid on;ax=gca;ax.GridLineStyle = ':';ax.GridAlpha =
0.5;ax.Layer='top';
    xlabel('Eslora [m]');ylabel('Momento Flector
[t·m]');title('Resultado de la Curva de Momentos Flectores');
    set(handles.newfig,'Units','pixels');
    screenSize=get(0,'ScreenSize');
    position=get(handles.output,'Position');
    position(1)=(screenSize(3)-position(3))/2;
    position(2)=(screenSize(4)-position(4))/2;
    set(handles.newfig,'Position',position);
    movegui(handles.newfig,'center');
end


% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox1
```

```
% --- Executes during object creation, after setting all properties.
function checkbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to checkbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called


% --- Executes during object creation, after setting all properties.
function text13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5
as a double


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6
as a double


% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

11.4 Programa para la obtención de la curva de Cargas

```
%% Programa para obtener la representación gráfica de la curva de
cargas para el equilibrio en aguas tranquilas
% Cargas = Empuje - Peso
function [q,peso,emp,secc]=cargasro(xi,xf,n,nombrearch)
%% Variables de entrada:
    % xi => valor del extremo inferior del intervalo
    % xf => valor del extremo superior del intervalo
    % n => numero de subintervalos
%% Variables de salida:
    % q => valor de la carga a lo largo de la eslora
%% Inicializamos
h=1;
q=[];
datos=importdata(nombrearch); % Importamos los datos desde un archivo,
puede ser .txt o .xlsx, aunque para la aplicación práctica usamos
.xlsx
%% Obtención del Área por el método de los trapecios
% [emp,secc]=empuje(xi,xf,n,nombrearch);
for i=1:n
    peso(i)=-datos.data.Peso(i,2);
    emp(i)=datos.data.Peso(i,3);
    q(i)=emp(i)+peso(i);
    secc(i)=datos.data.Peso(i,1);
end
%% Representación gráfica de los pesos y empujes
% plot(secc(1:end),emp(1:end),'-*');
% hold on;
% grid on;  ax = gca; ax.GridLineStyle = ':'; ax.GridAlpha = 0.5;
ax.Layer = 'top';
% xlabel('Eslora [m]');
% ylabel('Carga [t/m]');
% title('Resultado Curva de Pesos y Empujes');
% plot(secc(1:end),peso(1:end),'-x');
% grid on;  ax = gca; ax.GridLineStyle = ':'; ax.GridAlpha = 0.5;
ax.Layer = 'top';
% xlabel('Eslora [m]');
% ylabel('Carga [t/m]');
% legend('Empujes','Pesos');
%% Representación gráfica de la solución
% figure;plot(secc(1:end),q(1:end),'-o');
% grid on;  ax = gca; ax.GridLineStyle = ':'; ax.GridAlpha = 0.5;
ax.Layer = 'top';
% xlabel('Eslora [m]');
% ylabel('Cargas [t/m]');
% title('Resultado Curva de Cargas');
% legend('Curva de Cargas','Location','Northeast');
```

```
%% Programa para obtener la representación gráfica de la curva de
cargas para la condición de equilibrio sobre cresta
% Cargas = Empuje - Peso
function [q,peso,emp,secc]=cargasro_cresta(A,xi,xf,n,nombrearch,B)
%% Variables de entrada:
    % xi => valor del extremo inferior del intervalo
    % xf => valor del extremo superior del intervalo
    % n => numero de subintervalos
%% Variables de salida:
    % q => valor de la carga a lo largo de la eslora
%% Inicializamos
h=1;
q=[];
datos=importdata(nombrearch); % Importamos los datos desde un archivo,
puede ser .txt o .xlsx, aunque para la aplicación práctica usamos
.xlsx
%% Obtención del Área por el método de los trapecios
[emp,secc]=empuje_cresta(A,xi,xf,n,nombrearch,B);
for i=1:n
    peso(i)=-datos.data.Peso(i,2);
    q(i)=emp(i)+peso(i);
    secc(i)=datos.data.Peso(i,1);
end
%% Representación gráfica de los pesos y empujes
% plot(secc(1:end),emp(1:end),'-*');
% hold on;
% grid on;  ax = gca; ax.GridLineStyle = ':'; ax.GridAlpha = 0.5;
ax.Layer = 'top';
% xlabel('Eslora [m]');
% ylabel('Carga [t/m]');
% title('Resultado Curva de Pesos y Empujes');
% plot(secc(1:end),peso(1:end),'-x');
% grid on;  ax = gca; ax.GridLineStyle = ':'; ax.GridAlpha = 0.5;
ax.Layer = 'top';
% xlabel('Eslora [m]');
% ylabel('Carga [t/m]');
% legend('Empujes','Pesos');
%% Representación gráfica de la solución
% figure;plot(secc(1:end),q(1:end),'-o');
% grid on;  ax = gca; ax.GridLineStyle = ':'; ax.GridAlpha = 0.5;
ax.Layer = 'top';
% xlabel('Eslora [m]');
% ylabel('Cargas [t/m]');
% title('Resultado Curva de Cargas');
% legend('Curva de Cargas','Location','Northeast');


%% Programa para obtener la representación gráfica de la curva de
cargas para la condición de equilibrio sobre seno
% Cargas = Empuje - Peso
function [q,peso,emp,secc]=cargasro_seno(A,xi,xf,n,nombrearch,B)
%% Variables de entrada:
    % xi => valor del extremo inferior del intervalo
    % xf => valor del extremo superior del intervalo
    % n => numero de subintervalos
%% Variables de salida:
    % q => valor de la carga a lo largo de la eslora
%% Inicializamos
h=1;
q=[];
```

```
datos=importdata(nombrearch); % Importamos los datos desde un archivo,
puede ser .txt o .xlsx, aunque para la aplicación práctica usamos
.xlsx
%% Obtención del Área por el método de los trapecios
[emp,secc]=empuje_seno(A,xi,xf,n,nombrearch,B);
for i=1:n
    peso(i)=-datos.data.Peso(i,2);
    q(i)=emp(i)+peso(i);
    secc(i)=datos.data.Peso(i,1);
end
%% Representación gráfica de los pesos y empujes
% plot(secc(1:end),emp(1:end),'-*');
% hold on;
% grid on;  ax = gca; ax.GridLineStyle = ':'; ax.GridAlpha = 0.5;
ax.Layer = 'top';
% xlabel('Eslora [m]');
% ylabel('Carga [t/m]');
% title('Resultado Curva de Pesos y Empujes');
% plot(secc(1:end),peso(1:end),'-x');
% grid on;  ax = gca; ax.GridLineStyle = ':'; ax.GridAlpha = 0.5;
ax.Layer = 'top';
% xlabel('Eslora [m]');
% ylabel('Carga [t/m]');
% legend('Empujes','Pesos');
%% Representación gráfica de la solución
% figure;plot(secc(1:end),q(1:end),'-o');
% grid on;  ax = gca; ax.GridLineStyle = ':'; ax.GridAlpha = 0.5;
ax.Layer = 'top';
% xlabel('Eslora [m]');
% ylabel('Cargas [t/m]');
% title('Resultado Curva de Cargas');
% legend('Curva de Cargas','Location','Northeast');
```

11.5 Programa para la obtención de la curva de Empuje

```
%% Programa para el cálculo del empuje para obtener el equilibrio de
pesos para el equilibrio en aguas tranquilas
% Cargas = Empuje - Peso
% Equilibrio de Momentos respecto de Popa: SUM_Mpp=0
function [emp,secc]=empuje(xi,xf,n,nombrearch)
%% Variables de entrada:
    % xi => valor del extremo inferior del intervalo
    % xf => valor del extremo superior del intervalo
    % n => numero de subintervalos
%% Variables de salida:
    % q => valor de la carga a lo largo de la eslora
%% Inicializamos
h=1;
xi;
L=xf;
q=[];
sum_peso=0;
sum_mompp=0;
datos=importdata(nombrearch); % Importamos los datos desde un archivo,
puede ser .txt o .xlsx, aunque para la aplicación práctica usamos
.xlsx
for i=1:n
    peso(i)=datos.data.Peso(i,2);
    secc(i)=datos.data.Peso(i,1);
end
```

```
desp=sum(peso);%El desplazamiento será la suma de todos los pesos que
existan en el buque
%% Para cumplir con la condición del equilibrio de fuerzas, sabemos
que si
% el buque se encuentra en equilibrio sobre el mar en calma, debido a
la
% distribución de las cargas sobre el buque, éste se encontrará
trimado,
% para este caso en particular, se da que: Empuje en
popa=a=(desp*2/L)-b
% Si sustituimos este valor en el equilibrio de momentos, obtenemos:
% Empuje en proa=b=(2*(desp*XG+(desp*L/3))/L^2), de esta forma
sabremos los empujes y por
% tanto los calados para esta condición
%% Calculamos el empuje
for i=1:n
    sum_peso=sum_peso+peso(i);
    sum_mompp=sum_mompp+peso(i)*secc(i); %Realizamos el sumatorio de
momentos en popa
    XG=sum_mompp/sum_peso;%Obtenemos la posición del centro de
gravedad del buque a partir de la curva de áreas
end
% syms a b
% Ec_equilibrio=[sum_peso-(a+b)*L/2==0,peso*secc'-((b*L^2)/2+(a-
b)/2*(L^2)/3)==0];
% [emp_pp emp_pr]=solve(Ec_equilibrio,[a,b]);
emp_pp=((6*(peso*secc')/L^2-(2*sum_peso/L))); %b
emp_pr=(sum_peso*2/L)-emp_pp; %a
for i=1:n
    emp(i)=emp_pp+(secc(i)-secc(1))/(secc(end)-secc(1))*(emp_pr-
emp_pp);
end
% figure;plot(secc(1:end),empuje(1:end),'-o');
% grid on;  ax = gca; ax.GridLineStyle = ':'; ax.GridAlpha = 0.5;
ax.Layer = 'top';
% xlabel('Eslora [m]');
% ylabel('Empuje [t/m]');
% title('Resultado Curva de Empuje');


%% Programa para el cálculo del empuje para obtener el equilibrio de
pesos en la situación de quebranto
% Cargas = Empuje - Peso
% Equilibrio de Momentos respecto de Popa: SUM_Mpp=0
function [emp,secc]=empuje_cresta(A,xi,xf,n,nombrearch,B)
%% Variables de entrada:
    % xi => valor del extremo inferior del intervalo
    % xf => valor del extremo superior del intervalo
    % n => numero de subintervalos
    % A => Altura de la ola
%% Variables de salida:
    % q => valor de la carga a lo largo de la eslora
%% Inicializamos
h=1;
L=xf;
lambda=L;%La longitud de la ola coincidirá con la eslora del buque
delta=0;%Desfase respecto de la ola inicial
q=[];
sum_peso=0;
sum_mompp=0;
```

```
datos=importdata(nombrearch); % Importamos los datos desde un archivo,
puede ser .txt o .xlsx, aunque para la aplicación práctica usamos
.xlsx
for i=1:n
    peso(i)=datos.data.Peso(i,2);
    secc(i)=datos.data.Peso(i,1);
end
desp=sum(peso);%El desplazamiento será la suma de todos los pesos que
existan en el buque
%% Para cumplir con la condición del equilibrio de fuerzas, sabemos
que si
% el buque se encuentra en equilibrio sobre el mar en calma, debido a
la
% distribución de las cargas sobre el buque, éste se encontrará
trimado,
% para este caso en particular, se da que: Empuje en
popa=a=(desp*2/L)-b
% Si sustituimos este valor en el equilibrio de momentos, obtenemos:
% Empuje en proa=b=(2*(desp*XG+(desp*L/3))/L^2), de esta forma
sabremos los empujes y por
% tanto los calados para esta condición
%% Calculamos el empuje
for i=1:n-1
    if secc(i+1)==secc(i)
        sum_peso=sum_peso;
        sum_mompp=sum_mompp;
    else
        sum_peso=sum_peso+peso(i)*h;
        sum_mompp=sum_mompp+peso(i)*secc(i); %Realizamos el sumatorio
de momentos en popa
    end
end
XG=sum_mompp/sum_peso;%Obtenemos la posición del centro de gravedad
del buque a partir de la curva de áreas
sum_peso=abs(sum_peso);
%Necesitamos calcular el calado mínimo (a) mediante el equilibrio de
pesos
%y empuje
a=(sum_peso-(A/2*L*B))/(L*B);
for i=1:n

empuje(i)=a*B+A/2*B+A/2*B*cos((2*pi()*secc(i)/lambda)+pi()/2);%Ecuació
n de la ola senoidal con el seno en el centro
end
emp=empuje;
% figure;plot(secc(1:end),empuje(1:end),'-o');
% grid on;  ax = gca; ax.GridLineStyle = ':'; ax.GridAlpha = 0.5;
ax.Layer = 'top';
% xlabel('Eslora [m]');
% ylabel('Empuje [t/m]');
% title('Resultado Curva de Empuje');


%% Programa para el cálculo del empuje para obtener el equilibrio de
pesos en la situación de arrufo
% Cargas = Empuje - Peso
% Equilibrio de Momentos respecto de Popa: SUM_Mpp=0
function [emp,secc]=empuje_seno(A,xi,xf,n,nombrearch,B)
%% Variables de entrada:
    % xi => valor del extremo inferior del intervalo
    % xf => valor del extremo superior del intervalo
```

```
    % n => numero de subintervalos
    % A => Altura de la ola
%% Variables de salida:
    % q => valor de la carga a lo largo de la eslora
%% Inicializamos
h=1;
lambda=xf;%La longitud de la ola coincidirá con la eslora del buque
delta=0;%Desfase respecto de la ola inicial
q=[];
sum_peso=0;
sum_mompp=0;
datos=importdata(nombrearch); % Importamos los datos desde un archivo,
puede ser .txt o .xlsx, aunque para la aplicación práctica usamos
.xlsx
for i=1:n
    peso(i)=datos.data.Peso(i,2);
    secc(i)=datos.data.Peso(i,1);
end
desp=sum(peso);%El desplazamiento será la suma de todos los pesos que
existan en el buque
%% Para cumplir con la condición del equilibrio de fuerzas, sabemos
que si
% el buque se encuentra en equilibrio sobre el mar en calma, debido a
la
% distribución de las cargas sobre el buque, éste se encontrará
trimado,
% para este caso en particular, se da que: Empuje en
popa=a=(desp*2/L)-b
% Si sustituimos este valor en el equilibrio de momentos, obtenemos:
% Empuje en proa=b=(2*(desp*XG+(desp*L/3))/L^2), de esta forma
sabremos los empujes y por
% tanto los calados para esta condición
%% Calculamos el empuje
for i=1:n-1
    if secc(i+1)==secc(i)
        sum_peso=sum_peso;
        sum_mompp=sum_mompp;
    else
        sum_peso=sum_peso+peso(i)*h;
        sum_mompp=sum_mompp+peso(i)*secc(i); %Realizamos el sumatorio
de momentos en popa
    end
end
XG=sum_mompp/sum_peso;%Obtenemos la posición del centro de gravedad
del buque a partir de la curva de áreas
sum_peso=abs(sum_peso);
%Necesitamos calcular el calado mínimo (a) mediante el equilibrio de
pesos
%y empuje
a=(sum_peso-(A/2*lambda*B))/(lambda*B);
for i=1:n

empuje(i)=a*B+A/2*B+A/2*B*sin((2*pi()*secc(i)/lambda)+pi()/2);%Ecuació
n de la ola senoidal con el seno en el centro
end
emp=empuje;
% figure;plot(secc(1:end),empuje(1:end),'-o');
% grid on;  ax = gca; ax.GridLineStyle = ':'; ax.GridAlpha = 0.5;
ax.Layer = 'top';
% xlabel('Eslora [m]');
% ylabel('Empuje [t/m]');
```

```
% title('Resultado Curva de Empuje');
```

11.6 Programa para la obtención de la curva de Fuerzas Cortantes mediante el Método de los Trapecios

```
%% Programa para la aplicación del Método de los Trapecios
% Método de los Trapecios: (h/2)*(f0+f1)
function [ftrapro,q,peso,emp,secc]=trapeciosfro(xi,xf,n,nombrearch)
%% Variables de entrada:
    % xi => valor del extremo inferior del intervalo
    % xf => valor del extremo superior del intervalo
    % n => numero de subintervalos
%% Variables de salida:
    % AT => valor del área bajo la curva mediante aproximación por
trapecios
%% Inicializamos
h=1;
i1=1;
j1=1;
% ftrapro(1)=0;
% nombrearch=input('Introduzca el nombre del archivo junto con su
formato:\n','s');
datos=importdata(nombrearch); % Importamos los datos desde un archivo,
puede ser .txt o .xlsx, aunque para la aplicación práctica usamos
.xlsx
%% Obtención del Área por el método de los trapecios
[q,peso,emp,secc]=cargasro(xi,xf,n,nombrearch);
for j=j1:24
    for i=i1:n
        if i==1
            ftrapro(i)=0;
        elseif secc(i)~=secc(i-1)
            ftrapro(i)=(h/2)*(q(i-1)+q(i))+ftrapro(i-1);
        elseif secc(i)==secc(i-1)
            ftrapro(i)=ftrapro(i-1);
            i1=i+1;
            j1=j+1;
            break
        end
    end
end

%% Representación gráfica de la solución
% figure; plot(secc(1:end),ftrapro(1:end),'-og');
% grid on;  ax = gca; ax.GridLineStyle = ':'; ax.GridAlpha = 0.5;
ax.Layer = 'top';
% xlabel('Eslora [m]');
% ylabel('Fuerza Cortante [t]');
% title('Resultado Fuerza Cortante por trapecios');


%% Programa para la aplicación del Método de los Trapecios
% Método de los Trapecios: (h/2)*(f0+f1)
function
[ftrapro,q,peso,emp,secc]=trapeciosfro_cresta(A,xi,xf,n,nombrearch,B)
%% Variables de entrada:
    % xi => valor del extremo inferior del intervalo
    % xf => valor del extremo superior del intervalo
    % n => numero de subintervalos
%% Variables de salida:
```

```
    % AT => valor del área bajo la curva mediante aproximación por
trapecios
%% Inicializamos
h=1;
i1=1;
% ftrapro(1)=0;
% nombrearch=input('Introduzca el nombre del archivo junto con su
formato:\n','s');
datos=importdata(nombrearch); % Importamos los datos desde un archivo,
puede ser .txt o .xlsx, aunque para la aplicación práctica usamos
.xlsx
%% Obtención del Área por el método de los trapecios
[q,peso,emp,secc]=cargasro_cresta(A,xi,xf,n,nombrearch,B);
for j=1:24
    for i=i1:n
        if i==1
            ftrapro(i)=0;
        elseif secc(i)~=secc(i-1)
            ftrapro(i)=(h/2)*(q(i-1)+q(i))+ftrapro(i-1);
        elseif secc(i)==secc(i-1)
            ftrapro(i)=(h/2)*(q(i-1)+q(i))+ftrapro(i-1);
            ftrapro(i+1)=ftrapro(i);
            i1=i+2;
            j=j+1;
            break
        end
    end
end

%% Representación gráfica de la solución
% figure; plot(secc(1:end),ftrapro(1:end),'-og');
% grid on;  ax = gca; ax.GridLineStyle = ':'; ax.GridAlpha = 0.5;
ax.Layer = 'top';
% xlabel('Eslora [m]');
% ylabel('Fuerza Cortante [t]');
% title('Resultado Fuerza Cortante por trapecios');


%% Programa para la aplicación del Método de los Trapecios
% Método de los Trapecios: (h/2)*(f0+f1)
function
[ftrapro,q,peso,emp,secc]=trapeciosfro_seno(A,xi,xf,n,nombrearch,B)
%% Variables de entrada:
    % xi => valor del extremo inferior del intervalo
    % xf => valor del extremo superior del intervalo
    % n => numero de subintervalos
%% Variables de salida:
    % AT => valor del área bajo la curva mediante aproximación por
trapecios
%% Inicializamos
h=1;
i1=1;
% ftrapro(1)=0;
% nombrearch=input('Introduzca el nombre del archivo junto con su
formato:\n','s');
datos=importdata(nombrearch); % Importamos los datos desde un archivo,
puede ser .txt o .xlsx, aunque para la aplicación práctica usamos
.xlsx
%% Obtención del Área por el método de los trapecios
[q,peso,emp,secc]=cargasro_seno(A,xi,xf,n,nombrearch,B);
for j=1:24
    for i=i1:n
```

```
        if i==1
            ftrapro(i)=0;
        elseif secc(i)~=secc(i-1)
            ftrapro(i)=(h/2)*(q(i-1)+q(i))+ftrapro(i-1);
        elseif secc(i)==secc(i-1)
            ftrapro(i)=(h/2)*(q(i-1)+q(i))+ftrapro(i-1);
            ftrapro(i+1)=ftrapro(i);
            i1=i+2;
            j=j+1;
            break
        end
    end
end

%% Representación gráfica de la solución
% figure; plot(secc(1:end),ftrapro(1:end),'-og');
% grid on;  ax = gca; ax.GridLineStyle = ':'; ax.GridAlpha = 0.5;
ax.Layer = 'top';
% xlabel('Eslora [m]');
% ylabel('Fuerza Cortante [t]');
% title('Resultado Fuerza Cortante por trapecios');
```

11.7 Programa para la obtención de la curva de Momentos Flectores mediante el Método de los Trapecios

```
%% Programa para la aplicación del Método de los Trapecios
% Método de los Trapecios: (h/2)*(f0+f1)
function
[mtrapro,ftrapro,q,peso,emp,secc]=trapeciosmro(xi,xf,n,nombrearch)
%% Variables de entrada:
    % xi => valor del extremo inferior del intervalo
    % xf => valor del extremo superior del intervalo
    % n => numero de subintervalos
%% Variables de salida:
    % AT => valor del área bajo la curva mediante aproximación por
trapecios
%% Inicializamos
h=1;
i1=1;
mtrapro(1)=0;
%% Obtención del Área por el método de los trapecios
% [q,peso,emp,secc]=cargasro(xi,xf,n,nombrearch);
[ftrapro,q,peso,emp,secc]=trapeciosfro(xi,xf,n,nombrearch);
for j=1:24
    for i=i1:n
        if i==1
            mtrapro(i)=0;
        elseif secc(i)~=secc(i-1)
            mtrapro(i)=(h/2)*(ftrapro(i-1)+ftrapro(i))+mtrapro(i-1);
        else
            mtrapro(i)=(h/2)*(ftrapro(i-1)+ftrapro(i))+mtrapro(i-1);
            mtrapro(i+1)=mtrapro(i);
            i1=i+2;
            j=j+1;
            break
        end
    end
end
```

```
%% Representación gráfica de la solución
% figure; plot(secc(1:end),mtrapro(1:end),'-or');
% grid on;  ax = gca; ax.GridLineStyle = ':'; ax.GridAlpha = 0.5;
ax.Layer = 'top';
% xlabel('Eslora [m]');
% ylabel('Momento Flector [txm]');
% title('Resultado Momento Flector por trapecios');


%% Programa para la aplicación del Método de los Trapecios
% Método de los Trapecios: (h/2)*(f0+f1)
function
[mtrapro,ftrapro,q,peso,emp,secc]=trapeciosmro_cresta(A,xi,xf,n,nombre
arch,B)
%% Variables de entrada:
    % xi => valor del extremo inferior del intervalo
    % xf => valor del extremo superior del intervalo
    % n => numero de subintervalos
%% Variables de salida:
    % AT => valor del área bajo la curva mediante aproximación por
trapecios
%% Inicializamos
h=1;
i1=1;
mtrapro(1)=0;
%% Obtención del Área por el método de los trapecios
% [q,peso,emp,secc]=cargasro(xi,xf,n,nombrearch);
[ftrapro,q,peso,emp,secc]=trapeciosfro_cresta(A,xi,xf,n,nombrearch,B);
for j=1:24
    for i=i1:n
        if i==1
            mtrapro(i)=0;
        elseif secc(i)~=secc(i-1)
            mtrapro(i)=(h/2)*(ftrapro(i-1)+ftrapro(i))+mtrapro(i-1);
        else
            mtrapro(i)=(h/2)*(ftrapro(i-1)+ftrapro(i))+mtrapro(i-1);
            mtrapro(i+1)=mtrapro(i);
            i1=i+2;
            j=j+1;
            break
        end
    end
end

%% Representación gráfica de la solución
% figure; plot(secc(1:end),mtrapro(1:end),'-or');
% grid on;  ax = gca; ax.GridLineStyle = ':'; ax.GridAlpha = 0.5;
ax.Layer = 'top';
% xlabel('Eslora [m]');
% ylabel('Momento Flector [txm]');
% title('Resultado Momento Flector por trapecios');


%% Programa para la aplicación del Método de los Trapecios
% Método de los Trapecios: (h/2)*(f0+f1)
function
[mtrapro,ftrapro,q,peso,emp,secc]=trapeciosmro_seno(A,xi,xf,n,nombrear
ch,B)
%% Variables de entrada:
```

```
    % xi => valor del extremo inferior del intervalo
    % xf => valor del extremo superior del intervalo
    % n => numero de subintervalos
%% Variables de salida:
    % AT => valor del área bajo la curva mediante aproximación por
trapecios
%% Inicializamos
h=1;
i1=1;
mtrapro(1)=0;
%% Obtención del Área por el método de los trapecios
% [q,peso,emp,secc]=cargasro(xi,xf,n,nombrearch);
[ftrapro,q,peso,emp,secc]=trapeciosfro_seno(A,xi,xf,n,nombrearch,B);
for j=1:24
    for i=i1:n
        if i==1
            mtrapro(i)=0;
        elseif secc(i)~=secc(i-1)
            mtrapro(i)=(h/2)*(ftrapro(i-1)+ftrapro(i))+mtrapro(i-1);
        else
            mtrapro(i)=(h/2)*(ftrapro(i-1)+ftrapro(i))+mtrapro(i-1);
            mtrapro(i+1)=mtrapro(i);
            i1=i+2;
            j=j+1;
            break
        end
    end
end

%% Representación gráfica de la solución
% figure; plot(secc(1:end),mtrapro(1:end),'-or');
% grid on;  ax = gca; ax.GridLineStyle = ':'; ax.GridAlpha = 0.5;
ax.Layer = 'top';
% xlabel('Eslora [m]');
% ylabel('Momento Flector [txm]');
% title('Resultado Momento Flector por trapecios');
```

11.8 Programa para la obtención de la curva de Fuerzas Cortantes mediante el Método de Simpson

```
%% Programa para la aplicación del Método de Simpson
% 1ª Regla del método de Simpson: (h/3)*(f0+4*f1+f2)
% 2ª Regla del método de Simpson: (3*h/8)*(f0+3*f1+3*f2+f3)
function
[fsimpro,q,peso,emp,secc,secc_fuerza]=segsimpsonfro(xi,xf,n,nombrearch
)
%% Variables de entrada:
    % xi => valor del extremo inferior del intervalo
    % xf => valor del extremo superior del intervalo
    % n => numero de subintervalos
%% Variables de salida:
    % AT => valor del área bajo la curva mediante aproximación por
Simpson
%% Inicializamos
h=1;
suma=2;
A=0;
A1=0;
i1=1;
j1=1;
```

```
secc_fuerza=0;
%% Obtención del Área por el método de Simpson
[q,peso,emp,secc]=cargasro(xi,xf,n,nombrearch);
for i=i1:n
    if i==1
        fsimpro(1)=0;
    elseif secc(i-1)==secc(i) && i~=n
        i1=i-1;
        if rem(i1,2)==0 %Si el numero i1 es par, necesito aplicar las
dos reglas de Simpson
            for j=j1:i1
                if j==j1
                    A1=(h/3)*(q(j)+4*q(j+1)+q(j+2))+A1;
                    A(j)=A1;
                    secc_fuerza=[secc_fuerza secc(j+2)];
                elseif j==i1
                    A1=(3*h/8)*(q(j-3)+3*q(j-2)+3*q(j-1)+q(j))+A1;
                    A(j)=A1;
                    secc_fuerza=[secc_fuerza secc(j)];
                    break
                end
            end
            indice=find(A); %Encontramos los valores distintos de cero
para la matriz A y los almacenamos en fsimpro
            for k=2:length(indice)+1
                fsimpro(k)=A(indice(k-1));
            end
            j1=j+1;
        elseif rem(i1,2)==1 %Si el número es impar, puedo aplicar
directamente la primera regla de Simpson
            for j=j1:2:i1
                A1=(h/3)*(q(j-1)+4*q(j)+q(j+1))+A1;
                A(j)=A1;
                secc_fuerza=[secc_fuerza secc(j)];
            end
            indice=find(A); %Encontramos los valores distintos de cero
para la matriz A y los almacenamos en fsimpro
            for k=2:length(indice)+1
                fsimpro(k)=A(indice(k-1));
            end
            j1=j+1;
        end
        suma=suma+1;%Numero de veces que repetimos la iteración
%         secc_fuerza=[secc_fuerza secc(i)]; %Intentamos almacenar el
valor de la sección que estamos calculando para luego graficar
    elseif secc(i)==secc(end) || i==n
        i1=i-1;
        if rem(i1,2)==0 %Si el numero i1 es par, no puedo aplicar de
forma directa Simpson, necesito un paso de trapecios
            for j=j1:2:i1
                if j==j1
                    A1=(h/3)*(q(j-1)+4*q(j)+q(j+1))+A1;
                    secc_fuerza=[secc_fuerza secc(j+2)];
                else
                    A1=(3*h/8)*(q(j-2)+3*q(j-1)+3*q(j)+q(j+1))+A1;
                    secc_fuerza=[secc_fuerza secc(j)];
                end
                A(j)=A1;
            end
            indice=find(A); %Encontramos los valores distintos de cero
para la matriz A y los almacenamos en fsimpro
```

```
            for k=2:length(indice)+1
                fsimpro(k)=A(indice(k-1));
            end
            j1=j+1;
        elseif rem(i1,2)==1 %Si el número es impar, puedo aplicar
directamente Simpson
            for j=j1:2:i1
                A1=(h/3)*(q(j-1)+4*q(j)+q(j+1))+A1;
                A(j)=A1;
                secc_fuerza=[secc_fuerza secc(j)];
            end
            indice=find(A); %Encontramos los valores distintos de cero
para la matriz A y los almacenamos en fsimpro
            for k=2:length(indice)+1
                fsimpro(k)=A(indice(k-1));
            end
            j1=j+1;
        end
        suma=suma+1;%Numero de veces que repetimos la iteración
%         secc_fuerza=[secc_fuerza secc(i)];
    elseif secc(i-1)~=secc(i)
        continue
    end
end
%% Representación gráfica de la solución
% figure; plot(secc_fuerza,fsimpro(1:end),'-or');
% hold on;
% grid on;  ax = gca; ax.GridLineStyle = ':'; ax.GridAlpha = 0.5;
ax.Layer = 'top';
% xlabel('Eslora [m]');
% ylabel('Fuerza Cortante [t]');
% title('Resultado Fuerza Cortante por el Método de Simpson');


%% Programa para la aplicación del Método de Simpson
% 1ª Regla del método de Simpson: (h/3)*(f0+4*f1+f2)
% 2ª Regla del método de Simpson: (3*h/8)*(f0+3*f1+3*f2+f3)
function
[fsimpro,q,peso,emp,secc,secc_fuerza]=segsimpsonfro_cresta(A,xi,xf,n,n
ombrearch,B)
%% Variables de entrada:
    % xi => valor del extremo inferior del intervalo
    % xf => valor del extremo superior del intervalo
    % n => numero de subintervalos
%% Variables de salida:
    % AT => valor del área bajo la curva mediante aproximación por
Simpson
%% Inicializamos
h=1;
suma=2;
A2=0;
A1=0;
i1=1;
j1=1;
secc_fuerza=0;
%% Obtención del Área por el método de Simpson
[q,peso,emp,secc]=cargasro_cresta(A,xi,xf,n,nombrearch,B);
for i=i1:n
    if i==1
        fsimpro(1)=0;
    elseif secc(i-1)==secc(i) && i~=n
```

```
        i1=i-1;
        if rem(i1,2)==0 %Si el numero i1 es par, necesito aplicar las
dos reglas de Simpson
            for j=j1:i1
                if j==j1
                    A1=(h/3)*(q(j)+4*q(j+1)+q(j+2))+A1;
                    A2(j)=A1;
                    secc_fuerza=[secc_fuerza secc(j+2)];
                elseif j==i1
                    A1=(3*h/8)*(q(j-3)+3*q(j-2)+3*q(j-1)+q(j))+A1;
                    A2(j)=A1;
                    secc_fuerza=[secc_fuerza secc(j)];
                    break
                end
            end
            indice=find(A2); %Encontramos los valores distintos de
cero para la matriz A y los almacenamos en fsimpro
            for k=2:length(indice)+1
                fsimpro(k)=A2(indice(k-1));
            end
            j1=j+1;
        elseif rem(i1,2)==1 %Si el número es impar, puedo aplicar
directamente la primera regla de Simpson
            for j=j1:2:i1
                A1=(h/3)*(q(j-1)+4*q(j)+q(j+1))+A1;
                A2(j)=A1;
                secc_fuerza=[secc_fuerza secc(j)];
            end
            indice=find(A2); %Encontramos los valores distintos de
cero para la matriz A y los almacenamos en fsimpro
            for k=2:length(indice)+1
                fsimpro(k)=A2(indice(k-1));
            end
            j1=j+1;
        end
        suma=suma+1;%Numero de veces que repetimos la iteración
%         secc_fuerza=[secc_fuerza secc(i)]; %Intentamos almacenar el
valor de la sección que estamos calculando para luego graficar
    elseif secc(i)==secc(end) || i==n
        i1=i-1;
        if rem(i1,2)==0 %Si el numero i1 es par, no puedo aplicar de
forma directa Simpson, necesito un paso de trapecios
            for j=j1:2:i1
                if j==j1
                    A1=(h/3)*(q(j-1)+4*q(j)+q(j+1))+A1;
                    secc_fuerza=[secc_fuerza secc(j+2)];
                else
                    A1=(3*h/8)*(q(j-2)+3*q(j-1)+3*q(j)+q(j+1))+A1;
                    secc_fuerza=[secc_fuerza secc(j)];
                end
                A2(j)=A1;
            end
            indice=find(A2); %Encontramos los valores distintos de
cero para la matriz A y los almacenamos en fsimpro
            for k=2:length(indice)+1
                fsimpro(k)=A2(indice(k-1));
            end
            j1=j+1;
        elseif rem(i1,2)==1 %Si el número es impar, puedo aplicar
directamente Simpson
            for j=j1:2:i1
```

```
                A1=(h/3)*(q(j-1)+4*q(j)+q(j+1))+A1;
                A2(j)=A1;
                secc_fuerza=[secc_fuerza secc(j)];
            end
            indice=find(A2); %Encontramos los valores distintos de
cero para la matriz A y los almacenamos en fsimpro
            for k=2:length(indice)+1
                fsimpro(k)=A2(indice(k-1));
            end
            j1=j+1;
        end
        suma=suma+1;%Numero de veces que repetimos la iteración
%         secc_fuerza=[secc_fuerza secc(i)];
    elseif secc(i-1)~=secc(i)
        continue
    end
end
%% Representación gráfica de la solución
% figure; plot(secc_fuerza,fsimpro(1:end),'-or');
% hold on;
% grid on;  ax = gca; ax.GridLineStyle = ':'; ax.GridAlpha = 0.5;
ax.Layer = 'top';
% xlabel('Eslora [m]');
% ylabel('Fuerza Cortante [t]');
% title('Resultado Fuerza Cortante por el Método de Simpson');


%% Programa para la aplicación del Método de Simpson
% 1ª Regla del método de Simpson: (h/3)*(f0+4*f1+f2)
% 2ª Regla del método de Simpson: (3*h/8)*(f0+3*f1+3*f2+f3)
function
[fsimpro,q,peso,emp,secc,secc_fuerza]=segsimpsonfro_seno(A,xi,xf,n,nom
brearch,B)
%% Variables de entrada:
    % xi => valor del extremo inferior del intervalo
    % xf => valor del extremo superior del intervalo
    % n => numero de subintervalos
%% Variables de salida:
    % AT => valor del área bajo la curva mediante aproximación por
Simpson
%% Inicializamos
h=1;
suma=2;
A2=0;
A1=0;
i1=1;
j1=1;
secc_fuerza=0;
%% Obtención del Área por el método de Simpson
[q,peso,emp,secc]=cargasro_seno(A,xi,xf,n,nombrearch,B);
for i=i1:n
    if i==1
        fsimpro(1)=0;
    elseif secc(i-1)==secc(i) && i~=n
        i1=i-1;
        if rem(i1,2)==0 %Si el numero i1 es par, necesito aplicar las
dos reglas de Simpson
            for j=j1:i1
                if j==j1
                    A1=(h/3)*(q(j)+4*q(j+1)+q(j+2))+A1;
                    A2(j)=A1;
```

```
                    secc_fuerza=[secc_fuerza secc(j+2)];
                elseif j==i1
                    A1=(3*h/8)*(q(j-3)+3*q(j-2)+3*q(j-1)+q(j))+A1;
                    A2(j)=A1;
                    secc_fuerza=[secc_fuerza secc(j)];
                    break
                end
            end
            indice=find(A2); %Encontramos los valores distintos de
cero para la matriz A y los almacenamos en fsimpro
            for k=2:length(indice)+1
                fsimpro(k)=A2(indice(k-1));
            end
            j1=j+1;
        elseif rem(i1,2)==1 %Si el número es impar, puedo aplicar
directamente la primera regla de Simpson
            for j=j1:2:i1
                A1=(h/3)*(q(j-1)+4*q(j)+q(j+1))+A1;
                A2(j)=A1;
                secc_fuerza=[secc_fuerza secc(j)];
            end
            indice=find(A2); %Encontramos los valores distintos de
cero para la matriz A y los almacenamos en fsimpro
            for k=2:length(indice)+1
                fsimpro(k)=A2(indice(k-1));
            end
            j1=j+1;
        end
        suma=suma+1;%Numero de veces que repetimos la iteración
%         secc_fuerza=[secc_fuerza secc(i)]; %Intentamos almacenar el
valor de la sección que estamos calculando para luego graficar
    elseif secc(i)==secc(end) || i==n
        i1=i-1;
        if rem(i1,2)==0 %Si el numero i1 es par, no puedo aplicar de
forma directa Simpson, necesito un paso de trapecios
            for j=j1:2:i1
                if j==j1
                    A1=(h/3)*(q(j-1)+4*q(j)+q(j+1))+A1;
                    secc_fuerza=[secc_fuerza secc(j+2)];
                else
                    A1=(3*h/8)*(q(j-2)+3*q(j-1)+3*q(j)+q(j+1))+A1;
                    secc_fuerza=[secc_fuerza secc(j)];
                end
                A2(j)=A1;
            end
            indice=find(A2); %Encontramos los valores distintos de
cero para la matriz A y los almacenamos en fsimpro
            for k=2:length(indice)+1
                fsimpro(k)=A2(indice(k-1));
            end
            j1=j+1;
        elseif rem(i1,2)==1 %Si el número es impar, puedo aplicar
directamente Simpson
            for j=j1:2:i1
                A1=(h/3)*(q(j-1)+4*q(j)+q(j+1))+A1;
                A2(j)=A1;
                secc_fuerza=[secc_fuerza secc(j)];
            end
            indice=find(A2); %Encontramos los valores distintos de
cero para la matriz A y los almacenamos en fsimpro
            for k=2:length(indice)+1
```

```
                fsimpro(k)=A2(indice(k-1));
            end
            j1=j+1;
        end
        suma=suma+1;%Numero de veces que repetimos la iteración
%         secc_fuerza=[secc_fuerza secc(i)];
    elseif secc(i-1)~=secc(i)
        continue
    end
end
%% Representación gráfica de la solución
% figure; plot(secc_fuerza,fsimpro(1:end),'-or');
% hold on;
% grid on;  ax = gca; ax.GridLineStyle = ':'; ax.GridAlpha = 0.5;
ax.Layer = 'top';
% xlabel('Eslora [m]');
% ylabel('Fuerza Cortante [t]');
% title('Resultado Fuerza Cortante por el Método de Simpson');
```

11.9 Programa para la obtención de la curva de Momentos Flectores mediante el Método de Simpson

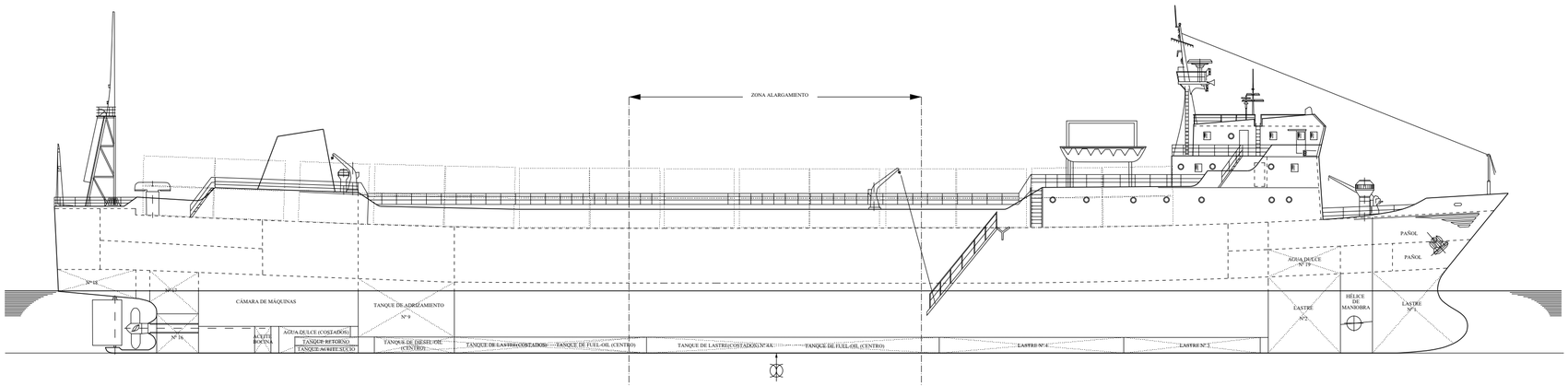
```
%% Programa para la aplicación del Método de Simpson
% 1ª Regla del método de Simpson: (h/3)*(f0+4*f1+f2)
% 2ª Regla del método de Simpson: (3*h/8)*(f0+3*f1+3*f2+f3)
function
[msimpro,fsimpro,q,peso,emp,secc,secc_fuerza,secc_mom]=segsimpsonmro(x
i,xf,n,nombrearch)
%% Variables de entrada:
    % xi => valor del extremo inferior del intervalo
    % xf => valor del extremo superior del intervalo
    % n => numero de subintervalos
%% Variables de salida:
    % AT => valor del área bajo la curva mediante aproximación por
Simpson
%% Inicializamos
h=1;
suma=2;
A=0;
A1=0;
i1=2;
j1=1;
secc_mom=0;
msimpro(1)=0;
%% Obtención del Área por el método de los trapecios
[fsimpro,q,peso,emp,secc,secc_fuerza]=segsimpsonfro(xi,xf,n,nombrearch
);
for i=i1:2:length(fsimpro)-1
    A1=(h/3)*(fsimpro(i-1)+4*fsimpro(i)+fsimpro(i+1))+A1;
    A(i)=A1;
    suma=suma+1;%Numero de veces que repetimos la iteración
end
indice=find(A); %Encontramos los valores distintos de cero para la
matriz A y los almacenamos en fsimpro
[secc_unicas,ind_unicas]=unique(secc);
ind_repetidas=setdiff(1:length(secc),ind_unicas);
secc_repetidas=secc(ind_repetidas);
secc_mom=[0 secc_repetidas secc(end)];
for k=2:length(indice)+1
```

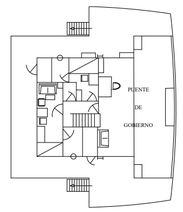
```
    msimpro(k)=A(indice(k-1));
end
%% Representación gráfica de la solución
% figure; plot(secc_mom,msimpro(1:end),'-or');
% hold on;
% grid on;ax=gca;ax.GridLineStyle=':';ax.GridAlpha=0.5;ax.Layer='top';
% xlabel('Eslora [m]');
% ylabel('Momento Flector [t·m]');
% title('Resultado Momento Flector por el Método de Simpson');



%% Programa para la aplicación del Método de Simpson
% 1ª Regla del método de Simpson: (h/3)*(f0+4*f1+f2)
% 2ª Regla del método de Simpson: (3*h/8)*(f0+3*f1+3*f2+f3)
function
[msimpro,fsimpro,q,peso,emp,secc,secc_fuerza,secc_mom]=segsimpsonmro_c
resta(A,xi,xf,n,nombrearch,B)
%% Variables de entrada:
    % xi => valor del extremo inferior del intervalo
    % xf => valor del extremo superior del intervalo
    % n => numero de subintervalos
%% Variables de salida:
    % AT => valor del área bajo la curva mediante aproximación por
Simpson
%% Inicializamos
h=1;
suma=2;
A2=0;
A1=0;
i1=2;
j1=1;
secc_mom=0;
msimpro(1)=0;
%% Obtención del Área por el método de los trapecios
[fsimpro,q,peso,emp,secc,secc_fuerza]=segsimpsonfro_cresta(A,xi,xf,n,n
ombrearch,B);
for i=i1:2:length(fsimpro)-1
    A1=(h/3)*(fsimpro(i-1)+4*fsimpro(i)+fsimpro(i+1))+A1;
    A2(i)=A1;
    suma=suma+1;%Numero de veces que repetimos la iteración
end
indice=find(A2); %Encontramos los valores distintos de cero para la
matriz A y los almacenamos en fsimpro
[secc_unicas,ind_unicas]=unique(secc);
ind_repetidas=setdiff(1:length(secc),ind_unicas);
secc_repetidas=secc(ind_repetidas);
secc_mom=[0 secc_repetidas secc(end)];
for k=2:length(indice)+1
    msimpro(k)=A2(indice(k-1));
end
%% Representación gráfica de la solución
% figure; plot(secc_mom,msimpro(1:end),'-or');
% hold on;
% grid on;ax=gca;ax.GridLineStyle=':';ax.GridAlpha=0.5;ax.Layer='top';
% xlabel('Eslora [m]');
% ylabel('Momento Flector [t·m]');
% title('Resultado Momento Flector por el Método de Simpson');
```

```
%% Programa para la aplicación del Método de Simpson
% 1ª Regla del método de Simpson: (h/3)*(f0+4*f1+f2)
% 2ª Regla del método de Simpson: (3*h/8)*(f0+3*f1+3*f2+f3)
function
[msimpro,fsimpro,q,peso,emp,secc,secc_fuerza,secc_mom]=segsimpsonmro_s
eno(A,xi,xf,n,nombrearch,B)
%% Variables de entrada:
    % xi => valor del extremo inferior del intervalo
    % xf => valor del extremo superior del intervalo
    % n => numero de subintervalos
%% Variables de salida:
    % AT => valor del área bajo la curva mediante aproximación por
Simpson
%% Inicializamos
h=1;
suma=2;
A2=0;
A1=0;
i1=2;
j1=1;
secc_mom=0;
msimpro(1)=0;
%% Obtención del Área por el método de los trapecios
[fsimpro,q,peso,emp,secc,secc_fuerza]=segsimpsonfro_seno(A,xi,xf,n,nom
brearch,B);
for i=i1:2:length(fsimpro)-1
    A1=(h/3)*(fsimpro(i-1)+4*fsimpro(i)+fsimpro(i+1))+A1;
    A2(i)=A1;
    suma=suma+1;%Numero de veces que repetimos la iteración
end
indice=find(A2); %Encontramos los valores distintos de cero para la
matriz A y los almacenamos en fsimpro
[secc_unicas,ind_unicas]=unique(secc);
ind_repetidas=setdiff(1:length(secc),ind_unicas);
secc_repetidas=secc(ind_repetidas);
secc_mom=[0 secc_repetidas secc(end)];
for k=2:length(indice)+1
    msimpro(k)=A2(indice(k-1));
end
%% Representación gráfica de la solución
% figure; plot(secc_mom,msimpro(1:end),'-or');
% hold on;
% grid on;ax=gca;ax.GridLineStyle=':';ax.GridAlpha=0.5;ax.Layer='top';
% xlabel('Eslora [m]');
% ylabel('Momento Flector [t·m]');
% title('Resultado Momento Flector por el Método de Simpson');
```

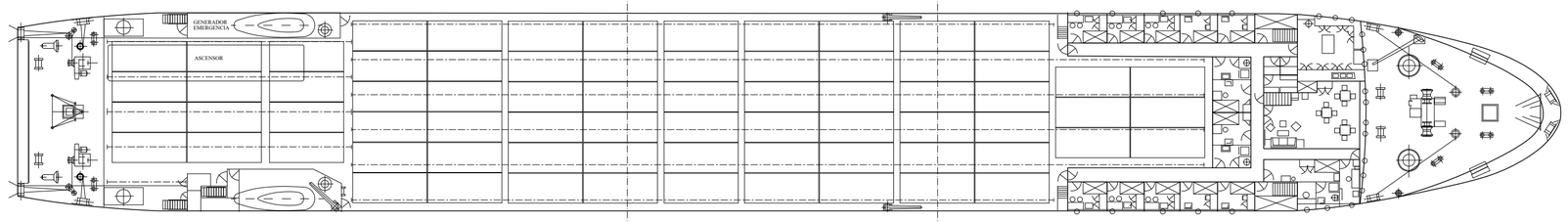
12 Anexo II

En el presente anexo se adjuntan el plano de formas y el plano de distribución general del buque empleado, una vez alargado el cuerpo cilíndrico.

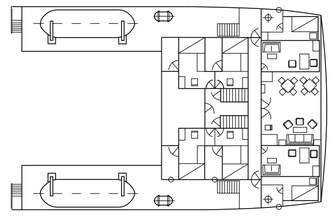




CUBIERTA SUPERIOR

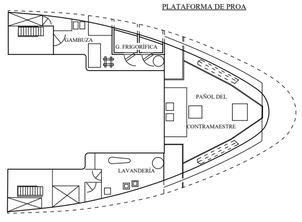


CUBIERTA DE BOTES

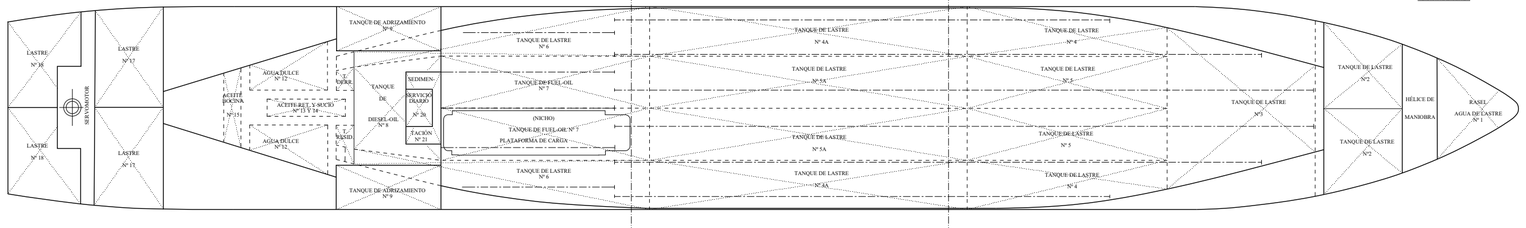


CUBIERTA SEGUNDA

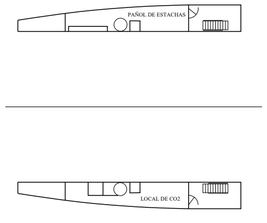
PLATAFORMA DE PROA



DOBLE FONDO



PLATAFORMA DE POPA



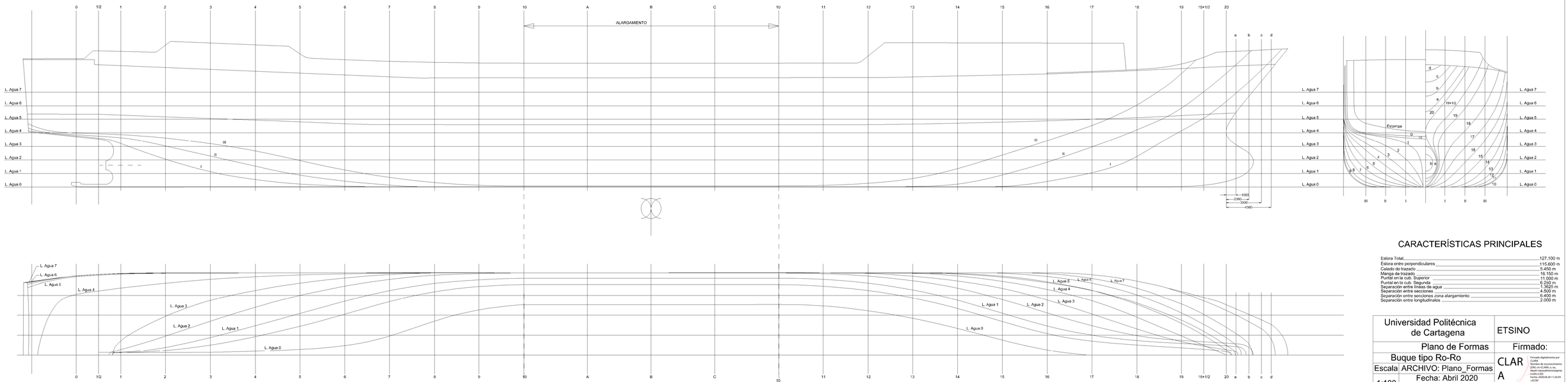
CARACTERÍSTICAS PRINCIPALES

Eslora entre perpendiculares	115,600 m.
Calado de trazado	5,450 m.
Manga de trazado	16,500 m.
Puntal cubierta superior	11,000 m.
Puntal cubierta segunda	6,250 m.
Metros lineales	1.202,500 m.

Universidad Politécnica de Cartagena		ETSINO
PLANO DE DISPOSICIÓN GENERAL		
Buque tipo Ro-Ro		
Escala	ARCHIVO: Disposición	
1:350	FECHA: Abril 2020	
	Nº PLANO: 1	Nº HOJAS: 1

CLARA — Firmado digitalmente por CLARA. Nombre de reconocimiento (DN): cn=CLARA, o, ou, email=clara@mercon.it@gmail.com, c=ES. Fecha: 2020.04.30 11:24:20 +02'00'

Clara Salmerón Bermúdez



CARACTERÍSTICAS PRINCIPALES

Eslora Total	127,100 m
Eslora entre perpendiculares	115,600 m
Calado de trazado	5,450 m
Manga de trazado	16,150 m
Puntal en la cub. Superior	11,000 m
Puntal en la cub. Segunda	6,200 m
Separación entre líneas de agua	0,3625 m
Separación entre secciones	4,500 m
Separación entre secciones zona alargamiento	6,400 m
Separación entre longitudinales	2,000 m

Universidad Politécnica de Cartagena			ETSINO
Plano de Formas			Firmado:
Buque tipo Ro-Ro			CLARA Firmado digitalmente por CLARA. Nombre de reconocimiento (opcional): CN=CLARA, email=clarasalmeron@gmail.com, C=ES. Fecha: 2020.04.30 11:24:39 +02'00'
Escala	ARCHIVO: Plano_Formas		
1:100	Fecha: Abril 2020		
	Plano nº: 1	Nº Hojas: 1	Clara Salmerón Bermúdez

13 Anexo III

En el presente anexo se adjuntan las tablas con los datos correspondientes a los tanques empleadas para la obtención de la curva de cargas del buque.

Principales características de los Tanques					
Tanque	Capacidad		LCG	TCG	VCG
	(m^3)	(T)	(m)	(m)	(m)
Nº 1 Rasel de Proa	94.22	96.57	112.84	0	4.11
Nº 2 Er. Trimado	132.63	135.94	103.86	1.94	4.38
Nº 2 Br. Trimado	132.63	135.94	103.86	-1.94	4.38
Nº 3 Tanque Lastre	103.37	105.96	93.47	0	0.82
Nº 4 Er. Tanque Lastre	46.22	47.38	78.73	5.43	0.87
Nº 4 Br. Tanque Lastre	46.22	47.38	78.73	-5.43	0.87
Nº 5 Er. Tanque Lastre	89.1	91.33	80.16	2.08	0.74
Nº 5 Br. Tanque Lastre	89.1	91.33	80.16	-2.08	0.74
Nº 4A Er. Tanque Lastre	102.41	104.97	59.26	5.66	0.82
Nº 4A Br. Tanque Lastre	102.41	104.97	59.26	-5.66	0.82
Nº 5A Er. Tanque Fuel-Oil	144.61	134.48	59.32	2.1	0.73
Nº 5A Br. Tanque Fuel-Oil	144.61	134.48	59.32	-2.1	0.73
Nº 6 Er. Tanque Lastre	59.16	60.64	38.18	5.2	0.894
Nº 6 Br. Tanque Lastre	59.16	60.64	38.18	5.2	0.894
Nº 7 Er. Tanque Fuel-Oil	51.05	47.47	38.28	2.07	0.39
Nº 7 Br. Tanque Fuel-Oil	94.62	87.99	38.2	-2.09	0.74
Nº 8 Tanque Diesel-Oil	56.88	49.48	26.43	0	0.82
Nº 9 Er. Tanque Adrizamiento	103.69	106.28	25.7	6.09	4.37
Nº 9 Br. Tanque Adrizamiento	103.69	106.28	25.7	-6.09	4.37
Nº 10 Derrames	2.35	2.17	22.04	-2.17	0.96
Nº 11 Residuos	2.35	2.17	22.04	2.17	0.96
Nº 12 Er. Agua Dulce	16.54	16.54	18.05	2.41	1.73
Nº 12 Br. Agua Dulce	16.54	16.54	18.05	-2.41	1.73
Nº 13 Aceite Retorno	6.33	5.85	18.86	0	1.07
Nº 14 Aceite Sucio	5.94	5.49	18.91	0	0.38
Nº 15 Aceite Bocina	2.14	1.98	12.93	0	1.49
Nº 16 Tanque Lastre	12.33	12.64	5.9	0	2.34
Nº 17 Er. Tanque Lastre	84.68	86.79	5.27	3.37	5.86
Nº 17 Br. Tanque Lastre	84.68	86.79	5.27	-3.37	5.86
Nº 18 Er. Tanque Lastre	56.74	58.15	-2.28	6.54	3.78
Nº 18 Br. Tanque Lastre	56.74	58.15	-2.28	6.54	-3.78
Nº 19 Er. Agua Dulce	13.49	13.49	104.03	6.12	8.34
Nº 19 Br. Agua Dulce	13.49	13.49	104.03	-6.12	8.34
Nº 20 Servicio Diario	28.48	26.49	28.05	0	3.5
Nº 21 Sedimentación	38.34	35.65	28.58	0	3.9

Tabla 13.1 – Características principales de los tanques.

Sección	Peso/m	Sección	Peso/m	Sección	Peso/m
0	0	39	77.8179563	77	69.8493093
1	55.0441145	40	77.8179563	78	69.8493093
2	55.0441145	40	77.8179563	79	69.8493093
3	69.8493093	41	77.8179563	80	55.0441145
4	69.8493093	42	77.8179563	80	55.0441145
5	69.8493093	43	77.8179563	81	55.0441145
5	69.8493093	44	77.8179563	82	55.0441145
6	69.8493093	45	77.8179563	83	55.0441145
7	69.8493093	45	77.8179563	84	55.0441145
8	69.8493093	46	77.8179563	85	55.0441145
9	69.8493093	47	80.1942323	85	55.0441145
10	69.8493093	48	80.1942323	86	55.0441145
10	69.8493093	49	80.1942323	87	55.0441145
11	69.8493093	50	80.1942323	88	55.0441145
12	71.8253093	50	80.1942323	89	55.0441145
13	71.8253093	51	80.1942323	90	55.0441145
14	69.8493093	52	80.1942323	90	33.4441145
15	69.8493093	53	80.1942323	91	33.4441145
15	75.3633093	54	80.1942323	92	33.4441145
16	77.2538093	55	80.1942323	93	33.4441145
17	77.2538093	55	80.1942323	94	33.4441145
18	77.2538093	56	80.1942323	95	33.4441145
19	77.2538093	57	80.1942323	95	33.4441145
20	77.2538093	58	80.1942323	96	33.4441145
20	71.7398093	59	80.1942323	97	33.4441145
21	71.7398093	60	80.1942323	98	33.4441145
22	69.8493093	60	80.1942323	99	33.4441145
23	76.9184521	61	80.1942323	100	33.4441145
24	76.9184521	62	80.1942323	100	33.4441145
25	76.9184521	63	80.1942323	101	33.4441145
25	76.9184521	64	80.1942323	102	60.5974478
26	76.9184521	65	80.1942323	103	60.5974478
27	102.044285	65	80.1942323	104	60.5974478
28	102.044285	66	80.1942323	105	60.5974478
29	88.8017854	67	80.1942323	105	60.5974478
30	76.9184521	68	80.1942323	106	60.5974478
30	77.8179563	69	80.1942323	107	60.5974478
31	77.8179563	70	80.1942323	108	33.4441145
32	77.8179563	70	80.1942323	109	33.4441145
33	77.8179563	71	80.1942323	110	33.4441145
34	77.8179563	72	80.1942323	110	49.5391145
35	77.8179563	73	69.8493093	111	49.5391145
35	77.8180	74	69.8493093	112	38.9985739
36	77.8180	75	69.8493093	113	38.9985739
37	77.8180	75	69.8493093	114	38.9985739
38	77.8180	76	69.8493093	115	38.9985739

Tabla 13.2 – Peso en cada sección del buque para una distribución lineal.

Sección	Peso/m	Sección	Peso/m	Sección	Peso/m
0	33.4441	39	77.3336	77	69.8493
1	33.4441	40	76.8476	78	55.0441
2	33.4441	40	76.8476	79	55.0441
3	69.8493	41	76.3616	80	55.0441
4	69.8493	42	75.8756	80	55.0441
5	69.8493	43	75.3896	81	55.0441
5	69.8493	44	74.9036	82	55.0441
6	69.8493	45	74.4176	83	55.0441
7	69.8493	45	74.4176	84	55.0441
8	69.8493	46	89.6012	85	55.0441
9	69.8493	47	85.1127	85	55.0441
10	69.8493	48	84.7029	86	55.0441
10	69.8493	49	84.2931	87	55.0441
11	69.8493	50	83.8833	88	55.0441
12	71.5125	50	83.8833	89	55.0441
13	71.8293	51	83.4735	90	55.0441
14	78.4037	52	83.0637	90	33.4441
15	75.8573	53	82.6539	91	33.4441
15	75.8573	54	82.2441	92	33.4441
16	78.2889	55	81.8343	93	33.4441
17	77.7254	55	81.8343	94	33.4441
18	77.1613	56	81.4245	95	33.4441
19	76.5972	57	81.0147	95	33.4441
20	76.0351	58	80.6049	96	33.4441
20	71.2661	59	80.1951	97	33.4441
21	70.9495	60	79.7853	98	33.4441
22	69.8493	60	79.7853	99	33.4441
23	79.6342	61	79.3755	100	33.4441
24	78.8579	62	78.9657	100	72.8415
25	78.0818	63	78.5559	101	69.3418
25	78.0818	64	78.1461	102	65.8435
26	77.3057	65	77.7363	103	62.3452
27	104.6579	65	77.7363	104	58.8469
28	101.5953	66	77.3265	105	55.3486
29	98.5315	67	76.9167	105	55.3486
30	83.9459	68	76.5069	106	51.8503
30	81.7060	69	76.0971	107	48.3534
31	81.2216	70	75.6873	108	33.4441
32	80.7356	70	75.6873	109	33.4441
33	80.2496	71	75.2775	110	33.4441
34	79.7636	72	74.8664	110	58.0695
35	79.2776	73	69.8493	111	54.6610
35	79.2776	74	69.8493	112	40.7084
36	78.7916	75	69.8493	113	37.2963
37	78.3056	75	69.8493	114	33.8842
38	77.8196	76	69.8493	115	30.4682

Tabla 13.3 – Peso en cada sección del buque para una distribución trapezoidal.