-

Universidad Politécnica de Cartagena

Trabajo de fin de Grado

“Métodos numéricos aplicados a la ingeniería Naval. Énfasis en los módulos de integración numérica”.

Grado en Arquitectura Naval e Ingeniería de Sistemas Marinos

ALBERTO MARTÍNEZ REDONDO

Profesor:
JUAN CARLOS TRILLO MOYA

AGRADECIMIENTOS

Quiero agradecer este trabajo ante todo a mi madre Esther Raquel Redondo, que es la persona que mas ha sufrido la dilatación de este en el tiempo que, si bien ha sido más que suficientemente largo, para ella sin duda alguna ha sido eterno.

Por otro lado, quiero dedicarle este trabajo a mi pareja Patricia Lozano Blanco, madre de mi futuro hijo y persona que me ha cuidado, animado y empujado día a día a hacer este trabajo de fin de grado.

Por último, pero no menos importante, quiero agradecer a mi profesor Juan Carlos Trillo Moya, que me ha dado la oportunidad de hacer y presentar un trabajo fin de grado que me eleva a la categoría de graduado. Sin duda alguna el mejor profesor que he tenido nunca, gracias por esa humanidad que ha hecho posible que una persona como yo pueda cumplir su sueño de ser ingeniero.

Nunca olvidaré a estas tres personas que han marcado mi vida y gracias a las cuales he aprendido tanto en principios y valores que aplicaré a mi día a día de ahora en adelante. Gracias.

ÍNDICE

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

Un proyecto de métodos numéricos aplicados al ámbito naval podría enfocarse en el diseño y análisis de estructuras y sistemas navales utilizando técnicas numéricas para la simulación y optimización de diferentes aspectos, como la hidrodinámica, la resistencia, la estabilidad y la maniobrabilidad de los buques.

El proyecto podría incluir la implementación y validación de modelos matemáticos y numéricos para la simulación de diferentes fenómenos, como la interacción entre el agua y el casco, la propulsión, la respuesta en el mar, entre otros. Se podrían utilizar métodos de elementos finitos para el análisis de la estructura y resistencia del casco, y métodos de simulación de dinámica de fluidos computacional (CFD) para la simulación de la hidrodinámica del buque.

Además, se podría investigar el impacto de diferentes diseños y configuraciones en la resistencia y la eficiencia del buque, y utilizar métodos de optimización para encontrar el mejor diseño posible, dado un conjunto de restricciones y objetivos.

El proyecto también podría abordar aspectos relacionados con la navegación, como la predicción de la maniobrabilidad y la estabilidad del buque en diferentes condiciones de mar y viento. Se podrían utilizar técnicas de simulación numérica para evaluar diferentes escenarios de navegación y evaluar la capacidad del buque para mantener su estabilidad y maniobrabilidad en diferentes condiciones.
En resumen, un proyecto de métodos numéricos aplicados al ámbito naval podría ser muy amplio y abarcar diferentes aspectos relacionados con el diseño, análisis y optimización de buques y sistemas navales, utilizando técnicas numéricas avanzadas para la simulación y evaluación de diferentes fenómenos y escenarios.

1.1. INTRODUCCIÓN A LA INTEGRACIÓN NUMÉRICA

La integración numérica es una técnica utilizada en el cálculo para aproximar el valor de una integral definida de una función en un intervalo dado. Esta técnica se utiliza cuando la integral no puede ser evaluada analíticamente o cuando se desea obtener una aproximación numérica rápida y precisa.

Existen varios métodos de integración numérica, cada uno con sus propias ventajas y desventajas. Algunos de los métodos más comunes incluyen la Regla del Punto Medio, la Regla del trapecio, la Regla de Simpson y la Regla de Cuadratura de Gauss.

La regla del punto medio es el método más simple, que utiliza el valor de la función en el punto medio del intervalo como una aproximación de la integral en ese intervalo. La regla del trapecio utiliza una aproximación lineal de la función para aproximar la integral en un intervalo dado. La regla de Simpson utiliza una aproximación cuadrática de la función para aproximar la integral en un intervalo dado. La regla de Cuadratura de Gauss es un método más avanzado que utiliza una combinación de nodos y pesos específicos para aproximar la integral. Este método puede ser muy preciso para funciones suaves, pero puede requerir un número significativo de nodos y pesos para obtener una buena aproximación.

En general, la elección del método de integración numérica dependerá de la precisión requerida, la complejidad de la función y el tiempo de cálculo disponible.

1.1.1.Contextualización de la integración numérica

En el transcurso de nuestra formación académica en el análisis matemático, particularmente en la disciplina de cálculo integral, adquirimos habilidades para calcular una integral definida de una función continua. Este procedimiento se lleva a cabo a través de la implementación del Teorema Fundamental del Cálculo:

Teorema Fundamental del Cálculo (Regla de Barrow).

Sea *f(x)* una función continua, siendo $f(x) \geq 0$ y definida en el intervalo [a,b] y sea *F(x)* una función primitiva de *f(x)*, entonces:

$$I = \int_a^b f(x)\, dx = F(b) - F(a),$$

es el área de la región del plano delimitada por la gráfica de la función, el eje de abscisas y rectas verticales $x=a$ y $x=b$ (ver *Figura 1*).

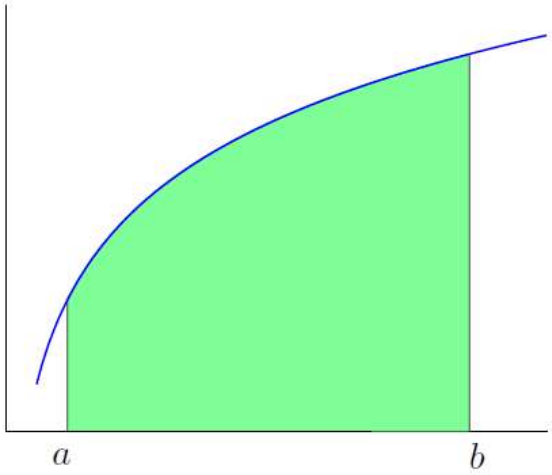


Figura 1: Área encerrada entre la gráfica de la función f y el intervalo [a,b].

En la práctica, el problema surge cuando no somos capaces de identificar la función primitiva necesaria. Este es también el caso de algunas integrales que, a primera vista, pueden parecer sencillas, como, por ejemplo:

$$I = \int_0^1 e^{x^2}\, dx,$$

la cual es simplemente imposible de resolver mediante el Teorema Fundamental del Cálculo.

En tales situaciones, la **Integración Numérica** emerge como un recurso matemático vital, ofreciendo métodos y ecuaciones para el cálculo aproximado de integrales definidas. Este instrumental permite la computación, aunque sea aproximada, de valores de integrales definidas que de otro modo serían incalculables mediante métodos analíticos. Lo que es más importante, permite realizar estos cálculos de manera efectiva en una computadora.

Este proceso de aproximación se lleva a cabo integrando una función más simple que sirve como aproximación de la función original. Dicha función simplificada se puede derivar a través de la utilización de polinomios. Esta es la estrategia más común. Por ejemplo, la Regla de Simpson, que es una de las más famosas, surge del uso del polinomio de interpolación de Lagrange de segundo grado:

$$p_2(x) = \frac{(x-c)(x-b)}{(a-c)(a-b)}f(a) + \frac{(x-a)(x-b)}{(c-a)(c-b)}f(b) + \frac{(x-a)(x-c)}{(b-a)(b-c)}f(c),$$

que pasa por los puntos $\big(a, f(a)\big), \big(b, f(b)\big), \big(c, f(c)\big)$.

En general el Polinomio de Lagrange de grado n que pasa por los puntos $(x_i, f(x_i))$ siendo $i = 0, \dots, n$ quedaría de la forma:

$$p_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i),$$

donde $L_i(x)$ son los Polinomios de Lagrange de grado n.

$$L_i(x) = \frac{(x-x_1)\dots(x-x_{i-1})(x-x_{i+1})\dots(x-x_n)}{(x_i-x_1)\dots(x_i-x_{i-1})(x_i-x_{i+1})\dots(x_i-x_n)} = \prod_{\substack{j=1:n \\ j\neq i}} \frac{x-x_j}{x_i-x_j},$$

$$L_i(x_j) = \begin{cases} 1 \; si \; i = j \\ 0 \; si \; i \neq j \end{cases}.$$

Si denotamos $w_i \coloneqq \int_a^b L_i(x)\, dx$, entonces:

$$\int_a^b p_n(x)\, dx = \sum_{i=0}^{n} w_i f(x_i)\,,$$

siendo ésta la fórmula de aproximación $\int_a^b f(x)\, dx \approx \int_a^b p_n(x)\, dx$ que se emplea para derivar las diversas Reglas de Newton Cotes, las cuales estudiaremos a lo largo de este proyecto.

1.2. INTEGRACIÓN NUMÉRICA EN EL ÁMBITO NAVAL

Las fórmulas aproximadas son fórmulas que se utilizan para obviar las indefiniciones o desconocimientos que tengamos a la hora de determinar algún parámetro que necesitemos, sobre todo durante las fases de anteproyecto. Existen fórmulas de este tipo en todos los campos, no solamente en Teoría del Buque. En este proyecto estudiaremos la integración numérica como método para la aproximación de distintas características del buque.

A continuación, vamos a comentar algunas de ellas, si bien en este proyecto estudiaremos únicamente la integración numérica como método para la aproximación de distintas características del buque.

A la hora de aplicar las fórmulas aproximadas conviene ser cautos y tener siempre presente dos cosas:

a) Como su denominación sugiere, estas fórmulas son simplemente aproximaciones, y su precisión disminuye a medida que aumenta el grado de indefinición de los parámetros que emplean. Siempre que sea posible, se debería optar por un método exacto (aún dentro de la inexactitud con la que se

trabaja en Teoría del Buque: interpolaciones, integraciones aproximadas, etc.) antes que una fórmula aproximada.

b) Las fórmulas aproximadas pueden derivarse de fundamentos matemáticos o geométricos, o bien de una base estadística (mediante el análisis de datos relacionados con una serie de buques). Las primeras siempre son válidas, mientras que las segundas podrían presentar problemas si los barcos de los que se obtuvieron los datos ya no son relevantes o están obsoletos.

La integración numérica se usa habitualmente en el ámbito naval. En particular, usaremos las fórmulas de Newton Cotes cerradas para aproximar integrales que aparecen en fenómenos hidrostáticos y también veremos algún ejemplo en el que se pueden aproximar magnitudes asociadas al campo de la propulsión.

1.2.1 Introducción a los fenómenos hidrostáticos

Las curvas hidrostáticas son diagramas que ilustran diversas características geométricas de un buque, las cuales se calculan a partir del plano de formas o de la cartilla de trazado. En términos generales, podemos clasificarlas en tres categorías o familias:

a) Las curvas hidrostáticas normales, que también se conocen como carenas rectas. Es usual que, al mencionar "curvas hidrostáticas" sin especificar más, se haga referencia a este tipo.
b) Las curvas de Bonjean.
c) Las curvas transversales de estabilidad, también conocidas como curvas KN, carenas inclinadas o pantocarenas.

Pero nosotros vamos a centrarnos únicamente en las curvas hidrostáticas normales, las cuales serán el foco de nuestra investigación y análisis desde el punto de vista de la integración numérica en este proyecto.

1.2.1.1 Curvas hidrostáticas o carenas rectas

Estas son curvas o diagramas que ilustran ciertos parámetros del barco con relación al calado T, con la peculiaridad de que el calado se representa en el eje de ordenadas. Vamos a introducirlas todas y más tarde profundizaremos en algunas de ellas.

1. Área de flotación (A_F). La integral a aproximar mediante los métodos que veremos en este proyecto será:

$$Af = \int_{Lf} y\, dx,$$

2. Toneladas por centímetro de inmersión *(TCI)*. Esta curva no es más que una variante adaptada de la previamente mencionada. Si variamos el peso del buque, para que éste siga estando en equilibrio, este peso debe ser compensado con un incremento de empuje $\partial\Delta$ correspondiente al ∂T, y tal que:

$$\partial\Delta = \gamma \cdot \partial\nabla = \gamma \cdot A_f \cdot \partial T\ ,$$

si consideramos ∂T como 1 centímetro ($\partial T = 1\ cm = \frac{1}{100}\ m$), entonces tenemos las *(TCI)*:

$$\partial\Delta = \gamma \cdot A_f \cdot \partial T \;\; ; \;\; \partial T = \frac{1}{100}\, m \quad \rightarrow \quad TCI = \frac{\gamma \cdot A_f}{100}.$$

3. Abscisa del centro de gravedad *(c.d.g.)* de la flotación, bien con respecto a la perpendicular de la popa (x_F), o bien referida a la sección media $(\otimes_F)$. La fórmula para calcular esta magnitud queda definida como:

$$\otimes F = \frac{m_\otimes}{A_F}.$$

Como hemos visto en el anterior ejemplo, esto aplica si las flotaciones resultan ser simétricas. La integral para el cálculo del momento queda:

$$m_\otimes = 2 \cdot \int_{L_f} xy\, dx \qquad \text{y} \qquad A_F = 2 \cdot \int_{L_F} y\, dx,$$

4. Volumen de trazado o volumen de carena (∇). La integral a aproximar queda como:

$$\nabla = \int_L A_F \; dx,$$

5. Desplazamiento total (Δ). Esta magnitud se determina multiplicando el volumen de la carena (∇) por la densidad, así obtendríamos los empujes o desplazamientos y la integral es:

$$\Delta = \gamma \cdot \nabla = \int_L A_F \cdot \gamma \; dx.$$

6. Ordenada o altura del centro de carena *(c.d.c.) (KB)*. Por definición de la altura del *c.d.c.*:

$$KB = \frac{M_k}{\nabla},$$

donde las integrales a aproximar son:

$$M_k = \int_0^T A_f \cdot z \cdot dz, \qquad \text{y} \qquad \nabla = \int_0^T A_f \cdot dz.$$

7. Abscisa del *c.d.c.* $(x_B \;\; \text{ó} \;\; \otimes_B)$. Si elegimos la sección media como punto de referencia,

$$\otimes_B = \frac{M_\otimes}{\nabla},$$

siendo las integrales a aproximar mediante métodos numéricos:

$$M_\otimes = \int_0^T \otimes F \cdot A_f \cdot dz, \qquad \text{y} \qquad \nabla = \int_0^T A_f \cdot dz.$$

8. Radio metacéntrico transversal $(BM \;\; \text{ó} \;\; BM_t)$. A veces los valores del radio metacéntrico se suman a los del *KB* para cada calado y se representa el KM_t $(KM_t = KB + BM_t)$. La fórmula es:

$$BM_t = \frac{I_t}{\nabla},$$

siendo en el cálculo de I_t donde nos correspondría utilizar los métodos de integración numérica

$$I_t = 2\int_{L_f} \frac{1}{3} y^3 \, dx.$$

9. Radio metacéntrico longitudinal (BM_l). Al igual que en el caso anterior, a veces se representa KM_l ($KM_l = KB + BM_l$). La fórmula queda como sigue:

$$BM_l = \frac{I_l}{\nabla}, \qquad ; \qquad I_l = I_{\otimes} - \otimes F^2 \cdot A_f \, .$$

Y si, como suele suceder, la flotación es simétrica:

$$I_{\otimes} = 2 \cdot \int_{L_f} x^2 \cdot y \, dx \, .$$

En muchas ocasiones se dibuja la distancia del metacentro, que sería:

$$KM_l = KB + BM_l \, .$$

10. Momento para alterar el trimado un centímetro *(MTC)*. En las publicaciones inglesas y americanas se suele utilizar su tonelada por pulgada.

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

11. Coeficientes de forma $(\delta, \beta, \alpha, \varphi)$. Los valores que se utilizan son los correspondientes a cada calado. Como lo normal es que estén comprendidos entre 0 y 1, se representan como se ve en la *Figura 2*.

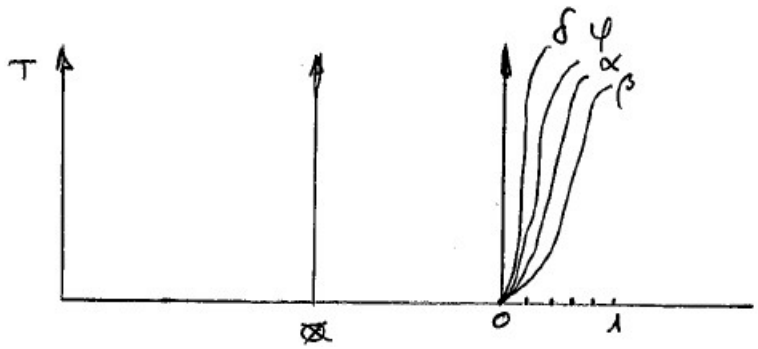
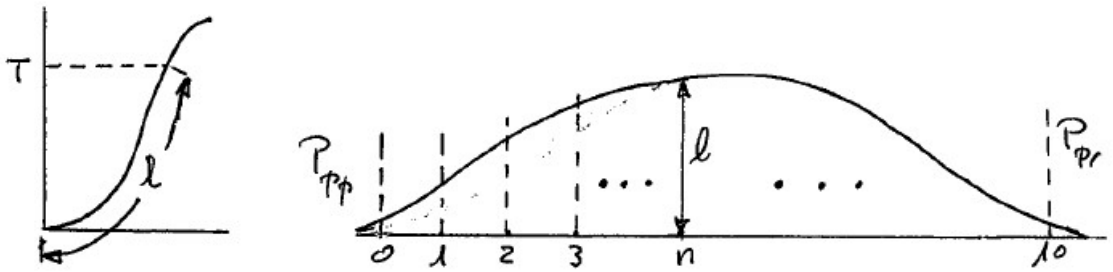


Figura 2. Coeficientes de forma.

12. Superficie mojada. Esta es el área de la superficie exterior de la carena que entra en contacto con el agua para un calado específico. Se emplea para calcular la resistencia al avance del buque debido a la fricción con el agua.



Figuras 3 y 4. Valores de longitud de las semisecciones para diferentes calados.

Los distintos valores de estos desarrollos se llevan sobre la línea base, que representa la eslora, como ordenadas sobre los puntos correspondientes a las secciones. La integral que nos da el área encerrada por esta nuevas curvas (*Figuras 3 y 4*) nos aproxima el área de trazado de la superficie mojada.

Todas estas curvas, pueden calcularse, o tienen calculos asociados a la integración numérica, ya que ésta facilita mucho los calculos de los mismos. Más adelante, en el Capítulo 5 de este proyecto estudiaremos algunas de ellas más en profundidad y veremos algunos de los programas en lenguaje Matlab que hemos realizado.

Y ahora vamos a pasar a contextualizar la integración numérica que es el objeto de nuestro proyecto y a estudiar en detenimiento algunos de los métodos más utilizados.

1.2.1 Introducción a la propulsión y la resistencia

Cuando un buque se desplaza en el agua, surge una fuerza que se resiste a este movimiento, es decir, una resistencia al avance. Para superar esta resistencia, es necesario encontrar algún tipo de mecanismo que ejerza una fuerza en dirección contraria. Esta fuerza impulsora se conoce como empuje y el estudio hidrodinámico de los dispositivos capaces de generarla da lugar a una sección de la Teoría del Buque conocida como propulsión.

Ambas disciplinas, pueden ser calculadas con la ayuda de la integración numérica. Vamos a introducir cada una de ellas desde el punto de vista teórico.

1.2.1.1 Propulsión e integración numérica

La hélice propulsora representada en la *Figura 5* está compuesta por una cantidad de palas idénticas, dispuestas en ángulos equidistantes y unidas en su base a un núcleo que se encuentra montado en el extremo de popa del eje propulsor del buque.

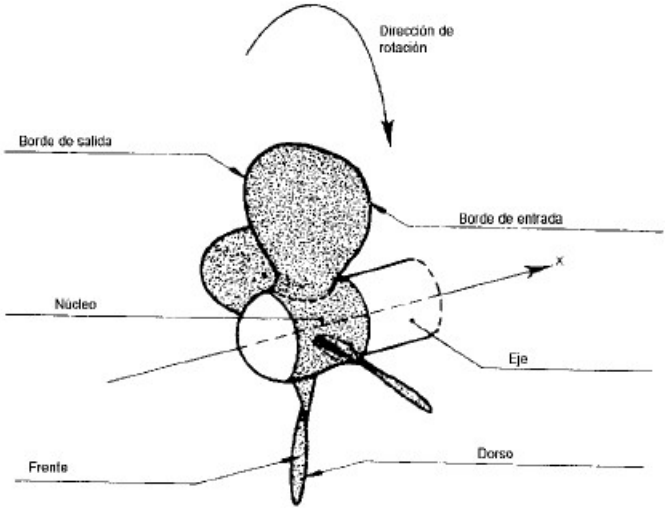


Figura 5. Geometría de la hélice propulsora.

Las palas de la hélice generan empuje debido a la diferencia de presión entre sus caras: la cara de presión, donde se produce una sobrepresión, y la cara de succión, donde hay una depresión. La cara de presión o cara activa es la que un observador ubicado detrás de la hélice vería, y la cara de succión es la contraria. Véase la *Figura 6*.

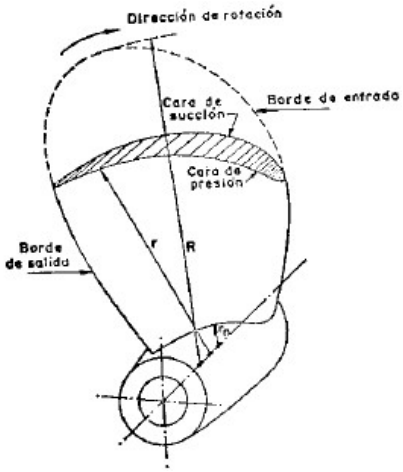


Figura 6. Caras de presión y succión de la pala de una hélice.

La cara de presión de cada pala es parte de una superficie helicoidal o helicoide. Y es aquí cuando juega un papel fundamental la integración numérica a la hora de aproximar estas superficies.

La integración numérica tiene aún una mayor relevancia y aplicación en este campo ya que es aplicable computacionalmente e introduciendo el número adecuado de intervalos, con computadores muy potentes, puedes alcanzar la precisión deseada.

1.2.1.2. Resistencia total y sus componentes

En la *Figura 7* se ha representado un curva típica del coeficiente de resistencia total específico C_T en función de *Rn*.

En esta figura se pueden observar las diferentes maneras de descomponer la resistencia total de un buque.
El eje de abscisas *0Rn* representa la resistencia de un buque inmerso en un fluido perfecto (donde la resistencia es nula).

La línea *A* corresponde a la resistencia de un buque de superficie que navega en un fluido ideal (donde la resistencia se debe únicamente a la formación de olas).

La línea *z* corresponde a la resistencia por fricción de una placa plana que equivale al buque (R_F de Froude).

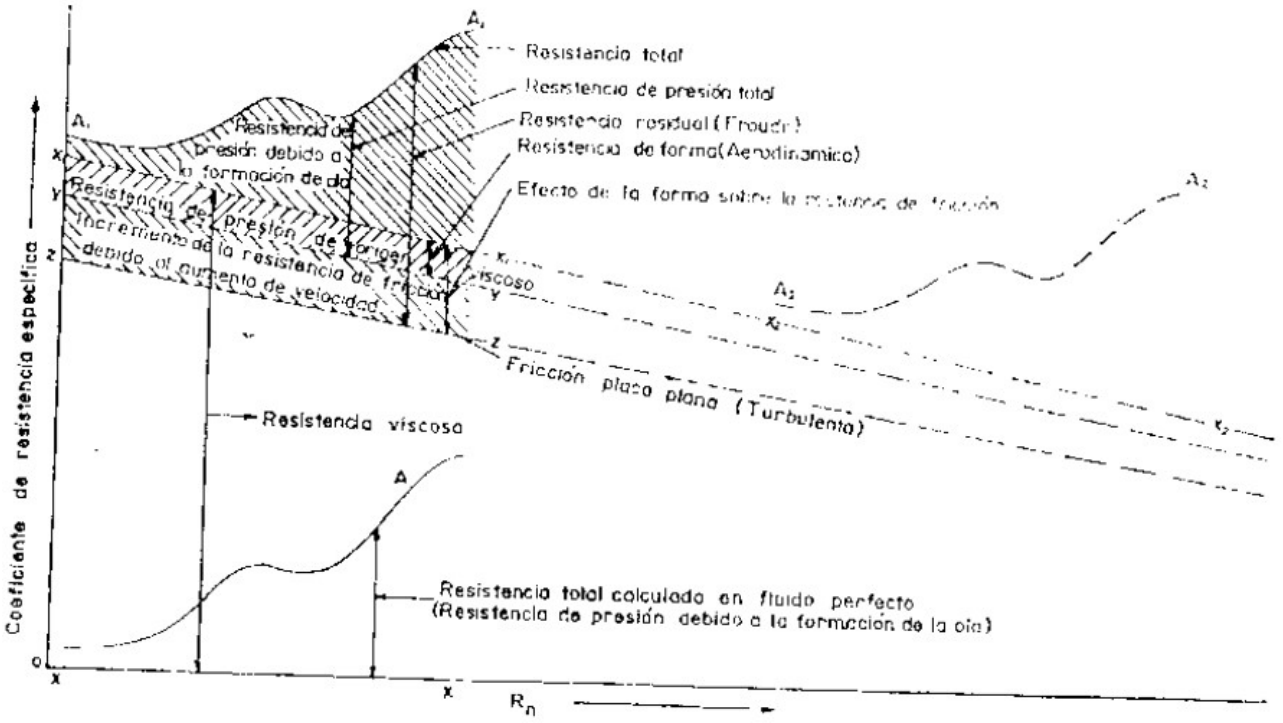


Figura 7. Variación de las distintas componentes de la resistencia con *Rn*.

La línea *y* representa la resistencia por fricción derivada de la integración de los componentes tangenciales de las fuerzas (tercera descomposición de la resistencia total) de un buque que navega inmerso en un fluido real.

La diferencia entre las líneas *x* e *y* corresponde a la resistencia de presión (resultado de la integración de las componentes normales de las fuerzas), con origen en la viscosidad, de un buque que navega inmerso en un fluido real. *A* la diferencia entre las líneas *x* y *z* se llama factor de forma.

La línea A_1 corresponde a la resistencia de un buque que navega en un fluido real. La forma de esta línea no coincide exactamente con la línea A debido a cómo la viscosidad afecta la resistencia por la formación de olas.

Es en estas curvas que encierran áreas contra los ejes de coordenadas donde entra en juego la integración numérica para simplificar el cálculo de estas áreas.

2. REGLAS DE NEWTON COTES CERRADAS

Estas fórmulas son los tipos de integración numérica más comunes. La base sobre la que se fundamentan las fórmulas de Newton-Cotes, es la idea de reemplazar una función complicada o un conjunto de datos tabulados por un polinomio de aproximación, el cual sea fácil de integrar.

$$I = \int_a^b f(x)dx \cong \int_a^b P_n(x)dx,$$

siendo

$$P_n(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} + a_n x^n,$$

donde *n* indica el grado del polinomio. En las *Figuras 8a y 8b* que aparecen a continuación, se utilizan dos polinomios diferentes para aproximar una misma función, en el caso de la *Figura 8a* se utiliza un polinomio de primer grado, una línea recta, mientras que en la *Figura 8b* utilizamos una parábola con el mismo propósito de aproximar la función representada.

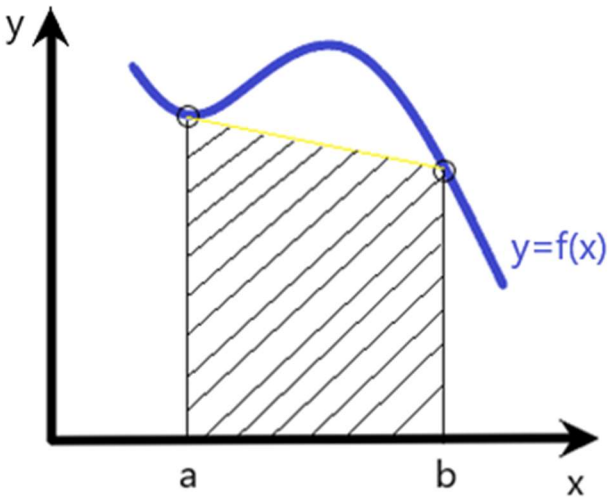


Figura 8a. Aproximación de una integral mediante el área bajo una línea recta.

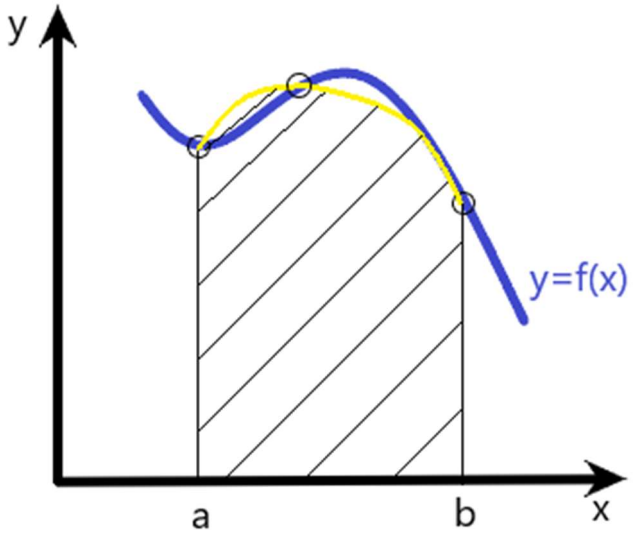


Figura 8b. Aproximación de una integral mediante el área bajo una parábola.

Dentro de las Fórmulas de Newton-Cotes, podemos diferenciar entre las formas cerradas y abiertas. En las fórmulas cerradas conocemos el principio y el final de los límites de integración, mientras que, en las fórmulas abiertas, los límites de integración se extienden más allá de los datos conocidos. En las *Figuras 9a y 9b* se pueden visualizar ambos casos.

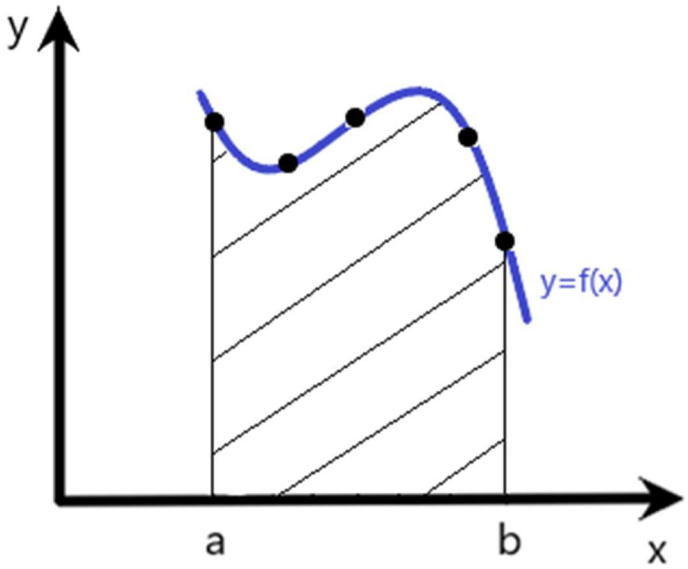


Figura 9a. Fórmula de integración cerrada.

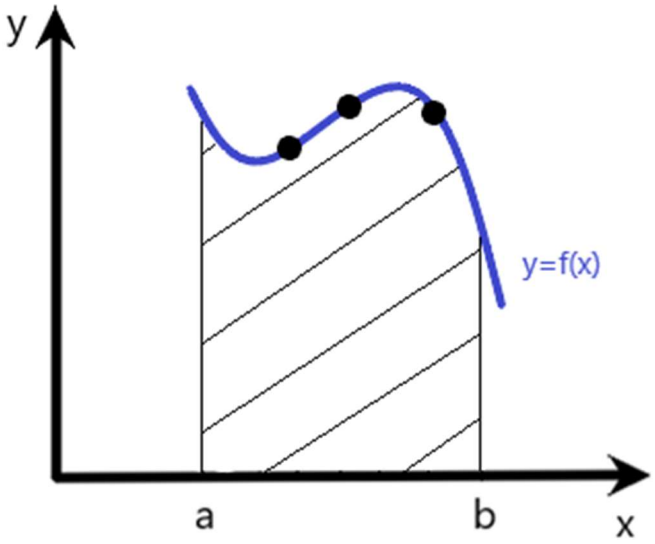


Figura 9b. Fórmula de integración abierta.

En este proyecto estudiaremos únicamente las fórmulas de integración numérica cerradas. Vamos a exponer algunos de los motivos si bien la elección entre el uso de una técnica de integración numérica cerrada o abierta depende en gran medida de la función a integrar y de la precisión deseada.

La integración numérica cerrada se basa en la evaluación de la función a integrar en los extremos del intervalo de integración y en uno o más puntos intermedios, y utiliza una fórmula matemática específica para aproximar el valor de la integral. Por otro lado, la integración numérica abierta utiliza una fórmula que no requiere la evaluación en los extremos del intervalo de integración.

Una de las principales ventajas de la integración numérica cerrada es que puede proporcionar una mayor precisión en la aproximación de la integral, especialmente para funciones suaves y regulares. Además, las fórmulas cerradas son más fáciles de implementar y son menos propensas a errores de redondeo que las fórmulas abiertas.

En general, la elección entre una técnica de integración numérica cerrada o abierta dependerá de la naturaleza de la función a integrar y de la precisión requerida. En muchos casos, la integración numérica cerrada es preferible debido a su mayor precisión y facilidad de implementación.

Existen numerosas Reglas de Newton-Cotes cerradas, las cuales se utilizan en función de la precisión que deseemos obtener de la aproximación, que a su vez dependerá del número de intervalos que tomemos en ésta.

Estos métodos de integración numérica sirven para realizar cálculos estructurales, de momentos, de fuerzas hidrostáticas y otras magnitudes. Dependiendo de la exactitud que requiera la función a aproximar recurriremos a una regla u otra.

Pasamos a estudiar las fórmulas de Newton Cotes más usadas.

2.1. REGLA DE LOS TRAPECIOS

La Regla de los Trapecios es la más sencilla, dentro de las diferentes reglas de integración numérica que estudiaremos en este proyecto. Está basada, como veremos a continuación, en la utilización de polinomios de grado 1.

Empezaremos con la Regla de los Trapecios Simple, siendo su uso y estudio únicamente de valor didáctico, y como medio para conseguir la regla compuesta. Esta situación suele ocurrir siempre con las reglas simples.

2.1.1. Regla de los Trapecios Simple

La Regla de los Trapecios Simple (*Figura 10*) consistirá en aproximar el área encerrada entre una función positiva y el eje x en el intervalo [a,b] mediante la creación de un trapecio cuyos puntos serán $((a,0),(a,f(a)),(b,0),(b,f(b)))$, cuya altura irá del extremo *a* al extremo *b*.

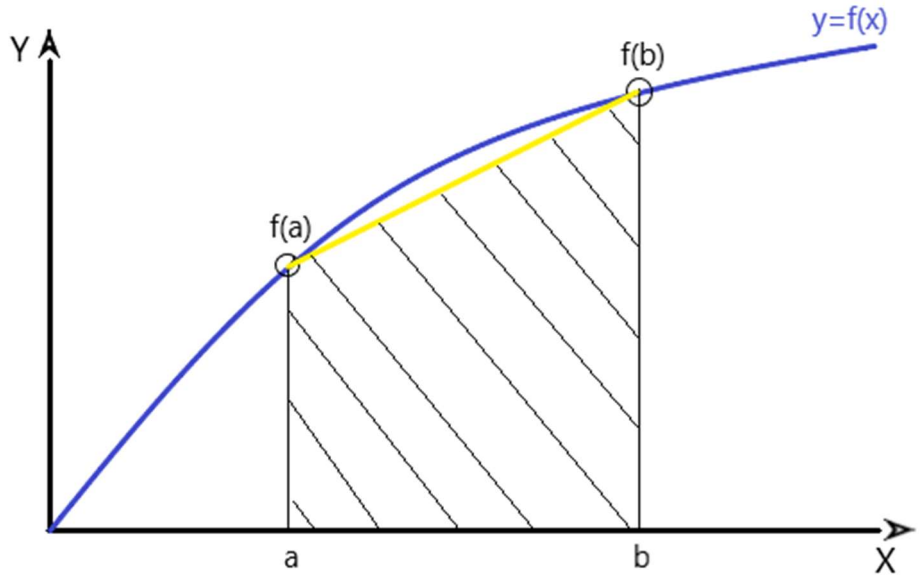


Figura 10. Representación gráfica de la Regla de Trapecios Simple.

La fórmula de la Regla de Trapecios Simple a través de la ecuación de Lagrange queda:

$$\int_a^b f(x)\,dx \approx \frac{h}{2}\big(f(x_0)+f(x_1)\big)\,.$$

Geométricamente, la regla del trapecio equivale a aproximar el área del trapecio bajo la línea recta que une *f(a)* y *f(b)* en la Figura 4. Esta interpretación con áreas en realidad es fácilmente adaptable para funciones *f(x)* no siempre positivas en el intervalo [a,b].

2.1.2. Regla de los Trapecios Compuesta

Existe también la Regla de los Trapecios Compuesta (*Figura 11*) que consiste en dividir el intervalo en varios subintervalos de igual longitud aplicando en cada uno de ellos la Regla de Trapecios Simple. La aproximación será más exacta cuantos más subintervalos creemos.

Es esta regla de los trapecios compuesta la que estudiaremos aquí a través del programa MATLAB creando varios ejercicios con fines didácticos y un ejemplo aplicado al ámbito naval.

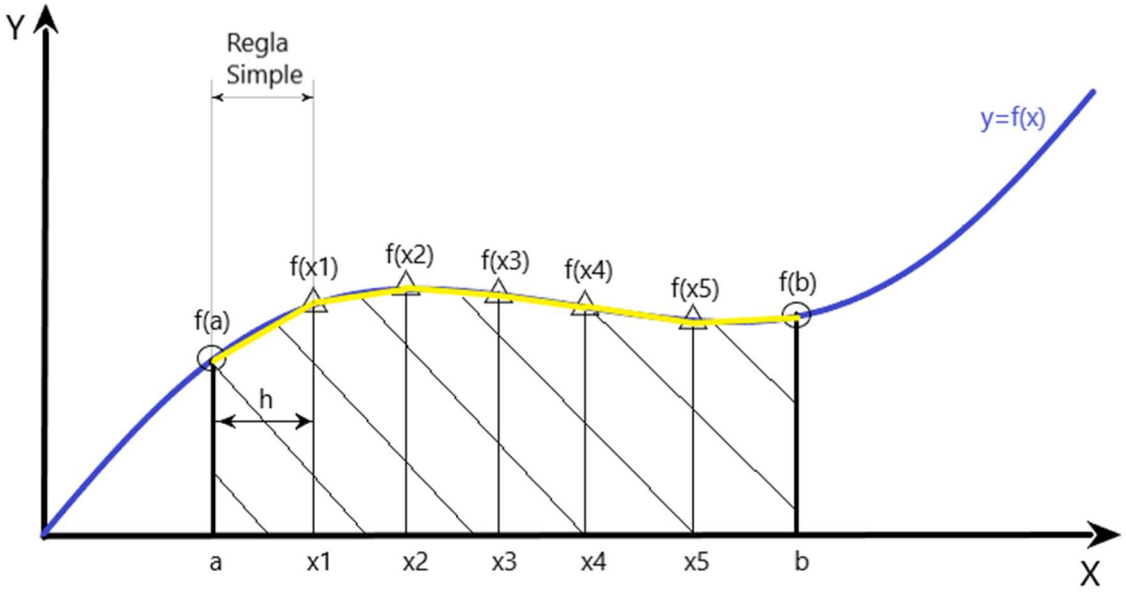


Figura 11. Representación gráfica de la Regla de Trapecios Compuesta.

Dividiendo el intervalo [a,b] en n subintervalos de igual longitud, obtenemos los nodos:

$$x_0 = a, \quad x_{(i+1)} = x_i + h, \quad i = 1 \dots n. \quad \text{Siendo} \quad h = \frac{(b-a)}{n}.$$

Y, aplicando en cada subintervalo la Regla de Trapecios Simple tenemos la Regla de los Trapecios Compuesta que es:

$$\int_a^b f(x)dx \approx \frac{h}{2}\big(f(x_0) + f(x_1)\big) + \frac{h}{2}\big(f(x_1) + f(x_2)\big) + \frac{h}{2}\big(f(x_2) + f(x_3)\big) + \cdots + \frac{h}{2}\big(f(x_{n-1}) + f(x_n)\big).$$

Sacando factor común:

$$\int_a^b f(x)dx \approx h(\frac{E}{2} + I)\,,$$

donde

$$E = f(x_0) + f(x_n),$$
$$I = f(x_1) + f(x_2) + \cdots + f(x_{n-1}).$$

Una de las ventajas de este método es su flexibilidad al dividir los intervalos en dos puntos, permitiendo su aplicación a cualquier división del dominio que hayamos realizado en la curva.

Aun siendo bastante más exacto que el método de la Regla de los Trapecios Simple, este método sigue teniendo un error considerable, siendo los métodos que veremos a lo largo del proyecto más precisos que éste.

Sin embargo, este método es ampliamente utilizado debido a su sencillez tanto teórica como de implementación. Con la potencia de los ordenadores de hoy en día muchas veces se prefiere usar este método, aunque sea a base de tomar un número de subintervalos muy grande para poder tener una precisión adecuada.

2.1.3. Error de la Regla de los Trapecios.

La regla de los trapecios es un método de integración numérica que se utiliza para aproximar el valor de una integral definida. Aunque es relativamente sencillo de implementar, la regla de los trapecios no siempre proporciona una aproximación muy precisa y puede ser propenso a errores.

La principal fuente de error en la regla de los trapecios es la aproximación lineal utilizada para conectar los puntos de la función en el intervalo de integración. Esta aproximación puede no ser muy precisa para funciones con curvas pronunciadas o con cambios abruptos en la pendiente.

El error para la Regla de los Trapecios Simple es:

$$E_h(f) = \frac{-f''(\mu)}{12}(b-a)^3,$$

donde vemos que el valor de la derivada nos indica el grado de exactitud de la fórmula, que en este caso es 1 y el valor de la potencia de h nos indica el orden de aproximación local, que es 3.

Por otro lado, el error de la Regla de los Trapecios Compuesta es:

$$E_h(f) = \frac{-h^3}{12}\frac{\sum_{i=0}^{n} f''(\mu_i)}{n}n = \frac{-h^3}{12}f''(\mu)\frac{(b-a)}{h},$$

este error es exactamente de grado 1 y tiene un orden de aproximación global 2, observando que se ha reducido en 1 respecto al orden de aproximación local.

En conclusión, aunque la regla de los trapecios no siempre proporciona la mejor aproximación, sigue siendo una técnica útil de integración numérica en ciertas situaciones. Es importante tener en cuenta sus limitaciones y utilizar otras técnicas de integración numérica cuando se requiere una mayor precisión.

2.1.4. Método de los Trapecios programado en lenguaje Matlab

En este proyecto hemos realizado 4 programas en lenguaje Matlab en los que introduciremos diferentes datos de entrada. En ellos aplicaremos la Regla de Trapecios Compuesta para aproximar la integral de una función.

2.1.4.1. Trapecios Compuesto dado el número de subintervalos

En este primer ejemplo hemos escrito en lenguaje Matlab un código para aproximar la integral de una determinada función, que puede ser introducida como conjunto de datos o como expresión simbólica, teniendo como datos de entrada la función, los extremos del intervalo [a,b] y el número de subintervalos a aplicar.

A continuación, se ofrece el código de programación desarrollado al efecto:

```
function [int]=Trapecios_P1_Mab(f,M,a,b)

% Este programa tiene como objetivo calcular el valor de la integral
% mediante la Regla de los Trapecios Compuesta de una determinada función
% introducida como una expresión simbólica o como un conjunto de datos
% para la cual definiremos el número total de subintervalos que sería el
% número de veces que vamos a aplicar Trapecios Simple y los extremos del
% intervalo donde queremos aproximar la integral.
%
% [int]=Trapecios_P1_Mab(f,M,a,b)
%
% Datos de entrada:
% f: función introducida como expresión simbólica o como conjunto de datos.
% M: número de subintervalos.
% a,b: extremos del intervalo.
%
% Datos de salida:
% [int]: integral de la función calculada para los datos de entrada
% (M y a,b).
%
% Ejemplo 1:
% syms x
% [int]=Trapecios_P1_Mab(sin(x),100,0,pi)
%
% Ejemplo 2:
%     [int]=Trapecios_P1_Mab([1.000000000000000                0.995004165278026
0.980066577841242    0.955336489125606    0.921060994002885    0.877582561890373
0.825335614909678     0.764842187284488     0.696706709347165  0.621609968270665
0.540302305868140],100,0,pi)
%
% Ejercicio para el cálculo del área de flotación de un buque:
%  [int]=Trapecios_P1_Mab([0  8.9  11.8  15.5  15.7  15.7  15.7  13.6  9.6  5.8
0],100,0,130)


% Introducimos los datos de entrada
syms x

% Calculamos la distancia entre nodos
h=(b-a)/M;

% Ahora pasamos a inicializar las variables

I=0;  % acumulador para el valor de la integral
```

```
if ~isfloat(f)

    % cálculo de los nodos
    xn(1)=a;
    xn(M+1)=b;

    for i=2:M
        xn(i)=xn(i-1)+h;
    end

    % Regla de los Trapecios

    % Denotaremos por E a la suma de las ordenadas en los extremos
    E=double(subs(f,a))+double(subs(f,xn(M+1)));

    % Denotaremos por I a la suma de las ordenadas intermedias del método
    for i=2:M
        I=I+double(subs(f,xn(i)));
    end

else

    I=0;

    M=length(f)-1;

    fprintf('\n');
    fprintf('El valor de M ha sido actualizado a la longitud del vector de datos
introducido menos una unidad, M= %d',M);

    % calculamos el espaciado h entre nodos

    h=(b-a)/M;

    % Regla de los trapecios

    % Denotaremos por E a la suma de las ordenadas en los extremos
    E=f(1)+f(M+1);

    % Denotaremos por I a la suma de las ordenadas intermedias del método
    for i=2:M
        I=I+f(i);
    end

end

% Calculando el valor de la aproximación
[int]=h*((E/2)+I);
```

FINAL DEL PROGRAMA

Dentro del código del propio programa de Matlab se han incluido algunos ejemplos que pueden ser utilizados para testar su funcionamiento. También se ha incluido un ejemplo de aplicación naval.

Ejecutando, el ejemplo 1 incluido en el programa expuesto, en el cual se introduce una función como expresión simbólica.

$$syms\ x,$$
$$[int] = Trapecios\_P1\_Mab(\sin(x), 100, 0, pi)\ .$$

El resultado que nos devuelve el programa es:

$$[int] = 1.9998\ .$$

2.1.4.2. Trapecios Compuesto dada la distancia entre nodos.

En el segundo ejemplo hemos escrito en lenguaje Matlab un código para aproximar la integral de una determinada función, que puede ser introducida como conjunto de datos o como expresión simbólica, teniendo como datos de entrada la función, los extremos del intervalo [a,b] y la distancia entre nodos.

A continuación, se ofrece el código de programación desarrollado al efecto:

```
function [int]=Trapecios_P2_hab(f,h,a,b)

% En este ejercicio calcularemos la integral de una función pasada como una
% expresión simbólica o un conjunto de datos, en este
% problema introduciremos también la distancia entre valores, es decir, la
% distancia entre subintervalos (distancia entre nodos). También
% indicaremos los extremos entre los cuales queremos aproximar la integral.
%
% [int]=Trapecios_P2_hab(f,h,a,b)
%
% Datos de entrada:
% f: función introducida como expresión simbólica o como conjunto de datos
(vector).
% h: distancia entre nodos.
% a,b: extremos del intervalo.
%
% Datos de salida:
% [int]: integral de la función calculada para los datos de entrada (h y a,b).
%
% Ejemplo 1:
% syms x
% [int]=Trapecios_P2_hab(tan(x)+2,0.5,0,10)
%
% Ejemplo 2:
% [int]=Trapecios_P2_hab([2.0000    2.0527    2.1057    2.1592    2.2137
2.2694    2.3267    2.3860    2.4478    2.5126    2.5810    2.6537    2.7315
2.8156    2.9073    3.0082    3.1204    3.2466    3.3906    3.5574],100,0,pi)

% Introducimos los datos de entrada

syms x;

% Ahora pasamos a inicializar las variables

I=0;  % acumulador para el valor de la integral

if ~isfloat(f)

    % Primero debemos calcular el número de subintervalos

    M=ceil((b-a)/h);

    h1=(b-a)/M;
    if abs(h-h1)>10^(-15)
        fprintf('El valor del espaciado h ha sido actualizado a %f',h1);
        h=h1;
    end

    % cálculo de los nodos
    xn(1)=a;
```

```
    xn(M+1)=b;

    for i=2:M
        xn(i)=xn(i-1)+h;
    end

    % Regla de los Trapecios

    % Denotaremos por E a la suma de las ordenadas en los extremos
    E=double(subs(f,a))+double(subs(f,xn(M+1)));

    % Denotaremos por I a la suma de las ordenadas intermedias del método
    for i=2:M
        I=I+double(subs(f,xn(i)));
    end

else

    M=length(f)-1;

    % calculamos el espaciado h entre nodos

    h=(b-a)/M;

    fprintf('\n');
    fprintf('El valor de M ha sido establecido a la longitud del vector de datos
introducido menos una unidad, M= %d',M);

    % Regla de los Trapecios

    % Denotaremos por E a la suma de las ordenadas en los extremos
    E=f(1)+f(M+1);

    % Denotaremos por I a la suma de las ordenadas intermedias del método
    for i=2:M
        I=I+f(i);
    end

end

% Calculando el valor de la aproximación
int=h*((E/2)+I);
```

FINAL DEL PROGRAMA

Al igual que en el anterior programa, se han incluido algunos ejemplos dentro del código del programa que pueden ser utilizados para testar su funcionamiento.

Veamos ahora el ejemplo 2 de este programa, en esta ocasión vamos a introducir la función como un conjunto de datos.

$$[int] = Trapecios\_P2\_hab([2.0000 \quad 2.0527 \quad 2.1057 \quad 2.1592 \quad 2.2137 \quad 2.2694 \quad 2.3267 \quad 2.3860 \quad 2.4478 \quad 2.5126 \quad 2.5810 \quad 2.6537 \quad 2.7315 \quad 2.8156 \quad 2.9073 \quad 3.0082 \quad 3.1204 \quad 3.2466 \quad 3.3906 \quad 3.5574],100,0,pi).$$

El programa nos devuelve los siguientes inputs:

$$El\ valor\ de\ M\ ha\ sido\ establecido\ a\ la\ longitud\ del\ vector\ de\ datos\ introducido\ menos\ una\ unidad, M = 19\ ,$$

$$[int] = 8.2190\ .$$

2.1.4.3. Trapecios Compuesto dada la tolerancia máxima del error

En este tercer ejemplo hemos escrito en lenguaje Matlab un código para aproximar la integral de una determinada función, que será introducida como una expresión simbólica, teniendo como datos de entrada la función $f(x)$, los extremos del intervalo [a,b] y la tolerancia máxima del error, es decir, el error de la aproximación debe ser menor que la tolerancia introducida.

A continuación, se ofrece el código de programación desarrollado al efecto:

```
function [int]=Trapecios_P3_tolab(f,tol,a,b)

% Este programa tiene como objetivo calcular la integral de una función
% introducida como una expresión simbólica, en este caso introduciremos la
% tolerancia que tenemos que cumplir, es decir, el error de la integral
% debe ser menor que esta tolerancia.
% También introduciremos los extremos del intervalo entre los que queremos
% que se evalúe esta función para una determinada tolerancia en el
% intervalo (a,b).
%
% [int]=Trapecios_P3_tolab(f,tol,a,b)
%
% Datos de entrada:
% f: función introducida como una expresión simbólica.
% tol: tolerancia con la que queremos trabajar. Tolerancia máx. del error.
% (a,b): extremos del intervalo.
%
% Datos de salida:
% [int]: integral de la función para los datos de entrada (tol, a y b).
%
% Ejemplo 1:
% syms x
% [int]=Trapecios_P3_tolab(cos(x+2)+3,10^(-3),0,50)
% Ejemplo 2:
% syms x
% [int]=Trapecios_P3_tolab(sin(x)*(1-x),10^(-4),0,20)

% Primero realizamos el cálculo aproximado del máximo de la segunda derivada
% de la función  en un intervalo [a,b] evaluando en un mallado fino

syms x;

nx=1000;
hx=(b-a)/nx;

% Para hallar el máximo necesitamos conocer la segunda derivada de f

xe=zeros(nx);   % Predefinimos los arrays para reducir cálculos
vd2f=zeros(nx);

d1=diff(f);
d2=diff(d1);
xe(1)=a;
vd2f(1)=double(subs(d2,x,xe(1)));  % valor de la segunda derivada en el primer
punto del mallado
maxd2=abs(vd2f(1));


% Ahora calculamos el máximo mediante un mallado fino
```

```
for i=2:nx
    xe(i)=xe(i-1)+hx;
    vd2f(i)=double(subs(d2,x,xe(i)));
    if abs(vd2f(i))> maxd2   % si el siguiente punto es mayor que maxd2, lo
tomamos como máximo y así sucesivamente
        maxd2=abs(vd2f(i));
    end
end

% Tomamos un margen de error en la estimación del máx.
maxd2=maxd2+0.1;

% Utilizamos la fórmula del error para hallar n
M=sqrt((b-a)^3/12/tol*maxd2);

% Redondeamos
M=ceil(M);

fprintf('\n');
fprintf('La tolerancia introducida como margen de error se cumple para M=
%d\n',M)

% Ahora aplicamos la Regla de los Trapecios, utilizando el mismo método que
% en el P1, ya que ahora tenemos los mismos datos de partida (M, a y b)

% Antes de inicializar las variables necesitamos calcular la distancia
% entre nodos

h=(b-a)/M;

% Ahora pasamos a inicializar las variables

I=0;  % acumulador para el valor de la integral

% cálculo de los nodos
xn(1)=a;
xn(M+1)=b;

for i=2:M
    xn(i)=xn(i-1)+h;
end

% Regla de los Trapecios

% Denotaremos por E a la suma de las ordenadas en los extremos
E=double(subs(f,xn(1)))+double(subs(f,xn(M+1)));

% Denotaremos por I a la suma de las ordenadas intermedias del método
for i=2:M
    I=I+double(subs(f,xn(i)));
end

% Calculando el valor de la aproximación
[int]=h*((E/2)+I);

end
```

FINAL DEL PROGRAMA

También en esta ocasión, se han incluido algunos ejemplos en el código del programa que pueden ser utilizados para testar el correcto funcionamiento del programa.

Veamos ahora el ejemplo 1 de este programa, en esta ocasión introducida la función como expresión simbólica.

$$syms\ x,$$
$$[int] = Trapecios\_P3\_tolab(\cos(x+2)+3, 10\text{^}(-3), 0, 50)\ .$$

El programa nos devolverá los siguientes inputs:

$$La\ tolerancia\ introducida\ como\ margen\ de\ error\ se\ cumple\ para\ M = 3386,$$

$$[int] = 150.0773\ .$$

2.1.4.4. Trapecios Compuesto exportando datos de un fichero Excel

En este último ejemplo hemos escrito en lenguaje Matlab un código para aproximar la integral de una determinada función, que en esta ocasión será un conjunto de datos que exportaremos de un fichero Excel, el otro dato de entrada que debemos introducir es el rango de datos con los que queremos trabajar dentro del fichero.

A continuación, se ofrece el código de programación desarrollado al efecto:

```
function [int]=Trapecios_P4_exph(fichero,rango)

% En este ejercicio vamos a calcular la integral de una función introducida
% como un conjunto de datos exportados de una tabla Excel que debe estar
% guardada en la misma carpeta que el programa de Matlab (importante: si
% el archivo Excel no está guardado en la misma carpeta que el programa,
% éste no funcionará), también introduciremos el rango de valores con los
% que queremos trabajar dentro del fichero.
%
% [int]=Trapecios_P4_exph(fichero,rango)
%
% Datos de entrada:
% fichero: nombre del fichero que contiene el conjunto de datos en una
% tabla de Excel.
% rango: rango de valores con los que vamos a trabajar dentro del fichero.
%
% Datos de salida:
% [int]: resultado final de la integral.
%
% Ejemplo 1:
% fichero='DatosaexportarTRAPECIOS.xlsx';
% rango='A2:B12';
% [int]=Trapecios_P4_exph(fichero,rango)

A=xlsread(fichero,rango);

[M,Mc]=size(A);

h=A(1,2)-A(1,1);

I=0;

% Regla de los Trapecios

% Denotaremos por E a la suma de las ordenadas en los extremos
E=A(M,1)+A(M,Mc);

% Denotaremos por I a la suma de las ordenadas intermedias del método
for i=2:Mc-1
    I=I+A(M,i);
end

% Calculando el valor de la aproximación
int=h*((E/2)+I);
```

FINAL DEL PROGRAMA

Al igual que en los anteriores programas, se ha incluido un ejemplo que puede ser utilizado para testar su funcionamiento.

Veamos ahora el ejemplo de este programa, en esta ocasión los datos son exportados de un fichero Excel. Como hemos explicado anteriormente, debemos introducir el fichero y el rango de valores.

$$fichero = DatosaexportarTRAPECIOS,$$

$$rango =' A2:B12' ,$$

$$[int] = Trapecios\_P4\_exph(fichero, rango) .$$

Siendo el resultado devuelto por el programa:

$$[int] = 6.8562\ e + 04 .$$

2.2. REGLA DE SIMPSON

La regla de integración numérica de Simpson es un método numérico utilizado para aproximar el valor de una integral definida de una función en un intervalo especificado. Este método es más preciso que la Regla del Trapecio, ya que emplea una aproximación cuadrática de la función en lugar de una lineal.

La regla de integración numérica de Simpson es un método preciso y útil para aproximar el valor de una integral definida de una función en un intervalo dado.

2.2.1. Regla de Simpson 1/3

La Regla de Simpson 1/3 nos proporciona una aproximación bastante más exacta que el método de los trapecios y un error relativamente pequeño, esto la convierte en una de las reglas más utilizadas para aproximación de integrales. En este método tenemos los puntos a y b y un punto intermedio en el centro de este intervalo *(a,b)*, creando una parábola de 3 puntos que hará la aproximación más exacta.

Por lo que el área bajo la curva queda aproximada por el área bajo la parábola definida por los 3 puntos $((a,f(a)),\left(\frac{a+b}{2},f\left(\frac{a+b}{2}\right)\right),(b,f(b)))$ como vemos representado en la *Figura 12*.

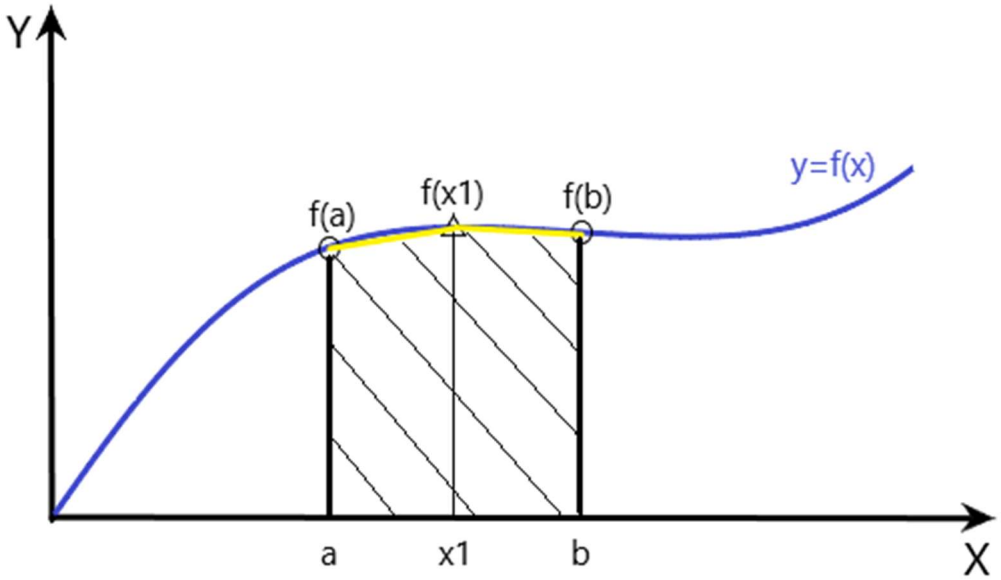


Figura 12. Representación gráfica de la Regla de Simpson 1/3 simple.

Aplicando el Polinomio Interpolador de Lagrange, la ecuación que obtenemos y que representa esta regla es:

$$p_n(x) = L_0(x)f(x_0) + L_1(x)f(x_1) + L_2(x)f(x_2),$$

quedando finalmente de la forma:

$$\int_a^b f(x)dx \approx \frac{h}{3}(f(x_0) + 4f(x_1) + f(x_2)).$$

Esta regla de Simpson 1/3 compuesta, al igual que la Regla de los Trapecios. puede aplicarse de forma múltiple para así aumentar la precisión de la aproximación, tanto más, cuantos más subintervalos tengamos dentro del intervalo de integración (ver *Figura 13*).

Para que esta regla se pueda llevar a cabo, es imperativo que el dominio de la función esté dividido en un número par de subintervalos. Definiendo $h = \frac{(b-a)}{2n}$, $x_i = a + ih$, $i = 0, \dots, n$ donde n es el número de veces que aplicamos la Regla de Simpson 1/3 simple.

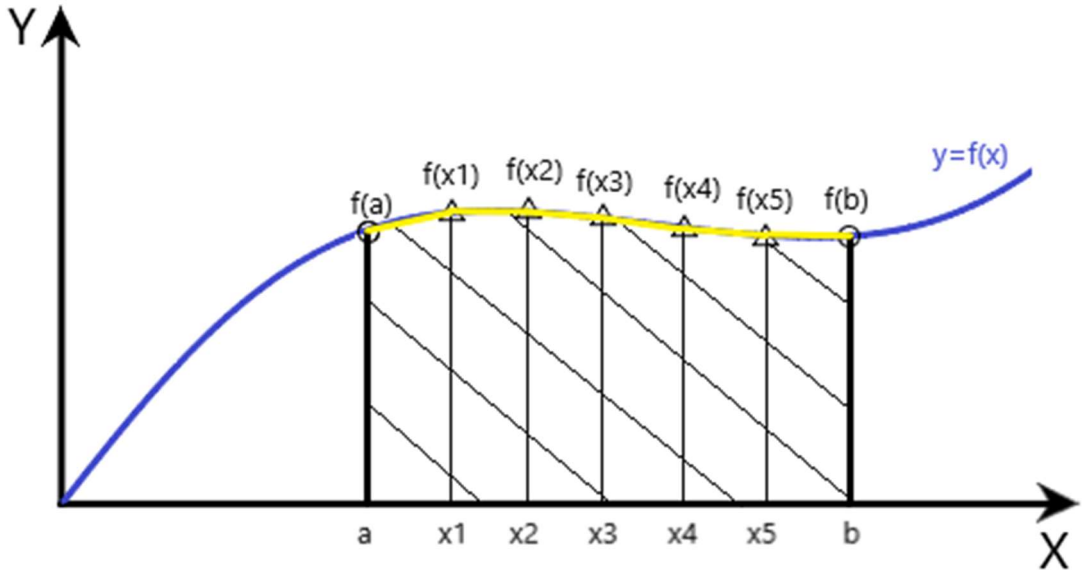


Figura 13: Representación gráfica de la Regla de Simpson 1/3 compuesta.

La aproximación de la fórmula de la Regla de Simpson 1/3 compuesta quedará como sigue:

$$\int_a^b f(x)dx \approx \frac{h}{3}\big(f(x_0) + 4f(x_1) + f(x_2)\big) + \frac{h}{3}\big(f(x_2) + 4f(x_3) + f(x_4)\big) + \cdots$$
$$\dots + \frac{h}{3}\big(f(x_{2n-2}) + 4f(x_{2n-1}) + f(x_{2n})\big) = \frac{h}{3}(E + 4I + 2P),$$

simplificando la ecuación,

$$\int_a^b f(x)dx \approx \frac{h}{3}(E + 4I + 2P),$$

dónde:

$$E = f(x_0) + f(x_n),$$
$$I = f(x_1) + f(x_3) + \cdots + f(x_{2n-1}),$$
$$P = f(x_2) + f(x_4) + \cdots + f(x_{2n-2}).$$

2.2.2. Regla de Simpson 3/8

En esta regla, nuestro objetivo es determinar el área bajo una función aproximándola mediante un polinomio de grado 3, utilizando para ello 4 puntos. (los dos extremos y dos puntos intermedios equidistantes) como podemos observar en la *Figura 14*.

Este método se aplica cuando el número de puntos que tenemos es par, o lo que es lo mismo, cuando el número de subintervalos es impar y no se puede utilizar la regla de Simpson 1/3 . También es posible combinar ambas reglas, como veremos más adelante en los ejercicios que hemos programado en Matlab.

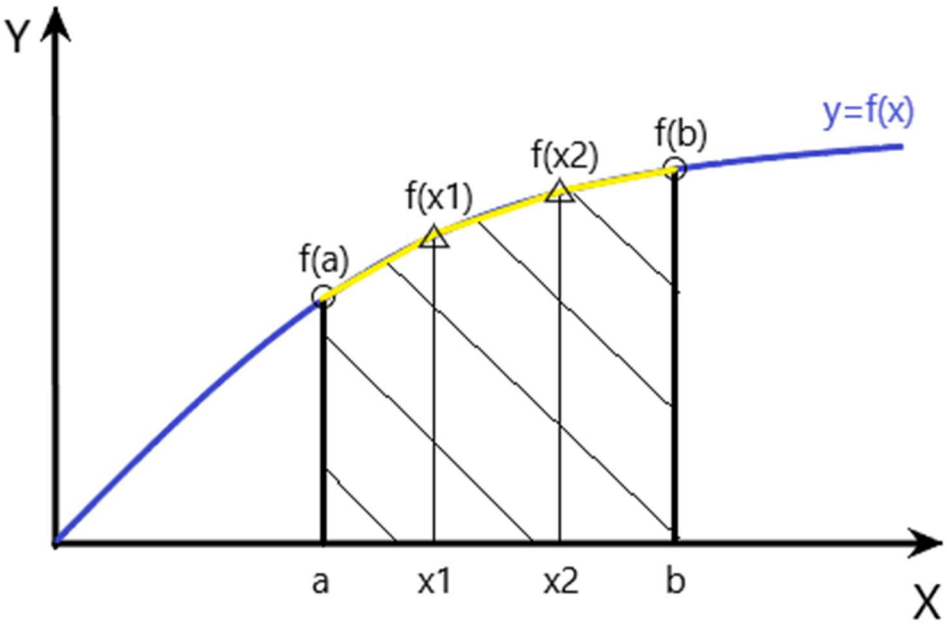


Figura 14. Representación gráfica de la Regla de Simpson 3/8.

Para obtener la fórmula de esta Regla de Simpson de 3/8 tenemos que ajustar un polinomio de Lagrange de tercer grado a 4 puntos e integrarlo.

El polinomio de Lagrange de grado 3 por lo tanto sería:

$$p_3(x) = L_0(x)f(x_0) + L_1(x)f(x_1) + L_2(x)f(x_2) + L_3(x)f(x_3),$$

e integrándolo nos daría la expresión:

$$\int_a^b f(x)dx \approx \frac{3}{8}h\big(f(x_0) + 3f(x_1) + 3f(x_1) + f(x_2)\big).$$

2.2.3. Error de la Regla de Simpson.

El error en la Regla de Simpson se refiere a la diferencia entre el valor exacto de la integral y el valor aproximado obtenido mediante este método de integración. Este error es dependiente de varios aspectos, como el intervalo de integración, la cantidad de puntos de integración y la función que se está integrando.

En general, el error de la Regla de Simpson disminuye cuando se aumenta el número de puntos de integración. Sin embargo, existe un límite en el que depende de la precisión con la que se trabaje en el

ordenador (usualmente 32 dígitos significativos) tal que a partir de ese límite aumentar el número de puntos ya no disminuye significativamente el error y se vuelve ineficiente computacionalmente.

La fórmula del error de la Regla de Simpson 1/3 simple es:

$$E_h(f) = \frac{f^{IV}(\mu)}{90}\left(\frac{(b-a)}{2}\right)^5,$$

cómo se puede ver en la fórmula, el error de Simpson tiene un grado de exactitud de 3 y un orden de aproximación local de 5, lo que representa un incremento de 2 en comparación con la Regla de los Trapecios.

El error de la Regla de Simpson 1/3 compuesta será:

$$E = -\frac{(b-a)}{180}h^4 f^{IV}(\mu)\,.$$

En este caso tendremos un grado de exactitud 3 con un orden de aproximación global 4 (indicado por la potencia de h).

La fórmula del error para la Regla de Simpson 3/8 simple es:

$$E(f) = -\frac{3}{80}h^5 f^{IV}(\mu)\,,$$

con $\mu \in (a, b)$.

Observamos que se trata de un error con un grado de exactitud 3, igual que en Simpson 1/3, y con un orden de aproximación local 5, también igual que en Simpson 1/3.

La fórmula del error de la Regla de Simpson 3/8 compuesta queda:

$$E_h(f) = -\frac{(b-a)}{80}h^4 f^{IV}(\mu)\,,$$

con $\mu \in (a, b)$.

Esta fórmula del error tiene un grado de exactitud 3 y un orden de aproximación global 4.

Es importante destacar que el orden de aproximación en la fórmula compuesta, como siempre sucede, se reduce en una unidad en comparación con el obtenido en la Regla de Simpson 3/8 simple, como ocurre también en los demás métodos.

Es importante destacar que la regla de Simpson es una técnica de integración numérica que proporciona una aproximación del valor de la integral, y no el valor exacto de la misma. No obstante, las fórmulas del error en prácticamente todos los métodos tienen una utilidad más cualitativa que práctica.

2.2.4 Método de Simpson programado en lenguaje Matlab

Ahora vamos a ver los 4 programas en lenguaje Matlab que hemos realizado, en los que introduciremos diferentes datos de entrada para llegar a la aproximación de la integral de una función dada. en ellos aplicaremos la Regla de Simpson.

2.2.41. Simpson dado el número de subintervalos

En este ejemplo hemos escrito en lenguaje Matlab un código para aproximar la integral mediante el método de Simpson de una determinada función, que puede ser introducida como conjunto de datos o como expresión simbólica. Teniendo como datos de entrada la función, los extremos del intervalo [a,b] y el número de subintervalos a aplicar.

A continuación, se ofrece el código de programación desarrollado al efecto:

```
function [int]=Simpson_P1_Mab(f,M,a,b)

% Este programa tiene como objetivo calcular la integral mediante el método
% de Simpson de una determinada función introducida como una expresión
% simbólica o un conjunto de datos para lo cual definiremos
% el número total de subintervalos y los extremos del intervalo donde
% queremos aproximar la integral.
%
%[int]=Simpson_P1_Mab(f,M,a,b)
%
% Datos de entrada:
% f: función introducida como expresión simbólica o como conjunto de datos.
% M: número de subintervalos.
% a,b: extremos del intervalo.
%
% Datos de salida:
% [int]: integral de la función calculada para los datos de entrada (M y a,b).
%
% Ejemplo 1:
% syms x
% [int]=Simpson_P1_Mab(sin(x),100,0,3*pi/4)
% Ejemplo 2:
% [int]=Simpson_P1_Mab([1.0000    1.0714    1.1424    1.2126    1.2818    1.3496
1.4156    1.4794    1.5408    1.5995    1.6551    1.7073    1.7560    1.8008
1.8415],100,0,pi)

if ~isfloat(f) % Caso de la función introducida como simbólica

    % Inicialización de variables

    syms x

    % Calculamos la distancia entre intervalos

    h=(b-a)/M;

    % Cálculo de los nodos

    xn(1)=a;
    xn(M+1)=b;

    for i=2:M
```

```
    xn(i)=xn(i-1)+h;
end

% Ahora incluimos un condicional con el que veremos si aplicamos la regla
% 1/3 o la de 1/3 y 3/8 juntas

if rem(M,2)==0

    % Si es par el número de subintervalos que tenemos, debemos aplicar la
    % regla de Simpson 3/8 en el último intervalo de datos, ya que si solo
    % aplicáramos la regla de 1/3 el último dato quedaría fuera de la aprox.

    % Aplicaremos la regla de 3/8 a los 4 últimos nodos

    int1=(3*h/8)*(double(subs(f,xn(M-2)))+3*double(subs(f,xn(M-1)))+...
        3*double(subs(f,xn(M)))+double(subs(f,xn(M+1))));

    % Ahora pasamos a calcular el resto de los valores mediante Simpson 1/3

    % Inicialización de los acumuladores

    Ii=0;
    Ip=0;

    % Denotaremos por E a la suma de los valores de los extremos

    E=double(subs(f,xn(1)))+double(subs(f,(M-2)));

    % Denotaremos por Ii a la suma de las ordenadas intermedias impares
    % (pares en Matlab) en el método

    for i=2:2:M-3
        Ii=Ii+double(subs(f,xn(i)));
    end

    % Denotaremos por Ip a la suma de las ordenadas intermedias pares
    % (impares en Matlab) en el método

    for i=3:2:M-4
        Ip=Ip+double(subs(f,xn(i)));
    end

    % Calculando el valor de la aproximación

    int2=(h/3)*(E+4*Ii+2*Ip);

    % Finalmente sumamos las dos partes y obtenemos la integral completa

    [int]=int1+int2;

else % Aplicamos únicamente la regla de 1/3

    % Regla de Simpson 1/3

    % Inicialización de los acumuladores

    Ii=0;
```

```
        Ip=0;

        % Denotaremos por E a la suma de los valores de los extremos

        E=double(subs(f,xn(1)))+double(subs(f,xn(M+1)));

        % Denotaremos por Ii a la suma de las ordenadas intermedias impares
        % (pares en Matlab) en el método

        for i=2:2:M
            Ii=Ii+double(subs(f,xn(i)));
        end

        % Denotaremos por Ii a la suma de las ordenadas intermedias pares
        % (impares en Matlab) en el método

        for i=3:2:2*M-1
            Ip=Ip+double(subs(f,xn(i)));
        end

        % Calculando el valor de la aproximación

        [int]=(h/3)*(E+4*Ii+2*Ip);

    end

else % función introducida como conjunto de datos

    if length(f)~=M+1;

        str1='This is a warning';
        str2=['El número de datos introducidos no coincide con
',num2str(length(f)),...
            ' que es el número de datos de que disponemos. Por lo tanto vamos a
trabajar  ',num2str(length(f)),' puntos'];
        mydlg=warndlg(str2, str1);
        waitfor(mydlg);

        M=length(f)-1;
        h=(b-a)/M;
    end

    % Vemos si hay que aplicar la regla 1/3 o las reglas 1/3 y 3/8 juntas

    if rem(M,2)==0

        % Si es par el número de subintervalos que tenemos, debemos aplicar la
        % regla de Simpson 3/8 en el último intervalo de datos, ya que si solo
        % aplicáramos la regla de 1/3 el último dato quedaría fuera de la aprox.

        % Aplicaremos la regla de 3/8 a los 4 últimos nodos

        int1=(3*h/8)*(f(M-2)+3*f(M-1)+3*f(M)+f(M+1));

        % Ahora pasamos a calcular el resto de valores mediante Simpson 1/3

        % Inicialización de variables
```

```
    Ii=0;
    Ip=0;

    % Denotaremos por E a la suma de los valores de los extremos

    E=f(1)+f(M-2);

    % Denotaremos por Ii a la suma de las ordenadas intermedias impares
    % (pares en Matlab) en el método

    for i=2:2:M-3
        Ii=Ii+f(i);
    end

    % Denotaremos por Ii a la suma de las ordenadas intermedias pares
    % (impares en Matlab) en el método

    for i=3:2:M-4
        Ip=Ip+f(i);
    end

    % Calculando el valor de la aproximación

    int2=(h/3)*(E+4*Ii+2*Ip);

    % Finalmente sumamos las dos partes y obtenemos la integral completa

    [int]=int1+int2;

else % Aplicamos únicamente la regla de 1/3

    % Inicialización de variables

    Ii=0;
    Ip=0;

    % Regla de Simpson 1/3

    % Denotaremos por E a la suma de los valores de los extremos

    E=f(1)+f(M+1);

    % Denotaremos por Ii a la suma de las ordenadas intermedias impares
    % (pares en Matlab) en el método

    for i=2:2:M
        Ii=Ii+f(i);
    end

    % Denotaremos por Ii a la suma de las ordenadas intermedias pares
    % (impares en Matlab) en el método

    for i=3:2:M-1
        Ip=Ip+f(i);
    end
```

```
        % Calculando el valor de la aproximación

        [int]=(h/3)*(E+4*Ii+2*Ip);

    end

end

end
```

FINAL DEL PROGRAMA

Dentro del código del propio programa de Matlab se han incluido algunos ejemplos que pueden ser utilizados para testar su funcionamiento.

Ejecutando, el ejemplo 2 incluido en el programa expuesto, esta vez vamos a introducir la función como un conjunto de datos, aunque el programa está diseñado para trabajar con una expresión simbólica si así lo deseamos.

$$[int] = Simpson\_P1\_Mab([1.0000 \quad 1.0714 \quad 1.1424 \quad 1.2126 \quad 1.2818 \quad 1.3496 \quad 1.4156 \quad 1.4794 \quad 1.5408 \quad 1.5995 \quad 1.6551 \quad 1.7073 \quad 1.7560 \quad 1.8008 \quad 1.8415],100,0,pi).$$

Al introducir el programa nos salta un cuadro "warning" debido a que el número de subintervalos introducido no coincide con el número de datos introducido, por lo que nos avisa que trabajaremos con el nuevo número de datos introducidos.

Inputs recibidos de Matlab:

This is a warning:
El número de datos introducidos no coincide con 15 que es el número de datos de que disponemos.
Por lo tanto, vamos a trabajar con 15 puntos.

$$[int] = 4.2124 \ .$$

2.2.4.2. Simpson dada la distancia entre nodos

En este ejemplo hemos escrito en lenguaje Matlab un código para aproximar la integral mediante el método de Simpson de una determinada función, que puede ser introducida como conjunto de datos o como expresión simbólica. Teniendo como datos de entrada la función, los extremos del intervalo [a,b] y la distancia entre nodos.

A continuación, se ofrece el código de programación desarrollado al efecto:

```
function [int]=Simpson_P2_hab(f,h,a,b)

% Este programa tiene como objetivo calcular la integral mediante el método
% de Simpson de una determinada función introducida como una expresión
% simbólica o un conjunto de datos para la cual definiremos
% la longitud de los subintervalos y los extremos del intervalo donde
% queremos aproximar la integral.
%
% [int]=Simpson_P2_hab(f,h,a,b)
%
% Datos de entrada
% f: función introducida como expresión simbólica o como conjunto de datos.
% h: distancia entre nodos.
% a,b: extremos del intervalo.
%
% Datos de salida:
% [int]: integral de la función calculada para los datos de entrada (M y a,b).
%
% Ejemplo 1:
% syms x
% [int]=Simpson_P2_hab(cos(x)+2*x,0.1,0,pi)
% Ejemplo 2:
% syms x
% [int]=Simpson_P2_hab(sin(x),4,0,100)
% Ejemplo 3:
% [int]=Simpson_P2_hab([1.0000    1.2161    1.4199    1.6116    1.7917    1.9607
2.1192    2.2680    2.4081    2.5403],20,0,100)

% Introducimos los datos de entrada

syms x

if ~isfloat(f)  % Caso de la función introducida como simbólica

    % Primero calculamos el número de subintervalos que tenemos

    M=ceil((b-a)/h);

    h1=(b-a)/M;
    if abs(h-h1)>10^(-15)
        fprintf('El valor del espaciado h ha sido actualizado a %f',h1);
        h=h1;
    end

    % Cálculo de los nodos

    xn(1)=a;
    xn(M+1)=b;
```

```
    for i=2:M
        xn(i)=xn(i-1)+h;
    end

    % Ahora incluimos un condicional con el que veremos si aplicamos la regla
    % 1/3 o la de 1/3 y 3/8 juntas

    if rem(M,2)==0

        % Si es par el número de subintervalos que tenemos, debemos aplicar la
        % regla de Simpson 3/8 en el último intervalo de datos, ya que si solo
        % aplicáramos la regla de 1/3 el último dato quedaría fuera de la aprox.

        % Aplicaremos la regla de 3/8 a los 4 últimos nodos

        int1=(3*h/8)*(double(subs(f,xn(M-2)))+3*double(subs(f,xn(M-1)))+...
            3*double(subs(f,xn(M)))+double(subs(f,xn(M+1))));

        % Ahora pasamos a calcular el resto de los valores mediante Simpson 1/3

        % Inicializamos los acumuladores

        Ii=0;
        Ip=0;

        % Denotaremos por E a la suma de los valores de los extremos

        E=double(subs(f,xn(1)))+double(subs(f,xn(M-2)));

        % Denotaremos por Ii a la suma de las ordenadas intermedias impares
        % (pares en Matlab) en el método

        for i=2:2:M-3
            Ii=Ii+double(subs(f,xn(i)));
        end

        % Denotaremos por Ip a la suma de las ordenadas intermedias pares
        % (impares en Matlab) en el método

        for i=3:2:M-4
            Ip=Ip+double(subs(f,xn(i)));
        end

        % Calculando el valor de la aproximación

        int2=(h/3)*(E+4*Ii+2*Ip);

        % Finalmente sumamos las dos partes y tenemos el resultado de la
integral

        [int]=int1+int2;

    else % Aplicamos únicamente la regla de 1/3

        % Inicialización de acumuladores

        Ii=0;
```

```
        Ip=0;

        % Regla de Simpson 1/3

        % Denotaremos por E a la suma de los valores de los extremos

        E=double(subs(f,xn(1)))+double(subs(f,xn(2*(M+1))));

        % Denotaremos por Ii a la suma de las ordenadas intermedias impares
        % (pares en Matlab) en el método

        for i=2:2:M
            Ii=Ii+double(subs(f,xn(i)));
        end

        % Denotaremos por Ii a la suma de las ordenadas intermedias pares
        % (impares en Matlab) en el método

        for i=3:2:2*M-1
            Ip=Ip+double(subs(f,xn(i)));
        end

        % Calculando el valor de la aproximación

        [int]=(h/3)*(E+4*Ii+2*Ip);

    end

else % función introducida como conjunto de datos

    % calculamos el número de subintervalos que tenemos segun los datos de
    % entrada (extremos del intervalo [a,b] y distancia entre nodos).

    M=ceil((b-a)/h);

    if length(f)~=M+1;

        str1='This is a warning';
        str2=['El número de datos introducidos no coincide
con',num2str(length(f)),...
            'que es el número de datos de que disponemos. Por lo tanto vamos a
trabajar con' ,num2str(length(f)),'puntos'];
        mydlg=warndlg(str2, str1);
        waitfor(mydlg);

        M=length(f)-1;
        h=(b-a)/M;
    end

    % Vemos si hay que aplicar la regla 1/3 o las reglas 3/8 y 1/3 juntas

    if rem(M,2)==0

        % Si es par el número de subintervalos que tenemos, debemos aplicar la
        % regla de Simpson 3/8 en el último intervalo de datos, ya que si solo
        % aplicáramos la regla de 1/3 el último dato quedaría fuera de la aprox.
```

```
    % Aplicaremos la regla de 3/8 a los 4 últimos nodos

    int1=(3*h/8)*(f(M-2)+3*f(M-1)+3*f(M)+f(M+1));

    % Ahora pasamos a calcular el resto de los valores mediante Simpson 1/3

    % Inicialización de acumuladores

    Ii=0;
    Ip=0;

    % Denotaremos por E a la suma de los valores de los extremos

    E=f(1)+f(M-2);

    % Denotaremos por Ii a la suma de las ordenadas intermedias impares
    % (pares en Matlab) en el método

    for i=2:2:M-3
        Ii=Ii+f(i);
    end

    % Denotaremos por Ii a la suma de las ordenadas intermedias pares
    % (impares en Matlab) en el método

    for i=3:2:M-4
        Ip=Ip+f(i);
    end

    % Calculando el valor de la aproximación

    int2=(h/3)*(E+4*Ii+2*Ip);

    % Finalmente sumamos las dos partes y obtenemos la integral completa

    [int]=int1+int2;

else % Aplicamos únicamente la regla de 1/3

    % Inicialización de acumuladores

    Ii=0;
    Ip=0;

    % Regla de Simpson 1/3

    % Denotaremos por E a la suma de los valores de los extremos

    E=f(1)+f(M+1);

    % Denotaremos por Ii a la suma de las ordenadas intermedias impares
    % (pares en Matlab) en el método

    for i=2:2:M
        Ii=Ii+f(i);
    end
```

```
        % Denotaremos por Ii a la suma de las ordenadas intermedias pares
        % (impares en Matlab) en el método

        for i=3:2:M-1
            Ip=Ip+f(i);
        end

        % Calculando el valor de la aproximación

        [int]=(h/3)*(E+4*Ii+2*Ip);

    end

end

end
```

FINAL DEL PROGRAMA

Vamos a ejecutar uno de los ejemplos que hay incluidos en el código de Matlab.

Ejecutando el ejemplo 1 en el cual introducimos la función como expresión simbólica tenemos.

$$syms\ x$$
$$[int] = Simpson\_P2\_hab(cos(x), 1{,}0, pi),$$

Y los inputs recibidos del programa:

$$El\ valor\ del\ espaciado\ h\ ha\ sido\ actualizado\ a\ 0.098175,$$

$$[int] = 9.4180\ .$$

2.2.4.3. Simpson dada la tolerancia máxima del error

En este tercer ejemplo hemos escrito en lenguaje Matlab un código para aproximar la integral de una determinada función mediante el método de Simpson introducida como una expresión simbólica. Teniendo como datos de entrada la función $f(x)$, los extremos del intervalo [a,b] y la tolerancia máxima del error, es decir, el error de la aproximación debe ser menor que la tolerancia introducida.

A continuación, se ofrece el código de programación desarrollado al efecto:

```
function [int]=Simpson_P3_tolab(f,tol,a,b)

% Este programa tiene como objetivo calcular la integral de una función
% introducida como una expresión simbólica, en este caso introduciremos una
% tolerancia que tenemos que cumplir, es decir que el error de la integral
% debe ser menor que esta tolerancia.
% También introduciremos los extremos del intervalo entre los que queremos
% que se evalúe esta función para una determinada tolerancia en el
% intervalo [a,b].
%
% [int]=Simpson_P3_tolab(f,tol,a,b)
%
% Datos de entrada:
% f: función introducida como una expresión simbólica.
% tol: tolerancia con la que queremos trabajar. Tolerancia máx. del error.
% [a,b]: extremos del intervalo.
%
% Datos de salida:
% [int]: integral de la función para los datos de entrada (tol, a y b).
%
% Ejemplo 1:
% syms x
% [int]=Simpson_P3_tolab(cos(x+2)+3,10^(-3),0,50)
% Ejemplo 2:
% syms x
% [int]=Simpson_P3_tolab(sin(x)+2*x+1,10^(-5),0,10)

% Primero realizamos el cálculo aproximado del máximo de la segunda derivada
% de la función en un intervalo [a,b] evaluando en un mallado fino

syms x;

nx=1000;
hx=(b-a)/nx;

% Para hallar el máximo necesitamos conocer la segunda derivada de f

xe=zeros(nx);   % Predefinimos los arrays para reducir cálculos
vd2f=zeros(nx);

% Calculamos la segunda derivada y asignamos el máximo al primer punto
% del mallado como valor inicial para el bucle que sigue

d1=diff(f);
d2=diff(d1);
xe(1)=a;
vd2f(1)=double(subs(d2,x,xe(1)));
maxd2=abs(vd2f(1));
```

```
% Ahora calculamos el máximo mediante un mallado fino, comparando cada
% valor con el punto siguiente y quedándonos con el mayor de ellos

for i=2:nx
    xe(i)=xe(i-1)+hx;
    vd2f(i)=double(subs(d2,x,xe(i)));
    if abs(vd2f(i))> maxd2
        maxd2=abs(vd2f(i));
    end
end

% Tomamos un margen de error en la estimación del máx.

maxd2=maxd2+0.1;

% Utilizamos la fórmula del error para hallar M

M=sqrt((b-a)^3/12/tol*maxd2);

% Redondeamos

M=ceil(M);

fprintf('\n');
fprintf('La tolerancia introducida como margen de error se cumple para M=
%d\n',M)

% Ahora aplicamos Simpson, utilizando el mismo método que en Simpson_P1_Mab,
% ya que ahora tenemos los datos de partida de ese caso (M, a y b)

% Calculamos la distancia entre intervalos

h=(b-a)/M;

% Cálculo de los nodos

xn(1)=a;
xn(M+1)=b;

for i=2:M
    xn(i)=xn(i-1)+h;
end

% Ahora incluimos un condicional con el que veremos si aplicamos la regla
% 1/3 o la de 3/8 y 1/3 juntas

if rem(M,2)==0

    % Si es par el número de subintervalos que tenemos, debemos aplicar la
    % regla de Simpson 3/8 en el último intervalo de datos, ya que si solo
    % aplicáramos la regla de 1/3 el último dato quedaría fuera de la aprox.

    % Aplicaremos la regla de 3/8 a los 4 últimos nodos

    int1=(3*h/8)*(double(subs(f,xn(M-2)))+3*double(subs(f,xn(M-1)))+...
        3*double(subs(f,xn(M)))+double(subs(f,xn(M+1))));
```

```
    % Ahora pasamos a calcular el resto de los valores mediante Simpson 1/3

    % Inicialización de los acumuladores

    Ii=0;
    Ip=0;

    % Denotaremos por E a la suma de los valores de los extremos

    E=double(subs(f,xn(1)))+double(subs(f,xn(M-2)));

    % Denotaremos por Ii a la suma de las ordenadas intermedias impares
    % (pares en Matlab) en el método

    for i=2:2:M-3
        Ii=Ii+double(subs(f,xn(i)));
    end

    % Denotaremos por Ip a la suma de las ordenadas intermedias pares
    % (impares en Matlab) en el método

    for i=3:2:M-4
        Ip=Ip+double(subs(f,xn(i)));
    end

    % Calculando el valor de la aproximación

    int2=(h/3)*(E+4*Ii+2*Ip);

    % Finalmente sumamos las dos partes y obtenemos la integral completa

    [int]=int1+int2;

else % Aplicamos únicamente la regla de 1/3

    % Inicialización de los acumuladores

    Ii=0;
    Ip=0;

    % Regla de Simpson 1/3

    % Denotaremos por E a la suma de los valores de los extremos

    E=double(subs(f,xn(1)))+double(subs(f,xn(2*(M+1))));

    % Denotaremos por Ii a la suma de las ordenadas intermedias impares
    % (pares en Matlab) en el método

    for i=2:2:M
        Ii=Ii+double(subs(f,xn(i)));
    end

    % Denotaremos por Ii a la suma de las ordenadas intermedias pares
    % (impares en Matlab) en el método

    for i=3:2:2*M-1
```

```
        Ip=Ip+double(subs(f,xn(i)));
    end

    % Calculando el valor de la aproximación

    [int]=(h/3)*(E+4*Ii+2*Ip);

end

end
```

FINAL DEL PROGRAMA

También en esta ocasión, se han incluido algunos ejemplos que pueden ser utilizados para testar su funcionamiento. Igualmente se ha incluido un ejemplo de aplicación naval.

Veamos ahora el ejemplo 1 de este programa, en esta ocasión introducida la función como expresión simbólica.

$$syms\ x\ ,$$
$$[int] = Simpson\_P3\_tolab(cos(x+2) + 3{,}10\text{^}(-3){,}0{,}50)\ .$$

El programa nos devolverá los siguientes inputs:

La tolerancia introducida como margen de error se cumple para M = 3386,

$$[int] = 150.0346\ .$$

2.2.4.4. Simpson exportando datos de un fichero Excel

En este último ejemplo hemos escrito en lenguaje Matlab un código para aproximar la integral de una determinada función, que en esta ocasión será un conjunto de datos que exportaremos de un fichero Excel, el otro dato de entrada que debemos introducir es el rango de datos con los que queremos trabajar dentro del fichero.

A continuación, se ofrece el código de programación desarrollado al efecto:

```
function [int]=Simpson_P4_exph(fichero,rango)

% En este ejercicio vamos a calcular la integral de una función introducida
% como un conjunto de datos exportados de una tabla Excel que debe estar
% guardada en la misma carpeta que el programa de Matlab (importante: si
% el archivo Excel no está guardado en la misma carpeta que el programa,
% éste no funcionará), este conjunto de datos exportados se evaluará en el
% intervalo a,b.
%
% [int]=Simpson_P4_exph(fichero,rango)
%
% Datos de entrada:
% fichero: nombre del fichero en formato Excel que contiene los datos.
% rango: rango en el que trabajaremos dentro del fichero.
%
% Datos de salida:
% [int]: resultado final de la integral.
%
% Ejemplo:
% fichero='DatosaexportarSIMPSON.xlsx';
% rango='A2:B12';
% [int]=Simpson_P4_exph(fichero,rango)

A=xlsread(fichero,rango);

[M,Mc]=size(A);

h=A(2,1)-A(1,1);

a=A(1,2);    % a=xlsread(fichero,'B2'); me da error
b=A(M,2);    % b=xlsread(fichero,'B11');  me da error

% Vemos si hay que aplicar la regla 1/3 o las reglas 3/8 y 1/3 juntas

if rem(M,2)==0

    % Si es par el número de subintervalos que tenemos, debemos aplicar la
    % regla de Simpson 3/8 en el último intervalo de datos, ya que si solo
    % aplicáramos la regla de 1/3 el último dato quedaría fuera de la aprox.

    % Aplicaremos la regla de 3/8 a los 4 últimos nodos

    int1=(3*h/8)*(A((M-3),Mc)+3*A((M-2),Mc)+3*A((M-1),Mc)+A(M,Mc));

    % Ahora pasamos a calcular el resto de los valores mediante Simpson 1/3

    % Inicialización de acumuladores

    Ii=0;
```

```
    Ip=0;

    % Denotaremos por E a la suma de los valores de los extremos

    E=a+b;

    % Denotaremos por Ii a la suma de las ordenadas intermedias impares
    % (pares en Matlab) en el método

    for i=2:2:M-3
        Ii=Ii+A(i,2);
    end

    % Denotaremos por Ii a la suma de las ordenadas intermedias pares
    % (impares en Matlab) en el método

    for i=3:2:M-4
        Ip=Ip+A(i,2);
    end

    % Calculando el valor de la aproximación

    int2=(h/3)*(E+4*Ii+2*Ip);

    % Finalmente sumamos las dos partes y obtenemos la integral completa

    [int]=int1+int2;

else % Aplicamos únicamente la regla de 1/3

    % Inicialización de acumuladores

    Ii=0;
    Ip=0;

    % Regla de Simpson 1/3

    % Denotaremos por E a la suma de los valores de los extremos

    E=a+b;

    % Denotaremos por Ii a la suma de las ordenadas intermedias impares
    % (pares en Matlab) en el método

    for i=2:2:M
        Ii=Ii+A(i,2);
    end

    % Denotaremos por Ii a la suma de las ordenadas intermedias pares
    % (impares en Matlab) en el método

    for i=3:2:M-1
        Ip=Ip+A(i,2);
    end

    % Calculando el valor de la aproximación
```

```
    [int]=(h/3)*(E+4*Ii+2*Ip);

end

end
```

FINAL DEL PROGRAMA

Se ha incluido en el código del programa un ejemplos creando un fichero Excel del que extraer los datos que puede ser utilizado para testar su funcionamiento.

Veamos ahora el ejemplo de este programa, en esta ocasión los datos son exportados de un fichero Excel. Como hemos explicado anteriormente, debemos introducir el nombre del fichero y el rango de valores.

$$fichero = DatosaexportarSIMPSON\ ,$$

$$rango =' A2{:}B12',$$

$$[int] = Simpson\_P4\_exph(fichero, rango)\ .$$

Siendo el resultado:

$$[int] = 5.6583e + 03\ .$$

En el siguiente capítulo vamos a estudiar el método de Newton Cotes cerrado general y el método de Newton Cotes cerrado basado en 6 intervalos en particular Siendo éste un método en el que se toman más puntos en cada intervalo será éste un método más preciso que los estudiados anteriormente.

2.3. REGLA DE NEWTON COTES CERRADA GENERAL

La regla de Newton-Cotes cerrada general es un método valioso de integración numérica que se emplea para estimar el valor de una integral definida en un intervalo cerrado [a,b]. Su precisión depende del grado del polinomio utilizado para aproximar la función a integrar y será más precisa cuanto más disminuya el tamaño de los intervalos de integración.

En este proyecto vamos a ver la Regla de Newton Cotes cerrada basada en 6 intervalos, ya sabemos que, a más subintervalos, mayor será la precisión de la aproximación. Entendiendo este método podemos aplicar cualquiera de ellos ampliando o disminuyendo el número de subintervalos en función de la precisión requerida.

2.3.1. Regla de Newton Cotes cerrada basada en 6 intervalos

En este método, utilizaremos 7 puntos para definir el polinomio (ver *Figura 16*) que aproximará la función, como en los casos anteriores, integraremos el polinomio formado por estos puntos en lugar de la función, con su consiguiente error, que será mucho menor que en los métodos analizados anteriormente debido a que estamos tomando muchos más puntos en cada subintervalo del dominio de la función.

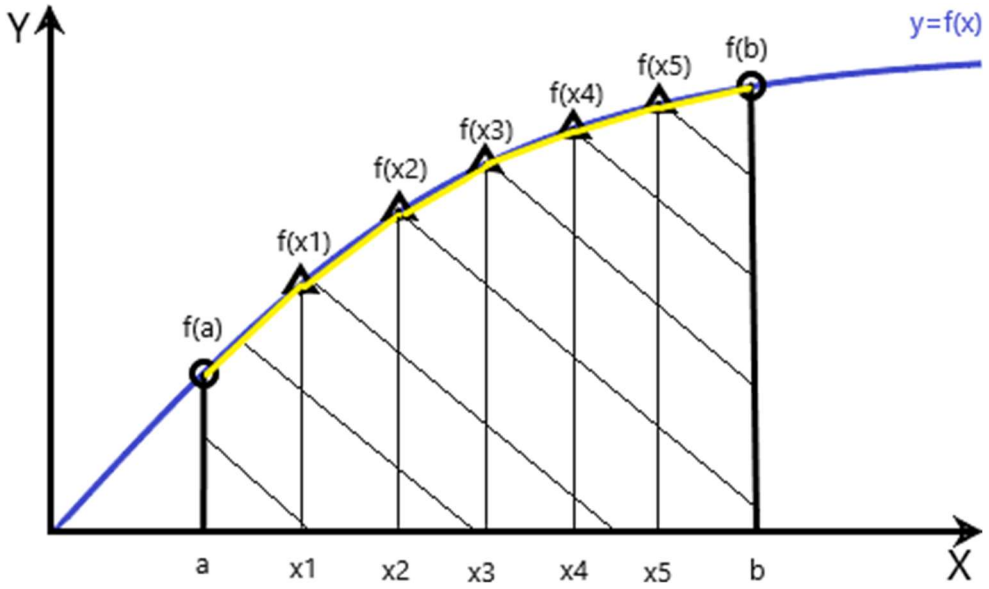


Figura 15: Representación método Newton Cotes cerrado basado en 6 intervalos.

Al igual que en los métodos anteriores, utilizaremos el Polinomio interpolador de Lagrange para obtener la siguiente expresión:

$$\int_a^b f(x)dx \approx \frac{2}{45}h\big(41f(x_0) + 216f(x_1) + 27f(x_2) + 272f(x_3) + 27f(x_4) + 216f(x_5) + 41f(x_6)\big).$$

Esta fórmula se aplica para el método de Newton Cotes de 6 intervalos simple, pero al igual que hemos estudiado en métodos anteriores, si aplicáramos varias veces el mismo método, pasaríamos a estar aplicando el método de Newton Cotes de 6 intervalos compuesto, el cual será más exacto cuanto más número de veces apliquemos la regla y cuya fórmula sería, una vez ya simplificada:

$$I = \frac{h}{140}(41E + 216M_1 + 27M_2 + 272C + 82J).$$

2.3.2. Error de la Regla de Newton Cotes cerrada basada en 6 intervalos

La regla de Newton-Cotes es una técnica de integración numérica que se utiliza para aproximar el valor de una integral definida. La regla de Newton-Cotes cerrada basada en 6 intervalos es una variante de esta técnica que utiliza una fórmula que incluye seis puntos equidistantes en el intervalo de integración.

El error de la Regla de Newton Cotes cerrada basada en 6 intervalos simple es:

$$E_h(f) = -\frac{9}{1400} h^9 f^{VIII}(\mu)\,,$$

con $\mu \in (a, b)$.

Este error tiene un grado de exactitud de 7 y un orden de aproximación local de 9.

El error de la Regla de Newton Cotes cerrada basada en 6 intervalos compuesta es:

$$E = -\frac{(b-a)}{840} h^8 f^{VIII}(\mu)\,,$$

con $\mu \in (a, b)$.

Este error tiene un grado de exactitud de 7 y un orden de aproximación global de 8.

El error de la regla de Newton-Cotes cerrada basada en 6 intervalos depende de varios factores, como la función que se está integrando, el intervalo de integración y la cantidad de puntos de integración. En términos generales, el error tiende a disminuir a medida que se incrementa el número de puntos de integración, pero a partir de un cierto número de puntos, el aumento adicional de puntos puede resultar en un aumento del error debido a la sensibilidad a errores de redondeo. Para evitar este efecto habría que trabajar con mayor precisión de cálculo, por ejemplo, en Matlab estableciendo el número de dígitos significativos y trabajando en aritmética de precisión variable.

2.3.3. Método de Newton Cotes cerrado con 6 intervalos programado en lenguaje Matlab

Ahora vamos a ver los 4 programas en lenguaje Matlab que hemos realizado, en los que introduciremos diferentes datos de entrada para llegar a la aproximación de la integral de una función dada. En ellos aplicaremos el método de Newton Cotes cerrado basado en 6 intervalos.

2.3.3.1. Newton Cotes cerrado con 6 intervalos dado el número de subintervalos

En este ejemplo hemos escrito en lenguaje Matlab un código para aproximar la integral mediante el método de Newton Cotes cerrado basado en 6 intervalos de una determinada función, que puede ser introducida como conjunto de datos o como expresión simbólica. Teniendo como datos de entrada la función, los extremos del intervalo [a,b] y el número de subintervalos a aplicar.

A continuación, se ofrece el código de programación desarrollado al efecto:

```
function [int]=Ncotes6_P1_Mab(f,M,a,b)

% Este programa tiene como objetivo calcular la integral mediante el método
% de Newton Cotes basada en 6 intervalos de una determinada función
% introducida como una expresión simbólica o un conjunto de datos para la
% cual definiremos el número de veces que vamos a aplicar Ncotes6 y los
% extremos del intervalo donde queremos aproximar la integral.
%
% [int]=Ncotes6_P1_Mab(f,M,a,b)
%
% Datos de entrada
% f: función introducida como expresión simbólica o como conjunto de datos.
% M: número de subintervalos.
% a,b: extremos del intervalo.
%
% Ejemplo 1:
% syms x
% [int]=Ncotes6_P1_Mab(tan(2-x),60,0,pi/2)
% Ejemplo 2:
% [int]=Ncotes6_P1_Mab([0    0.1003    0.2027    0.3093    0.4228    0.5463
0.6841    0.8423    1.0296    1.2602    1.5574    1.9648    2.5722    3.6021
5.7979   14.1014   -34.2325   -7.6966   -4.2863   -2.9271   -2.1850],7,0,100)

% Inicializamos las variables

syms x

% Calculamos la distancia entre nodos

h=(b-a)/M;

% Inicialización de acumuladores

J=0;
M1=0;
M2=0;
C=0;

% Cálculo de los nodos

xn(1)=a;
```

```
xn(M+1)=b;

for i=2:M
    xn(i)=xn(1)+(i-1)*h;
end

if ~isfloat(f) % Función introducida como simbólica

    % Aplicamos la regla Ncotes6

    % E será la suma de los extremos

    E=double(subs(f,a))+double(subs(f,xn(M+1)));

    % J será la suma de los extremos acumulados en el método

    for i=7:6:M-5
        J=J+double(subs(f,xn(i)));
    end

    % M1 será la suma de las ordenadas intermedias de índices 1 y 5

    for i=2:6:M-4
        M1=M1+double(subs(f,xn(i)));
        M1=M1+double(subs(f,xn(i+4)));
    end

    % M2 será la suma de las ordenadas intermedias de índices 2 y 4

    for i=3:6:M-3
        M2=M2+double(subs(f,xn(i)));
        M2=M2+double(subs(f,xn(i+2)));
    end

    % C será la suma de las ordenadas centrales del método

    for i=4:6:M-2
        C=C+double(subs(f,xn(i)));
    end

else

    % Función introducida como un conjunto de datos

    M=length(f)-1;

    fprintf('\n');
    fprintf('El valor de M ha sido actualizado a la longitud del vector de datos
introducido menos una unidad, M= %d',M);

    % Calculamos la distancia entre nodos

    h=(b-a)/M;

    % Aplicamos la regla Ncotes6

    % E será la suma de los extremos
```

```
    E=f(1)+f(M+1);

    % J será la suma de los extremos acumulados en el método

    for i=7:6:M-5
        J=J+f(i);
    end

    % M1 será la suma de las ordenadas intermedias de índices 1 y 5

    for i=2:6:M-4
        M1=M1+f(i);
        M1=M1+f(i+4);
    end

    % M2 será la suma de las ordenadas intermedias de índices 2 y 4

    for i=3:6:M-3
        M2=M2+f(i);
        M2=M2+f(i+2);
    end

    % C será la suma de las ordenadas centrales del método

    for i=4:6:M-2
        C=C+f(i);
    end

end

% Finalmente obtenemos el resultado de la integral

int=(h/140)*(41*E+216*M1+27*M2+272*C+82*J);

end
```

FINAL DEL PROGRAMA

Veamos ahora el ejemplo incluido en el código para testar el correcto funcionamiento del programa.

Introducimos la función como expresión simbólica:

$$syms\ x\ ,$$
$$[int] = Ncotes6\_P1\_Mab(tan(2-x),60,0,pi/2)\ .$$

El resultado que nos da el programa es:

$$[int] = 2.0352\ .$$

2.3.3.2. Newton Cotes cerrado con 6 intervalos dada la distancia entre nodos

En este ejemplo hemos escrito en lenguaje Matlab un código para aproximar la integral mediante el método de Newton Cotes cerrado basado en 6 intervalos de una determinada función, que puede ser introducida como conjunto de datos o como expresión simbólica. Teniendo como datos de entrada la función, los extremos del intervalo [a,b] y la distancia entre nodos.

A continuación, se ofrece el código de programación desarrollado al efecto:

```
function [int]=Ncotes6_P2_hab(f,h,a,b)
```


```
function [int]=Ncotes6_P2_hab(f,h,a,b)
```


```
% Este programa tiene como objetivo calcular la integral mediante el método
% de Newton Cotes basada en 6 intervalos de una determinada función
% introducida como una expresión simbólica o un conjunto de datos para la
% cual definiremos la longitud de los subintervalos y los extremos del
% intervalo donde queremos aproximar la integral.
%
% [int]=Ncotes6_P2_hab(f,h,a,b)
%
% Datos de entrada
% f: función introducida como expresión simbólica o como conjunto de datos.
% h: longitud del subintervalo (distancia entre nodos).
% a,b: extremos del intervalo.
%
% Ejemplo 1:
% syms x
% [int]=Ncotes6_P2_hab(cos(2*x)+3,1,0,21)
% Ejemplo 2:
% [int]=Ncotes6_P2_hab([4.0000    3.9861    3.9450    3.8776    3.7859    3.6724
3.5403    3.3932    3.2352    3.0707    2.9043    2.7405    2.5839],3,0,21)

% Inicializamos las variables

syms x

% Calculamos el número subintervalos que tenemos

M=(b-a)/h;

% Inicialización de acumuladores

J=0;
M1=0;
M2=0;
C=0;

if ~isfloat(f) % Función introducida como simbólica

    % Cálculo de los nodos

    xn(1)=a;
    xn(M+1)=b;

    for i=2:M
        xn(i)=xn(1)+(i-1)*h;
    end
```

```
    % Aplicamos la regla de Ncotes6

    % E será la suma de los extremos

    E=double(subs(f,xn(1)))+double(subs(f,xn(M+1)));

    % J será la suma de los extremos acumulados en el método

    for i=7:6:M-5
        J=J+double(subs(f,xn(i)));
    end

    % M1 será la suma de las ordenadas intermedias de índices 1 y 5

    for i=2:6:M-4
        M1=M1+double(subs(f,xn(i)));
        M1=M1+double(subs(f,xn(i+4)));
    end

    % M2 será la suma de las ordenadas intermedias de índices 2 y 4

    for i=3:6:M-3
        M2=M2+double(subs(f,xn(i)));
        M2=M2+double(subs(f,xn(i+2)));
    end

    % C será la suma de las ordenadas centrales del método

    for i=4:6:M-2
        C=C+double(subs(f,xn(i)));
    end

else

    % Función introducida como un conjunto de datos

    M=length(f)-1;

    fprintf('\n');
    fprintf('El valor de M ha sido actualizado a la longitud del vector de datos
introducido menos una unidad, M= %d',M);

    % Calculamos la longitud entre intervalos

    h=(b-a)/M;

    % Aplicamos la regla Ncotes6

    % E será la suma de los extremos

    E=f(1)+f(M+1);

    % J será la suma de los extremos acumulados en el método
    for i=7:6:M-5
        J=J+f(i);
    end
```

```
    % M1 será la suma de las ordenadas intermedias de índices 1 y 5

    for i=2:6:M-4
        M1=M1+f(i);
        M1=M1+f(i+4);
    end

    % M2 será la suma de las ordenadas intermedias de índices 2 y 4

    for i=3:6:M-3
        M2=M2+f(i);
        M2=M2+f(i+2);
    end

    % C será la suma de las ordenadas centrales del método

    for i=4:6:M-2
        C=C+f(i);
    end

end

% Finalmente obtenemos el resultado de la integral

int=(h/140)*(41*E+216*M1+27*M2+272*C+82*J);

end
```

FINAL DEL PROGRAMA

Hemos realizado dos ejemplos que están incluidos dentro del propio código del programa.

Ejecutamos el ejemplo 1 para comprobar su correcto funcionamiento.
Introducimos la función como expresión simbólica:

$$syms\ x\ ,$$
$$[int] = Ncotes6\_P2\_hab(cos(2 * x) + 3,1,0,21)\ .$$

Dándonos como resultado:

$$[int] = 54.5020\ .$$

2.3.3.3. Newton Cotes cerrado con 6 intervalos dada la tolerancia máxima del error

En este tercer ejemplo hemos escrito en lenguaje Matlab un código para aproximar la integral de una determinada función mediante el método de Newton Cotes cerrado basado en 6 intervalos introducida como una expresión simbólica. Teniendo como datos de entrada la función $f(x)$, los extremos del intervalo [a,b] y la tolerancia máxima del error, es decir, el error de la aproximación debe ser menor que la tolerancia introducida.

A continuación, se ofrece el código de programación desarrollado al efecto:

```
function [int]=Ncotes6_P3_tolab(f,tol,a,b)

% Este programa tiene como objetivo calcular la integral de una función
% introducida como una cadena de caracteres o un conjunto de datos (vector
% 1x1) mediante el método de Newton Cotes de 6 intervalos, en este caso
% introduciremos una tolerancia que tenemos que cumplir, es decir, que el
% error de la integral debe ser menor que esta tolerancia y también
% introduciremos los extremos del intervalo entre los que queremos que se
% evalue esta función.
%
% [int]=Ncotes6_P3_tolab(f,tol,a,b)
%
% Datos de entrada:
% f: función introducida como una expresión simbólica.
% tol: tolerancia con la que queremos trabajar. Tolerancia máx. del error.
% (a,b): extremos del intervalo.
%
% Datos de salida:
% [int]: integral de la función para los datos de entrada (tol, a y b).
%
% Ejemplo 1:
% syms x;
% [int]=Ncotes6_P3_tolab(cos(x+2)*(4*x),10^(-3),0,50)

% Primero realizamos el cálculo aproximado del máximo de la segunda derivada
% de la función  en un intervalo [a,b] evaluando en un mallado fino

syms x;

nx=1000;
hx=(b-a)/nx;

% Para hallar el máximo necesitamos conocer la segunda derivada de f

xe=zeros(nx);    % Predefinimos los arrays para reducir cálculos
vd2f=zeros(nx);

% Calculamos la segunda derivada y asignamos el máximo al primer punto
% del mallado como valor inicial para el bucle que sigue

d1=diff(f);
d2=diff(d1);
xe(1)=a;
vd2f(1)=double(subs(d2,x,xe(1)));
maxd2=abs(vd2f(1));


% Ahora calculamos el máximo mediante un mallado fino, comparando cada
```

```
% valor con el punto siguiente y quedándonos con el mayor de ellos

for i=2:nx
    xe(i)=xe(i-1)+hx;
    vd2f(i)=double(subs(d2,x,xe(i)));
    if abs(vd2f(i))> maxd2
        maxd2=abs(vd2f(i));
    end
end

% Tomamos un margen de error en la estimación del máx.

maxd2=maxd2+0.1;

% Utilizamos la fórmula del error para hallar M (número de veces que
% aplicamos NCotes6)

M=sqrt((b-a)^3/12/tol*maxd2);

% Redondeamos

M=ceil(M);

fprintf('\n');
fprintf('La tolerancia introducida como margen de error se cumple para M=
%d\n',M)

% Ahora aplicamos Ncotes6, utilizando el mismo método que en el P1, ya que
% ahora tenemos los datos de partida de ese caso (M, a y b).

% Calculamos la longitud de los intervalos

h=(b-a)/(M);

% Inicialización de acumuladores

J=0;
M1=0;
M2=0;
C=0;

% Cálculo de los nodos

xn(1)=a;
xn(M+1)=b;

for i=2:M
    xn(i)=xn(1)+(i-1)*h;
end

% Aplicamos la regla Ncotes6

% E será la suma de los extremos

E=double(subs(f,a))+double(subs(f,b));

% J será la suma de los extremos acumulados en el método
```

```
for i=7:6:M-5
    J=J+double(subs(f,xn(i)));
end

% M1 será la suma de las ordenadas intermedias de índices 1 y 5

for i=2:6:M-4
    M1=M1+double(subs(f,xn(i)));
    M1=M1+double(subs(f,xn(i+4)));
end

% M2 será la suma de las ordenadas intermedias de índices 2 y 4

for i=3:6:M-3
    M2=M2+double(subs(f,xn(i)));
    M2=M2+double(subs(f,xn(i+2)));
end

% C será la suma de las ordenadas centrales del método

for i=4:6:M-2
    C=C+double(subs(f,xn(i)));
end

% Finalmente obtenemos el resultado de la integral

int=(h/140)*(41*E+216*M1+27*M2+272*C+82*J);

end
```

FINAL DEL PROGRAMA

Ejecutamos el ejemplo incluido en el código del programa para testar su correcto funcionamiento.

$$syms\ x,$$
$$[int] = Ncotes6\_P3\_tolab(cos(x+2)*(4*x),10\text{^}(-3),0,50)\ .$$

Devolviendo el programa los siguientes inputs:

$$La\ tolerancia\ introducida\ como\ margen\ de\ error\ se\ cumple\ para\ M = \ 44894\ ,$$
$$[int] = 198.4101\ .$$

2.3.3.4. Newton Cotes cerrado con 6 intervalos exportando datos de un fichero Excel

En este último ejemplo hemos escrito en lenguaje Matlab un código para aproximar la integral de una determinada función, que en esta ocasión será un conjunto de datos que exportaremos de un fichero Excel, el otro dato de entrada que debemos introducir es el rango de datos con los que queremos trabajar dentro del fichero.

A continuación, se ofrece el código de programación desarrollado al efecto:

```
function [int]=Ncotes6_P4_exph(fichero,rango)

% En este ejercicio vamos a calcular la integral de una función introducida
% como un conjunto de datos exportados de una tabla Excel que debe estar
% guardada en la misma carpeta que el programa de Matlab (importante: si
% el archivo Excel no está guardado en la misma carpeta que el programa,
% éste no funcionará), este conjunto de datos exportados se evaluará en el
% intervalo a,b.
%
% [int]=Ncotes6_P4_exph(fichero,rango)
%
% Datos de entrada:
% f: nombre del fichero que contiene el conjunto de datos en un Excel.
% a,b: extremos del intervalo.
%
% Datos de salida:
% [int]: resultado final de la integral.
%
% Ejemplo 1:
% fichero='DatosaexportarNCOTES6.xlsx';
% rango='A2:B12';
% [int]=Ncotes6_P4_exph(fichero,rango)

A=xlsread(fichero,rango);

[M,Mc]=size(A);

h=A(2,1)-A(1,1);

J=0;
M1=0;
M2=0;
C=0;

% Aplicamos la regla Ncotes6

a=A(1,2);
b=A(M,Mc);

% E será la suma de los extremos

E=a+b;

% J será la suma de los extremos acumulados en el método

for i=7:6:M-5
    J=J+A(i,Mc);
end
```

```
% M1 será la suma de las ordenadas intermedias de índices 1 y 5

for i=2:6:M-4
    M1=M1+A(i,Mc);
    M1=M1+A(i+4,Mc);
end

% M2 será la suma de las ordenadas intermedias de índices 2 y 4

for i=3:6:M-3
    M2=M2+A(i,Mc);
    M2=M2+A(i+2,Mc);
end

% C será la suma de las ordenadas centrales del método

for i=4:6:M-2
    C=C+A(i,Mc);
end

% Finalmente obtenemos el resultado de la integral

[int]=(h/140)*(41*E+216*M1+27*M2+272*C+82*J);
```

FINAL DEL PROGRAMA

Ahora vamos a ejecutar el ejemplo incluido dentro del código del programa, como hemos hecho anteriormente para comprobar el correcto funcionamiento de este.

$$fichero = '\ DatosaexportarNCOTES6.xlsx',$$

$$rango = '\ A2:B12',$$

$$[int] = Ncotes6\_P4\_exph(fichero, rango)\ .$$

Y el programa nos devuelve el resultado:

$$[int] = 2.9396e + 03\ .$$

3. FORMULAS INTERPOLATORIAS

Las fórmulas interpolatorias son herramientas matemáticas utilizadas para identificar una función que atraviese un conjunto de puntos proporcionados. La meta es hallar una función que se ajuste de la manera más precisa posible a los puntos dados, de manera que se pueda utilizar para estimar valores para cualquier punto dentro del rango de los datos.

Un ejemplo de fórmula interpolatoria muy común es la interpolación por splines, que se basa en encontrar varios polinomios que se unen de manera suave en los puntos de interpolación. Esto permite una mayor flexibilidad en la función interpolante y un mejor ajuste a los datos. Otra manera muy usual, y quizás la más sencilla de obtener una fórmula interpolatoria, es la conocida interpolación de Lagrange.

Las fórmulas interpolatorias son ampliamente utilizadas en campos como la estadística, la física y la ingeniería, y son un instrumento esencial en el análisis de datos y en la modelización matemática.

3.1. REGLAS DE CUADRATURA

Cómo hemos explicado anteriormente, la aproximación numérica de una integral definida tiene sentido cuando la expresión analítica de la función es desconocida que queremos integrar. Esta expresión puede aproximarse mediante interpolaciones usando valores puntuales de dicha función obtenidos mediante mediciones. Siendo esta valor la aproximación del área comprendida entre la *f(x)* y el intervalo [a,b] (véase *Figura 16*).

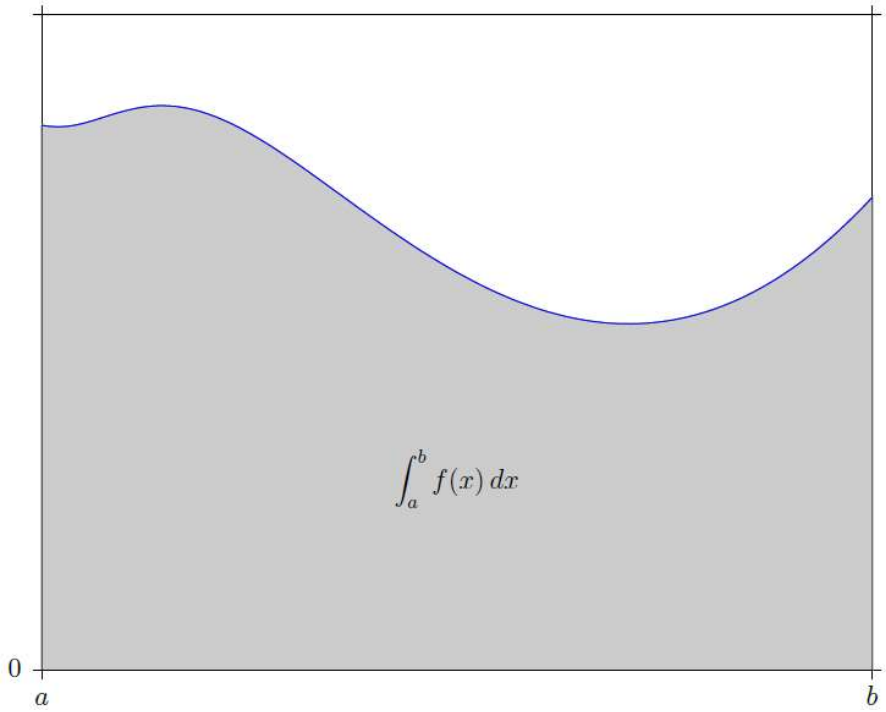


Figura 16: Área comprendida entre *f(x)* y el eje Y = 0.

Mediante el método de la Cuadratura, podremos calcular la aproximación de integrales definidas

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

donde *f* es la función integrable en el intervalo [a,b]. Si la *f(x)* es una función positiva estaríamos hallando el área comprendida entre la función y el eje x en el intervalo [a,b], tal y como se muestra en la Figura 10.
Por Reglas de Cuadratura denominamos a las fórmulas que se derivan de utilizar como función aproximante el polinomio interpolador de Lagrange basado en ciertos puntos cualesquiera del intervalo [a,b]. Las fórmulas de Newton Cotes de los capítulos anteriores son casos particulares de estas Reglas de Cuadratura, cuando los nodos están igualmente espaciados.

3.1.1. Coeficientes de la Regla de Cuadratura

Para la resolución del problema planteado, dado un conjunto de nodos *ci* conocidos, se han de encontrar de una forma razonablemente sencilla los coeficientes α_i que aproximen

$$\int_a^b f(x)dx \cong \sum_{i=1}^{n} \alpha_i f\big(a + c_i(b-a)\big).$$

Para simplificar los cálculos significativamente, en lugar de trabajar con un intervalo genérico [a,b], buscaremos un intervalo más sencillo [0,1]. Para ello realizamos el cambio de variable.

$$x = a + t(b-a),$$

fórmula mediante la cual obtenemos las siguientes equivalencias,

$$x = a \leftrightarrow t = 0,$$
$$x = b \leftrightarrow t = 1,$$

transformando así la variable $x \in [a,b]$ en $t \in [0,1]$. Podemos realizar un cambio de nombre en la función.

$$g(t) = f(x) = f(a + t(b-a).$$

Buscando ahora coeficientes β_i adaptados a la nueva función:

$$\int_0^1 g(t)dt \cong \sum_{i=1}^{n} \beta_i g(c_i).$$

Vamos a construir la regla buscando exactitud para polinomios, cuya integral es fácil de calcular a partir de las integrales de los elementos de la base canónica de polinomios $\{1, t, \dots, t^j, \dots\}$, que son:

$$\int_0^1 t^j dt = \frac{t^{j+1}}{j+1}\bigg|_0^1 = \frac{1}{j+1},$$

quedando así la ecuación

$$\sum_{i=1}^{n} \beta_i c_i^j = \frac{1}{j+1}\ .$$

Al tener n grados de libertad, impondremos las n primeras ecuaciones de este tipo. El sistema lineal resultante expresado de forma matricial queda de la siguiente forma:

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ c_1 & c_2 & c_3 & \cdots & c_n \\ c_1^2 & c_2^2 & c_3^2 & \cdots & c_n^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_1^{n-1} & c_2^{n-1} & c_3^{n-1} & \cdots & c_n^{n-1} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \\ \beta_n \end{pmatrix} = \begin{pmatrix} 1 \\ 1/2 \\ 1/3 \\ \vdots \\ 1/n \end{pmatrix}.$$

La matriz de ese sistema se llama matriz de Vandermonde.

Para pasar de la ecuación que trabaja en el intervalo [0,1] a la ecuación que lo hace en el intervalo [a,b] también hay que tener en cuenta el diferencial del cambio de variable, que es

$$dx = (b-a)dt,$$

siendo los coeficientes para el caso general:

$$\alpha_i = (b-a)\beta_i\,.$$

Finalmente, la Regla de Cuadratura queda definida como:

$$\int_a^b f(x)dx \cong (b-a)\sum_{i=0}^{n} \beta_i f(a + c_i(b-a).$$

3.1.2. Error de la Regla de Cuadratura

La regla de cuadratura es una técnica de integración numérica utilizada para aproximar el valor de una integral definida. Esta técnica se apoya en la aproximación de la función a integrar mediante otra función polinómica de bajo grado, como una línea recta o una parábola. El error de la regla de cuadratura es la diferencia entre el valor real de la integral y el valor aproximado obtenido mediante esta técnica de integración.

Suponiendo que se ha utilizado una fórmula interpolatoria para aproximar la integral $I(f) = \int_a^b f(x)\,dx$, el error cometido será:

$$E_{n+1}(f) = I(f) - I(P_n) = \int_a^b \big(f(x) - P_n(x)\big)\,dx\,,$$

luego

$$|E_{n+1}(f)| \;\leq\; (b-a) \sup_{x\,\in\,[a,\,b]} |f(x) - P_n(x)|\,,$$

de modo que si P_n es una buena aproximación a f, entonces $I(P_n)$ es una buena aproximación a $I(f)$.
Por otra parte, usando el error de interpretación $f(x) - P_n(x)$ en la forma de Newton se tiene:

$$E_{n+1}(f) = \int_a^b f[x_0, \dots, x_n, x]\prod(x)\,dx = \int_a^b \frac{f^{(n+1)}(\varepsilon(x))}{(n+1)!}\prod(x)\,dx\;,$$

donde $\prod(x) = (x - x_0) \dots (x - x_n)$. Entonces si $\prod(x)$ no cambiade signo en [a,b] y $f^{(n+1)}$ es continua, es posible aplicar el teorema del valor medio de integración para obtener:

$$E_{n+1}(f) = \frac{f^{(n+1)}(\alpha)}{(n+1)!} \int_a^b \prod(x)\, dx \ ,$$

con $\min\{x_j, a\} < \alpha < \max\{x_j, b\}$, y siempre se cumple la siguiente desigualdad en valor absoluto,

$$|E_{n+1}(f)| \leq (\max\{x\ en\ [a,b]\} \left| \frac{f^{(n+1)}(\alpha)}{(n+1)!} \int_a^b \prod(x)\, dx \right|) .$$

3.1.3. Método de la Cuadratura programado en lenguaje Matlab

Vamos a ver los 4 programas en lenguaje Matlab que hemos realizado, en los que introduciremos diferentes datos de entrada para llegar a la aproximación de la integral de una función dada. En ellos aplicaremos el método de la Cuadratura.

3.1.3.1. Cuadratura dados el vector de valores de la función en los nodos y el vector de porcentajes

En este ejemplo hemos escrito en lenguaje Matlab un código para aproximar la integral mediante el método de la Cuadratura en el cual hemos introducido como datos de entrada el vector de valores de la función en los nodos x_i y el vector de porcentajes c_i.

A continuación, se ofrece el código de programación desarrollado al efecto:

```
function [int]=Cuadratura_P1_dc(fi,a,b,c)

% En este programa vamos a implementar una regla de cuadratura
% de tipo interpolatorio para una función definida en un intervalo [a,b]
% de la cual tenemos sus valores puntuales en un conjunto de nodos x_i
% pertenecientes a dicho intervalo.
% En este programa los nodos x_i se calculan a partir del vector de
% porcentajes c. Y a partir de los c_i también se calculan los coeficientes
% de peso alpha_i que nos definen la regla de cuadratura para aproximar la
% integral deseada.
%
% [int]=Cuadratura_P1_dc(fi,a,b,c)
%
% Datos de entrada:
% fi: vector de valores puntuales de la función elegida en los nodos x_i.
% a,b: extremos del intervalo
% c: vector de porcentajes cumpliendo x_i=a+c_i(b-a)
%
% Datos de salida:
% int: aproximación a la integral de la función calculada para un conjunto
% de valores en los nodos dado.
%
% Ejemplo:
% a=5 ; b=7;
% c=[0 1/2 1];
% fi=[2.2837 2.9602 2.7539];
% [int]=Cuadratura_P1_dc(fi,a,b,c)

% Primero nos guardamos la longitud del vector de porcentajes

nc=length(c);

% Ahora tenemos que hallar los coeficientes x_i a través del vector c_i

xi=zeros(length(c));    % Predefinimos el array para reducir cálculos.

for i=1:nc
    xi(i)=(c(i)*(b-a))+a;
end

% Ahora calculamos la matriz de Vandermonde
```

```
v=vander(c);

% Tenemos que transponer la matriz y utilizamos la función "flipud" para
% darle la vuelta y así disponer los datos en el orden que necesitamos

v=v';

v=flipud(v);

% Hay que resolver un sistema lineal con matriz de coeficientes la matriz
% de Vandermonde v y con vector de términos independientes bx.

bx=zeros(length(xi));   % Predefinimos el array para reducir cálculos

for i=1:nc
    bx(i)=1/i;
end

bx=bx';

% Orden para resolver el sistema lineal.

beta=bx/v;

% Así obtendríamos los coeficientes beta que buscábamos en el intervalo
% [0,1].

% Una vez obtenidos los coeficientes beta, podemos hallar los coeficientes
% alpha buscados, con la fórmula siguiente

alpha=zeros(length(xi));

for i=1:nc
    alpha(i)=beta(i)*(b-a);
end

% Una vez obtenidos los coeficientes alpha, simplemente tenemos que
% aplicar la fórmula de cuadratura

int=0;

for i=1:nc
    int=int+alpha(i)*fi(i);
end

end
```

FINAL DEL PROGRAMA

Hemos incluido un ejemplo dentro del código del programa para testear su funcionamiento. En este caso, los datos de entrada serán introducidos como un vector de valores puntuales de una cierta función en los nodos correspondientes, más los porcentajes que indican los nodos a considerar.

Los datos de entrada son:

$$a = 5\ ;\ b = 7,$$

$$c = [0 \quad 1/2 \quad 1]\ ,$$

$$fi = [2.2837 \quad 2.9602 \quad 2.7539]\ ,$$

$$[int] = Cuadratura\_P1\_dc(fi, a, b, c)\ ,$$

o lo que es lo mismo:

$$[int] = Cuadratura\_P1\_dc([2.2837\ 2.9602\ 2.7539],5,7,[0\ 1/2\ 1])\ .$$

Siendo el resultado dado por el programa:

$$[int] = 4.5674\ .$$

3.1.3.2. Cuadratura dados el vector de valores de la función en los nodos y el vector de nodos

En este programa hemos escrito en lenguaje Matlab un código para aproximar la integral mediante el método de la Cuadratura. Hemos introducido como datos de entrada el vector de valores de la función en los nodos x_i y el vector de nodos pertenecientes al intervalo [a,b].

A continuación, se ofrece el código de programación desarrollado al efecto:

```
function [int]=Cuadratura_P2_dx(fi,a,b,xi)

% En este programa vamos a implementar una regla de cuadratura
% de tipo interpolatorio para una función definida en un intervalo [a,b]
% de la cual tenemos sus valores puntuales en un conjunto de nodos x_i
% pertenecientes a dicho intervalo.
% A partir de estos x_i se calculan los porcentajes c_i. Y a partir de los
% c_i se calculan los coeficientes de peso alpha_i que nos definen la regla
% de cuadratura para aproximar la integral deseada.
%
% [int]=Cuadratura_P2_dx(fi,a,b,xi)
%
% Datos de entrada:
% fi: vector de valores puntuales de la función elegida en los nodos x_i.
% a,b: extremos del intervalo
% xi: vector de nodos pertenecientes al intervalo [a,b], cumpliendo
% x_i=a+c_i(b-a) .
%
% Datos de salida:
% int: aproximación a la integral de la función calculada para un conjunto
% de valores en los nodos dado.
%
% Ejemplo:
% a=5 ; b=7;
% xi=[5 6 7];
% fi=[2.2837 2.9602 2.7539];
% [int]=Cuadratura_P2_dx(fi,a,b,xi)
% [int]=Cuadratura_P2_dx([2.2837 2.9602 2.7539],5,7,[5 6 7])

% Primero nos guardamos la longitud del vector de nodos

nx=length(xi);

% Ahora tenemos que  hallar los coeficientes c_i a través del vector x_i

c=zeros(1,length(xi));    % Predefinimos el array para reducir calculos.

for i=1:nx
    c(i)=(xi(i)-a)/(b-a);
end

% Ahora calculamos la matriz de Vandermonde

v=vander(c);

% Tenemos que transponer la matriz y utilizamos la función "flipup" para
% darle la vuelta y así disponer los datos en el orden que necesitamos

v=v';
```

```
v=flipud(v);

% Hay que resolver un sistema lineal con matriz de coeficientes la matriz
% de Vandermonde v y con vector de términos independientes bx.

% Predefinimos el array para reducir cálculos

bx=zeros(length(xi),1);

nc=nx;

for i=1:nc
    bx(i)=1/i;
end


% Orden para resolver el sistema lineal.

beta=v\bx;

% Así obtendríamos los coeficientes beta que buscabamos en el intervalo
% [0,1].

% Una vez obtenidos los coeficientes beta, podemos hallar los coeficientes
% alpha buscados, con la formula siguiente

alpha=zeros(length(xi));

for i=1:nx
    alpha(i)=beta(i)*(b-a);
end

% Una vez obtenidos los coeficientes alpha, simplemente tenemos que
% aplicar la fórmula de cuadratura

int=0;

for i=1:nx
    int=int+alpha(i)*fi(i);
end

end
```

FINAL DEL PROGRAMA

Hemos incluido otro ejemplos dentro del código del programa para comprobar su funcionamiento. Al igual que en el anterior programa, los datos de la función serán introducidos como un vector de valores f_i.

Los datos de entrada son:

$$a = 5\ ;\ b = 7\ ,$$

$$xi = [5 \quad 6 \quad 7]\ ,$$

$$fi = [2.2837 \quad 2.9602 \quad 2.7539]\ ,$$

$$[int] = Cuadratura\_P2\_dx(fi, a, b, xi)\ .$$

o lo que es lo mismo:

$$[int] = Cuadratura\_P2\_dx([2.2837\ 2.9602\ 2.7539],5,7,[5\ 6\ 7])$$

Devolviendo el programa el resultado:

$$[int] = 5.6261\ .$$

3.1.3.3. Cuadratura dadas la función y el vector de porcentajes

En este nuevo programa hemos escrito en lenguaje Matlab un código para aproximar la integral mediante el método de la Cuadratura usando como datos de entrada la función *f(x)* de manera simbólica y el vector de porcentajes c_i.

A continuación, se ofrece el código de programación desarrollado al efecto:

```
function [int]=Cuadratura_P3_fc(f,a,b,c)

% En este programa vamos a implementar una regla de cuadratura
% de tipo interpolatorio para una función definida en un intervalo [a,b].
% Cada regla viene definida mediante un conjunto de nodos xi que se calculan
% a través de unos porcentajes conocidos c_i. A partir de estos porcentajes
% se calculan los coeficientes de peso alpha_i que nos definen la regla de
% cuadratura para aproximar la integral deseada.
%
% [int]=Cuadratura_P3_fc(f,a,b,c)
%
% Datos de entrada:
% f: función introducida como expresión simbólica.
% a,b: extremos del intervalo
% c: vector de proporciones en que se parte el intervalo [a,b], cumpliendo
% x_i=a+c_i(b-a).
%
% Datos de salida:
% int: aproximación a la integral de la función calculada para un conjunto de
nodos dado.
%
% Ejemplo:
% syms x
% a=0 ; b=1;
% f=sin(x)*2;
% c=[1/4 3/4];
% [int]=Cuadratura_P3_fc(f,a,b,c)
% [int]=Cuadratura_P3_fc(sin(x)*2,0,1,[1/4 3/4])

% Definimos la variable x como simbólica

syms x

% Calculamos la matriz de Vandermonde

v=vander(c);

% Tenemos que transponer la matriz y utilizamos la función "flipud" para
% darle la vuelta y así disponer los datos en el orden que necesitamos

v=v';

v=flipud(v);

% Hay que resolver un sistema lineal con matriz de coeficientes la matriz
% de Vandermonde v y con vector de terminos independientes bx

nc=length(c);

% Ahora tenemos que  hallar los coeficientes x_i a través del vector c_i
```

```
xi=zeros(length(c));    % Predefinimos el array para reducir calculos.

for i=1:nc
    xi(i)=(c(i)*(b-a))+a;
end

% Hay que resolver un sistema lineal con matriz de coeficientes la matriz
% de Vandermonde v y con vector de términos independientes bx.

bx=zeros(length(xi));   % Predefinimos el array para reducir calculos

for i=1:nc
    bx(i)=1/i;
end

bx=bx';

% orden para resolver el sistema lineal

beta=v\bx;

% Así obtendríamos los coeficientes beta que buscabamos en el intervalo [0,1].

% Una vez obtenidos los coeficientes beta, podemos hallar los coeficientes
% alpha buscados, con la formula siguiente.

alpha=zeros(length(xi));

for i=1:nc
    alpha(i)=beta(i)*(b-a);
end

% Una vez obtenidos los coeficientes alpha, simplemente tenemos que
% aplicar la fórmula de cuadratura

int=0;

for i=1:nc
    int=int+alpha(i)*double(subs(f,x,xi(i)));
end

end
```

FINAL DEL PROGRAMA

En este programa, a diferencia de los anteriores en los que la función se ha introducido como vector de valores, vamos a introducir la función como una expresión simbólica.

Veamos los datos de entrada:

$$syms\ x\ ,$$

$$a = 0\ ;\ b = 1\ ,$$

$$f = sin(x) * 2\ ,$$

$$c = [1/4 \quad 3/4]\ ,$$

$$[int] = Cuadratura\_P3\_fc(f, a, b, c),$$

O lo que es lo mismo:

$$[int] = Cuadratura\_P3\_fc(sin(x) * 2{,}0{,}1, [1/4\ 3/4]).$$

Siendo el resultado devuelto por el programa:

$$[int] = 0.0606\ .$$

3.1.3.4. Cuadratura dadas la función y el vector de nodos

En este programa hemos escrito en lenguaje Matlab un código para aproximar la integral mediante el método de la Cuadratura en el cual hemos introducido como datos de entrada la función *f(x)* y el vector de nodos pertenecientes al intervalo [a,b].

A continuación, se ofrece el código de programación desarrollado al efecto:

```
function [int]=Cuadratura_P4_fx(f,a,b,xi)

% En este programa vamos a implementar una regla de cuadratura
% de tipo interpolatorio para una función definida en un intervalo [a,b].
% Cada regla viene definida mediante un conjunto de nodos x_i para del
% intervalo. A partir de estos x_i se calculan los porcentajes c_i.
% Y a partir de los c_i se calculan los coeficientes de peso alpha_i que
% nos definen la regla de cuadratura para aproximar la integral deseada.
%
% [int]=Cuadratura_P4_fx(f,a,b,xi)
%
% Datos de entrada:
% f: función introducida como expresión simbólica.
% a,b: extremos del intervalo
% xi: vector de valores de los puntos en que se divide el intervalo [a,b],
% cumpliendo x_i=a+c_i(b-a) .
%
% Datos de salida:
% int: aproximación a la integral de la función calculada para un conjunto
% de valores en los nodos dado.
%
% Ejemplo:
% syms x
% a=5 ; b=7;
% f=cos(x)*2;
% xi=[5 6 7];
% [int]=Cuadratura_P4_fx(f,a,b,xi)

% Definimos la variable x como simbólica
syms x

% En primer lugar hallamos los coeficientes c_i a través del vector x_i

nx=length(xi);
c=zeros(1,length(xi));    % Predefinimos el array para reducir cálculos.

for i=1:nx
    c(i)=(xi(i)-a)/(b-a);
end

% Ahora calculamos la matriz de Vandermonde

v=vander(c);

% Tenemos que transponer la matriz y utilizamos la función "flipup" para
% darle la vuelta y así disponer los datos en el orden que necesitamos

v=v';

v=flipud(v);
```

```
% Hay que resolver un sistema lineal con matriz de coeficientes la matriz
% de Vandermonde v y con vector de términos independientes bx

nc=nx;  % número de porcentajes ci
bx=zeros(length(xi),1);

for i=1:nc
    bx(i)=1/i;
end

% Orden para resolver el sistema lineal

beta=v\bx;

% Así obtendríamos los coeficientes beta que buscábamos en el intervalo [0,1].

% Una vez obtenidos los coeficientes beta, podemos hallar los coeficientes
% alpha buscados, con la formula siguiente

alpha=zeros(length(xi));

for i=1:nc
    alpha(i)=beta(i)*(b-a);
end

% Una vez obtenidos los coeficientes alpha, simplemente tenemos que
% aplicar la fórmula de cuadratura

int=0;

for i=1:nc
    int=int+alpha(i)*double(subs(f,x,xi(i)));
end

end
```

FINAL DEL PROGRAMA

En este programa, también se va a introducir la función como una expresión simbólica.

Veamos los datos de entrada:

$$syms\ x\ ,$$

$$a = 5\ ;\ b = 7\ ,$$

$$f = cos(x) * 2\ ,$$

$$xi = [5\ \ 6\ \ 7]\ ,$$

$$[int] = Cuadratura\_P4\_fx(f, a, b, xi),$$

O lo que es lo mismo:

$$[int] = Cuadratura\_P4\_fx(cos(x) * 2, 5, 7, [5\ \ 6\ \ 7]).$$

Siendo el resultado devuelto por el programa:

$$[int] = 3.2522\ .$$

4. REGLAS DE INTEGRACIÓN NUMÉRICA PARA VARIAS VARIABLES

La integración numérica para funciones de varias variables es un proceso utilizado para aproximar el valor de una integral definida en una región determinada en el espacio n-dimensional. Las reglas de integración numérica para varias variables son técnicas matemáticas que permiten aproximar el valor de una integral utilizando solamente valores de la función en puntos discretos de la región de integración.

Una regla común para la integración numérica en varias variables es la Regla del Trapecio otra regla común para la integración numérica en varias variables es la Regla de Simpson. Para calcular integrales de funciones de varias variables la manera más sencilla es mediante el cálculo reiterado de integrales en una variable.

Es importante tener en cuenta que, al igual que en la integración numérica de una sola variable, la precisión de estas reglas de integración numérica para varias variables dependerá de la cantidad de puntos utilizados para aproximar la integral y de la forma de la región de integración. En general, estas técnicas son útiles para aproximar integrales en casos en los que no es posible encontrar una solución exacta de manera analítica.

Vamos a introducir la teoría de la integración doble e integrales de superficie y acto seguido analizaremos y crearemos un programa en lenguaje Matlab para los dos casos mencionados en este epígrafe, la Regla de los Trapecios para varias variables y la Regla de Simpson para varias variables.

En este trabajo vamos a centrarnos en las integrales dobles, aunque la extensión a integrales de funciones con más variables se lleva a cabo de forma similar.

4.1. INTEGRALES DOBLES

Primero vamos a ver la teoría de la integración doble para así poder comprender los programas en lenguaje Matlab que se presentan más adelante.

Imaginemos que contamos con un Área A en $\mathbb{R}^2$ de la forma

$$A = [a,b]x[c,d] = \{(x,y): a \leq x \leq b, c \leq y \leq d\},$$

y sea $f: A \to \mathbb{R}$ una función que está acotada y es positiva definida en el área.

Considerando las particiones:

$$P_1 = \{a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b\},$$

y

$$P_2 = \{c = y_0 < y_1 < y_2 < \cdots < y_{n-1} < y_n = d\},$$

de los intervalos [a,b] y [c,d], respectivamente.

A raíz de éstas obtenemos lo que denominamos partición del área *A* de la siguiente forma

$$P_1 x P_2 = \{(x,y) : x \in P_1, y \in P_2\},$$

es decir, la partición P_1xP_2 está compuesta por los puntos de la forma (x_i, y_j) donde $i = 0, 1, \dots, n$ y $j = 0, 1, 2, \dots, m$.

Mediante esta partición se puede dividir el área A en las siguientes sub-áreas

$$\{[x_{i-1}, x_i]x[y_{j-1}, y_j]\ i = 1, \dots, n, j = 1, \dots, m\}.$$

Se llaman suma superior y suma inferior de la función asociada a la partición

$$S(P_1xP_2) = \sum_{\substack{i=1,\dots,n \\ j=1,\dots,m}} M_{ij}(x_i - x_{i-1})(y_j - y_{j-1}),$$

y

$$s(P_1xP_2) = \sum_{\substack{i=1,\dots,n \\ j=1,\dots,m}} m_{ij}(x_i - x_{i-1})(y_j - y_{j-1}),$$

donde

$$M_{ij} = \sup\{f(x,y): (x,y) \in [x_{i-1}, x_i]x[y_{j-1}, y_j]\},$$
$$m_{ij} = \inf\{f(x,y): (x,y) \in [x_{i-1}, x_i]x[y_{j-1}, y_j]\}.$$

Estas sumas representan dos cifras, la primera siendo mayor o igual y la segunda menor o igual, respectivamente, que el volumen que encierra la superficie. $z = f(x,y)$ y el plano $z = 0$ en A.

Si incrementamos progresivamente la cantidad de puntos en las particiones, notaremos que las sumas superiores e inferiores se acercan cada vez más. De hecho, el conjunto de las sumas superiores de f asociadas a las particiones de A tiene un límite inferior, y el conjunto de las sumas inferiores de f asociadas a las particiones de A tiene un límite superior.

Entonces, decimos que f es *integrable Riemann* en A cuando el valor mínimo del conjunto de las sumas superiores de f asociadas a las particiones de A es igual al valor máximo del conjunto de las sumas inferiores de f asociadas a las particiones de A. Llamamos a este valor común la integral doble de f en A, y lo denotamos como:

$$\iint_A f(x,y)dxdy.$$

Esta integral representa gráficamente el volumen que la superficie $z = f(x,y)$ y el plano $z = 0$ encierran en el rectángulo A (contando los signos adecuadamente) y puede calcularse por los siguientes límites:

$$\lim_{n\to+\infty} S(P_n) = \lim_{n\to+\infty} s(P_n),$$

donde (P_n) es una sucesión de particiones que cumplen que el área de todas las sub-áreas de la partición tienden a 0.

Otra condición equivalente a que f sea integrable en A es que exista un único número I que cumpla la condición $s(P) \leq I \leq S(P)$ para cualquier partición P. De hecho, dicho número será la integral anteriormente definida.

Proposición

Si una función *f* es continua dentro de un rectángulo, entonces es integrable en dicha área.

Teorema (Fubini)

Sea $f: R \to \mathbb{R}$ una función continua en el área $A = [a,b]x[c,d]$. Entonces:

$$\iint_A f(x,y)dxdy = \int_a^b (\int_c^d f(x,y)dy)dx = \int_c^d (\int_a^b f(x,y)dx)dy.$$

Nota:

Además, en funciones del tipo $f(x,y) = g(x)h(y)$, la integral sobre un área $A = [a,b]x[c,d]$ puede calcularse como sigue:

$$\iint_A f(x,y)dxdy = (\int_a^b g(x)dx)(\int_c^d h(y)dy).$$

4.1.1. Integrales dobles sobre recintos generales

Consideramos una región acotada *D* en $\mathbb{R}^2$, y un rectángulo *A* de $\mathbb{R}^2$ que contenga a dicha región.
Si $f : D \to \mathbb{R}$ es una función sobre *D*, podemos considerar la nueva función $F : A \to \mathbb{R}$ definida como:

$$F(x,y) = \begin{cases} f(x,y) & si\ (x,y) \in D \\ 0 & si\ (x,y) \notin D. \end{cases}$$

Llegados a este punto, diremos que *f* es integrable en la región acotada *D* si *F* es integrable en el rectángulo *R*, y definiremos en tal caso la integral doble de *f* en *D* como:

$$\iint_D f(x,y)dxdy = \iint_A F(x,y)dxdy.$$

Fubini general

Sea $D \subseteq \mathbb{R}^2, f\ integrable\ en\ D.$

Si $D = \{(x,y) \in \mathbb{R}^2 :\ a \le x \le b,\ l(x) \le y \le u(x)\}$, entonces:

$$\iint_D f(x,y)\ dxdy = \int_a^b \left(\int_{l(x)}^{u(x)} f(x,y)\ dy \right) dx.$$

Si $D = \{(x,y) \in \mathbb{R}^2 :\ c \le y \le d,\ q(y) \le x \le s(y)\}$, entonces:

$$\iint_D f(x,y)\ dxdy = \int_c^d \left(\int_{q(y)}^{s(y)} f(x,y)\ dx \right) dy.$$

Si $D = \{(x,y) \in \mathbb{R}^2 \colon\ a \le x \le b, l(x) \le y \le u(x)\}$, entonces:

$$\iint_D f(x,y)\,dxdy = \int_a^b \left(\int_{l(x)}^{u(x)} f(x,y)\,dy \right) dx = \int_a^b g(x)\,dx,$$

con

$$g(x) = \int_{l(x)}^{u(x)} f(x,y)\,dy.$$

La integral $\int_a^b g(x)\,dx$ se puede aproximar usando Trapecios o Simpson. Cada vez que haya que evaluar g(x) nos damos cuenta de que hay que aproximar otra integral numérica con la misma regla de integración numérica en nuestro caso Trapecios o Simpson.

4.1.2. Integrales dobles en coordenadas polares

Teorema (Cambio de variable)

Sea $T\colon U \to V$ una aplicación de clase C^1 entre dos recintos básicos *U, V* de $\mathbb{R}^2$ de modo que

1. *T* es biyectiva
2. $|J(T)| \neq 0$ en todo punto de *U*, entonces si $f\colon V \to \mathbb{R}$ es una función continua se tiene que:

$$\iint_D f(x,y)dxdy = \iint_U f\big(T(u,v)\big)|J(T)|\,dudv.$$

En particular, si $D \subset \mathbb{R}^2$ es un rectángulo polar $f\colon D \to \mathbb{R}$ es continua se verifica

$$\iint_D f(x,y)dxdy = \int_{r_1}^{r_2} \left(\int_{\theta_1}^{\theta_2} f(rcos\theta, rsen\theta) r d\theta \right) dr,$$

pudiéndose realizar la integración iterada en el orden inverso.

Más aún, si tomamos una región plana $D \subset \mathbb{R}^2$ que se expresa en coordenadas polares de la forma

$$D = \{(r,\theta)\colon a \le r \le b, h_1(r) \le \theta \le h_2(r)\},$$

siendo $h_1(r), h_2(r)\colon [a,b] \to \mathbb{R}$ funciones continuas y $f\colon D \to \mathbb{R}$ es continua sobre *D*, entonces se tiene que

$$\iint_D f(x,y)dxdy = \int_a^b \left(\int_{h_1(r)}^{h_2(r)} f(rcos\theta, rsen\theta) r d\theta \right) dr.$$

Por el contrario, si suponemos que,

$$D = \{(r,\theta): \alpha \leq \theta \leq \beta, g_1(\theta) \leq r \leq g_2(\theta)\},$$

nos quedaría la fórmula

$$\iint_D f(x,y)dxdy = \int_\alpha^\beta (\int_{g_1(\theta)}^{g_2(\theta)} f(cos\theta, rsen\theta)rd\theta)dr.$$

4.2. INTEGRALES DE SUPERFICIE

Las integrales de superficie son una herramienta matemática utilizada en el cálculo multivariable para medir la cantidad de flujo de un campo vectorial a través de una superficie. Estas integrales se utilizan comúnmente en campos como la física, la ingeniería y las ciencias de la computación, y son esenciales para entender la física de fluidos, la teoría electromagnética y la geometría diferencial.

Se puede calcular la integral de un campo vectorial sobre una superficie determinada en el espacio tridimensional. Para hacer esto, se divide la superficie en pequeñas áreas, y se calcula la contribución del campo vectorial a través de cada una de estas áreas. La suma de estas contribuciones proporciona una aproximación del flujo total del campo vectorial a través de la superficie. También se puede calcular la integral de superficie de un campo escalar, para determinar áreas superficiales, momentos de inercia y otras magnitudes sobre la superficie que son campos escalares.

Las integrales de superficie son importantes en una amplia variedad de áreas, incluyendo la física, la ingeniería, la geometría diferencial y las ciencias de la computación. Por ejemplo, en la física, las integrales de superficie se utilizan para calcular la cantidad de energía que fluye a través de una superficie, mientras que, en la ingeniería, se utilizan para diseñar sistemas de fluidos y de transporte. Esta sería la rama que más nos interesaría ya que podríamos llevar su aplicación al ámbito naval.

4.2.1. Superficies. Ejemplos

Una superficie parametrizada es una función $\Phi : \Omega \subseteq \mathbb{R}^2 \to \mathbb{R}^3$ de manera que dados $(u,v) \in \Omega$ le corresponde un punto de superficie

$$\Phi(u,v) = \big(x(u,v), y(u,v), z(u,v)\big).$$

En general y al igual que ocurría con las curvas, se tiende a confundir la superficie con su gráfica en el espacio. Dada la superficie Φ definimos su gráfica como

$$graf(\Phi) = \{\Phi(u,v): (u,v) \in \Omega\},$$

que es el conjunto de $\mathbb{R}^3$ que representa a Φ. Antes de introducir nuevas definiciones, veamos un ejemplo de superficie.

Supongamos que Φ es de clase C^1, entonces podemos definir los vectores tangentes a la superficie en un punto como:

$$\Phi_u = (x_u, y_u, z_u) \quad , \quad \Phi_v = (x_v, y_v, z_v)\ .$$

Si además pasa todo (u,v) $\Phi_u \times \Phi_v \neq (0,0,0)$ entonces la superficie es regular y se puede definir el vector normal en todos los puntos de la superficie como:

$$n(u,v) = \Phi_u \times \Phi_v \quad \text{ó} \quad n(v,u) = \Phi_v \times \Phi_u \,.$$

Normalmente, casi todas las superficies son orientables, es decir, tienen una cara exterior y un interior, y uno de los $n(u,v)$ o $n(v,u)$ es el vector normal exterior y el otro el interior.

Notad que fijado un valor de $u = u_0$, la expresión $C^1(v) = \Phi(u_0, v)$ es una curva coordenada de la superficie y Φ_v el vector tangente a la curva en (u_0, v).

Igualmente fijando un valor de $v_0 = v$, la expresión $C^2(u) = \Phi(u, v_0)$ es una curva coordinada de la superficie Φ_u el vector tangente a la curva (u, v_0).

Ejemplo *Figura 17.* $\Phi(u,v) = (5\sin(v)\cos(u), 5\sin(v)\sin(u), 5\cos(v))$

$u \in [0,2\pi]$, $v \in [0,\pi]$ define una esfera de centro el origen y radio 5.

$\Phi(\pi/4, v) = (\frac{5\sqrt{2}}{2}\sin(v), \frac{5\sqrt{2}}{2}\sin(v), 5\cos(v))$, $v \in [0,\pi]$ es un meridiano de la esfera.

$\Phi(u, \pi/2) = (5\cos(u), 5\sin(u), 0)$, $u \in [0,2\pi]$ que es el paralelo ecuatorial de la esfera.

Los vectores $\Phi_u(\pi/4, \pi/2)$, $\Phi_v(\pi/4, \pi/2)$ representan los vectores tangentes a la esfera en el punto.

Hacemos notar que la longitud del vector normal en una esfera siempre vale $\|\Phi_u \times \Phi_v\| = \mathbb{R}^2 \sin(v)$.

Así, en este caso:

$$\|\Phi_u(\pi/4, \pi/2) \times \Phi_v(\pi/4, \pi/2)\| = \|\Phi_v(\pi/4, \pi/2) \times \Phi_u(\pi/4, \pi/2)\| = 25\sin(\pi/2) = 25.$$

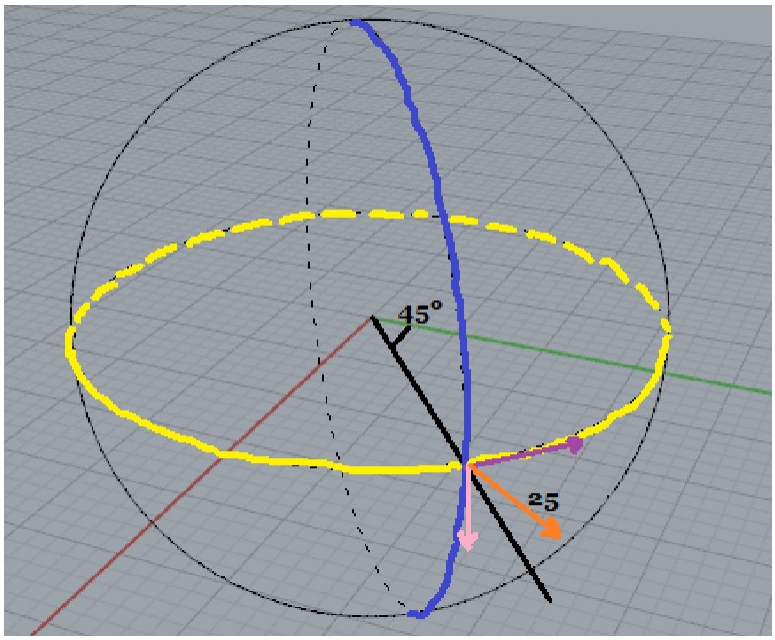


Figura 17. Intersección entre el meridiano y el paralelo de una esfera

4.2.2. Integral de superficie en un campo escalar

Sean $\Phi: \Omega \subset \mathbb{R}^2 \to \mathbb{R}^3$ una superficie regular y $f: A \subseteq \mathbb{R}^3 \to \mathbb{R}$ un campo escalar de manera que $graf(\Phi) \subset A$. Se define la integral de *f* sobre la superficie Φ como:

$$\iint_{\Phi} f dS = \iint_{\Omega} f(\Phi(u,v)) \cdot \|T_u x T_v\| du dv.$$

Si la superficie es regular a trozos, entonces:

$$\iint_{\Phi} f dS = \sum_{i=0}^{n} \iint_{\Phi_i} f dS,$$

donde Φ_i son cada una de las superficies regulares en las que se descompone Φ.
Si $f(x,y,z) = 1$, la integral se corresponde con el área de la superficie Φ.

Ejemplo *Figura 18*. Queremos evaluar la siguiente integral de superficie,

$$\iint_{\varphi(A)} xyz\, dS,$$

esta integral hace referencia a la tapa de una caja, representada en la *Figura 18*.

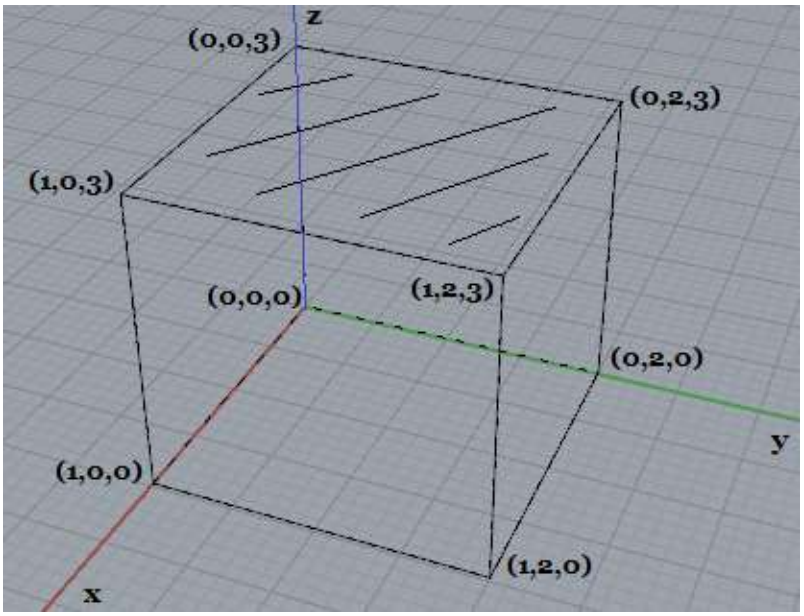


Figura 18. Superficie de la tapa de una caja.

Como vemos en la *Figura 18*, $z = 3$ y está en función de x e y por lo que $z = 3 = f(x,y)$. Siendo entonces el diferencial de superficie:

$$dS = \sqrt{1 + (\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2}\, dxdy = \sqrt{1 + 0^2 + 0^2}\, dxdy = dxdy,$$

ahora integraremos la superficie entre los límites de su proyección en los ejes x e y, siendo su parametrización

$$\varphi(x, y) = (x, y, 3),$$

quedando de este modo finalmente la integral siguiente:

$$\iint_{\varphi(A)} xyz\, dS = \int_0^2 \int_0^1 3xy\, dxdy.$$

Ahora pasaremos a explicar otro ejemplo más complejo.

Ejemplo *Figura 19*. Se pide evaluar la integral de la superficie lateral de un cono.

donde S es la superficie lateral del cono $x^2 + y^2 = z^2$ que yace entre los planos $z = 0$ y $z = 1$. Como lo indica la figura 1.

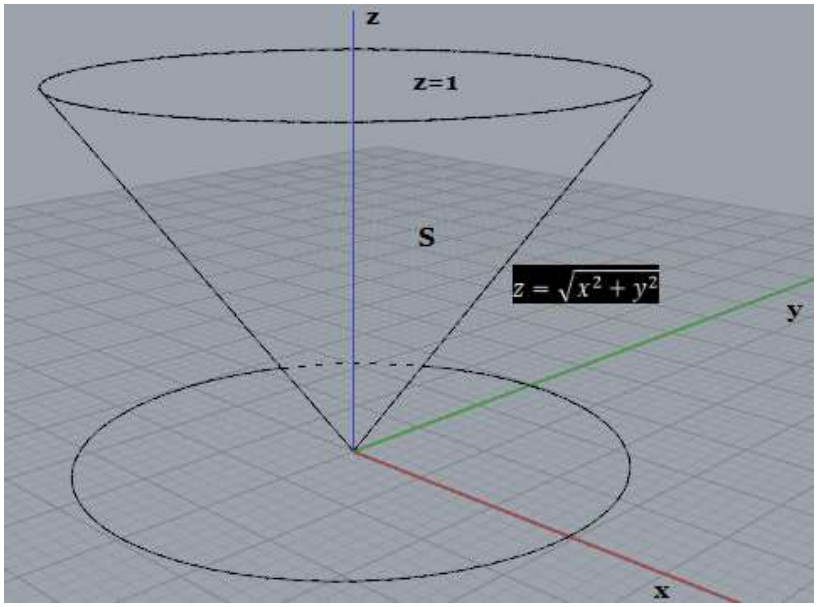


Figura 19. Cono y su proyección sobre los ejes *XY*.

Proyectando la superficie sobre el plano *XY*, y considerando que la superficie está entregada en su forma implícita, esto es $x^2 + y^2 - z^2 = 0$, esto es

$$F(x, y, z) = x^2 + y^2 - z^2 = 0,$$
$$F_x = 2x; F_y = 2y; F_z = -2z,$$

entonces, aplicando la fórmula

$$\iint_S y dS,$$

tenemos que

$$dS = \sqrt{\frac{x^2}{z^2} + \frac{y^2}{z^2} + 1} dxdy = \sqrt{\frac{x^2 + y^2}{z^2} + 1} dxdy = \sqrt{2} dxdy.$$

De modo que debemos integrar

$$\iint_S x^2 y^2 z^2 dS = \iint_R x^2 y^2 (x^2 + y^2) \sqrt{2} dxdy,$$

siendo R la región del interior de la circunferencia $x^2 + y^2 = 1$. Esta integral la podemos desarrollar utilizando el cambio de variable a coordenadas polares, esto es

$$x = \rho \cos\theta \; ; y = \rho \, sen\theta,$$

donde $dx \, dy = \rho \, d\rho \, d\theta$. Entonces la integral nos queda como

$$\iint_S x^2 y^2 z^2 \, dS = \iint_R x^2 y^2 (x^2 + y^2) \sqrt{2} dxdy = \int_0^{2\pi} \int_0^1 \sqrt{2} \rho^1 cos^2\theta sen^2\theta \, \rho d\rho d\theta.$$

Y esta doble integral vale

$$\int_0^{2\pi} \int_0^1 \sqrt{2} \rho^1 cos^2\theta sen^2\theta \, d\rho d\theta = \frac{\pi\sqrt{2}}{32}.$$

4.2.2. Cambio de variable (coordenadas polares). Independencia de la parametrización escogida.

Sea $\Phi: \Omega \subset \mathbb{R}^2 \to \mathbb{R}^3$ una superficie regular dada por $\Phi(u, v) = (x(u, v), y(u, v), z(u, v)$ y supongamos que hacemos un cambio de variables $\varphi: A \to \Omega$ de clase C^1 y biyectivo tal que

$$\begin{cases} u = u(a, b) \\ v = v(a, b). \end{cases}$$

Como vamos a comprobar a continuación, el cambio no afecta a la integral de superficie, es decir, ésta es independiente de la parametrización que elijamos. Para demostrarlo, sea $\theta: A \to \mathbb{R}^3$ la superficie dada por

$$\theta(a, b) = \Phi\big(u, (a, b), v(a, b)\big), (a, b) \in A,$$

veamos que

$$\iint_{\Phi} f dS = \iint_{\theta} f dS.$$

hay que calcular

$$T_a = \frac{\partial u}{\partial a} T_u + \frac{\partial v}{\partial a} T_v,$$

análogamente

$$T_b = \frac{\partial u}{\partial b} T_u + \frac{\partial v}{\partial b} T_v,$$

entonces,

$$\|T_a x T_b\| = |\det(J\varphi(a,b))| \cdot \|T_u x T_v\|.$$

Y por la fórmula del cambio de variable en la integral doble

$$\begin{aligned}\iint_{\Phi} f dS &= \iint_{\Omega} f(\Phi(u,v)) \left\| T_u x T_v \right\| du dv \\ &= \iint_{A} f(\Phi(u(a,b), v(a,b))) \frac{\|T_a x T_b\|}{|\det(J\varphi(a,b))|} |\det(J\varphi(a,b))| da db \\ &= \iint_{A} f(\theta(a,b)) \|T_a x T_b\| da db,\end{aligned}$$

con lo que comprobamos que la integral de superficie no depende de la parametrización elegida para calcularla.

4.2.3. Integral de superficie de un campo vectorial

Sean $\Phi: \Omega \subseteq \mathbb{R}^2 \to \mathbb{R}^3$ una superficie regular y $F: D \subseteq \mathbb{R}^3 \to \mathbb{R}^3$ un campo vectorial de manera que la gráfica $graf(\Phi) \subset D$. Se define la integral de F sobre la superficie Φ como

$$\iint_{\Phi} F dS = \iint_{\Omega} < F(\Phi(u,v)), T_u x T_v > du dv.$$

Si Φ es una superficie regular a trozos, entonces

$$\iint_{\Phi} F dS = \sum_{i=0}^{n} \iint_{\Phi_i} F dS,$$

donde Φ_i son cada una de las superficies regulares en la que se descompone Φ. La integral de superficie de un campo vectorial se corresponde físicamente con la noción de flujo de un campo a través de una superficie.

Ejemplo *Figura 20*. Integral de superficie de la sección de un cilindro.

como hemos visto en la teoría, la integral de superficie que tenemos que calcular sería

$$\iint_{\Phi(A)} F \cdot n \, dS,$$

y el campo vectorial para el que queremos calcularla en este ejemplo sería

$$F(x, y, z) = (xz, y, x + z).$$

La superficie con la que vamos a trabajar va a ser $y + z = 40$, estando esta dentro de $x^2 + y^2 = 9$ que quedaría representada en la *Figura 20*.

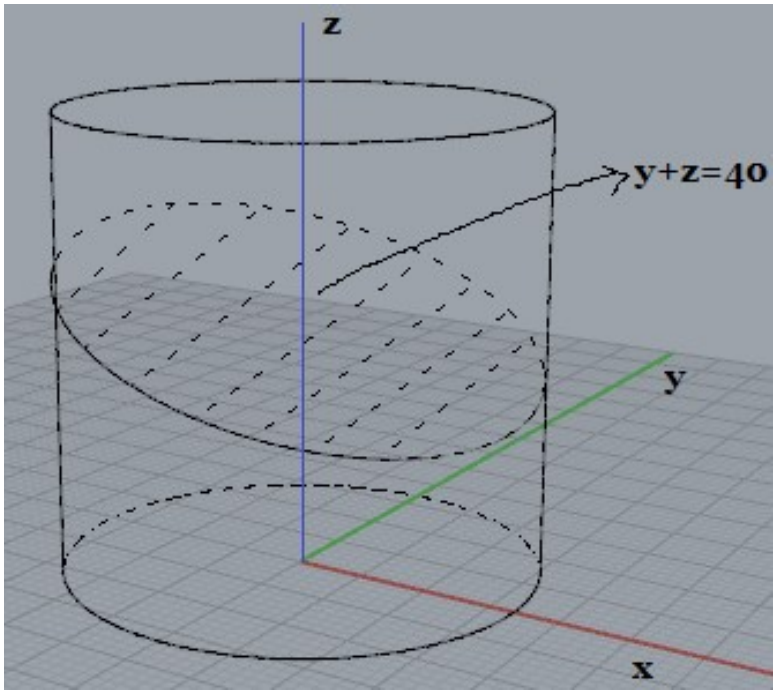


Figura 20. Superficie dentro de un cilindro.

Ahora procederemos a proyectar la superficie sobre el plano *XY*, siendo la parametrización

$$\varphi(x, y) = (x, y, 40 - y),$$

y variando *(x, y)* en el círculo $x^2 + y^2 \leq 1$. Calculamos el vector normal *n*

$$\frac{\partial f}{\partial x} \times \frac{\partial f}{\partial y} = \begin{vmatrix} i & j & k \\ 1 & 0 & 0 \\ 0 & 1 & -1 \end{vmatrix} = (0,1,1).$$

La proyección de la superficie sobre el plano *XY* será la que muestra la *Figura 21*.

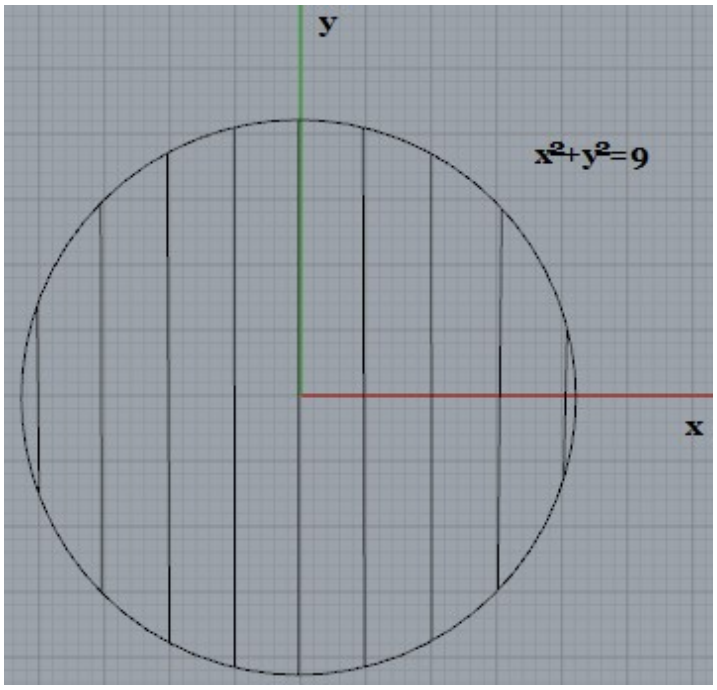


Figura 21. Proyección de la superficie encerrada en un cilindro sobre el plano *XY*.

Entonces, vamos a calcular la integral de superficie, que será, teniendo en cuenta la compuesta de $F(\varphi(x,y) = F(x,y,40-y) = (x\,(40-y), y, x+40-y)$.

$$\iint_{\varphi(A)} F \cdot n\, dS = \int_{-3}^{3} \int_{-\sqrt{9-x^2}}^{\sqrt{9-x^2}} (x\,(40-y)\,, y, x+40-y) \cdot (0,1,1)\, dydx = \int_{-3}^{3} \int_{-\sqrt{9-x^2}}^{\sqrt{9-x^2}} (x+40)\, dydx.$$

Y por último pasamos la integral a coordenadas polares, quedando finalmente

$$\int_{-3}^{3} \int_{-\sqrt{9-x^2}}^{\sqrt{9-x^2}} (x+40)\, dydx = \int_{0}^{2\pi} \int_{0}^{3} (rcos\theta + 40) r\, drd\theta.$$

Ejemplo *Figura 22.* Cálculo de la integral de superficie del campo vectorial $F(x,y,z) = xi + yj + zk$ en la superficie dada por $x + y + z = 1,\ x \geq 0,\ y \geq 0,\ z \geq 0$ representada en la *Figura 22*.

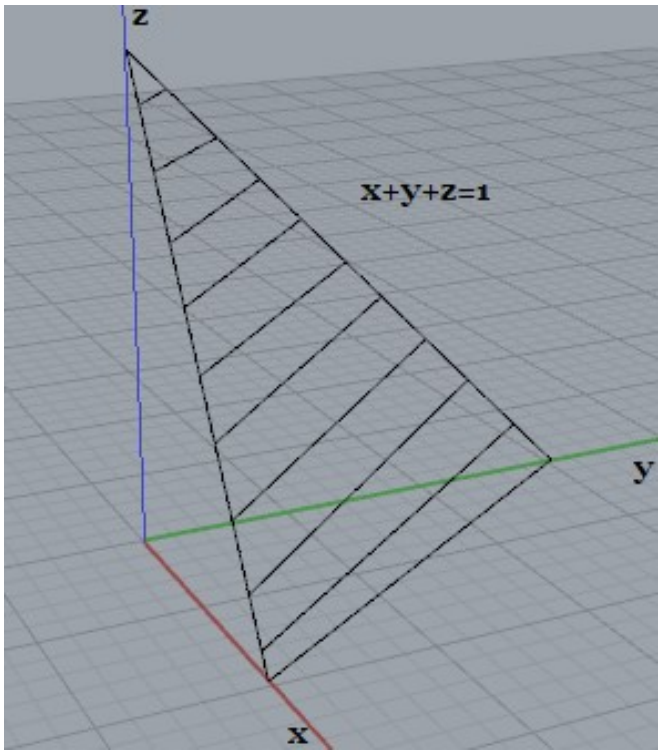


Figura 22. Superficie del plano $x + y + z = 1$.

Siendo la fórmula de la integral de superficie en un campo vectorial

$$\iint_{\varphi(A)} F \cdot n \, dS.$$

En este caso la calcularemos para el campo vectorial

$$F(x,y,z) = xi + yj + zk.$$

Lo primero que tenemos que hacer es parametrizar la superficie, proyectando sobre el plano *XY*, siendo esta

$$\varphi(x,y) = (x,y,1-x-y),$$

tendremos también que calcular el vector normal

$$\frac{\partial f}{\partial x} \times \frac{\partial f}{\partial y} = \begin{vmatrix} i & j & k \\ 1 & 0 & -1 \\ 0 & 1 & -1 \end{vmatrix} = (1,1,1).$$

(x, y) en *T*, siendo $T = \{(x,y)\ en\ \mathbb{R}^2 : 0 \leq x \leq 1,\ 0 \leq y \leq 1-x\}$

La proyección de la superficie sobre el plano *XY* se representa en la *Figura 23*.

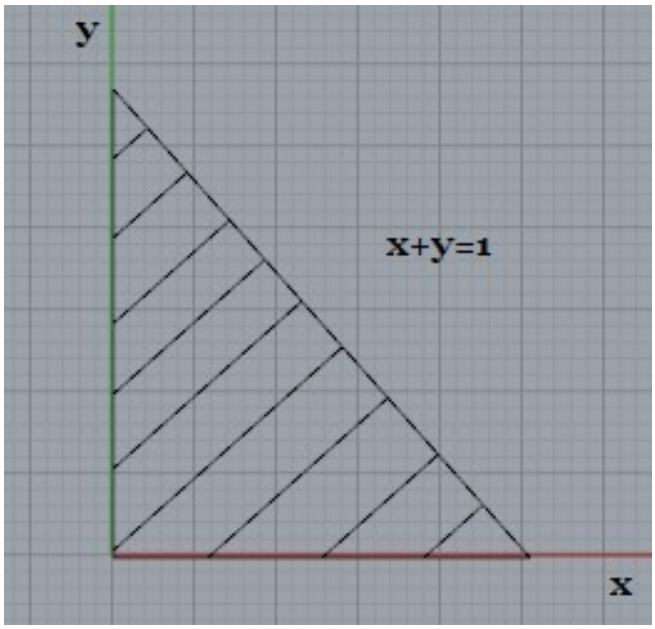


Figura 23. Proyección de la superficie sobre el plano *XY*.

Quedando de esta manera, la integral a calcular como

$$\iint_{\varphi(A)} F \cdot n \, dS = \int_0^1 \int_0^{1-x} (x + y + 1 - x - y) \, dydx,$$

$$\int_0^1 \int_0^{1-x} dydx = 1/2.$$

Queda anotar que cuando se realiza la integral para el cálculo de los campos vectoriales como escalares, el resultado de esta es el mismo independientemente de la parametrización en coordenadas cartesianas o en coordenadas polares, lo único que varía con las diferentes parametrizaciones será la orientación del vector normal.

4.3. REGLAS DE NEWTON COTES CERRADAS PARA VARIAS VARIABLES

De este punto para atrás hemos visto las integrales simples, ahora vamos a estudiar las integrales de varias variables, estas integrales se utilizan para realizar calculos más complejos como el cálculo de superficies o momentos de inercia. Vamos a introducir la integración doble en su forma teórica y más adelante realizaremos dos programas en código Matlab para la Regla de Trapecios y Regla de Simpson.

La forma en la que se expresa la integral doble si $D = \{(x, y) \in \mathbb{R}^2 : a \leq x \leq b, l(x) \leq y \leq u(x)\}$, entonces:

$$\iint_D f(x, y) \, dxdy = \int_a^b \left(\int_{l(x)}^{u(x)} f(x, y) \, dy \right) dx = \int_a^b g(x) \, dx,$$

con

$$g(x) = \int_{l(x)}^{u(x)} f(x,y)\, dy.$$

donde la región considerada está representada por los extremos del intervalo *a*, *b* en la variable *x* y por las funciones *l(x)* y *u(x)*. Las variables *n* y *m* ,respectivamente, definen la cantidad de veces que se aplica la Regla Simple en una variable en cada dirección *x* ó *y*.

Definimos:

$$h = \frac{b-a}{2n}, \qquad k = \frac{u-l}{2m},$$

donde, de ahora en adelante, omitiremos la dependencia funcional en la notación *l(x)* y *u(x)* de la variable *x*, escribiendo únicamente *l* y *u*.

En lo que sigue vamos a realizar la explicación para la Regla de los Trapecios y para la Regla de Simpson, siendo las ideas válidas igualmente para cualquiera de las otras reglas.

Considerando $j = 0, 1, 2 \ldots m$ y como valor de los distintos puntos $y_j = c + jk$:

$$\int_l^u f(x,y)\, dy \approx \frac{k}{3}\left[f(x,y_0) + 2\sum_{j=1}^{m-1} f\left(x,y_{2j}\right) + 4\sum_{j=1}^{m} f\left(x,y_{2j-1}\right) + f(x,y_{2m}) \right].$$

A este valor tendríamos que sumarle el valor del error correspondiente a la aproximación de la integral realizada:

$$Error = -\frac{(u-l)k^4}{180}\frac{\delta^{IV} f(x,\eta)}{\delta y^4},$$

con $\mu \in]a,b[$.

Ahora vamos a generar los programas de la Regla de Trapecios en 2 variables y Regla de Simpson en 2 variables. En ellos, además de ingresar los datos típicos de los intervalos (a, b, l y u), el número de repeticiones del método (m y n) y la función a integrar *f*, también se introduce *N*, que representa el número de subintervalos en los que se divide cada intervalo.

4.3.1. Regla de los Trapecios para varias variables (Matlab).

En este programa, vamos a implementar un código en lenguaje Matlab que permite aproximar una integral doble mediante la Regla de los Trapecios.

El programa es como sigue:

```
function [int]=Trapecios_P1_2var(f,c,d,a,b,m,n)

% Esta función calcula una integral doble por el método de los Trapecios
%
% Datos de entrada:
% f: cadena de caracteres conteniendo la función a integrar
% c: cadena de caracteres conteniendo el extremo inferior de la primera integral
a calcular
% d: cadena de caracteres conteniendo el extremo superior de la primera integral
a calcular
% a: extremo inferior de la segunda integral a calcular
% b: extremo superior de la segunda integral a calcular
% m,n: número de puntos en que se parte cada intervalo según la dimensión
%
% Ejemplo:
% Comparamos la integral numérica con la exacta calculada con el paquete
% de simbólico de Matlab
% format long
% [int]=double(Trapecios_P1_2var(sin(x1)+2*x2,x1,2*x1,0,1,100,100))
% [int]=double(Trapecios_P1_2var(x1*exp(x1+x2),x1,2*x1,0,1,100,100))
% Comprobación con el cálculo de las integrales exactas
% syms x y
% double(int(int(sin(x)+2*y,y,x,2*x),x,0,1))
% double(int(int(x*exp(x+y),y,x,2*x),x,0,1))

syms x1 x2;

h=(b-a)/n;

% Iniciamos los acumuladores

I1=0;
I2=0;

% calculamos el valor de la integral interior para x=a

% extremos para la integral en x2 para cada x1=a
dx=subs(d,{x1},{a}); % extremo superior
cx=subs(c,{x1},{a}); % extremo inferior

% paso de integración para la integral en la segunda variable
kx=(dx-cx)/m;

% realizamos la integral en x2 por trapecios de 1 variable
K1=subs(f,{x1,x2},{a,cx})+subs(f,{x1,x2},{a,dx});
K2=0;

for j=1:m-1
    y=cx+j*kx;
    z=subs(f,{x1,x2},{a,y});
    K2=K2+z;
```

```
end

% L guarda el resultado de la integral en x2 para cada valor de x1
L=kx*(K1/2+K2);

% acumulamos para calcular la integral en x1
I1=I1+L;

% bucle para la integral en la primera variable x1
for i=1:n-1
    % nodos en el eje x1
    x=a+i*h;
    % extremos para la integral en x2 para cada x1
    dx=subs(d,{x1},{x}); % extremo superior
    cx=subs(c,{x1},{x}); % extremo inferior
    % paso de integración para la integral en la segunda variable
    kx=(dx-cx)/m;
    % realizamos la integral en x2 por trapecios de 1 variable
    K1=subs(f,{x1,x2},{x,cx})+subs(f,{x1,x2},{x,dx});
    K2=0;
    for j=1:m-1
        y=cx+j*kx;
        z=subs(f,{x1,x2},{x,y});
        K2=K2+z;
    end
    % L guarda el resultado de la integral en x2 para cada valor de x1
    L=kx*(K1/2+K2);
    % acumulamos para calcular la integral en x1
    I2=I2+L;
end

% extremos para la integral en x2 para cada x1=b
dx=subs(d,{x1},{b}); % extremo superior
cx=subs(c,{x1},{b}); % extremo inferior

% paso de integración para la integral en la segunda variable
kx=(dx-cx)/m;

% realizamos la integral en x2 por simpson de 1 variable
K1=subs(f,{x1,x2},{b,cx})+subs(f,{x1,x2},{b,dx});
K2=0;

for j=1:m-1
    y=cx+j*kx;
    z=subs(f,{x1,x2},{b,y});
    K2=K2+z;
end

% L guarda el resultado de la integral en x2 para cada valor de x1
L=kx*(K1/2+K2);

% acumulamos para calcular la integral en x1
I1=I1+L;

% Valor de la integral doble
int=h*(I1/2+I2);
end
```

4.3.2. Regla de Simpson para varias variables (Matlab).

En esta ocasión, hemos realizado un código en lenguaje Matlab mediante el cual aproximaremos una integral doble mediante la Regla de Simpson.

A continuación, presentamos el código:

```
function [int]=Simpson_P1_2var(fsimp,c,d,a,b,m,n)

% Esta función calcula una integral doble por el método de Simpson
%
% Variables de entrada:
%
% fsimp: cadena de caracteres conteniendo la función integrando
% c: cadena de caracteres conteniendo el extremo inferior de la primera integral
a calcular
% d: cadena de caracteres conteniendo el extremo superior de la primera integral
a calcular
% a: extremo inferior de la segunda integral a calcular
% b: extremo superior de la segunda integral a calcular
% m,n: número de veces que se aplica Simpson simple en cada intervalo según la
dimensión
%
% Ejemplo:
% Comparamos la integral numérica con la exacta calculada con el paquete
% de simbólico de Matlab
% format long
% [int]=double(Simpson_P1_2var(x1*exp(x1+x2),x1,2*x1,0,1,100,100))
% Comprobación con la integral exacta:
% syms x y
% double(int(int(x*exp(x+y),y,x,2*x),x,0,1))

syms x1 x2;

h=(b-a)/(2*n);
I1=0;
I2=0;
I3=0;

% bucle para la integral en la primera variable x1
for i=0:(2*n)

    % nodos en el eje x1
    x=a+i*h;

    % extremos para la integral en x2 para cada x1
    dx=subs(d,{x1},{x}); % extremo superior
    cx=subs(c,{x1},{x}); % extremo inferior

    % paso de integración para la integral en la segunda variable
    kx=(dx-cx)./(2*m);

    % realizamos la integral en x2 por Simpson de 1 variable
    K1=subs(fsimp,{x1,x2},{x,cx})+subs(fsimp,{x1,x2},{x,dx});
    K2=0;
    K3=0;

    for j=1:(2*m-1)
```

```
        y=cx+j*kx;
        z=subs(fsimp,{x1,x2},{x,y});

        if gcd(2,j)==2
            K2=K2+z;
        else
            K3=K3+z;
        end
    end

    % L guarda el resultado de la integral en x2 para cada valor de x1
    L=(kx/3)*(K1+2*K2+4*K3);

    % acumulamos para calcular la integral en x1
    if i==0 || i==2*n
        I1=I1+L;
    else

        if gcd(2,i)==2
            I2=I2+L;
        else
            I3=I3+L;
        end
    end

    % Valor de la integral doble
    int=(h/3)*(I1+2*I2+4*I3);
end

end
```

5. APLICACIONES EN EL ÁMBITO NAVAL

En este capítulo vamos a centrarnos en los aspectos que más nos atañen de la integración numérica, es decir, su aplicación al ámbito naval, tanto a la rama de la propulsión como a la rama de la hidrostática del buque.

Como veremos a continuación, los métodos de integración numérica tienen infinidad de aplicaciones prácticas, nosotros vamos a analizar algunas de las más comunes. Primero vamos a introducir algunas de las magnitudes elegidas para su estudio y posteriormente vamos a realizar algunos ejemplos en lenguaje Matlab.

5.1. INTEGRACIÓN NUMÉRICA APLICADA A LA PROPULSIÓN Y A FENÓMENOS HIDROSTÁTICOS

La Arquitectura Naval es la rama de la ingeniería que se centra en el conocimiento especializado necesario para la construcción de los buques. Este conocimiento se puede clasificar en dos bloques generales.

a) Teoría del buque. Es el estudio del barco únicamente como un cuerpo flotante que interactúa con un líquido, y por lo tanto, solo considera las superficies externas del mismo, que son las que determinan sus condiciones de resistencia al movimiento en general, el avance, la evolución, etc. y su estabilidad. A su vez, dentro de ella se hacen dos subdivisiones:

a1) La Teoría del buque 1, que es la parte que se ocupa del equilibrio y la estabilidad del buque. Para ello se supone que, en general, el buque o flotador está en aguas tranquilas.

a2) La Teoría del buque 2 se encarga de estudiar la resistencia a la marcha, la propulsión, la maniobrabilidad y el comportamiento en el mar.

b) La Construcción naval estudia el buque desde la perspectiva de la construcción, y establece la forma y los espesores, o escantillones, que deben tener sus elementos, de acuerdo con los esfuerzos previstos a los que se espera que estén sometidos. Es decir, estudia la estructura del buque y la disposición del barco para los fines a los que se destina.

En la *Figura 24* podemos ver las partes más representativas de un buque, magnitudes que servirán para los cálculos hidrodinámicos e hidrostáticos.

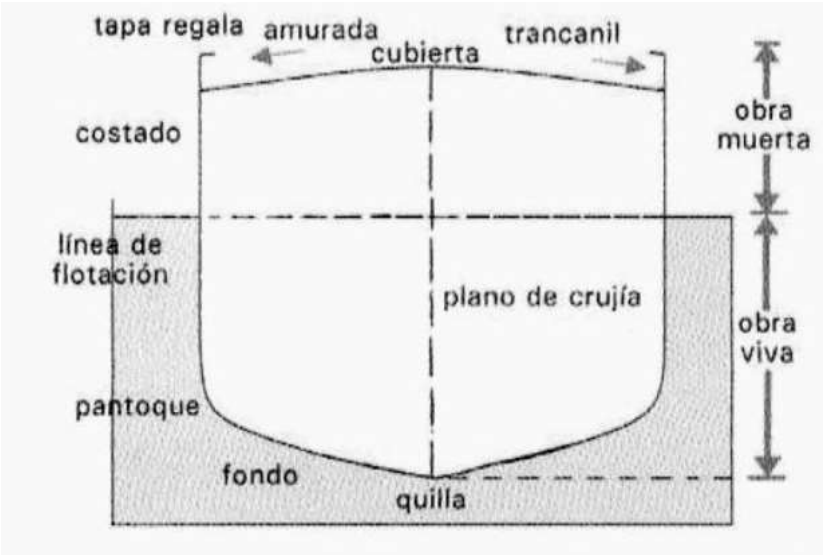


Figura 24. Buque con sus partes, líneas y planos.

En los siguientes puntos vamos a profundizar en algunos de los campos del mundo naval en los que se aplica la integración numérica.

5.1.1. Integración numérica aplicada a fenómenos hidrostáticos

En Construcción Naval en general, y sobre todo en Teoría del Buque, se necesita continuamente el cálculo de áreas como una herramienta de trabajo. Estos métodos son aplicados para calcular por ejemplo las fuerzas que afectan a los buques, como vemos representado en las siguientes figuras.

En las siguientes figuras, vemos las fuerzas (Peso y Empuje) que afectan al buque, en sentido transversal (*Figura 25*) y en sentido longitudinal (*Figura 26*).

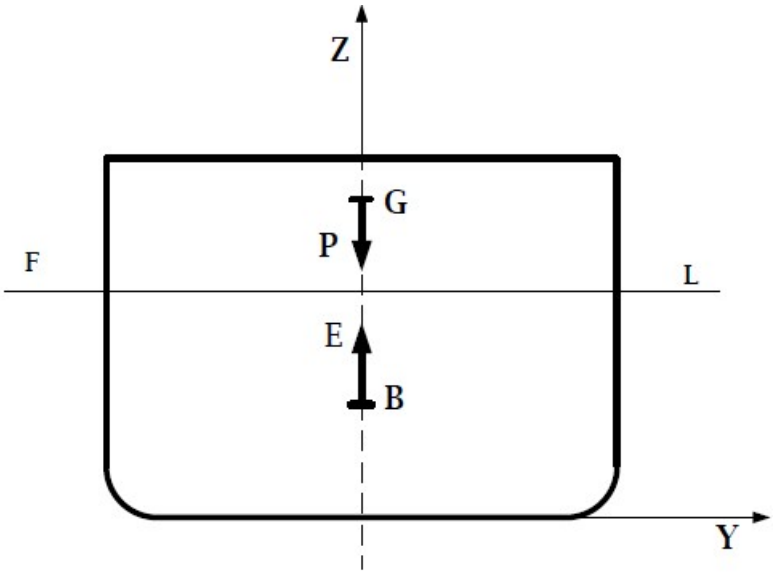


Figura 25. Fuerzas que afectan a un buque en el plano transversal.

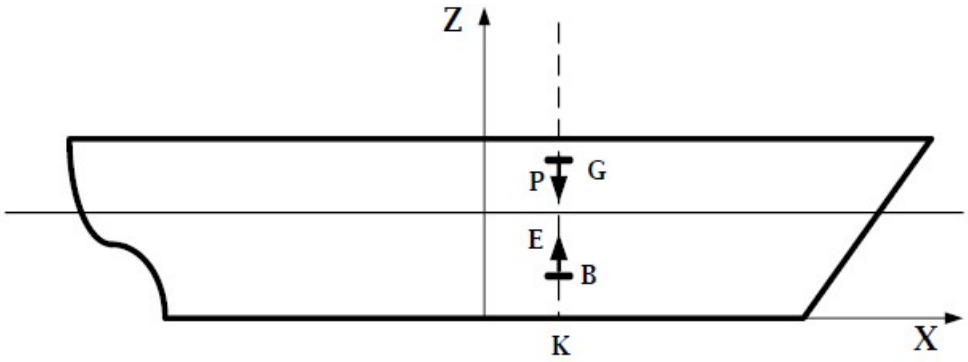


Figura 26. Fuerzas que afectan a un buque en el plano longitudinal.

Estudiemos ahora algunas de las variables hidrostáticas más comunes, las cuales se aproximan usando los métodos de integración numérica que estamos estudiando.

5.1.1.1. Cálculo del Área de Flotación

La teoría nos dice que:

Para un calado específico, el área de flotación se obtiene integrando las diferentes mangas o semimangas (siempre que la flotación del buque sea simétrica a lo largo de la eslora) en la línea de agua correspondiente. (véase *Figura 27*)). Siendo la fórmula para calcular ésta:

$$A_F = \int_{L_f} y\, dx,$$

siendo L_f la eslora para una determinada flotación e y las semimangas.

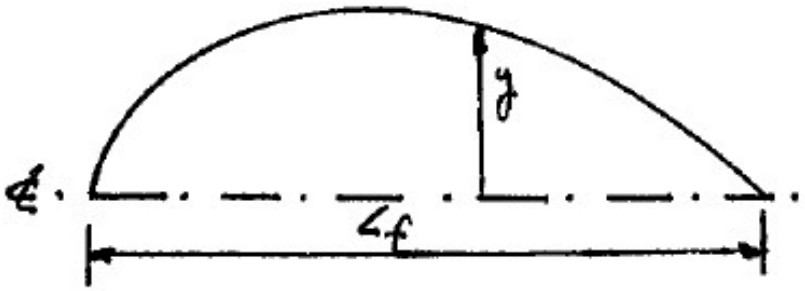


Figura 27: Semimangas a lo largo de la eslora para una flotación elegida.

Normalmente estas áreas se calculan para las diferentes líneas de agua, comenzando por la línea de agua 0, lo que nos da una serie de pares de valores (T, A_F), que, al trasladarlos al papel milimetrado, tendremos un conjunto de puntos. La curva $A_F = f(T)$ se obtiene trazando la curva que pasa por todos ellos.

En la *Figura 28* se ha dibujado una curva que corresponde a un buque con astilla muerta, en el cual $A_F = 0$ para $T = 0$.

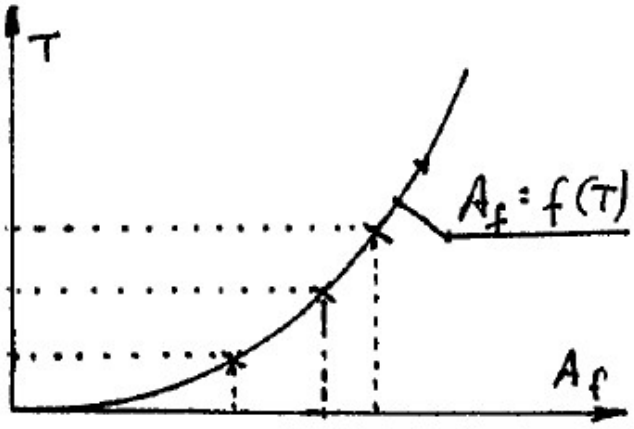


Figura 28. Curva de áreas de flotación para un buque de astilla muerta.

5.1.1.2. Cálculo de la abscisa del centro de gravedad de la flotación (*c.d.f.*)

La teoría nos dice que:

La posición del *c.d.f.* del área de las distintas flotaciones, se obtiene dividiendo el momento estático de esta área, respecto del origen que hayamos considerado, entre el área. Por mayor sencillez de cálculo, vamos a tomar como origen la sección media, siendo la fórmula.

$$\otimes F = \frac{m_{\otimes}}{A_F}.$$

Si consideramos como origen la perpendicular de popa, el proceso es exactamente el mismo. Como hemos visto en el anterior ejemplo, siempre que las flotaciones sean simétricas, trabajaremos con semimangas y duplicaremos el resultado de la integral para obtener los valores totales para el buque. Siendo las fórmulas:

$$m_{\otimes} = 2 \cdot \int_{L_f} xy\, dx \qquad \text{y} \qquad A_F = 2 \cdot \int_{L_f} y\, dx,$$

siendo L_f la eslora en una determinada flotación e *y* las semimangas.

Estos valores serán positivos o negativos dependiendo de si están a proa (Pr) o a popa (Pp) de la sección media ⊗, que es nuestro nuevo origen, como se muestra en la *Figura 29*.

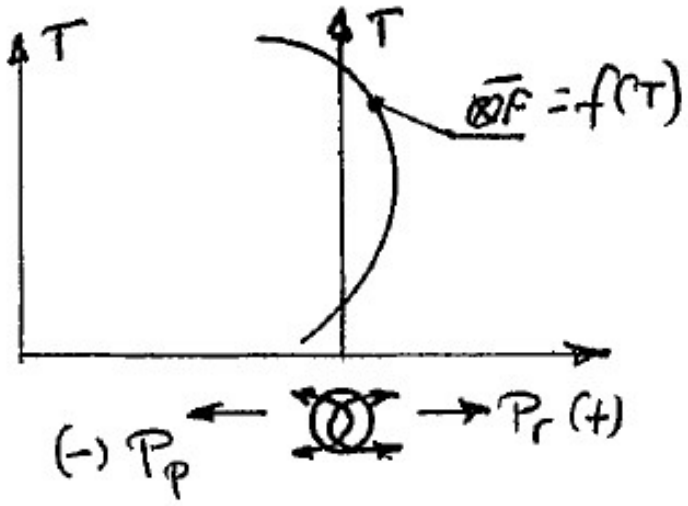


Figura 29. Valores del c.d.f. positivos y negativos en relación con ⊗.

5.1.1.3. Cálculo del Volumen de Carena

Partiendo del área de las secciones:

$$\partial\nabla = A_F \cdot dx, \qquad \text{luego} \qquad \nabla = \int_L A_F \; dx,$$

quedando esta ecuación representada en la *Figura 30*.

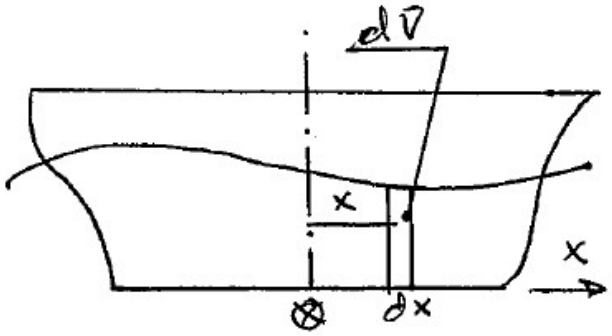


Figura 30. Representación del Volumen de Carena.

Esta integral se calculará a lo largo de la eslora, sobre el plano base. Si estos valores los multiplicamos por el peso específico obtendríamos los empujes o desplazamientos.

$$\Delta = \gamma \cdot \nabla = \int_L A_F \cdot \gamma \; dx.$$

En este caso, a la función $A_F \cdot \gamma = f(T)$ para cada sección se la identifica como curva o función de empujes por unidad de longitud.

5.1.1.4. Altura del centro de carena (*c.d.c.)* sobre la base.

Por definición de *c.d.c.*

$$KB = \frac{M_k}{\nabla},$$

siendo M_k el momento estático del volumen de carena (∇) respecto al plano base (ver *Figura 31*).

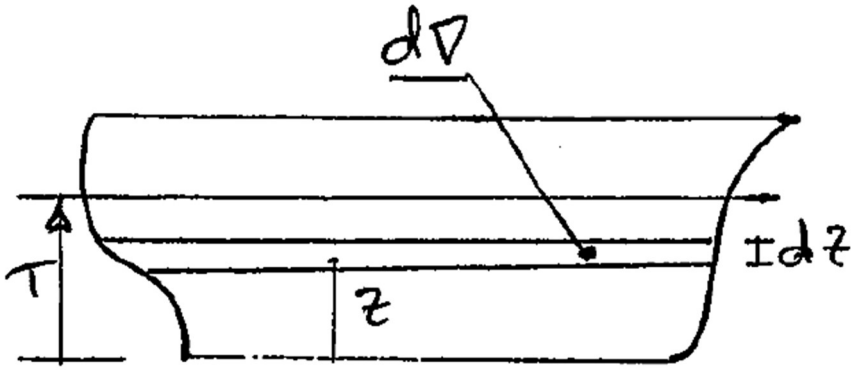


Figura 31. Representación de *c.d.c.*

Si partimos de las áreas de las flotaciones:

$$d\nabla = A_f \cdot dz, \qquad \text{y} \qquad dM_k = d\nabla \cdot z = A_f \cdot dz \cdot z,$$

con lo que

$$M_k = \int_0^T A_f \cdot z \cdot dz, \qquad \text{y} \qquad \nabla = \int_0^T A_f \cdot dz.$$

5.1.1.5. Radio metacéntrico transversal (BM_t).

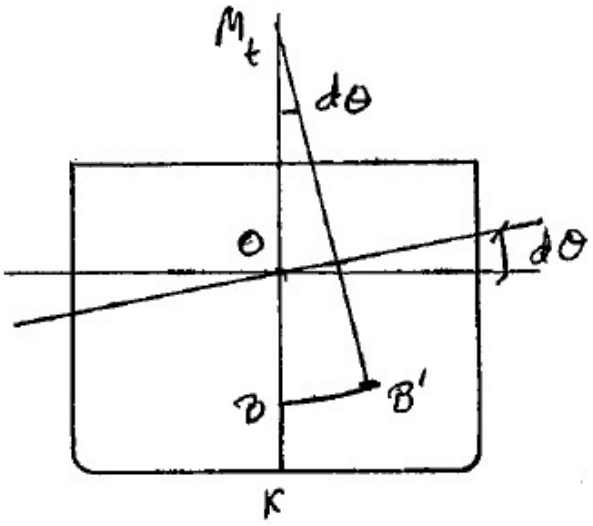


Figura 32. Representación del centro metacéntrico transversal.

El radio metacéntrico transversal (BM_t o simplemente *BM*) es el radio de curvatura de la curva que describe el *c.d.c.* al girar el buque un ángulo $'\partial\sigma'$ infinitesimal en el plano transversal, permaneciendo constante el volumen sumergido (*Figura 32*), y se halla mediante la expresión:

$$BM_t = \frac{I_t}{\nabla},$$

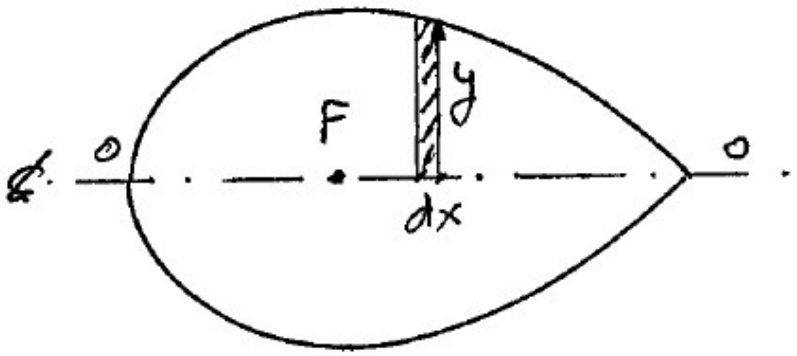


Figura 33. Representación del eje 00 tomado para calcular I_t.

siendo I_t el momento de inercia del área de la flotación, respecto de un eje 00 perpendicular al plano de giro y que pasa por su *c.d.f.* En nuestro caso, este eje 00 coincide con la línea de crujía de la flotación (flotaciones simétricas), por lo tanto el momento de inercia como queda representado en la *Figura 33*.

$$I_t = 2\int_{L_f} \frac{1}{3} y^3 \, dx,$$

siendo *y* las semimangas.

5.1.2. Integración numérica aplicada a la propulsión

En este epígrafe, el principal caso práctico que vamos a introducir, si bien desde un punto de vista teórico, es el del cálculo de las presiones generadas por la pala de una hélice, cuyas partes vienen reflejadas en la *Figura 34*, que se mueve en un determinado fluido. Nosotros al centrarnos en el ámbito naval, nos referiremos, como es lógico, a la pala de una hélice que propulsa un buque.

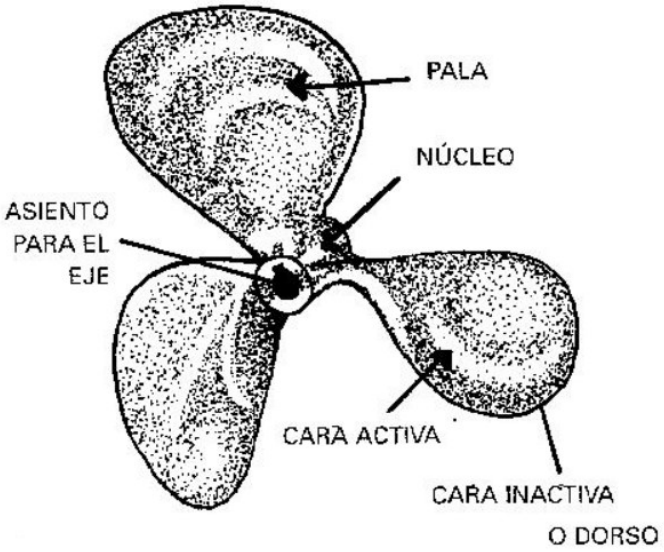


Figura 34. Hélice de 3 palas.

La hélice genera una fuerza a través de la combinación de la sobrepresión generada por la cara de presión y la depresión generada en la cara de succión, como se muestra en la *Figura 35*. Ambas se describen mediante una curva de presiones cuyas características dependen de las formas de la pala, la relación entre el área expandida y el área del disco, así como otras características individuales de cada pala.

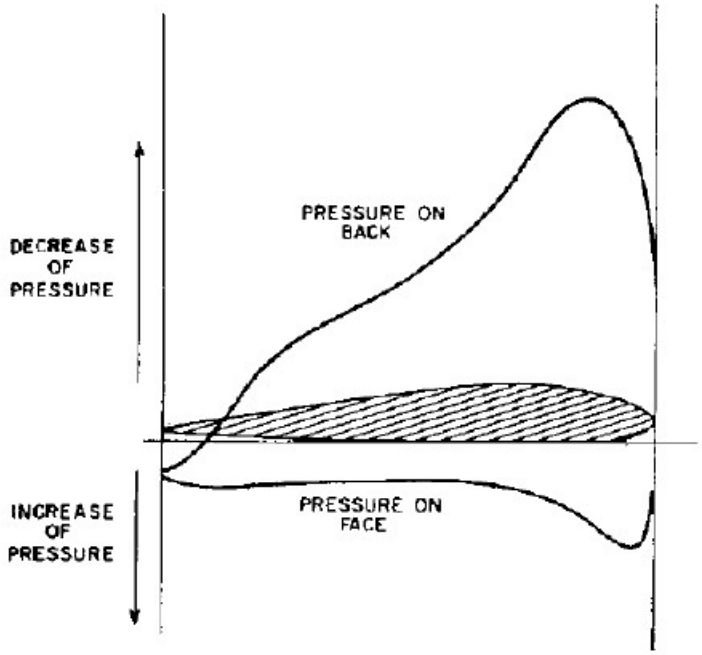


Figura 35. Curvas de presión de una pala en caras de succión y presión.

El área entre estas curvas se puede integrar para calcular la presión generada y, por lo tanto, la presión total que cada pala experimenta es ahí donde entra en juego la integración numérica como método para aproximar estas áreas, ya que su cálculo exacto se torna sumamente complicado.

Las curvas pueden ser representadas mediante la función que corresponde a cada una de ellas. Otra posibilidad es que los datos de entrada sean los puntos de una malla que cubre la región entre las curvas.

La integración numérica tiene aún una mayor relevancia y aplicación en este campo ya que es aplicable computacionalmente e introduciendo el número adecuado de intervalos, con computadores muy potentes, puedes alcanzar la precisión deseada.

En el siguiente punto, vamos a realizar el cálculo de las aproximaciones de algunas de las magnitudes vistas anteriormente mediante programación en lenguaje Matlab.

5.2. APROXIMACIÓN DE VARIABLES HIDROSTÁTICAS MEDIANTE INTEGRACIÓN NUMÉRICA EN MATLAB

En este capítulo calcularemos algunas de las variables hidrostáticas expuestas anteriormente utilizando los mismos códigos que hemos desarrollado en los *puntos 2.1.4., 2.2.4. y 2.3.4.*

Y, tras calcular las variables hidrostáticas mediante los diferentes métodos, en el siguiente epígrafe valoraremos los resultados obtenidos.

5.2.1. Aproximación de variables hidrostáticas aplicando la Regla de los Trapecios en lenguaje Matlab

Primero calcularemos las curvas mediante la Regla de los Trapecios, podemos utilizar cualquiera de los 4 programas en lenguaje Matlab que hemos codificado y presentado en el *punto 2.1.4.* para tal fin.

5.2.1.1. Aproximación del Área de Flotación mediante la Regla de los Trapecios

Utilicemos uno de los programas en lenguaje Matlab que hemos expuesto en el *punto 2.1.4.* para calcular el área de flotación de un buque, en este caso hemos elegido el programa "*Trapecios_P1_Mab".*

Cómo hemos visto en el epígrafe anterior, la fórmula a aproximar es:

$$A_f = \int_{L_f} y\, dx,$$

siendo L_f la eslora de la flotación elegida e y las semimangas.

El buque elegido tiene una eslora $L = 130\ m$, esta eslora será nuestro intervalo de integración. Los valores de las semimangas vienen reflejados en la siguiente tabla:

Nodo	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
Semimanga (m)	0	8,9	11,8	15,5	15,7	15,7	15,7	13,6	9,6	5,8	0

Introducimos estos datos en el programa desarrollado en el *punto 2.1.4.1.* como vemos a continuación:

$$[int] = Trapecios\_P1\_Mab([0\ \ 8.9\ \ 11.8\ \ 15.5\ \ 15.7\ \ 15.7\ \ 15.7\ \ 13.6\ \ 9.6\ \ 5.80], 100, 0, 130)$$

dando como resultado:

$$[int] = 1459.9\ m^2, \qquad \text{(resultado calculado para las semimangas)}$$

Este resultado habrá que multiplicarlo por 2 para obtener el Área de Flotación completa:

$$Área\ de\ Flotación\ total = 2919.8\ m^2.$$

5.2.1.2. Aproximación de la altura del centro de carena (*c.d.c.*) mediante la Regla de los Trapecios

Para este ejemplo tenemos un buque de eslora $L = 52{,}50\ m$ y que tiene los siguientes valores extraídos de la siguiente tabla en la que se relacionan las diferentes secciones de éste con su área de la sección y su momento estático.

Sec.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
As(m^2)	0	5,4	9,9	15,7	19,8	23,4	27,9	28,8	29,7	30,6	32,4	32,4	30,6	28,3	24,3	18,9	10,3	0
mk(m^3)	0	14,6	20,1	27,4	32,8	36,5	41,9	47,4	51,1	54,7	62,1	62,1	58,4	56,5	47,5	45,6	29,2	0

Por definición de *c.d.c.*

$$KB = \frac{M_k}{\nabla},$$

siendo M_k el momento estático del volumen de carena (∇) respecto al plano base con lo que:

$$M_k = \int_0^L A_s \cdot z \cdot dx, \qquad \text{y} \qquad \nabla = \int_0^L A_s \cdot dx.$$

Introduciendo los datos de la tabla en nuestro programa, en primer lugar, calcularemos ∇:

$[int]$
$= Trapecios\_P2\_hab([0\ 5.4\ 9.9\ 15.7\ 19.8\ 23.4\ 27.9\ 28.8\ 29.7\ 30.6\ 32.4\ 32.4\ 30.6\ 28.3\ 24.3\ 18.9\ 10.3\ 0], 17, 0, 52.5)$

$$[int] = 1.1377e + 03\ m^3.$$

A continuación, calculamos el momento estático del volumen de carena M_k:

$[int]$
$= Trapecios\_P2\_hab([0\ 14.6\ 20.1\ 27.4\ 32.8\ 36.5\ 41.9\ 47.4\ 51.4\ 54.7\ 62.1\ 62.1\ 58.4\ 56.5\ 47.5\ 45.6\ 29.2\ 0], 17, 0, 52.5)$

$$[int] = 2.1253e + 03\ m^4.$$

Siendo finalmente la altura del centro de carena (*c.d.c.*):

$$KB = \frac{M_k}{\nabla} = \frac{2125.3}{1137.7} = 1.868\ m\,.$$

5.2.2. Aproximación de curvas hidrostáticas aplicando la Regla de Simpson en lenguaje Matlab

Veamos ahora algunos de los programas desarrollados en el *punto 2.2.4.* del presente proyecto aplicados al cálculo de dos variables hidrostáticas, para el resto de las variables habría que seguir el mismo procedimiento.

5.2.2.1. Aproximación del Área de Flotación mediante la Regla de Simpson

Hemos utilizado uno de los programas en lenguaje Matlab que hemos expuesto en el *punto 2.2.4.* para calcular el área de flotación de un buque. En este caso hemos elegido el programa "*Simpson_P1_Mab*".

Cómo hemos visto en el epígrafe anterior, la fórmula a aproximar es:

$$Af = \int_{Lf} y\, dx,$$

siendo *y* las semimangas.

El buque elegido tiene una eslora $L = 130\ m$, esta eslora será nuestro intervalo de integración. Los valores de las semimangas vienen reflejados en la siguiente tabla:

Nodo	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
Semimanga (m)	0	8,9	11,8	15,5	15,7	15,7	15,7	13,6	9,6	5,8	0

Introducimos estos datos en el programa desarrollado en el *punto 2.1.4.1.* como vemos a continuación:

$$[int] = Simpson\_P1\_Mab([0\ 8.9\ 11.8\ 15.5\ 15.7\ 15.7\ 15.7\ 13.6\ 9.6\ 5.80],100,0,130)$$

dando como resultado

$$[int] = 1.478.1\ m^2, \qquad \text{(resultado calculado para las semimangas)}$$

Este resultado habrá que multiplicarlo por 2 para obtener el Área de Flotación completa:

$$Área\ de\ Flotación\ total = 2.956{,}2\ m^2.$$

5.2.2.2. Aproximación de la altura del centro de carena (*c.d.c.*) mediante la Regla de Simpson

Elegimos el mismo buque para comprar resultados, un buque de eslora $L = 52.50\,m$ y que tiene los siguientes valores extraídos de la siguiente tabla en la que se relacionan las diferentes secciones de éste con su área de la sección y su momento estático.

Sec.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
As(m^2)	0	5,4	9,9	15,7	19,8	23,4	27,9	28,8	29,7	30,6	32,4	32,4	30,6	28,3	24,3	18,9	10,3	0
mk(m^3)	0	14,6	20,1	27,4	32,8	36,5	41,9	47,4	51,1	54,7	62,1	62,1	58,4	56,5	47,5	45,6	29,2	0

Por definición de *c.d.c.*

$$KB = \frac{M_k}{\nabla},$$

siendo M_k el momento estático del volumen de carena (∇) respecto al plano base con lo que:

$$M_k = \int_0^L A_s \cdot z \cdot dx, \qquad \text{y} \qquad \nabla = \int_0^L A_s \cdot dx.$$

Introduciendo los datos de la tabla en nuestro programa, en primer lugar, calcularemos ∇:

$[int]$
$= Simpson\_P2\_hab([0\ 5.4\ 9.9\ 15.7\ 19.8\ 23.4\ 27.9\ 28.8\ 29.7\ 30.6\ 32.4\ 32.4\ 30.6\ 28.3\ 24.3\ 18.9\ 10.3\ 0], 17, 0, 52.5)$

$$[int] = 1.1151e + 03\ m^3.$$

A continuación, calculamos el momento estático del volumen de carena M_k:

$[int]$
$= Simpson\_P2\_hab([0\ 14.6\ 20.1\ 27.4\ 32.8\ 36.5\ 41.9\ 47.4\ 51.4\ 54.7\ 62.1\ 62.1\ 58.4\ 56.5\ 47.5\ 45.6\ 29.2\ 0], 17, 0, 52.5)$

$$[int] = 2.0666e + 03\ m^4.$$

Siendo finalmente la altura del centro de carena (*c.d.c.*):

$$KB = \frac{M_k}{\nabla} = \frac{2066.6}{1115.1} = 1.853\ m\,.$$

5.2.3. Aproximación de curvas hidrostáticas aplicando el método de Newton Cotes cerrado basado en 6 intervalos en lenguaje Matlab

5.2.3.1. Aproximación del Área de Flotación mediante el método de Newton Cotes cerrado basado en 6 intervalos.

Aquí vamos a utilizar uno de los programas en lenguaje Matlab que hemos expuesto en el *punto 2.1.4.* para calcular el área de flotación de un buque.

Cómo hemos visto en el epígrafe anterior, la fórmula a aproximar es:

$$Af = \int_{Lf} y\, dx,$$

siendo *y* las semimangas.

El buque elegido tiene una eslora $L = 130\ m$, esta eslora será nuestro intervalo de integración. Los valores de las semimangas vienen reflejados en la siguiente tabla:

Nodo	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
Semimanga (m)	0	8,9	11,8	15,5	15,7	15,7	15,7	13,6	9,6	5,8	0

Introducimos estos datos en el programa desarrollado en el *punto 2.1.4.1.* como vemos a continuación:

$$[int] = Ncotes6\_P1\_Mab([0\ \ 8.9\ \ 11.8\ \ 15.5\ \ 15.7\ \ 15.7\ \ 15.7\ \ 13.6\ \ 9.6\ \ 5.80],100,0,130)$$

dando como resultado

$$[int] = \ 1084.4\ m^2, \qquad \text{(resultado calculado para las semimangas)}$$

Este resultado habrá que multiplicarlo por 2 para obtener el Área de Flotación completa:

$$Área\ de\ Flotación\ total = 2.168{,}8\ m^2.$$

5.2.3.2. Aproximación de la altura del centro de carena (*c.d.c.*) mediante método de Newton Cotes cerrado basado en 6 intervalos.

Un buque de L=52,50m y que tiene los siguientes valores extraídos de la siguiente tabla en la que se relacionan las diferentes secciones de éste con su área de la sección y su momento estático.

Sec.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
As(m^2)	0	5,4	9,9	15,7	19,8	23,4	27,9	28,8	29,7	30,6	32,4	32,4	30,6	28,3	24,3	18,9	10,3	0
mk(m^3)	0	14,6	20,1	27,4	32,8	36,5	41,9	47,4	51,1	54,7	62,1	62,1	58,4	56,5	47,5	45,6	29,2	0

Por definición de *c.d.c.*

$$KB = \frac{M_k}{\nabla},$$

siendo M_k el momento estático del volumen de carena (∇) respecto al plano base con lo que:

$$M_k = \int_0^L A_s \cdot z \cdot dx, \qquad \text{y} \qquad \nabla = \int_0^L A_s \cdot dx.$$

Introduciendo los datos de la tabla en nuestro programa, en primer lugar, calcularemos ∇:

$$[int] = Ncotes6\_P2\_hab([0\ 5.4\ 9.9\ 15.7\ 19.8\ 23.4\ 27.9\ 28.8\ 29.7\ 30.6\ 32.4\ 32.4\ 30.6\ 28.3\ 24.3\ 18.9\ 10.3\ 0], 17, 0, 52.5)$$

$$[int] = 811.7647\ m^3.$$

A continuación, calculamos el momento estático del volumen de carena M_k:

$$[int] = Simpson\_P2\_hab([0\ 14.6\ 20.1\ 27.4\ 32.8\ 36.5\ 41.9\ 47.4\ 51.4\ 54.7\ 62.1\ 62.1\ 58.4\ 56.5\ 47.5\ 45.6\ 29.2\ 0], 17, 0, 52.5)$$

$$[int] = 1.4327e + 03\ m^4.$$

Siendo finalmente la altura del centro de carena (*c.d.c.*):

$$KB = \frac{M_k}{\nabla} = \frac{1432.7}{811.765} = 1.765\ m\ .$$

5.2.4. Ejemplo de aplicación de los métodos numéricos a un problema de ingeniería naval

Queremos calcular la fuerza que sufre la tapa de una escotilla de un submarino que está sumergido a 100m de profundidad en el mar representada en la *Figura 36*. Para ello vamos a utilizar los programas en código Matlab recogidos en los *puntos 4.3.1 y 4.3.2.* referentes a los métodos de Simpson y Trapecios para varias variables.

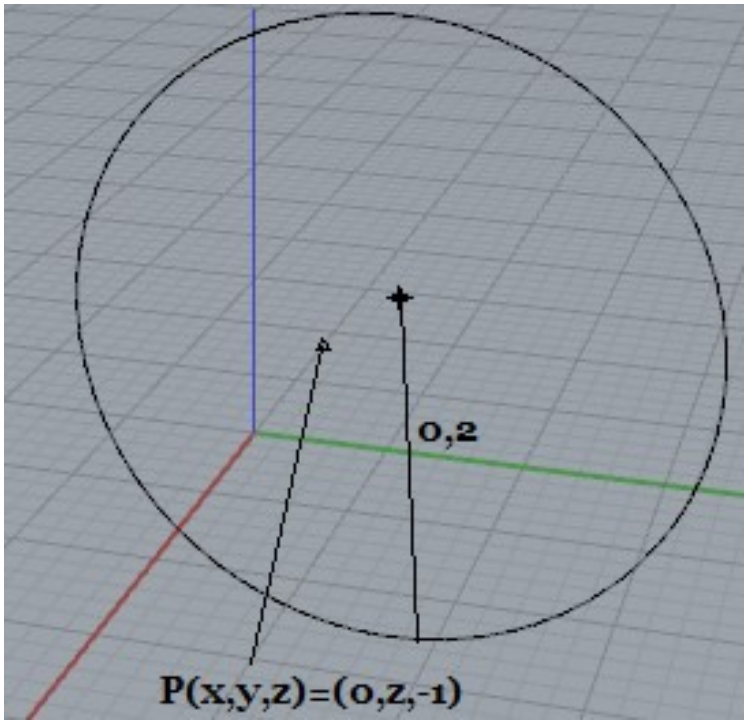


Figura 36. Tapa de la escotilla de un submarino.

La integrar a aproximar para calcular la fuerza que sufre la escotilla es:

$$\iint_S P \ dS \ ,$$

siendo la superficie de la tapa de la escotilla

$$x^2 + (z + 100)^2 \leq 0.04$$

Y la parametrización que queremos realizar

$$P(x, y, z) = (0, -\sqrt{|z|^3}, -1)$$

El cambio de variable a coordenadas polares queda

$$\Phi(\rho, \theta) = (x, y, z)$$

$$x = \rho \cos(\theta)$$
$$y = 0$$
$$z = -100 + \rho \sin(\theta)$$

siendo los intervalos de integración:

$$\rho \in [0, 0.2]$$
$$\theta \in [0, 2\pi]$$

y siendo los vectores tangentes

$$\Phi_\rho = (\cos(\theta), 0, \sin(\theta))$$
$$\Phi_\theta = (-\rho \sin(\theta), 0, \rho \cos(\theta))$$

el vector normal quedaría de la siguiente forma

$$\Phi_\rho \times \Phi_\theta = \begin{vmatrix} i & j & k \\ \cos(\theta) & 0 & \sin(\theta) \\ -\rho\sin(\theta) & 0 & \rho\cos(\theta) \end{vmatrix} = (0, -\rho, 0)$$

siendo este un vector normal interior.

$$\iint_S P\, dS = \int_0^{0,2} \int_0^{2\pi} < \left(0, \sqrt{(100 - \rho \sin(\theta))^3}, -1\right), (0, -\rho, 0) > d\theta\, d\rho$$
$$= \int_0^{0,2} \int_0^{2\pi} \rho\sqrt{(100 - \rho\sin(\theta))^3}\; d\theta\, d\rho$$

Y es aquí cuando entra en juego nuestro programa codificado en lenguaje Matlab para realizar el cálculo de la integral doble final para conocer la fuerza a la que está sometida nuestra tapa de escotilla. De este modo:

1. Solucionando la integral con el código del punto 4.3.1. referente a la Regla de los Trapecios para varias variables:

$$[int] = double(Trapecios\_P1\_2var(x1 * sqrt((100 - x1 * sin(x2))\text{^}3),0,0.2,0,2 * pi, 100,100))$$

$$[int] = 3.923156791802257e + 03\ Newton$$

2. Solucionando la integral con el código del punto 4.3.1. referente a la Regla de Simpson para varias variables:

$$[int] = double(Simpson\_P1\_2var(x1 * sqrt((100 - x1 * sin(x2))\text{^}3),0,0.2,0,2 * pi, 100,100))$$

$$[int] = 3.923158013908437e + 03\ Newton$$

Podemos apreciar que difieren en el sexto decimal por lo tanto las dos aproximaciones devuelven resultados considerablemente similares.

5.3. VALORACIÓN DE LOS RESULTADOS

Hemos podido comprobar en la comparación de los ejemplos de aplicación naval, tanto con las variables hidrostáticas como en el problema de la tapa de la escotilla que la exactitud de la aproximación aumenta de grado conforme más complejo es el método utilizado y más subintervalos contiene el mismo, como hemos comentado a lo largo del proyecto.

Exponemos a continuación, siempre partiendo de la premisa anterior algunas de las ventajas y desventajas que tienen los métodos analizados durante el presente proyecto.

Regla del Trapecio: Este método aproxima el área bajo la curva como una serie de trapezoides. Es un método simple y fácil de entender, pero puede ser inexacto si la función varía mucho. La regla del trapecio es generalmente más precisa cuando los datos son uniformemente espaciados y la función es aproximadamente lineal.

Regla de Simpson: Este es un método más sofisticado que aproxima la integral utilizando polinomios de segundo grado. La regla de Simpson es generalmente más precisa que la regla del trapecio para el mismo número de puntos, especialmente si la función es suave y los datos están uniformemente espaciados. Sin embargo, puede ser inexacto si la función tiene muchas curvas pronunciadas.

Método de Newton Cotes cerrado: Es importante destacar que la Regla de Boole, y en general los métodos de Newton-Cotes, proporcionan resultados bastante precisos si la función es suave y bien modelada por polinomios de bajo grado en el intervalo de integración. Sin embargo, si la función tiene comportamientos más complejos (oscilaciones rápidas, discontinuidades, etc.), la precisión de estos métodos puede disminuir significativamente.

Cuadratura de Gauss: Este método selecciona los puntos de manera óptima en lugar de utilizar puntos uniformemente espaciados como en los métodos anteriores. La cuadratura de Gauss es muy precisa, especialmente para polinomios, pero es más complicada de implementar y puede ser inexacta si la función tiene discontinuidades o comportamiento extremo en los extremos del intervalo.

En resumen, no existe un "mejor" método de integración numérica para todas las situaciones. La elección del método depende de las características de la función a integrar y de las limitaciones computacionales. Aunque si que, dependiendo de la exactitud que queramos conseguir en la aproximación vemos que, a mayor número de subintervalos, mayor exactitud en la aproximación calculada.

6. CONCLUSIONES

Tras analizar en detalle los resultados obtenidos y la aplicación de los métodos numéricos en la ingeniería naval, con un enfoque particular en los módulos de integración numérica, es posible concluir que este proyecto de fin de grado ha sido exitoso y ha aportado importantes contribuciones al campo.

Durante el desarrollo de este proyecto, se ha demostrado que los métodos numéricos son herramientas fundamentales para resolver problemas complejos en la ingeniería naval, permitiendo obtener soluciones aproximadas con precisión adecuada en situaciones donde las soluciones analíticas no son viables o requerirían un esfuerzo computacional considerable. Los módulos de integración numérica, en particular, han mostrado ser esenciales para resolver ecuaciones diferenciales y calcular áreas, volúmenes y centroides en aplicaciones navales específicas.

A través de la revisión bibliográfica y el análisis de casos de estudio, se ha confirmado la eficacia y versatilidad de diversas técnicas numéricas como el método de Simpson, el método de los Trapecio, el método de Newton Cotes basado en 6 intervalos y otros métodos de cuadratura, que se han aplicado exitosamente en la resolución de problemas prácticos en la ingeniería naval. Estas técnicas han demostrado su capacidad para reducir la complejidad de los cálculos, ahorrar tiempo y recursos, y proporcionar resultados confiables.

Además, se ha evidenciado que el uso de herramientas computacionales y software especializado ha sido crucial para la implementación y automatización de los métodos numéricos en el ámbito de la ingeniería naval. La programación y el desarrollo de algoritmos específicos han permitido aprovechar al máximo el potencial de estos métodos y adaptarlos a situaciones navales específicas, mejorando así la precisión y eficiencia de los resultados obtenidos.

Por otro lado, se ha comprobado que una comprensión sólida de los principios matemáticos subyacentes es esencial para la correcta aplicación y selección de los métodos numéricos adecuados en cada escenario de ingeniería naval. La elección de una técnica inapropiada puede llevar a errores significativos y afectar la validez de los resultados.

En resumen, este proyecto de fin de grado ha demostrado que la aplicación de métodos numéricos en la ingeniería naval, con un enfoque en los módulos de integración numérica, es una estrategia valiosa y eficiente para abordar problemas complejos y mejorar el diseño, análisis y optimización de estructuras y sistemas navales. La implementación adecuada de estas técnicas proporciona resultados precisos y confiables, lo que se traduce en un importante avance para la industria naval y su desarrollo tecnológico.

7. BIBLIOGRAFÍA

[1] Becerra, B. (2008). Integración numérica. Algunas reglas básicas.

[2] Seminario, R. (s.f.). Métodos numéricos para ingeniería. ING. Ricardo Seminario Vasquez.

[3] Apuntes de Hidrodinámica y propulsión Universidad Politécnica de Cartagena 4º curso Domingo García López.

[4] Métodos Numéricos Integración numérica. Luis E. Brito.

[5] Métodos numéricos aplicados a la ingeniería naval. Énfasis en los módulos de integración numérica y ecuaciones diferenciales. Trabajo de fin de grado Carlos Tapia Marfil.

[6] P. Puig Adam, Curso teórico práctico de cálculo integral aplicado a la física y la técnica, Biblioteca Matemática, Madrid, 1973.

[7] F. Granero, Cálculo integral y aplicaciones, Prentice Hall, 2001.