UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Naval y Oceánica

Proyecto final de carrera

Cálculo de datos hidrostáticos y la botadura de un barco en Matlab

Titulación: **Ingeniería Naval y Oceánica**

Alumno: **Pedro Daniel Clemente Reverte**

Director: **Juan Carlos Trillo Moya**

Departamento: **Matemática aplicada y estadística**

Septiembre, 2014

Agradecimientos

Al Doctor Juan Carlos Trillo Moya, por la ayuda y atención prestada durante la realización de este proyecto y por todo el apoyo mostrado en todo momento.

A todos los compañeros que me animaron en los momentos más difíciles, especialmente con los que tuve la suerte de compartir años de vivencias, y convivencias.

Y en especial, a mis familiares, amigos y a mi pareja, que día a día se han interesado en mi trabajo y esfuerzo y han estado apoyándome hasta el día de hoy.

Resumen

Uno de los problemas matemáticos históricamente conocidos, es la integral de formas con ecuaciones no definidas, para ello se usa actualmente la integral aproximada, la cual se explica en este proyecto. Eso nos pasa con los barcos, sus formas no se rigen por fórmulas empíricas conocidas. Por lo tanto, en este proyecto se han unido la integral indefinida con la ingeniería naval.

Además de todo ello, en la construcción de los barcos, uno de los grandes problemas recae en el momento de llevar a éste al agua. Por ello vimos interesante la realización de un proyecto en el cual se pudiese ver intuitivamente como sucede el proceso y calcular todos los posibles efectos indeseados que se pueden presentar en el transcurso de la botadura.

El proyecto nos aporta una serie de datos de gran ayuda, tanto para el proceso de botadura como en la obtención de datos hidrostáticos. Se reparte la obtención de datos en dos sencillas interfaces explicadas detalladamente y con una ayuda para todas las dudas posibles.

Cada término está explicado por separado, y se puede acceder a él fácilmente desde la interfaz gráfica.

1. Índice general

Índice de ilustraciones

Índice de tablas

1. Introducción

En este proyecto voy a explicar la importancia de la integración numérica aproximada en la ingeniería, y la vamos a usar para la creación de un programa que estudie la botadura de un barco. En construcción naval, se necesita el cálculo integral continuamente para la resolución de áreas y volúmenes de formas no regulares. Para este tipo de integrales, no tenemos ecuaciones matemáticas exactas, por ello nos es imprescindible el uso de integración aproximada.

Disponiendo de una función integrando y unos límites, estaría totalmente definida la integral, pero al no tener una ecuación que nos defina la función, históricamente se ha recurrido a dispositivos que nos calculaban el área encerrada a partir de las cartillas de trazado dibujadas a escala, o tomando medidas directamente de la superficie o volumen a calcular.

Estos métodos inducían a errores mayores que los que se cometen actualmente, ya que las integraciones eran manuales y los métodos de medida eran a escala, lo cual agrava más los pequeños errores de medida.

Hoy en día, el cálculo de las integrales se realiza de una manera mucho más fácil en ordenadores, ya que traen programas específicos para ello.

Estos programas también usan un sistema de integración aproximada, usando un número mucho más alto de datos del que usamos en los cálculos manuales. De esa forma su error es menor.

En este proyecto, además diseñaremos un programa para calcular los datos hidrostáticos en adrizado y la botadura de un barco cualquiera, a partir de dos programas: Matlab y Rhinoceros.

El proyecto está organizado de la siguiente manera:

- **Sección 1:** Introducción al proyecto.
- **Sección 2:** Explicación de los método de integración.
- **Sección 3:** Usos de la integración aproximada en la ingeniería naval.
- **Sección 4:** Introducción al proceso de la botadura y diferentes tipos de botadura.
- **Sección 5:** Método de cálculo de la botadura.

- **Sección 6:** Cómo se obtienen, paso a paso, las formas para Matlab a partir de unas formas de barco en Rhinoceros.
- **Sección 7:** Explicación detallada de la obtención de valores hidrostáticos y datos de la botadura a partir del programa Matlab.
- **Sección 8:** Vista de los tutoriales que se presentan en la interfaz gráfica.
- **Sección 9:** ANEXO, en él se incluyen todos los comandos de la programación de Matlab.
- **Sección 10:** Bibliografía.

2. Métodos de integración

2.1 Integral definida

Dada una función f(x) ≥ 0 en intervalo [a, b], la **integral definida** es igual al área limitada entre la gráfica de f(x), el eje de abscisas, y las rectas verticales x = a y x = b.

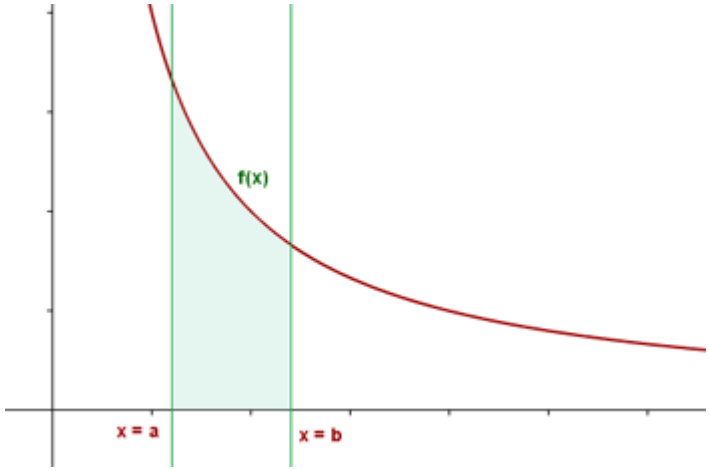

Imagen 1. Integral definida

La **integral definida** se representa por:

$$I = \int_a^b y\,dx = \int_a^b f(x)dx.$$

- **∫** es el signo de integración.
- **a** límite inferior de la integración.
- **b** límite superior de la integración.
- **f(x)** es el **integrando** o función a integrar.
- **dx** es **diferencial de x**, e indica cuál es la variable de la función que se integra.

2.2 Integral aproximada.

La integral aproximada o integración numérica, es la única solución cuando no tenemos la función definida ni la posibilidad de definirla, o para unas funciones cuyas primitivas no se pueden expresar como una combinación en un número finito de pasos de las llamadas funciones elementales. Por otra parte, un ingeniero en su actividad diaria suele trabajar con funciones cuya ley no conoce, son funciones "empíricas", vale decir que son mediciones. Como por ejemplo, el área de flotación de una sección de un barco. No sabemos la función que encierra esa área, pero sí tenemos las mangas y semimangas a lo largo de la eslora para cada sección. Con esas mediciones, podemos realizar una integración aproximada para obtener el área de la flotación, sabiendo el error máximo que cometemos en ese proceso. Vamos a ver unos procesos por los cuales podemos calcular tal valor aproximado y una estimación del error obtenido.

Sean $a, b \in \mathbb{R}/ a<b$, y sea f: $[a, b] \rightarrow \mathbb{R}$ cualquier función, a la cual no le pedimos en principio ninguna hipótesis. Sin embargo, sugerimos acompañar esta lectura con dibujos donde f será positiva y continua(o seccionalmente continua), y donde pensar en áreas contribuirá a una mejor comprensión de las ideas.

Sea P una partición o subdivisión de [a, b] dada por medio de los puntos

$$a= x_0 < x_1 < x_2 < x_3 < \dots < x_{k-1} < x_k < \dots < x_{n-1} < x_n = b$$

Para cada k = 1,2,…,n consideramos el subintervalo $I_k= [x_{k-1},x_k]$, cuya longitud es $\Delta x_k= x_k- x_{k-1}$, y elegimos un punto $\xi_k \in I_k$. Sea Q el conjunto de tales puntos "intermedios" : $Q=\{ \xi_1, \xi_2, \xi_k, \dots , \xi_n\}$. Tenemos entonces cuatro datos: un intervalo, una función, una partición del intervalo y una selección de puntos intermedios compatible con la partición. Llamamos suma de RIEMANN (asociada a esta cuaterna de datos) a:

$$\boldsymbol{S([a,b],f,P,Q) = \sum_{k=1}^{n} f(\xi_k)\Delta_{x_k}.}$$

Algunas propiedades son:

- Si f está acotada en [a,b] entonces *f* es integrable en [a,b]
- Si *f* es continua (o seccionalmente continua) en [a,b] entonces *f* es integrable en [a,b]
- Si *f* es monótona (o seccionalmente monótona) en [a,b] entonces *f* es integrable en [a,b]
- Si $C_1 \leq f(x) \leq C_2 \ \forall\, x \in$ [a,b] y f es integrable en [a,b], entonces

$$C_1(b-a) \leq \int_a^b f(x)dx \leq C_2(b-a).$$

A partir de esta desigualdad, y usando apropiadamente las propiedades de las funciones continuas, se demuestra fácilmente el útil e importante Teorema del Valor Medio (TVM) del cálculo integral:

$$f \ continua \ en \ [a,b] \rightarrow \exists \xi \in [a,b] / \int_a^b f(x)dx = f(\xi)(b-a).$$

Generalicemos la última desigualdad y el TVM. Sean dos funciones f y g tales que $C_1 \leq f(x) \leq C_2$, $g(x) \geq 0, \ \forall x \in [a,b]$. Si las funciones *g* y $f \cdot g$ son integrables en [a,b],

$$C_1 \int_a^b g(x)dx \leq \int_a^b f(x)g(x)dx \ \leq C_2 \int_a^b g(x)dx.$$

Si f es además continua en [a,b], se prueba de forma rápida que existe $\xi \in$ [a,b] tal que

$$\int_a^b f(x)g(x)dx = f(\xi) \int_a^b g(x)dx,$$

lo que se denomina Teorema generalizado del Valor Medio del Cálculo Integral.

Nótese que si *g* vale constantemente 1, las dos nuevas propiedades se reducen a las dos anteriores.

Con mínimas variantes, hubiese podido demostrar la misma generalización del TVM en el caso en que $g(x) \leq 0, \ \forall x \in$ [a,b]. En definitiva, lo que interesa no es que *g* sea positiva, sino que no cambie de signo.

2.3 Intentos elementales de integración aproximada

Dado que la integral se define como un límite de sumas de Riemann, la idea inicial es aproximar el valor de la integral mediante una conveniente suma de Riemann. Hay una suposición subyacente: creer que el valor $f(\xi_k)$ es un buen representante de los distintos valores que asume f en el intervalo $[x_{k-1},x_k]$.

Pero, ¿quién elige un ξ_k que sea bastante representativo? Una persona con experiencia tal vez pueda hacerlo frente a un gráfico particular, pero aquí se desea dar ideas generales.

Por obvias razones de comodidad en los cálculos, usualmente se construyen particiones donde todos los subintervalos tienen la misma longitud, es decir que

$$\Delta x_k = \frac{b-a}{n}, \forall k = 1,2,\dots,n.$$

De este modo

$$\sum_{k=1}^{n} f(\xi_k)\Delta x_k = \frac{b-a}{n}\sum_{k=1}^{n} f(\xi_k).$$

En tal caso se dice que la partición es regular, o también que los puntos x_k están equiespaciados. Ahora bien, esta condición es bastante realista para las funciones empíricas que analiza un ingeniero, pues las mediciones casi siempre se realizan a intervalos regulares. Por lo tanto, de ahora en adelante todas nuestras particiones serán regulares.

En adelante usamos la notación $y_k=f(x_k)$. Las elecciones más comunes para ξ_k son:

- Método del extremo izquierdo: $\xi_k = x_{k-1}$,
- Método del extremo derecho: $\xi_k = x_k$,
- Método del punto medio: $\xi_k = (x_{k-1} + x_k)/2$,

las cuales conducen, respectivamente, a las siguientes sumas de Riemann:

$$I_n = \frac{b-a}{n}\sum_{k=1}^{n} f(x_{k-1}) = \frac{b-a}{n}(y_0 + y_1 + \cdots + y_{n-1}).$$

$$D_n = \frac{b-a}{n}\sum_{k=1}^{n} f(x_k) = \frac{b-a}{n}(y_1 + y_2 + \cdots + y_n).$$

$$M_n = \frac{b-a}{n}\sum_{k=1}^{n} f\left(\frac{x_{k-1}+x_k}{2}\right).$$

Si *f* es creciente en [a,b], I_n (respectivamente D_n) nos dará una aproximación por defecto (respectivamente por exceso) del valor de la integral. Aparentemente M_n nos da una mejor aproximación. Si *f* es decreciente en [a,b], se intercambian los roles entre I_n y D_n, pero M_n vuelve a parecer más confiable.

La intuición nos indica que tanto la confiabilidad de I_n como la de D_n deben mejorar si *f* tiene algunos tramos donde crece y otros donde decrece, pues entonces se va produciendo una "compensación" entre errores de distinto signo. También la intuición nos hace sospechar que en cualquiera de estos casos el error irá disminuyendo a medida que *n* vaya creciendo. Y efectivamente, bajo ciertas hipótesis sobre la función *f* se puede demostrar que en los 3 métodos el error tiende a cero cuando n→+∞, pero no con la "misma velocidad". Para I_n y D_n veremos que el error es del orden de $1/n$, mientras que en el tercer método el error será del orden de $1/n^2$. Por esta razón en el método del punto medio no se necesita un valor tan grande de *n* para lograr aproximaciones satisfactorias.

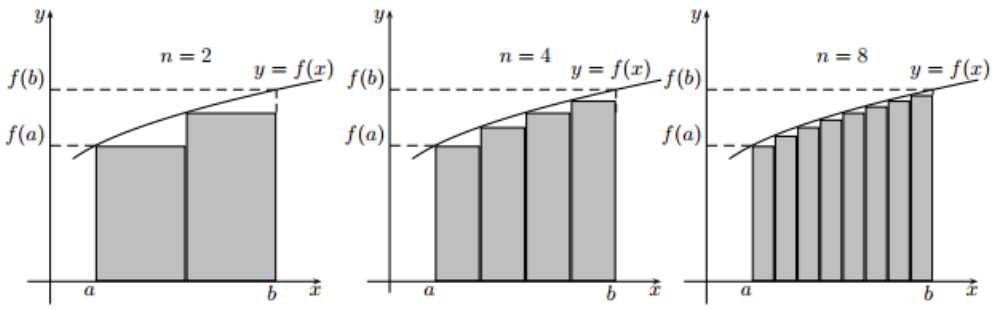

Imagen 2. Diferentes representaciones de la integral aproximada

Algunas representaciones gráficas de I_n que nos van dando aproximaciones cada vez mejores (siempre por defecto, pues esta *f* es creciente) pueden verse en la imagen 2. Cuando mayor es el número de intervalos, menor es el error cometido.

Ya se ha comentado que si *f* es creciente en [a,b], I_n da una aproximación por defecto y D_n una aproximación por exceso. Entonces parece muy natural promediar I_n con D_n para lograr una mejor aproximación del valor de la integral.

Con mínimos cambios en el razonamiento, la propuesta de promediar I_n con D_n también suena interesante si f es decreciente, y más generalmente en el caso de una función seccionalmente monótona.

Esta idea da origen al denominado "Método de los Trapecios" cuyo nombre está inspirado en gráficos como los de la imagen 3.

Cuanto mayor sea el número de intervalos, mejor será la aproximación.

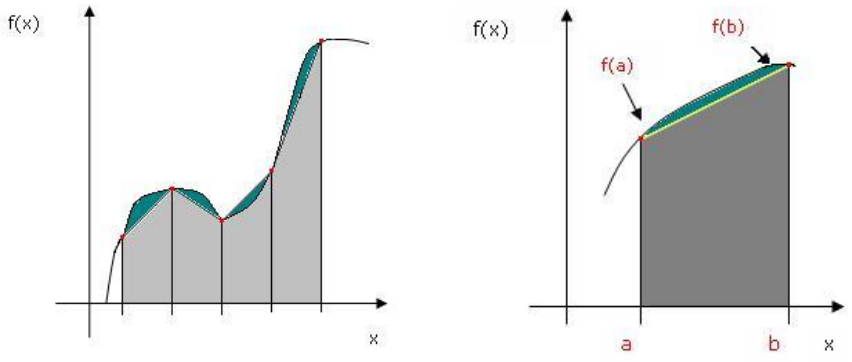

Imagen 3. Método de los trapecios

Formalmente:

$$T_n = \frac{1}{2}(I_n + D_n) = \frac{b-a}{2n}(y_0 + 2y_1 + 2y_2 + \cdots + 2y_{n-1} + y_n).$$

Hasta aquí aproximo la integral mediante I_n, D_n y M_n, que en definitiva eran sumas de Riemann. En cambio T_n no está pensada como una suma de Riemann (aunque bien podría coincidir con una), sino como un promedio de ellas. Desde el punto analítico, en T_n está subyacente la idea de interpolación lineal.

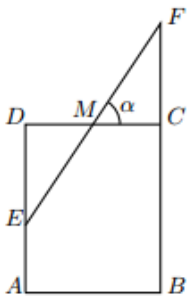

Imagen 4. Método del Punto Medio

El punto medio de un lado del rectángulo tiene una sencilla pero útil propiedad, que vemos en la figura 4, donde *M* es el punto medio de *CD*. El área del trapecio *ABFE* resulta igual a la del rectángulo *ABCD*, independientemente de cuál sea el ángulo α. Esto surge de la congruencia entre los triángulos rectángulos *MCF* y *MDE*.

En las figuras anteriores, *f* es continua. Si además le pedimos que sea derivable, podemos considerar la recta tangente a su gráfica en algún punto particular, por ejemplo el correspondiente a $m_k=(x_{k-1}+x_k)/2$. Entonces el método del punto medio se puede interpretar como "método del trapecio tangente por el punto de la abcisa media".

2.4 Fórmula de los trapecios

Al tomar la suma de Riemann con aproximación del verdadero valor de una integral, estamos procediendo como si la función *f* en el intervalo $[x_{k-1},x_k]$ se mantuviese constante, vale decir como si su gráfica sobre dicho intervalo fuese un segmento horizontal. En cambio en el método de los trapecios y por influjo de la interpolación lineal, estamos mirando a *f* en I_k como si fuese una función lineal afín, y al correspondiente arco de curva lo estamos aproximando por su cuerda.

Dado un $n\in\mathbb{N}$, la denominada Fórmula de los trapecios nos dice que

$$\int_a^b f(x)dx \approx T_n = \frac{\Delta x}{2}(y_0 + 2y_1 + 2y_2 + \cdots + 2y_{n-1} + y_n).$$

También podemos escribirla en la forma

$$\int_a^b f(x)dx = T_n + R_n.$$

Donde R_n simboliza el resto o error (cantidad que hay que sumar al valor aproximado para llegar al valor exacto).

2.5 Regla de Simpson

Esta regla lleva el nombre de Thomas Simpson, matemático británico que la utilizó a mitad del siglo XVIII, aunque ya había sido formulada en 1668 por James Gregory.

Sabemos que la fórmula de los trapecios integra exactamente los polinomios de primer grado utilizando dos nodos en el intervalo [a,b]. Añadiendo un nodo más, esperamos que para ciertos pesos a determinar, la fórmula

$$I = p_1 f(x_1) + p_2 f(x_2) + p_3 f(x_3),$$

sea exacta para polinomios de segundo grado. Los pesos pueden determinarse sustituyendo f por las funciones $\{1, x, x^2\}$ y resolviendo el sistema lineal resultante de exigir que el error sea nulo.

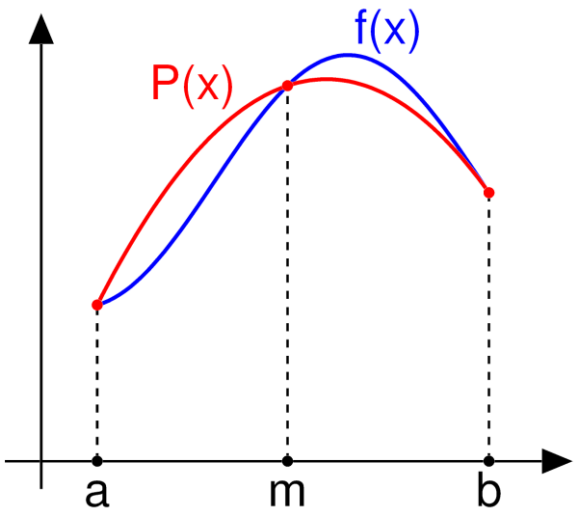

Imagen 5. Regla de Simpson

Por otra parte, aproximando la función del integrando por el polinomio interpolador de Lagrange,

$$f(x) \approx L_1(x) \cdot f(a) + L_2(x) \cdot f(c) + L_3(x) f(b),$$

donde $c = \frac{a+b}{2}$, y las funciones de Lagrange son $L_1(x) = \frac{(x-c)(x-b)}{(a-c)(a-b)}$, $L_2(x) = \frac{(x-a)(x-b)}{(c-a)(c-b)}$, $L_3(x) = \frac{(x-a)(x-c)}{(b-a)(b-c)}$,

la integral puede aproximarse por:

$$\int_a^b f(x)dx \approx f \cdot \int_a^b L_1(x)dx + f(c) \cdot \int_a^b L_2(x)dx + f(b) \cdot \int_a^b L_3(x)dx$$

$$= \frac{b-a}{6}\big(f(a) + 4f(c) + f(b)\big).$$

De este modo obtenemos tanto la *fórmula de cuadratura*, que recibe el nombre de *Simpson simple*:

$$I_s = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right),$$

como la expresión del error cometido en la aproximación:

$$E_s = I - I_s = -\frac{(b-a)^5}{90} f^{IV}(\xi), \qquad \xi \in]a, b[.$$

La característica más destacable de esta fórmula es que resulta exacta para polinomios de tercer grado, mientras que en su construcción sólo se exigía la exactitud para grado 2.

Al igual que en el método de los trapecios, es posible reducir el error, dividiendo el intervalo [a,b] en un número par de subintervalos, de manera que sobre cada pareja de subintervalos se le aplique la fórmula de cuadratura de Simpson. De este modo obtenemos la fórmula de Simpson compuesta: si dividimos el intervalo de integración [a,b] en *2m* subintervalos iguales de amplitud $h = \frac{b-a}{2m}$, los extremos de dichos subintervalos se definen como $x_k = a + kh, k = 0,1, \dots, 2m$. La integral puede expresarse como suma de *m* integrales, cada una de las cuales está definida sobre los subintervalos consecutivos:

$$\int_a^b f(x)dx = \int_{x_1}^{x_3} f(x)dx + \int_{x_3}^{x_5} f(x)dx + \cdots + \int_{x_{2m-3}}^{x_{2m-1}} f(x)dx + \int_{x_{2m-1}}^{x_{2m+1}} f(x)dx.$$

Aplicando la fórmula de Simpson simple sobre cada una de estas integrales:

$$\int_a^b f(x)dx \cong \left(\frac{x_3 - x_1}{6}\right) \cdot \big(f(x_1) + 4f(x_2) + f(x_3)\big) +$$

$$+ \left(\frac{x_5 - x_3}{6}\right) \cdot \big(f(x_3) + 4f(x_4) + f(x_5)\big) + \cdots$$

$$\dots + \left(\frac{x_{2m-1} - x_{2m-3}}{6}\right) \cdot \big(f(x_{2m-3}) + 4f(x_{2m-2}) + f(x_{2m-1})\big) + \cdots$$

$$\dots + \left(\frac{x_{2m+1} - x_{2m-1}}{6}\right) \cdot \big(f(x_{2m-1}) + 4f(x_{2m}) + f(x_{2m+1})\big),$$

donde $x_{2k+1} - x_{2k-1} = 2h, para\ k = 1, \dots, m$ y además notamos que, mientras las imágenes del primer nodo y el último tienen peso unidad, las imágenes de los nodos con subíndice par tienen peso 4 y las de subíndice impar (distintos de 1 y 2m+1) tienen peso 2. Así, obtenemos la *fórmula de Simpson compuesta.*

$$\int_a^b f(x)dx \cong \frac{h}{3}(f(x_1) + 4f(x_2) + 2f(x_3) + 4f(x_4) + \cdots + 2f(x_{2m-1}) + 4f(x_{2m}) + f(x_{2m+1})).$$

Además, si denotamos la imagen del integrando en los nodos por $y_k = f(x_k), k = 1,2, \dots, 2m + 1$, la fórmula de Simpson compuesta queda

$$I_s[h] = \frac{h}{3}(y_1 + 4y_2 + 2y_3 + \cdots + 2y_{2m-1} + 4y_{2m} + y_{2m+1}).$$

Podemos expresar el error de la fórmula compuesta de Simpson en términos del error cometido en cada subintervalo $[x_{2k-1}, x_{2k+1}], k = 1,2, \dots, m, E_S^k = -\frac{h^5}{90} f^{IV}(\xi_k)$, donde $\xi_k \epsilon]x_{2k-1}, x_{2k+1}[$. Así pues, el error total cometido es

$$E_S = \sum_{k=1}^{m} E_S^k = -\sum_{k=1}^{m} \frac{h^5}{90} f^{IV}(\xi_k), \qquad \xi_k \epsilon]x_{2k-1}, x_{2k+1}[.$$

Aplicando el Teorema del Valor Medio, para cada ξ_k se verifica la desigualdad

$$\min_{x\epsilon[a,b]} f^{IV}(x) \leq f^{IV}(\xi_k) \leq \max_{x\epsilon[a,b]} f^{IV}(x).$$

Y por tanto

$$m \cdot \min_{x\epsilon[a,b]} f^{IV}(x) < \sum_{k=1}^{m} f^{IV}(\xi_k) < m \cdot \max_{x\epsilon[a,b]} f^{IV}(x).$$

Así, utilizando de nuevo el Teorema del Valor Medio, existe $\xi \epsilon]a, b[$ tal que

$$f^{IV}(\xi) = \frac{1}{m} \sum_{k=1}^{m} f^{IV}(\xi_k).$$

De este modo, el error de la fórmula compuesta de Simpson es

$$E_T = -\frac{h^5}{90} m \cdot f^{IV}(\xi) = -\frac{h^4}{180}(b - a) f^{IV}(\xi),$$

para cierto $\xi \epsilon]a, b[$.

Notemos que si la derivada cuarta del integrando (supuesta continua y derivable) es aproximadamente constante, se puede suponer que el error es proporcional a h^4, lo que significa que al duplicar el número de subintervalos, esperamos que el error se divida por 16, 2^4=16, con lo que se obtiene al menos una cifra decimal exacta más en el resultado.

2.5.1 Simpson adaptativo

Hay ocasiones, sobre todo en el ámbito naval, en que en los extremos surge la necesidad de obtener una mayor precisión por los cambios bruscos de áreas en los mismos. Para estos casos se utiliza el método de Simpson adaptativo, en el cual los intervalos en los extremos que se toman son de tamaños inferiores y proporcionales a los tomados durante el resto de la integral. Esta adaptación se consigue haciendo proporcionales también los coeficientes de Simpson.

Como vemos en la imagen 6, hay dos intervalos diferentes, el intervalo de longitud h y el intervalo de longitud h'. De esta manera, en los extremos se aumenta la precisión de la medida de la integración aproximada. Para realizar esta integración debemos ajustar los coeficientes multiplicadores de Simpson de la siguiente manera (ver tabla 1).

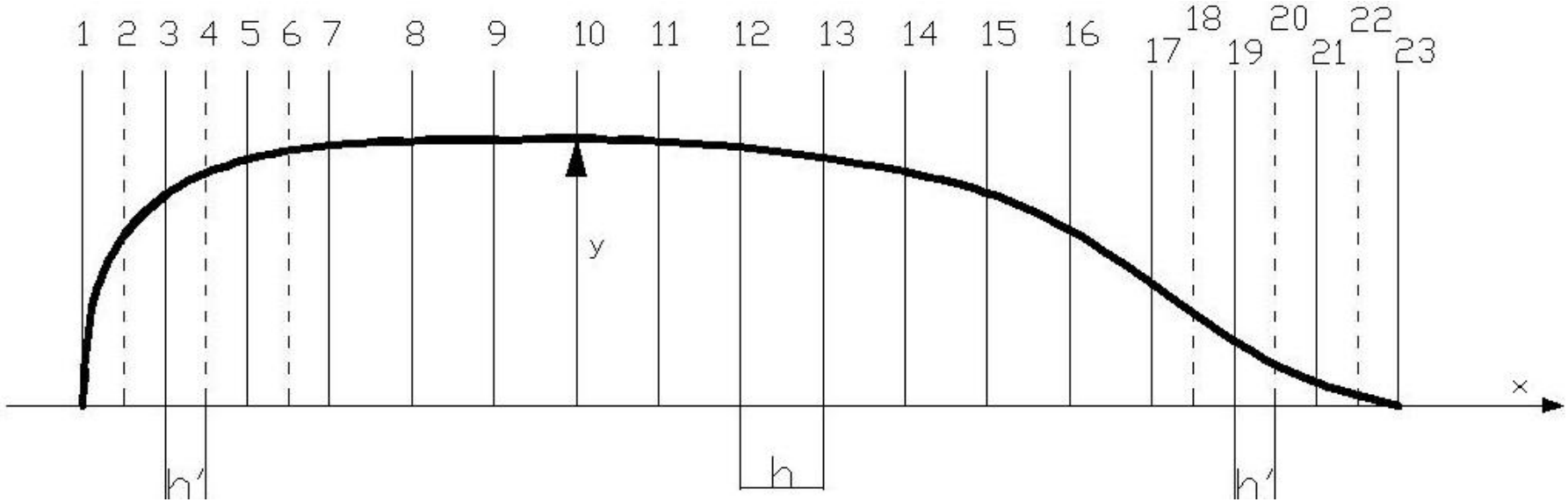

Imagen 6. Uso de los multiplicadores de Simpson

	Sección	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
	Abcisa	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}	y_{16}	y_{17}	y_{18}	y_{19}	y_{20}	y_{21}	y_{22}	y_{23}
Simpson normal	Multiplicadores	1	-	4	-	1+1	-	4	1+1	4	1+1	4	1+1	4	1+1	4	1+1	4	-	1+1	-	4	-	1
	Multiplicador	1	-	4	-	2	-	4	2	4	2	4	2	4	2	4	2	4	-	2	-	4	-	1
Simpson adaptativo	Multiplicadores	0,5	2	0,5+0,5	2	0,5+0,5	2	1+0,5	4	2	4	2	4	2	4	2	4	1+0,5	2	0,5+0,5	2	0,5+0,5	2	0,5
	Multiplicador	0,5	2	1	2	1	2	1,5	4	2	4	2	4	2	4	2	4	1,5	2	1	2	1	2	0,5

Tabla 1. Multiplicadores de Simpson

Para estos coeficientes, se usa el mismo valor de h, ya que, lo que se ha hecho es ajustar los valores de los multiplicadores. Las abscisas $y_{2,}\ y_4, y_6, y_{18}\ , y_{20}\ e\ y_{22}$ no se usarán en el proceso de Simpson normal, ya que las se ha colocado para el apartado de Simpson adaptativo. De esta manera las integrales quedarán de la siguiente manera.

Para la integral aproximada por el método de Simpson normal:

$$A = \int_x y \cdot dx \cong \frac{h}{3} \cdot (y_1 + 4y_3 + 2y_5 + 4y_7 + 2y_8 + 4y_9 + 2y_{10} + 4y_{11} + 2y_{12} + 4y_{13} + 2y_{14} + 4y_{15} + 2y_{16} + 4y_{17} + 2y_{19} + 4y_{21} + y_{23})$$

Para la integral aproximada por el método de Simpson adaptativo será:

$$A = \int_x y \cdot dx \cong \frac{h}{3} \cdot (0{,}5y_1 + 2y_2 + y_3 + 2y_4 + y_5 + 2y_6 + 1{,}5y_7 + 4y_8 + 2y_9 + 4y_{10} + 2y_{11} + 4y_{12} + 2y_{13} + 4y_{14} + 2y_{15} + 4y_{16} + 1{,}5y_{17} + 2y_{18} + y_{19} + 2y_{20} + y_{21} + 2y_{22} + y_{23})$$

De esta manera la aproximación por el método de Simpson es más precisa y podemos confiar más en los datos obtenidos. Este método de integración no solo sirve para la integración del área, sirve para cualquier uso en las integraciones aproximadas que se realizan en el barco.

3. Uso de la integración aproximada en la ingeniería naval

3.1 Introducción

Para entender el uso de la integración aproximada en la rama de ingeniería naval, voy a hacer una breve introducción acerca de diversas utilizaciones.

Comenzamos por la necesidad de que un barco flote. Según la teoría del buque, un buque o flotador es considerado como un sólido en equilibrio, que se encuentra en reposo y en aguas tranquilas. Por lo tanto:

- La superficie libre del líquido en el que flota siempre será un plano, y su intersección con las formas del casco (flotación) siempre será un área plana.
- Las únicas fuerzas que actúan, de momento, son: el peso P, pasando por el centro de gravedad del buque G, y el empuje E, aplicado en el centro de gravedad del volumen sumergido o carena, al que se le llama centro de carena B. Por tanto no consideramos ni las fuerzas de inercia ni las resistencias del agua o del aire.

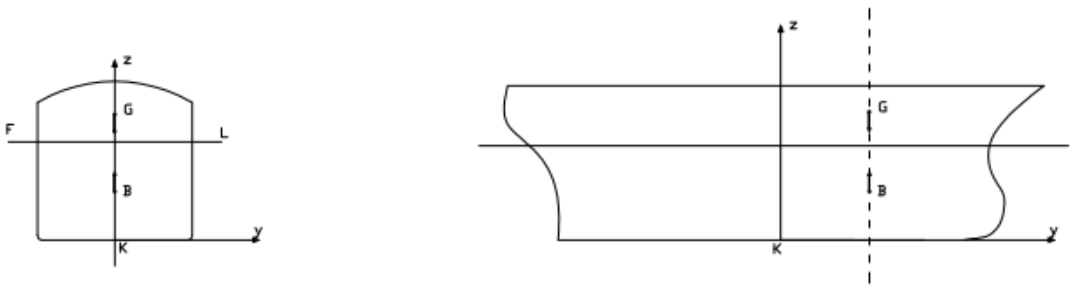

Imagen 7. Representación de las fuerzas de gravedad de un barco

En la física mecánica se estudia que el equilibrio de un cuerpo puede ser: estable, inestable o indiferente, según que, si al desplazarse un infinitésimo de su posición de equilibrio, el cuerpo vuelva a su posición inicial, se aleje de ella o simplemente no se mueva, respectivamente.

En este caso, como P y E son siempre verticales al plano de la flotación FL, de los 6 posibles grados de libertad del flotador solo nos quedan tres: la traslación vertical y los giros alrededor de los ejes X e Y, ya que los otros tres van a ser intrínsecamente indiferentes.

3.2 Equilibrio de traslación sobre el eje Z

Al estar en equilibrio P=E, si se produjese un desplazamiento *dz* por alguna causa ajena al flotador, variaría el volumen sumergido ∇ y por tanto, el empuje, E→E', mientras que el peso no cambia, P=cte.

- Si *dz>0*, aumenta el volumen sumergido [E'=P], el flotador empieza a emerger, disminuyendo ∇, hasta que E=P.
- Si *dz<0* disminuye el volumen sumergido y por tanto el empuje [E'=P] el flotador se sumergirá hasta que nuevamente E=P.

Por lo tanto, el equilibrio sobre el eje Z siempre es estable, haciendo la salvedad de que no pierda la estanqueidad y entre agua.

3.3 Equilibrio de rotación alrededor de los ejes X e Y

En principio, al estudiar el equilibrio del buque por rotación, el flotador puede girar respecto de cualquier eje contenido en un plano XY. Si por alguna causa ajena al mismo se produjese un giro alrededor de uno cualquiera de estos infinitos ejes, variaría la geometría del volumen sumergido, y por tanto la posición del centro de carena (B → B_1), aunque no el valor absoluto de ∇, puesto que en todo momento se tiene que mantener la igualdad P=E.

El lugar geométrico formado por todos los posibles puntos ocupados por el c.d.c. (centro de carena), considerando todos los ángulos de giro respecto de los infinitos ejes del plano horizontal, será una superficie. Aquí solo vamos a ver el equilibrio según dos planos principales: transversal (respecto de un eje en el sentido del eje X, que dará los valores de estabilidad mínimos), y longitudinal (eje en el sentido del eje Y, que dará los valores máximos). La intersección de la superficie descrita por los c.d.c. con uno de estos planos nos dará una curva.

Veamos el equilibrio de un eje paralelo al eje X o equilibrio transversal. Supongamos que por alguna causa ajena al barco se produce un giro (escora) infinitesimal, dΘ. La curva descrita por el movimiento del c.d.c. será un arco infinitesimal, BB'=ds, que según se estudia en la geometría diferencial tendrá un centro de curvatura, M, llamado metacentro, y un radio de curvatura, BM, llamado radio metacéntrico. Si el plano de corte es transversal, como en este caso, el metacentro y el radio metacéntrico se llaman transversales, mientras que si el plano de corte fuese longitudinal se les llamaría longitudinales.

Una vez producida la escora dθ, que desplaza el c.d.c. desde B hasta B_1, las líneas de acción del peso y el empuje (siempre perpendiculares a la flotación correspondiente) ya no coinciden. Se produce un par Δ·GZ. El sentido de giro debido a este par es el que hace el equilibrio sea estable, inestable o indiferente.

3.3.1 Equilibrio estable

En estos casos, su sentido de giro hace que el buque regrese a su posición de flotación, ya que el brazo adrizante es positivo y el par que se genera contribuye a ello, como se indica en el sentido de giro de las flechas en las figuras de la imagen 8.

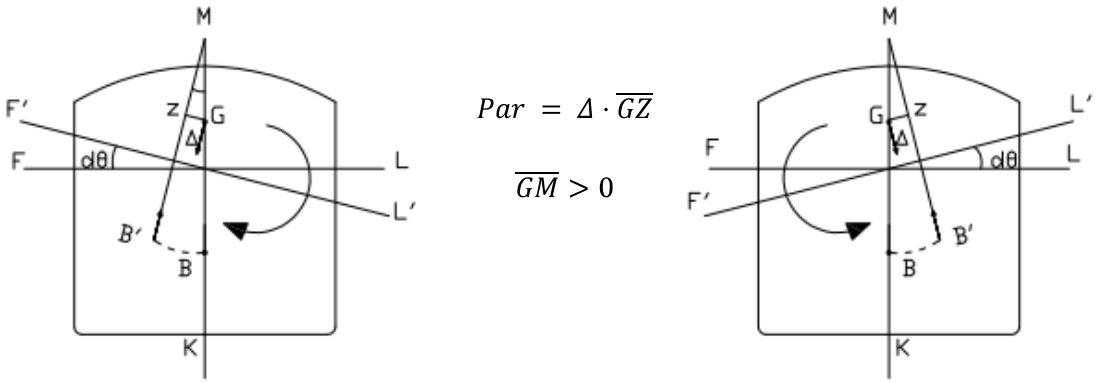

Imagen 8. Equilibrio estable

3.3.2 Equilibrio indiferente

En el caso del equilibrio indiferente, el par es nulo, ya que las fuerzas actúan en la misma dirección y sentidos opuestos, de esta manera el buque no tiene capacidad de reacción y se mantendrá en esa posición.

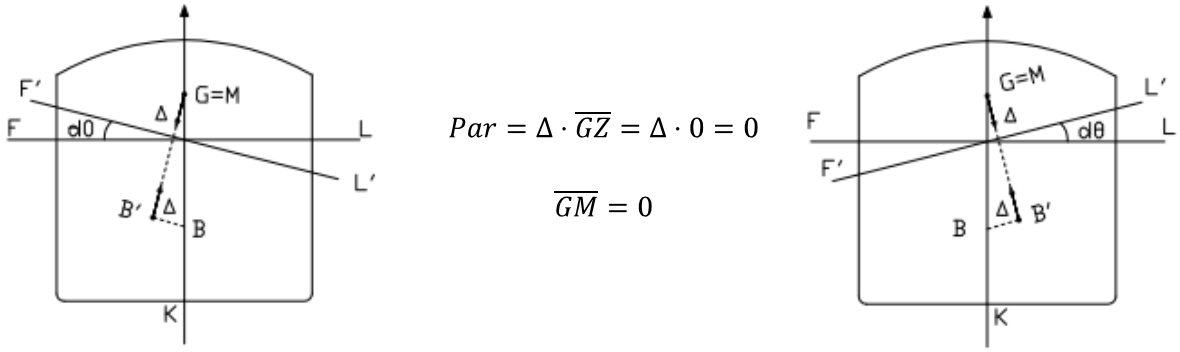

Imagen 9. Equilibrio indiferente

3.3.3 Equilibrio inestable

Esto sucede cuando el sentido del giro en ambos casos, hace que el buque se separe de su posición de adrizado. En este caso el par afecta negativamente a los intentos de mantener adrizado el buque.

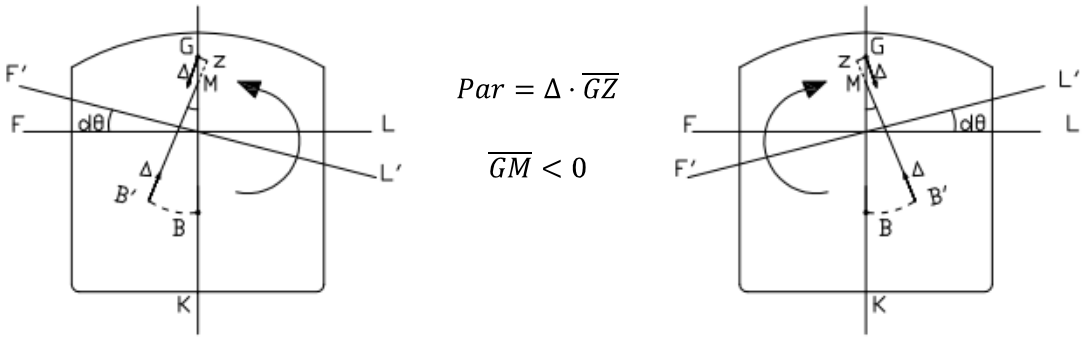

Imagen 10. Equilibrio inestable

3.3.4 Estabilidad permanente

Un caso particular es cuando $\overline{KB} > \overline{KG}$, en el que siempre va a ser estable el buque, pero esta situación se da muy pocas veces en la práctica. Solamente en casos especiales y en buques que precisen de esos requerimientos. De esta manera $\overline{GM}$ siempre nos dará positivo.

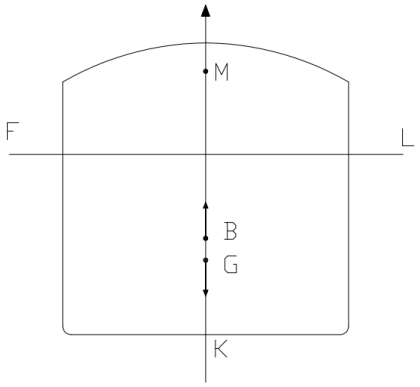

Imagen 11. Equilibrio permanente

Como hemos visto, el buque es estable es aquel en el que $\overline{GM} > 0$, es decir, la altura metacéntrica es positiva. Esta altura es la que nos da el criterio de estabilidad en la zona de estabilidad inicial tanto en el sentido transversal como en el sentido longitudinal.

La ecuación que nos define $\overline{GM}$ es:

$$\overline{GM} = \overline{KB} + \overline{BM} - \overline{KG}.$$

Ecuación que podemos comprobar en la imagen 11.

3.4 Cálculo de datos hidrostáticos

Estos valores ($\overline{KB}, \overline{BM}$ y $\overline{KG}$) se calcularán normalmente mediante los métodos de integración aproximada ya que las formas irregulares de los cuerpos a estudiar no nos harán posible tener unas ecuaciones que definan perfectamente los datos necesarios para ello.

3.4.1 Área de la flotación

Un área es una extensión de una superficie. Para calcular el área de una determinada flotación se integran las semimangas de dicha flotación a lo largo de la eslora.

$$dA = 2 \cdot y \cdot dx.$$

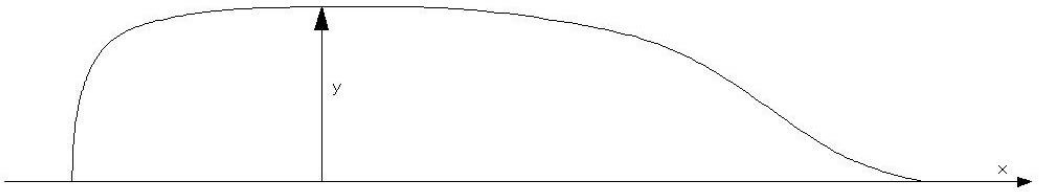

Imagen 12. Área de la flotación

Donde y son las medidas de las semimangas en dicha flotación, de esa manera se obtiene el área como:

$$A = 2 \cdot \int_L y \cdot dx.$$

3.4.2 Volumen de carena ∇

Un volumen siempre es un área multiplicada por una distancia perpendicular a ella $dV = A \cdot dx$. El volumen total de un cuerpo será el sumatorio de estos volúmenes elementales tal que:

$$V = \int dV = \int_a^b A \cdot dx.$$

Desde el punto de vista matemático de las integrales definidas, un volumen es el área bajo la curva de áreas, por lo que previamente debemos conocer esta función (o por lo menos una serie de valores discretos para poder utilizar un método de integración aproximada.

El caso más común que se presenta es el cálculo del volumen de la carena de un buque correspondiente a un determinado calado. Este volumen se representa con el símbolo ∇, y para ello tenemos dos procedimientos.

1. A partir de las áreas horizontales, también llamadas flotaciones, e integrar a lo largo del eje vertical (normalmente el eje z). Este procedimiento se usará sobre todo para sacar los cálculos de las curvas hidrostáticas normales.

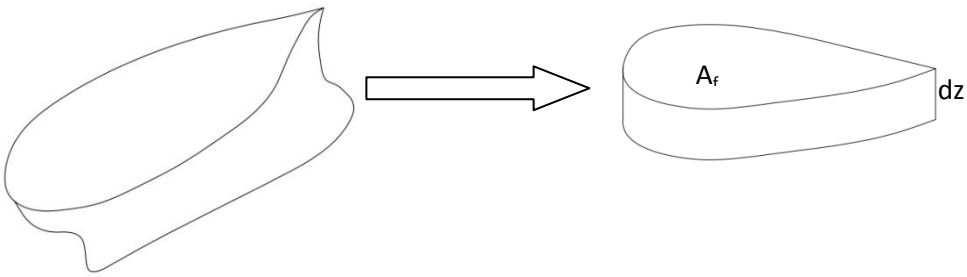

Imagen 13. Volumen de carena a partir de las flotaciones

2. A partir de las áreas transversales o secciones, para integrar a lo largo de la eslora (normalmente el eje x). Este procedimiento se utiliza normalmente para obtener resultados a partir de las curvas de Bonjean.

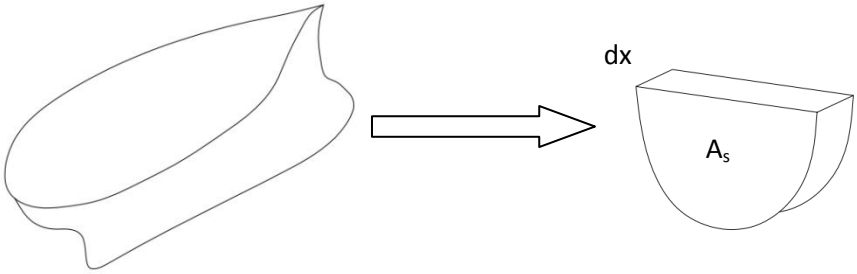

Imagen 14. Volumen de carena a partir de las secciones

3.4.3 Desplazamiento de trazado Δ

El desplazamiento de trazado se calcula a partir del volumen de trazado, una vez obtenido éste basta con multiplicar ese volumen por la densidad del fluido en el que el barco está situado como se indica en la siguiente ecuación.

$$\Delta = \delta \cdot \nabla.$$

Donde δ es la densidad del fluido y ∇ es el volumen de trazado.

3.4.4 Toneladas por centímetro de inmersión TCI

Sea un buque al que para un calado T, le corresponde un área de la flotación A_f y un volumen de carena ∇, y por lo tanto un desplazamiento Δ. Supongamos que a partir de esta situación se le carga un peso que le produzca una variación paralela infinitesimal de calado dT. Si el buque estaba en equilibrio, su peso tendrá que ser igual a su empuje. Al cargarle un peso, para que siga estando en equilibrio, este peso tendrá que ser compensado con el incremento de empuje correspondiente al dT, tal que:

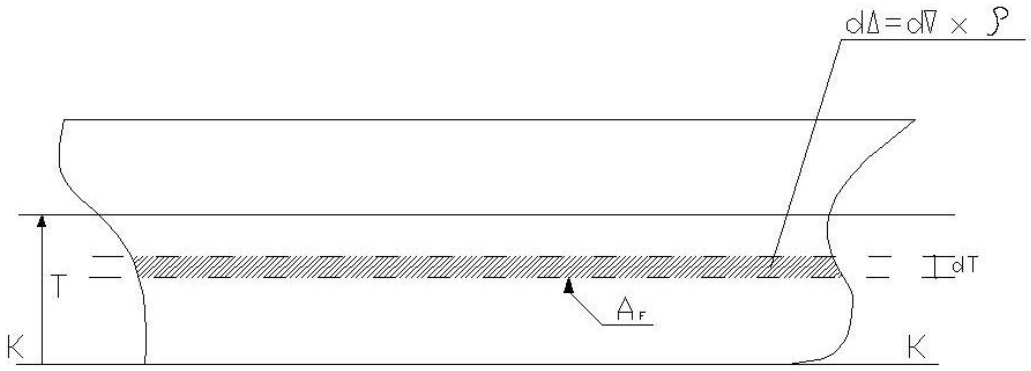

Imagen 15. Toneladas por centímetro de inmersión (TCI)

$$d\Delta = \rho \cdot d\nabla = \rho \cdot A_f \cdot dT.$$

Si consideramos dT como 1 centímetro, entonces el $d\Delta$ será igual a las toneladas por centímetro de inmersión (TCI), es decir, las toneladas que son necesarias cargar (descargar) para que el calado aumente (disminuya) un centímetro.

$$d\Delta = \rho \cdot A_f \cdot dT \rightarrow dT = \frac{1}{100} m \rightarrow TCI = \frac{\rho \cdot A_f}{100}.$$

Para esta ecuación es necesario que $\rho = {}^{t}/_{m^3}$ y $A_f = m^2$.

Desde el punto de vista práctico, este concepto se maneja como una simple regla de tres. Si con las TCI correspondientes el calado dado varía 1 centímetro, cuando cargamos (descargamos) un peso p habrá una variación de calado $dT = {}^{p}/_{TCI}$.

Evidentemente, se debe ser muy cauto cuando se aplique este concepto, pues la hipótesis o suposición que estamos haciendo es considerar que el área de la flotación permanece constante en toda la variación de calado que se produzca. Por tanto, las variaciones de calado tendrán que ser forzosamente pequeñas, a no ser que el buque sea del tipo costados rectos. Debido a ello, los errores cometidos pueden llegar a ser muy grandes.

Para obtener las TCI de cada flotación debemos conocer el A_f correspondiente a cada una. Este dato se obtiene normalmente mediante el método de integración aproximada de Simpson como se ha explicado previamente.

3.4.5 Posición del centro de carena. $\overline{KB}$ y $\overline{\otimes B}$

El término $\overline{KB}$ es el que identificamos como centro de carena de un buque, este punto es muy importante, ya que es la distancia en la que se genera el empuje del volumen desalojado por el buque con respecto a la línea base K. El punto B es el centro de carena.

Normalmente, un barco es simétrico en formas con respecto al plano diametral, así que tendremos ayudas a la hora de calcular la estabilidad transversal ya que, esa coordenada no nos interesará tanto como las otras dos coordenadas de B.

La ordenada del centro de carena $\overline{KB}$ se obtiene de la siguiente manera:

$$\overline{KB} = \frac{M_k}{\nabla}.$$

Siendo M_k el momento estático de todo el volumen sumergido ∇ que estamos considerando con respecto al plano base k.

La posición longitudinal o abscisa del centro de carena se puede representar de dos maneras, con respecto a la sección media donde se expresa como $\overline{\otimes B}$ o con respecto a la perpendicular de popa, que se expresa como $\overline{XB}$. Aquí indicaré como origen la sección media, y el $\overline{\otimes B}$ quedará expresado como:

$$\overline{\otimes B} = \frac{M_{\otimes}}{\nabla}.$$

Para ello hemos de conocer el momento estático del volumen $M_{\otimes}$. Para este caso también tenemos dos posibilidades dependiendo del tipo de diferencial de volumen $d\nabla$ que estemos utilizando.

1. A partir de las flotaciones se pueden obtener los dos valores, para ello hay que tener en cuenta en todos los cálculos donde se sitúa el $\overline{KB}$ particular de cada flotación. El resultado será:

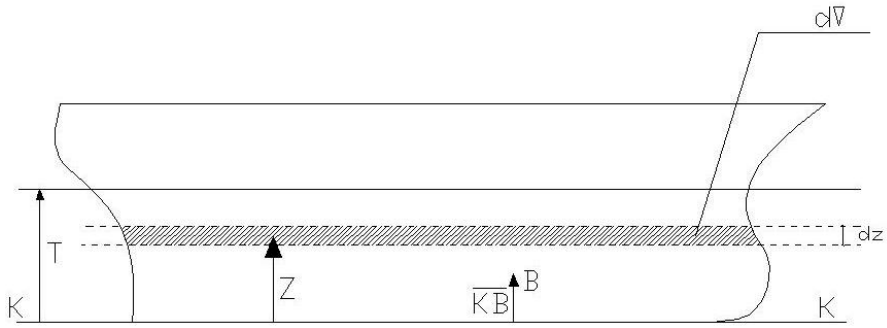

Imagen 16. Cálculo del KB mediante flotaciones

$$M_k = \int dM_k.$$

Siendo: $dM_k = z \cdot d\nabla = z \cdot A_f \cdot dz \quad \rightarrow \quad M_k = \int_0^T z \cdot A_f \cdot dz$.

Mediante la integración aproximada por el método de Simpson que nos quedará de la siguiente manera:

$$M_k \approx \frac{h}{3} \sum FS_i \cdot \left(z_i \cdot A_{f_i}\right) \text{ ó } M_k$$

De esta manera tenemos el M_k necesario para obtener el $\overline{KB}$ según la línea de referencia utilizada, siendo FS_i el factor multiplicador de Simpson y A_{f_i} el área de cada una de las flotaciones .

Para la obtención del $\overline{\otimes B}$ el método es el mismo, sólo que esta vez, la referencia que tomaremos para ello es el $\overline{\otimes b}$, de esta manera las ecuaciones que obtenemos serán:

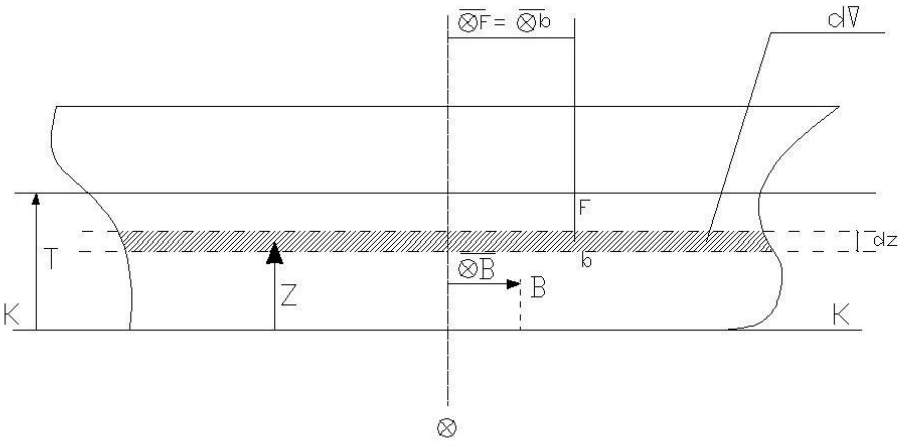

Imagen 17. Cálculo del OB mediante flotaciones

$$M_{\otimes} = \int dM_{\otimes}.$$

Siendo: $dM_{\otimes} = \overline{\otimes b} \cdot d\nabla = \overline{\otimes F} \cdot d\nabla = \overline{\otimes F} \cdot A_f \cdot dz$.

Y de ahí obtenemos el momento de todo nuestro volumen:

$$M_{\otimes} = \int_0^T \overline{\otimes F} \cdot A_f \cdot dz\,.$$

Utilizando el método de Simpson nos quedará de la siguiente manera:

$$M_{\otimes} \approx \frac{h}{3} \sum FS_i \cdot \left(\overline{\otimes F_i} \cdot A_{f_i}\right) \text{ ó } M_{\otimes}.$$

2. Otra forma es considerando las secciones de la flotación. De esa manera obtendremos igualmente los dos valores.

   Para el M_k calcularemos el momento del d∇ con respecto al plano base de la siguiente manera:

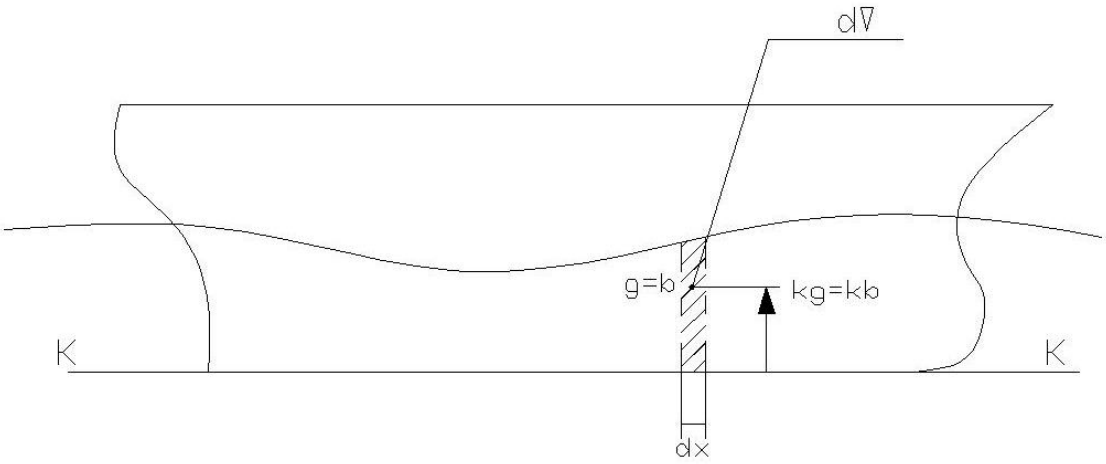

Imagen 18. Cálculo del KB por secciones

$$dM_k = \overline{kb} \cdot d\nabla = \overline{kg} \cdot d\nabla = \overline{kg} \cdot A_f \cdot dx.$$

El momento respecto a todo el volumen queda:

$$M_k = \int_0^T z \cdot A_f \cdot dz \; = M_k \int_L dM_k.$$

Como el producto $\overline{kg} \cdot A_s$ es precisamente el momento estático m_k del área de la sección A_s que estamos considerando, respecto al plano base, nos quedará

$$M_k = \int_L m_k \cdot dx.$$

Para el $M_{\otimes}$ debemos tomar el momento del volumen elemental genérico, d∇, con respecto a la sección media:

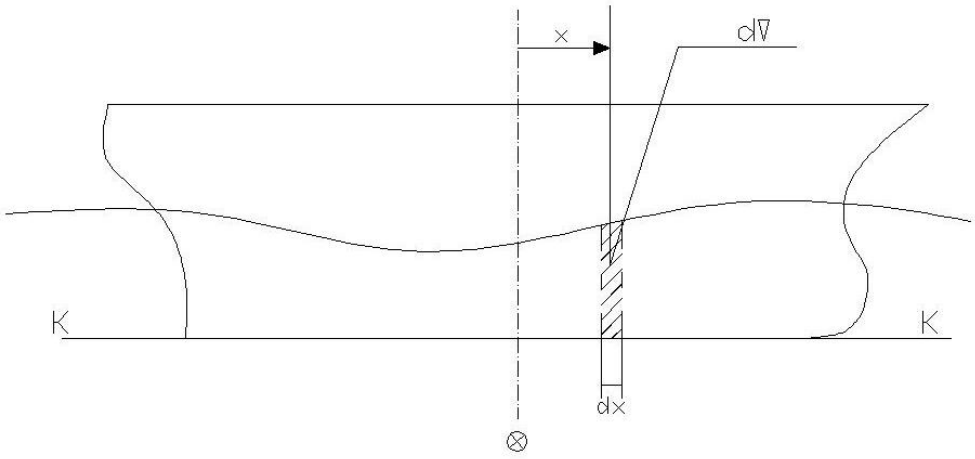

Imagen 19. Cálculo del OB mediante secciones

$$dM_{\otimes} = x \cdot d\nabla = \int_L x \cdot A_s \cdot dx.$$

Y el momento estático de todo el volumen de carena ∇ será:

$$M_{\otimes} = \int d\,M_{\otimes} \rightarrow M_{\otimes} = \int_L x \cdot A_s \cdot d_x.$$

Aplicando el método de Simpson nos quedará:

$$M_{\otimes} \approx \frac{h}{3} \sum FS_i \cdot x_i \cdot As_i \quad \text{ó} \quad M_{\otimes}.$$

Podemos simplificar esta ecuación, si coincide que x_i es un múltiplo sencillo de la clara (espaciado) entre secciones, h. De esa manera $x_i = n_i \cdot h$, que sustituyendo en las ecuaciones anteriores y sacando factor común queda:

$$M_{\otimes} \approx \frac{h^2}{3} \sum FS_i \cdot n_i \cdot A_{s_i} \quad \text{ó} \quad M_{\otimes}.$$

Para obtener la abscisa del c.d.c., una vez conocido el volumen ∇, sólo tendríamos que dividir:

$$\overline{\otimes B} = \frac{M_{\otimes}}{\nabla} \rightarrow \overline{\otimes B} \approx \frac{\frac{h^2}{3} \sum FS_i \cdot n_i \cdot A_{S_i}}{\frac{h}{3} \sum FS_i \cdot A_{S_i}}.$$

Simplificando:

$$\overline{\otimes B} \approx h\frac{\sum FS_i \cdot n_i \cdot A_{S_i}}{\sum FS_i \cdot A_{S_i}}.$$

3.4.6 Metacentro transversal

A partir de la situación de equilibrio del buque, al producirse una escora infinitesimal, se trazan unas rectas que unen las fuerzas de empuje vertical pasando por los centros de carena inicial y final. Estas se cortarán en un punto que se denomina metacentro. Como la situación de equilibrio inicial corresponde al buque adrizado, la línea de empuje para esta condición coincidirá con la línea central, y con el metacentro, ya que se sitúa sobre ella. Este metacentro se llama metacentro transversal inicial, M. Para los cálculos de estabilidad inicial, se supone que durante los primeros grados de escora, las líneas de empuje pasarán por ese punto M, afirmación que no es del todo cierta, aunque tampoco muy desacertada.

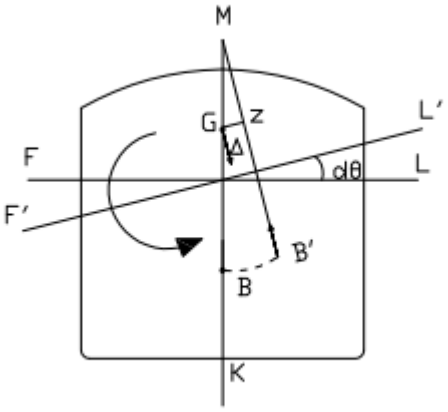

Imagen 20. Metacentro transversal

3.4.7 Radio metacéntrico transversal BM_t

El valor del radio metacéntrico transversal, $\overline{BM_t}$, se denomina así porque, haciendo centro en M y con radio $\overline{BM}$, la circunferencia trazada coincidiría, muy aproximadamente, con la curva que pasa por el centro de carena para escoras infinitesimales.

El valor del radio metacéntrico transversal se obtiene a partir de los valores de los movimientos transversal, longitudinal y vertical del centro de carena. El radio metacéntrico tiene como ecuación:

$$\overline{BM_t} = \frac{I_t}{\nabla}.$$

Siendo I_t el momento de inercia de la superficie de flotación con respecto al eje longitudinal y su método de cálculo:

$$I_t = 2 \cdot \int_L \frac{1}{3} \cdot y^3 \cdot dx.$$

Siendo y las semimangas de la flotación.

Por el método de la integración aproximada de Simpson quedaría

$$I_t \cong \frac{h^3}{3} 2 \cdot \int_L \frac{1}{3} \cdot y^3 \cdot dx.$$

3.4.8 Metacentro longitudinal

El metacentro longitudinal se obtiene de manera similar al metacentro transversal. Para una inclinación longitudinal, los empujes que pasan por la posición inicial y final del centro de carena, intersectarán en un punto denominado metacentro longitudinal. A partir de la situación de equilibrio para el buque sin asiento, el empuje correspondiente a un ángulo infinitesimal, cortará a la línea de empuje del centro de carena inicial en un punto, M_l, metacentro longitudinal inicial.

Dentro de los primeros grados de inclinación longitudinal, las diferentes líneas de empuje pasarán, prácticamente, por el punto M_l.

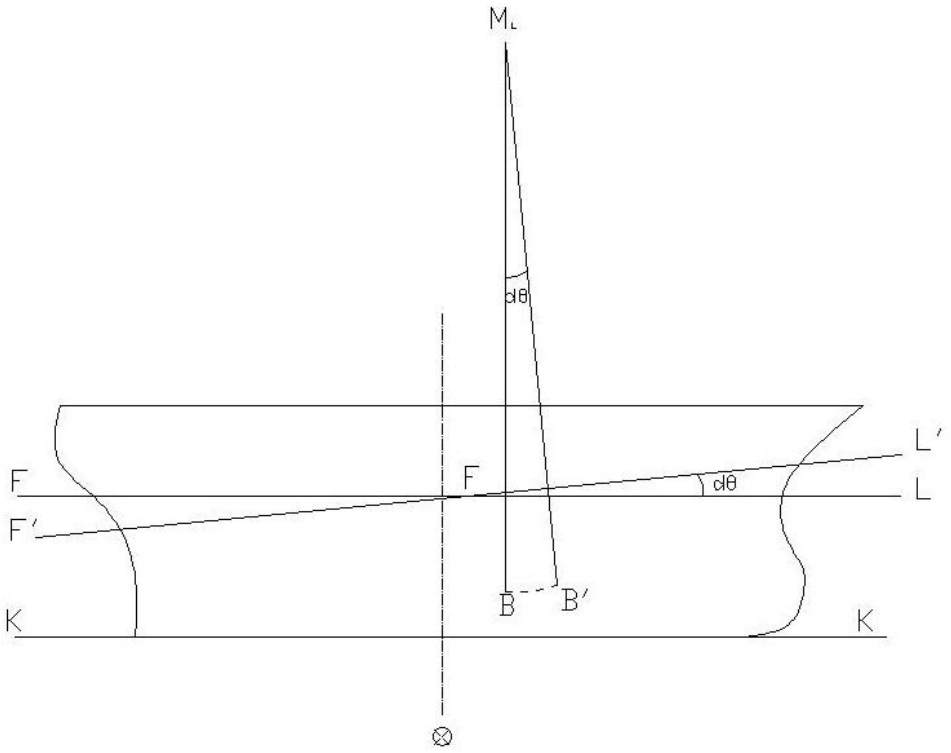

Imagen 21. Metacentro longitudinal

3.4.9 Radio metacéntrico longitudinal BM_l

El valor del radio metacéntrico longitudinal, $\overline{BM_l}$, se denomina así porque, haciendo centro en M y con radio $\overline{BM}$, la circunferencia trazada coincidiría, muy aproximadamente, con la curva que pasa por el centro de carena para escoras infinitesimales, hecho similar al que ocurre en el radio metacéntrico transversal.

El valor del radio metacéntrico longitudinal se obtiene a partir de los valores de los movimientos transversal, longitudinal y vertical del centro de carena. El radio metacéntrico longitudinal tiene como ecuación:

$$\overline{BM_l} = \frac{I_l}{\nabla},$$

$$I_l \cong \frac{h^3}{3}\int_L \frac{1}{3}\cdot x^2 \cdot y \cdot dx.$$

Siendo I_l el momento de inercia de la superficie de flotación con respecto al eje transversal y x la distancia de la manga de la flotación al eje de inercia.

3.4.10 Momento para alterar el trimado un centímetro MTC

Es el momento necesario para modificar el trimado del buque (diferencia de calados en las perpendiculares de proa y popa) un centímetro para cada flotación, y se calcula mediante la siguiente expresión:

$$MTC = \frac{I_l \cdot \gamma}{100 \cdot L}.$$

Siendo I_l el mismo momento de inercia que se utiliza para calcular $\overline{BM_l}$, γ el peso especifico supuesto para el fluido donde flota el agua y L la eslora del barco, que debería ser entre perpendiculares.

Se considera el ángulo de trimado que vamos a alterar, suficientemente pequeño como para considerarlo un infinitésimo a efectos prácticos. Si el buque gira, es por el efecto de un momento de trimado, este giro se realiza alrededor de un eje que pase por el centro de la flotación.

Los motivos pueden ser por causas externas o causas internas, aquí veremos los dos ejemplos.

- Causa interna, como un peso p que se traslada una distancia d.

Por el momento de transferencia, el centro de gravedad del buque se trasladará desde G hasta G_1. Esta traslación hace que el centro de carena deje de estar alineado con B. Por ello aparece un momento de trimado M_t.

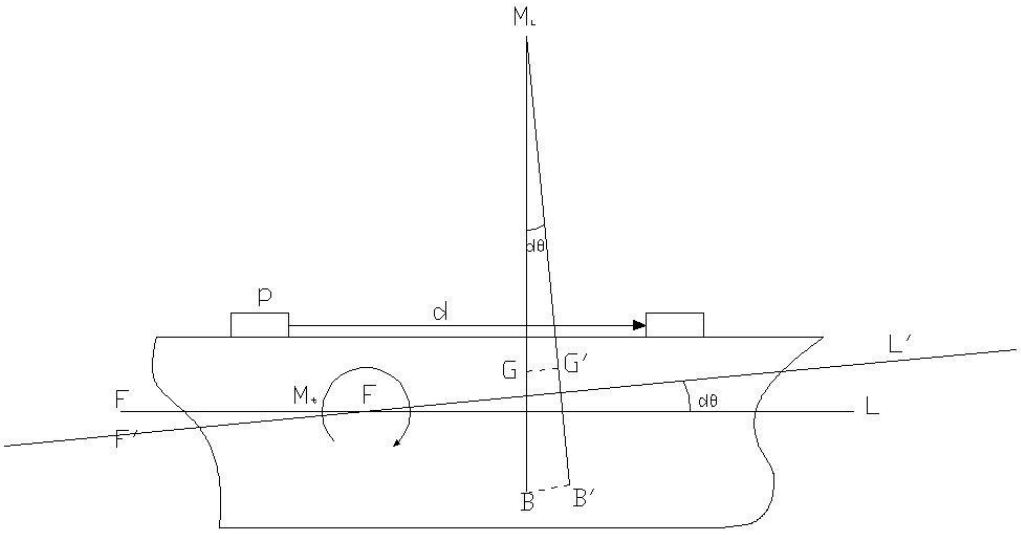

Imagen 22. Momento de transferencia

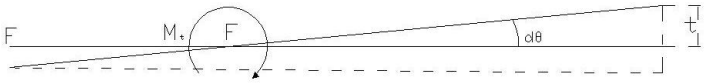

Imagen 23. Variación del trimado debido al momento de transferencia

$$M_t = \Delta \cdot \overline{GG'} = \Delta \cdot \overline{GM_l} \cdot tg\ d\theta,$$

$$tg\ d\theta = \frac{t}{L},$$

$$M_t = \Delta \cdot \overline{GM_l} \cdot \frac{t}{L}.$$

Si lo que se quiere, es que el trimado sea de un centímetro, para que $tg\ d\theta$ quede adimensional con la eslora en metros:

$$tg\ d\theta = \frac{t}{L} = \frac{1/100\,(m)}{L(m)} \rightarrow M_t = MTC = \Delta \cdot \overline{GM_l} \cdot \frac{1}{100 \cdot L},$$

$$MTC = \frac{\Delta \cdot \overline{GM_l}}{100 \cdot L}.$$

- Causa externa, cambio de centro de carena por el estado de la mar

Debido al momento de trimado externo, M_t, la flotación pasará de la indicada mediante $F - L$ a $F' - L'$ girando sobre el eje que pasa por el centro de la flotación. El centro de carena pasará de B a B' girando sobre el metacentro longitudinal M_l.

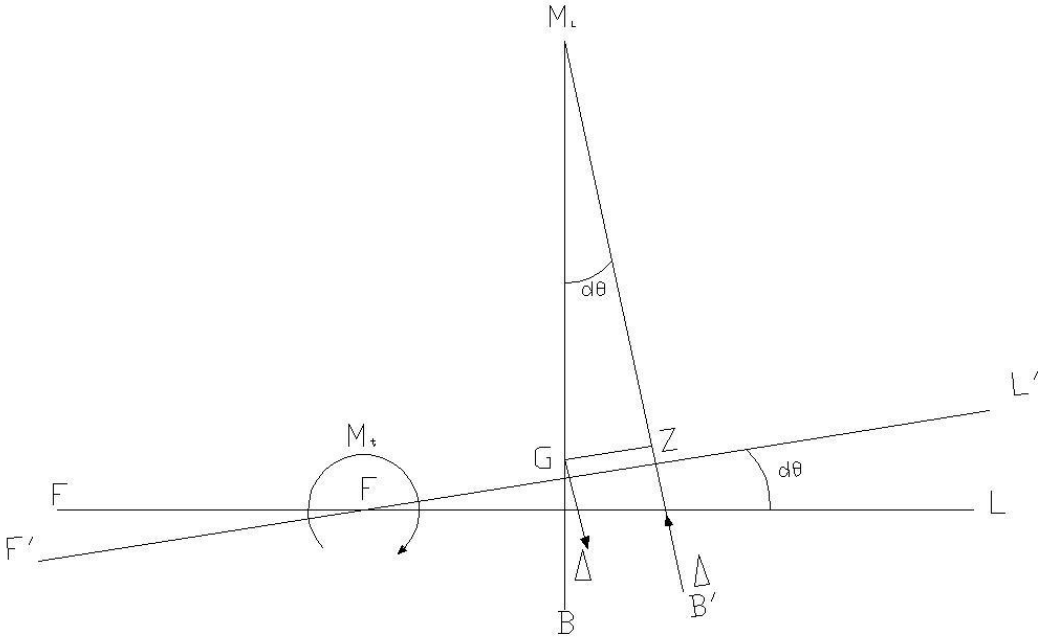

Imagen 24. Variación de trimado

En el ángulo de equilibrio final, el momento de trimado se verá equilibrado por el par adrizante $P_a = \Delta \cdot \overline{GZ} = M_t$. Pero:

$$\overline{GZ} = \overline{GM_l} \cdot sen\, d\theta \cong \overline{GM_l} \cdot \tan d\theta = \overline{GM_l} \cdot {}^{t}/_{L}.$$

Sustituyendo en la ecuación queda:

$$M_t = \Delta \cdot \overline{GM_l} \cdot {}^{t}/_{L}.$$

A partir de esta ecuación, si hacemos $t = 1\ centímetro$, y manteniendo adimensional ${}^{t}/_{L}$, nos queda la misma expresión que en la causa anterior:

$$MTC = \frac{\Delta \cdot \overline{GM_l}}{100 \cdot L}.$$

3.4.11 Curvas de Bonjean

Las curvas de Bonjean se utilizan para calcular el desplazamiento y la posición del centro de carena cuando se requiere una mayor aproximación al dato real en aquellos casos en que el asiento del buque difiera del asiento con el que han sido construidas las curvas hidrostáticas.

Sobre el plano longitudinal del buque se sitúan un número de cuadernas o secciones transversales equidistantes. Junto a cada cuaderna está la curva de áreas de su sección y los momentos verticales a diferentes calados con respecto a la cuaderna maestra.

Suponiendo el buque con los calados correspondientes a una flotación cualquiera, se traza ésta sobre el plano longitudinal, obteniéndose unos puntos de corte en cada una de las cuadernas. Desde estos puntos se trazarán unas paralelas a la línea base, dando sus intersecciones con las curvas, las áreas y momentos de cada sección para la flotación considerada. Aplicando cualquiera de los métodos aproximados de integración se hallará el volumen sumergido, el desplazamiento del buque y la posición longitudinal del centro de carena.

El uso de las curvas de Bonjean permite calcular el desplazamiento del buque para unos calados cualesquiera con asiento diferente al del cálculo de las curvas hidrostáticas, siendo más necesaria su utilización cuanto mayor sea la alteración de los asientos.

4. Botadura de un barco

Los barcos, después de su construcción hay que ponerlos a flote. Para ello existen varias maneras, la más popular entre los grandes barcos es el lanzamiento por popa. Ese será el procedimiento que vamos a estudiar. Los otros procesos de botadura de un barco son:

- **Botadura por puente móvil.** En este proceso el barco se sustenta mediante unas cinchas y se suspende en el aire para llevarlo al agua con una grúa o puente móvil grúa hasta que llegue al agua y se sueltan las cinchas.
- **Botadura por carro varadero.** Se sitúa el barco sobre una cama rodante dispuesta en la grada inclinada sobre la cual existen unos raíles que dirigen el barco y el carro al agua de manera controlada
- **Botadura en dique seco**. Para este proceso es necesario construir el barco en el dique seco, y una vez finalizada la construcción, se inunda el dique y el barco se pone a flote. Una vez igualados los niveles de agua dentro y fuera del dique, se abren las puertas del dique y se saca el barco del dique para vaciar el agua y dejar el dique dispuesto para otra nueva construcción.
- **Botadura por lanzamiento de costado.** Este proceso es el más parecido al que vamos a explicar detalladamente, pero con la particularidad del lanzamiento realizarlo de costado.
- **Botadura mediante plataforma de elevación vertical.** Se construye el barco en una zona próxima a la plataforma de elevación, y una vez finalizada la construcción se lleva a la misma mediante cama con ruedas para botar el barco. La plataforma está ayudada por chigres para su mejor control del proceso.
- **Botadura mediante dique flotante.** Se coloca el dique al final del muelle, se lleva el barco hasta el interior del dique y éste lleva el barco hasta una zona donde se producirá la botadura del barco. Se realiza mediante la inundación del dique y de esa manera el barco se quedará a flote libremente. Estas operaciones de inundación se realizan con bombas de lastrado que controladamente lo realizan todo para evitar perder la estabilidad del barco durante el proceso. El gran inconveniente de este proceso es el poder de fuerza ascensional que el dique posee.
- **Botadura mediante lanzamiento por popa.** Se realiza en grada inclinada y es el más utilizado para los buques de gran tamaño. Barco y cuna caen por gravedad de manera controlada hasta el agua y el barco queda a flote libremente. Este proceso es el que vamos a estudiar más detalladamente.

Los datos necesarios para el cálculo de la botadura por popa son: el peso del barco con su cuna, la inclinación de la grada, la distancia de la zona sumergida de la grada, la reacción de las imadas y la fuerza de rozamiento entre imadas y anguilas.

El barco comenzará a deslizar sobre la grada, una vez liberadas las llaves que lo sujetan, solamente si el conjunto barco mas cuna realiza una fuerza mayor que el coeficiente estático de imadas y anguilas.

Una vez el barco se desliza sobre la grada, las siguientes fuerzas que aparecen son: el empuje del agua sobre la zona sumergida y el rozamiento hidrodinámico que comenzará a frenar el barco. El momento del empuje del agua irá aumentando conforme el barco vaya introduciéndose más en el agua, hasta un límite donde el momento empuje en los santos de proa iguale al momento peso que genera el barco con respecto al mismo punto. En ese instante el barco comenzará a girar alrededor del extremo de proa de los santos de proa. El barco quedará apoyado en ese punto hasta flotar libremente.

La reacción que soportan los santos de proa es grande, de ahí que su estudio sea determinante para que la botadura salga como se prevee, y la estructura del barco está sometida a grandes esfuerzos. Para que la botadura sea exitosa, el giro debe haber concluido antes de que la resultante del peso del barco y cuna traspase el punto K que determina el final de la grada. Si no hubiese sucedido y el momento peso fuera mayor al momento empuje con respecto al punto K, el barco realizaría un giro indeseado denominado **arfada**. Este proceso transcurriría hasta que el momento empuje igualase de nuevo al momento peso y se corregirá el giro, golpeando al final del giro la proa contra la grada, ocasionando un posible deterioro indeseado a la estructura de los santos de proa y del barco en sí mismo.

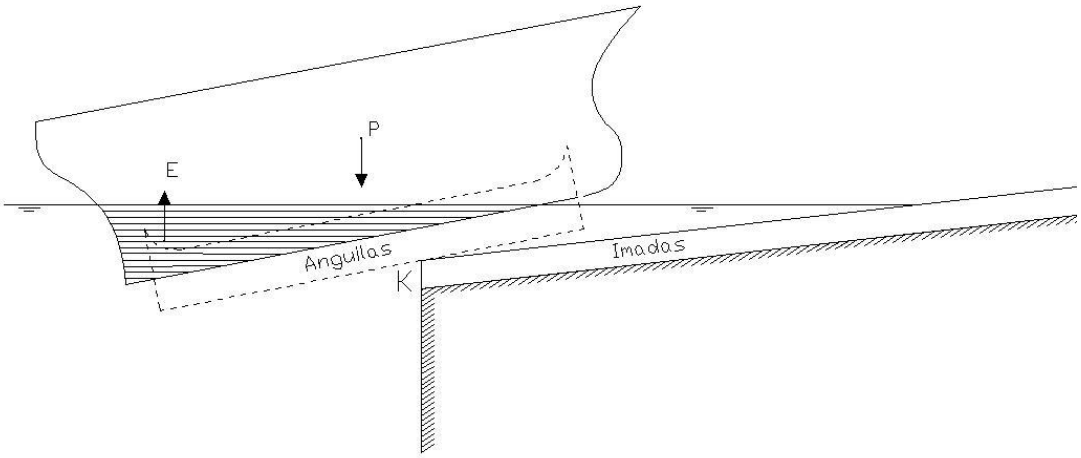

Imagen 25. Arfada

Más tarde nos podría acontecer otro problema llamado **saludo**. En él lo que hemos de tener en cuenta es que el barco ha de flotar libremente antes de llegar al final de la grada, si no ocurre esto podemos tener el problema de la falta de calado en proa y por lo tanto, cuando el barco finalice el trayecto en la grada, sumergirá la proa, pudiendo golpear indeseadamente ésta con la grada ocasionando graves desperfectos a la misma.

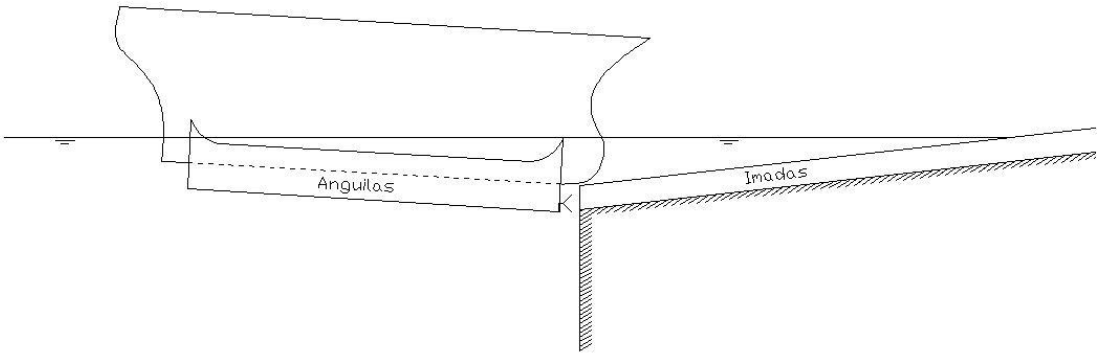

Imagen 26. Saludo

Por todo ello, hay una serie de condiciones que se deben estudiar y para eso se calculará previamente a la botadura:

- Fuerzas verticales que se desarrollan a lo largo del lanzamiento para determinar condiciones como giro, saludo, arfada, y conocer los valores de las reacciones que se dan en los apoyos.
- Análisis de los movimientos del barco para asegurar que durante el lanzamiento, no van a sufrir golpes ninguna de las partes del barco
- Estudio dinámico del lanzamiento, para determinar las condiciones de arranque y deslizamiento.

Por lo tanto se podría decir que hay cinco partes o etapas a estudiar en el lanzamiento por popa:

- Periodo durante el cual apoyan las anguilas sobre las imadas sin que el agua toque el barco o la cuna
- Periodo desde que el agua comienza a mojar el barco o la cuna hasta que inicia el movimiento de giro con respecto a los santos de proa
- Periodo durante el cual el barco solo se apoya longitudinalmente en los santos de proa, es decir, mientras se produce el giro.
- Periodo desde que acaba el giro y el barco deja de apoyarse en la grada hasta que empiezan a actuar los dispositivos de frenado.
- Periodo desde que empiezan a actuar los dispositivos de frenado hasta que el barco se para.

Para el análisis que vamos a realizar, lo primero que se han de conocer son los datos relativos al plano de lanzamiento, es decir, ángulo de inclinación de la grada. La pendiente nunca será muy grande, ya que, no se recomiendan unas velocidades altas, de esa manera no hay velocidad excesiva en el lanzamiento, y la altura del barco en la zona de proa de la grada, no es muy elevada. Pero debe ser una altura suficiente como para que el barco deslice libremente al soltar las llaves de retención y no se produzca enclavamiento.

Por otro lado se debe conocer la altura del agua en el final de la grada para saber si el barco quedará a flote antes de acabar el lanzamiento o se producirá el efecto indeseado de saludo. Ello variará con la situación de la marea a esa hora.

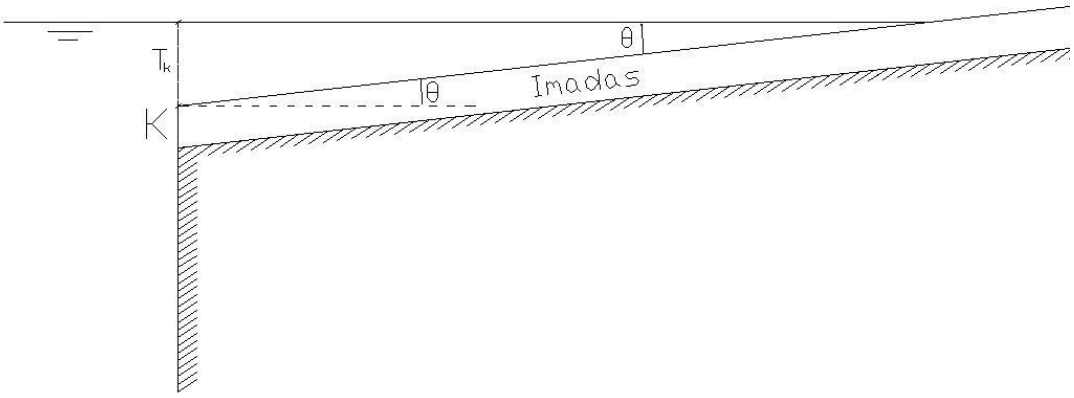

Imagen 27. Tk, Altura de la grada

Se debe conocer también la posición exacta del buque con la grada, tanto en posición perpendicular como longitudinal y el estado de pesos y posición del centro de gravedad de los mismos, esto nos dará la posición del centro de gravedad del buque y su valor. Si el valor y posición son indeseados, aún podremos modificarlos con lastres.

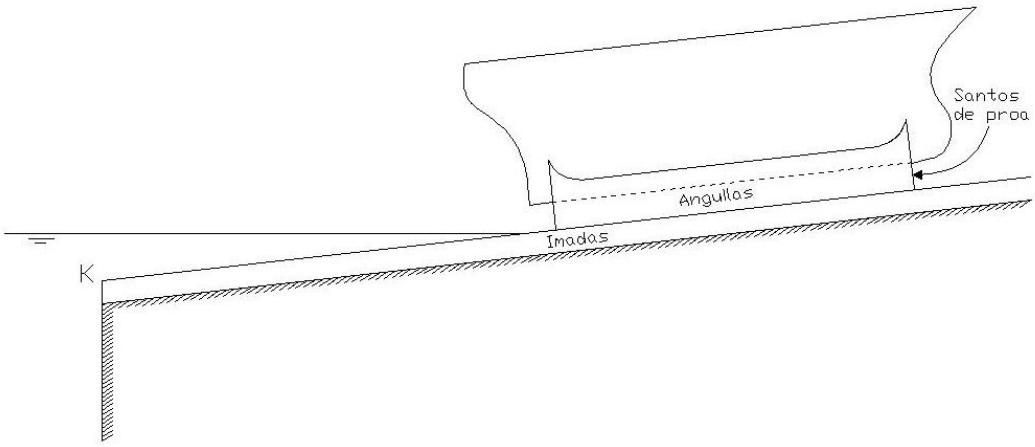

Imagen 28. Buque en grada para la botadura

También hemos de conocer el valor del empuje del barco en cada momento del deslizamiento, eso lo calcularemos, ya que conocemos la pendiente de la grada y estipularemos los empujes que sufre el barco y el momento empuje M_E.

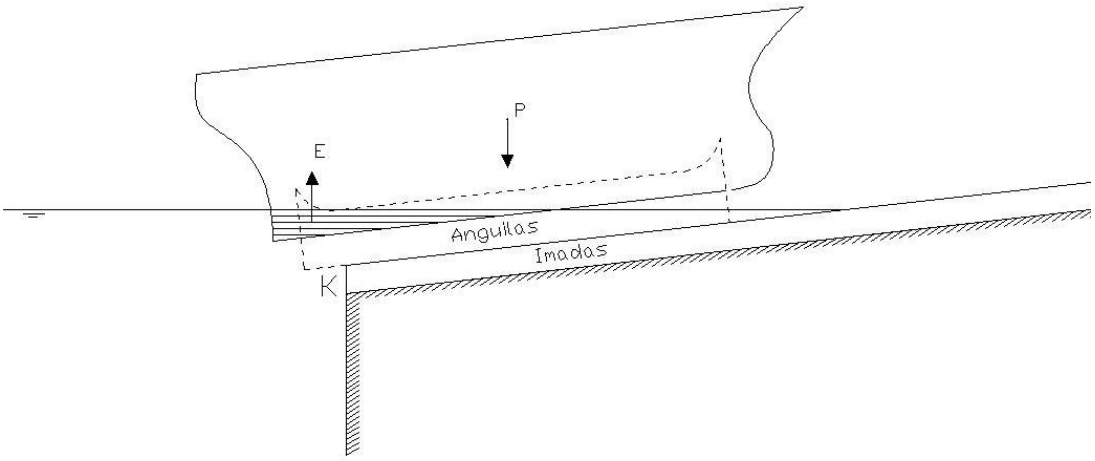

Imagen 29. Fuerzas sufridas por el barco en el momento de la botadura

Hemos de tener en cuenta el valor de los calados a proa y a popa del barco. Este dato nos será importante para calcular cuándo y cómo flotará libremente.

El cálculo de estos procesos se realizará matemáticamente a partir de los datos previos que se facilitan (peso y posición del centro de gravedad del barco) y los que calcularemos del proceso de botadura (empujes y centros de gravedad de los mismos).

Para obtener los empujes usaremos un programa matemático, Matlab, en el cual se realizarán los cálculos de la siguiente manera.

Una vez situado el barco en la grada, sabemos que los empujes iniciales se realizarán por flotaciones paralelas, todas ellas con la misma inclinación (la de la grada) hasta el momento en el que el barco comience a realizar el giro. De esa manera conoceremos el empuje realizado por el agua en la que se ha sumergido y el momento empuje que genera ese empuje. A partir del punto del inicio del giro, los cálculos no los realizaremos, ya que sabremos si el barco sufre o no arfada en ese momento.

$$M_p = M_E.$$

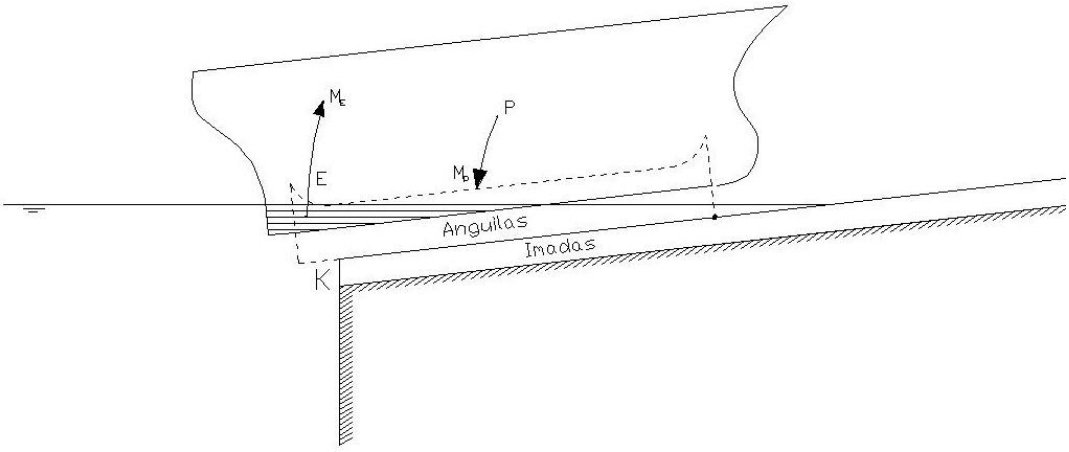

Imagen 30. Momentos sufridos por el barco en el momento de la botadura

El otro punto crítico de la botadura será el punto en el que el barco flota libremente. Para ello, sabiendo los calados a proa y a popa del barco, y conociendo los datos de la grada y el estado de la altura de la mar en el punto K de la misma, sabremos si se produce el indeseado **saludo**. Para ello, no utilizaremos esta vez las flotaciones inclinadas anteriores, si no, unas paralelas al fondo del barco.

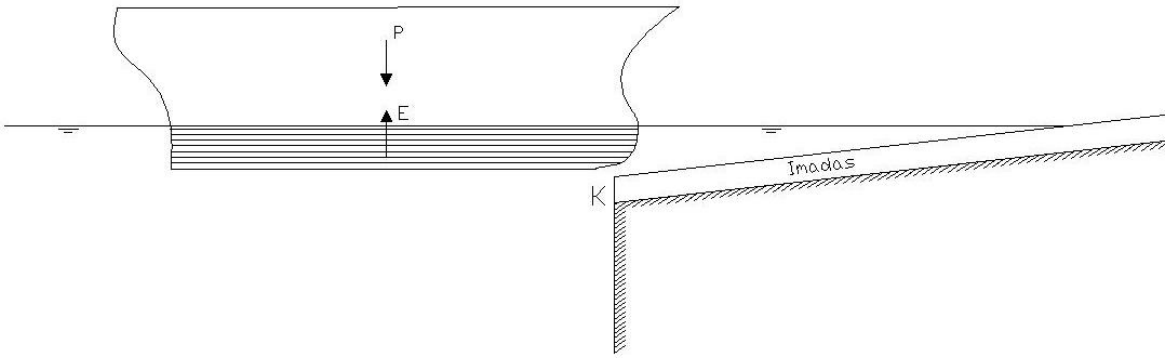

Imagen 31. Barco flotando libremente tras la botadura

De esta manera obtendremos mediante la integración aproximada el empuje que sufre el barco hacia arriba para que iguale al peso, y como sabremos los calados a proa y a popa del barco. Estos resultados los usaremos para saber si flota antes de que los santos de proa pasen el punto K de la grada.

5. Método de cálculo de la botadura

5.1 Introducción a los procesos a calcular

El programa utilizado para realizar los cálculos de este tipo de problema es Matlab. En él hemos de indicar los datos de la grada (inclinación, longitud de la zona de grada sumergida) la posición del barco con respecto al final de la grada, los pesos del barco y la posición de su centro de gravedad, la altura de las anguilas, la posición de las anguilas para conocer dónde queda el extremo de los santos de proa, los calados del barco a proa y a popa. De esa manera, seremos capaces de calcular el momento peso M_p que genera el barco y mediante los cálculos que se programarán en el trabajo, podremos saber si el barco realizará una botadura sin problemas.

Los cálculos que realizará el programa en la parte inicial del proceso, serán los volúmenes de desplazamiento del barco en posición adrizada. Conociendo el peso y centro de gravedad del conjunto del barco, se irán calculando los diferentes empujes que genera el barco y sus centros de empuje. En el momento en el que se igualen peso y empuje, el barco quedará a flote.

Pero los calados, no suelen ser iguales en proa y en popa, para ello, los centros de gravedad del conjunto pesos y del empuje han de estar en la misma perpendicular, cosa que extrañamente sucederá con el barco en posición de adrizado.

El proceso de cálculo de la botadura que vamos a usar en el programa es el siguiente:

A la entrada del barco al agua, éste sufre un empuje y un momento empuje. Esto se contrarresta con sus pesos y sus centros de gravedad. Por lo que se irá calculando el momento empuje M_E durante la entrada al agua y el momento peso. De esa manera, irá comprobando si se igualan para saber el momento en el que comienza el giro. Los volúmenes se toman en el barco con una inclinación igual a la de la grada, como se ve en las imágenes siguientes:

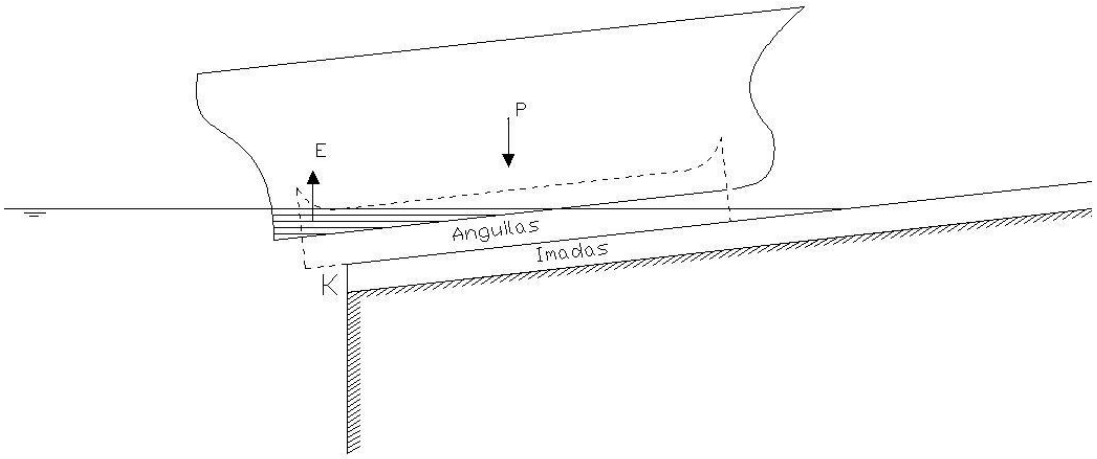

Imagen 32. Entrada del barco al agua en el proceso de botadura

De esa manera, el método de integración se realizará:

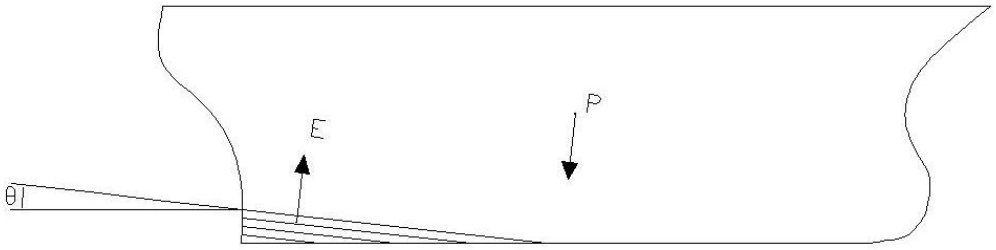

Imagen 33. Representación gráfica del proceso de sumergido inclinado

Se ha colocado el barco en posición adrizada, pero las líneas de los volúmenes de integración siguen la pendiente de la grada, de esa manera se calcula el empuje que sufre al entrar al agua. Si a este empuje le indicamos el punto sobre el cual genera un momento, se obtiene el momento empuje necesario para los cálculos.

Tras conocer el punto en el que el barco gira, se comprobará si ha sufrido o sufre arfada el barco en ese instante.

El siguiente cálculo a realizar, es el de flotación libre. Aquí hemos de tener en cuenta el valor de los calados a proa y a popa, y a partir de ahí, realizaremos el mismo tipo de integración, pero con un ángulo distinto. Este ángulo será igual a cero si los valores a proa y a popa del barco son iguales, en caso contrario, se calculará el valor del ángulo mediante la siguiente ecuación:

$$\alpha \rightarrow tg\ \alpha = \frac{t}{L} = \frac{T_{pr} - T_{pp}}{L}.$$

Siendo:

- t: Diferencia entre calados de proa y popa.
- L: Eslora entre perpendiculares.
- T_{pr}: Calado en proa.
- T_{pp}: Calado en popa.

El proceso de integración será similar al anterior, con la diferencia de que no es necesario el cálculo de los momentos que genera el barco y que la integración será con un ángulo diferente al de la grada o incluso cero.

Con inclinación nula, las líneas de los volúmenes de integración, serán como se indica en la siguiente figura:

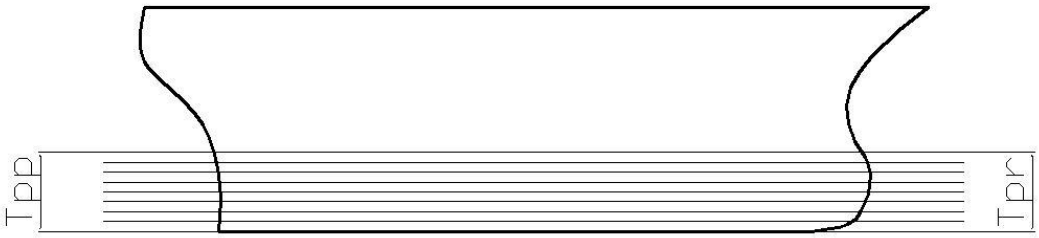

Imagen 34. Integración paralela en posición de adrizado

Si los calados son diferentes las líneas de los volúmenes de integración serán de la siguiente manera:

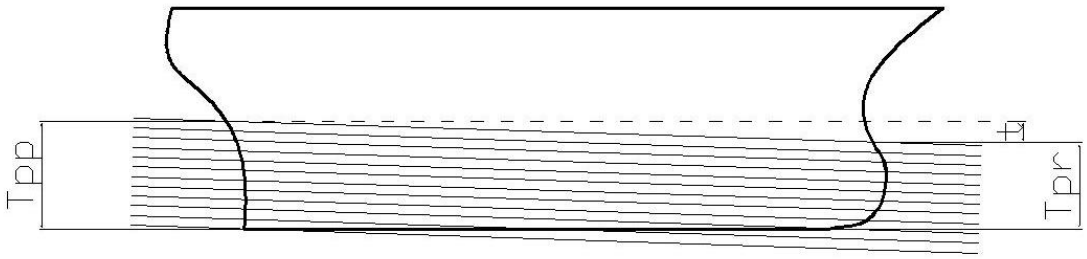

Imagen 35. Integración paralela en posición inclinada

Los datos de estas líneas de volumen los obtendremos del archivo .gf que obtendremos en Rhinoceros. Este archivo se sacará a partir de las formas del barco que hemos de tener previamente. Tras incluirlas en Rhinoceros tendremos todos los datos necesarios para hacer los cálculos que hemos indicado previamente.

5.2 Poner el barco en calados

El proceso de cálculo de los calados que vamos a seguir es el siguiente:

1) Cálculo de las curvas hidrostáticas del barco. En ellas se encontrarán datos para diferentes calados en la posición de adrizado del barco de:
   - Área de la flotación A_f
   - Toneladas por centímetro de inmersión (TCI)
   - Abcisa del centro de gravedad de la flotación, que puede ser desde la perpendicular de popa $(\overline{XF})$ o desde la sección media $(\overline{\otimes F})$
   - Volumen de trazado
   - Desplazamiento
   - Ordenada o altura del centro de carena $(\overline{KB})$
   - Abcisa del centro de carena desde la perpendicular de popa $(\overline{XB})$ o desde la sección media $(\overline{\otimes B})$
   - Radio metacéntrico transversal $\overline{(BM_t)}$
   - Radio metacéntrico longitudinal $\overline{(BM_l)}$
   - Momento para alterar el trimado un centímetro (MTC)
   - Coeficientes de forma

2) A partir de estos datos obtendremos los calados para cada desplazamiento. Una vez obtenidas las curvas hidrostáticas, entraremos con el peso del barco y obtendremos unos datos de calados del barco para su peso, puede ser directamente o mediante interpolación de los datos. En esos datos tendremos unos calados para proa y para popa iguales. Sabemos que esos calados solamente serán iguales si $\overline{XB} = \overline{XG}$, hecho que raramente sucede. Por lo tanto hemos de corregir los calados de proa y de popa. Esto se realizará después de obtener todos los datos de las hidrostáticas para ese desplazamiento, ya sea directamente obtenidos de la tabla que se realizará o mediante interpolación de los datos que tenemos en la tabla.

3) La corrección de calados se realizará también en Matlab. A partir de los datos hidrostáticos para ese peso, obtenemos un $\overline{XF}$ que nos indica el punto que hace de eje de giro del barco. De esa manera, y con los datos también obtenidos de los MTC del barco para ese calado, mediante las dos siguientes ecuaciones calculamos el trimado que sufrirá el barco para contrarrestar la variación longitudinal de las posiciones de $\overline{XB}$ y $\overline{XG}$.

$$MTC = \frac{\Delta \cdot \overline{GM_l}}{100 \cdot L} \approx \frac{\Delta \cdot \overline{BM_l}}{100 \cdot L} = \frac{I_l \cdot \gamma}{100 \cdot L}.$$

$$t = \frac{\Delta \cdot (\overline{\otimes G} - \overline{\otimes B})}{MTC}.$$

Siendo t el trimado total del barco.

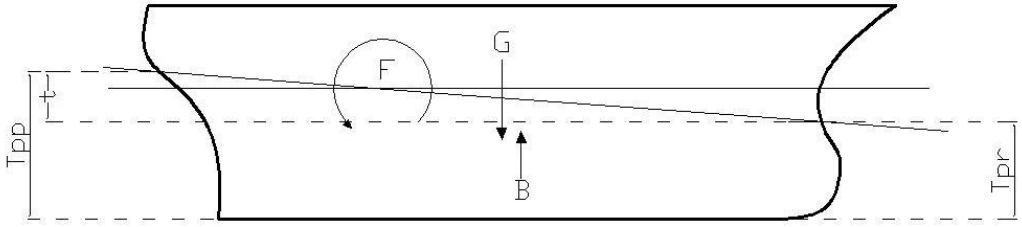

Imagen 36. Trimado total del barco

4) De esa manera, y mediante la ecuación

$$tg\varphi = \frac{t}{L_{pp}},$$

Siendo L_{pp} la eslora entre perpendiculares.

Podemos obtener el ángulo φ que es necesario girar el barco para conseguir los calados que hacen para ese peso $\overline{XB} = \overline{XG}$, una vez girado ese ángulo, cambian los datos de la flotación y por lo tanto el "eje de giro" del barco varía. Pero en nuestro caso, haremos solamente una corrección de calados.

5) Una vez realizado el giro, sabremos los calados a proa y a popa de nuestro barco gracias a las ecuaciones:

$$T_{pp} = T_m - a,$$

$$T_{pr} = T_m + f,$$

$$\frac{t}{L_{pp}} = \frac{a}{\frac{L_{pp}}{2} - \overline{\otimes F}},$$

$$\frac{t}{L_{pp}} = \frac{f}{\frac{L_{pp}}{2} + \overline{\otimes F}},$$

Suponiendo que: $T_{pp} < T_{pr}$.

Siendo T_m el calado medio obtenido a partir de los datos hidrostáticos, a la distancia vertical desde el T_m al T_{pp} en la perpendicular de popa, y , f la distancia vertical desde el T_m al T_{pr} en la perpendicular de proa.

6) Una vez obtenidos los calados, podemos abordar el problema de la botadura. Nuestra resolución del problema de la botadura necesita como datos iniciales:

- Formas del barco.
- Conjunto de pesos del barco.
- Posición vertical y longitudinal del centro de gravedad del conjunto de pesos.
- Posición del extremo de proa de los santos de proa.
- Altura del nivel del agua al final de la grada, ya sea, indicando la altura de la grada o la distancia de la grada.
- Ángulo de inclinación de la grada.
- Posición del buque en relación con la grada.

Las formas del barco hemos de obtenerlas a partir de Rhinoceros. Teniendo la carena en el programa, y con la opción AsociarDatosGHS podemos crear un archivo para exportar los datos de la carena del buque y usarlos en Matlab.

6. Obtención de las formas en un archivo adecuado para usarlo en Matlab

Lo primero de todo, tenemos que tener una carena que estudiar para hacer los procesos de cálculo en Matlab. La carena la cargaremos en Rhinoceros, puede estar formada por una o varias formas. En esta explicación usaremos una carena formada por más de una forma.

- Abrimos el programa Rhinoceros, cargamos las formas y comenzamos seleccionando el comando "AsociarDatosGHS". Para ello, habrá que escribirlo en la barra de comandos.

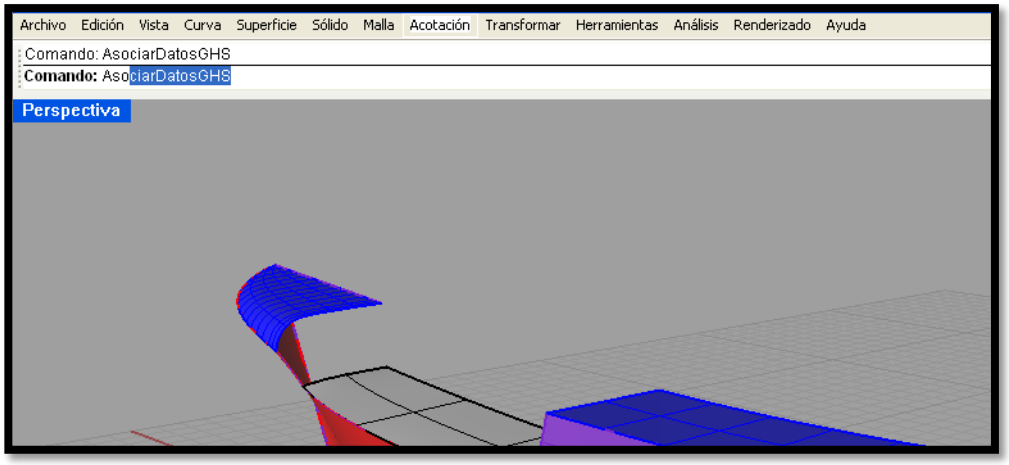

Imagen 37. Obtención de las formas a partir de Rhinoceros, paso 1

- Seleccionamos un título al grupo de formas que vamos a crear para mayor facilidad a la hora de usarlo y las unidades las colocamos en el sistema métrico, inicialmente aparecen en pies.

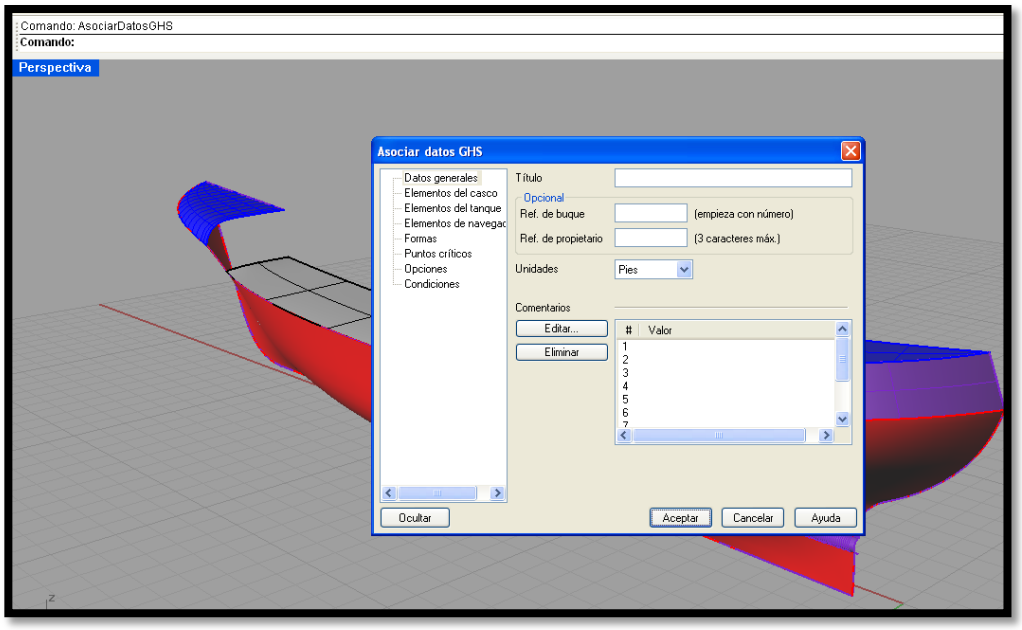

Imagen 38. Obtención de las formas a partir de Rhinoceros, paso 2

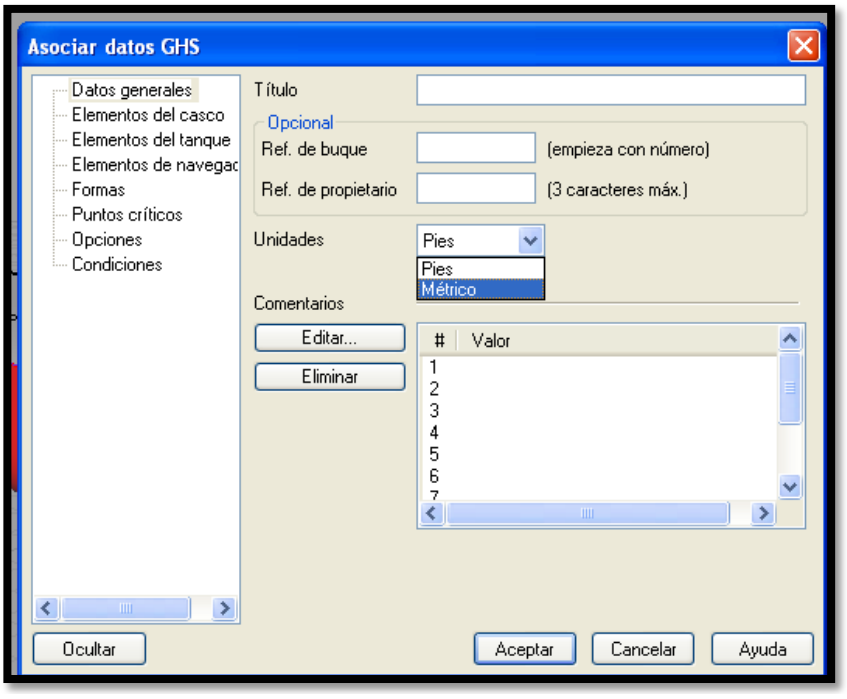

Imagen 39. Obtención de las formas a partir de Rhinoceros, paso 3

- Incluimos las partes de las formas que incluyen el casco, todas esas formas se irán almacenando en el conjunto de elementos del casco. Tenemos que tener en cuenta seleccionar todas las partes que queramos usar para el estudio de la carena y no incluir partes que no vayamos a usar para evitar excesivos cálculos y pérdidas de tiempo en el proceso de cálculo.

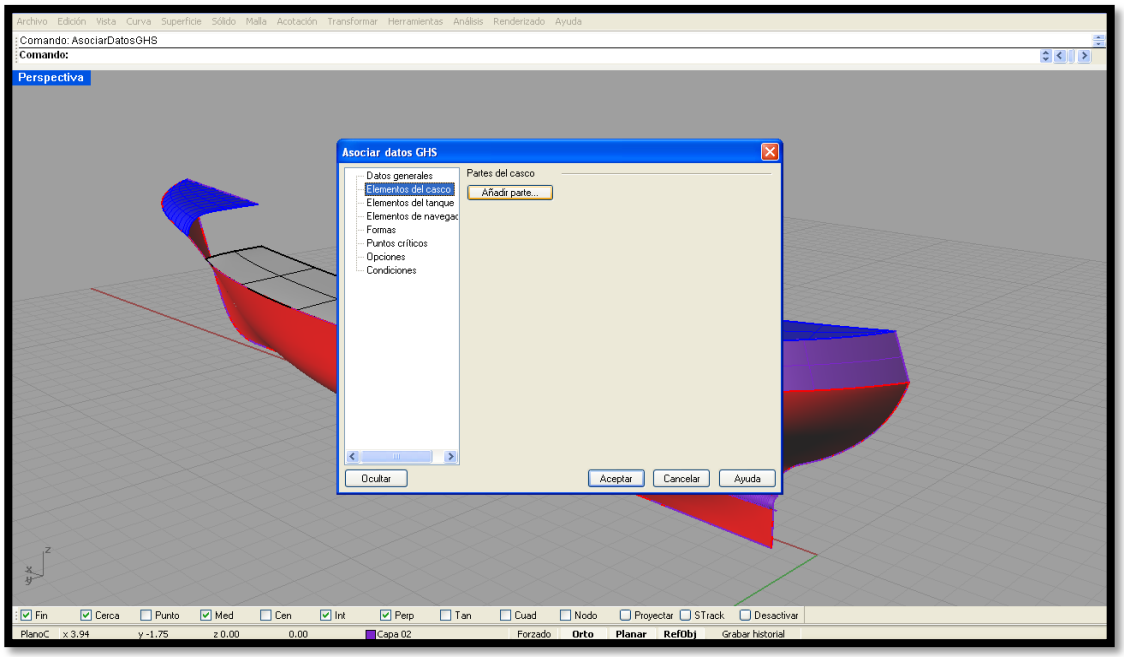

Imagen 40. Obtención de las formas a partir de Rhinoceros, paso 4

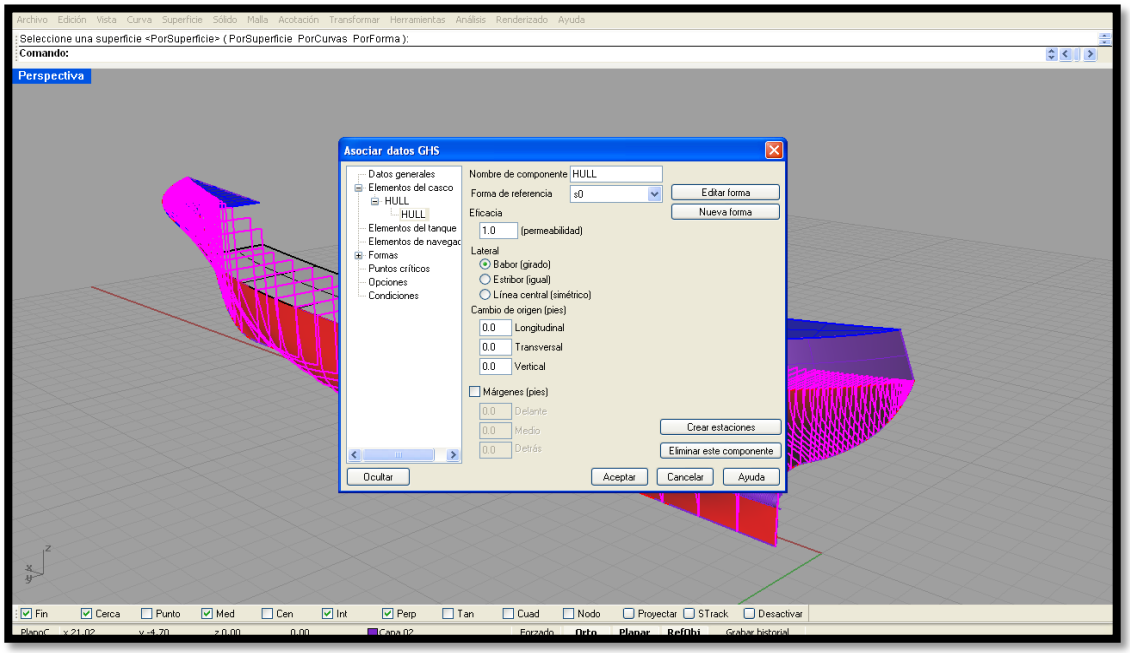

Imagen 41. Obtención de las formas a partir de Rhinoceros, paso 5

- Podemos renombrar todas las superficies para tener claro las que están incluidas y no inducir a error, ya sea dejando alguna sin incluir o incluyendo una de ellas más de una vez.

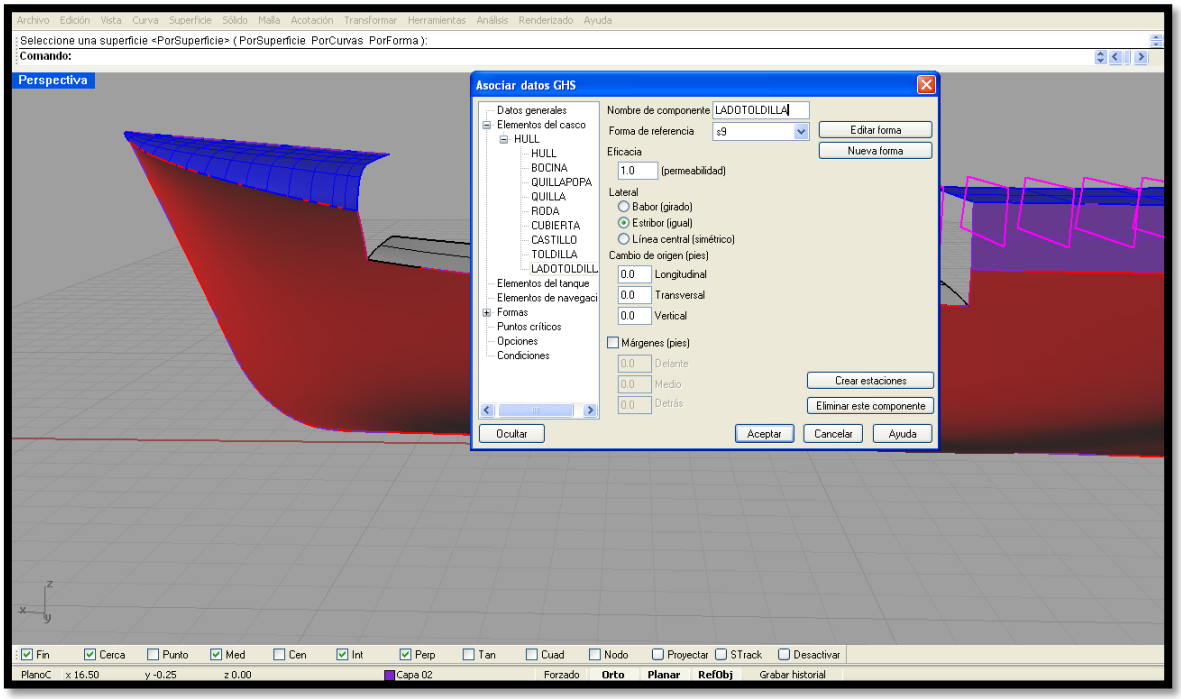

Imagen 42. Obtención de las formas a partir de Rhinoceros, paso 6

- Seleccionamos “Aceptar” y de esa manera tenemos guardadas las formas que hemos escogido. Una vez guardadas, se han de generar esas formas que deseamos estudiar en Matlab. Para guardar las formas en un archivo legible en Matlab, seleccionamos “Archivo” → “Guardar como” en el desplegable “Tipo” elegimos “Archivo de geometría GHS (.gf)” → “Guardar”.

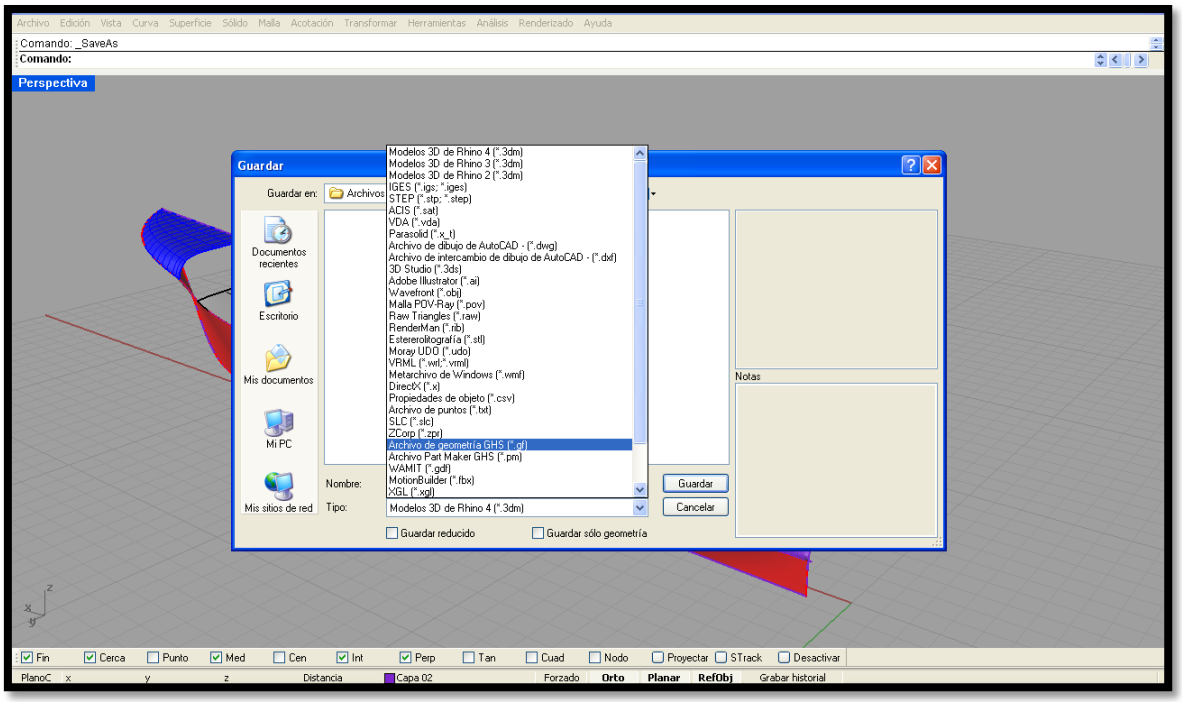

Imagen 43. Obtención de las formas a partir de Rhinoceros, paso 7

- Generado el archivo .gf, éste almacena los datos de las coordenadas de los puntos que forman las formas que hemos seleccionado para formar el archivo. Este archivo es el que se cargará más tarde en el programa de Matlab a partir del cual realizaremos todos los cálculos.

7. Obtención de datos en Matlab y comando "Ayuda" del programa informático

Tras obtener los datos de las coordenadas de los puntos que forman la carena que queremos estudiar, hemos de incluir en el programa Matlab todos los datos necesarios para estudiarlos y conseguir los resultados que buscamos.

Para este proceso, hemos de abrir inicialmente el programa de Matlab con el que trabajaremos. Todo este proceso va a ser explicado para la versión de Matlab "MATLAB R2007b".

7.1 Iniciar el programa

Inicialmente, abrimos Matlab como cualquier otro programa, bien sea seleccionándolo y dándole al botón Intro, o con doble click sobre el acceso directo que tengamos en nuestro ordenador.

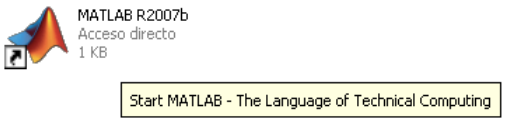

Imagen 44. Icono de MATLAB

A partir de ahí, se nos abrirá el programa de Matlab en el que nos aparecerá el editor, como en la siguiente imagen:

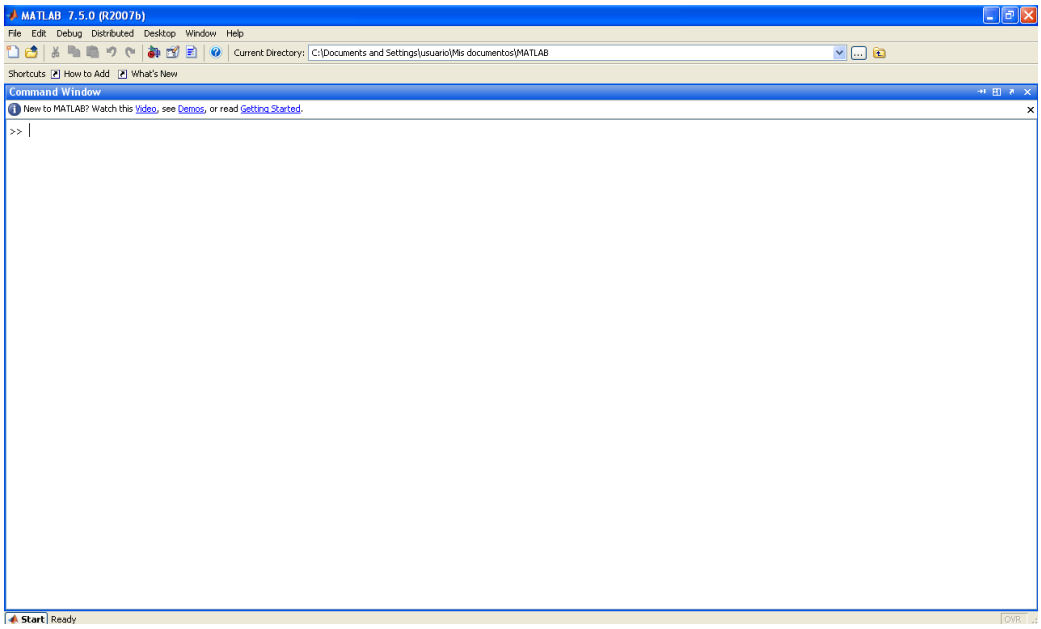

Imagen 45. Editor del programa MATLAB

Lo primero que se ha de verificar, es el directorio a partir del cual Matlab va a trabajar, en este caso es:

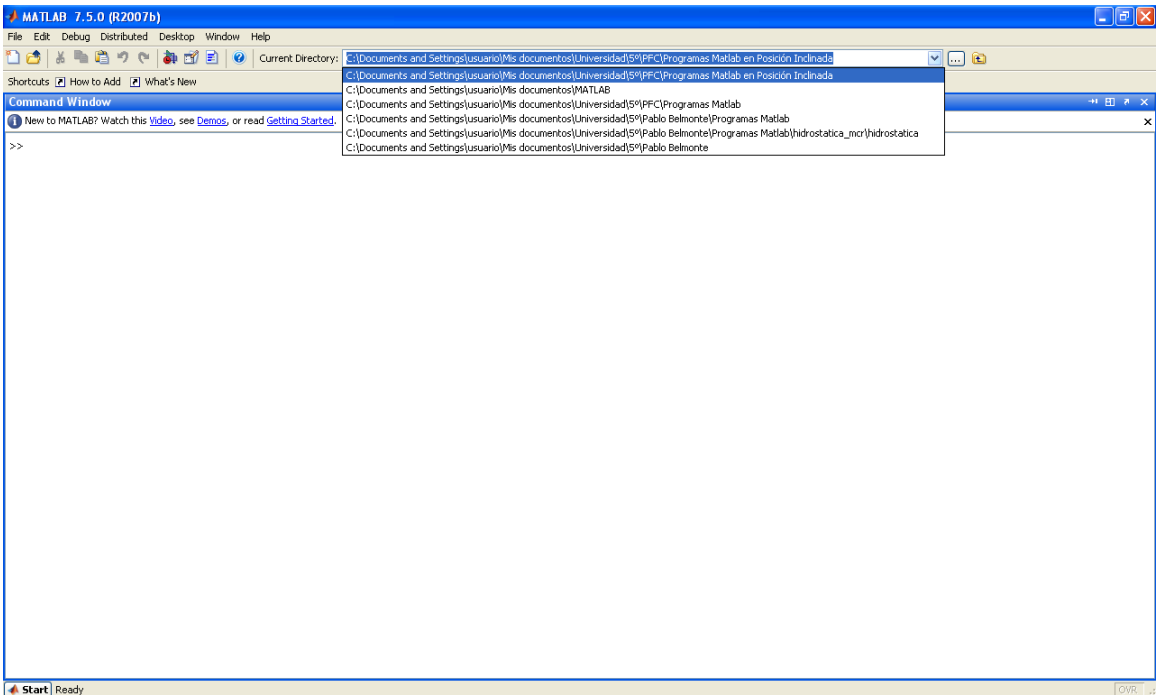

Imagen 46. Modificación del directorio en MATLAB

Para que el programa muestre los archivos que hay en el directorio, basta con escribir el comando “ls ”, presionar Intro, y mostrará en pantalla todo lo que el directorio contiene. Esto ayudará para no cometer errores tipográficos a la hora de la escritura de los comandos que se necesitan.

Para abrir el programa de Matlab que se ha de usar, hay que escribir “ValoresHidrostaticos” y presionar Intro.

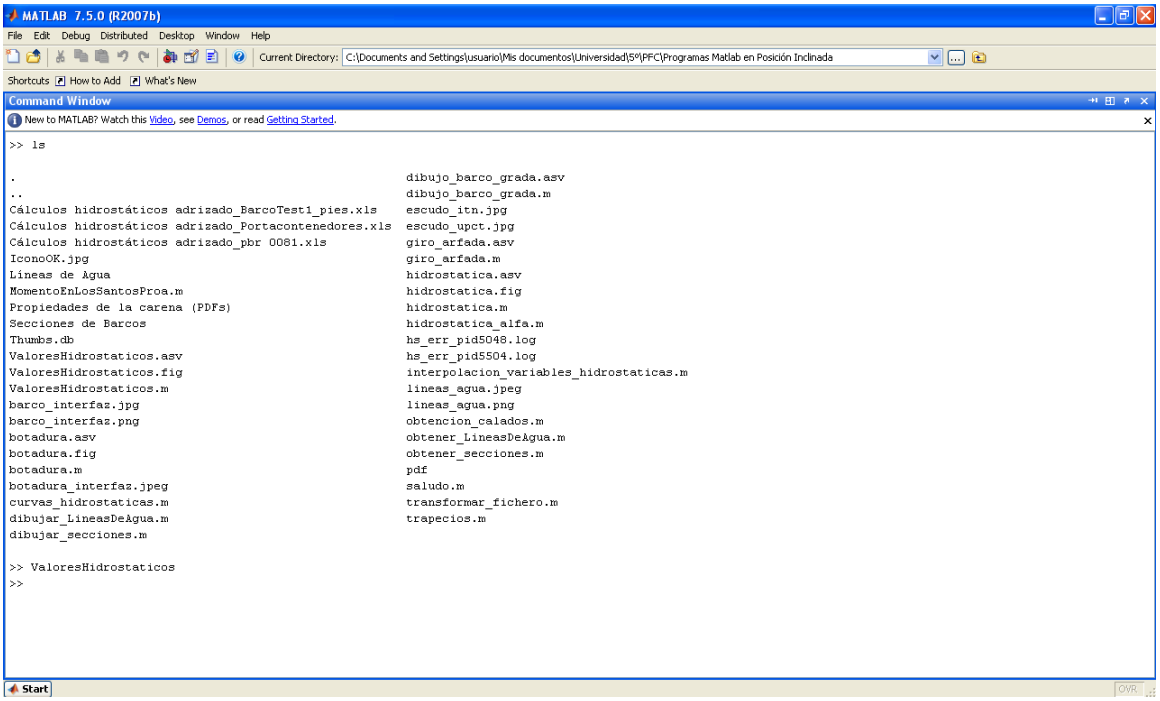

Imagen 47. Archivos del directorio seleccionado en MATLAB

De esa manera, se abrirá el programa con el cual se va a poder seleccionar los programas para obtener todos los datos hidrostáticos del barco o los datos de la botadura, como muestra la siguiente imagen.

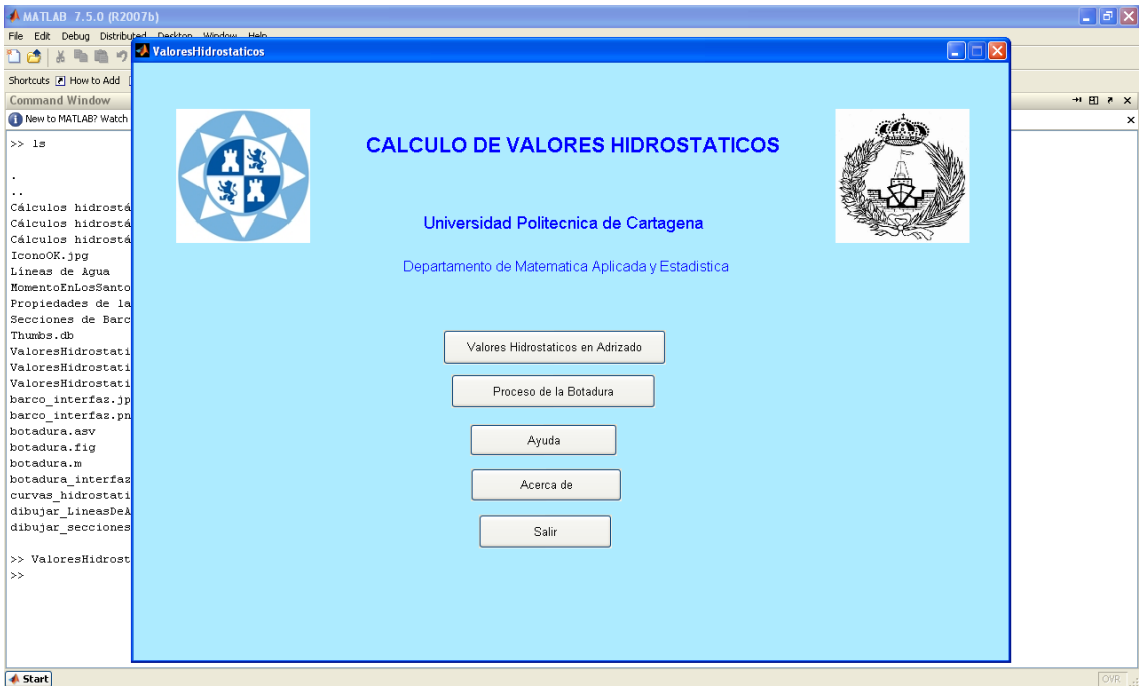

Imagen 48. Pantalla inicial del programa en MATLAB

7.2 Apariencia e inicio de las diferentes secciones iniciales.

Una vez se haya abierto el programa, hay cinco diferentes opciones.

- **Valores hidrostáticos en adrizado:** En esta sección se podrán obtener todos los datos hidrostáticos del barco.
- **Proceso de la botadura:** A partir de este botón, se abrirá el apartado relacionado con el cálculo de la botadura.
- **Ayuda:** Este botón abre un archivo .pdf el cual explica el uso del programa y los diferentes apartados.
- **Salir:** Botón destinado a cerrar el programa. Antes de cerrar el mismo realizará una verificación.

7.3 Cómo usar el apartado "Valores Hidrostáticos en Adrizado"

Al presionar el botón "Valores Hidrostáticos en Adrizado" aparece la siguiente pantalla:

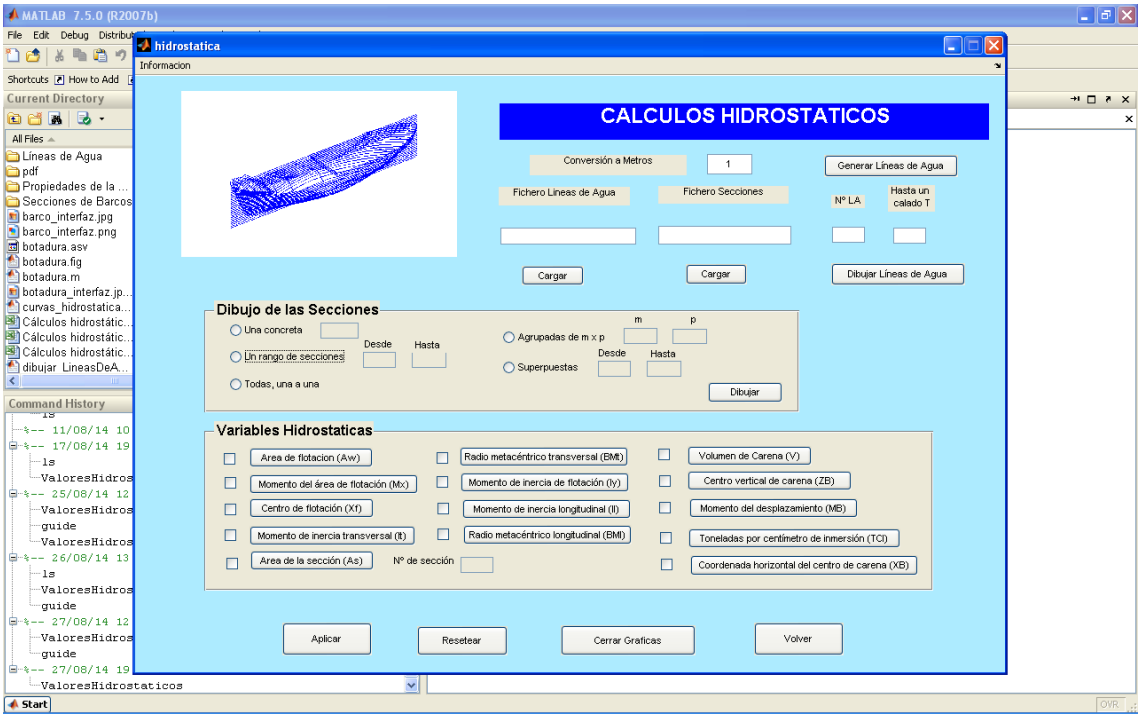

Imagen 49. Apartado "Valores Hidrostáticos en Adrizado"

En esta pantalla, se han de indicar los cálculos a realizar, y se han de incluir los datos a partir de los cuales se obtendrán dichos resultados.

7.3.1 Generar líneas de agua

Inicialmente, hay que generar el fichero de líneas de agua, a partir del archivo .gf que se genera en Rhinoceros. Para ello, se incluye el fichero .gf en el apartado "Fichero Secciones". Para generar las líneas de agua, se ha de seleccionar el archivo .gf en el apartado "Fichero Secciones" después de darle al botón "Cargar" que hay debajo de él. Tras presionar "Cargar", mostrará todos los archivos .gf que haya generados en la carpeta "Secciones de barcos". Se elige el archivo del que se van a obtener los datos, se especifica el número de líneas de agua y el calado máximo y se presiona el botón "Generar líneas de agua". El número de líneas de agua a generar y el calado máximo hasta el cual se va a crear hay que indicarlo en los espacios requeridos para ello. El número de líneas de agua se indica en el cuadro nombrado "Nº LA" y el calado máximo hasta el que se quiere calcular se indica en "Hasta un calado T".

Tras esto, el programa generará un archivo .xls con el mismo nombre que el anterior donde guardará las líneas de agua calculadas.

7.3.2 Dibujar líneas de agua

Para el dibujo de las líneas de agua, se necesitan los dos archivos cargados. El archivo .gf que previamente se debía haber cargado para generar el archivo .xls y este mismo archivo .xls generado en el cuadro de "Fichero líneas de agua". Tras cargar los dos ficheros, presionar el botón "Dibujar líneas de agua" y se generará el dibujo de las líneas de agua, indicando la altura del barco y el número de secciones como en la imagen 50.

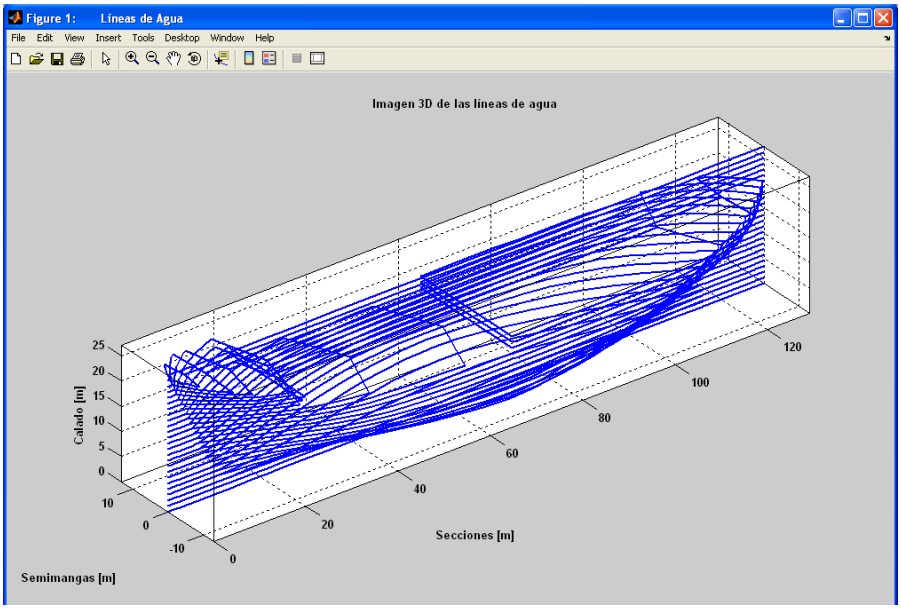

Imagen 50. Dibujo de las líneas de agua de un barco

En caso de haber seleccionado un calado superior al máximo de la embarcación, el programa advertirá mediante un mensaje de error indicando que se ha errado en la introducción de datos e indicará cuál es el máximo calado posible.

Imagen 51. Mensaje de error de calado

7.3.3 Dibujo de las secciones

Para el dibujo de las secciones se muestran cinco diferentes opciones.

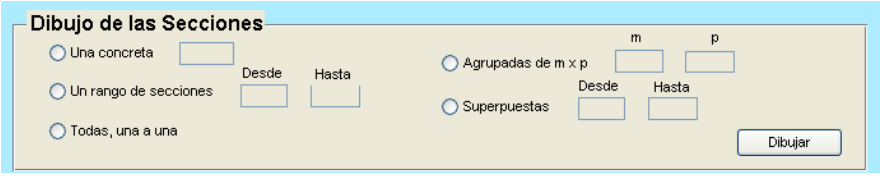

Imagen 52. Diferentes opciones de "Dibujo de las Secciones"

No se puede elegir más de una opción, solamente una de ellas. El resultado que se va a obtener de cada una es:

- **Una concreta:** Dibujará la sección que se le indique en el recuadro colocado para ella. Se puede tener una idea intuitiva de cual elegir, una vez se haya visto el barco dibujado en la imagen que se genera al presionar el botón “Dibujar”. La imagen que aparecerá es como la imagen 53, donde indica el número de sección y la distancia al origen x=0 en la parte superior de la imagen:

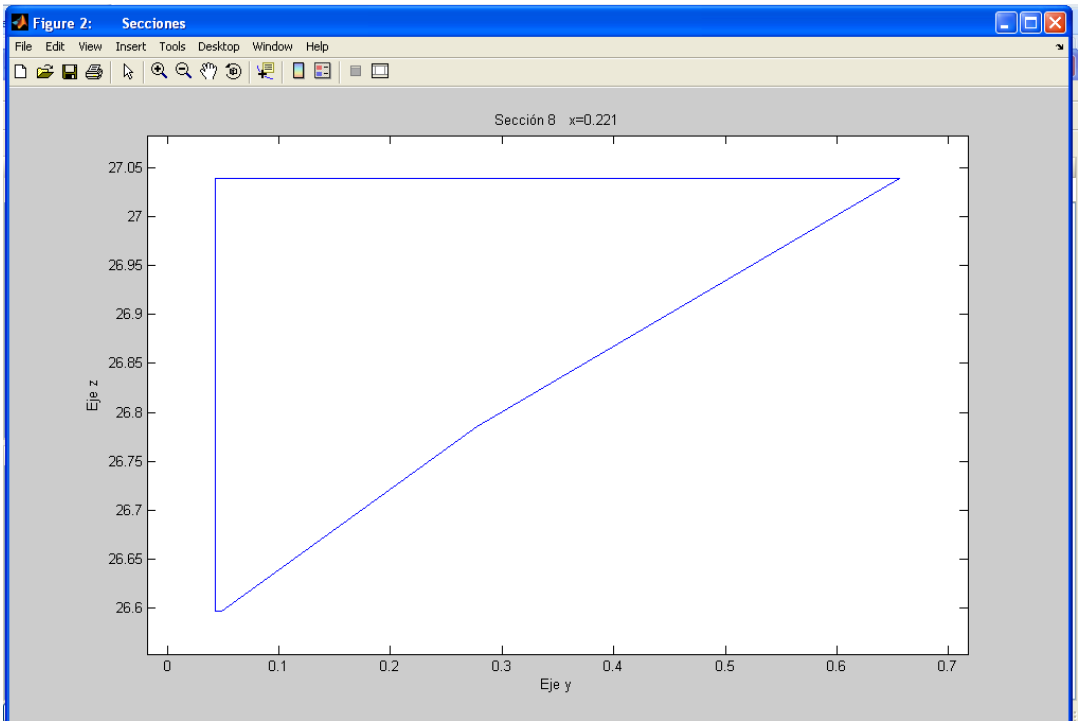

Imagen 53. Representación de una sección concreta

- **Un rango de secciones:** El programa mostrará las secciones que se le indiquen y haya entre las secciones indicadas en los recuadros “Desde” y “Hasta”. Irá mostrándolas desde la inicial indicada en el recuadro “Desde” hasta la última indicada en el recuadro “Hasta”. Para cambiar de una a otra se ha de indicar a partir de las flechas del teclado. Para cada sección el programa indica en la parte superior, el número de sección, la distancia de la sección al origen, y el tamaño de la sección a partir de los ejes Z e Y.

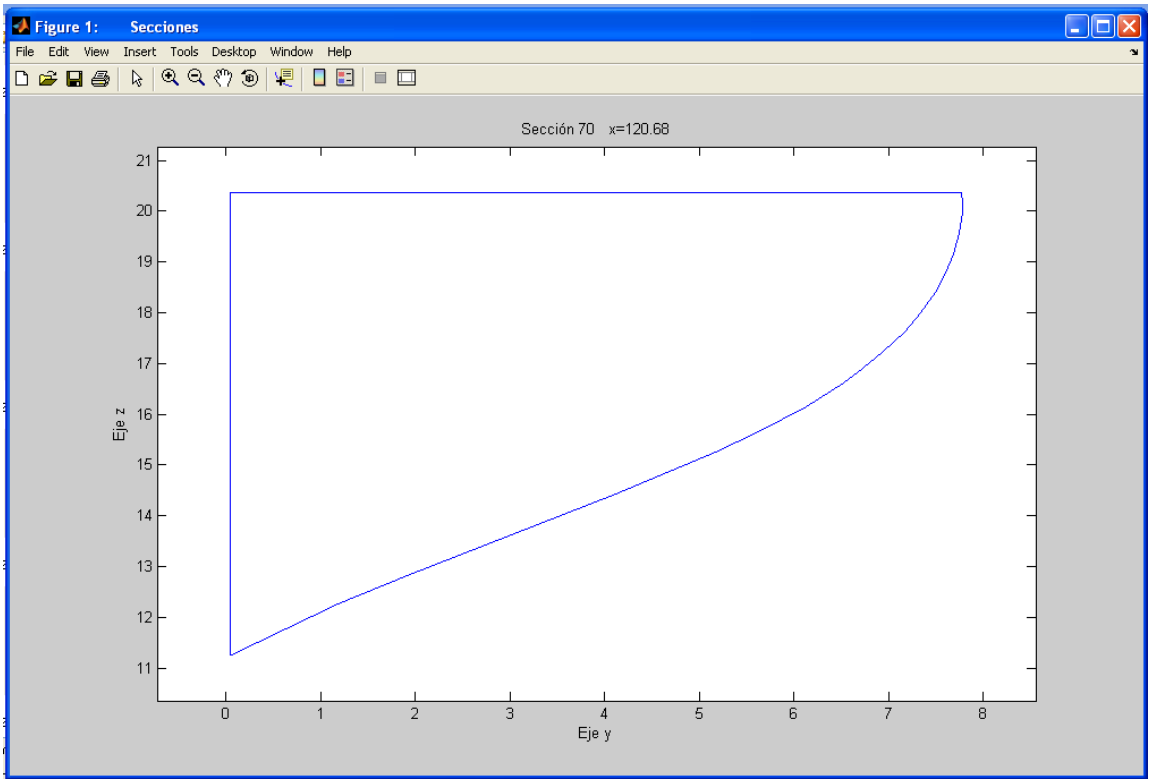

Imagen 54. Representación de una de las imágenes de la opción "Un rango de secciones"

- **Todas, una a una:** El programa mostrará todas las secciones del barco. No las superpondrá en una misma imagen, si no que las mostrará todas por separado como en las imágenes anteriores. Solamente se abre una ventana en la cual irá dibujando una a una conforme vayamos cambiando de imagen en las flechas del teclado de nuestro ordenador. La imagen que aparecerá al inicio es la de la sección 1, como en la imagen 55, donde apenas hay valores correspondientes a esta primera sección.

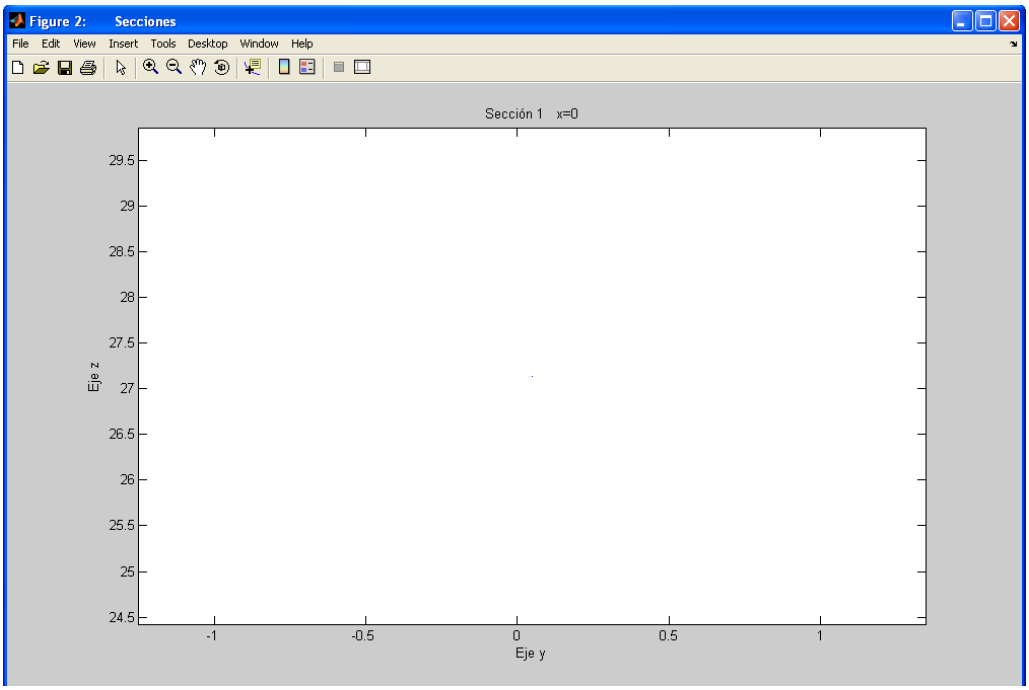

Imagen 55. Representación de la primera figura de la opción "Todas, de una en una"

- **Agrupadas de m x p:** Dibuja las secciones de manera agrupada en grupos de m filas y p columnas como se puede ver en la siguiente imagen, donde se ha seleccionado para esta ayuda; m=4 y p=4. Como seguramente todas las secciones no quepan en una misma imagen, para cambiar de un grupo de imágenes a otro, el proceso es el mismo que los anteriores. Presionando las flechas del teclado de nuestro ordenador.

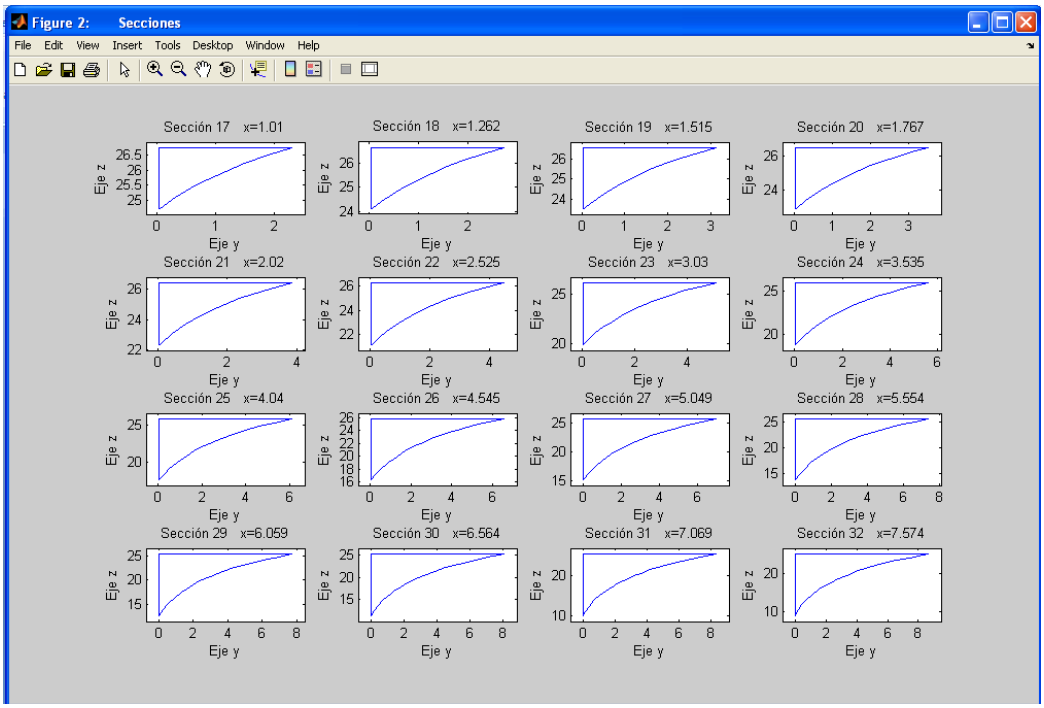

Imagen 56. Representación de las secciones de manera "Agrupada de m x p"

- **Superpuestas:** El muestreo de las secciones ahora se realizará de manera superpuesta, es decir, todas una sobre otra. Las secciones que dibujará el programa serán las que entren en el rango indicado en las casillas indicadas para ello. Tales casillas están mostradas con: “Desde” y “Hasta”. El resultado que se obtiene es similar al que aparece en la siguiente imagen una vez hayamos ido pasando desde la inicial hasta la final que queríamos ver:

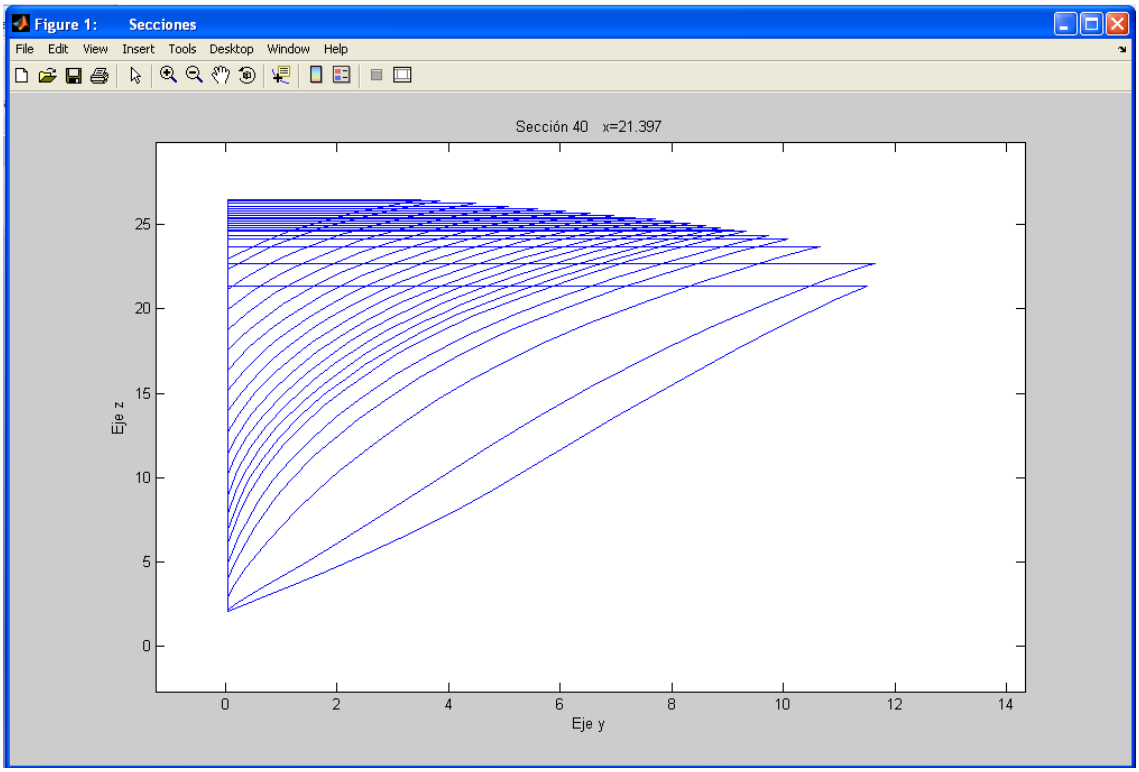

Imagen 57. Representación de las secciones "Superpuestas"

7.3.4 Variables hidrostáticas

Las variables hidrostáticas que se pueden obtener a partir del programa son las que aparecen en la siguiente imagen:

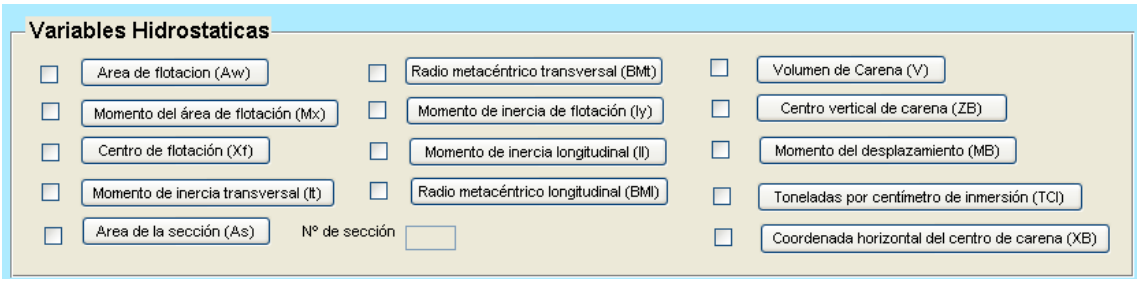

Imagen 58. Cuadro de selección de las Variables Hidrostáticas

Se han de indicar los datos que se quiere para que el programa muestre tras el proceso de cálculo. Para ello se han de seleccionar los recuadros de cada una de las siguientes opciones:

- **Área de la flotación:** Este valor dará el total del área encerrada por el contorno que forman la línea de flotación del barco en el calado indicado, con el casco del barco. La magnitud de este valor se mostrará en metros cuadrados.
- **Momento del área de flotación:** El valor que muestra es el momento de inercia del área de la flotación, calculada a partir del eje transversal o eje OY para el calado que se haya seleccionado.
- **Centro de flotación:** Indicará la posición del centro de carena para ese calado, referenciándolo al inicio del eje de coordenadas, es decir, al punto 0. Ese punto estará en el punto más a popa del barco. Por lo tanto, mostrará la distancia a la que el centro de flotación está situado desde el punto más a popa del barco.
- **Momento de inercia transversal:** Este dato mostrará el valor del momento de inercia transversal del barco al calado seleccionado. Este momento se calculará a partir del eje longitudinal que cruza el barco de proa a popa por la sección media, para una altura igual al calado de cálculo. Se obtendrá a partir de los datos de las semimangas del área de la flotación para ese calado.
- **Área de la sección:** En caso que querer obtener el área de alguna sección en concreto en este apartado, se puede elegir el número de la sección a calcular en el apartado "Nº de sección" y de esa manera mostrará el programa el valor de la sección indicada.

- **Radio metacéntrico transversal:** El valor del radio metacéntrico da un valor intuitivo de la estabilidad del barco. A mayor radio metacéntrico, mayor beneficio para la estabilidad, en este caso, para la estabilidad transversal. Este valor se dará en metros.
- **Momento de inercia de flotación:** Si se selecciona esta opción, se obtiene el valor del momento de inercia de la flotación para el calado que se haya indicado en el programa para el proceso.
- **Momento de inercia longitudinal:** Este momento de inercia, dará el valor del momento de inercia referenciado al eje OX o eje transversal. Este dato se usará para calcular el radio metacéntrico longitudinal, por lo que es un dato importante para el cálculo de la estabilidad longitudinal.
- **Radio metacéntrico longitudinal:** El valor del radio metacéntrico longitudinal es el valor que indica la estabilidad longitudinal del barco, aunque normalmente el barco no sufre mucho en este aspecto, siempre es un dato de gran relevancia e interés.
- **Volumen de carena:** El valor del volumen de carena varía con el calado y las formas del casco. Muestra el volumen de agua desalojado por el barco para el calado indicado en el proceso. A partir del mismo se podrán obtener gran cantidad de datos, incluyendo la capacidad de carga.
- **Centro vertical de carena:** Indica el valor de la altura de la posición del centro de carena respecto del eje con altura 0, el cual coincide con el punto más bajo del barco. En la mayoría de los casos coincide con la posición de la quilla.
- **Momento de desplazamiento:** Se calcula a partir de integración vertical a partir de la sección maestra o de la sección de la perpendicular de popa.
- **Toneladas por centímetro de inmersión:** Las TCI o toneladas por centímetro de inmersión, indican las toneladas que hay que cargar (descargar) para que el barco inunde (emerja) un centímetro su calado actual. Este valor es diferente para cada calado.
- **Coordenada horizontal del centro de carena:** Muestra la distancia desde la posición más a popa del barco al centro de carena para cada calado. Este valor se da para calados en posición adrizado, no en posición inclinada en esta sección.

El método de selección de las opciones, se muestra en la imagen siguiente, donde solamente se ha indicado en esta ocasión que muestre el área de la flotación:

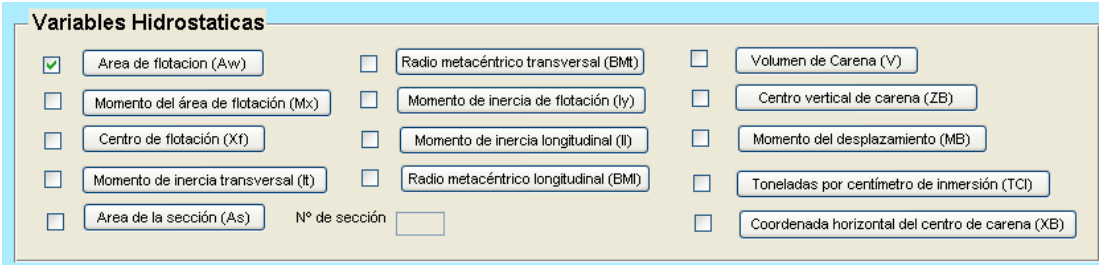

Imagen 59. Selección de alguna de las Variables Hidrostáticas

Para que el programa realice los cálculos, hay que darle al botón “Aplicar” que aparece en la parte de debajo de la pantalla como muestra la imagen 60.

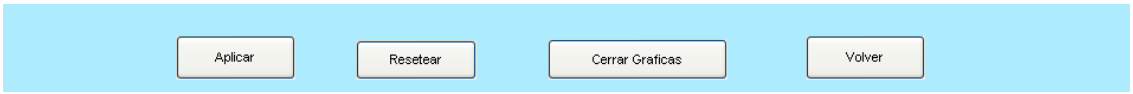

Imagen 60. Botones de ejecución del programa

Una vez se hayan introducido todos los datos del barco, junto con los archivos necesarios para el cálculo de todos los datos. Si se selecciona que represente todas las Variables Hidrostáticas de la imagen 59, los datos que aparecerán serán los que se pueden ver en las imágenes siguientes. Cada una representa un dato diferente para los diferentes calados de cálculo, y la imagen 61 muestra los valores de todas ellas numéricamente para el calado final de trabajo.

7.3.4.1 Valores hidrostáticos

En la ventana de "Valores Hidrostáticos" aparecerán los datos de todas las variables seleccionadas, expresadas numéricamente para la línea de flotación de calado máximo indicado.

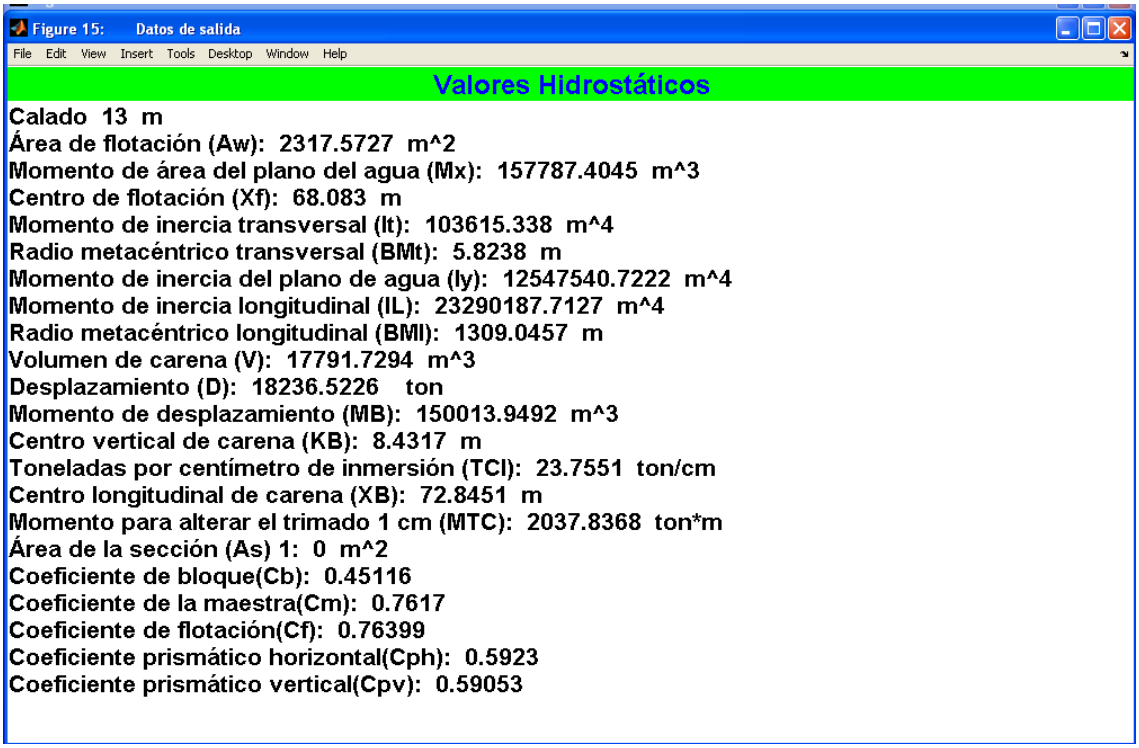

Imagen 61. Datos numéricos de los Valores Hidrostáticos

7.3.4.2 Coordenada x del centro de carena

En esta imagen se muestra la variación de la coordenada x del centro de carena, o $\overline{XB}$ con respecto a cada calado. Esta variación la muestra gráficamente. En la columna de la izquierda se muestran los diferentes calados, y en la línea inferior los datos de la coordenada del centro de carena.

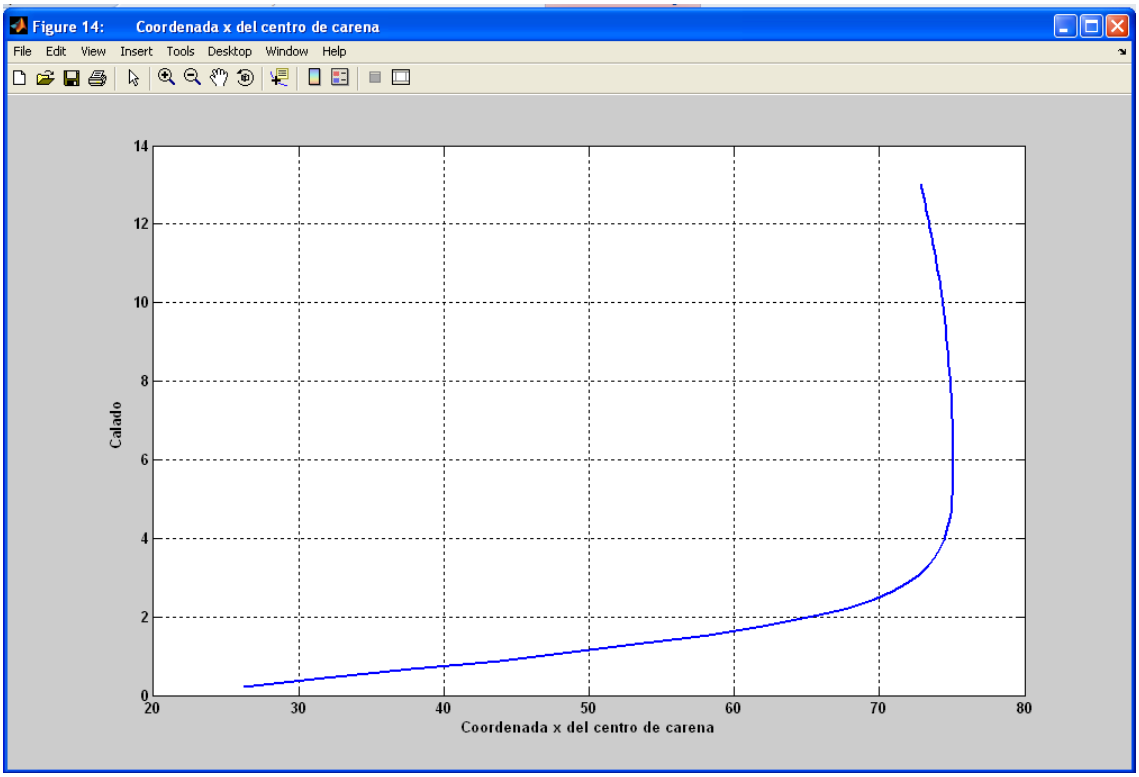

Imagen 62. Representación gráfica de la variación de la coordenada x del centro de carena

7.3.4.3 Radio metacéntrico longitudinal

En esta imagen se muestra la variación del radio metacéntrico longitudinal, o $\overline{BM_l}$ con respecto a cada calado. Esta variación la muestra gráficamente. En la columna de la izquierda se muestran los diferentes calados, y en la línea inferior los datos del radio metacéntrico longitudinal.

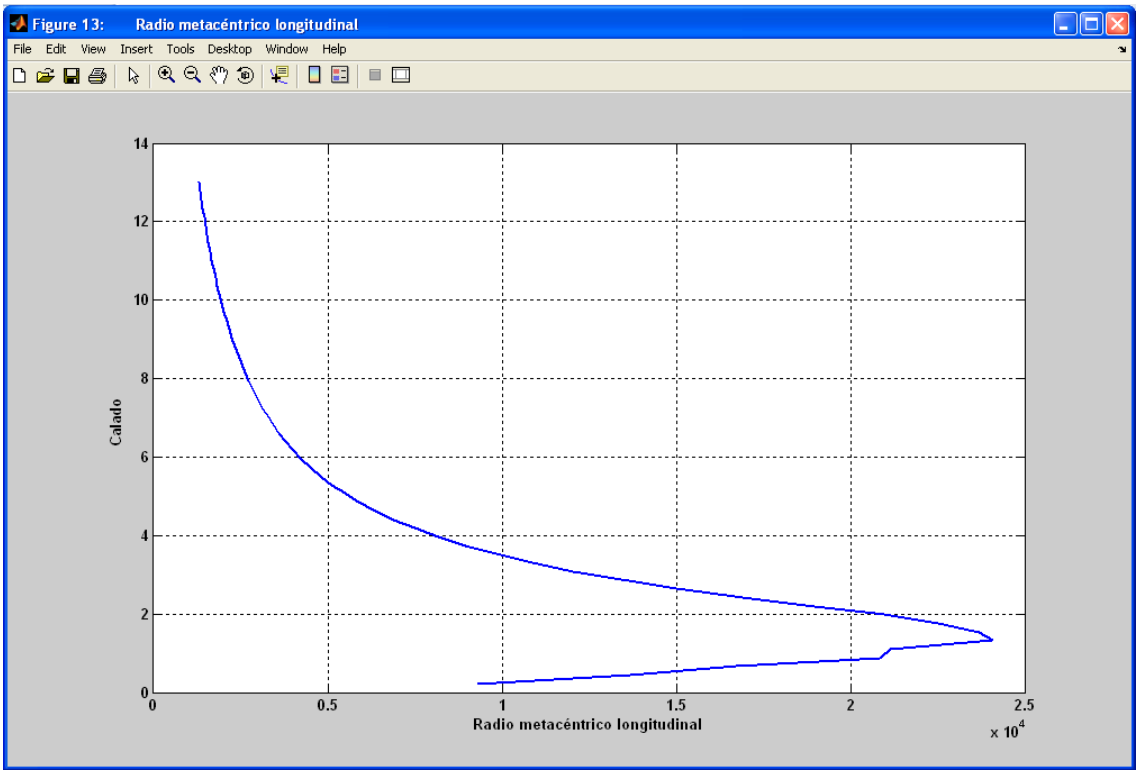

Imagen 63. Representación gráfica de la variación del radio metacéntrico longitudinal

7.3.4.4 Radio metacéntrico transversal

En esta imagen se muestra la variación del radio metacéntrico transversal, o $\overline{BM_t}$ con respecto a cada calado. Esta variación la muestra gráficamente. En la columna de la izquierda se muestran los diferentes calados, y en la línea inferior los datos del radio metacéntrico transversal.

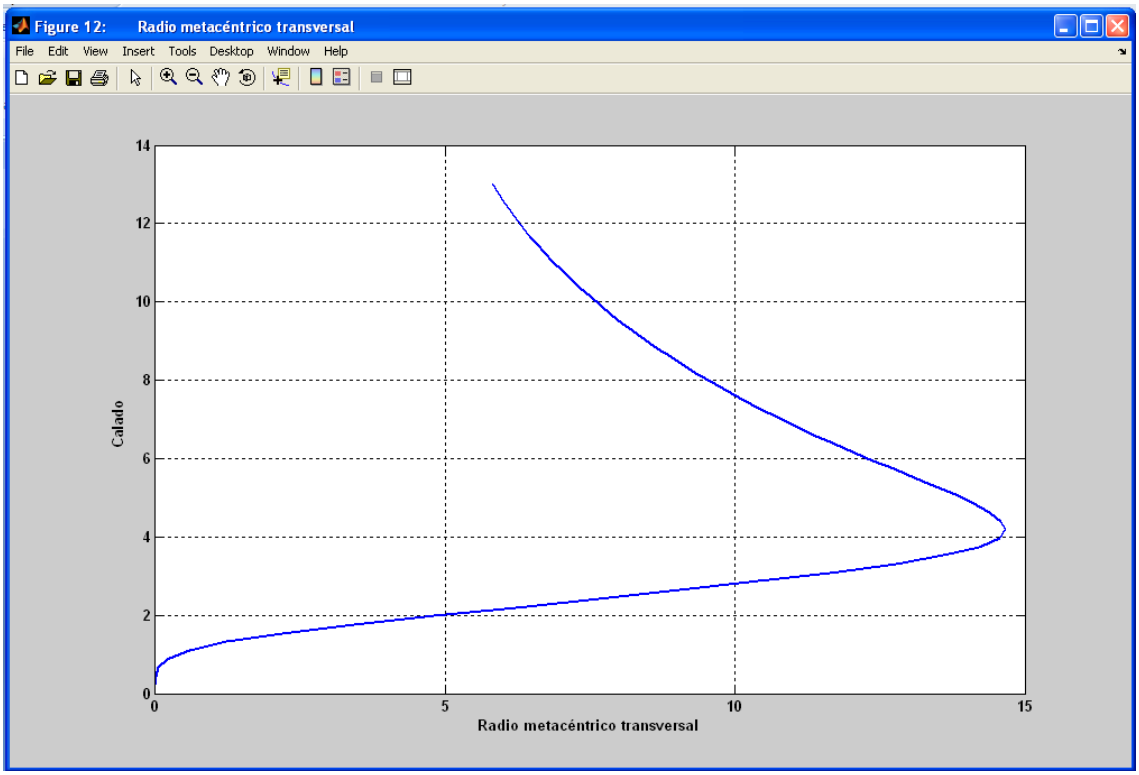

Imagen 64. Representación gráfica de la variación del radio metacéntrico transversal

7.3.4.5 Coordenada z del centro de carena

En esta imagen se muestra la variación de la coordenada Z del centro de carena, o $\overline{KB}$ con respecto a cada calado. Esta variación la muestra gráficamente. En la columna de la izquierda se muestran los diferentes calados, y en la línea inferior los datos de la coordenada vertical del centro de carena.

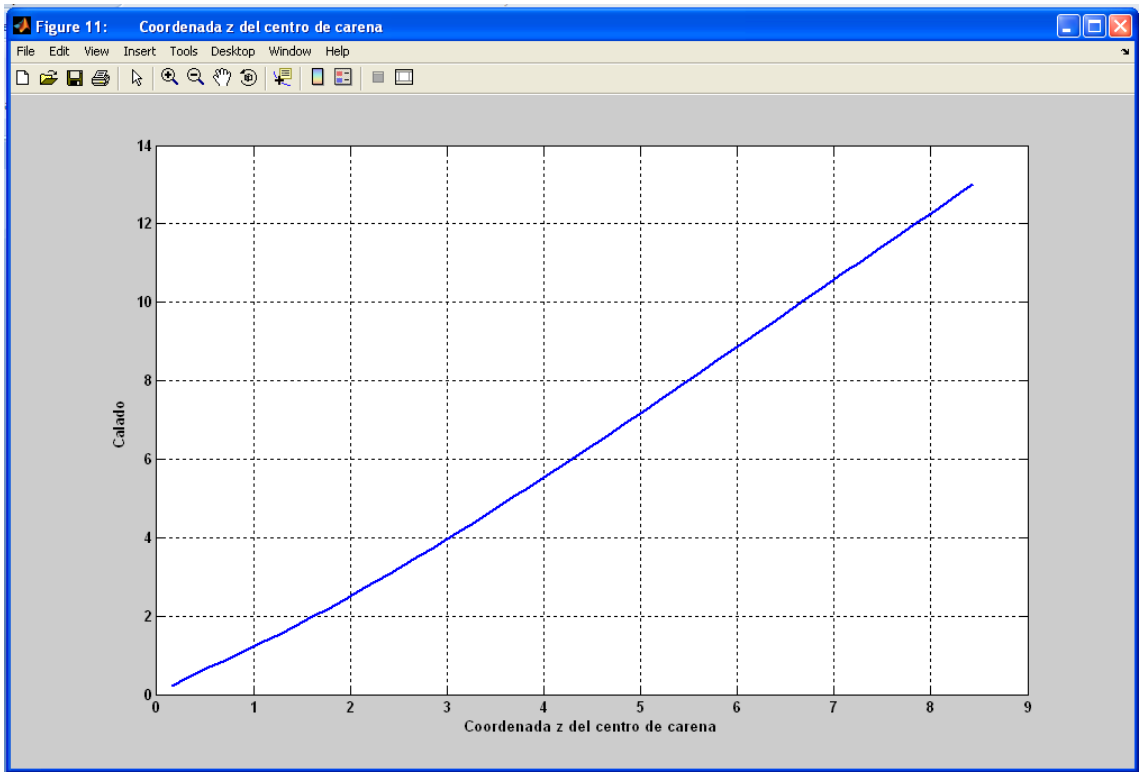

Imagen 65. Representación gráfica de la coordenada z del centro de carena

7.3.4.6 Momento de desplazamiento respecto de Z

En esta imagen se muestra la variación del radio momento de desplazamiento respecto de Z, o $\overline{MB}$ con respecto a cada calado. Esta variación la muestra gráficamente. En la columna de la izquierda se muestran los diferentes calados, y en la línea inferior los datos del momento de desplazamiento respecto de Z.

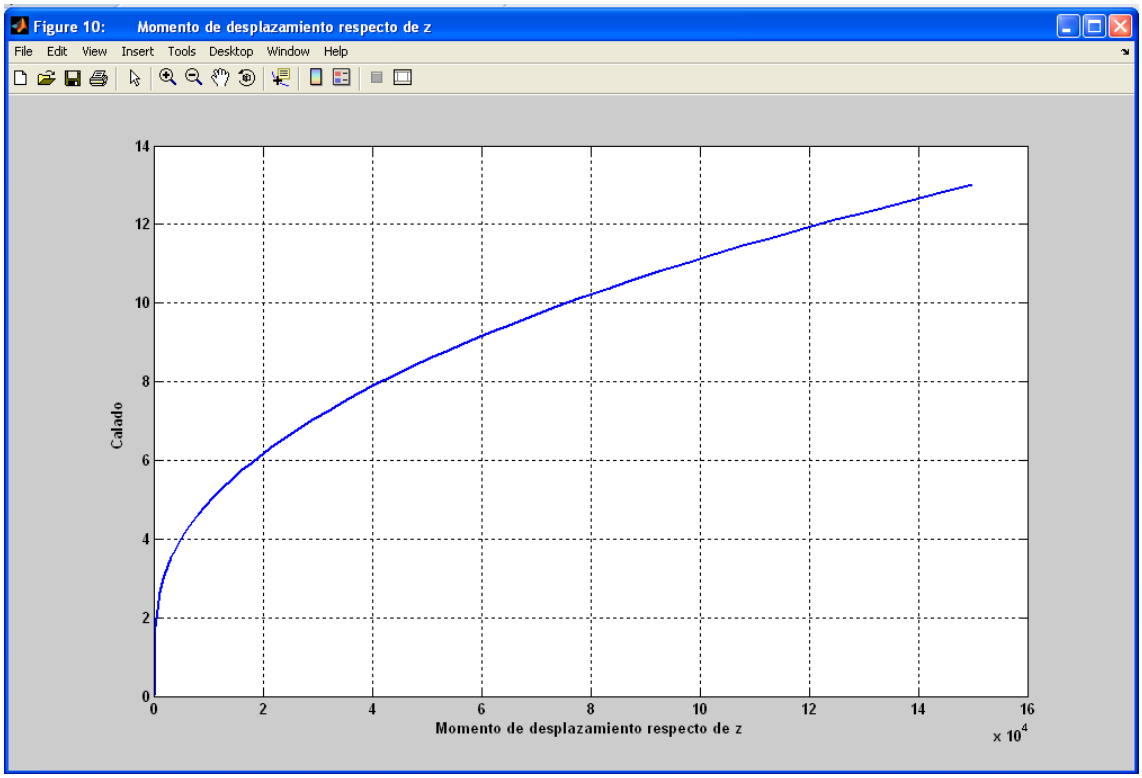

Imagen 66. Representación gráfica de la variación del momento de desplazamiento respecto de z

7.3.4.7 Volumen de desplazamiento

En esta imagen se muestra la variación del volumen de desplazamiento, o ∇ con respecto a cada calado. Esta variación la muestra gráficamente. En la columna de la izquierda se muestran los diferentes calados, y en la línea inferior los datos del volumen desplazamiento para cada calado.

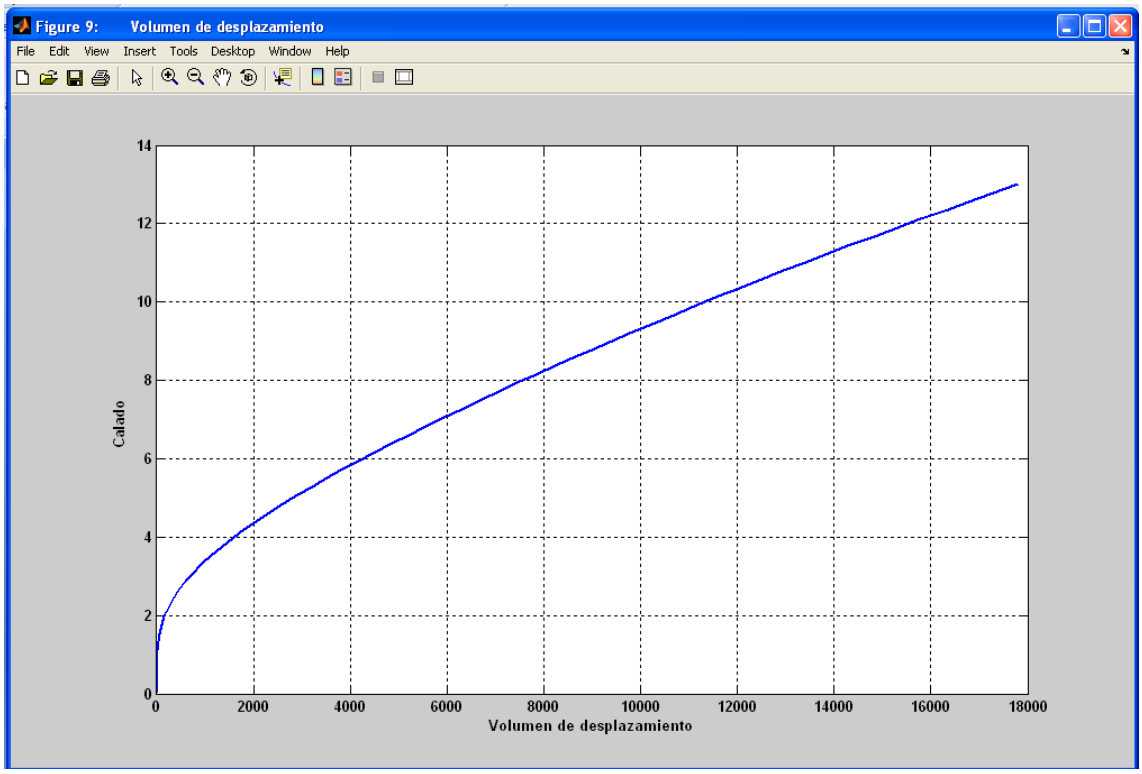

Imagen 67. Representación gráfica de la variación del volumen de desplazamiento

7.3.4.8 Toneladas por centímetro de inmersión

En esta imagen se muestra la variación de las toneladas por centímetro de inmersión, o TCI con respecto a cada calado. Esta variación la muestra gráficamente. En la columna de la izquierda se muestran los diferentes calados, y en la línea inferior los datos de las toneladas por centímetro de inmersión para cada calado.

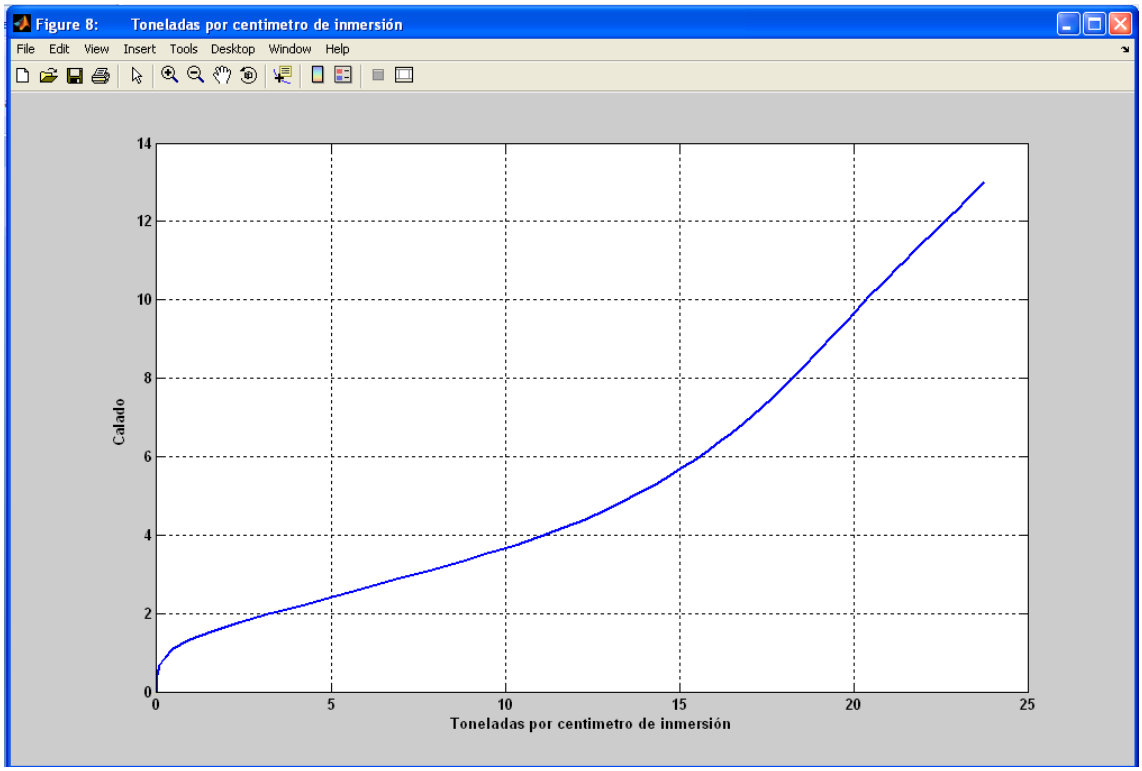

Imagen 68. Representación gráfica de la variación de las toneladas por centímetro de inmersión

7.3.4.9 Momento longitudinal de inercia

En esta imagen se muestra la variación del momento longitudinal de inercia, o I_l con respecto a cada calado. Esta variación la muestra gráficamente. En la columna de la izquierda se muestran los diferentes calados, y en la línea inferior los datos del momento de inercia longitudinal para cada calado.

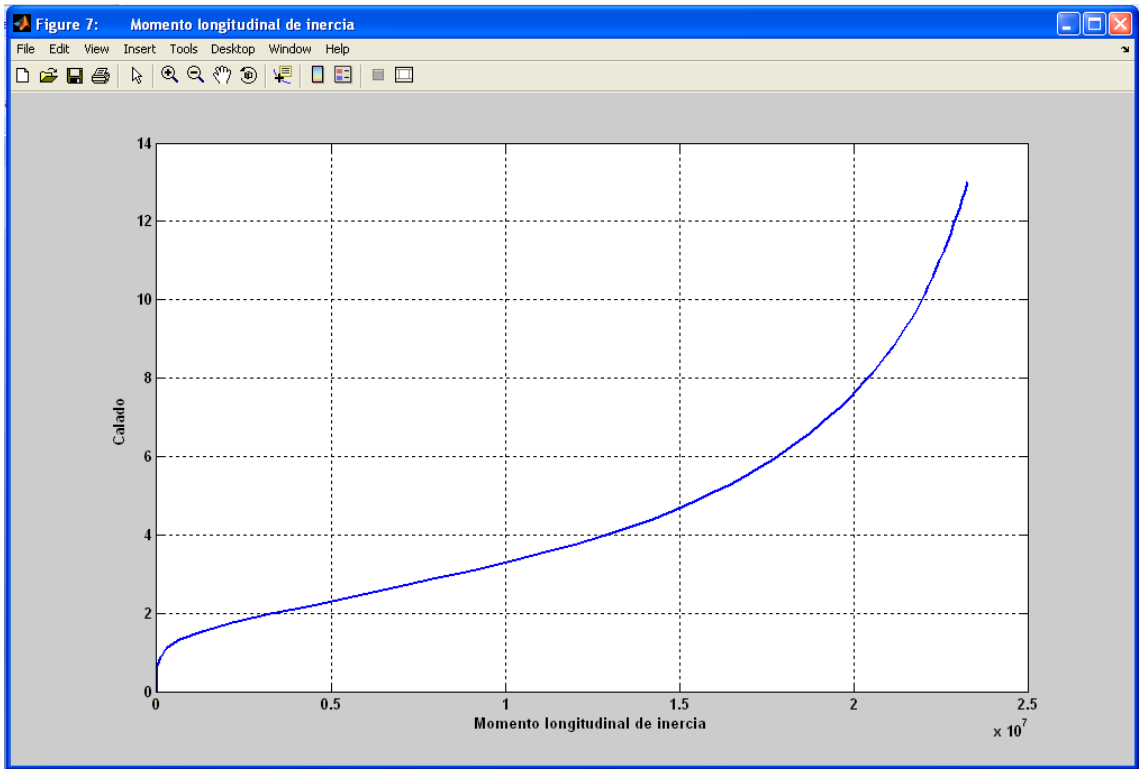

Imagen 69. Representación gráfica de la variación del momento longitudinal de inercia

7.3.4.10 Momento de inercia del plano de agua

En esta imagen se muestra la variación del momento de inercia del plano de agua, o I_l con respecto a cada calado. Esta variación la muestra gráficamente. En la columna de la izquierda se muestran los diferentes calados, y en la línea inferior los datos del momento de inercia del plano de agua para cada calado.

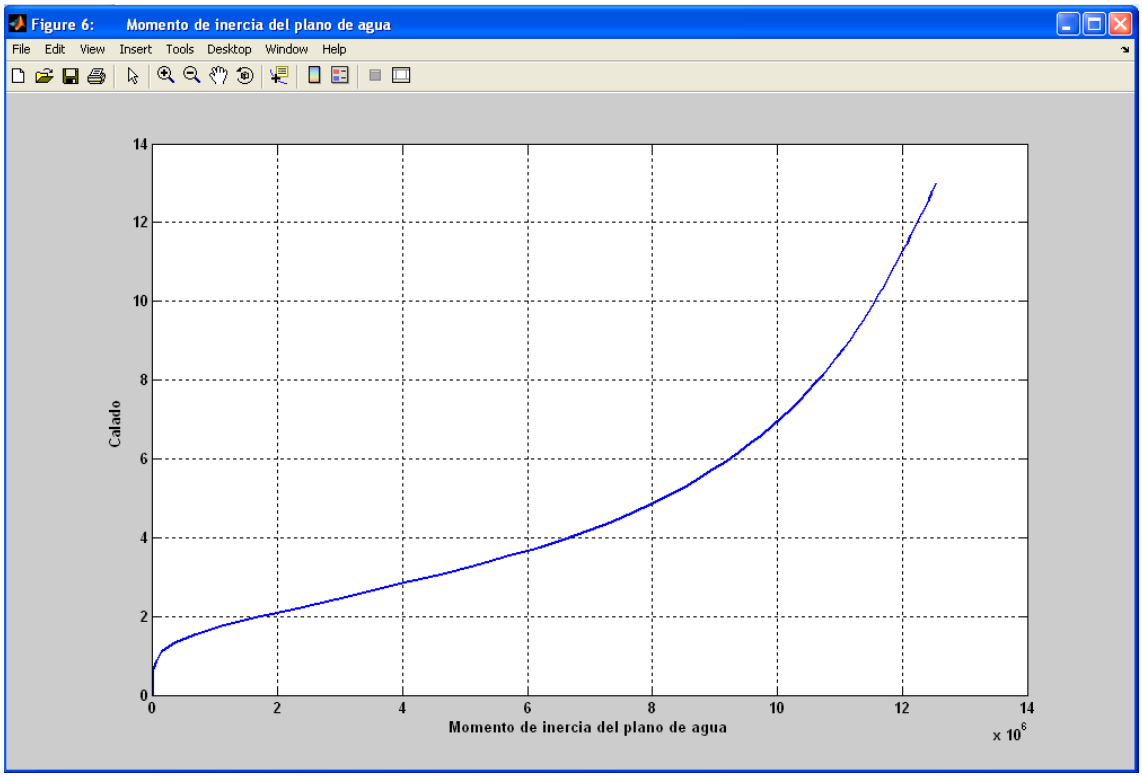

Imagen 70. Representación gráfica de la variación del momento de inercia del plano de agua

7.3.4.11 Momento de inercia transversal

En esta imagen se muestra la variación del momento de inercia transversal, o I_t con respecto a cada calado. Esta variación la muestra gráficamente. En la columna de la izquierda se muestran los diferentes calados, y en la línea inferior los datos del momento de inercia transversal para cada calado.

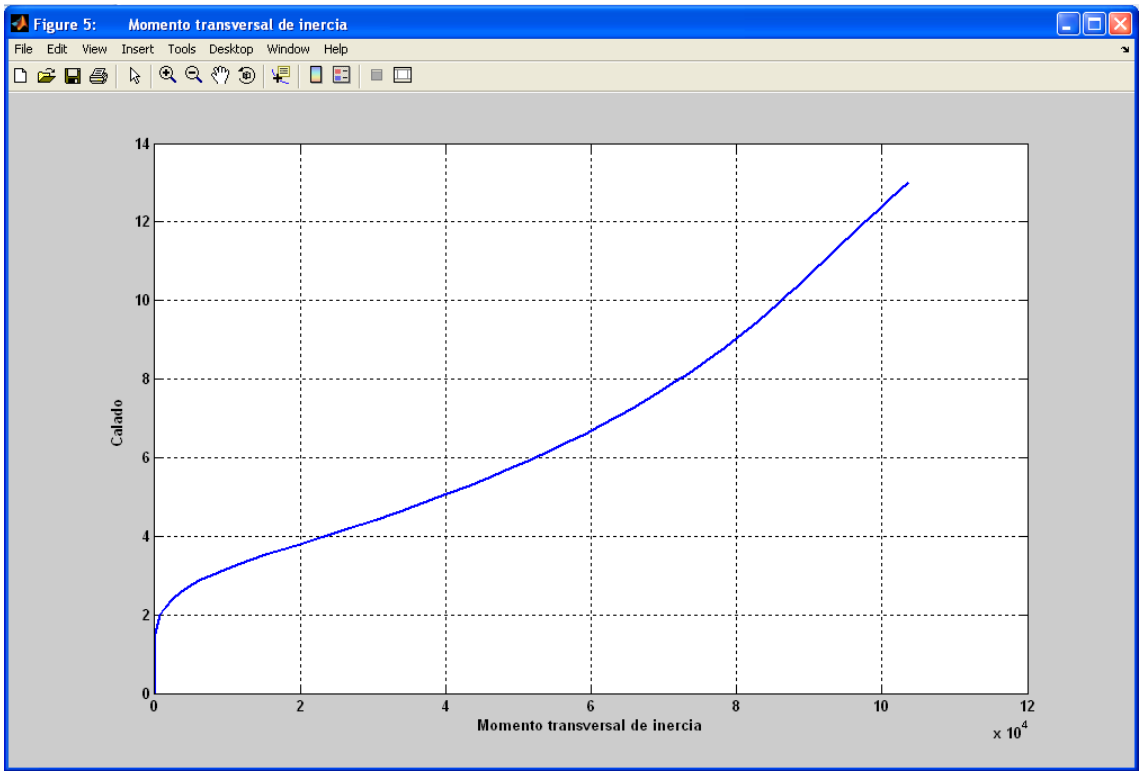

Imagen 71. Representación gráfica de la variación del momento transversal de inercia

7.3.4.12 Centro de flotación

En esta imagen se muestra la variación del centro de flotación longitudinal, o $\overline{XB}$ con respecto a cada calado. Esta variación la muestra gráficamente. En la columna de la izquierda se muestran los diferentes calados, y en la línea inferior los datos de la posición del centro de flotación longitudinal para cada calado.

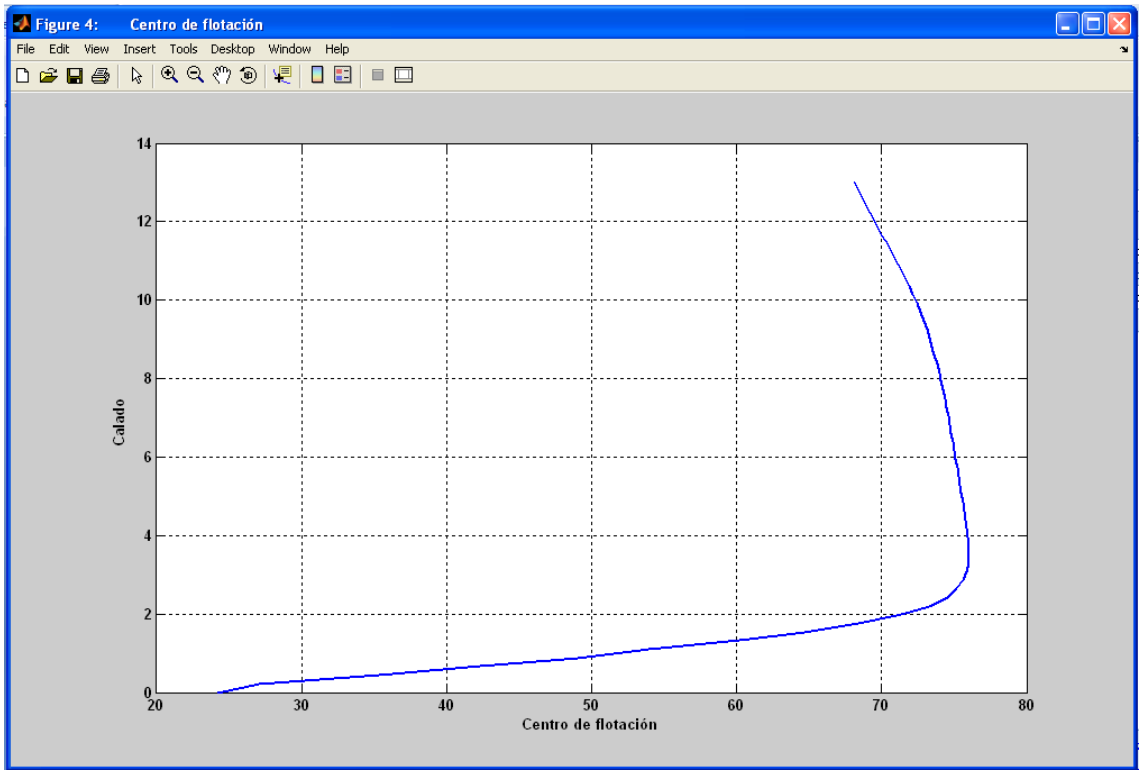

Imagen 72. Representación gráfica de la variación del dentro centro de flotación

7.3.4.13 Momento de área del plano de agua

En esta imagen se muestra la variación del momento de área del plano de agua, o $\overline{M_x}$ con respecto a cada calado. Esta variación la muestra gráficamente. En la columna de la izquierda se muestran los diferentes calados, y en la línea inferior los datos del momento de área del plano de agua para cada calado.

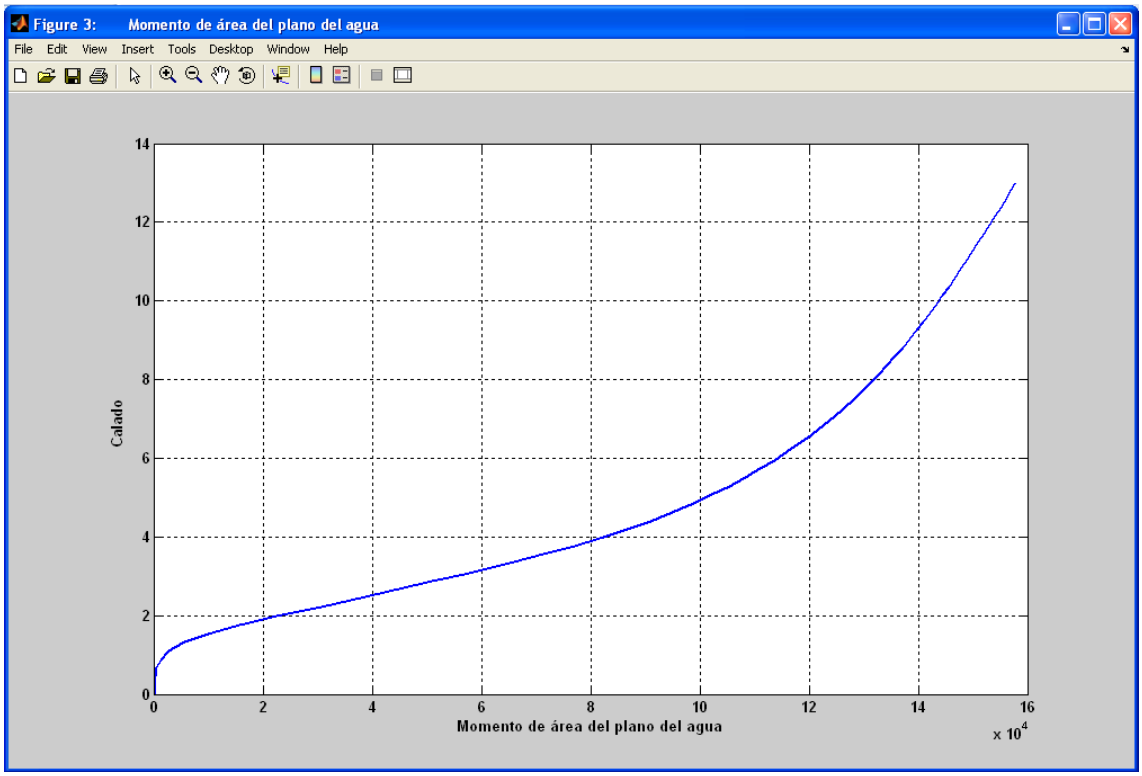

Imagen 73. Representación gráfica de la variación del momento del área del plano del agua

7.3.4.14 Área de la flotación

En esta imagen se muestra la variación del área de la flotación, o A_w con respecto a cada calado. Esta variación la muestra gráficamente. En la columna de la izquierda se muestran los diferentes calados, y en la línea inferior los datos del área de la flotación para cada calado.

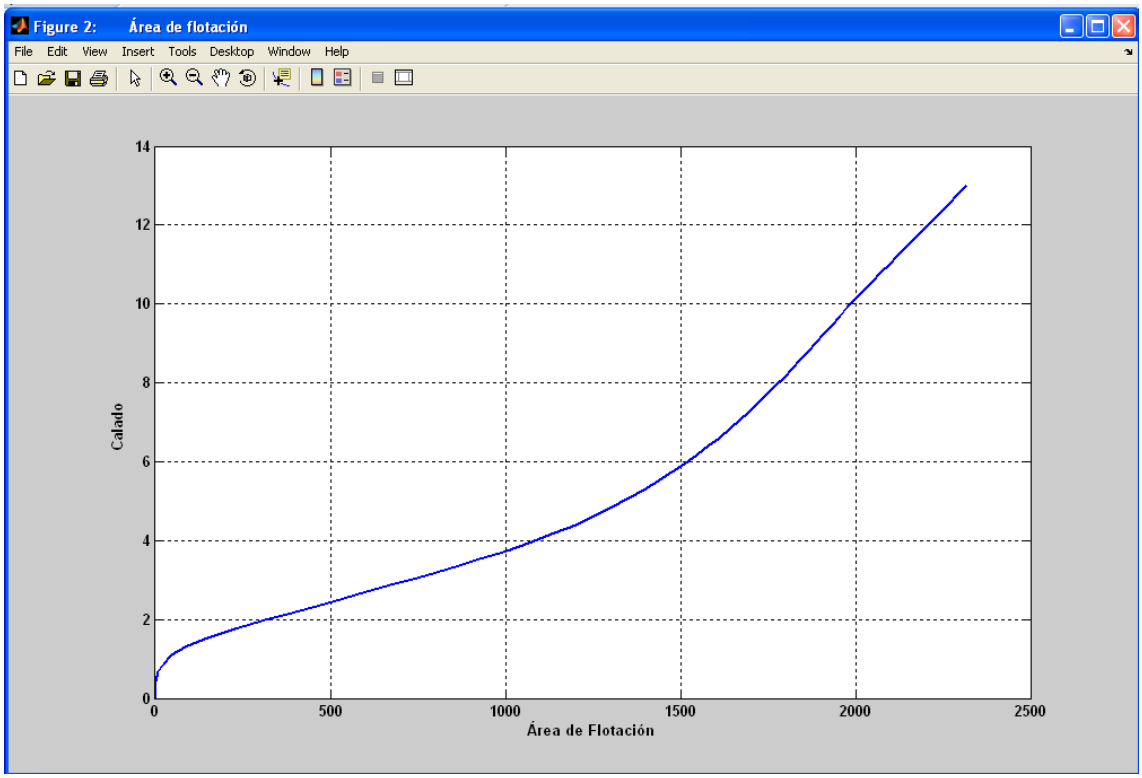

Imagen 74. Representación gráfica de la variación del área de la flotación

7.3.4.15 Área de las secciones

En esta imagen se muestra la variación del área de las secciones referenciadas a cada calado. Esta variación la muestra gráficamente. En la columna de la izquierda se muestran los diferentes valores del área de las secciones, y en la línea inferior los datos de las secciones para cada calado.

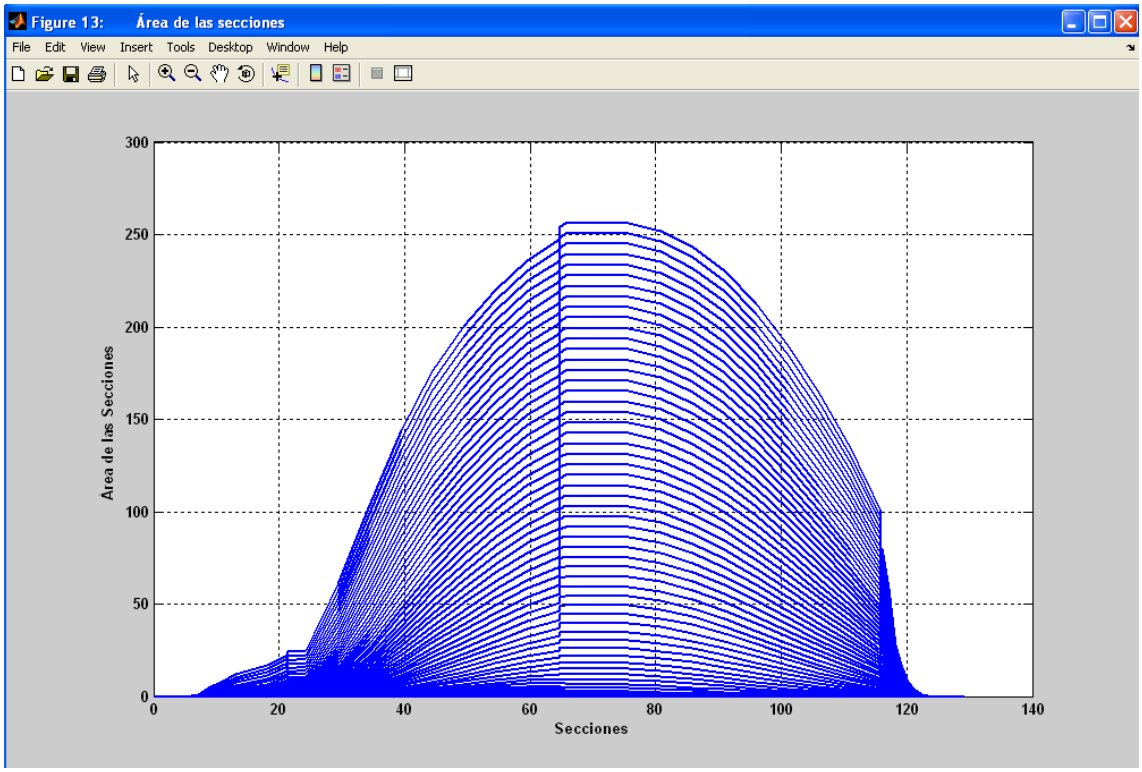

Imagen 75.. Representación gráfica de la variación de las secciones

Además de todo esto, se generará un archivo .xls con todos los datos hidrostáticos que se han calculado para poder utilizarlo en el programa “Proceso de la Botadura”.

Con todos estos datos, se ha terminado la obtención y representación gráfica de todos los valores que se pueden obtener del programa en este apartado.

7.4 Cómo usar el apartado "Proceso de la Botadura"

A partir del inicio del programa, donde se encuentra la siguiente imagen se presiona el botón "Proceso de la Botadura". Al presionarlo aparecerá la siguiente pantalla:

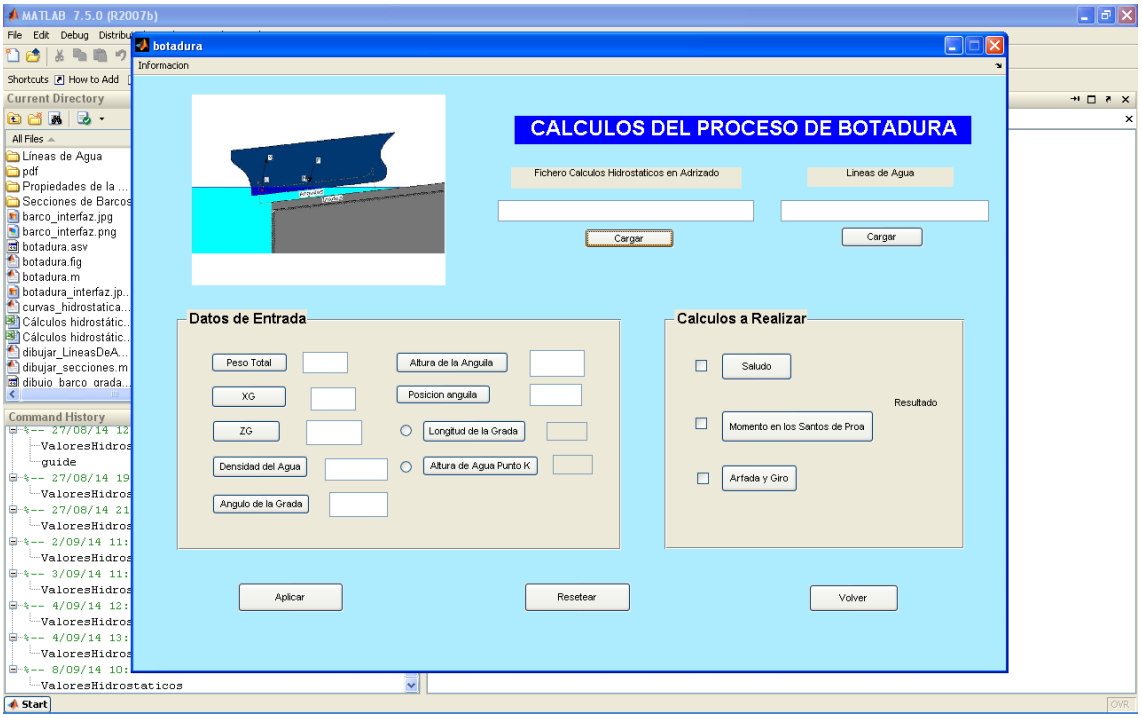

Imagen 76. Pantalla del programa botadura

Para comenzar a utilizar este apartado, se deben tener los datos de entrada que necesita el programa. Esos datos de entrada son:

- **Fichero Cálculos Hidrostáticos en Adrizado:** En este apartado se ha de incluir el fichero .xls que se obtiene a partir del cálculo de todos los datos del barco en el apartado "Valores Hidrostáticos en Adrizado" de este programa.
- **Líneas de agua:** Se han de incluir los valores de las líneas de agua en formato .xls como en el programa "Valores Hidrostáticos en Adrizado". Este archivo también se ha de haber obtenido previamente.
- **Peso total:** El valor del peso total a incluir, es la suma de los pesos de: barco en el momento de la botadura, pesos ajenos al barco que están incluidos en él en el momento de la botadura, etc.

- $\overline{XG}$**, posición longitudinal del centro de gravedad:** El valor de la posición del centro de gravedad es de gran importancia, por ello, se ha de incluir tanto la posición horizontal del centro de gravedad como la posición vertical del mismo. En este caso se debe incluir la distancia horizontal desde el punto más a popa del barco hasta la posición del centro de gravedad.
- $\overline{ZG}$**, posición vertical del centro de gravedad:** En este caso, se debe indicar la posición vertical del centro de gravedad del conjunto de pesos que componen el barco en el momento de la botadura. Se tomará la distancia vertical desde la posición más baja del barco hasta el centro de gravedad.
- **Densidad del agua:** El valor de la densidad del agua ha de ser introducida también, ya que el proceso sería diferente de ser en un tipo de agua que en otra. El valor normal del agua de río es 1,000 kg/dm^3 mientras que el valor habitual del agua de mar es de 1,025 kg/dm^3 exceptuando casos peculiares.
- **Ángulo de la grada:** La grada tiene una inclinación suficiente para que el barco caiga por su propio peso una vez se haya soltado de las "llaves" que le sujetan y le impiden el movimiento, pero no mucha inclinación para que no acelere demasiado y haya problemas en el momento de la botadura. El valor de este dato se incluirá en grados.
- **Altura de la anguila:** Como está explicado en el tutorial, esta altura indica la distancia perpendicular a la posición horizontal del barco desde la base del barco a la base de la anguila.
- **Posición de la anguila:** Se ha de indicar la posición en la que se sitúa el barco en referencia a la anguila para conocer el momento en el cual el barco dejará de estar apoyado en la anguila y debe de flotar por sí solo.
- **Longitud de la grada:** Es indiferente indicar esta distancia o la altura del agua en el punto K, a partir de ambas se pueden obtener los resultados deseados, ya que una depende de la otra. Esta distancia se toma en dirección paralela a la pendiente como se indica en el tutorial para ello.
- **Altura de agua punto K:** Se trata de la distancia vertical desde el nivel del mar en el momento de la botadura hasta el punto K de la grada.

Si se tiene dudas al incluir cualquiera de los datos anteriores, cada uno de los datos incluidos en el programa viene acompañado de un tutorial. Para ver el tutorial, se ha de presionar el nombre de la opción que sugiere dudas y se abrirá un PDF explicativo donde se aclara el término que incluye.

Tras incluir todos los datos anteriores en el programa, se debe indicar que procesos ha de resolver el programa. Para ello hay que seleccionar entre las opciones del recuadro “Cálculos a realizar” que aparece también en el programa:

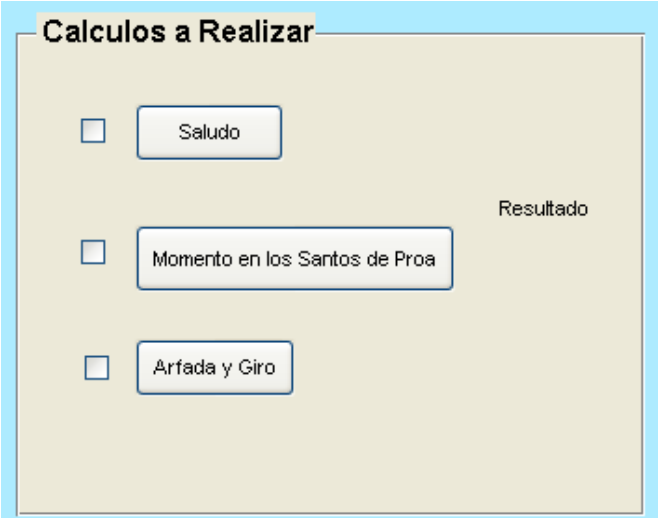

Imagen 77. Zona de selección de los "Cálculos a realizar" por el programa "Botadura"

Entre las opciones que existen para el cálculo, se incluyen:

- **Saludo:** En el momento de la botadura, la posibilidad de saludo es alta si el cálculo de pesos y la altura del agua en el momento del lanzamiento no es la adecuada. Si se produce este efecto indeseado, el barco golpearía con la proa en la grada y sufriría desperfectos en la estructura.
- **Momento en los Santos de Proa:** Esta fuerza es interesante de conocer para saber si la estructura de los santos de proa está bien diseñada y soporta el momento que genera el barco en el momento del giro y no se dañará el barco en el momento de la botadura. El resultado aparecerá a la derecha de la selección.
- **Arfada y Giro:** Si se selecciona esta opción, se podrá también elegir que la dibuje como aparece en la imagen 78. Si se da la arfada, el barco sufre riesgo de desperfectos en el momento de volver a la posición normal de la botadura. También indica el momento en el cual se realiza el giro si todo el proceso se realiza correctamente.

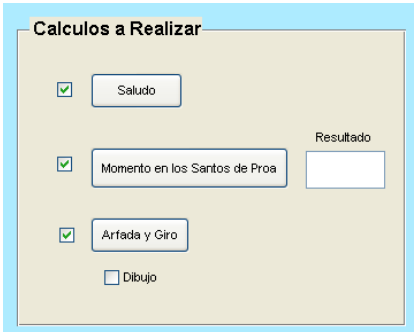

Imagen 78. Mostrado de la opción "Dibujo" y del recuadro de resultados de "Momento en los Santos de Proa"

Una vez se hayan incluido todos los datos de entrada y se haya seleccionado los cálculos a realizar, le daremos al botón “Aplicar” y el programa realizará los cálculos. Si se produce el efecto indeseado de saludo, mostrará un error en la pantalla. Si no lo hay, seguirá con el proceso del cálculo y mostrará el resultado del Momento en los Santos de Proa. Más tarde indicará la posición donde se inicia el giro, si no sufre arfada durante el proceso. Todo este proceso, se puede mostrar gráficamente para ayudar intuitivamente a quien realice el proceso de cálculo si señalamos la opción “Dibujo”. En ese caso mostrará imágenes del proceso de la botadura como las que se muestran a continuación.

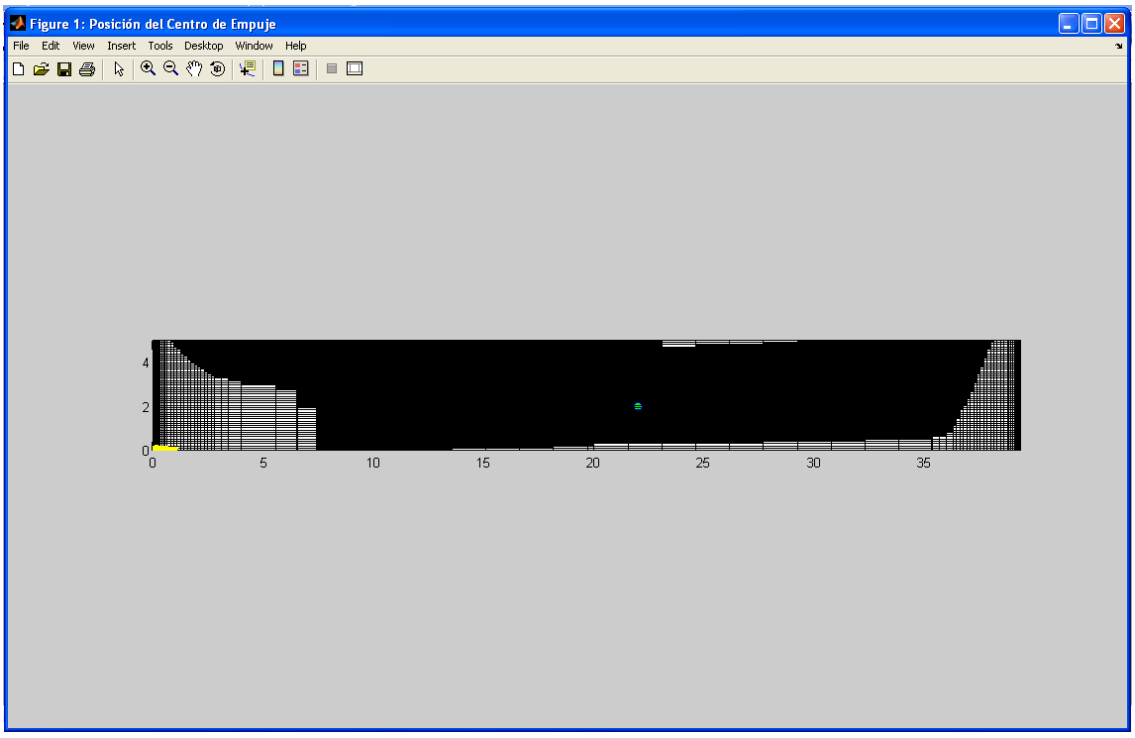

Imagen 79. Primeras imágenes del "Dibujo" del proceso de cálculo

Esta es una de las imágenes iniciales de un proceso de cálculo de botadura. En él se muestra el barco en color negro, el fondo en blanco, el centro de pesos o centro de gravedad de los pesos, en verde dentro de la figura del barco, y la línea de agua en amarillo.

El proceso se dibuja con el barco en horizontal y el agua inclinada, poco a poco irá introduciéndose el barco en el agua, aunque la sensación es que el agua "moja" al barco poco a poco.

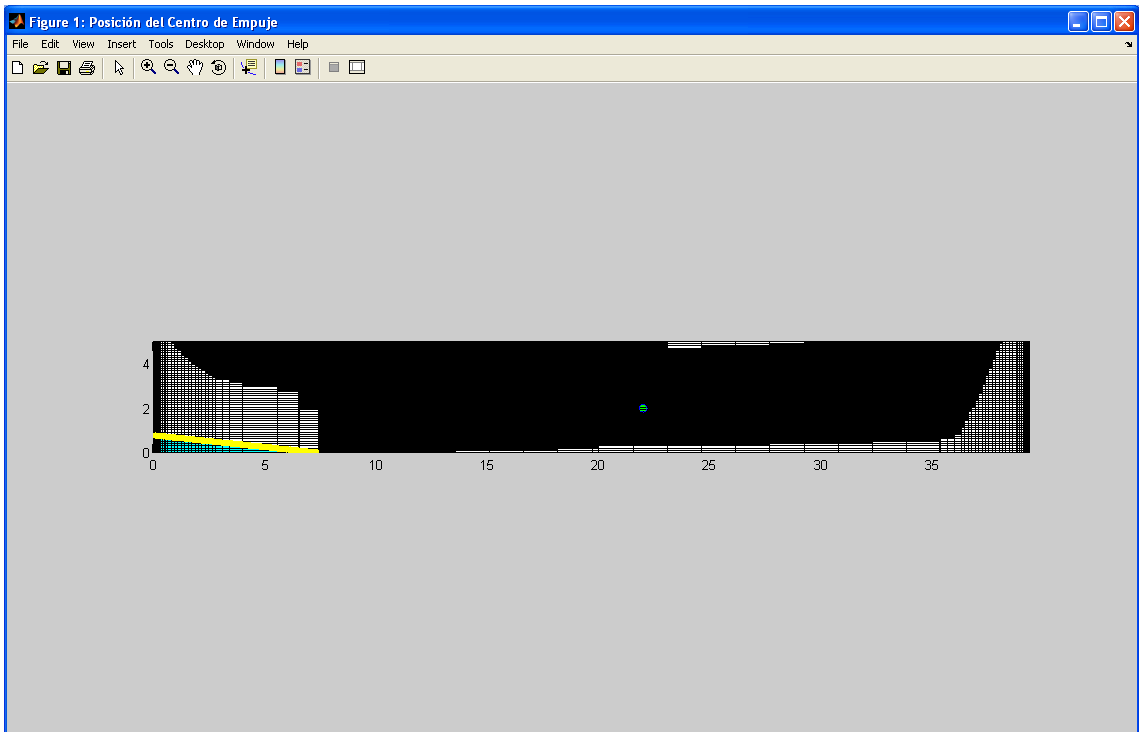

Imagen 80. Representación del agua antes de comenzar a sumergirse el barco

En la imagen 80, se ve ya parte del agua en la cual el barco va a realizar la botadura, el color del agua se representa con un color azul claro. Sobre ese azul sigue la línea de agua. Los colores de barco (negro) y aire (blanco) siguen siendo los mismos durante todo el proceso, al igual que la posición del centro de gravedad de todos los pesos.

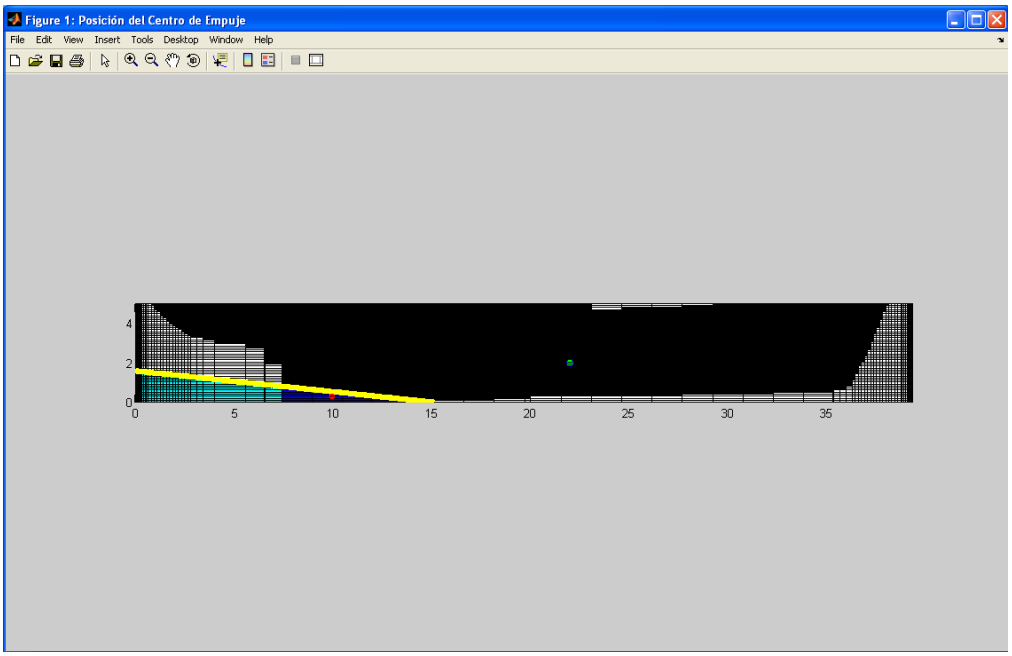

Imagen 81. Representación del momento en el que el barco comienza a sumergirse

En la imagen 81, se muestra ya parte del barco sumergido. La parte del barco que se haya sumergido se representará en color azul marino. Otro dato visual que nos representa es la posición del centro de carena, o posición del centro de gravedad del agua desalojada. Este dato es interesante durante el proceso. Además irá modificándose durante todo el proceso.

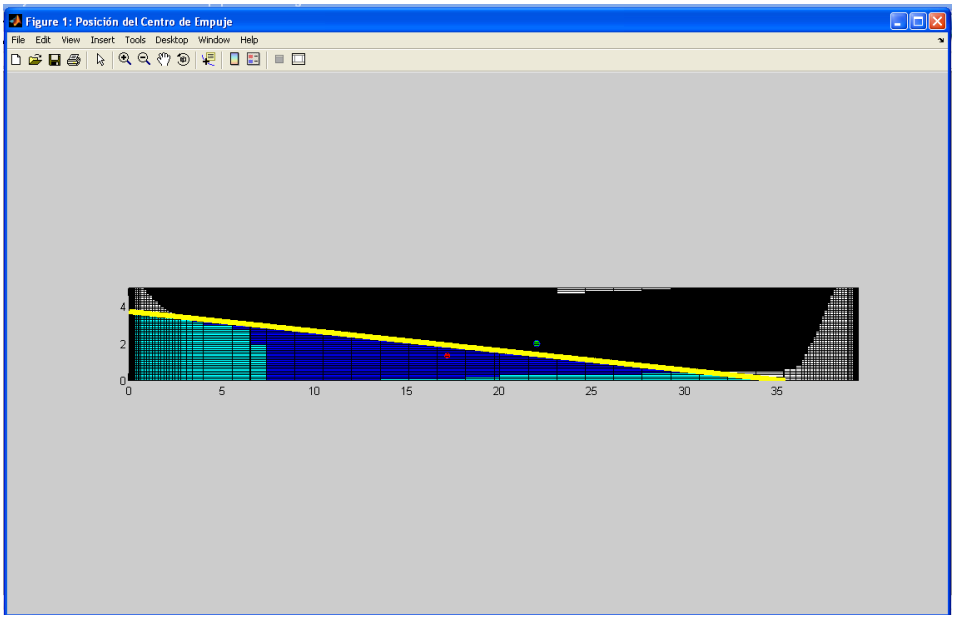

Imagen 82. Posición final de la botadura antes del giro del barco

La imagen 82, ha sido la posición final del cálculo que se ha usado como ejemplo. En él se muestra la parte que el barco ha sumergido antes de comenzar con el giro, los centros de gravedad del barco (en verde) y el centro de gravedad del agua desalojada, o centro de carena (en rojo). El programa finaliza de mostrar imágenes cuando el barco comienza a realizar el giro. En ese momento, nos muestra el dato del giro y cierra la pantalla de dibujo.

Durante el proceso, se muestra el resultado del Momento en los Santos de Proa como se muestra en la siguiente imagen:

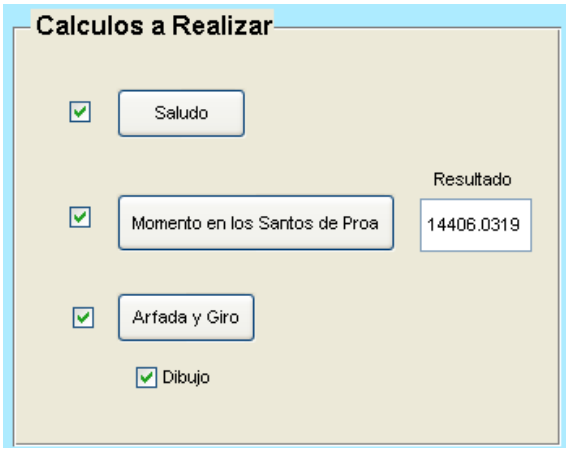

Imagen 83. Resultado del Momento en los Santos de Proa

Si no surgen errores, el programa finaliza dando el dato de la posición donde el barco comienza el giro.

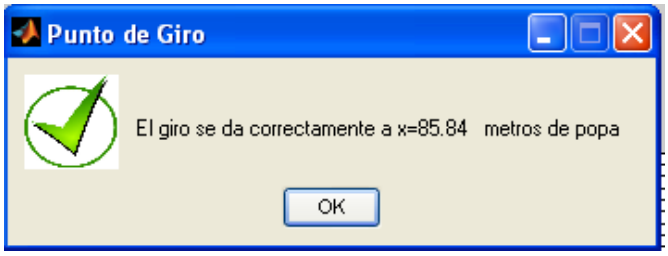

Imagen 84. Punto de Giro

El punto de giro, está referenciado al punto más a popa del barco.

De esta manera ya se ha terminado con el proceso de cálculo y se puede proceder a la botadura o construcción del barco sabiendo que es posible la botadura del barco.

8. Tutoriales de ayuda de la interfaz gráfica

Los tutoriales de esta sección, incluyen los mismos textos e imágenes que los tutoriales que hay incluidos en el programa. Se pueden ver directamente desde el programa seleccionándolos uno a uno en el título que muestra cada uno. Tras su selección, se ejecutará directamente la ayuda con la definición.

Todos ellos, están explicados de manera breve y concisa, incluyendo en la mayoría de los casos una imagen que ayuda a la comprensión del término que está explicando.

8.1 Peso total

El peso total, es la suma de todos los pesos que se incluyen en el momento de la botadura. Para ello hemos de sumar tanto el peso del casco del barco, motores, accesorios del barco y todos los pesos que estén a bordo en el momento de la botadura.

El valor a incluir debe ser en toneladas (tn).

8.2 $\overline{XG}$

El $\overline{XG}$, es la coordenada longitudinal del centro de gravedad de los pesos. Este dato se indicará referenciándolo con el origen de referencia, que en el caso del programa, es el punto más a popa del barco.

Este dato se obtiene a partir de los centros de gravedad de todos los pesos que están incluidos en el barco. La fórmula para obtener el $\overline{XG}$ total es:

$$\overline{XG} = \frac{\sum(\overline{xg}_i \cdot p_i)}{Peso\ total}$$

Siendo:

- $\overline{xg}_i$ el centro longitudinal de gravedad de cada peso,
- p_i cada uno de los pesos considerados
- Peso total, la suma de todos los pesos a bordo del barco en el momento de la botadura.

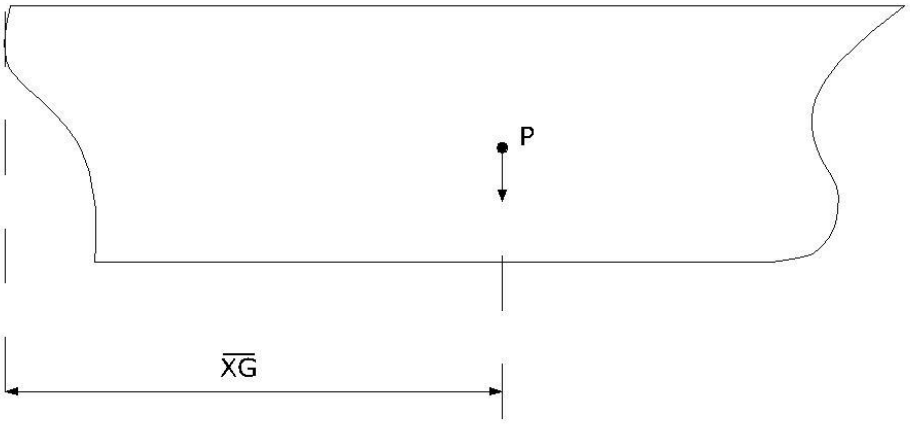

Imagen 85. XG

8.3 $\overline{ZG}$

El $\overline{ZG}$, es la coordenada vertical del centro de gravedad de los pesos. Este dato se indicará referenciándolo con la base del barco, que en el caso del programa, es la quilla del barco.

Este dato se obtiene a partir de los centros de gravedad de todos los pesos que están incluidos en el barco. La fórmula para obtener el $\overline{ZG}$ total es:

$$\overline{ZG} = \frac{\sum(\overline{zg}_i \cdot p_i)}{Peso\ total}$$

Siendo:

- $\overline{zg}_i$ el centro vertical de gravedad de cada peso
- P_i es el peso de cada uno de los considerados
- Peso total, la suma de todos los pesos a bordo del barco en el momento de la botadura

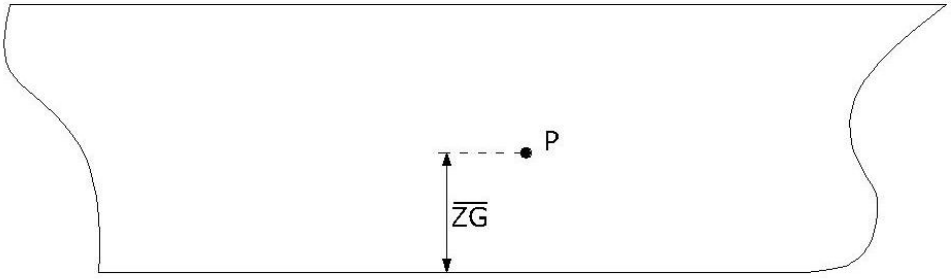

Imagen 86. ZG

8.4 Densidad

Es la magnitud del líquido referida a la cantidad de masa en un determinado volumen de sustancia.

La densidad es la razón entre la masa de un cuerpo y el volumen que ocupa, su fórmula es la siguiente:

$$\rho = \frac{m}{V}.$$

Siendo:

- ρ es la densidad
- m es la masa de la sustancia en un volumen determinado
- V es el volumen de medición de la masa indicada anteriormente
- Los valores de la densidad del agua son:
- Agua salada: 1,025 t/m^3
- Agua dulce: 1,000 t/m^3

8.5 Ángulo de la grada

El ángulo de la grada (θ) es la inclinación de la grada con respecto al plano del agua en grados. Este ángulo suele estar entre los 3 y 8 grados de inclinación.

El dato a introducir ha de ser en grados, no en radianes.

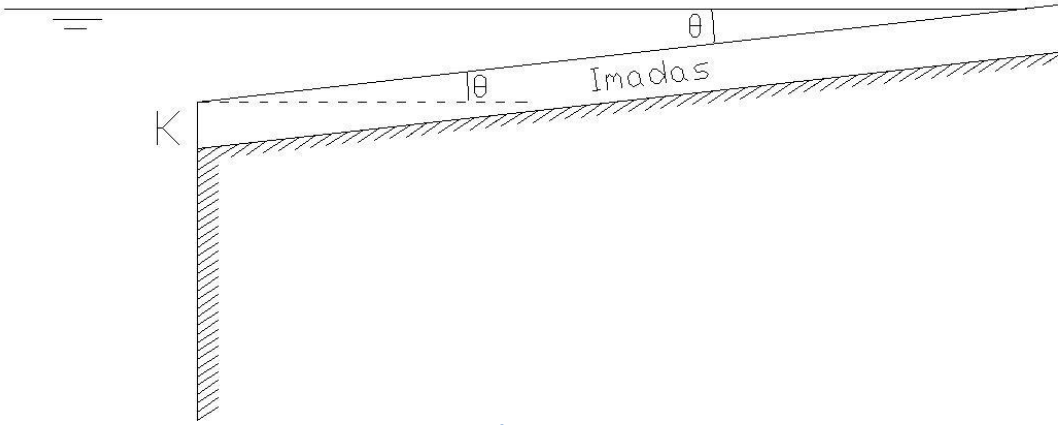

Imagen 87. Ángulo de la grada

8.6 Altura de la anguila

La altura de la anguila es la distancia vertical medida desde la base de la anguila hasta la quilla del barco en la zona más a proa de la anguila. Esta altura es un dato imprescindible, ya que eleva el barco y necesitaremos mayor recorrido de grada para que el barco llegue al calado necesario.

La altura de la anguila está definida en la siguiente imagen como $h_{an.}$

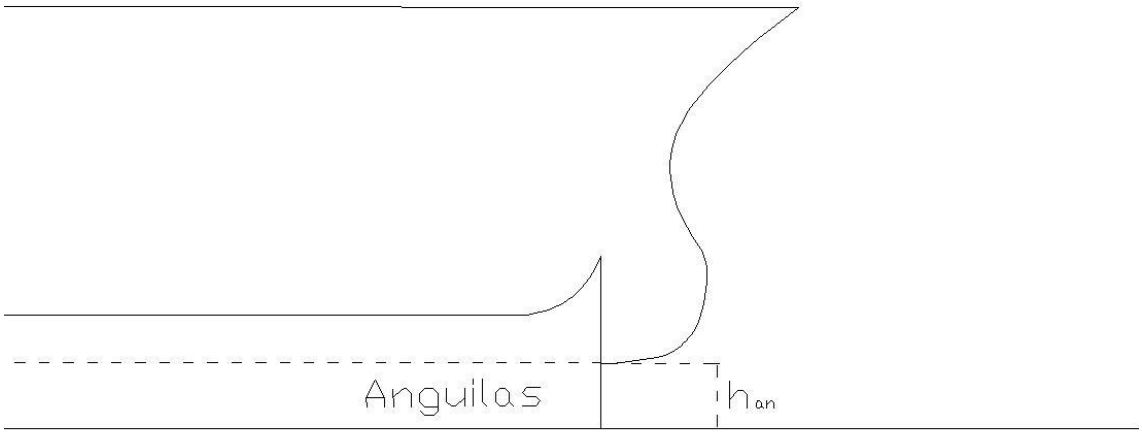

Imagen 88. Altura de la anguila

8.7 Longitud de la grada

En este apartado, hemos de introducir la distancia de grada mojada, medida en la posición inclinada, como se indica en la siguiente imagen con la notación d_{grada}.

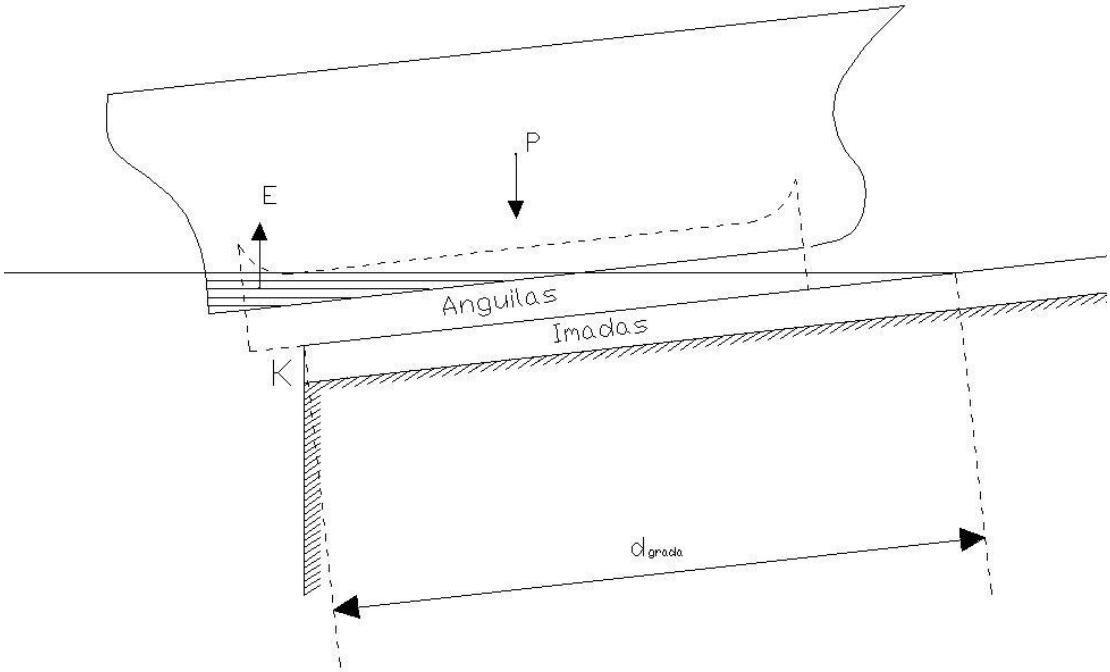

Imagen 89. Longitud de la grada

Esta distancia varía con la marea, por ello hay que tener en cuenta el valor de la misma en el momento de la botadura.

8.8 Distancia de la proa de la anguila a la proa del barco

La posición de la anguila, es la distancia horizontal desde el punto más a proa de la anguila hasta la proa del barco. Este dato lo usará el programa para saber el momento en el que el barco dejaría de estar apoyado en la grada por la anguila y flota libremente, o golpea con la grada sufriendo el indeseado efecto de la arfada.

El dato a introducir en el programa, es la distancia indicada como d_{an} en la siguiente imagen:

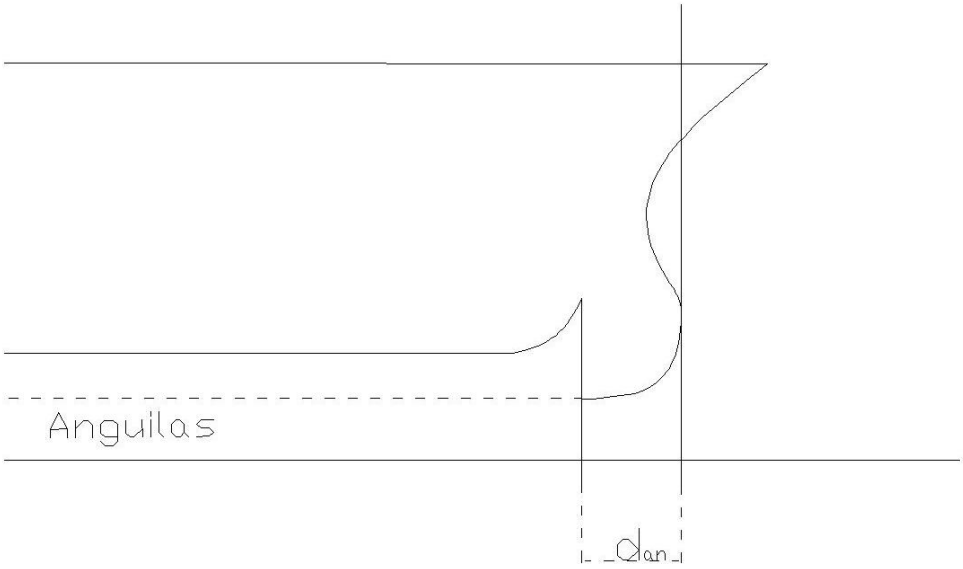

Imagen 90. Distancia desde la proa de la anguila a la proa del barco

8.9 Altura en el punto K

La altura en el punto K (T_K), es la distancia del punto K de la grada, hasta el nivel del agua en la que el barco va a ser botado.

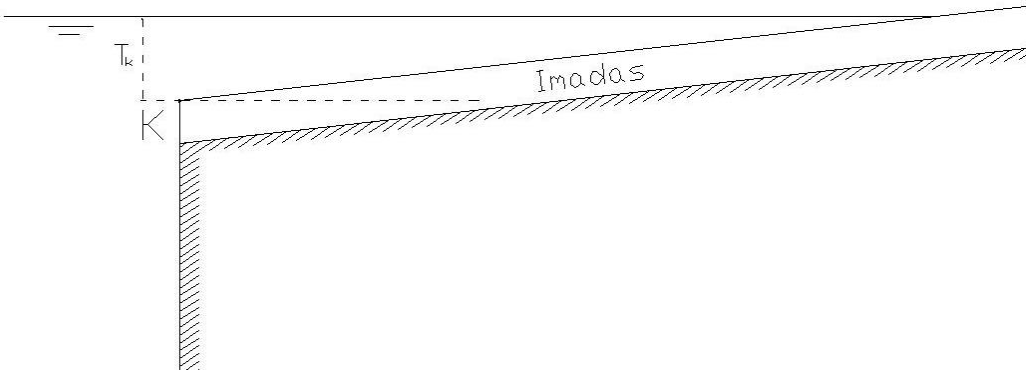

Imagen 91. Altura en el punto K

8.10 Saludo

Este fenómeno es un movimiento indeseado que realiza el barco al finalizar el trayecto de las anguilas sobre la grada. En él, lo que hemos de tener en cuenta es que el barco ha de flotar libremente antes de llegar la proa de las anguilas al final de la grada, si no ocurre esto podemos tener el problema de la falta de calado en proa y por lo tanto, cuando el barco finalice el trayecto en la grada, sumergirá la proa, pudiendo golpear indeseadamente ésta con la grada ocasionando graves desperfectos a la estructura de proa y a la grada.

El suceso es el que aparece en la siguiente imagen:

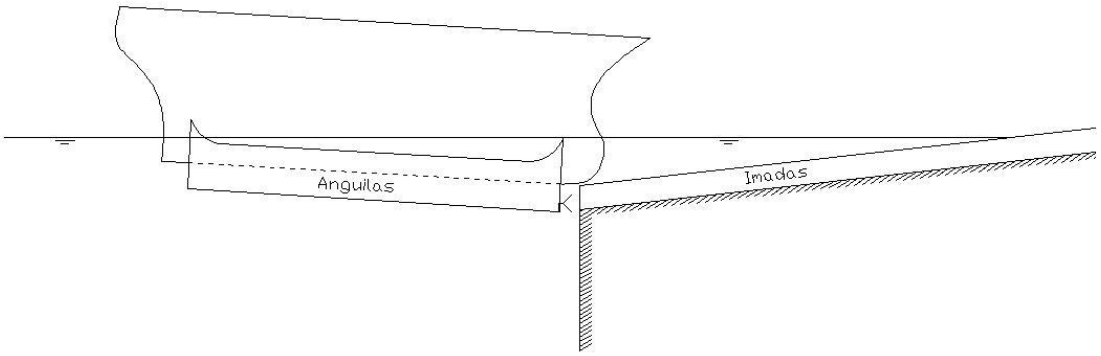

Imagen 92. Saludo

8.11 Momento en los santos de proa

Este momento es el que sufren los santos de proa (zona más a proa de las anguilas) cuando recae sobre ellas todo el peso del barco en el momento del giro.

La fuerza que soporta esta zona de las anguilas, es de gran interés, ya que, de no resistir toda la fuerza que recibe en ese momento, romperían. Si esto ocurriese, el barco golpearía la proa del barco directamente sobre la grada, generando graves desperfectos en la estructura del mismo.

El momento crítico es el que aparece en la imagen siguiente y el punto más perjudicado, es el que está coloreado de rojo:

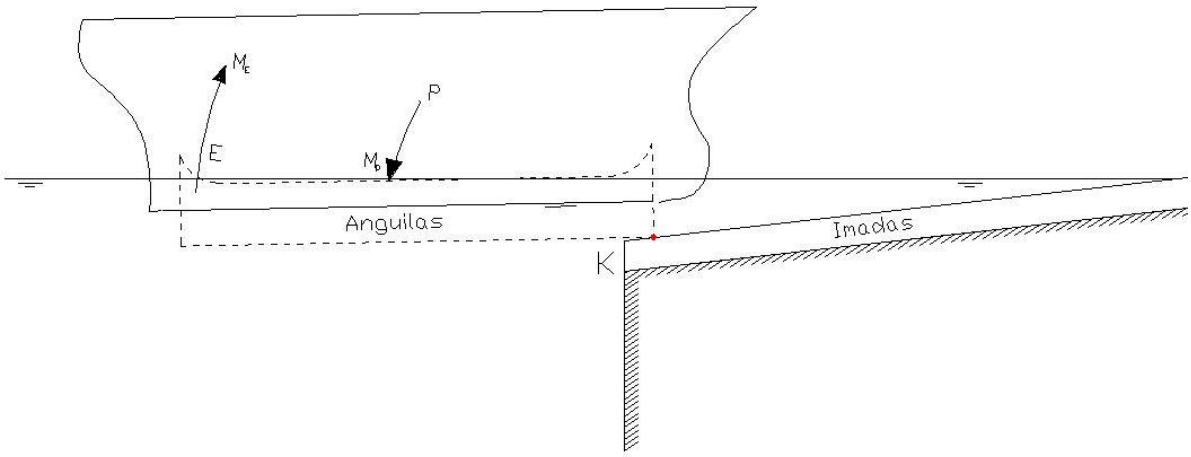

Imagen 93. Momento en los santos de proa

8.12 Arfada

El movimiento de arfada, es un giro indeseado que se produce durante el proceso de la botadura si no se han calculado bien los momentos que se van a generar durante el recorrido del barco al agua.

Este fenómeno, sucede cuando el momento del peso que se genera desde el punto K, es mayor que el momento empuje que está generando el fluido desalojado por el barco. Esto hace que el barco levante la proa, para más tarde, recuperarse y golpear bruscamente la grada sufriendo desperfectos, o inundarse directamente. Este fenómeno solo aparece desde el momento en el que el punto P no se encuentra sobre la grada.

El giro que se produce es el que aparece en la imagen siguiente:

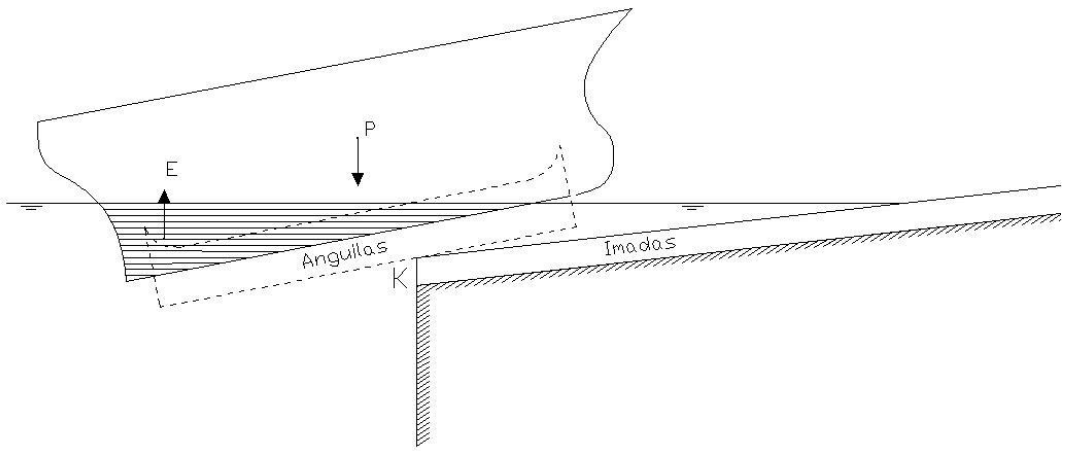

Imagen 94. Arfada

Tras ese giro, el barco debe recuperar su posición o, mediante una vía de agua, se inunda. Por ello hay que evitar este fenómeno durante el proceso de la botadura.

8.13 Volumen de carena

Volumen de carena es el volumen de la parte sumergida del barco, este volumen es diferente para cada flotación. El volumen de carena nos ayudará a calcular el empuje a partir del principio de Arquímedes. ⋘ Un cuerpo total o parcialmente sumergido en un fluido en reposo, recibe un empuje de abajo hacia arriba igual al peso del volumen del fluido que desaloja ⋙.

Gráficamente, el volumen de carena es el volumen representado de amarillo en la siguiente imagen:

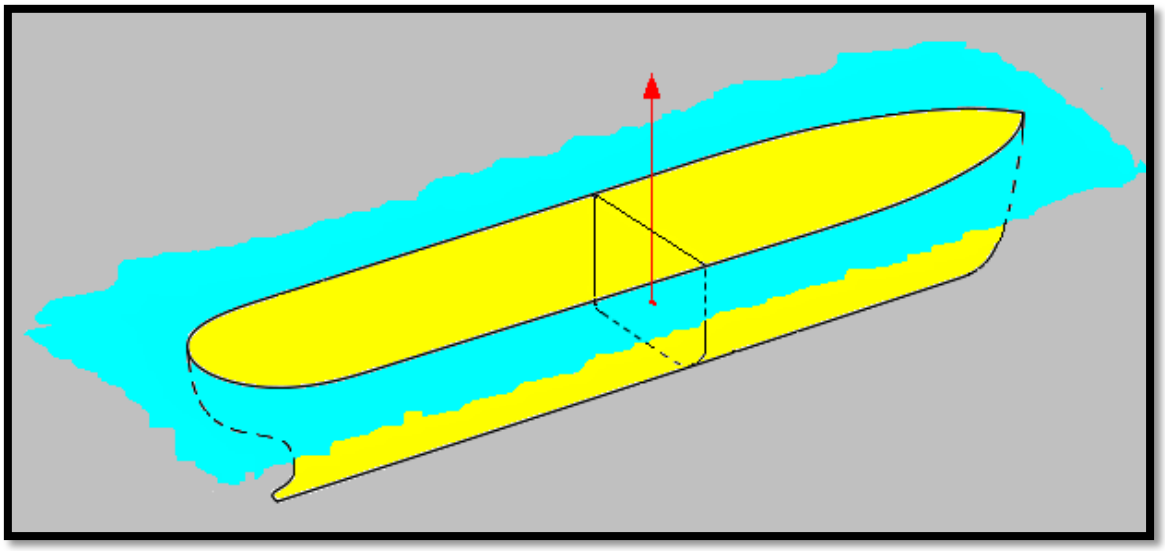

Imagen 95. Volumen de carena

8.14 Área de la flotación

El área del plano de la flotación o área de la flotación, es el área encerrada entre la intersección del casco con el agua en la línea de flotación. Este área encerrada se calcula a partir de una línea de agua dada.

Para cada flotación o línea de agua, los valores de las semimangas correspondientes a las diferentes cuadernas de trazado se obtienen del plano de formas, por lo que la integral puede resolverse mediante alguno de los métodos de integración numérica explicados en el apartado de integración numérica.

El área de la flotación, se representa en la siguiente imagen:

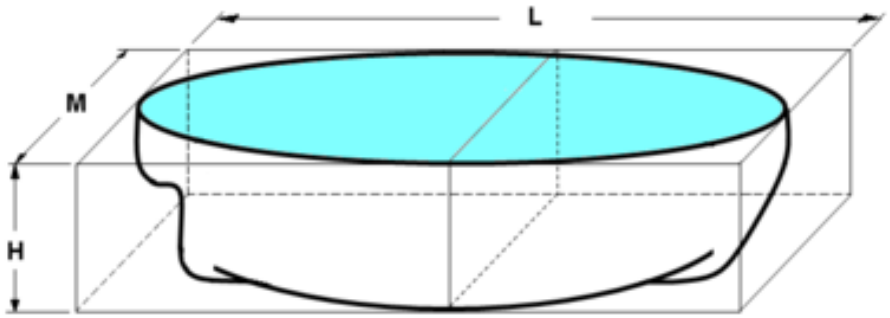

Imagen 96. Área de la flotación

8.15 Área de la sección

Es el área de la sección transversal del casco en un determinado punto de la eslora. En la imagen siguiente puede verse una representación de la misma.

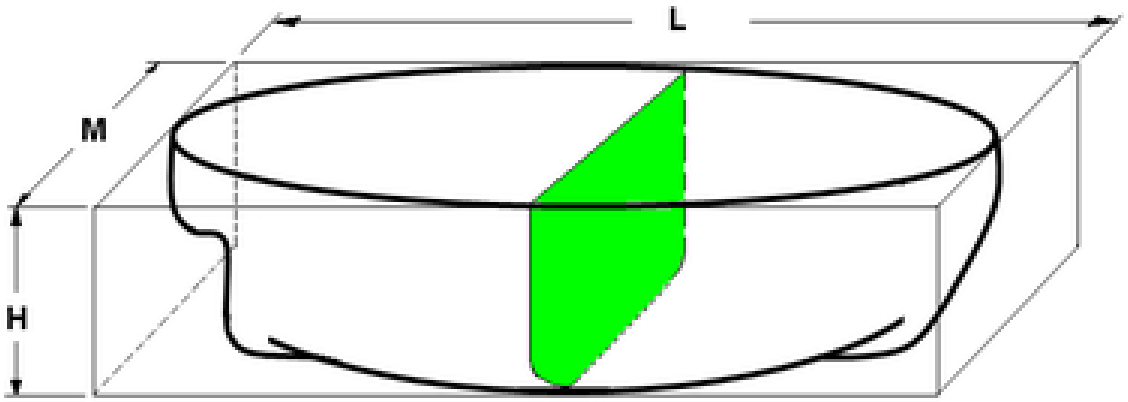

Imagen 97. Área de la sección

8.16 Momento del área de la sección

El momento del área de la sección, es el momento estático o de primer orden del área de dicha sección. Es una magnitud geométrica que se define para un área plana. Este momento, coincide con el producto del área total multiplicado por la distancia entre el punto considerado al centroide del área.

8.17 Centro de flotación

Se denomina centro de flotación al centro geométrico del área que abarca el plano de flotación, generado por el casco del barco y la línea de agua a lo largo del mismo. Este centro de flotación se situará normalmente en la línea vertical a la quilla, aunque puede ser que a veces esté desviado del mismo y se acerque más a uno de los costados del barco que al otro.

8.18 Toneladas por centímetro de inmersión

Las TCI o Toneladas por centímetro de inmersión, es un dato de gran interés, ya que indica las toneladas necesarias que hay que añadir (quitar) para que el barco sumerja (emerja) la línea de flotación actual del barco un centímetro.

Este dato estará incluido en los datos hidrostáticos del barco para cada una de las flotaciones indicadas y se obtiene multiplicando el área de la flotación por 1 centímetro por la densidad del fluido.

Las TCI van variando conforme va variando el calado, ya que el área de la flotación no es constante a lo largo del calado del barco.

8.19 Centro vertical de carena

La posición del centro de carena es imprescindible para conocer la estabilidad del barco. Tanto la posición vertical como la posición horizontal de dicho punto, son necesarios para obtener la estabilidad. La coordenada vertical del centro de carena, nos indica la distancia vertical que hay desde la posición del plano base, que en muchas ocasiones coincide con la quilla del barco, hasta la posición del centro de gravedad del volumen de agua (u otra sustancia) desalojado.

Esta distancia estará referida como $\overline{ZB}$ o $\overline{KB}$.

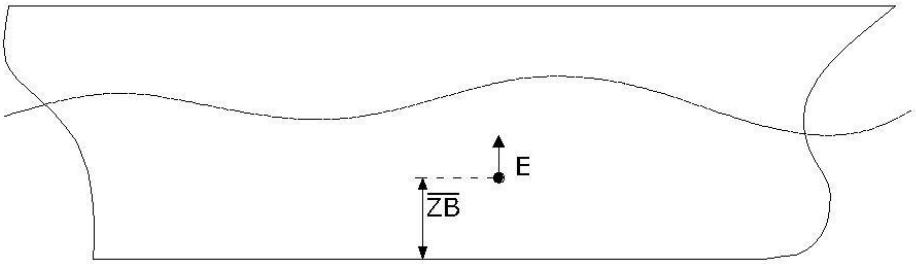

Imagen 98. Centro vertical de carena (ZB)

8.20 Coordenada horizontal del centro de carena

La posición del centro de carena es imprescindible para conocer la estabilidad del barco. Tanto la posición vertical como la posición horizontal de dicho punto, son necesarios para obtener la estabilidad. La coordenada horizontal del centro de carena, nos indica la distancia que hay desde la posición x=0, que en este programa coincide con la parte más a popa del barco, hasta la posición del centro de gravedad del volumen de agua (u otra sustancia) desalojado.

Esta distancia estará referida en el programa como $\overline{XB}$. También puede estar referida en algunos documentos esa misma distancia a la cuaderna maestra como $\overline{\otimes B}$.

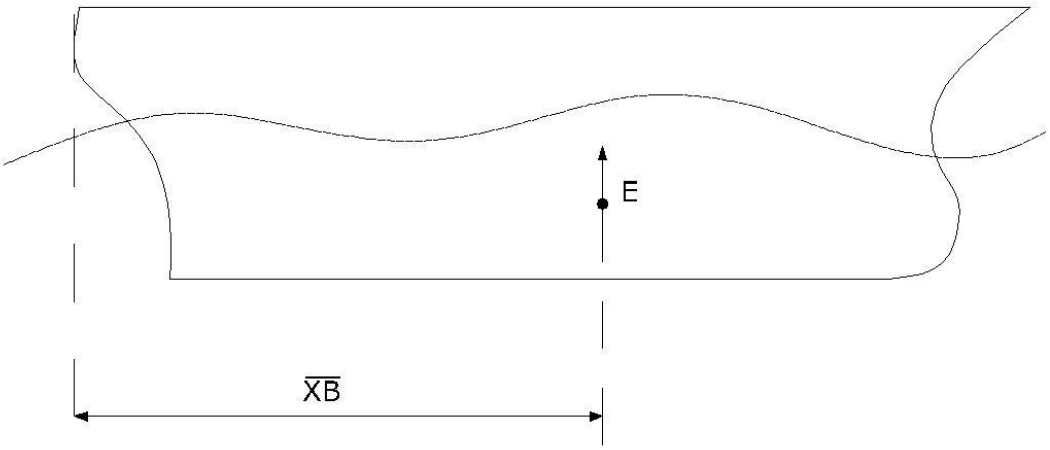

Imagen 99. Coordenada horizontal del centro de carena (XB)

8.21 Radio metacéntrico transversal

El valor del radio metacéntrico transversal, $\overline{\mathrm{BM_t}}$, se denomina así porque, haciendo centro en M y girando por B con radio $\overline{\mathrm{BM}}$, la circunferencia trazada coincidiría, muy aproximadamente, con la curva que pasa por el centro de carena para escoras infinitesimales.

El valor del radio metacéntrico transversal se obtiene a partir de los valores de los movimientos transversal, longitudinal y vertical del centro de carena. El radio metacéntrico tiene como ecuación:

$$\overline{\mathrm{BM_t}} = \frac{\mathrm{I_t}}{\nabla}.$$

Siendo I_t el momento de inercia de la superficie de flotación con respecto al eje longitudinal y siendo ∇ el volumen de carena.

El radio metacéntrico transversal es un dato imprescindible para la obtención de la estabilidad transversal del barco.

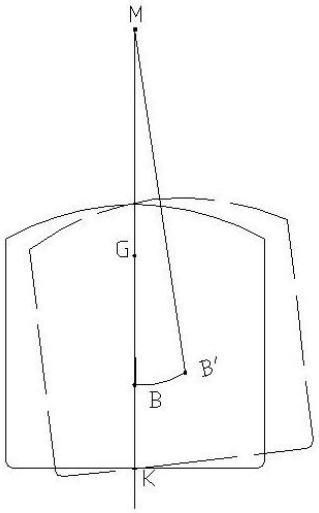

Imagen 100. Radio metacéntrico transversal

8.22 Radio metacéntrico longitudinal

El valor del radio metacéntrico longitudinal, $\overline{\mathrm{BM_l}}$, se denomina así porque, haciendo centro en M y girando por B longitudinalmente con radio $\overline{\mathrm{BM}}$, la circunferencia trazada coincidiría, muy aproximadamente, con la curva que pasa por el centro de carena para escoras infinitesimales.

El valor del radio metacéntrico longitudinal se obtiene a partir de los valores de los movimientos transversal, longitudinal y vertical del centro de carena. El radio metacéntrico longitudinal tiene como ecuación:

$$\overline{\mathrm{BM_l}} = \frac{\mathrm{I_l}}{\nabla}.$$

Siendo I_l el momento de inercia de la superficie de flotación con respecto al eje transversal y siendo ∇ el volumen de carena.

El radio metacéntrico longitudinal es un dato imprescindible para la obtención de la estabilidad longitudinal del barco.

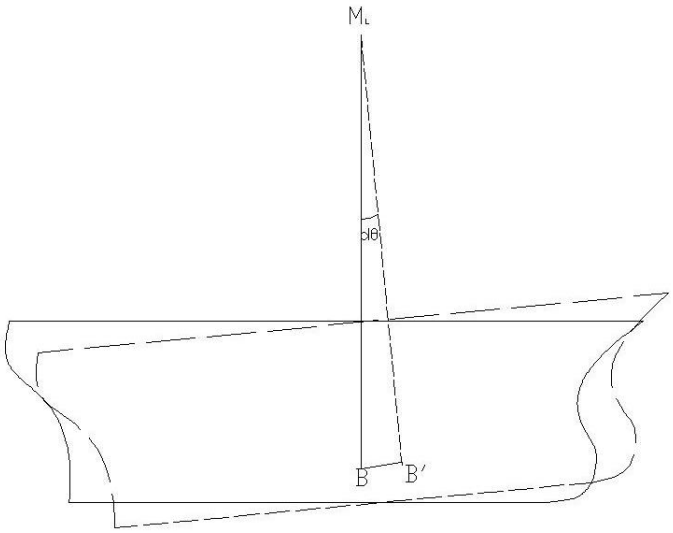

Imagen 101. Radio metacéntrico longitudinal

8.23 Momento de inercia transversal

El momento de inercia es una medida de la inercia rotacional de un cuerpo. Cuando un cuerpo gira en torno a uno de los ejes principales de inercia, la inercia rotacional puede ser representada como una magnitud escalar llamada momento de inercia.

El momento de inercia refleja la distribución de masa de un cuerpo o de un sistema de partículas en rotación, respecto a un eje de giro. En este caso, el momento de inercia sólo depende de la geometría del cuerpo y de la posición del eje de giro; pero no depende de las fuerzas que intervienen en el movimiento.

El momento de inercia desempeña un papel análogo al de la masa inercial en el caso del movimiento rectilíneo y uniforme. Es el valor escalar del momento angular longitudinal de un sólido rígido.

Nuestro cálculo, se centrará en el momento que se genera a partir del eje longitudinal que cruza el barco de proa a popa por el eje de giro, que pasa por el centro de masa y coincidirá con la proyección vertical de la quilla, normalmente.

Se calculará a partir del teorema de Steiner o teorema de los ejes paralelos el cual se define como:

“El momento de inercia con respecto a cualquier eje paralelo a un eje que pasa por el centro de masa, es igual al momento de inercia con respecto al eje que pasa por el centro de masa más el producto de la masa por el cuadrado de la distancia entre los dos ejes”.

8.24 Momento de inercia longitudinal

El momento de inercia es una medida de la inercia rotacional de un cuerpo. Cuando un cuerpo gira en torno a uno de los ejes principales de inercia, la inercia rotacional puede ser representada como una magnitud escalar llamada momento de inercia.

El momento de inercia refleja la distribución de masa de un cuerpo o de un sistema de partículas en rotación, respecto a un eje de giro. En este caso, el momento de inercia sólo depende de la geometría del cuerpo y de la posición del eje de giro; pero no depende de las fuerzas que intervienen en el movimiento.

El momento de inercia desempeña un papel análogo al de la masa inercial en el caso del movimiento rectilíneo y uniforme. Es el valor escalar del momento angular longitudinal de un sólido rígido.

Nuestro cálculo, se centrará en el momento que se genera a partir del eje transversal que cruza el barco de babor a estribor por el eje de giro, que pasa por el centro de masa del barco.

Se calculará a partir del teorema de Steiner o teorema de los ejes paralelos el cual se define como:

“El momento de inercia con respecto a cualquier eje paralelo a un eje que pasa por el centro de masa, es igual al momento de inercia con respecto al eje que pasa por el centro de masa más el producto de la masa por el cuadrado de la distancia entre los dos ejes”.

8.25 Momento de inercia de la flotación

El momento de inercia de la flotación, se calculará a partir del eje transversal o eje transversal OY.

Momento de inercia entre los límites x_1 y x_2:

$$I_{OY} = \int_{x_1}^{x_2} x^2 \cdot y \cdot dx$$

Función a integrar:

$$x^2 \cdot y$$

Por lo que el momento de inercia de la flotación respecto al eje transversal que pasa por el origen de coordenadas es calculado como:

$$I_y = 2 \cdot \int_a^b x^2 \cdot y \cdot dx$$

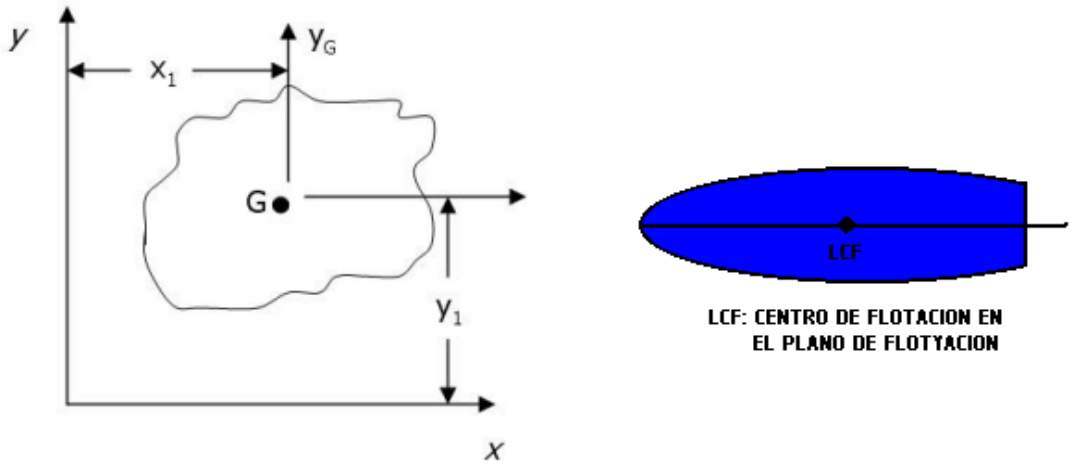

Imagen 102. Momento de inercia de la flotación

8.26 Momento de desplazamiento

El momento de volumen de desplazamiento (MB), se calcula mediante la integración "vertical" de las áreas de flotación desde el punto más bajo del casco hasta la línea de flotación o calado. La integral a calcular sería la siguiente:

$$MB = \int_0^T T \cdot A_z \cdot dz.$$

Donde T es el calado de cálculo y A_z el área de la flotación.

Para cada calado, la carena derecha correspondiente tendrá un determinado centro de carena. Por razones obvias de simetría, sobre el plano ce crujía, bastará con dos coordenadas para definir su posición perfectamente: su distancia a una sección transversal determinada, generalmente la sección maestra o la perpendicular de popa, y su altura sobre el plano de construcción.

La coordenada vertical del centro de carena se calculará como:

$$\overline{KB} = \overline{ZB} = \frac{MB}{\nabla}.$$

Calculamos el momento del volumen de desplazamiento respecto de la sección maestra o de la sección de perpendicular de popa como:

$$MB_x = \int_0^x x \cdot As \cdot dx.$$

Anexo I

9. ANEXO

En este ANEXO I, se van a incluir todos los datos de los comandos del programa informático.

El programa "ValoresHidrostaticos" viene separado en varios grupos de comandos, donde unos "se llaman" a otros para la ejecución correcta del programa.

Explicaré muy brevemente la función de cada serie de comandos. Además, en cada serie de comandos viene explicado de manera escueta la finalidad de cada paso de la programación.

Toda esta programación es la necesaria para que MATLAB realice los cálculos, muestre las diferentes interfaces gráficas, introduzca los datos en los cálculos, y represente todos los resultados que se le indican.

Este ANEXO I viene acompañado de un índice para poder moverse más fácilmente entre las series de comandos que nos interesen consultar.

10. Índice del ANEXO I y funciones del ANEXO

10.1 Función "Botadura"

En el programa botadura, se incluyen todos los comandos necesarios para la ejecución del programa, cargar la imagen y mandar los datos al programa. En estos comandos se introducen los datos en las funciones definidas para ello.

Indica de los archivos que hay que cargar, nos da información acerca de los comandos que aparecen en la interfaz, siendo la encargada de abrir los documentos que explican cada término.

Muestra el dibujo del proceso de cálculo de la arfada, los mensajes de información o error en cuanto a los cálculos de saludo y arfada, indica los resultados del momento en los santos de proa.

Si los datos de entrada no son correctos o no están indicados, muestra mensaje de aviso indicando donde está el error.

Se encarga también de ejecutar el programa botadura, resetear los datos y restablecerlos a los valores iniciales, y avisa si queremos salir del programa.

```
function varargout = botadura(varargin)
% BOTADURA M-file for botadura.fig
%       BOTADURA, by itself, creates a new BOTADURA or raises the
existing
%      singleton*.
%
%       H = BOTADURA returns the handle to a new BOTADURA or the
handle to
%      the existing singleton*.
%
%         BOTADURA('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in BOTADURA.M with the given input
arguments.
%
%       BOTADURA('Property','Value',...) creates a new BOTADURA or
raises the
%         existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before botadura_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%        stop.  All inputs are passed to botadura_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 botadura

% Last Modified by GUIDE v2.5 16-Jul-2014 01:44:17

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @botadura_OpeningFcn, ...
    'gui_OutputFcn',  @botadura_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 botadura is made visible.
function botadura_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 botadura (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%%%%%%%%%%%%%%%    CENTRAR    INTERFAZ    GRÁFICA    EN    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 );


% --- Outputs from this function are returned to the command line.
function varargout = botadura_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;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%%%%%%%%%%%%%%%%%    INCLUIMOS    IMAGEN    DE    LA    BOTADURA
%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

bot=imread('PresentacionBotadura.jpg');
axes(handles.axes_botadura);
imshow(bot); axis off



function edit_fa_Callback(hObject, eventdata, handles)
% hObject    handle to edit_fa (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 edit_fa as text
%        str2double(get(hObject,'String')) returns contents of
edit_fa as a double


% --- Executes during object creation, after setting all
properties.
function edit_fa_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_fa (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 Cargar.
      function Cargar_Callback(hObject, eventdata, handles)
      % hObject    handle to Cargar (see GCBO)
      % eventdata  reserved - to be defined in a future version of
MATLAB
      % handles    structure with handles and user data (see GUIDATA)

      [filename,pathname,filterindex]=uigetfile( ...
          { '*.xls', 'Archivo que contiene los cálculos hidrostáticos en
adrizado (*.xls)'},'Seleccione un archivo');

      if filterindex == 1
          set(handles.edit_fa,'String',filename);
      end

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

      cd('Líneas de Agua');
      [filename,pathname,filterindex]=uigetfile( ...
          {  '*.xls',  'Archivo  que  contiene  las  líneas  de  agua
(*.xls)'},'Seleccione un archivo');
      cd ..

      if filterindex == 1
          set(handles.edit_lineas,'String',filename);
      end


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

      open('pdf/Peso total.pdf');

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

      open('pdf/XG.pdf');

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

```
        open('pdf/ZG.pdf');

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

        open('pdf/Densidad.pdf');

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


        open('pdf/AnguloGrada.pdf');

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


        open('pdf/AlturaAnguila.pdf');

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


        open('pdf/DistanciaAnguila.pdf');

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


        open('pdf/LongitudGrada.pdf');

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

```
        open('pdf/AlturaK.pdf');

        function edit_peso_Callback(hObject, eventdata, handles)
        % hObject    handle to edit_peso (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 edit_peso as
text
        %        str2double(get(hObject,'String')) returns contents of
edit_peso as a double


        % --- Executes during object creation, after setting all
properties.
        function edit_peso_CreateFcn(hObject, eventdata, handles)
        % hObject    handle to edit_peso (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 radiobutton_longitud.
        function radiobutton_longitud_Callback(hObject, eventdata,
handles)
        % hObject    handle to radiobutton_longitud (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
radiobutton_longitud

        if get(handles.radiobutton_longitud,'Value')

            set(handles.radiobutton_altk,'Value',0);
            set(handles.edit_altk,'Enable','off');
            set(handles.edit_longitud,'Enable','on');

        else
            set(handles.edit_longitud,'Enable','off');
        end


        % --- Executes on button press in radiobutton_altk.
        function radiobutton_altk_Callback(hObject, eventdata, handles)
        % hObject    handle to radiobutton_altk (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
radiobutton_altk

      if get(handles.radiobutton_altk,'Value')

          set(handles.radiobutton_longitud,'Value',0);
          set(handles.edit_longitud,'Enable','off');
          set(handles.edit_altk,'Enable','on');

      else
          set(handles.edit_altk,'Enable','off');
      end


      function edit_xg_Callback(hObject, eventdata, handles)
      % hObject    handle to edit_xg (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 edit_xg as text
      %           str2double(get(hObject,'String')) returns contents of
edit_xg as a double


      %  ---  Executes  during  object  creation,  after  setting  all
properties.
      function edit_xg_CreateFcn(hObject, eventdata, handles)
      % hObject    handle to edit_xg (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 edit_zg_Callback(hObject, eventdata, handles)
      % hObject    handle to edit_zg (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 edit_zg as text
      %           str2double(get(hObject,'String')) returns contents of
edit_zg as a double


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

```
function edit_zg_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_zg (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 edit_densidad_Callback(hObject, eventdata, handles)
% hObject    handle to edit_densidad (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 edit_densidad
as text
%        str2double(get(hObject,'String')) returns contents of
edit_densidad as a double


% --- Executes during object creation, after setting all
properties.
function edit_densidad_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_densidad (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 edit_angulo_Callback(hObject, eventdata, handles)
% hObject    handle to edit_angulo (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 edit_angulo as
text
%        str2double(get(hObject,'String')) returns contents of
edit_angulo as a double
```

```
      % --- Executes during object creation, after setting all
properties.
      function edit_angulo_CreateFcn(hObject, eventdata, handles)
      % hObject    handle to edit_angulo (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 edit_altura_Callback(hObject, eventdata, handles)
      % hObject    handle to edit_altura (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 edit_altura as
text
      %          str2double(get(hObject,'String')) returns contents of
edit_altura as a double


      % --- Executes during object creation, after setting all
properties.
      function edit_altura_CreateFcn(hObject, eventdata, handles)
      % hObject    handle to edit_altura (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 edit_dang_Callback(hObject, eventdata, handles)
      % hObject    handle to edit_dang (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 edit_dang as
text
      %          str2double(get(hObject,'String')) returns contents of
edit_dang as a double
```

```
      % --- Executes during object creation, after setting all
properties.
      function edit_dang_CreateFcn(hObject, eventdata, handles)
      % hObject    handle to edit_dang (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 edit_longitud_Callback(hObject, eventdata, handles)
      % hObject    handle to edit_longitud (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 edit_longitud
as text
      %          str2double(get(hObject,'String')) returns contents of
edit_longitud as a double


      % --- Executes during object creation, after setting all
properties.
      function edit_longitud_CreateFcn(hObject, eventdata, handles)
      % hObject    handle to edit_longitud (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 edit_altk_Callback(hObject, eventdata, handles)
      % hObject    handle to edit_altk (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 edit_altk as
text
      %          str2double(get(hObject,'String')) returns contents of
edit_altk as a double
```

```
      % --- Executes during object creation, after setting all
properties.
      function edit_altk_CreateFcn(hObject, eventdata, handles)
      % hObject    handle to edit_altk (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 pushbutton_saludo.
      function pushbutton_saludo_Callback(hObject, eventdata, handles)
      % hObject    handle to pushbutton_saludo (see GCBO)
      % eventdata  reserved - to be defined in a future version of
MATLAB
      % handles    structure with handles and user data (see GUIDATA)

      open('pdf/Saludo.pdf');

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

      open('pdf/MomentoSantos.pdf');

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

      open('pdf/Arfada.pdf');


      % --- Executes on button press in checkbox_saludo.
      function checkbox_saludo_Callback(hObject, eventdata, handles)
      % hObject    handle to checkbox_saludo (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
checkbox_saludo
```

```
% --- Executes on button press in checkbox_santos.
function checkbox_santos_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox_santos (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
checkbox_santos

if get(handles.checkbox_santos,'Value')
    set(handles.edit_santos,'Visible','on');
else
    set(handles.edit_santos,'Visible','off');
end


% --- Executes on button press in checkbox_arfada.
function checkbox_arfada_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox_arfada (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
checkbox_arfada


if get(handles.checkbox_arfada,'Value')
    set(handles.checkbox_dibujo,'Visible','on');
else
    set(handles.checkbox_dibujo,'Visible','off');
end

% --- Executes on button press in checkbox_dibujo.
function checkbox_dibujo_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox_dibujo (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
checkbox_dibujo


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


% leemos los ficheros xls

ficheroH= get(handles.edit_fa,'String');
ficheroH=fliplr(deblank(fliplr(deblank(ficheroH))));   %Eliminamos
huecos en blanco
```

```
        % Comprobamos que no está vacío
        if isempty(ficheroH)==1
            uiwait(msgbox('Debe introducir un fichero .xls con los
cálculos hidrostáticos en posición de adrizado', 'Mensaje de error',...
                'error','modal'))
            return;
        end

        ficheroL= get(handles.edit_lineas,'String');
        ficheroL=fliplr(deblank(fliplr(deblank(ficheroL))));   %Eliminamos
huecos en blanco

        % Comprobamos que no está vacío
        if isempty(ficheroL)==1
            uiwait(msgbox('Debe introducir un fichero .xls con los datos
de las lineas de agua', 'Mensaje de error',...
                'error','modal'))
            return;
        end


        % leemos el peso

        aux = get(handles.edit_peso,'String');
        peso=sscanf(aux, '%f');

        % Comprobamos que no está vacío
        if isempty(peso)==1 | peso<=0
            uiwait(msgbox('Debe introducir el peso correctamente.',
'Mensaje de error',...
                'error','modal'))
            return;
        end


        % leemos XG

        aux = get(handles.edit_xg,'String');
        Xg=sscanf(aux, '%f');

        % Comprobamos que no está vacío
        if isempty(Xg)==1 | Xg<=0
            uiwait(msgbox('Debe introducir XG correctamente.', 'Mensaje de
error',...
                'error','modal'))
            return;
        end


        % leemos ZG

        aux = get(handles.edit_zg,'String');
        Zg=sscanf(aux, '%f');

        % Comprobamos que no está vacío
        if isempty(Zg)==1 | Zg<=0
```

```
            uiwait(msgbox('Debe introducir ZG correctamente.', 'Mensaje de
error',...
                'error','modal'))
            return;
        end


        % leemos la densidad

        aux = get(handles.edit_densidad,'String');
        densidad=sscanf(aux, '%f');

        % Comprobamos que no está vacío
        if isempty(densidad)==1 | densidad<=0
            uiwait(msgbox('Debe introducir la densidad correctamente.',
'Mensaje de error',...
                'error','modal'))
            return;
        end

        % leemos la altura de la anguila

        aux = get(handles.edit_altura,'String');
        hang=sscanf(aux, '%f');

        % Comprobamos que no está vacío
        if isempty(hang)==1 | hang<=0
            uiwait(msgbox('Debe introducir la altura de la anguila
correctamente.', 'Mensaje de error',...
                'error','modal'))
            return;
        end

        % leemos la distancia entre la proa de la anguila y la proa del
barco

        aux = get(handles.edit_dang,'String');
        dang=sscanf(aux, '%f');

        % Comprobamos que no está vacío
        if isempty(dang)==1 | dang<=0
            uiwait(msgbox('Debe introducir la posición de la anguila
correctamente.', 'Mensaje de error',...
                'error','modal'))
            return;
        end


        % leemos el ángulo de la grada

        aux = get(handles.edit_angulo,'String');
        alfa=sscanf(aux, '%f');
        alfa=alfa*pi/180;

        % Comprobamos que no está vacío
        if isempty(alfa)==1 | alfa<=0
            uiwait(msgbox('Debe introducir el ángulo correctamente.',
'Mensaje de error',...
                'error','modal'))
```

```
            return;
        end


        if get(handles.radiobutton_longitud,'Value')

            op='1';
        % leemos la longitud de la grada

        aux = get(handles.edit_longitud,'String');
        dg=sscanf(aux, '%f');
        variable2=dg;

        % Comprobamos que no está vacío
        if isempty(dg)==1 | dg<=0
            uiwait(msgbox('Debe introducir la longitud de la grada
correctamente.', 'Mensaje de error',...
                'error','modal'))
            return;
        end


        elseif get(handles.radiobutton_altk,'Value')

          op='2';
        % leemos la altura del agua en el punto k

        aux = get(handles.edit_altk,'String');
         caladok=sscanf(aux, '%f');
         variable2=caladok;

        % Comprobamos que no está vacío
        if isempty(caladok)==1 | caladok<=0
            uiwait(msgbox('Debe introducir la altura del agua en el punto
K correctamente.', 'Mensaje de error',...
                'error','modal'))
            return;
        end


        else

             uiwait(msgbox('Debe introducir o bien la longitud de la grada
o bien la altura del agua en el punto K.', 'Mensaje de error',...
                'error','modal'))
            return;

        end


        if get(handles.checkbox_saludo,'Value')
            [v]=interpolacion_variables_hidrostaticas(peso,ficheroH);

[dnec,dg,sal,caladok]=saludo(ficheroH,v,Xg,hang,dang,alfa,op,variable2);
        end

        if get(handles.checkbox_santos,'Value')
            [v]=interpolacion_variables_hidrostaticas(peso,ficheroH);
```

```
            [Ms]=MomentoEnLosSantosProa(v,hang,dang,Xg,Zg,alfa);
            set(handles.edit_santos,'String',num2str(Ms));
        end

        if get(handles.checkbox_arfada,'Value')

            if get(handles.checkbox_dibujo,'Value')

[xgiro,arf,xarf]=giro_arfada(ficheroH,ficheroL,peso,Xg,Zg,hang,dang,alfa
,densidad,op,variable2,1);
            else

[xgiro,arf,xarf]=giro_arfada(ficheroH,ficheroL,peso,Xg,Zg,hang,dang,alfa
,densidad,op,variable2,0);
            end

        end



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


        % Reseteamos el nombre del fichero

        set(handles.edit_fa,'String','');
        set(handles.edit_lineas,'String','');

        %  Reseteamos el resto de edit

        set(handles.edit_peso,'String','');
        set(handles.edit_xg,'String','');
        set(handles.edit_zg,'String','');
        set(handles.edit_densidad,'String','');
        set(handles.edit_altura,'String','');
        set(handles.edit_dang,'String','');
        set(handles.edit_angulo,'String','');
        set(handles.edit_altk,'String','');
        set(handles.edit_altk,'Enable','off');
        set(handles.edit_longitud,'String','');
        set(handles.edit_longitud,'Enable','off');
        set(handles.edit_santos,'String','');
        set(handles.edit_longitud,'Visible','off');

        % Reseteamos los radiobutton

        set(handles.radiobutton_longitud,'Value',0);
        set(handles.radiobutton_altk,'Value',0);

        % Reseteamos los checkbox

        set(handles.checkbox_saludo,'Value',0);
        set(handles.checkbox_santos,'Value',0);
```

```
      set(handles.checkbox_arfada,'Value',0);
      set(handles.checkbox_dibujo,'Value',0);
      set(handles.checkbox_dibujo,'Visible','off');


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

      salir=questdlg('¿Desea salir del programa?','Salida del
Programa','Si','No','No');
      switch salir
          case 'Si'
              close
          case 'No'
              return;
      end


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


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

      open('pdf/Acerca De.pdf');

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

      open('pdf/Acerca De.pdf');

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

```
      open('pdf/Acerca De.pdf');


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

      salir=questdlg('¿Desea salir del programa?','Salida del Programa','Si','No','No');
      switch salir
          case 'Si'
              close
          case 'No'
              return;
      end



      function edit_lineas_Callback(hObject, eventdata, handles)
      % hObject    handle to edit_lineas (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 edit_lineas as text
      %        str2double(get(hObject,'String')) returns contents of edit_lineas as a double


      % --- Executes during object creation, after setting all properties.
      function edit_lineas_CreateFcn(hObject, eventdata, handles)
      % hObject    handle to edit_lineas (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 edit_santos_Callback(hObject, eventdata, handles)
      % hObject    handle to edit_santos (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 edit_santos as
text
      %          str2double(get(hObject,'String')) returns contents of
edit_santos as a double


      % --- Executes during object creation, after setting all
properties.
      function edit_santos_CreateFcn(hObject, eventdata, handles)
      % hObject    handle to edit_santos (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
```

10.2 Función "Curvas Hidrostáticas"

En esta función, se calculan los datos hidrostáticos del barco a partir de los archivos en los que están contenidos los datos de las formas del barco. Estos cálculos se realizan para los posibles calados del barco.

Esta función es de las más completas, y de las que más formulario incluye. En ella se incluyen las fórmulas, ya explicadas en el proyecto, de:

- Área de la flotación
- Momento del área de la flotación
- Centro de flotación
- Momento transversal de inercia
- Momento de inercia del plano de agua
- Momento longitudinal de inercia
- TCI (Toneladas por centímetro de inmersión)
- Volumen de desplazamiento
- Desplazamiento
- Momento de desplazamiento respecto de z
- Coordenada del centro de carena
- Radio metacéntrico longitudinal
- Radio metacéntrico transversal
- Área de cada sección
- Momento respecto de la sección
- Eslora
- MTC (Momento para alterar el trimado un centímetro)
- Coeficientes (de bloque, de la maestra, de flotación, prismático horizontal y prismático vertical)

Además de ello, se encarga de dibujarlos todos y representarlos gráfica y numéricamente. Ya sea para cada calado o para un valor discreto.

```
function  curvas_hidrostaticas(fichero,op,ksec)
% Esta función realiza cálculos hidrostáticos a partir
```

```
% del conocimiento de la posición de las secciones, y
% de las semimangas para diferentes calados
% Los cálculos hidrostáticos que realiza son:
% Aw  Área de flotación para cada línea de agua
% Mx  Momento de área del plano del agua
% Xf  Centro de flotación
% It  Momento de inercia transversal
% Iy  Momento de inercia del plano de agua
% IL  Momento de inercia longitudinal
% TPC Toneladas por centimetro de inmersión
% V   Volumen de desplazamiento o de carena
% MB  Momento de desplazamiento del volumen
% ZB  Centro vertical de carena
% BMt Radio metacéntrico transversal
% BMl Radio metacéntrico longitudinal
% As  Área de las secciones
% MBx Momento respecto de la sección (sin curva)
% XB  Centro longitudinal de carena  (sin curva)
%
% curvas_hidrostaticas(fichero,op,ksec)
% fichero es el nombre del archivo excel que contiene las lineas
de agua
% op indica si quiere gráfica de los distintos elementos o no
%     op='sssssssssssn' indica que se quieren todas las gráficas
menos la
%    de As
%    Notar que se trata de 13 curvas hidróstaticos distintos
% ksec número de sección del cual se quiere saber el área


% Leemos el fichero con la posición de las secciones, y
% las semimangas para cada altura z

a=xlsread(['./Líneas de Agua/',fichero]);
[n,m]=size(a);

% secciones
x=a(1,2:m);     % tamaño 1 x m-1

% alturas
z=a(2:n,1);     % tamaño n-1 x 1

% semimangas
y=a(2:n,2:m);   % tamaño n-1 x m-1


% Densidad del aguas en el oceano en el Sistema Internacional
pw = 1.025;     % en ton/m^3


% Realiza el cálculo de los parámetros hidrostáticos de cada uno
de los calados

for k=1:n-1

    y_linea=y(k,1:m-1);
    Aw(k)=2*trapecios(x,y_linea);           % Área de flotación de
cada línea de agua
```

```
          Mx(k) = 2*trapecios(x,x.*y_linea);       % Momento de área del
plano del agua
          Xf(k) = Mx(k)/Aw(k);                     % Centro de flotación
          It(k) = 2/3*trapecios(x,y_linea.^3);     % Momento transversal
de inercia
          Iy(k) = 2*trapecios(x,(x.^2).*y_linea); % Momento de inercia
del plano de agua
          IL(k) = Iy(k)+ Xf(k)^2* Aw(k);           % Momento longitudinal
de inercia
          TCI (k) = Aw(k)*pw/100;                  % Toneladas por
centimetro de inmersión
      end


      for k=1:n-1
          V(k)=trapecios(z(k:n-1)',Aw(k:n-1));     % Volumen de
desplazamiento para cada calado
          D(k)=pw*V(k);                            % Desplazamiento para
cada calado
          MB(k)=trapecios(z(k:n-1)',z(k:n-1)'.*Aw(k:n-1)); % Momento de
desplazamiento respecto de z
          KB(k)=MB(k)/V(k);                        % Coordenada z del
centro de carena
          BMt(k)=It(k)/V(k);                       % Radio metacéntrico
transversal
          BMl(k)=IL(k)/V(k);                       % Radio metacéntrico
longitudinal
      end

      % Realiza el cálculo de los parámetros hidrostáticos de cada una
de las secciones
      % para cada calado


      for k=1:n-1
          for i=1:m-1
              y_linea=y(k:n-1,i)';
              As(k,i)=2*trapecios(z(k:n-1)',y_linea);     % Área de
cada sección
          end

          MBx(k)=trapecios(x,x.*As(k,:));                 % Momento
respecto de la sección
          XB(k)=MBx(k)/V(k);                               % Centro
Longitudinal de la carena

      end



      for k=1:n-1
          ind=find(y(k,1:m-1)~=0);
          xmin(k)=x(ind(1));                          % Mínimo valor
de x a cada calado
          xMax(k)=x(ind(length(ind)));                % Máximo valor
de x a cada calado
          Eslora(k)=xMax(k)-xmin(k);                       % Eslora
correspondiente a cada calado
```

```
            Manga_maxima(k)=2*max(y(k,1:m-1));              % Manga máxima
a cada calado
            MTC(k)=D(k)*BMl(k)/(100*Eslora(k));             % Momento para
alterar el trimado 1 cm
            Cb(k)=V(k)/Eslora(k)/Manga_maxima(k)/z(k);      % Coeficiente
de bloque
            Seccion_maxima(k)=max(As(k,:));
            Cm(k)=Seccion_maxima(k)/Manga_maxima(k)/z(k);   % Coeficiente
de la maestra
            Cf(k)=Aw(k)/Eslora(k)/Manga_maxima(k);          % Coeficiente
de Flotación
            Cpv(k)=Cb(k)/Cf(k);                             % Coeficiente
prismático vertical
            Cph(k)=Cb(k)/Cm(k);                             % Coeficiente
prismático horizontal
        end



        % Dibujamos el área de flotación

        if op(1)=='s'

            h1=figure('Name',[blanks(6),'Área                          de
flotación'],'Position',[100 100 1000 600]);

            plot(Aw,z,'b','LineW',2)

            grid on
            box on
            set(gca,'FontW','Bold')
            xlabel('Área de Flotación','FontW','Bold')
            ylabel('Calado ','FontW','Bold')

            % Centramos y escalamos
            set( h1,'Units', 'pixels' );
            screenSize = get(0, 'ScreenSize');
            position = get( h1,'Position');
            position(1) = (screenSize(3)-position(3))/2;
            position(2) = (screenSize(4)-position(4))/2;
            set( h1,'Position', position );


        end

        % Dibujamos el Momento de área del plano del agua

        if op(2)=='s'

            h2=figure('Name',[blanks(6),'Momento de área del plano del
agua'],'Position',[100 100 1000 600]);

            plot(Mx,z,'b','LineW',2)

            grid on
            box on
            set(gca,'FontW','Bold')
            xlabel('Momento de área del plano del agua','FontW','Bold')
```

```
            ylabel('Calado ','FontW','Bold')

            % Centramos y escalamos
            set( h2,'Units', 'pixels' );
            screenSize = get(0, 'ScreenSize');
            position = get( h2,'Position');
            position(1) = (screenSize(3)-position(3))/2;
            position(2) = (screenSize(4)-position(4))/2;
            set( h2,'Position', position );

        end

        % Dibujamos el Centro de flotación

        if op(3)=='s'

            h3=figure('Name',[blanks(6),'Centro                                    de
flotación'],'Position',[100 100 1000 600]);

            plot(Xf,z,'b','LineW',2)

            grid on
            box on
            set(gca,'FontW','Bold')
            xlabel('Centro de flotación','FontW','Bold')
            ylabel('Calado ','FontW','Bold')

            % Centramos y escalamos
            set( h3,'Units', 'pixels' );
            screenSize = get(0, 'ScreenSize');
            position = get( h3,'Position');
            position(1) = (screenSize(3)-position(3))/2;
            position(2) = (screenSize(4)-position(4))/2;
            set( h3,'Position', position );


        end


        % Dibujamos el Momento transversal de inercia

        if op(4)=='s'

            h4=figure('Name',[blanks(6),'Momento      transversal      de
inercia'],'Position',[100 100 1000 600]);

            plot(It,z,'b','LineW',2)

            grid on
            box on
            set(gca,'FontW','Bold')
            xlabel('Momento transversal de inercia','FontW','Bold')
            ylabel('Calado ','FontW','Bold')

            % Centramos y escalamos
            set( h4,'Units', 'pixels' );
            screenSize = get(0, 'ScreenSize');
            position = get( h4,'Position');
```

```
            position(1) = (screenSize(3)-position(3))/2;
            position(2) = (screenSize(4)-position(4))/2;
            set( h4,'Position', position );


        end


        % Dibujamos el Momento de inercia del plano de agua

        if op(5)=='s'

            h5=figure('Name',[blanks(6),'Momento de inercia del plano de
agua'],'Position',[100 100 1000 600]);

            plot(Iy,z,'b','LineW',2)

            grid on
            box on
            set(gca,'FontW','Bold')
            xlabel('Momento de inercia del plano de agua','FontW','Bold')
            ylabel('Calado ','FontW','Bold')

            % Centramos y escalamos
            set( h5,'Units', 'pixels' );
            screenSize = get(0, 'ScreenSize');
            position = get( h5,'Position');
            position(1) = (screenSize(3)-position(3))/2;
            position(2) = (screenSize(4)-position(4))/2;
            set( h5,'Position', position );

        end


        % Dibujamos el Momento longitudinal de inercia

        if op(6)=='s'

            h6=figure('Name',[blanks(6),'Momento      longitudinal      de
inercia'],'Position',[100 100 1000 600]);

            plot(IL,z,'b','LineW',2)

            grid on
            box on
            set(gca,'FontW','Bold')
            xlabel('Momento longitudinal de inercia','FontW','Bold')
            ylabel('Calado ','FontW','Bold')

            % Centramos y escalamos
            set( h6,'Units', 'pixels' );
            screenSize = get(0, 'ScreenSize');
            position = get( h6,'Position');
            position(1) = (screenSize(3)-position(3))/2;
            position(2) = (screenSize(4)-position(4))/2;
            set( h6,'Position', position );

        end
```

```
% Dibujamos el Toneladas por centimetro de inmersión

if op(7)=='s'

    h7=figure('Name',[blanks(6),'Toneladas por centimetro de
inmersión'],'Position',[100 100 1000 600]);

    plot(TCI,z,'b','LineW',2)

    grid on
    box on
    set(gca,'FontW','Bold')
    xlabel('Toneladas por centimetro de inmersión','FontW','Bold')
    ylabel('Calado ','FontW','Bold')

    % Centramos y escalamos
    set( h7,'Units', 'pixels' );
    screenSize = get(0, 'ScreenSize');
    position = get( h7,'Position');
    position(1) = (screenSize(3)-position(3))/2;
    position(2) = (screenSize(4)-position(4))/2;
    set( h7,'Position', position );

end

% Dibujamos el Volumen de desplazamiento

if op(8)=='s'

    h8=figure('Name',[blanks(6),'Volumen de
desplazamiento'],'Position',[100 100 1000 600]);

    plot(V,z,'b','LineW',2)

    grid on
    box on
    set(gca,'FontW','Bold')
    xlabel('Volumen de desplazamiento','FontW','Bold')
    ylabel('Calado ','FontW','Bold')

    % Centramos y escalamos
    set( h8,'Units', 'pixels' );
    screenSize = get(0, 'ScreenSize');
    position = get( h8,'Position');
    position(1) = (screenSize(3)-position(3))/2;
    position(2) = (screenSize(4)-position(4))/2;
    set( h8,'Position', position );

end

% Dibujamos el Momento de desplazamiento respecto de z

if op(9)=='s'

    h9=figure('Name',[blanks(6),'Momento de desplazamiento
respecto de z'],'Position',[100 100 1000 600]);
```

```
            plot(MB,z,'b','LineW',2)

            grid on
            box on
            set(gca,'FontW','Bold')
            xlabel('Momento de desplazamiento respecto de
z','FontW','Bold')
            ylabel('Calado ','FontW','Bold')

            % Centramos y escalamos
            set( h9,'Units', 'pixels' );
            screenSize = get(0, 'ScreenSize');
            position = get( h9,'Position');
            position(1) = (screenSize(3)-position(3))/2;
            position(2) = (screenSize(4)-position(4))/2;
            set( h9,'Position', position );

      end

      % Dibujamos el Coordenada z del centro de carena

      if op(10)=='s'

            h10=figure('Name',[blanks(6),'Coordenada z del centro de
carena'],'Position',[100 100 1000 600]);

            plot(KB,z,'b','LineW',2)

            grid on
            box on
            set(gca,'FontW','Bold')
            xlabel('Coordenada z del centro de carena','FontW','Bold')
            ylabel('Calado ','FontW','Bold')

            % Centramos y escalamos
            set( h10,'Units', 'pixels' );
            screenSize = get(0, 'ScreenSize');
            position = get( h10,'Position');
            position(1) = (screenSize(3)-position(3))/2;
            position(2) = (screenSize(4)-position(4))/2;
            set( h10,'Position', position );

      end


      % Dibujamos el Radio metacéntrico transversal

      if op(11)=='s'

            h11=figure('Name',[blanks(6),'Radio metacéntrico
transversal'],'Position',[100 100 1000 600]);

            plot(BMt,z,'b','LineW',2)

            grid on
            box on
            set(gca,'FontW','Bold')
            xlabel('Radio metacéntrico transversal','FontW','Bold')
```

```
            ylabel('Calado ','FontW','Bold')

            % Centramos y escalamos
            set( h11,'Units', 'pixels' );
            screenSize = get(0, 'ScreenSize');
            position = get( h11,'Position');
            position(1) = (screenSize(3)-position(3))/2;
            position(2) = (screenSize(4)-position(4))/2;
            set( h11,'Position', position );

        end

        % Dibujamos el Radio metacéntrico longitudinal

        if op(12)=='s'

            h12=figure('Name',[blanks(6),'Radio                metacéntrico
longitudinal'],'Position',[100 100 1000 600]);

            plot(BMl,z,'b','LineW',2)

            grid on
            box on
            set(gca,'FontW','Bold')
            xlabel('Radio metacéntrico longitudinal','FontW','Bold')
            ylabel('Calado ','FontW','Bold')

            % Centramos y escalamos
            set( h12,'Units', 'pixels' );
            screenSize = get(0, 'ScreenSize');
            position = get( h12,'Position');
            position(1) = (screenSize(3)-position(3))/2;
            position(2) = (screenSize(4)-position(4))/2;
            set( h12,'Position', position );

        end

        % Dibujamos el Área de las Secciones

        if op(13)=='s'

            h=figure('Name',[blanks(6),'Área              de              las
secciones'],'Position',[100 100 1000 600]);
            plot(x,As,'b','LineW',2)

            grid on
            box on
            set(gca,'FontW','Bold')
            xlabel('Secciones','FontW','Bold')
            ylabel('Área de las Secciones','FontW','Bold')

            % Centramos y escalamos
            set( h,'Units', 'pixels' );
            screenSize = get(0, 'ScreenSize');
            position = get( h,'Position');
            position(1) = (screenSize(3)-position(3))/2;
            position(2) = (screenSize(4)-position(4))/2;
            set( h,'Position', position );
```

```
    end

    % Dibujamos el Coordenada z del centro de carena

    if op(14)=='s'

        h10=figure('Name',[blanks(6),'Coordenada x del centro de
carena'],'Position',[100 100 1000 600]);

        plot(XB,z,'b','LineW',2)

        grid on
        box on
        set(gca,'FontW','Bold')
        xlabel('Coordenada x del centro de carena','FontW','Bold')
        ylabel('Calado ','FontW','Bold')

        % Centramos y escalamos
        set( h10,'Units', 'pixels' );
        screenSize = get(0, 'ScreenSize');
        position = get( h10,'Position');
        position(1) = (screenSize(3)-position(3))/2;
        position(2) = (screenSize(4)-position(4))/2;
        set( h10,'Position', position );

    end



    % Mandamos los datos a una figura en la pantalla
    g=figure('Tag','datos','Name',[blanks(6),'Datos de
salida'],'Position',[100 100 1000 600]);
    axis off

    % Definimos una cadena de carácteres que contiene los datos de
salida
    str={['Calado',blanks(2),num2str(z(1)),blanks(2),'m'],...
        ['Área de flotación
(Aw):',blanks(2),num2str(Aw(1)),blanks(2),'m^2'],...
        ['Momento de área del plano del agua
(Mx):',blanks(2),num2str(Mx(1)),blanks(2),'m^3'],...
        ['Centro de flotación
(Xf):',blanks(2),num2str(Xf(1)),blanks(2),'m'],...
        ['Momento de inercia transversal
(It):',blanks(2),num2str(It(1)),blanks(2),'m^4'],...
        ['Radio metacéntrico transversal
(BMt):',blanks(2),num2str(BMt(1)),blanks(2),'m'],...
        ['Momento de inercia del plano de agua
(Iy):',blanks(2),num2str(Iy(1)),blanks(2),'m^4'],...
        ['Momento de inercia longitudinal
(IL):',blanks(2),num2str(IL(1)),blanks(2),'m^4'],...
        ['Radio metacéntrico longitudinal
(BMl):',blanks(2),num2str(BMl(1)),blanks(2),'m'],...
        ['Volumen de carena
(V):',blanks(2),num2str(V(1)),blanks(2),'m^3'],...
        ['Desplazamiento
(D):',blanks(2),num2str(D(1)),blanks(2),blanks(2),'ton'],...
```

```
          ['Momento                    de                    desplazamiento
(MB):',blanks(2),num2str(MB(1)),blanks(2),'m^3'],...
          ['Centro              vertical              de              carena
(KB):',blanks(2),num2str(KB(1)),blanks(2),'m'],...
          ['Toneladas       por       centímetro       de       inmersión
(TCI):',blanks(2),num2str(TCI(1)),blanks(2),'ton/cm'],...
          ['Centro           longitudinal           de           carena
(XB):',blanks(2),num2str(XB(1)),blanks(2),'m'],...
          ['Momento     para     alterar     el     trimado     1     cm
(MTC):',blanks(2),num2str(MTC(1)),blanks(2),'ton*m'],...
          ['Área          de          la          sección          (As)
',num2str(ksec),':',blanks(2),num2str(As(1,ksec)),blanks(2),'m^2'],...
          ['Coeficiente de bloque(Cb):',blanks(2),num2str(Cb(1))],...
          ['Coeficiente                        de                        la
maestra(Cm):',blanks(2),num2str(Cm(1))],...
          ['Coeficiente de flotación(Cf):',blanks(2),num2str(Cf(1))],...
          ['Coeficiente                                          prismático
horizontal(Cph):',blanks(2),num2str(Cph(1))],...
          ['Coeficiente                                          prismático
vertical(Cpv):',blanks(2),num2str(Cpv(1))]};

      g1=uicontrol('Style','Text','String',str,'FontSize',16,'FontWeight
','bold','Units','normalized',...
          'Position',[0                    0                    1
0.95],'Background','White','HorizontalAlignment','left');

      g2=uicontrol('Style','Text','String',  {'Valores  Hidrostáticos'},
'FontSize',18,'FontWeight','bold',...
          'Units','normalized','Position',[0          0.95          1
0.05],'Background','green','HorizontalAlignment','center','ForegroundCol
or','blue');

      % Centramos y escalamos
      set(g,'Units', 'pixels' );
      screenSize = get(0, 'ScreenSize');
      position = get(g,'Position');
      position(1) = (screenSize(3)-position(3))/2;
      position(2) = (screenSize(4)-position(4))/2;
      set(g,'Position', position );


      % Sacamos los datos en forma de tabla a un fichero excel

      tabla=[z,Aw',TCI',Xf',V',D',KB',XB',BMt',BMl',MTC',xmin',xMax',Esl
ora'];
      nombre_fichero_xls=['Cálculos                    hidrostáticos
adrizado_',fichero(1:end-3),'xls'];
      delete(nombre_fichero_xls);
      rango=['a2:N',num2str(n-1)];
      xlswrite(nombre_fichero_xls,tabla,rango);
      str_xls={'Calado','Aw','TCI','Xf','V','D','KB','XB','BMt','BMl','M
TC','xmin','xMax','Eslora'};
      xlswrite(nombre_fichero_xls,str_xls,'A1:N1');
```

10.3 Función "Líneas de agua"

Esta función te dibuja las líneas de agua del barco que le indicamos en 3D. Esta función se usa en el apartado del programa "Valores Hidrostáticos en Adrizado" donde nos representa el programa el barco de trabajo en 3D.

```
function dibujar_LineasDeAgua(fichero)

% Esta función dibuja en 3D las líneas de agua de un barco
% contenidas en un fichero dentro del directorio ./Líneas de Agua
% dibujar_LineasDeAgua(fichero)
% Variables de entrada:
% fichero nombre del archivo que contiene las líneas de agua del
barco


% Lee las alturas de las líneas de agua, las secciones, y las
semimangas

a=xlsread(['./Líneas de Agua/',fichero]);
[n,m]=size(a);

% secciones
x=a(1,2:m);
x=fliplr(x);

% alturas
z=a(2:n,1);

% semimangas
y=a(2:n,2:m);


% Creamos una figura
h=figure('Tag','lineas de Agua','Name',[blanks(6),'Líneas de
Agua'],'Position',[100 100 1000 600]);
axis off

% Centramos y escalamos
set( h,'Units', 'pixels' );
screenSize = get(0, 'ScreenSize');

position = get( h,'Position');
position(1) = (screenSize(3)-position(3))/2;
position(2) = (screenSize(4)-position(4))/2;
set( h,'Position', position );

% dibujamos las líneas de agua

for k=1:n-1
    y_linea = y(k,1:m-1);
```

```
    z_linea = ones(1,m-1)*z(k);
    plot3(x,y_linea,z_linea,'LineW',2)
    hold on
    plot3(x,-y_linea,z_linea,'LineW',2)
    hold on
end


% Ponemos las indicaciones de las variables

axis('image')
grid on
box on
zlabel('Calado [m]','FontW','Bold')
xlabel('Secciones [m]','FontW','Bold')
ylabel('Semimangas [m]','FontW','Bold')
title('Imagen 3D de las líneas de agua','FontW','Bold')
set(gca,'FontW','Bold')
```

10.4 Función "Dibujar Secciones"

Este programa dibuja las diferentes secciones de un barco, a partir de unos ciertos puntos de control que hemos de introducirle a partir de un archivo específico. Estas secciones se usarán para los cálculos del programa.

```
function dibujar_secciones(s,varargin)

% Este programa dibuja las diferentes secciones de un barco
% dados unos ciertos puntos de control por sección
% dibujar_secciones(s,varargin)
% Variables de entrada:
% s estructura que contiene la información de las secciones.
%   Sus campos son los siguientes:
%   s.x  posición espacial x de la sección
%      s.numero   número de puntos de control en la sección
considerada
%   s.puntos coordenadas (y,z) de los puntos de control
%   s.tipo tipo de poligonal entre un punto y el siguiente. Puede
%          ser paralela al eje y, paralela al eje z, u oblicua
%   s.poligonal información necesaria para construir la poligonal
%   s.limites indica de dónde a dónde se extiende cada línea de la
%             poligonal
% k argumento opcional que indica el número de sección que se
quiere
%   dibujar
% rango  vector [k1,k2] que indica desde que sección a qué sección
se
%   quiere dibujar
% all dibuja todas las secciones secuencialmente con una pausa
entre
%     ellas
% grupo,m,p  las dibuja agrupadas en grupos de m x p secciones
% superpuestas,[k1,k2]

% creamos una ventana nueva donde dibujar
h=figure('Tag','secciones','Name',[blanks(6),'Secciones'],'Positio
n',[100 100 1000 600]);
axis off

% Centramos y escalamos
set( h,'Units', 'pixels' );
screenSize = get(0, 'ScreenSize');

position = get( h,'Position');
position(1) = (screenSize(3)-position(3))/2;
position(2) = (screenSize(4)-position(4))/2;
set( h,'Position', position );


% número de secciones
n=length(s);
```

```
        % calculamos los valores mínimos y máximos de las variables y,z

        ym=min(s(1).puntos(:,1));
        yM=max(s(1).puntos(:,1));
        zm=min(s(1).puntos(:,2));
        zM=max(s(1).puntos(:,2));

        for i=2:n

            aux_ym=min(s(i).puntos(:,1));
            if aux_ym < ym
                ym=aux_ym;
            end
            aux_yM=max(s(i).puntos(:,1));
            if aux_yM>yM
                yM=aux_yM;
            end
            aux_zm=min(s(i).puntos(:,2));
            if aux_zm<zm
                zm=aux_zm;
            end
            aux_zM=max(s(i).puntos(:,2));
            if aux_zM>zM
                zM=aux_zM;
            end

        end

        dy=(yM-ym)/10;
        dz=(zM-zm)/10;

        if nargin==2

            in=varargin{1};

            if isnumeric(in)     % o una sección o un rango de secciones

                if length(in)==1   % es una sección

                    aux=s(in).puntos;
                    aux=[aux;aux(1,1:2)];
                    plot(aux(:,1),aux(:,2));
                    ym=min(aux(:,1));
                    yM=max(aux(:,1));
                    zm=min(aux(:,2));
                    zM=max(aux(:,2));
                    if yM~=ym
                        dy=(yM-ym)/10;
                    end
                    if zM~=zm
                        dz=(zM-zm)/10;
                    end
                    axis([ym-dy yM+dy zm-dz zM+dz]);
                    title(['Sección                          ',num2str(in),'
x=',num2str(s(in).x)])
                    xlabel('Eje y')
                    ylabel('Eje z')
```

```
            elseif length(in)==2    % es un rango de secciones

                for k=in(1):in(2)

                    aux=s(k).puntos;
                    aux=[aux;aux(1,1:2)];
                    plot(aux(:,1),aux(:,2));
                    ym=min(aux(:,1));
                    yM=max(aux(:,1));
                    zm=min(aux(:,2));
                    zM=max(aux(:,2));
                    if yM~=ym
                        dy=(yM-ym)/10;
                    end
                    if zM~=zm
                        dz=(zM-zm)/10;
                    end
                    axis([ym-dy yM+dy zm-dz zM+dz]);
                    title(['Sección                    ',num2str(k),'
x=',num2str(s(k).x)])
                    xlabel('Eje y')
                    ylabel('Eje z')
                    pause

                end
            end

        elseif ischar(in)   % todas las secciones


            for k=1:n

                aux=s(k).puntos;
                aux=[aux;aux(1,1:2)];
                plot(aux(:,1),aux(:,2));
                ym=min(aux(:,1));
                yM=max(aux(:,1));
                zm=min(aux(:,2));
                zM=max(aux(:,2));
                if yM~=ym
                    dy=(yM-ym)/10;
                end
                if zM~=zm
                    dz=(zM-zm)/10;
                end
                axis([ym-dy yM+dy zm-dz zM+dz]);
                title(['Sección ',num2str(k),'   x=',num2str(s(k).x)])
                xlabel('Eje y')
                ylabel('Eje z')
                pause

            end


        end

    elseif nargin==3

        in=varargin{2};
```

```
        for k=in(1):in(2)

            aux=s(k).puntos;
            aux=[aux;aux(1,1:2)];
            plot(aux(:,1),aux(:,2));
            hold on
            ym=min([ym,min(aux(:,1))]);
            yM=max([yM,max(aux(:,1))]);
            zm=min([zm,min(aux(:,2))]);
            zM=max([zM,max(aux(:,2))]);
            if yM~=ym
                dy=(yM-ym)/10;
            end
            if zM~=zm
                dz=(zM-zm)/10;
            end
            axis([ym-dy yM+dy zm-dz zM+dz]);
            title(['Sección ',num2str(k),'   x=',num2str(s(k).x)])
            xlabel('Eje y')
            ylabel('Eje z')
            pause

        end

    elseif nargin==4

        m=varargin{2};
        p=varargin{3};
        mp=m*p;

        co=floor(n/mp);           % número de plots a hacer menos 1
        re=rem(n,mp);              % número de secciones en el último
plot


        for np=1:co


            for k=1:mp

                indice=mp*(np-1)+k;
                aux=s(indice).puntos;
                aux=[aux;aux(1,1:2)];
                subplot(m,p,k);
                plot(aux(:,1),aux(:,2));
                ym=min(aux(:,1));
                yM=max(aux(:,1));
                zm=min(aux(:,2));
                zM=max(aux(:,2));
                if yM~=ym
                    dy=(yM-ym)/10;
                end
                if zM~=zm
                    dz=(zM-zm)/10;
                end
                axis([ym-dy yM+dy zm-dz zM+dz]);
                title(['Sección                    ',num2str(indice),'
x=',num2str(s(indice).x)])
```

```
                xlabel('Eje y')
                ylabel('Eje z')

            end

            pause


        end


        hold off
        close(h)
        % creamos una ventana nueva donde dibujar

h=figure('Tag','secciones','Name',[blanks(6),'Secciones'],'Position',[10
0 100 1000 600]);
        axis off


        for k=1:re


            indice=mp*co+k;
            aux=s(indice).puntos;
            aux=[aux;aux(1,1:2)];
            subplot(m,p,k);
            plot(aux(:,1),aux(:,2));
            ym=min(aux(:,1));
            yM=max(aux(:,1));
            zm=min(aux(:,2));
            zM=max(aux(:,2));
            if yM~=ym
                dy=(yM-ym)/10;
            end
            if zM~=zm
                dz=(zM-zm)/10;
            end
            axis([ym-dy yM+dy zm-dz zM+dz]);
            title(['Sección                          ',num2str(indice),'
x=',num2str(s(indice).x)])
            xlabel('Eje y')
            ylabel('Eje z')

        end


    end
```

10.5 Función "Dibujo barco grada"

Esta función dibuja el barco sobre la grada en el programa de "Proceso de la Botadura" si seleccionamos la opción dibujo en el apartado "Arfada y Giro".

Este dibujo está realizado para representar las celdas en las que el archivo excell tiene unos valores diferentes, depende del valor de cada celda, el programa colorea la celda de un color diferente siendo:

- **Blanco:** Zona en la que no hay barco ni agua.
- **Azul claro:** Zona en la que hay agua pero no hay barco.
- **Amarillo:** Zona que identifica la línea de agua.
- **Negro:** Zona donde hay barco pero no hay agua, es decir, barco no sumergido.
- **Azul marino:** Zona donde hay barco y agua, es decir, barco sumergido.

También están representados: el centro de gravedad de los pesos del barco como un punto verde y el centro de carena, o posición del centro de gravedad del agua desalojada, en rojo.

```
      function
dibujo_barco_grada(i0,xi0,alfa,semimangas,densidad,Xg,Zg)

      % Este programa dibuja de manera esquemática el barco con el que
se trabaja
      % a diferentes alturas de agua en la grada durante la botadura. Se
indica
      % la posición del centro de empuje
      %
      % dibujo_barco_grada(i0,xi0,alfa,semimangas,densidad,Xg,Zg,M);
      %
      % Variables de entrada:
      % i0 número de sección a la que llega el nivel del agua
      % xi0 coordenada x correspondiente o bien a la última sección
mojada o bien
      %     a una distancia superior a ella que indica el nivel del agua
      % alfa ángulo de inclinación de la grada
      % semimangas matriz que contiene las dimensiones x,y,z del barco
      % densidad del líquido donde se sumerge el barco
      %                densidad agua dulce=1
      %                densidad agua salada=1.025
      % Xg posición longitudinal del centro de pesos
      % Zg posición vertical del centro de pesos


      screenSize = get(0, 'ScreenSize');
      p1=screenSize(1)+50;
      p2=screenSize(2)+50;
```

```
p3=screenSize(3)-150;
p4=screenSize(4)-150;


[n,m]=size(semimangas);
x=semimangas(1,2:end);
z=semimangas(2:end,1);
y=semimangas(2:end,2:end);
[X,Z]=meshgrid(x,z);


cond1=(y~=0);
cond2=~(-tan(alfa)*(X-xi0)<Z);
M=ones(size(y));


yc=1*(cond1 & (~cond2)).*M+0.7*(cond1 & cond2).*M+ 0.4*((~cond1) &
cond2).*M+0*((~cond1) & (~cond2)).*M;


if isempty(find(yc==0.7))

    % dibujamos el barco sin mojar
    figure(1);     set(1,'Name','Posición     del     Centro     de
Empuje','Position',[p1 p2 p3 p4]);
    RGB1=[1,1,1; 0,1,1; 0,0,0];
    colormap(RGB1);
    pcolor(X,Z,yc);
    axis('image');
else

    % dibujamos el barco mojado
    figure(1);     set(1,'Name','Posición     del     Centro     de
Empuje','Position',[p1 p2 p3 p4]);
    RGB1=[1,1,1; 0,1,1; 0,0,1;0,0,0];
    colormap(RGB1);
    pcolor(X,Z,yc);
    axis('image');


end


% dibujamos el límite de la parte mojada

hold on
xr=0:1/100:xi0;
yr=-tan(alfa)*(xr-xi0);


plot(xr,yr,'y','LineWidth',3)

% dibujamos el centro de pesos

plot(Xg,Zg,'o','MarkerSize',7,'MarkerFaceColor','g');

% dibujamos el centro de empuje
```

```
      if i0>m-1
          [empuje,centro_empuje]=hidrostatica_alfa(m-
1,xi0,alfa,semimangas,densidad);
      else

[empuje,centro_empuje]=hidrostatica_alfa(i0,xi0,alfa,semimangas,densidad
);
      end

      plot(centro_empuje(1),centro_empuje(3),'ro','MarkerSize',5,'Marker
FaceColor','r')

      hold off
```

10.6 Función "Giro Arfada"

Esta función calcula si se produce o no arfada, y el momento en el que se produce el giro. Para ello va comparando los momentos que genera el barco, tanto el momento empuje como el momento peso, y si el momento empuje es mayor que el momento peso, el barco comienza el giro.

Si el barco comienza a sufrir arfada, el programa manda un mensaje de error. Esto se calcula en el programa comparando el momento arfada frente al momento empuje. Si en algún instante el momento arfada es superior al momento empuje, la botadura no es segura realizarla y lo mostrará el programa.

```
      function
[xgiro,arf,xarf]=giro_arfada(ficheroHidrostatica,ficheroLineasAgua,peso,
Xg,Zg,hang,dang,alfa,densidad,varargin)

      % Este programa calcula la posición longitudinal cuando el barco
empieza el
      % giro. Además determina si en algún momento anterior al giro se
producirá
      % el efecto indeseable de la arfada.
      %
      %
giro_arfada(ficheroHidrostatica,ficheroLineasAgua,peso,Xg,Zg,hang,dang,a
lfa,densidad,varargin);
      %
      % Variables de entrada:
      % ficheroHidrostatica  nombre del fichero excel que contiene los
cálculos
      %                          hidrostáticos del barco en posición de
adrizado
      % ficheroLineasAgua    nombre del fichero excel que contiene los
datos de las semimangas
      %                     para cada sección y calado
      % peso variable que indica el peso real del barco
      % Xg posición longitudinal del centro de pesos
      % Zg posición vertical del centro de pesos
      % hang altura de la anguila
      % dang distancia desde la proa de la anguila a la proa del barco
      % alfa inclinación de la grada
      % densidad del líquido donde se sumerge el barco
      %                densidad agua dulce=1
      %                densidad agua salada=1.025
      % varargin
      %           posibilidad '1', dg (distancia mojada de la grada),
      %            posibilidad '2', caladok (calado en el punto k de la
grada)
      %            dibujo si la variable vale 1 se mostrará la posición
del centro
      %           de empuje, si vale cero no
      %
      % Variables de salida:
```

```
      % xgiro  posición longitudinal donde comienza el giro del barco
      % arf       variable que indica si hay arfada (arf='1') o no
(arf='0')
      % xarf     en caso de que haya arfada indica en qué posición
longitudinal


      % Definimos el calado en el punto k y la distancia de grada mojada

      if varargin{1}=='1'
          dg=varargin{2};
          caladok=sin(alfa)*dg;
      elseif varargin{1}=='2'
          caladok=varargin{2};
          dg=caladok/sin(alfa);
      end

      if nargin~=12
          uiwait(msgbox('Faltan variables de entrada en el programa
giro_arfada.m','Revisar datos de entrada','error'));
          return;
      else
          dibujo=varargin{3};
      end

      % Lee las variables hidrostáticas calculadas

      numeric=xlsread(ficheroHidrostatica);
      [nh,mh]=size(numeric);

      % Lee las alturas de las líneas de agua, las secciones, y las
semimangas

      semimangas=xlsread(['./Líneas de Agua/',ficheroLineasAgua]);
      [n,m]=size(semimangas);

      % secciones, desde i0=1 hasta i0=m-1
      x=semimangas(1,2:m);

      % alturas
      z=semimangas(2:n,1);

      % semimangas
      y=semimangas(2:n,2:m);

      % inicializamos los valores de la arfada
      arf='0';
      xarf=-1;
      xgiro=-1;

      % Calculamos distancias para el cálculo del momento del empuje y
del
      % momento del peso

      d1=hang/sin(alfa);
      d3=dg*cos(alfa);
      d4p=Xg/cos(alfa);
```

```
      % Ampliamos x para que permita ir más allá de la eslora máxima

      xaux=[x,x(m-1)+0.5:0.5:z(1)/tan(alfa)+10];
      maux=length(xaux);


      % Bucle que va sumergiendo el barco deslizandolo sobre la grada,
subido
      % en la anguila

      for i0=1:maux

          % Comprobamos la condición de Arfada

          d2=xaux(i0)/cos(alfa);

          % calculamos el empuje y centro de empuje para una posición
          % determinada del barco en la botadura

          if i0>m-1
              [empuje,centro_empuje]=hidrostatica_alfa(m-
1,xaux(i0),alfa,semimangas,densidad);
          else

[empuje,centro_empuje]=hidrostatica_alfa(i0,xaux(i0),alfa,semimangas,den
sidad);
          end


          d4e=centro_empuje(1)/cos(alfa);
          de=d1+d2-d3-d4e;
          dp=d1+d2-d3-d4p;

          MomentoEmpuje=empuje*de;
          MomentoPeso=peso*dp;

          if MomentoPeso>MomentoEmpuje
              arf='1';
              xarf=xaux(i0);
              uiwait(msgbox('Peligro,      se      da      la      Arfada
','Arfada','error'));
              return;
          end

          % Comprobamos la condición de Hundimiento

          naux=round(0.25*(m-1));
          yaux=y(:,1:naux);
          ind=find(yaux'~=0);
          naux=ceil(ind(1)/naux);   % z(naux) contiene el calado máximo
en popa

          if (xaux(i0)*sin(alfa)+0.5)>=z(naux)

              uiwait(msgbox('Peligro, se hunde el barco por Inundación
','Inundación','error'));
              return;
```

```
            end

            % Calculamos donde se da el giro

            % obtenemos la eslora de flotación


[v]=interpolacion_variables_hidrostaticas(peso,ficheroHidrostatica);
            Lf=v(14);

            % distancia desde el centro de pesos longitudinal a la proa de
la anguila

            dpGiro=Lf-dang-Xg;

            % distancia desde el centro de empuje longitudinal a la proa
de la
            % anguila

            deGiro=Lf-dang-centro_empuje(1);

            % calculamos el momento del peso para el giro

            MomentoPesoGiro=peso*(dpGiro*cos(alfa)+(Zg+hang)*sin(alfa));

MomentoEmpujeGiro=empuje*(deGiro*cos(alfa)+(centro_empuje(3)+hang)*sin(a
lfa));



            if dibujo==1

dibujo_barco_grada(i0,xaux(i0),alfa,semimangas,densidad,Xg,Zg);
            end

            if MomentoEmpujeGiro>=MomentoPesoGiro

                xgiro=xaux(i0);
                IconoOk=imread('IconoOk.jpg');
                uiwait(msgbox(['El    giro    se    da    correctamente    a
x=',num2str(xgiro),blanks(3),'metros      de      popa'],'Punto      de
Giro','custom',IconoOk));
                return;
            end

        end
```

10.7 Función "Hidrostática"

Esta función incluye todos los comandos necesarios para la ejecución de la ventana donde se introducen los datos del barco para comenzar a realizar los cálculos y se señalan los datos a mostrar tras los cálculos realizados.

Para la introducción de datos, se necesita haber obtenido las formas en Rhinoceros y a partir de ellos se obtienen los valores de los datos hidrostáticos que se indican en el programa.

Todos estos cálculos se realizan a partir de las fórmulas típicas de la hidrostática naval y mediante la integración aproximada.

```
function varargout = hidrostatica(varargin)
% HIDROSTATICA M-file for hidrostatica.fig
%        HIDROSTATICA, by itself, creates a new HIDROSTATICA or
raises the existing
%      singleton*.
%
%       H = HIDROSTATICA returns the handle to a new HIDROSTATICA
or the handle to
%      the existing singleton*.
%
%           HIDROSTATICA('CALLBACK',hObject,eventData,handles,...)
calls the local
%       function named CALLBACK in HIDROSTATICA.M with the given
input arguments.
%
%            HIDROSTATICA('Property','Value',...) creates a new
HIDROSTATICA or raises the
%        existing singleton*.  Starting from the left, property
value pairs are
%        applied to the GUI before hidrostatica_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to hidrostatica_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 hidrostatica

% Last Modified by GUIDE v2.5 15-Jul-2014 19:41:32

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
```

```
    gui_State = struct('gui_Name',       mfilename, ...
        'gui_Singleton',  gui_Singleton, ...
        'gui_OpeningFcn', @hidrostatica_OpeningFcn, ...
        'gui_OutputFcn',  @hidrostatica_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 hidrostatica is made visible.
    function hidrostatica_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 hidrostatica (see VARARGIN)

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

    % Update handles structure
    guidata(hObject, handles);

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

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
    %%%%%%%%%%%%%%%    CENTRAR    INTERFAZ    GRÁFICA    EN    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 );



    % --- Outputs from this function are returned to the command line.
    function varargout = hidrostatica_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;


      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
      %%%%%%%%%%%%%%%%%     INCLUIMOS     LA     IMAGEN     DEL     BARCO
%%%%%%%%%%%%%%%%%%
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

      barco=imread('barco_interfaz.png');
      axes(handles.axes1);
      imshow(barco); axis off


      function edit_Conv_Callback(hObject, eventdata, handles)
      % hObject    handle to edit_Conv (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 edit_Conv as
text
      %          str2double(get(hObject,'String')) returns contents of
edit_Conv as a double


      % --- Executes during object creation, after setting all
properties.
      function edit_Conv_CreateFcn(hObject, eventdata, handles)
      % hObject    handle to edit_Conv (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 edit_LA_Callback(hObject, eventdata, handles)
      % hObject    handle to edit_LA (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 edit_LA as text
```

```
      %           str2double(get(hObject,'String')) returns contents of
edit_LA as a double


      %  ---  Executes  during  object  creation,  after  setting  all
properties.
      function edit_LA_CreateFcn(hObject, eventdata, handles)
      % hObject    handle to edit_LA (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 pushbutton_LA.
      function pushbutton_LA_Callback(hObject, eventdata, handles)
      % hObject    handle to pushbutton_LA (see GCBO)
      % eventdata   reserved - to be defined in a future version of
MATLAB
      % handles    structure with handles and user data (see GUIDATA)

      cd('Líneas de Agua');
      [filename,pathname,filterindex]=uigetfile( ...
          {  '*.xls',  'Archivo  que  contiene  las  líneas  de  agua
(*.xls)'},'Seleccione un archivo');
      cd ..

      if filterindex == 1
          set(handles.edit_LA,'String',filename);
      end


      function edit_S_Callback(hObject, eventdata, handles)
      % hObject    handle to edit_S (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 edit_S as text
      %           str2double(get(hObject,'String')) returns contents of
edit_S as a double


      %  ---  Executes  during  object  creation,  after  setting  all
properties.
      function edit_S_CreateFcn(hObject, eventdata, handles)
      % hObject    handle to edit_S (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 pushbutton_S.
        function pushbutton_S_Callback(hObject, eventdata, handles)
        % hObject    handle to pushbutton_S (see GCBO)
        % eventdata  reserved - to be defined in a future version of
MATLAB
        % handles    structure with handles and user data (see GUIDATA)

        cd('Secciones de Barcos')
        [filename,pathname,filterindex]=uigetfile( ...
            {  '*.gf',  'Archivo  que  contiene  las  secciones  del  barco
(*.gf)'},'Seleccione un archivo');
        cd ..

        if filterindex == 1
            set(handles.edit_S,'String',filename);
        end


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

        % leemos el fichero .gf

        fichero= get(handles.edit_S,'String');
        fichero=fliplr(deblank(fliplr(deblank(fichero))));     %Eliminamos
huecos en blanco

        % Comprobamos que no está vacío
        if isempty(fichero)==1
            uiwait(msgbox('Debe  introducir  un  fichero  .gf  con  las
secciones', 'Mensaje de error',...
                'error','modal'))
            return;
        end

        % Transformamos el fichero a .xls  convirtiendo las unidades a
metros

        % Leemos el factor de conversion

        aux = get(handles.edit_Conv,'String');
        conversion=sscanf(aux, '%f');

        % Comprobamos que no está vacío
        if isempty(conversion)==1 | conversion<=0
```

```
            uiwait(msgbox('Debe introducir un factor de conversión.',
'Mensaje de error',...
                'error','modal'))
            return;
        end

        % Eliminamos la extensión del archivo para poder llamar a la
función
        fichero_out= strtok(fichero,'.');

        fichero_out=[fichero_out,'.xls'];
        transformar_fichero(fichero,fichero_out,conversion);

        % Obtenemos las secciones

        s=obtener_secciones(fichero_out);

        % Obtenemos las líneas de agua

        % leemos cuántas líneas de agua se quiere considerar

        aux = get(handles.edit_NLA,'String');
        nl=round(sscanf(aux, '%f'));

        % Comprobamos que no está vacío
        if isempty(nl)==1
            uiwait(msgbox('Debe introducir el número de líneas de agua.',
'Mensaje de error',...
                'error','modal'))
            return;
        end

        % leemos el calado

        % número de secciones
        n=length(s);

        % calculamos el valor mínimo y máximo de la variable z

        zm=min(s(1).puntos(:,2));
        zM=max(s(1).puntos(:,2));

        for i=2:n

            aux_zm=min(s(i).puntos(:,2));
            if aux_zm<zm
                zm=aux_zm;
            end
            aux_zM=max(s(i).puntos(:,2));
            if aux_zM>zM
                zM=aux_zM;
            end

        end


        aux = get(handles.edit_T,'String');
        calado=sscanf(aux, '%f');
```

```
        % Comprobamos que no está vacío
        if isempty(calado)==1 | calado<zm
            str=['Debe introducir un valor para el calado entre
',num2str(zm),' y ',num2str(zM)];
            uiwait(msgbox(str, 'Mensaje de error',...
                'error','modal'))
            return;
        end

        if calado>zM
            calado=zM;
            uiwait(msgbox(['Se va a trabajar con el calado máximo
zM=',num2str(calado)], 'Mensaje de error',...
                'error','modal'));
            set(handles.edit_T,'String',num2str(calado));
        end

        [semimanga]=obtener_LineasDeAgua(s,nl,fichero_out,calado);


        % escribe en la interfaz el nombre del fichero que contiene las
líneas de
        % agua

        set(handles.edit_LA,'String',fichero_out);



        function edit_NLA_Callback(hObject, eventdata, handles)
        % hObject    handle to edit_NLA (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 edit_NLA as
text
        %        str2double(get(hObject,'String')) returns contents of
edit_NLA as a double


        % --- Executes during object creation, after setting all
properties.
        function edit_NLA_CreateFcn(hObject, eventdata, handles)
        % hObject    handle to edit_NLA (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 edit_T_Callback(hObject, eventdata, handles)
% hObject    handle to edit_T (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 edit_T as text
%        str2double(get(hObject,'String')) returns contents of edit_T as a double


% --- Executes during object creation, after setting all properties.
function edit_T_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_T (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 pushbutton_DLA.
function pushbutton_DLA_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_DLA (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% leemos el fichero .gf

fichero= get(handles.edit_LA,'String');
fichero=fliplr(deblank(fliplr(deblank(fichero))));     %Eliminamos huecos en blanco

% Comprobamos que no está vacío
if isempty(fichero)==1
    uiwait(msgbox('Debe introducir o generar un fichero .xls con las líneas de agua', 'Mensaje de error',...
        'error','modal'))
    return;
end

% llamamos a la función que dibuja las líneas de agua

dibujar_LineasDeAgua(fichero);


% --- 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


      if get(handles.radiobutton1,'Value')==1
          set(handles.edit_d1,'Enable','on');
          set(handles.radiobutton2,'Value',0);
          set(handles.edit_d2,'Enable','off');
          set(handles.edit_d3,'Enable','off');
          set(handles.radiobutton3,'Value',0);
          set(handles.radiobutton4,'Value',0);
          set(handles.edit_d4,'Enable','off');
          set(handles.edit_d4b,'Enable','off');
          set(handles.radiobutton5,'Value',0);
          set(handles.edit_d5,'Enable','off');
          set(handles.edit_d6,'Enable','off');
      else
          set(handles.edit_d1,'Enable','off');
      end




      function edit_d1_Callback(hObject, eventdata, handles)
      % hObject    handle to edit_d1 (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 edit_d1 as text
      %        str2double(get(hObject,'String')) returns contents of
edit_d1 as a double


      % --- Executes during object creation, after setting all
properties.
      function edit_d1_CreateFcn(hObject, eventdata, handles)
      % hObject    handle to edit_d1 (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 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

      if get(handles.radiobutton2,'Value')==1
          set(handles.radiobutton1,'Value',0);
          set(handles.edit_d1,'Enable','off');
          set(handles.edit_d2,'Enable','on');
          set(handles.edit_d3,'Enable','on');
          set(handles.radiobutton3,'Value',0);
          set(handles.radiobutton4,'Value',0);
          set(handles.edit_d4,'Enable','off');
          set(handles.edit_d4b,'Enable','off');
          set(handles.radiobutton5,'Value',0);
          set(handles.edit_d5,'Enable','off');
          set(handles.edit_d6,'Enable','off');
      else
          set(handles.edit_d2,'Enable','off');
          set(handles.edit_d3,'Enable','off')
      end




      function edit_d2_Callback(hObject, eventdata, handles)
      % hObject    handle to edit_d2 (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 edit_d2 as text
      %        str2double(get(hObject,'String')) returns contents of
edit_d2 as a double


      % --- Executes during object creation, after setting all
properties.
      function edit_d2_CreateFcn(hObject, eventdata, handles)
      % hObject    handle to edit_d2 (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 edit_d3_Callback(hObject, eventdata, handles)
      % hObject    handle to edit_d3 (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 edit_d3 as text
      %        str2double(get(hObject,'String')) returns contents of
edit_d3 as a double


      %  ---  Executes  during  object  creation,  after  setting  all
properties.
      function edit_d3_CreateFcn(hObject, eventdata, handles)
      % hObject    handle to edit_d3 (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 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


      if get(handles.radiobutton3,'Value')==1
          set(handles.radiobutton1,'Value',0);
          set(handles.edit_d1,'Enable','off');
          set(handles.radiobutton2,'Value',0);
          set(handles.edit_d2,'Enable','off');
          set(handles.edit_d3,'Enable','off');
          set(handles.radiobutton4,'Value',0);
          set(handles.edit_d4,'Enable','off');
          set(handles.edit_d4b,'Enable','off');
          set(handles.radiobutton5,'Value',0);
          set(handles.edit_d5,'Enable','off');
          set(handles.edit_d6,'Enable','off');
      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
```

```
    if get(handles.radiobutton4,'Value')==1
        set(handles.radiobutton1,'Value',0);
        set(handles.edit_d1,'Enable','off');
        set(handles.radiobutton2,'Value',0);
        set(handles.edit_d2,'Enable','off');
        set(handles.edit_d3,'Enable','off');
        set(handles.radiobutton3,'Value',0);
        set(handles.edit_d4,'Enable','on');
        set(handles.edit_d4b,'Enable','on');
        set(handles.radiobutton5,'Value',0);
        set(handles.edit_d5,'Enable','off');
        set(handles.edit_d6,'Enable','off');
    else
        set(handles.edit_d4,'Enable','off');
        set(handles.edit_d4b,'Enable','off');
    end




    function edit_d4_Callback(hObject, eventdata, handles)
    % hObject    handle to edit_d4 (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 edit_d4 as text
    %        str2double(get(hObject,'String')) returns contents of
edit_d4 as a double


    % --- Executes during object creation, after setting all
properties.
    function edit_d4_CreateFcn(hObject, eventdata, handles)
    % hObject    handle to edit_d4 (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 edit_d4b_Callback(hObject, eventdata, handles)
    % hObject    handle to edit_d4b (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 edit_d4b as
text
```

```
      %           str2double(get(hObject,'String')) returns contents of
edit_d4b as a double


      %  ---  Executes  during  object  creation,  after  setting  all
properties.
      function edit_d4b_CreateFcn(hObject, eventdata, handles)
      % hObject    handle to edit_d4b (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

      if get(handles.radiobutton5,'Value')==1
          set(handles.radiobutton1,'Value',0);
          set(handles.edit_d1,'Enable','off');
          set(handles.radiobutton2,'Value',0);
          set(handles.edit_d2,'Enable','off');
          set(handles.edit_d3,'Enable','off');
          set(handles.radiobutton3,'Value',0);
          set(handles.radiobutton4,'Value',0);
          set(handles.edit_d4,'Enable','off');
          set(handles.edit_d4b,'Enable','off');
          set(handles.edit_d5,'Enable','on');
          set(handles.edit_d6,'Enable','on');
      else
          set(handles.edit_d5,'Enable','off');
          set(handles.edit_d6,'Enable','off');
      end




      function edit_d5_Callback(hObject, eventdata, handles)
      % hObject    handle to edit_d5 (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 edit_d5 as text
```

```
%           str2double(get(hObject,'String')) returns contents of
edit_d5 as a double


%  ---  Executes  during  object  creation,  after  setting  all
properties.
function edit_d5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_d5 (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 edit_d6_Callback(hObject, eventdata, handles)
% hObject    handle to edit_d6 (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 edit_d6 as text
%           str2double(get(hObject,'String')) returns contents of
edit_d6 as a double


%  ---  Executes  during  object  creation,  after  setting  all
properties.
function edit_d6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_d6 (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 pushbutton_dS.
function pushbutton_dS_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_dS (see GCBO)
% eventdata   reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
      % leemos el fichero .gf

      fichero= get(handles.edit_S,'String');
      fichero=fliplr(deblank(fliplr(deblank(fichero))));      %Eliminamos
huecos en blanco

      % Comprobamos que no está vacío
      if isempty(fichero)==1
          uiwait(msgbox('Debe introducir un fichero .gf con las
secciones', 'Mensaje de error','...
              'error','modal'))
          return;
      end

      % Transformamos el fichero a .xls

      % Leemos el factor de conversion

      aux = get(handles.edit_Conv,'String');
      conversion=sscanf(aux, '%f');

      % Comprobamos que no está vacío
      if isempty(conversion)==1 | conversion<=0
          uiwait(msgbox('Debe introducir un factor de conversión.',
'Mensaje de error','...
              'error','modal'))
          return;
      end

      % Eliminamos la extensión del archivo para poder llamar a la
función
      fichero_out= strtok(fichero,'.');

      fichero_out=[fichero_out,'.xls'];
      transformar_fichero(fichero,fichero_out,conversion);

      % Obtenemos las secciones

      s=obtener_secciones(fichero_out);

      % Miramos qué opción de dibujo hay marcada y dibujamos las
secciones

      if get(handles.radiobutton1,'Value')==1

          aux = get(handles.edit_d1,'String');
          k=round(sscanf(aux,'%f'));

          % Comprobamos que no está vacío
          if isempty(k)==1 | k==0 | k > length(s)
              str=['Debe introducir el número de sección que quiere
dibujar, k <= ',num2str(length(s))];
              uiwait(msgbox(str, 'Mensaje de error','error','modal'))
              return;
          end
          dibujar_secciones(s,k);
```

```
    elseif get(handles.radiobutton2,'Value')==1

        aux1 = get(handles.edit_d2,'String');
        k1=round(sscanf(aux1,'%f'));

        % Comprobamos que no está vacío
        if isempty(k1)==1 | k1==0 | k1 > length(s)
            str=['Debe introducir un número correcto para la sección
inicial, k1 <= ',num2str(length(s))];
            uiwait(msgbox(str, 'Mensaje de error','error','modal'))
            return;
        end

        aux2 = get(handles.edit_d3,'String');
        k2=round(sscanf(aux2,'%f'));

        % Comprobamos que no está vacío
        if isempty(k2)==1 | k2<=k1 | k2 > length(s)
            str=['Debe introducir un número correcto para la sección
final, k1 <= k2, k2<= ',num2str(length(s))];
            uiwait(msgbox(str, 'Mensaje de error','error','modal'))
            return;
        end

        dibujar_secciones(s,[k1,k2]);

    elseif get(handles.radiobutton3,'Value')==1

        dibujar_secciones(s,'all');

    elseif get(handles.radiobutton4,'Value')==1

        aux1 = get(handles.edit_d4,'String');
        m=round(sscanf(aux1,'%f'));

        % Comprobamos que no está vacío
        if isempty(m)==1 | m<=0 | m > 5
            str=['Debe introducir un número correcto para m, m<=5 '];
            uiwait(msgbox(str, 'Mensaje de error','error','modal'))
            return;
        end

        aux2 = get(handles.edit_d4b,'String');
        p=round(sscanf(aux2,'%f'));

        % Comprobamos que no está vacío
        if isempty(p)==1 | p<=0 | p > 5
            str=['Debe introducir un número correcto para p, p<=5'];
            uiwait(msgbox(str, 'Mensaje de error','error','modal'))
            return;
        end

        dibujar_secciones(s,'grupo',m,p);

    elseif get(handles.radiobutton5,'Value')==1

        aux1 = get(handles.edit_d5,'String');
        k1=round(sscanf(aux1,'%f'));
```

```
            % Comprobamos que no está vacío
            if isempty(k1)==1 | k1==0 | k1 > length(s)
                str=['Debe introducir un número correcto para la sección
inicial, k1 <= ',num2str(length(s))];
                uiwait(msgbox(str, 'Mensaje de error','error','modal'))
                return;
            end

            aux2 = get(handles.edit_d6,'String');
            k2=round(sscanf(aux2,'%f'));

            % Comprobamos que no está vacío
            if isempty(k2)==1 | k2<=k1 | k2 > length(s)
                str=['Debe introducir un número correcto para la sección
final, k1 <= k2, k2<= ',num2str(length(s))];
                uiwait(msgbox(str, 'Mensaje de error','error','modal'))
                return;
            end

            dibujar_secciones(s,'superpuestas',[k1,k2]);

        end


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

        open('./pdf/Área de la sección.pdf');


        % --- Executes on button press in checkbox_As.
        function checkbox_As_Callback(hObject, eventdata, handles)
        % hObject    handle to checkbox_As (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 checkbox_As


        if get(handles.checkbox_As,'Value')==1
            set(handles.edit_NS,'Enable','on');
        else
            set(handles.edit_NS,'Enable','off');
        end

        function edit_NS_Callback(hObject, eventdata, handles)
        % hObject    handle to edit_NS (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 edit_NS as text
```

```
      %           str2double(get(hObject,'String')) returns contents of
edit_NS as a double


      %  ---  Executes  during  object  creation,  after  setting  all
properties.
      function edit_NS_CreateFcn(hObject, eventdata, handles)
      % hObject    handle to edit_NS (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 checkbox_Aw.
      function checkbox_Aw_Callback(hObject, eventdata, handles)
      % hObject    handle to checkbox_Aw (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 checkbox_Aw



      % --- Executes on button press in checkbox_Mx.
      function checkbox_Mx_Callback(hObject, eventdata, handles)
      % hObject    handle to checkbox_Mx (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 checkbox_Mx


      % --- Executes on button press in checkbox_Xf.
      function checkbox_Xf_Callback(hObject, eventdata, handles)
      % hObject    handle to checkbox_Xf (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 checkbox_Xf



      % --- Executes on button press in checkbox_It.
      function checkbox_It_Callback(hObject, eventdata, handles)
      % hObject    handle to checkbox_It (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 checkbox_It


      % --- Executes on button press in checkbox_BMt.
      function checkbox_BMt_Callback(hObject, eventdata, handles)
      % hObject    handle to checkbox_BMt (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 checkbox_BMt


      % --- Executes on button press in checkbox_Iy.
      function checkbox_Iy_Callback(hObject, eventdata, handles)
      % hObject    handle to checkbox_Iy (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 checkbox_Iy


      % --- Executes on button press in checkbox_Il.
      function checkbox_Il_Callback(hObject, eventdata, handles)
      % hObject    handle to checkbox_Il (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 checkbox_Il


      % --- Executes on button press in checkbox_BMl.
      function checkbox_BMl_Callback(hObject, eventdata, handles)
      % hObject    handle to checkbox_BMl (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 checkbox_BMl



      % --- Executes on button press in checkbox_TPC.
      function checkbox_TPC_Callback(hObject, eventdata, handles)
      % hObject    handle to checkbox_TPC (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 checkbox_TPC
```

```
% --- Executes on button press in checkbox_V.
function checkbox_V_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox_V (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 checkbox_V


% --- Executes on button press in checkbox_ZB.
function checkbox_ZB_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox_ZB (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 checkbox_ZB


% --- Executes on button press in checkbox_MB.
function checkbox_MB_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox_MB (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 checkbox_MB


% --- Executes on button press in checkbox_XB.
function checkbox_XB_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox_XB (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 checkbox_XB



% --- Executes on button press in pushbutton_Aw.
function pushbutton_Aw_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_Aw (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
open('./pdf/Área de flotación.pdf');



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

```
        open('./pdf/Momento de inercia de flotación.pdf');


        % --- Executes on button press in pushbutton_Xf.
        function pushbutton_Xf_Callback(hObject, eventdata, handles)
        % hObject    handle to pushbutton_Xf (see GCBO)
        % eventdata  reserved - to be defined in a future version of
MATLAB
        % handles    structure with handles and user data (see GUIDATA)
        open('./pdf/Centro de flotación.pdf');


        % --- Executes on button press in pushbutton_It.
        function pushbutton_It_Callback(hObject, eventdata, handles)
        % hObject    handle to pushbutton_It (see GCBO)
        % eventdata  reserved - to be defined in a future version of
MATLAB
        % handles    structure with handles and user data (see GUIDATA)
        open('./pdf/Momento de inercia transversal.pdf');


        % --- Executes on button press in pushbutton_BMt.
        function pushbutton_BMt_Callback(hObject, eventdata, handles)
        % hObject    handle to pushbutton_BMt (see GCBO)
        % eventdata  reserved - to be defined in a future version of
MATLAB
        % handles    structure with handles and user data (see GUIDATA)
        open('./pdf/BMt Radio metacéntrico transversal.pdf');

        % --- Executes on button press in pushbutton_Iy.
        function pushbutton_Iy_Callback(hObject, eventdata, handles)
        % hObject    handle to pushbutton_Iy (see GCBO)
        % eventdata  reserved - to be defined in a future version of
MATLAB
        % handles    structure with handles and user data (see GUIDATA)
        open('./pdf/Momento de inercia de flotación.pdf');



        % --- Executes on button press in pushbutton_Il.
        function pushbutton_Il_Callback(hObject, eventdata, handles)
        % hObject    handle to pushbutton_Il (see GCBO)
        % eventdata  reserved - to be defined in a future version of
MATLAB
        % handles    structure with handles and user data (see GUIDATA)
        open('./pdf/Momento de inercia longitudinal.pdf');


        % --- Executes on button press in pushbutton_BMl.
        function pushbutton_BMl_Callback(hObject, eventdata, handles)
        % hObject    handle to pushbutton_BMl (see GCBO)
        % eventdata  reserved - to be defined in a future version of
MATLAB
        % handles    structure with handles and user data (see GUIDATA)
        open('./pdf/BMl Radio metacéntrico longitudinal.pdf');

        % --- Executes on button press in pushbutton_TPC.
        function pushbutton_TPC_Callback(hObject, eventdata, handles)
        % hObject    handle to pushbutton_TPC (see GCBO)
```

```
      % eventdata   reserved - to be defined in a future version of
MATLAB
      % handles    structure with handles and user data (see GUIDATA)
      open('./pdf/Toneladas por centímetro de inmersión.pdf');



      % --- Executes on button press in pushbutton_V.
      function pushbutton_V_Callback(hObject, eventdata, handles)
      % hObject    handle to pushbutton_V (see GCBO)
      % eventdata   reserved - to be defined in a future version of
MATLAB
      % handles    structure with handles and user data (see GUIDATA)
      open('./pdf/Volumen de carena.pdf');


      % --- Executes on button press in pushbutton_ZB.
      function pushbutton_ZB_Callback(hObject, eventdata, handles)
      % hObject    handle to pushbutton_ZB (see GCBO)
      % eventdata   reserved - to be defined in a future version of
MATLAB
      % handles    structure with handles and user data (see GUIDATA)
      open('./pdf/ZB Centro vertical de carena.pdf');


      % --- Executes on button press in pushbutton_MB.
      function pushbutton_MB_Callback(hObject, eventdata, handles)
      % hObject    handle to pushbutton_MB (see GCBO)
      % eventdata   reserved - to be defined in a future version of
MATLAB
      % handles    structure with handles and user data (see GUIDATA)
      open('./pdf/Momento de desplazamiento.pdf');



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

      open('./pdf/XB Coordenada horizontal del centro de carena.pdf');

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


      % leemos el fichero .xls

      fichero= get(handles.edit_LA,'String');
      fichero=fliplr(deblank(fliplr(deblank(fichero))));      %Eliminamos
huecos en blanco
```

```
        % Comprobamos que no está vacío
        if isempty(fichero)==1
            uiwait(msgbox('Debe introducir o generar un fichero .xls con
las líneas de agua', 'Mensaje de error',...
                'error','modal'))
            return;
        end

        % Mira en los checkbox a ver qué gráficas tiene que dibujar

        if get(handles.checkbox_Aw,'Value')==1
            op(1)='s';
        else
            op(1)='n';
        end
        if get(handles.checkbox_Mx,'Value')==1
            op(2)='s';
        else
            op(2)='n';
        end
        if get(handles.checkbox_Xf,'Value')==1
            op(3)='s';
        else
            op(3)='n';
        end
        if get(handles.checkbox_It,'Value')==1
            op(4)='s';
        else
            op(4)='n';
        end
        if get(handles.checkbox_Iy,'Value')==1
            op(5)='s';
        else
            op(5)='n';
        end
        if get(handles.checkbox_Il,'Value')==1
            op(6)='s';
        else
            op(6)='n';
        end
        if get(handles.checkbox_TPC,'Value')==1
            op(7)='s';
        else
            op(7)='n';
        end
        if get(handles.checkbox_V,'Value')==1
            op(8)='s';
        else
            op(8)='n';
        end
        if get(handles.checkbox_MB,'Value')==1
            op(9)='s';
        else
            op(9)='n';
        end
        if get(handles.checkbox_ZB,'Value')==1
            op(10)='s';
        else
            op(10)='n';
        end
        if get(handles.checkbox_BMt,'Value')==1
```

```
            op(11)='s';
        else
            op(11)='n';
        end
        if get(handles.checkbox_BMl,'Value')==1
            op(12)='s';
        else
            op(12)='n';
        end
        if get(handles.checkbox_As,'Value')==1
            op(13)='s';
            % lee el número de sección del que quiere calcular el área
            aux= get(handles.edit_NS,'String');
            ksec=round(sscanf(aux,'%f'));

            auxA=xlsread(['./Líneas de Agua/',fichero]);
            [n,m]=size(auxA);

            % Comprobamos que no está vacío
            if isempty(ksec)==1 | ksec<1 | ksec>m-1
                str=['Debe introducir un número correcto para la sección,
<=',num2str(m-1)];
                uiwait(msgbox(str, 'Mensaje de error','error','modal'))
                return;
            end



        else
            op(13)='n';
            ksec=1;
        end
        if get(handles.checkbox_XB,'Value')==1
            op(14)='s';
        else
            op(14)='n';
        end
        % Realizamos los cálculos hidrostáticos

        curvas_hidrostaticas(fichero,op,ksec);

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


        % Cerramos las gráficas
        allPlots=findall(0,'Type','figure','FileName',[]);
        delete(allPlots);

        % Ponemos a 1 el cambio de medida a metros
        set(handles.edit_Conv,'String','1');

        % Ponemos en blanco los edit activos
        set(handles.edit_S,'String','');
        set(handles.edit_LA,'String','');
```

```
        set(handles.edit_NLA,'String','');
        set(handles.edit_T,'String','');

        % Ponemos en blanco y desactivados el resto de edit
        set(handles.edit_d1,'String','');
        set(handles.edit_d1,'Enable','off');
        set(handles.edit_d2,'String','');
        set(handles.edit_d2,'Enable','off');
        set(handles.edit_d3,'String','');
        set(handles.edit_d3,'Enable','off');
        set(handles.edit_d4,'String','');
        set(handles.edit_d4,'Enable','off');
        set(handles.edit_d4b,'String','');
        set(handles.edit_d4b,'Enable','off');
        set(handles.edit_d5,'String','');
        set(handles.edit_d5,'Enable','off');
        set(handles.edit_d6,'String','');
        set(handles.edit_d6,'Enable','off');
        set(handles.edit_NS,'String','');
        set(handles.edit_NS,'Enable','off');

        % Desactivamos los radiobutton
        set(handles.radiobutton1,'Value',0);
        set(handles.radiobutton2,'Value',0);
        set(handles.radiobutton3,'Value',0);
        set(handles.radiobutton4,'Value',0);
        set(handles.radiobutton5,'Value',0);

        % Desactivamos los checkbox
        set(handles.checkbox_Aw,'Value',0);
        set(handles.checkbox_Mx,'Value',0);
        set(handles.checkbox_Xf,'Value',0);
        set(handles.checkbox_It,'Value',0);
        set(handles.checkbox_BMt,'Value',0);
        set(handles.checkbox_Iy,'Value',0);
        set(handles.checkbox_Il,'Value',0);
        set(handles.checkbox_BMl,'Value',0);
        set(handles.checkbox_TPC,'Value',0);
        set(handles.checkbox_V,'Value',0);
        set(handles.checkbox_ZB,'Value',0);
        set(handles.checkbox_MB,'Value',0);
        set(handles.checkbox_As,'Value',0);
        set(handles.checkbox_XB,'Value',0);




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

        % Cerramos las gráficas
        allPlots=findall(0,'Type','figure','FileName',[]);
        delete(allPlots);
```

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


      salir=questdlg('¿Desea salir del programa?','Salida del
Programa','Si','No','No');
      switch salir
          case 'Si'
              close
          case 'No'
              return;
      end


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


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

      open('./pdf/Acerca De.pdf');

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

      open('./pdf/Tutorial de la Interfaz Gráfica.pdf');

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

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

      open('./pdf/Proyecto Fin de Carrera. Pablo Belmonte
Rodriguez.pdf');

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

      salir=questdlg('¿Desea salir del programa?','Salida del
Programa','Si','No','No');
      switch salir
          case 'Si'
              close
          case 'No'
              return;
      end
```

10.8 Función "Hidrostática alfa"

Esta función calcula los datos de empuje y centro de empuje cuando el barco está inclinado. Para ello, los cálculos se realizan con el barco en posición horizontal y el agua inclinada, se realizan los cálculos introduciendo el agua en el barco, y no como en la realidad. Eso se representa en los dibujos que se pueden ver en el programa.

Los datos que obtenemos son imprescindibles para saber cuándo y cómo flota el barco libremente, y cuando comienza el giro. El cálculo se puede entender orientativamente en la siguiente imagen:

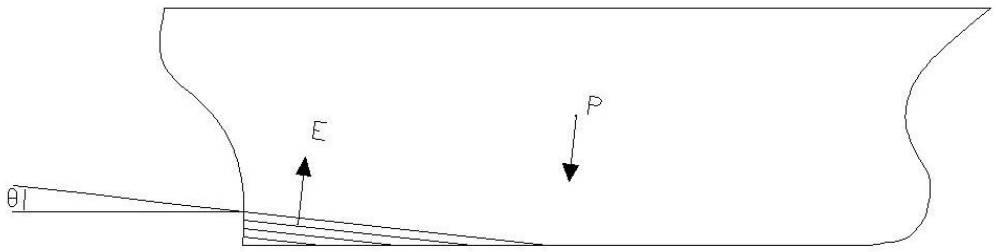

Imagen 103. Integración inclinada

```
      function
[empuje,centro_empuje]=hidrostatica_alfa(i0,xi0,alfa,semimangas,densidad
)

      % Esta función calcula el empuje (volumen por densidad) y el
centro de
      % empuje (centro de gravedad de la parte sumergida) de un barco
que está
      % siendo botado
      %
[empuje,centro_empuje]=hidrostatica_alfa(i0,xi0,alfa,semimangas,densidad
)
      % Variables de entrada:
      % i0 índice de la coordenada x0 de la última sección mojada
      % xi0 coordenada x correspondiente o bien a la última sección
mojada o bien
      %     a una distancia superior a ella que indica el nivel del agua
      % alfa ángulo de inclinación de la grada
      % semimangas matriz que contiene los datos de cada linea de agua
      % densidad del líquido donde se sumerge el barco
      %                densidad agua dulce=1
```

```
      %                densidad agua salada=1.025
      % Variables de salida:
      % empuje que se calcula como el volumen de la parte sumergida
multiplicado
      %        por la densidad del agua
      % centro_empuje   centro de gravedad de la parte sumergida del
barco


      % Tamaño de la matriz de datos que contiene las líneas de agua
      [n,m]=size(semimangas);

      % secciones
      x=semimangas(1,2:m);     % tamaño 1 x m-1

      % alturas
      z=semimangas(2:n,1);     % tamaño n-1 x 1

      % semimangas
      y=semimangas(2:n,2:m);   % tamaño n-1 x m-1


      % Realiza el cálculo de los parámetros hidrostáticos de cada uno
de los
      % calados

      for k=1:n-1

          % Determinamos la longitud de la semimanga mojada dependiendo
de la altura z
          y_linea=[];
          indicex=1;
          while -tan(alfa)*(x(indicex)-xi0)>=z(k) & indicex<i0
              y_linea=[y_linea,y(k,indicex)];
              indicex=indicex+1;
          end

          % Calculamos el área sumergida a altura z
          if isempty(y)
              Aw(k)=0;
          else
              Aw(k)=2*trapecios(x(1:length(y_linea)),y_linea);
          end

      end

      % Calculamos el volumen mojado y el centro de gravedad de dicho
volumen

      V=trapecios(z(1:n-1)',Aw(1:n-1));    % Volumen de desplazamiento
      MB=trapecios(z(1:n-1)',z(1:n-1)'.*Aw(1:n-1));      %   Momento   de
desplazamiento respecto de z
      cgz=MB/V;                            % Coordenada z del centro de
carena


      % Realiza el cálculo de los parámetros hidrostáticos de cada uno
de las secciones
```

```
      for i=1:i0

          % Determinamos la longitud de la semimanga mojada dependiendo
de la
          % sección x
          y_linea=[];
          indicez=n-1;
          while z(indicez)<=-tan(alfa)*(x(i)-xi0)
              y_linea=[y_linea,y(indicez,i)];
              indicez=indicez-1;
          end

          % Calculamos el área sumergida para cada sección
          if isempty(y_linea)
              As(i)=0;
          else

As(i)=2*trapecios(z(indicez+1:indicez+length(y_linea))',y_linea);
          end
      end

      MBx=trapecios(x(1:i0),x(1:i0).*As(1:i0));                          %
Momento respecto de la sección
      Vx=trapecios(x(1:i0),As(1:i0));                    % Volumen de
desplazamiento
      cgx=MBx/Vx;                                    % Coordenada x del
centro de empuje


      % Expresamos el empuje y el centro de empuje
      empuje=densidad*V;
      centro_empuje=[cgx,0,cgz];
```

10.9 Función "Interpolación variables"

Esta función se necesita para obtener los datos hidrostáticos cuando los datos que se introducen no son iguales a los de la tabla de datos que tiene el archivo. Se generan unos datos a partir de los que hay en la tabla para poder realizar los cálculos. Estos datos se obtienen por interpolación.

El dato a introducir es el peso, y el programa interpolará entre los pesos que tiene y el peso que se ha introducido.

```
function [v]=interpolacion_variables_hidrostaticas(peso,fichero)

% Esta función interpola linealmente los valores de las variables
% hidrostáticas correspondientes a un peso dado, que entra como
% variable de entrada en la columna de desplazamiento.
% Las variables hidrostáticas consideradas son:
% Calado
% Aw área de la flotación
% TCI toneladas por centímetro de inmersión
% Xf coordenada longitudinal del centro de flotación
% V volumen de carena
% D desplazamiento
% KB coordenada vertical del centro de carena
% XB coordenada longitudinal del centro de carena
% BMt radio metacéntrico transversal
% BMl radio metacéntrico longitudinal
% MTC momento para alterar el trimado un centímetro
%
% [v]=interpolacion_variables_hidrostaticas(peso,fichero)
% Variables de entrada:
% peso del barco, que se supone conocido
% fichero nombre del fichero que contiene los valores de las
%         variables hidrostáticas a diferentes calados
%
% Variables de salida:
% v vector que contiene los valores interpolados de las diferentes
%   variables hidrostáticas consideradas


% Lee las variables hidrostáticas calculadas

numeric=xlsread(fichero);
[n,m]=size(numeric);

% Introduce el peso en la columna de desplazamiento y calcula
entre que
% dos valores de desplazamiento se encuentra el peso

ind=find(numeric(:,6)<peso);
ind_sup=ind(1)-1;
```

```
        ind_inf=ind(1);

        % Interpolamos cada una de las variables

        v=numeric(ind_inf,:)+((numeric(ind_sup,:)-numeric(ind_inf,:))...
            /(numeric(ind_sup,6)-numeric(ind_inf,6)))*(peso-
numeric(ind_inf,6));
```

10.10 Función "Momento en los santos de proa"

Esta función calcula el momento en los santos de proa, el cual se realiza multiplicando el empuje con la distancia desde el apoyo de la proa de la anguila. Este dato es necesario para saber si la estructura de los santos de proa soportará el esfuerzo del barco en el giro.

```
function [Ms]=MomentoEnLosSantosProa(v,hang,dang,Xg,Zg,alfa)


% Esta función te calcula el momento que se produce en los santos
de
% proa (parte inferior de proa de la anguila). Esto sirve para
saber
% si el material de construcción es válido o no
%
% [Ms]=MomentoEnLosSantosProa(v,hang,dang,Xg,Zg,alfa)
%
% Variables de entrada:
% v datos hidrostáticos correspondientes a la flotación
% hang altura de la anguila
% dang distancia desde la proa de la anguila a la proa del barco
% Xg coordenada longitudinal del centro de pesos
% Zg coordenada vertical del centro de pesos
% alfa inclinación de la grada
%
% Variables de salida:
% Ms momento en los santos de proa


% Obtenemos el peso

peso=v(6);

% Obtenemos la eslora de flotación

Lf=v(14);

% Distancia desde el centro de pesos longitudinal a la proa de la
anguila

dp=Lf-dang-Xg;


% Calculamos el momento en los santos de proa

Ms=peso*(dp*cos(alfa)+(Zg+hang)*sin(alfa));
```

10.11 Función "Obtener calados"

Esta función calcula los calados en proa y en popa en el momento en el que el barco flota libremente a partir de los datos hidrostáticos calculados en el programa "Valores Hidrostáticos en Adrizado".

Estos datos son utilizados para saber cuándo flotará libremente el barco en el momento de la botadura y cuando comienza el giro.

El proceso de cálculo es: tras introducir un peso, el programa interpola entre los pesos dato que tiene, genera una interpolación entre los datos de calado que tiene para esos calados y obtiene el calado que tendría el barco, interpolando todos los datos hidrostáticos del barco.

```
function [Tpr, Tpp, beta]=obtencion_calados(fichero,v,Xg)

% Este programa calcula los calados de proa y de popa
% basándose en los cálculos hidrostáticos hechos en adrizado
%
% [Tpr, Tpp, beta]=obtencion_calados(fichero,v,Xg)
% Variables de entrada:
% fichero archivo excel que contiene los valores hidrostáticos
necesarios
%         a diferentes calados
% v  vector que contiene los valores hidrostáticos necesarios para
%    el calado de flotación
% Xg coordenada x del centro de gravedad del barco real
% Variables de salida:
% Tpr calado en proa
% Tpp calado en popa
% beta ángulo de inclinación del barco en flotación


% Cálculo del trimado

D=v(6);                % desplazamiento en calado de flotación
MTC=v(11);             % momento para el trimado de un centímetro
XB=v(8);                  % coordenada x del centro del centro de
carena
Xf=v(4);               % coordenada x del centro de flotación

t=D*abs(Xg-XB)/MTC/100;

% Cálculo del ángulo beta de giro del barco hasta la posición de
equilibrio

L=v(14);              % eslora en calado de flotación
beta=atan(t/L);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calado de proa y de popa
%%%%%%%%%%%%%%%%%%%%%%%%%%%%



% Bucles para calcular las distancias df y da


% Lee las variables hidrostáticas calculadas

numeric=xlsread(fichero);
[n,m]=size(numeric);

calados=numeric(:,1);

ind=find(calados<v(1));
ind=ind(1);

if Xg>XB

    % Bucle para calcular la distancia df

    i=ind-1;
    calado=v(1);
    calado_recta=calado+1;

    while calado_recta>calado &  i>=1
        xmax=numeric(i,13);
        calado_recta=v(1)+tan(beta)*(xmax-Xf);
        calado=calados(i);
        i=i-1;
    end

    xdf=(numeric(i+2,13)+xmax)/2;


    df=xdf-Xf;


    % Bucle para calcular la distancia da

    i=ind+2;
    calado=v(1);
    calado_recta=calado-1;

    while calado_recta<calado &  i<=length(calados)
        xmin=numeric(i,12);
        calado_recta=v(1)+tan(beta)*(xmin-Xf);
        calado=calados(i);
        i=i+1;
    end

    xda=(numeric(i-2,12)+xmin)/2;
```

```
    da=Xf-xda;

    % Calado de proa

    f=t/(da/df+1);
    Tpr=v(1)+f;

    % Calado de popa
    a=t-f;
    Tpp=v(1)-a;


elseif Xg<XB

    % Bucle para calcular la distancia da

    i=ind-1;
    calado=v(1);
    calado_recta=calado+1;

    while calado_recta>calado &  i>=1
        xmin=numeric(i,12);
        calado_recta=v(1)+tan(beta)*(xmin-Xf);
        calado=calados(i);
        i=i-1;
    end

    xda=(numeric(i+2,12)+xmin)/2;


    da=Xf-xda;


    % Bucle para calcular la distancia df

    i=ind+2;
    calado=v(1);
    calado_recta=calado-1;

    while calado_recta<calado &  i<=length(calados)
        xmax=numeric(i,13);
        calado_recta=v(1)+tan(beta)*(xmax-Xf);
        calado=calados(i);
        i=i+1;
    end

    xdf=(numeric(i-2,13)+xmax)/2;


    df=xdf-Xf;

    % Calado de proa

    f=t/(da/df+1);
    Tpr=v(1)-f;

    % Calado de popa
```

```
    a=t-f;
    Tpp=v(1)+a;

else

    Tpr=v(1);

    Tpp=Tpr;
end
```

10.12 Función "Obtener líneas de agua"

Esta función calcula para cada z la altura mínima y la altura máxima del barco, y de ahí se obtienen las líneas de agua del barco.

Además se encarga de mostrar el mensaje de error cuando el calado que hemos indicado no es adecuado para el barco, comprobando el calado máximo del barco e indica el máximo calado posible.

Si en el proceso de cálculo ya se ha realizado antes ese proceso, borra el proceso anterior y lo sobrescribe con el mismo nombre.

El fichero de salida que usa es un archivo excell .xls

```
function [semimanga]=obtener_LineasDeAgua(s,nl,fichero,varargin)

% Esta función calcula para cada z entre la altura mínima y la
altura
% máxima del barco (o el calado máximo introducido)
% tantas líneas de agua como sean necesarias
% [z,semimanga]=obtener_LineasDeAgua(s,nl,fichero,varargin)
% Variables de entrada
% s estructura que contiene la información de las secciones.
%   Sus campos son los siguientes:
%   s.x  posición espacial x de la sección
%      s.numero    número de puntos de control en la sección
considerada
%   s.puntos coordenadas (y,z) de los puntos de control
%   s.tipo tipo de poligonal entre un punto y el siguiente. Puede
%          ser paralela al eje y, paralela al eje z, u oblicua
%   s.poligonal información necesaria para construir la poligonal
%   s.limites indica de dónde a dónde se extiende cada línea de la
%             poligonal
% nl número de líneas de agua que se quieren calcular
% fichero nombre del fichero donde se quieren guardar las lineas
de agua
% calado valor máximo de la altura para el que se quieren realizar
%        los cálculos
% Variables de salida:
% semimanga contiene lo siguiente:
%           En la posición (1,1) un cero que no indica nada
%            En la primera fila, a partir de la segunda celda, la
coordenada
%               x de las cuadernas
%            En la primera columna, a partir de la segunda celda,
la
%           coordenada z que indica la altura
%            En las celdas (2:end, 2:end)  las semimangas para los
valores
```

```
%               de x y de z correspondientes



% número de secciones
n=length(s);

% calculamos el valor mínimo y máximo de la variable z

zm=min(s(1).puntos(:,2));
zM=max(s(1).puntos(:,2));

for i=2:n

    aux_zm=min(s(i).puntos(:,2));
    if aux_zm<zm
        zm=aux_zm;
    end
    aux_zM=max(s(i).puntos(:,2));
    if aux_zM>zM
        zM=aux_zM;
    end

end


% comprueba si quiere todas las alturas de calado, o
% desde un calado máximo dado

if nargin==4
    aux=zM;
    zM=varargin{1};
    if zM>aux
         zM=aux;
         uiwait(msgbox(['Se va a trabajar con el calado máximo
zM=',num2str(aux)], 'Mensaje de error',...
        'error','modal'));
         return;
    end
end



% calculamos la altura de las diferentes líneas de agua

z=fliplr(linspace(zm,zM,nl));

% calculamos la semimanga para cada z y para cada sección
% y la guardamos en la matriz semimanga
% La primera fila de semimanga contiene la línea de agua
% a altura máxima del barco

for i=1:nl

    % cortamos la recta z=altura de la línea de agua
    % con cada una de las secciones

    for j=1:n
```

```
                % miramos si la recta corta a cada una de los
                % lados de la poligonal que forma la sección

                semimanga(i,j)=0;

                for k=1:s(j).numero

                    if strcmp(deblank(s(j).tipo(k,:)),'Paralela al Eje z')
                        if           (z(i)>=s(j).limites(k,1))           &
(z(i)<=s(j).limites(k,2))
                            if s(j).poligonal(k,1)>semimanga(i,j)
                                semimanga(i,j)=s(j).poligonal(k,1);
                                break;
                            end
                        end
                    elseif strcmp(deblank(s(j).tipo(k,:)),'Paralela al Eje
y')
                        if abs(z(i)-s(j).poligonal(k,1))<10^(-15)
                            if s(j).limites(k,2)>semimanga(i,j)
                                semimanga(i,j)=s(j).limites(k,2);
                                break;
                            end
                        end
                    elseif strcmp(deblank(s(j).tipo(k,:)),'Oblicua')
                        aux=s(j).poligonal(k,1)*z(i)+s(j).poligonal(k,2);
                        if s(j).limites(k,1)<=aux & aux<=s(j).limites(k,2)
                            if aux>semimanga(i,j)
                                semimanga(i,j)=aux;
                                break;
                            end
                        end
                    end

                end




            end

        end

        % generamos las coordenadas x de las secciones de popa a proa

        for i=1:length(s)
            x(i)=s(i).x;
        end


        % % ponemos el vector de cuadernas en la primera fila de la
        % % matriz de semimangas

        semimanga=fliplr(semimanga);
        semimanga=[x;semimanga];

        % ponemos el vector de alturas z en la primera columna,
completando
```

```
% la primera entrada con el valor de 0

z=[0;z'];
semimanga=[z,semimanga];

% si el fichero de salida ya existe lo borramos antes
d=dir('./Líneas de Agua/');

nd=length(d);

for i=1:nd
    a=strmatch(d(i).name,fichero);
    if a==1
        delete(['./Líneas de Agua/',fichero]);
        break;
    end
end



% guardamos en un fichero excel .xls la matriz semimanga

success = xlswrite(['./Líneas de Agua/',fichero],semimanga);
```

10.13 Función "Obtener secciones"

Esta función nos da las secciones del barco en un formato diferente al que se las introducimos. Se incluye un archivo .xls con los datos de las coordenadas del barco y las transforma para poder usarlas más fácilmente en el programa.

Los datos de la tabla los va traduciendo para poder ir identificando la posición de cada punto del casco del barco y así poder trabajar con ellos más tarde, ya sea para la representación gráfica del casco del barco o para los cálculos.

```
function s=obtener_secciones(fichero)

% Esta función lee las secciones de un barco en una variable
% de tipo estructura a partir del nombre de un fichero .xls
% s=obtener_secciones(fichero);
% Variables de entrada:
% fichero nombre del archivo que contiene el fichero .xls
% Variables de salida:
% s estructura que contiene las diferentes secciones del barco
%   Sus campos son los siguientes:
%   s.x  posición espacial x de la sección
%      s.numero    número de puntos de control en la sección
considerada
%   s.puntos coordenadas (y,z) de los puntos de control
%   s.tipo tipo de poligonal entre un punto y el siguiente. Puede
%          ser paralela al eje y, paralela al eje z, u oblicua
%   s.poligonal información necesaria para construir la poligonal
%   s.limites indica de dónde a dónde se extiende cada línea de la
%             poligonal


fichero=['./Secciones de Barcos/',fichero];
datos=xlsread(fichero);

n=size(datos,1);
ns=0;  % número de secciones

% Leemos las secciones

s=struct;
m=1;
while m<n

    ns=ns+1;
    s(ns).x=datos(m,1);
    s(ns).numero=round(datos(m,2));
    s(ns).puntos=datos(m+1:m+s(ns).numero,:);

    % calcula la poligonal entre cada dos puntos
```

```
            s(ns).tipo='';
            s(ns).poligonal=[];
            s(ns).limites=[];

            for j=1:s(ns).numero-1

                puntoA=s(ns).puntos(j,:);
                puntoB=s(ns).puntos(j+1,:);

                if abs(puntoA(1)-puntoB(1))<10^(-15)
                    s(ns).tipo=strvcat(s(ns).tipo,'Paralela al Eje z');
                    s(ns).poligonal=[s(ns).poligonal;[puntoA(1),0]];

s(ns).limites=[s(ns).limites;[min(puntoA(2),puntoB(2)),max(puntoA(2),pun
toB(2))]];
                elseif abs(puntoA(2)-puntoB(2))<10^(-15)
                    s(ns).tipo=strvcat(s(ns).tipo,'Paralela al Eje y');
                    s(ns).poligonal=[s(ns).poligonal;[puntoA(2),0]];

s(ns).limites=[s(ns).limites;[min(puntoA(1),puntoB(1)),max(puntoA(1),pun
toB(1))]];
                else
                    s(ns).tipo=strvcat(s(ns).tipo,'Oblicua');
                    a=(puntoA(1)-puntoB(1))/(puntoA(2)-puntoB(2));
                    b=puntoA(1)-a*puntoA(2);
                    s(ns).poligonal=[s(ns).poligonal;[a,b]];

s(ns).limites=[s(ns).limites;[min(puntoA(1),puntoB(1)),max(puntoA(1),pun
toB(1))]];

                end
            end

            puntoA=s(ns).puntos(s(ns).numero,:);
            puntoB=s(ns).puntos(1,:);

            if abs(puntoA(1)-puntoB(1))<10^(-15)
                s(ns).tipo=strvcat(s(ns).tipo,'Paralela al Eje z');
                s(ns).poligonal=[s(ns).poligonal;[puntoA(1),0]];

s(ns).limites=[s(ns).limites;[min(puntoA(2),puntoB(2)),max(puntoA(2),pun
toB(2))]];
            elseif abs(puntoA(2)-puntoB(2))<10^(-15)
                s(ns).tipo=strvcat(s(ns).tipo,'Paralela al Eje y');
                s(ns).poligonal=[s(ns).poligonal;[puntoA(2),0]];

s(ns).limites=[s(ns).limites;[min(puntoA(1),puntoB(1)),max(puntoA(1),pun
toB(1))]];
            else
                s(ns).tipo=strvcat(s(ns).tipo,'Oblicua');
                a=(puntoA(1)-puntoB(1))/(puntoA(2)-puntoB(2));
                b=puntoA(1)-a*puntoA(2);
                s(ns).poligonal=[s(ns).poligonal;[a,b]];

s(ns).limites=[s(ns).limites;[min(puntoA(1),puntoB(1)),max(puntoA(1),pun
toB(1))]];

            end
```

```
        m=m+s(ns).numero+1;
    end
```

10.14 Función "Saludo"

Esta función nos indica si se produce o no saludo. Para ello va comparando el momento peso con el momento empuje hasta el instante en el que el barco comienza el giro.

Para indicar si hay o no saludo, muestra un cuadro informativo donde se indica si se produce o no saludo. En caso de que se produzca saludo, el programa dejará de calcular el proceso.

```
      function
[dnec,dg,sal,caladok]=saludo(fichero,v,Xg,hang,dang,alfa,varargin)

      % Este fichero comprueba si en la botadura hay o no saludo.
      %
      %
[dnec,dg,sal,caladok]=saludo(fichero,v,Xg,hang,dang,alfa,varargin);
      %
      % % Variables de entrada:
      % fichero archivo excel que contiene los valores hidrostáticos
necesarios
      %         a diferentes calados
      % v  vector que contiene los valores hidrostáticos necesarios para
      %    el calado de flotación
      % Xg coordenada x del centro de gravedad del barco real
      % hang altura de la anguila
      % dang distancia desde la proa de la anguila a la proa del barco
      % alfa inclinación de la grada
      % varargin
      %           posibilidad '1', dg (distancia mojada de la grada)
      %            posibilidad '2', caladok (calado en el punto k de la
grada)
      %
      % Variables de salida:
      % dnec distancia necesaria para que no haya saludo
      % dg distancia mojada de la grada
      % sal variable que indica si hay saludo (sal=1) o no (sal=0)
      % caladok calado en el punto K de la grada



      if varargin{1}=='1'
          dg=varargin{2};
          caladok=sin(alfa)*dg;
      elseif varargin{1}=='2'
          caladok=varargin{2};
          dg=caladok/sin(alfa);
      end


      % Calculamos el calado de proa, y el ángulo beta de inclinación
del barco
```

```
        [Tpr, Tpp, beta]=obtencion_calados(fichero,v,Xg);


        % Calculamos la distancia de grada necesaria

        dnec=(Tpr+dang*tan(beta)+hang)/tan(alfa);

        % Comprobamos si hay saludo o no

        if dnec <= dg

            sal='0';
            IconoOk=imread('IconoOk.jpg');
            uiwait(msgbox('No      hay      Saludo','Determinación      de
Saludo','custom',IconoOk));

        else

            sal='1';
            uiwait(msgbox('Cuidado,    hay    Saludo','Determinación    de
Saludo','error'));

        end
```

10.15 Función "Transformar fichero"

Esta función transforma el fichero .gf con los datos del barco obtenidos en Rhinoceros a un archivo legible por MATLAB como es un archivo .xls. Este archivo .xls es el que se usa en el proceso de los cálculos en MATLAB. Los datos los ordena en una tabla la cual indica el valor de las coordenadas de los puntos que forman las formas del casco del barco a estudiar.

Más tarde, guarda el fichero en la carpeta desde la cual MATLAB ha de cargar los datos del barco en un formato legible.

```
function transformar_fichero(fichero_in,fichero_out,conversion)

% Esta función transforma un fichero con formato .gf que contiene
% las secciones de un barco en un fichero con excel con formato
% .xls
% El fichero .gf es generado con el programa Rhinoceros
% transformar_fichero(fichero_in)
% Variables de entrada:
% fichero_in  archivo con extensión .gf extraído de Rhinoceros
%       que contiene la información de las secciones del barco
% fichero_out contiene el nombre del archivo excel con extensión
%      .xls que contendrá también la información de las secciones
% conversion factor de conversión de la medida en que viene dado el
%      barco a metros

% componemos para formar la ruta
fichero_in=['./Secciones de Barcos/',fichero_in];

% leemos todos los datos del archivo en una variable de celda
data=textread(fichero_in,'%s');

% localizamos la posición donde empiezan las secciones

k=1;
while ~strcmp(data{k}(1),'s')
    k=k+1;
end

% le sumamos 2 a la posición de s0, s1, ...
k=k+2;

p=k; % variable que controla el bucle

% Inicializamos el mínimo de z a 0
zmin=1000;


% Separamos el número a la izquierda y a la derecha de la coma
aux=data{p};   % aux es char y no cell
```

```
c=1;
while aux(c)~= ','
    c=c+1;
end

xmin=-conversion*str2num(aux(1:c-1));
ns=str2num(aux(c+1:end));

A(p-k+1,1)=0;
A(p-k+1,2)=ns;

for i=1:ns
    p=p+1;
    aux=data{p};
    c=1;
    while aux(c)~= ','
        c=c+1;
    end
    A(p-k+1,1)=conversion*str2num(aux(1:c-1));
    A(p-k+1,2)=conversion*str2num(aux(c+1:end));
    if A(p-k+1,2)<zmin
        zmin=A(p-k+1,2);
    end
end

p=p+1;

while ~strcmp(data{p},'0.,0.,0.')

    aux=data{p};
    c=1;
    while aux(c)~= ','
        c=c+1;
    end

    A(p-k+1,1)=conversion*str2num(aux(1:c-1))+xmin;
    ns=str2num(aux(c+1:end));
    A(p-k+1,2)=ns;

    for i=1:ns
        p=p+1;
        aux=data{p};
        c=1;
        while aux(c)~= ','
            c=c+1;
        end
        A(p-k+1,1)=conversion*str2num(aux(1:c-1));
        A(p-k+1,2)=conversion*str2num(aux(c+1:end));
        if A(p-k+1,2)<zmin
            zmin=A(p-k+1,2);
        end
    end



    p=p+1;
end
```

```
zmin=-zmin;

p=p-k;


i=1;
while i<=p

    ns=A(i,2);

    for j=1:ns
        i=i+1;
        A(i,2)=A(i,2)+zmin;
    end

    i=i+1;

end


% si el fichero de salida ya existe lo borramos antes
d=dir('./Secciones de Barcos/');

nd=length(d);

for i=1:nd
    a=strmatch(d(i).name,fichero_out);
    if a==1
        delete(['./Secciones de Barcos/',fichero_out]);
        break;
    end
end



% guardamos en un fichero excel .xls la matriz A

success = xlswrite(['./Secciones de Barcos/',fichero_out],A);
```

10.16 Función "Trapecios"

Esta es la función que aproxima la integral de una cantidad de datos mediante el método de los trapecios.

```
function integral=trapecios(x,y)

% Esta función aproxima el cálculo de una integral por el método
% de los trapecios
% integral=trapecios(x,y);
% Variables de entrada:
% x vector de abscisas
% y vector de ordenadas
% Variables de salida:
% integral valor aproximado de la integral


% número de abscisas

n=length(x);

if n<=1

    integral=0;

else

    % aplicamos la regla simple de los trapecios n-1 veces

    h=abs(diff(x));

    % fórmula que nos aproxima la integral

    integral=sum(((y(1:n-1)+y(2:n))/2).*h);

end
```

10.17 Función "Valores Hidrostáticos"

Esta función es la encargada de mostrar adecuadamente el programa en la interfaz inicial. En esta función se incluyen las imágenes y las referencias para proseguir en los diferentes apartados que ofrece el programa.

```
function varargout = ValoresHidrostaticos(varargin)
% VALORESHIDROSTATICOS M-file for ValoresHidrostaticos.fig
%              VALORESHIDROSTATICOS,  by  itself,  creates  a  new
VALORESHIDROSTATICOS or raises the existing
%      singleton*.
%
%          H  =  VALORESHIDROSTATICOS  returns  the  handle  to  a  new
VALORESHIDROSTATICOS or the handle to
%      the existing singleton*.
%
%
VALORESHIDROSTATICOS('CALLBACK',hObject,eventData,handles,...) calls the
local
%       function named CALLBACK in VALORESHIDROSTATICOS.M with the
given input arguments.
%
%       VALORESHIDROSTATICOS('Property','Value',...) creates a new
VALORESHIDROSTATICOS or raises the
%         existing  singleton*.   Starting  from  the  left,  property
value pairs are
%       applied to the GUI before ValoresHidrostaticos_OpeningFcn
gets called.  An
%      unrecognized property name or invalid value makes property
application
%                     stop.       All    inputs    are    passed    to
ValoresHidrostaticos_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
ValoresHidrostaticos

% Last Modified by GUIDE v2.5 15-Jul-2014 19:14:39

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@ValoresHidrostaticos_OpeningFcn, ...
                   'gui_OutputFcn',
@ValoresHidrostaticos_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 ValoresHidrostaticos is made visible.
    function ValoresHidrostaticos_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 ValoresHidrostaticos (see
VARARGIN)

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

    % Update handles structure
    guidata(hObject, handles);

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

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
    %%%%%%%%%%%%%%%    CENTRAR    INTERFAZ    GRÁFICA    EN    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 );




    % --- Outputs from this function are returned to the command line.
    function varargout = ValoresHidrostaticos_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;


      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
      %%%%%%%%%%%%%%%%% INCLUIMOS LOS ESCUDOS  %%%%%%%%%%%%%%%%%
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

      UPCT=imread('escudo_upct.jpg');
      axes(handles.escudoUPCT);
      imshow(UPCT); axis off

      itn=imread('escudo_itn.jpg');
      axes(handles.escudoitn);
      imshow(itn); axis off




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

      hidrostatica


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

      botadura


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

      open('pdf/Tutorial de la Interfaz Gráfica.pdf');


      % --- Executes on button press in acerca.
      function acerca_Callback(hObject, eventdata, handles)
      % hObject    handle to acerca (see GCBO)
```

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

      open('pdf/Acerca De.pdf');

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

      salir=questdlg('¿Desea    salir    del    programa?','Salida    del
Programa','Si','No','No');
      switch salir
          case 'Si'
              close all;
          case 'No'
              return;
      end
```

11. Bibliografía

- Métodos numéricos con Matlab. UPV. A cordero barbero, E Martínez Losada, J L Hueso Pagoaga, J R Torregrosa Sánchez.
- Libro-Apuntes Hidrostática y estabilidad, UPCT, Olavo Palomo López
- Ayuda de Matlab R2007B
- Ayuda de Rhinoceros 4.0
- Sistemas de Construcción de Buques y Artefactos. Alfonso Martínez Martínez
- Apuntes de clase de Matemáticas
- http://www.fceia.unr.edu.ar/~pmarangu/integracion_aproximada.pdf
- http://repositorio.bib.upct.es/dspace/bitstream/10317/34/11/9-Estabilidad.pdf
- http://repositorio.bib.upct.es/dspace/bitstream/10317/2008/1/pfc4017.pdf
- Ship Hydrostatics and Stability - A.B.Biran