Proyecto Fin de Grado

Splines Cúbicos Interpolantes a trozos en Ingeniería Naval

Alumno: ***Víctor Mayoral Alba***

Directores: ***Juan Carlos Trillo Moya***
Juan Ruiz

DEPARTAMENTO DE MATEMÁTICA APLICADA Y ESTADÍSTICA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA NAVAL Y OCEÁNICA

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

INDICE

Objetivos……**5**

Capítulo 1 – Introducción al mundo Naval y las Formas del Buque……**6**

1.1 – Conceptos Básicos……**7**

1.2 – Dimensiones Principales……**8**

Capítulo 2 – Introducción a la Interpolación……**10**

2.1– Conceptos Básicos……**11**

2.2 – Tipos de Interpolaciones……**14**

2.3 – Interpolación en el ámbito naval……**15**

Capítulo 3 – Métodos de Interpolación……**18**

3.1 – Método de Interpolación de Lagrange……**20**

3.2 – Algoritmo de Neville……**21**

3.3 – Polinomio de Newton……**22**

3.4 – Métodos Gráficos……**24**

3.4.1- Métodos de Alambre……**25**

3.4.2 – Métodos de Superficie……**26**

Capítulo 4 – Operadores de Reconstrucción y Subdivisión……**28**

4.1 – Introducción a los operadores de Reconstrucción y Subdivisión……**29**

4.1.1 – Operadores de Decimación y Predicción……**30**

4.1.2 – Operadores de Discretización y Reconstrucción……**30**

4.2 – Relación entre distintos Operadores……**31**

4.3 – Multirresolución de Harten……**32**

4.4 – Operadores de Reconstrucción en una dimensión……..**33**

4.4.1 – Lagrange……..**33**

4.4.2 – PPH……**35**

4.5 – Operadores de Reconstrucción en dos dimensiones. ……**38**

4.6 – Aplicaciones en el ámbito naval……..**38**

Capítulo 5 – SPLINES……**40**

5.1 – Definición de SPLINE……**42**

5.2 – Aplicación de los SPLINES……**43**

5.3 – SPLINES suavizantes y la analogía del muelle en el ámbito naval……..**44**

Capítulo 6 – SPLINES CUBICOS……**50**

6.1 – SPLINES Cúbicos Suavizantes……**51**

6.1.1 – Definición……..**52**

6.1.2 – Condiciones de Contorno……..**53**

6.2 – SPLINES Interpolantes en discontinuidades……**54**

6.2.1 – Introducción……..**55**

6.2.2 – Discretización del entorno de datos en los puntos Característicos y en la dimensión de la celda……..**56**

6.2.3 – Discretización de los datos de ajuste en puntos característicos……**57**

6.3 – Construcción de SPLINES Cúbicos adaptados a discontinuidades……**58**

6.3.1 – SPLINES Cúbicos Clásicos……**58**

6.3.2 – SPLINES Cúbicos Adaptados……**60**

6.4 – Detección de la posición de las discontinuidades……**61**

6.5 – Cálculos en la posición de las discontinuidades……**63**

6.5.1 – Obtención de los valores de f(x) y sus derivadas para discontinuidades en puntos característicos……..**63**

6.5.2 – Obtención de los valores de f(x) y sus derivadas para - discontinuidades en intervalos.....**66**
6.6 – Orden de aproximación cerca de las discontinuidades.....**69**
6.7 – Obtención de los valores F(x) y sus derivadas para discontinuidades en espaciados no uniformes.....**73**
6.8 – Finalidad del uso de SPLINES a trozos.....**77**
6.9 – Comprobación del ajuste realizado.....**87**

Capítulo 7 – Programación de SPLINES en MatLab.....**91**

7.1 – SCRIPTS.....**93**
7.2 – Datos de entrada y salida.....**94**
7.3 – Programación necesaria para el Cálculo de SPLINES.....**94**

Capítulo 8 – Interfaz Gráfica.....**110**

8.1 – Creación de la Interfaz Gráfica.....**111**
8.2 – Partes de la Interfaz Gráfica **112**

Capítulo 9 – Caso Práctico.....**131**

9.1 – Introducción.....**132**
9.2 – Casos a Analizar.....**133**
9.3 - Resolución.....**133**

Capítulo 10 – Conclusión.....**141**

Capítulo 11 – Bibliografía.....**142**

Objetivos

El objetivo del presente Proyecto Fin de Grado consistirá en desarrollar y estudiar la función Spline Cúbico Interpolante y más concretamente del uso de los Splines para la detección de las discontinuidades de esquina y la creación de polinomios que unan cada una de las partes de esta, que podrá utilizarse para realizar cálculos sobre la forma del buque o de la costa que vayamos a navegar pudiendo así diseñar buques con mejores características hidrodinámicas o bien calcular rutas por costas de una manera más precisa conociendo las formas de esta.

En este proyecto:

- Se estudiará la necesidad de localizar y unir los dos lados en dicha discontinuidad con la mayor precisión posible o bien precisar las discontinuidades en el diseño naval, se describirá el proceso de cálculo para la detección de curvas cerradas o de discontinuidades.
- Se describirán los procesos de alisado de las formas de una función bi o tridimensional, es decir, de superficies con aristas cerradas o curvas como otros usos de los Splines.
- Se estudiará la función SPLINE para la detección de discontinuidades de esquina.
- Se desarrollará un algoritmo de cálculo para la función SPLINE cúbico para discontinuidades de esquina.
- Este algoritmo será desarrollado en código M o Matlab, herramienta muy potente en el campo de las matemáticas.
- Se desarrollará una interfaz gráfica capaz de ejecutar este algoritmo de cálculo de SPLINES cúbicos para la detección e interpolación en discontinuidades y alisado de formas en general para así poder poner a disposición de cualquier interesado o usuario del programa para su uso.
- Será demostrada la eficiencia del algoritmo desarrollado mediante un caso práctico perteneciente al campo del diseño naval.

Capítulo I
Introducción al mundo naval y las formas del Buque

Capítulo I
Introducción al mundo naval y las formas del Buque

Para iniciar este proyecto, debemos antes hacer una pequeña introducción al mundo naval, para ello, a continuación explicaremos una serie de conceptos relacionados con este como por ejemplo las formas del buque, terminología naval así como la manera en la que se referencia al buque en sus formas.

1.1- Conceptos básicos.

Para iniciar a definir los conceptos básicos de un buque debemos empezar por las distintas partes que tiene el buque:

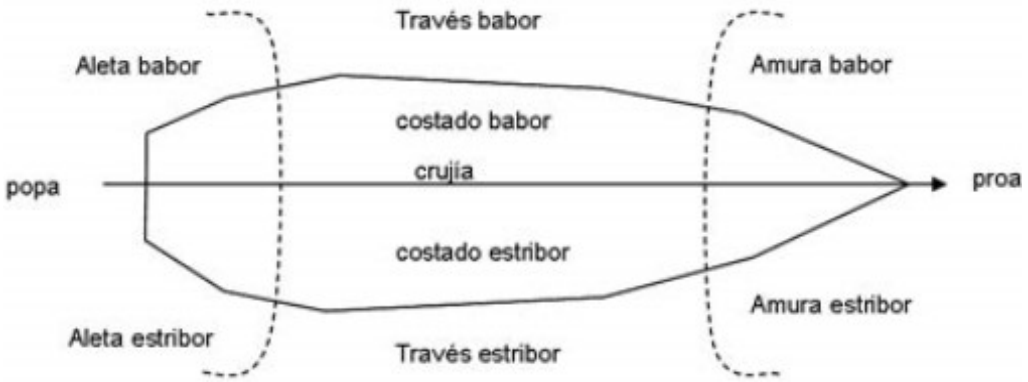


Fig.1 – Partes del Buque

- **Línea de Crujía**: es la más importante línea de referencia del buque, nos indica cual es la línea que pasa por la mitad de este y lo divide en dos partes.
- **Proa:** es la parte delantera del buque, hace referencia a lo que se encuentra por delante de nosotros.
- **Popa:** es la parte trasera del buque, hace referencia a lo que se encuentra por detrás de nosotros.
- **Babor:** una de las partes en las que la línea de crujía divide el buque, mirando desde popa a proa, es la parte izquierda del buque.
- **Estribor:** la otra parte en la que se divide el buque a través de la línea de crujía. Será la parte derecha mirando de popa a proa.
- **Amura:** se denomina así a los costados del buque que se encuentran a proa de la cuaderna central.
- **Aleta:** se denomina así a los costados del buque que se encuentran a popa de la cuaderna central.

También podemos definir distintas zonas si vemos el plano del buque de manera transversal

1.2- Dimensiones principales

Una vez definidos los conceptos básicos para situarnos en el buque podemos definir las diferentes dimensiones de las que se compone el buque:

- **Eslora:** Es la longitud del buque. Es frecuente medir la eslora en pies.

- **Eslora de flotación:** Es la longitud del plano de flotación medida entre proa y popa y es distinta para cada superficie de flotación

- **Eslora total:** Es la longitud total de barco medida entre sus extremos de proa y popa.

- **Manga:** Es la anchura del barco. Como la manga no es constante a lo largo de todo el barco, llamaremos **manga máxima** a la parte con más anchura del barco que normalmente suele coincidir con la cuaderna maestra.

- **Puntal:** Es la altura del buque o distancia vertical en metros medido desde la cara inferior del casco en su intersección con la quilla y la línea de cubierta principal.

- **Calado:** El calado es la altura de la parte sumergida del casco, también lo podemos definir como la medida vertical tomada desde la quilla hasta la línea de flotación.

- **Francobordo:** El francobordo es la distancia entre la línea de flotación y la cubierta estanca más alta. Si aumentamos la carga, disminuiremos el francobordo. Por seguridad está legislado un valor mínimo de francobordo que dependerá de cada barco. Este valor mínimo lo indica la línea de flotación que refleja el estado de máxima carga. La línea de flotación es obligatoria pintarla a los dos lados del barco.

- **Asiento:** Es la diferencia de calados entre proa y popa.

- **Desplazamiento**: Es la masa total del barco. Es también igual al peso del agua desalojada por él, por lo tanto el desplazamiento es también el peso del buque.

Capítulo II
Introducción a la Interpolación

Capítulo II

Introducción a la Interpolación

En primer lugar, para poder entender qué función vamos a realizar con los SPLINES es necesario definir adecuadamente el concepto de SPLINE, y más concretamente, lo que significa el que sea un SPLINE cúbico en lugar de uno normal. Para poder entender qué es un SPLINE, debemos hacer una pequeña introducción al mundo de la interpolación.

En los siguientes apartados trataremos de solucionar las dudas básicas que pudieran surgir en torno al conocimiento y uso de los SPLINES en el mundo matemático.

2.1- Conceptos básicos.

En este apartado describiremos los conceptos necesarios lo más sencillamente posible para facilitar al lector el uso de los SPLINES con una mayor facilidad y fluidez gracias a un conocimiento básico de la materia.

Una de las definiciones que es imprescindible saber para comprender qué es un SPLINE es la de interpolación numérica o de datos. Para definirla debemos dar a conocer

algún término fácilmente reconocible como conjunto de datos o función, que se definen a continuación.

Un **conjunto de datos** relacionados es una serie de valores medidos, tomados o calculados para una serie de condiciones que se necesitan conocer.

Una **función** es una relación establecida entre un conjunto de datos de varias variables, que serán el dominio, qué es la variable independiente de la función, y un conjunto de elementos que irán directamente relacionados con los valores de este, es decir, que serán las variables dependientes de la función. A este último conjunto de datos que resultan de realizar los cálculos de la función se denomina recorrido o imagen de dicha función. Las funciones se pueden representar de manera gráfica como se mostrará a continuación con un sencillo ejemplo.

Un dato importante para que se cumpla una función es que a cada valor del dominio solo se le relacione un resultado de cada conjunto de elementos de la función. Un ejemplo básico de función sería por ejemplo X = Y, cuya representación gráfica sería una recta.

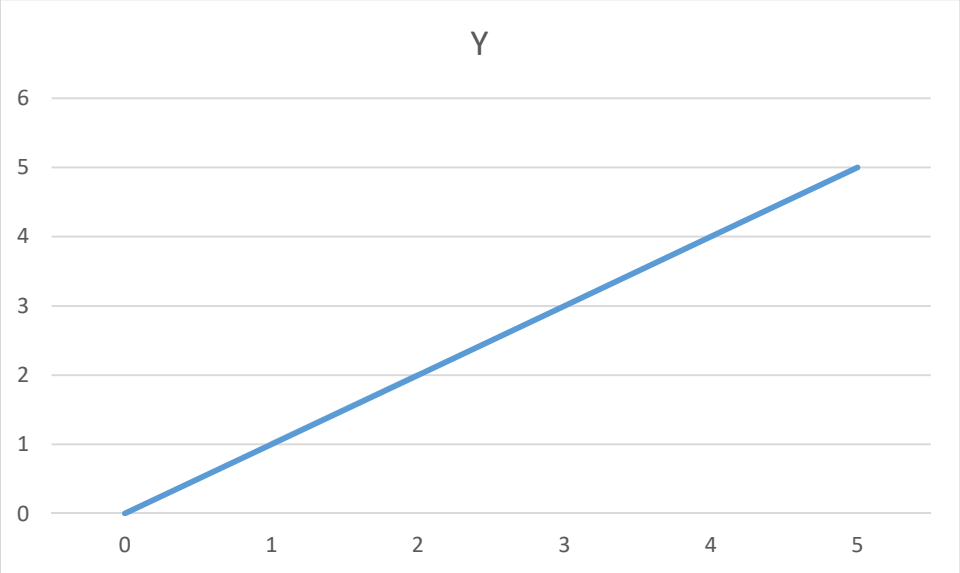


Fig.2 – Función simple

Estas funciones pueden tener dos o tres dimensiones, esto quiere decir que el dominio puede tener una o dos variables que definan nuestra función; en caso de tener una variable será una función 2D y se representará de la forma **f(x)**; sin embargo si el dominio presenta dos variables la función será 3D y se representará de la forma **f(x,y)**.

La interpolación se usa cuando para una serie de datos conocidos nosotros requerimos saber un valor que no pertenece a estos pero si está relacionado con ellos, es decir, que se pueden relacionar por medio de una función. Pues bien, lo que pretendemos con la interpolación es poder estimar f(x) para un x arbitrario, a partir de la construcción de una curva o superficie que une los puntos donde se han realizado las mediciones y cuyo valor si se conoce. Para que se pueda realizar es necesario que dicho valor de x se encuentre dentro de los límites de los puntos de medición. Un ejemplo sencillo de una interpolación sería el siguiente

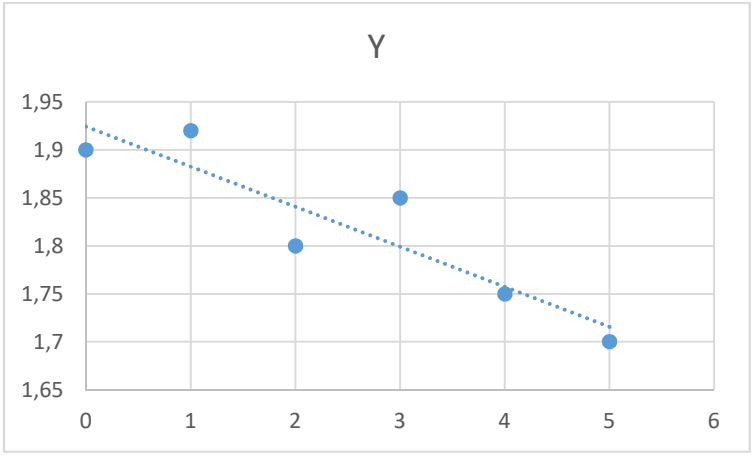


Fig.3 – Función Interpolada a partir de puntos (Interpolación aproximante por Mínimos Cuadrados)

La línea discontinua que se ve en la imagen sería el resultado de la interpolación.

2.2- Tipos de interpolaciones.

Una vez conocida su definición y su uso, tendremos que conocer cuáles son los tipos de interpolaciones para poder comprender el alcance que tienen y cuáles pueden ser sus aplicaciones.

2.2.1- Interpolaciones Lineales

Una interpolación bidimensional es la que se realiza para un par de conjuntos de datos, es decir, para una función de dos variables en la cual uno de ellos, el que es dependiente, presenta una dispersión de puntos que pueden relacionarse con facilidad con una recta.

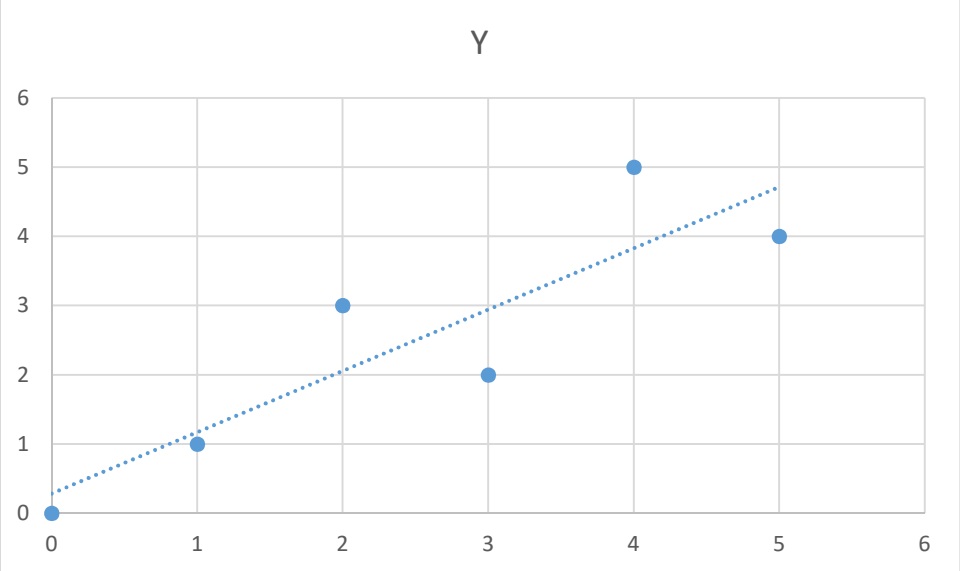


Fig.4 – Interpolación lineal

Este tipo de interpolaciones es el más sencillo y se calcula mediante la siguiente ecuación:

$$x = \frac{y-y_1}{x_2-x_1} \text{X } (y_2 - y_1) + y_1$$

Esta ecuación puede crear series de rectas a través de dos puntos y así poder suponer los valores que podrían darse para la variación de la X entre ambos puntos.

2.2.2- Interpolaciones Polinómicas

Las interpolaciones polinómicas también son llamadas interpolaciones cúbicas, estas se basan en la realización de la interpolación por medio de Polinomios interpolantes. El polinomio interpolante es aquel que pasa por los puntos conocidos para realizar su unión y crear la función. Cada uno de estos polinomios es único para una serie de datos y su grado depende del número de puntos para los cuales se va a crear el polinomio, siendo el grado de este igual o menor al número de datos a analizar. Este tipo de interpolaciones son las más usadas en el mundo naval y generalmente para cálculos matemáticos en el mundo de la ingeniería. La interpolación polinómica se puede realizar de varias maneras, algunas de las cuales definiremos a continuación. El Spline es en sí un tipo de interpolación polinómica.

2.3- Interpolación en el ámbito naval

En el ámbito naval las interpolaciones se pueden usar en diversas ocasiones que podríamos englobar en dos grandes usos generalizados:

2.3.1- Cálculo de valores concretos

Este uso se concentra sobre todo en la consecución de valores que deseamos conocer para realizar nuestros cálculos pero no conocemos con exactitud, es decir, conocemos valores suficientes para aproximar nuestro valor deseado pero éste no es conocido, por lo tanto realizaremos una interpolación de una serie de datos para hallar el valor exacto con el menor error posible para realizar nuestros cálculos con la mayor precisión posible.
Un ejemplo de este uso sería el cálculo de la densidad del agua, ya que las

tablas de densidad van en función de la temperatura, sin embargo las medidas experimentales van de 0.5 en 0.5 grados, por lo que para conocer valores intermedios de esta se realizan interpolaciones lineales entre las temperaturas donde se encuentra el valor que necesitamos conocer y así hallamos un valor aproximado con mucha precisión de la densidad del agua para la temperatura con la que trabajemos sin necesidad de hacer mediciones especiales, ya que la densidad del agua varia poco y el error cometido en la interpolación será muy pequeño.

2.3.2- Definición de rectas o superficies

Estos serán los usos más comunes del dibujo naval, la creación de rectas o superficies a partir de una serie de datos conocidos o bien calculados. La definición de rectas o curvas en el ámbito naval es necesario ya que nos sirven para la creación de las formas del buque, es decir, el alisado de la carena del buque o bien la definición de formas se realiza mediante interpolaciones, incluso se realizan con Splines Cúbicos Suavizantes, otro tipo de Spline que definiremos a continuación. Por otra parte, la definición de superficies por medio de interpolaciones se realiza, por ejemplo, cuando se desea conocer la forma del fondo oceánico que va a afectar a nuestro buque ya que es físicamente imposible tomar todos los valores de dicho fondo por lo que se realiza un muestreo de valores y una vez conseguidos suficientes se realizaran las interpolaciones necesarias para generar un mapa donde se muestren las distintas formas del fondo, es decir, sus profundidades en las distintas zonas.

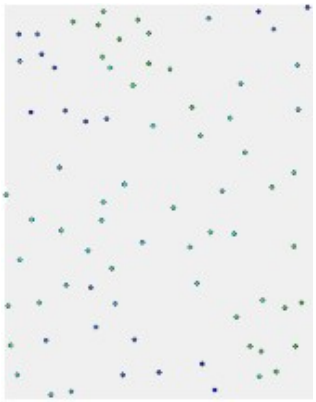


Fig.5 – Mapeo de puntos

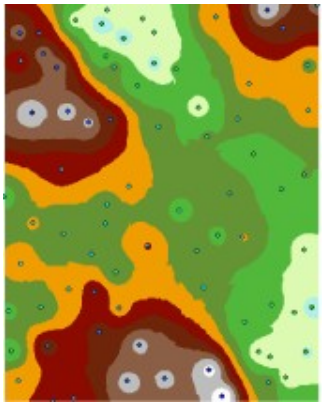


Fig.6 – Interpolación del mapeo

En la *Figura 5* podemos ver una serie de puntos en los que se ha medido la altura del fondo marino, donde los colores más oscuros serán los más profundos y aumentando su altura a la vez que se clarea su color. En la *Figura 6* se ve un mapa en el cual se ha interpolado toda la superficie y se han extendido los colores de profundidad a toda ella.

Capítulo III
Métodos de Interpolación

Capítulo III

Métodos de Interpolación

En el apartado anterior hemos dado una breve explicación de lo que es una interpolación y hemos dado algún ejemplo de para qué podría usarse, sin embargo, las interpolaciones son más complicadas de lo que podría parecer en el primer apartado, por ello hay varios métodos de interpolación que se utilizarán dependiendo de las necesidades tanto de la interpolación como de la forma de los datos, función o funciones a tratar.

Uno de los usos más importantes por el que se han desarrollado nuevos métodos de interpolación es la complejidad de las formas geométricas que se utilizan en el diseño naval y aeronáutico, por ello, en los últimos años se han popularizado los métodos que facilitan los cálculos de estas formas. Esta complejidad de formas hace que a la hora de preparación del material que va a crear esa geometría se creen en él tensiones internas o incluso defectos dentro de éste, por eso interesa que las formas queden alisadas, es decir, que sean formas fácilmente reproducibles en el material, evitando esquinas o incluso discontinuidades que se hayan podido crear en el cálculo de una superficie, a no ser que queramos localizar dicha discontinuidad y reproducirla para conocer los efectos hidrodinámicos que produce.

Los nuevos métodos matemáticos de interpolación se han popularizado debido a que facilita la reproducción de formas complicadas que poseen muchas variaciones en su forma y poseen mayor facilidad para su programación en programas matemáticos que son capaces de generar los gráficos de estas

funciones corregidas por las interpolaciones que se programan.

Sin embargo, conviene hacer una pequeña explicación de los métodos de interpolación para que así tengamos una idea general de cómo se pueden realizar las interpolaciones en la actualidad y así entender el aumento de popularidad de los métodos de interpolación basados en funciones definidas a tramos.

3.1- Método de interpolación de Lagrange.

Este método consiste en la construcción de un polinomio interpolador de grado n que pasa por n+1 puntos (x_i, y_i) de forma que se dé que:

$$P_n(x) = \sum_{i=0}^{n} L_i(x) y_i$$

Donde Pn será el polinomio interpolador. La función Li debe cumplir que $L_i(x_k) = 0$ si i ≠ k y $L_i(x_i) = 1$ ya que así se garantiza que se cumpla que $P_n(x_k) = y_k$.

Por tanto podemos definir el cálculo de la función $L_i$como:

$$L_i(x) = \frac{(x - x_1)(x - x_2) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)}{(x_i - x_1)(x_i - x_2) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)}$$

Finalmente comprobamos que se cumple la condición anterior y finalmente podemos definir el polinomio interpolador de una manera más compacta, es decir, expresándolo todo con una igualdad que será:

$$P_n(x) = \sum_{i=0}^{n} (x_i) \prod_{j=0,\, j \neq i}^{n} \frac{(x - x_i)}{(x_i - x_j)}$$

Este método nos permite una fácil construcción del polinomio interpolador de forma explícita. Se utiliza para interpolaciones lineales o parabólicas.

Para interpolaciones lineales una fórmula genérica sería:

$$P(x) = \frac{(x - x_0)}{(x_1 - x_0)} f_1 + \frac{(x - x_1)}{(x_0 - x_1)} f_0$$

Como vemos esta ecuación es muy similar a la general para interpolaciones lineales que se describió anteriormente en el apartado 1.

Para interpolaciones parabólicas una fórmula genérica sería:

$$P_2(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} f_0 + \frac{(x - x_0)(x - x_2)}{(x_1 - x_2)(x_1 - x_2)} f_1 + \frac{(x - x_0)(x - x_1)}{(x_2 - x_1)(x_2 - x_1)} f_2$$

Con esto vemos que el polinomio de Lagrange es sencillo, por lo que ayuda a entender el concepto de polinomio interpolador pero no se suele utilizar ya que no muestra los coeficientes de cada grado del polinomio, sin embargo se puede utilizar para ciertas demostraciones, una de las más importantes es la formulación del error cometido por el polinomio interpolador en sí.

3.2- Algoritmo de Neville

Este método se usa cuando se dificulta trabajar con el término de error de la interpolación ya que es complicado saber el grado del polinomio necesario para conseguir la precisión que necesitamos para realizar los cálculos. Habitualmente se suele trabajar con polinomios aproximantes y comparando sus diferentes resultados hasta que obtenemos la concordancia deseada. Este algoritmo trata de derivar estos polinomios de aproximación para así facilitar su uso, tanto de polinomios de segundo como de tercer grado. Este método posee la ventaja de que es recursivo, esto implica que podremos obtener información de los polinomios interpolantes de grado mayor partiendo de los polinomios interpolantes de menor grado.

El algoritmo de Neville servirá pues para la creación de aproximaciones dependiendo del grado del polinomio. Su forma será:

$$P_{i,j}(x) = \frac{(x - x_j)P_{0,1,\ldots,j-1,j+1,n}(x) - (x - x_i)P_{0,1,\ldots,i-1,i+1,n}(x)}{x_i - x_j}$$

Este algoritmo analiza los polinomios de Lagrange con facilidad.

3.3- Polinomio de Newton

Este método se usa para interpolaciones que contengan un número de puntos pequeño, ya que el grado del polinomio interpolador será igual al número de puntos menos 1 que entren en la interpolación, siendo así mucho más compleja para un número de puntos alto. Una ventaja sobre un método de interpolación como es el polinomio de Lagrange es que se pueden añadir puntos en la función teniendo que calcular sólo este último punto sin verse afectados los demás.

La forma de este polinomio será la siguiente:

$$P_n(x) = C_0 + C_1(x - x_0) + C_2(x - x_0)(x - x_1) + \cdots + C_n(x - x_0) \ldots (x - x_{n-1})$$

Siendo los coeficientes C_k, k = 0,…, n – 1 determinados mediante las restricciones que se imponen en las ecuaciones $P_n(x_i) = y_i$ e i = 0, …, n.

Estos coeficientes se pueden calcular en tres términos distintos, de los cuales explicaremos el primero ya que lo que queremos hacer es una pequeña introducción a los diferentes tipos de interpolación que se pueden realizar. Los tres

términos serán:

- Diferencias finitas hacia delante
- Diferencias finitas hacia atrás
- Diferencias finitas centradas

Como se ha dicho anteriormente, haremos una breve explicación de las diferencias finitas hacia delante.

- **Diferencias finitas hacia delante**

Consideramos un conjunto de valores (x_i, y_i) , i = 0,…, n. Los valores de y_i se obtienen evaluando la función $f(x_i)$. Pues bien, definiremos las diferencias con la siguiente expresión:

$$\Delta y_i = y_{i+1} - y_i, \qquad i = 0, \dots, n-1$$

Éstas son las llamadas diferencias de primer orden de f(x) sobre el intervalo $(x_{0,} x_n)$

Con esto podemos deducir que las diferencias pueden crecer en orden, como diferencias de éstas, así como diferencias de segundo orden, tercero… hasta un orden k-ésimo.

- 2º Orden: $\Delta^2 y_i = \Delta(\Delta y_i) = \Delta y_{i+1} - \Delta y_i \qquad i = 0, \dots, n-2$
- Orden k-ésimo: $\Delta^k y_i = \Delta^{k-1} y_{i+1} - \Delta^{k-1} y_i \qquad i = 0, \dots, n-k$

Podemos demostrar pues que:

$$\Delta^k y_0 = y_k - \binom{k}{1} y_{k-1} + \binom{k}{2} y_{k-2} - \cdots + (-1)^k y_0$$

El Coeficiente Binomial toma la expresión:

$$\binom{k}{i} = \frac{k!}{i!\,(k-i)!}$$

Para este método se asume una equidistancia entre los diferentes valores de x_i lo que nos indica que $(x_j - x_i) = (j - i)h$ que puede sustituir a las ecuaciones generadas que se dan en el polinomio de Newton mostrado anteriormente (restricciones) para obtener los valores de C_n

Resolviendo una serie de ecuaciones tendremos las siguientes igualdades para C_n

$$c_0 = y_0, \quad c_1 = \frac{\Delta y_0}{h}, \quad c_2 = \frac{\Delta^2 y_0}{2h^2}, \ldots, \quad c_n = \frac{\Delta^n y_0}{n!\,h^n}$$

Siendo h la distancia entre puntos.

Finalmente con estos datos podemos escribir cómo sería realmente el polinomio de Newton que será de la siguiente forma:

$$P_n(x) = y_0 + \frac{\Delta y_0}{h}(x - x_0) + \frac{\Delta^2 y_0}{2h^2}(x - x_0)(x - x_1) + \cdots + C_n(x - x_0)\ldots(x - x_{n-1})$$

3.4- Métodos gráficos.

Los métodos numéricos que se han citado en el apartado anterior, son utilizados de manera más teórica o para el cálculo complejo, sin embargo, otras maneras de interpolación que vamos a exponer brevemente son los Métodos Gráficos, que van a ser comentados debido a que son los de mayor utilización en el ámbito naval para el diseño de embarcaciones, por tanto, no podíamos dejar pasar una explicación de estos aunque no tengan total relación con las interpolaciones matemáticas.

Dentro de estos métodos hay dos tipos fundamentales:

1. *Modelos de alambre*, mediante la definición matemática de líneas que estructuran el buque.
2. *Modelos de superficie*, utilizando la definición de superficies que, mediante "parches", permiten ''cubrir'' toda la carena del barco.

A continuación, vamos a definir cada uno de ellos.

3.4.1- Modelos de alambre

Representan las formas exteriores del buque como un entramado de líneas. En los modelos de alambre, la definición matemática de las líneas curvas utiliza diferentes procedimientos, entre ellos destacan:

-**La simulación del junquillo**. Para utilizar este método es preciso el conocimiento de las ecuaciones de deformación elástica. Las líneas que vamos a obtener tienen que pasar por todos los puntos que se han definido previamente con la ayuda de las ecuaciones nombradas anteriormente. Es necesaria la definición adicional de dos puntos exteriores.

-Bi-arcos. Este método se basa en la utilización de pares de círculos tangentes entre sí que también pasan por todos los puntos que hemos definido. Adicionalmente a dichos puntos han de definirse dos tangentes en los extremos.

-Splines. Utilizan la ecuación de un polinomio de un cierto grado definido y pueden pasar o no por todos los puntos definidos. Si pasan o no por los puntos dependerá del tipo de SPLINE a utilizar.

-Curvas de Bezier. Utilizan la ecuación de Bezier, que obtiene líneas tangentes a polígonos de control, cuyos vértices son los puntos definidos, y sólo pasan por los puntos inicial y final.

-B-splines. Utilizan la ecuación de ‘‘B-Splines’’, que obtiene líneas tangentes a polígonos de control cuyos vértices son los puntos de control. Estas curvas se definen por su grado o su orden así como por el número de vértices. Con el grado se controlará la proximidad de la tangente a la poligonal. Las curvas tampoco pasan por ninguno de los puntos de control, excepto el inicial y el final. Las curvas de Bezier son un caso particular de este tipo de curvas.

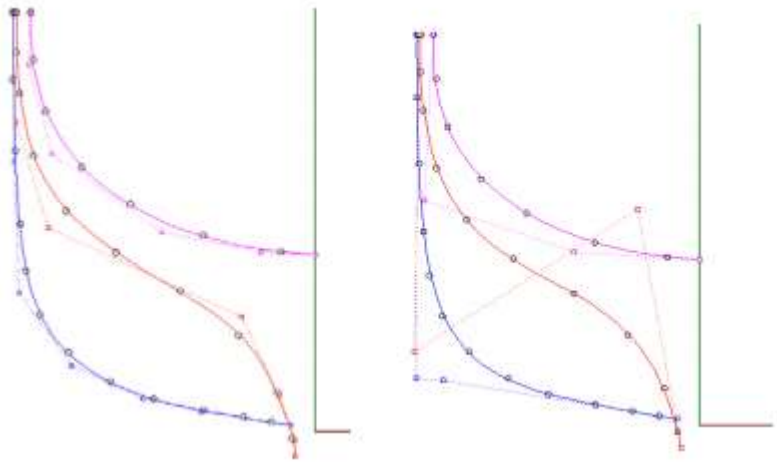


Fig.7 – Métodos de Interpolación Gráfica (Splines, B-Splines)

3.4.2- Modelos de Superficie

En este método se crea y ajusta una *superficie o parche*, la cual debe de pasar por una serie de puntos que definirán la forma del casco del buque o la superficie que se desea crear. Estos puntos iniciales pueden ser definidos mediante muchos tipos de opciones: formas conocidas de la superficie, cálculos realizados, experiencias realizadas… para el caso de un buque, refuerzos

estructurales del casco como pueden ser puntales u otros elementos tales como los mamparos, saltillos o la curvatura de la cubierta. Las superficies que utilizan los programas de diseño actuales son:

-Superficies de COONS

.-Superficies de Bezier.

-B-splines.

-***Superficies NURBS (Non Uniform Rational B-Splines).***

Uno de los principales problemas de estos métodos residirá en la elevada complicación para su manejo ya que requerirá un tiempo de aprendizaje más un tiempo para coger experiencia en el uso de éstos. Además, aunque los ‘‘mallados” tienden a ajustar muy bien los puntos que definen la superficie, el alisado entre mallados resultará a su vez complicado y requerirá de una cantidad elevada de conocimiento en este aspecto para poder así resolver los problemas que se puedan generar, ya que como se ha dicho anteriormente, no es un método sencillo, requiere aprendizaje y experiencia.

Debido a las dificultades que se encuentran en los modelos sencillos es habitual el uso de los modelos mixtos que se basan en la introducción de líneas que nos dan la información necesaria para realizar los mallados además de dichas líneas deben de poder modificarse para que así, a partir de ellas y de los datos que nos aportan, se pueda crear una superficie concreta. Las líneas deben ser móviles ya que es posible que la primera configuración de éstas no nos salga bien la superficie desarrollada.

Capítulo IV

Operadores de Reconstrucción y Subdivisión

Capítulo IV

Operadores de reconstrucción y subdivisión

4.1- Introducción a los operadores de Reconstrucción y Subdivisión.

Los esquemas de subdivisión son unas herramientas muy usadas en el diseño de curvas y superficies, y tienen también relación con otras aplicaciones interesantes en tratamiento digital de imágenes o en la resolución de ecuaciones diferenciales.

Uno de los problemas que se encuentra en el mundo naval, como ya se ha dicho anteriormente, es la necesidad de generar datos nuevos a partir de una serie de datos ya conocidos, o dicho de una manera más teórica, la reconstrucción de una función a partir de un conjunto discreto de datos. Una de las formas más comunes de realizarlo es crear una nueva función que una los distintos puntos para los cuales necesitamos saber los valores intermedios. Es importante que tengamos en cuenta que al usar estos operadores se debe mantener la convexidad y monotonía de la función inicial.

Uno de los usos de los operadores de reconstrucción es servir, como los operadores de discretización, que serán explicados más adelante, para ser los elementos a partir de los cuales definiremos los operadores de decimación y predicción en un esquema de Multirresolución. Ahora bien, debemos explicar qué es cada uno de los términos citados en esta definición del esquema de Multirresolución.

4.1.1- Operador de Decimación y Predicción.

Supongamos que deseamos discretizar una función *f*, cuya discretización será perteneciente a un espacio denominado $\boldsymbol{V^k}$en el cual *k* es el nivel de resolución. El operador de decimación realiza la transición entre distintos niveles de resolución, al igual que el que se definirá a continuación, el operador de predicción.

El operador de decimación, que debe ser lineal y sobreyectivo, proporciona información discreta a un nivel de resolución *k-1* a partir de la información contenida en *k*:

$$\boldsymbol{D_k^{k-1} = V^k \rightarrow V^{k-1}}$$

El operador de predicción, por su parte, realiza la labor contraria, es decir, da una aproximación a la información discreta en el nivel *k* a partir de la información contenida en el nivel *k-1,* este operador no tiene por qué se lineal:

$$\boldsymbol{P_{k-1}^k = V^{k-1} \rightarrow V^k}$$

4.1.2- Operador de discretización y reconstrucción.

El operador de discretización se usa para, como su nombre indica, la discretización de una función *f*, y esta dependerá del tipo de operador utilizado.

Este operador asigna a cada elemento del espacio $f \in \boldsymbol{F}$, donde $\boldsymbol{F}$es un espacio de funciones tal que $\boldsymbol{F} \subset \{\boldsymbol{f} \mid \boldsymbol{f}: \Omega\, \mathbb{R}^{\boldsymbol{m}} \rightarrow \mathbb{R}\}$, una secuencia $\boldsymbol{f^k}$de datos discretos pertenecientes al espacio $\boldsymbol{V^k}$, por lo que podemos definir al operador de discretización como:

$$\boldsymbol{D_k : F \rightarrow V^k}$$

Y éste debe ser lineal, sobreyectivo y se debe cumplir que para $f \in \boldsymbol{F}$ se le asocia:

$$\boldsymbol{f^k = D_k\,(f)}$$

No obstante, el operador de reconstrucción actúa de manera inversa, es decir,

toma una secuencia de datos discretos para reconstruir, a partir de la información proporcionada por dichos datos, la función de la que provienen y se puede definir como:

$$\boldsymbol{R_{k:}: V^k \rightarrow F}$$

No se le exige que sea lineal en el esquema de Multirresolución de Harten.

Sin embargo, por motivos de consistencia, se obliga a ambos operadores a cumplir que:

$$\boldsymbol{D_k R_k f^k = f^k \forall f^k \in V^k}$$

Que también podría expresarse como $\boldsymbol{D_k R_k = I_{V^k}}$, que quiere decir que si tomamos la información reconstruida a partir de unos datos discretos con una cierta resolución y la discretizamos, esta debe coincidir con la original.

4.2- Relación entre los distintos operadores

Para definir la relación existente entre los cuatro tipos de operadores que se han citado anteriormente, podremos usar la siguiente figura:

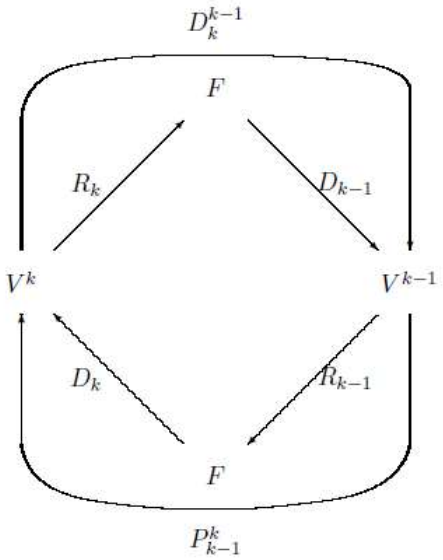


Fig.8 – Relación entre operadores

Esta figura nos indica que, según las relaciones que nos muestra, el operador

de decimación tendrá la siguiente forma:

$$\boldsymbol{D_k^{k-1} = D_{k-1} R_k}$$

Aparentemente el operador de decimación depende únicamente de la elección del operador de reconstrucción, sin embargo, no es así sí y solo sí la sucesión de operadores de discretización es anidada. Esto quiere decir que:

$$\boldsymbol{D_k f = 0 => D_{k-1} f = 0 \; \forall \; f \in F}$$

El anidamiento significa que la información contenida en los datos a un cierto nivel de resolución no será nunca mayor a la información obtenida en un nivel de resolución superior.

De forma muy similar construimos el operador de predicción:

$$\boldsymbol{P_{k-1}^k = D_k \, R_{k-1}}$$

Y a partir de estas expresiones obtenemos la relación de consistencia para los operadores de decimación y predicción

$$\boldsymbol{D_k^{k-1} P_{k-1}^k = D_{k-1} R_k \, D_k \, R_{k-1} = D_{k-1} R_{k-1} = I_{V^{k-1}}}$$

Esta relación nos indica que no nos inventamos información cuando utilizamos estos operadores. Esto quiere decir que si realizamos una decimación de la información obtenida a partir de una predicción realizada sobre una información con una resolución dada se obtiene la misma información de partida sin que se introduzca ningún nuevo elemento.

4.3- Multirresolución de Harten

Finalmente debemos explicar la Multirresolución de Harten, la cual se realiza con los operadores explicados anteriormente y se podría definir como una reordenación multiescala del contenido de un conjunto de datos discretos a una resolución concreta.

El paso fundamental en la construcción de un sistema de Multirresolución es la definición de un operador de reconstrucción adecuado para la discretización que estamos considerando. Normalmente la discretización se realiza de dos maneras, por valores puntuales o por medias en celda.

4.4- Operadores de reconstrucción en una dimensión.

Una vez definidos los operadores y el esquema de Multirresolución, cabe destacar que los operadores de reconstrucción se pueden dividir en dos clases, lineales o no lineales. Para la reconstrucción se trata de conseguir la función más sencilla posible que aproxime a la función original usando los datos de los que disponemos. Para esto podemos utilizar técnicas lineales que son independientes de los datos y son sencillas o bien técnicas no lineales que se adaptan a los datos para proporcionar un mejor resultado, sin embargo son mucho más complicadas que los anteriores. Para hacer una pequeña descripción de cada uno de ellos, tomaremos uno de cada tipo como ejemplo, en el caso de las técnicas lineales usaremos a Lagrange, ya que hemos explicado su método interpolatorio anteriormente, y para las técnicas no lineales usaremos la técnica PPH, que se explicará una vez finalizada la definición de la técnica lineal.

4.4.1- Lagrange

Consideramos los valores de la función $\boldsymbol{f_{j-1}, f_j, f_{j+1}, f_{j+2}}$ que se corresponden con las abcisas $\boldsymbol{x_{j-1}, x_j, x_{j+1}, x_{j+2}}$ de una malla regular X. Con esto, vamos a definir como construir un trozo polinómico del operador de reconstrucción y la predicción $\boldsymbol{\hat{f}_{2j+1}}$ en el punto medio $\frac{x_j+x_{j+1}}{2} = \boldsymbol{x_{j+\frac{1}{2}}}$, por lo que la reconstrucción en el intervalo $[\boldsymbol{x_j + x_{j+1}}]$ vendrá dada por:

$$\boldsymbol{P_j(x) = f_{j-1}L_{-1}(x) + f_jL_0(x) + f_{j+1}L_1(x) + f_{j+2}L_2(x)}$$

Donde $\boldsymbol{L_i(x)}$ será el polinomio de Lagrange el cual ya ha sido explicado en el capítulo anterior. Por tanto, definiremos $\boldsymbol{\hat{f}_{2j+1}}$ como la evaluación $\boldsymbol{x_{j+\frac{1}{2}}}$ del polinomio de Lagrange $\boldsymbol{P_j(x)}$, es decir:

$$\boldsymbol{P_j(x_{j+\frac{1}{2}}) = f_{j-1}L_{-1}\left(x_{j+\frac{1}{2}}\right) + f_jL_0\left(x_{j+\frac{1}{2}}\right) + f_{j+1}L_1\left(x_{j+\frac{1}{2}}\right) + f_{j+2}L_2(x_{j+\frac{1}{2}})}$$

Realizando una serie de cálculos obtendremos que:

$$\boldsymbol{P_j\left(x_{j+\frac{1}{2}}\right) = f_{j-1}(-\frac{1}{16}) + f_j(\frac{9}{16}) + f_{j+1}(\frac{9}{16}) + f_{j+2}(-\frac{1}{16})}$$

Estos valores que se han calculado $\boldsymbol{\left(-\frac{1}{16}\right), \left(\frac{9}{16}\right)}$vienen dados por una interpolación segmentaria centrada de Lagrange y se denominan máscaras del operador de predicción. Con esto tenemos que las máscaras cambiarán en función del grado del polinomio interpolador que usemos, que vendrá dado por el conjunto de puntos "s" con la siguiente expresión: *r = 2s – 1*

Podemos calcular más máscaras de predicción para distintos valores de S. Estos van a ser representados en la siguiente tabla:

	MÁSCARAS
S = 2	$\boldsymbol{\left\{-\frac{1}{16}, \frac{9}{16}, \frac{9}{16}, -\frac{1}{16}\right\}}$
S = 3	$\boldsymbol{\left\{\frac{3}{256}, -\frac{25}{256}, \frac{150}{256}, \frac{150}{256}, -\frac{25}{256}, \frac{3}{256}\right\}}$
S = 4	$\boldsymbol{\left\{-\frac{5}{2048}, \frac{49}{2048}, -\frac{245}{2048}, \frac{1225}{2048}, \frac{1225}{2048}, -\frac{245}{2048}, \frac{49}{2048}, -\frac{5}{2048}\right\}}$

4.4.2- PPH

Este operador posee tres características principales:

1- Cada parte está centrada con un stencil (conjunto de puntos usados para construir el polinomio interpolador) fijo centrado en 2s puntos.
2- Su reconstrucción para las partes suaves es tan exacta como su equivalente lineal.
3- Se reduce la exactitud de las zonas con singularidades, pero no se pierde por completo como en su equivalente lineal.

Ahora bien, tras esta breve introducción, vamos a definir como construir un trozo polinómico de orden 2s para el intervalo $[x_j, x_{j+1}]$. Partiendo de los 2s datos:

$$\{f_{j-s+1}, \cdots, f_{j-3}, f_{j-2}, f_{j-1}, f_j, f_{j+1}, f_{j+2}, f_{j+3}, f_{j+s}\}$$

Dentro de este conjunto vamos a buscar un valor al cual le vamos a realizar una modificación para suavizar la función de manera que luego podamos utilizar la interpolación de Lagrange sobre estos sin discontinuidades apreciables. Esta modificación tiene como objetivo dar una aproximación al valor de la función en el punto medio.

Vamos a definir el siguiente coeficiente, B_{s-1}^{i} mediante la siguiente relación de recurrencia:

$$B_{s-1}^{i} = 2$$

$$B_{s-1}^{q} = \sum_{j=1}^{s-1-q} (-1)^j \left(\binom{2(q+j)}{(j-1)} - \binom{2(q+j)}{j} \right) B_{s-1}^{q+j}$$

Donde $q = s-2,\ldots,1$

En la siguiente tabla mostramos la solución para distintos valores de s:

	$\boldsymbol{B_{s-1}^{i}}$
S = 2	*{2}*
S = 3	*{2, 6}*
S = 4	*{2, 10, 12}*

Para esta reconstrucción también vamos a necesitar las medias *p-power* ($\boldsymbol{power_p(x,y)}$, para cualquier número enterero $\boldsymbol{p \geq 1}$ y cualquier pareja *(x, y)* que vendrán definidas de la siguiente manera:

$$\boldsymbol{power_p(x,y) = \frac{sign(x) + sign(y)}{2} * \frac{|x+y|}{2}(1 - \left|\frac{x-y}{x+y}\right|)}$$

También utilizaremos las diferencias divididas, que actúan como indicadores de zonas de suavidad en la función, que para nodos equidistantes pueden ser calculadas mediante el triángulo de Tartaglia.

Finalmente, y con todo esto, se lleva a cabo una modificación progresiva de los datos diseñada para mantener el orden de interpolación en las regiones convexas suaves en el momento de aplicar la interpolación de Lagrange.

Un ejemplo sería:

- **Modificación de datos de entrada** $\boldsymbol{f \rightarrow f \rightarrow \tilde{f}}$

- <u>PASO 1</u>

  Consideramos $\{\boldsymbol{f_{j-1}, f_j, f_{j+1}, f_{j+2}}\}$

  Si $\left|\Delta_{j-1}^{2}\boldsymbol{f}\right| \leq \left|\Delta_{j}^{2}\boldsymbol{f}\right|$ tendremos

  $$\boldsymbol{\tilde{f}_{j+2} \coloneqq f_{j+1} + f_j - f_{j-1} + B_1^1 pow_{2s-2}(\Delta_{j-1}^{2}f, \Delta_{j}^{2}f)}$$

  Y en otro caso

  $$\boldsymbol{\tilde{f}_{j-1} \coloneqq f_{j+1} + f_j - f_{j+2} + B_1^1 pow_{2s-2}(\Delta_{j-1}^{2}f, \Delta_{j}^{2}f)}$$

Por tanto, definimos:

$$\tilde{\boldsymbol{f}} \coloneqq \begin{cases} (\dots, \boldsymbol{f_{j-2}}, \boldsymbol{f_{j-1}}, \boldsymbol{f_j}, \boldsymbol{f_{j+1}}, \tilde{\boldsymbol{f}}_{\boldsymbol{j+2}}, \boldsymbol{f_{j+3}}, \dots) & \boldsymbol{si}\ \left|\Delta^{\boldsymbol{2}}_{\boldsymbol{j-1}}\boldsymbol{f}\right| \leq \left|\Delta^{\boldsymbol{2}}_{\boldsymbol{j}}\boldsymbol{f}\right| \\ (\dots, \boldsymbol{f_{j-2}}, \boldsymbol{f_{j-1}}, \boldsymbol{f_j}, \boldsymbol{f_{j+1}}, \boldsymbol{f_{j+2}}, \boldsymbol{f_{j+3}}, \dots) & \boldsymbol{para\ otro\ caso} \end{cases}$$

- <u>PASO 2</u>

  Consideramos $\{\boldsymbol{f_{j-2}}, \tilde{\boldsymbol{f}}_{\boldsymbol{j-1}}, \tilde{\boldsymbol{f}}_{\boldsymbol{j}}, \tilde{\boldsymbol{f}}_{\boldsymbol{j+1}}, \tilde{\boldsymbol{f}}_{\boldsymbol{j+2}}, \boldsymbol{f_{j-3}}\}$

  Si $\left|\Delta^{\boldsymbol{4}}_{\boldsymbol{j-2}}\tilde{\boldsymbol{f}}\right| \leq \left|\Delta^{\boldsymbol{4}}_{\boldsymbol{j-1}}\tilde{\boldsymbol{f}}\right|$ tendremos

$$\tilde{\boldsymbol{f}}_{\boldsymbol{j-3}} \coloneqq \boldsymbol{f_{j+1}} + \boldsymbol{f_j} - \boldsymbol{f_{j-2}} + \boldsymbol{B^1_2 pow_{2s-2}}\left(\Delta^{\boldsymbol{2}}_{\boldsymbol{j-1}}\boldsymbol{f}, \Delta^{\boldsymbol{2}}_{\boldsymbol{j}}\boldsymbol{f}\right) + \boldsymbol{B^2_2 pow_{2s-4}}\left(\Delta^{\boldsymbol{4}}_{\boldsymbol{j-2}}\tilde{\boldsymbol{f}}_{\boldsymbol{j}}, \Delta^{\boldsymbol{4}}_{\boldsymbol{j-1}}\tilde{\boldsymbol{f}}\ \right)$$

Y en otro caso

$$\tilde{\boldsymbol{f}}_{\boldsymbol{j-2}} \coloneqq \boldsymbol{f_{j+1}} + \boldsymbol{f_j} - \boldsymbol{f_{j+3}} + \boldsymbol{B^1_2 pow_{2s-2}}\left(\Delta^{\boldsymbol{2}}_{\boldsymbol{j-1}}\boldsymbol{f}, \Delta^{\boldsymbol{2}}_{\boldsymbol{j}}\boldsymbol{f}\right) + \boldsymbol{B^2_2 pow_{2s-4}}\left(\Delta^{\boldsymbol{4}}_{\boldsymbol{j-2}}\tilde{\boldsymbol{f}}_{\boldsymbol{j}}, \Delta^{\boldsymbol{4}}_{\boldsymbol{j-1}}\tilde{\boldsymbol{f}}\ \right)$$

Con esto somos capaces de definir:

$$\boldsymbol{f} = \begin{cases} (\dots, \boldsymbol{f_{j-2}}, \boldsymbol{f_{j-1}}, \boldsymbol{f_j}, \boldsymbol{f_{j+1}}, \tilde{\boldsymbol{f}}_{\boldsymbol{j+2}}, \boldsymbol{f_{j+3}}, \dots) & \textbf{si}\ \left|\Delta^{\boldsymbol{2}}_{\boldsymbol{j-1}}\boldsymbol{f}\right| \leq \left|\Delta^{\boldsymbol{2}}_{\boldsymbol{j}}\boldsymbol{f}\right| \\ (\dots, \boldsymbol{f_{j-2}}, \boldsymbol{f_{j-1}}, \boldsymbol{f_j}, \boldsymbol{f_{j+1}}, \boldsymbol{f_{j+2}}, \boldsymbol{f_{j+3}}, \dots) & \textbf{para otro caso} \end{cases}$$

Y finalmente concluimos que:

$$\boldsymbol{f} = \begin{cases} (\dots, \boldsymbol{f_{j-3}}, \boldsymbol{f_{j-2}}, \tilde{\boldsymbol{f}}_{\boldsymbol{j-1}}, \tilde{\boldsymbol{f}}_{\boldsymbol{j}}, \tilde{\boldsymbol{f}}_{\boldsymbol{j+1}}, \tilde{\boldsymbol{f}}_{\boldsymbol{j+2}}, \tilde{\boldsymbol{f}}_{\boldsymbol{j+3}}, \boldsymbol{f_{j+4}}, \dots) & \textbf{si}\ \left|\Delta^{\boldsymbol{2}}_{\boldsymbol{j-1}}\boldsymbol{f}\right| \leq \left|\Delta^{\boldsymbol{2}}_{\boldsymbol{j}}\boldsymbol{f}\right| \\ (\dots, \boldsymbol{f_{j-3}}, \tilde{\boldsymbol{f}}_{\boldsymbol{j-2}}, \tilde{\boldsymbol{f}}_{\boldsymbol{j-1}}, \tilde{\boldsymbol{f}}_{\boldsymbol{j}}, \tilde{\boldsymbol{f}}_{\boldsymbol{j+1}}, \tilde{\boldsymbol{f}}_{\boldsymbol{j+2}}, \boldsymbol{f_{j+3}}, \boldsymbol{f_{j+4}}, \dots) & \textbf{para otro caso} \end{cases}$$

De este modo es como se realizan los distintos pasos hasta completar la modificación de los puntos.

Podemos comprobar pues que aplicando la interpolación de Lagrange de orden 2s en una función $\tilde{\boldsymbol{f}}$ sobre los datos mostrados en las ecuaciones anteriores obtendremos la interpolación no lineal que estábamos buscando.

El operador de reconstrucción que obtenemos mediante esta técnica de interpolación no lineal posee diversas características muy deseables, que serán:

1- La construcción lineal inicial posee una precisión igual a la reconstrucción realizada en las zonas convexas suaves.
2- Cada pieza polinómica es construida con un stencil de *2s* puntos.
3- La pérdida de precisión en las zonas con singularidades no es total.
4- Las reconstrucciones están libres de oscilaciones de Gibbs.

4.5- Operadores de reconstrucción en dos dimensiones.

Estas reconstrucciones se suele realizar para aumentar la resolución de las distintas series de valores que tenemos disponibles. Esta se puede realizar vía producto tensor (en matriz), primero mediante una reconstrucción por filas y posteriormente volviendo a reconstruir pero esta vez por columnas. Esto hará que crezca el número de datos disponibles, primero en sentido horizontal y posteriormente en sentido vertical, creando un mapa de puntos reconstruidos. Otro método para realizar la reconstrucción es tomando como un trabajo directo de dos dimensiones la zona que vayamos a reconstruir. Se realizará mediante la construcción de una expresión polinómica que no venga de realizar el producto tensor, sin embargo, con este método hablaríamos de reconstrucciones no lineales no separables. Estas presentan una ventaja importante con respecto al otro método, y es que es más sencillo en ellas evitar las discontinuidades ya que es el producto tensor el que dificulta el trabajo en dichas discontinuidades.

4.6- Aplicaciones en el ámbito naval.

En el ámbito naval, estos operadores tienen una gran cantidad de usos, empezando por la recreación de formas o parámetros necesarios para el ámbito naval, tanto para el diseño como para la ingeniería. Uno de los usos que tendremos es la creación de cartas náuticas o mapas de navegación del fondo marino. Estos dos elementos se componen de un gran número de datos los cuales son imposibles de tomar experimentalmente, es decir, es imposible medir toda una línea de costa o bien medir todo el fondo marino que nos interese para nuestro trabajo, para ello, se toma

un muestreo de puntos de la zona que queremos analizar y aplicamos los operadores de subdivisión de 2D necesarios para poder generar puntos nuevos en nuestra red con valores muy precisos en dichas zonas hasta que encontremos el valor que realmente nos interesa para nuestro cálculo. A esto se le suele llamar **ZOOM de datos o esquema de subdivisión.** Esta reconstrucción es sucesiva y se analiza en las zonas centrales donde su precisión es mayor.

Otro uso del Zoom de datos es la creación de diversas reglas y normas que nos permitan el cálculo de diversos parámetros (humedad, viscosidad, corrosividad, velocidades de las corrientes…) para los cuales solo hay una serie de valores tomados ya que tomar todos los casos posibles en todas la zonas posibles es una tarea imposible debido al gran tamaño que poseen los mares y océanos.

Puesto que hay operadores de reconstrucción que aumentan el número de datos, también los habrá que reduzcan la densidad de datos de una zona concreta o en general. Esto se realiza mediante el uso de operadores de decimación.

Por último, para el operador de decimación, el cual sirve para reducir el número de puntos que se encuentran en un mallado, tendrá sus uso para el diseño del buque ya que mediante su uso reduciremos los nodos de control necesarios para crear la curva lo que facilita el diseño de superficies ya que muchos programas no son capaces de crear curvas lisas con una gran cantidad de nodos o bien complican mucho el diseño que vamos a realizar. Esto es así debido a que, para realizar las interpolaciones requeridas para el diseño, se reduce el número de puntos por el cual debe de pasar nuestro Spline suavizante, facilitando el cálculo y reduciendo así el número de puntos necesarios para crear una curva suave.

Capítulo V
Splines

Capítulo V

SPLINES

Como ya se ha descrito en capítulos anteriores, las curvas con las que se trabaja en el diseño naval son bastante complejas, y en la mayoría de los casos no podrán ser expresadas analíticamente mediante una única función. Si nos planteáramos el expresar esas curvas mediante una única expresión analítica, nos daríamos cuenta que eso plantea muchos problemas. Por un lado, la forma de la función polinómica de un grado alto, no suele adaptarse de la manera deseada debido al gran número de extremos e inflexiones. Además, al intentar buscar esta expresión, nos daremos cuenta de que su cálculo es bastante complejo, lo que limita su utilidad en análisis numérico.

Por todas estas razones, es más conveniente dividir el intervalo de interés en subintervalos más pequeños y a la vez usar en estos subintervalos polinomios de un grado relativamente bajo, tratando de que la función a trozos definida de este modo tenga un aspecto final adecuado a las curvas que estamos buscando. De este modo cada fragmento será suave, pero tendremos que prestar mucha atención para que mantengan esa suavidad en las zonas donde se empalman los fragmentos.

Llegados a este punto dentro del subcampo matemático del análisis numérico definiremos el concepto ‘’Spline’’ como una curva diferenciable definida en porciones o trozos mediante polinomios. Los Splines más utilizados son los de grado tres o cúbicos, ya que éstos ofrecen una relación buena entre flexibilidad y velocidad de cálculo; comparando con polinomios de orden superior. Los Splines requieren de menos cálculos y menos memoria, a la vez que son más estables.

5.1- Definición de SPLINE

Supongamos que en un segmento [a,b] está dado un retículo

$$w: a = x_1 < x_2 \dots < x_{m-1} < x_m = b$$

Los puntos x_1 y x_m se llaman nodos fronterizos del retículo w, y los puntos x_2 y x_{m-1} son nodos interiores.

Una función S(x), definida en el segmento [a,b] , se llama SPLINE de orden p+1 (de grado p) si:

- Se da en cada segmento:

$$\Delta_i = [x_i, x_{i+1}] \forall i = 1, \dots, m;$$

Y será un polinomio de grado menor o igual que P todo aquel que cumpla

$$S(x) = S_i(x) = \sum_{k=0}^{P} a_k^{(i)} (x - t_i)^k \; \forall i = 1, \dots, m-1$$

En todo el segmento, el Spline es un polinomio de grado P con P+1 coeficientes. En total se tienen $(m-1)$ segmentos parciales. Consecuentemente, para determinar completamente el Spline es necesario hallar números.

- Es (P-1) veces diferenciable con continuidad en el segmento [a, b]:

$$S(x) \in C^{p-1}[a, b]$$

Esta condición implica la continuidad de la función $S(x)$ y sus derivadas $S(x)', S(x)'' \dots, S^{p-1}(x)$ en todos los nodos *m-2* interiores del retículo *w*. Por tanto, para hallar los coeficientes de todos los polinomios se dispone de *p(m-2)* condiciones (ecuaciones).

Para finalizar con la definición del Spline serán necesarias estas condiciones:

$$(p+1)(m-1) - p(m-2) = m + p + 1$$

5.2- Aplicaciones de los SPLINES.

Para ver las diferentes aplicaciones de los SPLINES, primero debemos saber con qué se usan, es decir, que necesito para poder aplicarlos a un caso concreto.

Proporcionaremos un conjunto de puntos conocidos, que serán una serie de coordenadas, que serán llamados puntos de control. Estos puntos son los que nos van a servir como directrices y son con los que posteriormente ajustaremos nuestras funciones polinómicas continuas. Las maneras de que esto se dé serán las siguientes:

a) La curva realiza **interpolación** del conjunto de puntos de control cuando las secciones polinómicas se ajustan de modo que la curva pasa a través de cada punto de control $(x_i, y_i) \forall\, i = 1, \dots, m$; lo que proporciona m condiciones. Las restantes ecuaciones para la construcción del SPLINE mediante interpolación serán los valores de las derivadas menores del SPLINE en los extremos del segmento analizado: condiciones de contorno o de frontera.

b) La curva realiza una **aproximación o suavizamiento** al conjunto de puntos de control cuando los polinomios se ajustan a la trayectoria general del punto de control *sin pasar necesariamente a través de ningún punto de control* $(x_i, y_i) \forall\, i = 1, \dots, m$. La medida de esta cercanía se puede definir de diferentes maneras, lo que genera una gran variedad de *SPLINES suavizantes.*

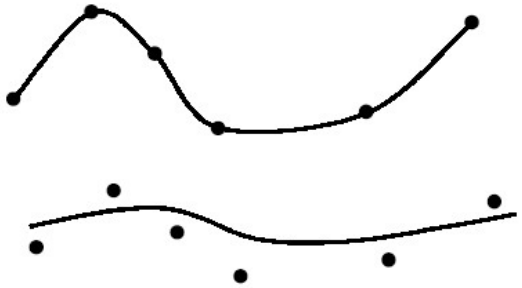


Fig.9 – Usos de SPLINES

En la figura se muestra la comparación entre una curva interpolada y una curva suavizada.

Una curva SPLINE se define y se modifica con operaciones sobre sus puntos de control. Los programas de diseño CAD, además podrán insertar puntos de control adicionales para ayudar al diseñador en el modelado y alisado.

El problema del *suavizamiento* que vamos a plantear consistirá en la obtención de una función suave a partir de sus valores en ciertos puntos. Está claro que este problema tiene infinitas soluciones diferentes. Imponiendo sobre la función condiciones adicionales, se podrá lograr la univocidad.

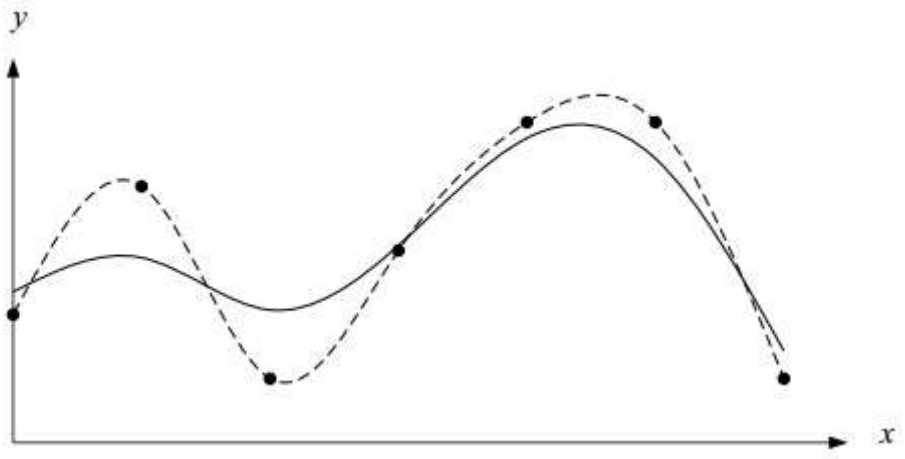


Fig.10: SPLINES interpolantes (discontinuo) vs SPLINES Suavizantes (continuo).

5.3- SPLINES Suavizantes y la analogía del muelle en el ámbito naval.

Hasta ahora se ha demostrado que las curvas suaves son necesarias en el dibujo naval, sin embargo en la construcción naval los SPLINES han sido utilizados desde siglos atrás mediante el uso de los denominados ‘’junquillos’’.

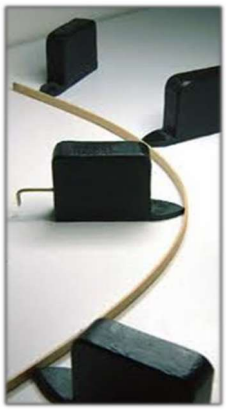


Fig.11 - Junquillo

El junquillo (figura 11), es un listón flexible que se puede utilizar para dibujar curvas suaves en dos dimensiones. Mediante el ajuste de los puntos finales del SPLINE o junquillo y aplicando unos *“*PESOS”, podremos crear una curva suave sobre la que podremos realizar el trazado. Esta curva dependerá básicamente de dos factores concretos:

1- La *rigidez* del material con el que está hecho el junquillo.
2- De los *“*pesos” y los puntos donde estos están aplicados.

Las superficies que compondrán el casco de una embarcación, se definirán por la posición de un conjunto de puntos de control que de forma global constituyen la malla de puntos de control. *El movimiento de estos puntos de control permite manipular una superficie hasta obtener la forma deseada*.

Podemos encontrar distintas formas de obtener los datos de partida que serán usados como puntos de control, un ejemplo de estas sería a través de la cartilla de trazado que posee las posiciones exactas de los nodos. De usar este método, las coordenadas de los nodos serán exactas, en este caso es útil realizar una interpolación.

También se pueden obtener las mediciones de manera manual mediante la medición directa de una embarcación como puede ser midiendo directamente en esta o mediante el escaneo de un plano de formas para luego trabajar con él, es decir asumiendo cierto ‘’ruido’’ en las mediciones, es mucho más útil realizar un proceso de suavizamiento.

La base del proceso de modelado mediante la ayuda del ordenador está en comprender cómo se pueden usar los puntos de control para obtener superficies con la forma deseada. Esto se explica de forma más clara con la siguiente analogía.

A continuación vamos a describir dos procesos básicos que se pueden realizar con el junquillo, una interpolación y un suavizamiento:

En un problema de ***interpolación*****,** como podemos ver en la siguiente figura, el junquillo se colocará recto en la mesa donde se realizan los trazados.

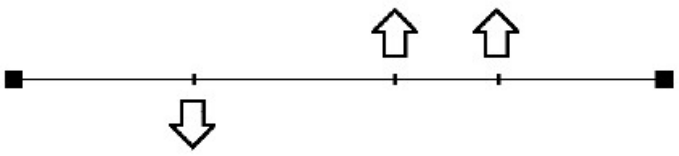


Fig. 12 – Junquillo Recto

A continuación arrastramos el “Spline” en un número finito de puntos mediante la ayuda de unos pesos, la rigidez inherente del material con el que está confeccionado el “Spline” dará como resultado una curva suave que usaremos para dibujar.

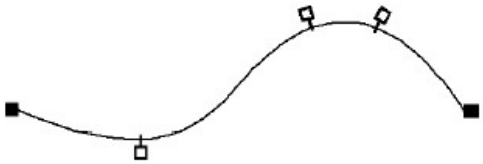


Fig.13 – Junquillo con Puntos de Edición

Estos puntos son los denominados puntos de edición, que son como los puntos de control excepto que siempre están situados en la curva y mover un punto de edición generalmente cambia la forma de toda la curva (mover un punto de control sólo cambia la forma de la curva en una subregión). Los puntos de edición son más útiles cuando se necesita un punto en el interior de una curva para que atraviese exactamente una determinada posición.

La edición de puntos de control es preferible cuando se quiere cambiar la apariencia de una curva y mantener la forma lisa es importante.

En un problema de *suavizamiento*, en lugar de hacer pasar la curva SPLINE a lo largo de unos puntos, nos centraremos en el efecto de atraer la curva SPLINE mediante unos resortes hacia dichos puntos, como mostramos en las siguientes figuras.

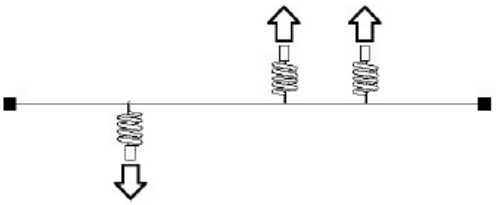


Fig.14 – Junquillo con Muelles antes de suavizar

Al mover un punto de control, la rigidez inherente de los SPLINES y los resortes o muelles es combinada para mantener la curva suave. Este efecto se puede observar ya que lo curva no necesita pasar por cada punto de control, si no que será atraída en mayor o menor medida hacia estos. La curva resultante es suave con sólo

los dos puntos finales de control fijos situados en la curva, como se muestra en la figura siguiente.

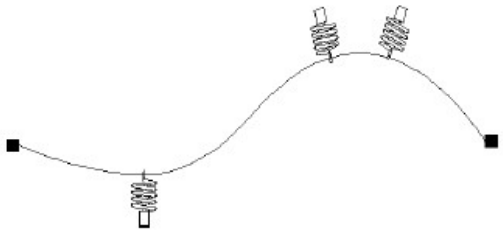


Fig.15 – Junquillo alisado

Moviendo los puntos de control a una u otra posición, será posible curvar el junquillo hasta una forma dada. La curvatura del junquillo estará libre de irregularidades debido a la elasticidad de los muelles y la flexibilidad del propio junquillo. Si el junquillo se hiciera más rígido o flexible, la curvatura sufriría un cambio en su magnitud.

Como hemos dicho, los puntos de control no se encuentran sobre la curva realizada, sino que la curva es atraída hacia la posición de los puntos de control, y entonces los puntos por los que ha de pasar la curva sólo coinciden con los puntos de control en los extremos.

Los puntos de control tienen una influencia local sobre la curva, de modo que el desplazamiento de cualquiera de ellos modificara la zona de la curva en la que tiene influencia, y no a toda la curva. Están dispuestos en forma de malla rectangular con filas y columnas, de modo que cada uno pertenece a una fila y a una columna de la malla. Debido a esto en el dibujo asistido por ordenador podemos encontrar dificultades a la hora de realizar el alisado, ya que no es fácil entender como variará la curva al mover uno de estos puntos, por ello el alisado naval es una operación manual que requiere cierto grado de experiencia.

Aunque éste es un ejemplo en dos dimensiones, los programas de dibujo asistido

por ordenador usan un procedimiento análogo para generar sus SPLINES en tres dimensiones para formar superficies. En la siguiente figura, vemos cómo quedaría un mallado realizado con SPLINES para un buque, más concretamente su proa.

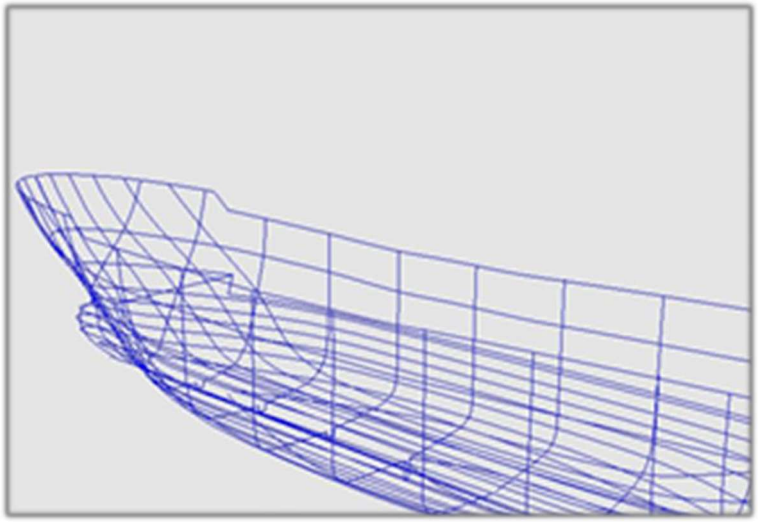


Fig. 16 – Buque creado mediante SPLINES

Del mismo modo que una fila de puntos de control bidimensionales puede definir una curva bidimensional, una red de puntos de control tridimensionales puede definir una superficie tridimensional completa. Si se considera una malla tridimensional de puntos de control es posible imaginar los junquillos situados a lo largo y a lo ancho de la red, definiendo de este modo una superficie. Así una superficie se creará generando los junquillos en tres dimensiones a partir de los puntos de control que constituyen la malla.

Por último, es preciso recordar que, para producir un cambio en la superficie, hay que cambiar la posición relativa de los puntos de control de la malla. El programa que se utilice para realizar el diseño recalculará y mostrará entonces la nueva forma de la superficie. Ocurre del mismo modo que con la analogía del muelle: sólo moviendo los puntos de control es posible cambiar la forma de la superficie, y no moviendo directamente los puntos de la propia superficie.

Capítulo VI
Splines Cúbicos

Capítulo VI
SPLINES Cúbicos

6.1- SPLINES cúbicos Suavizantes

Los SPLINES cúbicos suavizantes son aquellos que se utilizan para la creación de superficies en el dibujo naval, es decir, los que se utilizan en los programas matemáticos para el cálculo de superficie y el diseño de estas.

Se usan cuando una interpolación normal puede provocar oscilaciones muy grandes a partir de los valores dados inicialmente.

El objetivo de los SPLINES suavizantes es disminuir la aleatoriedad en los datos que nos den, ya sean mediciones o valores calculados mediante ensayos.

Como se ha dicho anteriormente el suavizado en si consiste en buscar la función que no pasa por los puntos dados, sino por zonas cercanas a ellos creando entre estos una transición suave.

6.1.1- Definición

Consideramos un retículo

$$\boldsymbol{\omega : a = x_1 < x_2 < \cdots < x_{m-1} < x_m = b}$$

Y dos conjuntos numéricos

$$\boldsymbol{y_1, y_2, \ldots, y_{m-1}, y_m}$$

Denominamos como SPLINE cúbico suavizante a la función S(x) que cumple que:

- Para todo el intervalo: $\boldsymbol{[x_i, x_{i+1}]} \quad \forall \quad \mathbf{i = 1, \ldots, m-1}$

  Será un polinomio de tercer grado

$$\boldsymbol{S(x) = S_i(x) = a_0^i + a_1^i(x - x_i) + a_2^i(x - x_i)^2 + a_3^i(x - x_i)^3}$$

En todo el intervalo, el Spline es un polinomio de tercer grado que se define mediante cuatro coeficientes. En total hay (m - 1) intervalos, por lo que, para definir el SPLINE es necesario hallar 4*(m - 1) números:

$$\boldsymbol{a_0^{(i)}, a_1^{(i)}, a_2^{(i)}, a_3^{(i)}, \forall\, i = 1, \ldots, m-1}$$

- Es dos veces diferenciable con continuidad en el intervalo [a,b], es decir, pertenece a la clase C^2 [a,b].

  Esta condición implica la continuidad de la función S(x) y de sus derivadas S'(x) y S''(x) en todos los nodos del retículo *w*. Como el número de nodos internos es *m – 2,* entonces tendremos *3 x (m – 2)* condiciones.

- En ella se alcanza su límite funcional

$$J(f) = \int_a^b (f''(x))^2 dx + \sum_{i=0}^{m} \frac{1}{\rho_i}(f(x_i) - y_i)^2$$

Donde y_i y ρ_i > 0 son números dados. A los datos de ρ_i se les denomina pesos.

- Satisface unas determinadas condiciones de contorno. Estas se explicarán a continuación.

6.1.2- Condiciones de Contorno

Para tener las condiciones necesarias para la definición unívoca del Spline, nos faltan dos ecuaciones, las cuales se formulan como restricciones sobre los valores del Spline y/o sus derivadas en los extremos del intervalo.

Serán necesarias *4 x (m-1) = 4m – 4* en el intervalo [a,b].

Generalmente para la construcción de un Spline cúbico suavizante se utilizan condiciones de entre los cuatro tipos siguientes:

- Condiciones de Primer tipo: se dan los valores que debe tomar la primera derivada de *S(x)* en los extremos del intervalo [a,b]

$$S'(a) = f'(a) \quad ; \quad S'(b) = f'(b)$$

- Condiciones de segundo tipo: Se dan los valores que debe tomar la segunda derivada de *S(x)* en los extremos del intervalo [a,b]

$$S''(a) = f''(a) \quad ; \quad S''(b) = f''(b)$$

- Condiciones de tercer tipo: serán las denominadas condiciones periódicas.

$$S'(a) = S'(b) \quad ; \quad S''(a) = S''(b)$$

Con esto podremos definir el siguiente teorema:

El SPLINE cúbico *S(x)* que minimiza el funcional y satisface las condiciones de contorno de uno de los tres tipos indicados esta inequívocamente definido.

Por último, podremos definir al SPLINE suavizador de i-ésimo tipo al SPLINE cúbico que minimiza al funcional *J(f)* y que satisface unas determinadas condiciones de

contorno. En nuestro caso al ser un entorno de tres dimensiones, tendremos un SPLINE suavizador cúbico.

Una vez definidas estas condiciones de contorno, lo siguiente que se debe hacer para la realización del SPLINE suavizante es de escoger estas condiciones. Esta elección es una de las mayores problemáticas de la interpolación y tiene una mayor importancia cuando es necesaria asegurar una alta precisión del SPLINE *S(x)* en las zonas próximas de los extremos del intervalo a analizar. Esto es así ya que las condiciones tienen una gran influencia en el comportamiento del SPLINE cerca de los extremos y disminuye conforme nos alejamos de estos.

Utilizaremos condiciones de contorno del primer tipo cuando se conocen los valores de la primera derivada de la función a analizar. Conocer estos valores indica que conocemos la dirección de la tangente a la curva en los extremos.

6.2. SPLINES Interpolantes en discontinuidades

Este trabajo está dedicado a la construcción y análisis de una nueva técnica que permite mejorar la exactitud de SPLINES cerca de esquinas y discontinuidades de salto.

En la literatura sobre SPLINES, siempre se asume que la función debe ser reconstruida y algunas de sus derivadas deben ser continuas. Sin embargo, este no es el caso en muchas situaciones prácticas. No se debe tener en cuenta la presencia de interrupciones, que directamente se traducen en la aparición de fallos en la reconstrucción.

Como consecuencia de asumir la continuidad de la función que tenemos que reconstruir, la difusión también aparecerá cuando el método no toma en cuenta la posición exacta de la discontinuidad. Vamos a ver que este análisis puede hacerse simultáneamente con la reconstrucción y que el tiempo de procesamiento no

se verá muy afectado. En este trabajo intentaremos detectar el problema de reconstruir una función muestreada utilizando SPLINES desde otra perspectiva. En la literatura sobre este problema, los autores siempre han atacado el problema utilizando las muestras de una función. Este método sólo permite detectar la posición de las discontinuidades de la esquina, perdiendo así toda la información sobre la posición de un salto dentro del intervalo.

6.2.1- Introducción

En los textos e investigaciones sobre interpolación y aproximación, el enfoque clásico consiste en el uso de métodos lineales, sin embargo, nosotros nos centraremos en la construcción de SPLINES no lineales. Parece que lo más aceptado es construir directamente una función polinómica por trozos del orden elegido utilizando los datos disponibles, sin tener en cuenta las posibles discontinuidades que pueden contener estos datos. Esta estrategia conduce directamente a la aparición del efecto Gibbs, difusión y otros fallos en la señal reconstruida. Una solución rápida a estos problemas es pre procesar los datos, buscando la presencia de características no deseadas, tales como discontinuidades en la función o sus derivadas.

El caso de la construcción de un SPLINE, es decir, una función polinómica dividida en trozos que coincide con una función discretizada dada f en n números sitios particulares y que satisfacen ciertas condiciones de regularidad en los lugares que queremos analizar, en particular porque las diferencias de los datos disponibles son necesarios a priori con el fin de obtener los coeficientes del SPLINE. Este hecho facilita el trabajo de analizar los datos, como la presencia de grandes diferencias divididas que generalmente indican la presencia de discontinuidades.

En el valor de punto de ajuste, los datos discretizados y los valores de la función continua original por trozos serán los mismos en un determinado intervalo. Sin embargo, en este escenario sólo es posible detectar discontinuidades de la esquina, ya que la posición de las discontinuidades de salto se pierde durante el proceso de discretización. En el marco de la media de la celda, los datos provienen de la integración

de una función en ciertos intervalos. En este contexto es posible detectar la posición de la esquina y discontinuidades de salto. Esto lo usaremos a continuación con el fin de mostrar cómo llevar a cabo la detección de discontinuidades.

A partir de los datos originales de discretizar en las medidas de la celda o de la evaluación por puntos, la función es entonces reconstruida usando interpolaciones, lo que significa que una predicción de un operador lineal o no lineal se utilizará para tratar de recuperar una aproximación de la función original. Si el operador de predicción es lineal, entonces usaremos una herramienta de interpolación lineal para conseguir una aproximación de la función original. El problema que surge del uso de los operadores lineales de predicción es la precisión de la aproximación cerca de las discontinuidades: el orden de aproximación se pierde debido al efecto de Gibbs y difusión, por lo que cualquier punto o zona que toque la discontinuidad se verá afectada por esta, lo cual se explicará con mayor detenimiento más adelante, provocando que la aproximación sea inexacta. Debido a este hecho, el aumento de la longitud de la zona a estudiar no mejorará la precisión de la aproximación y repercutirá en más regiones afectadas alrededor de las discontinuidades. Una solución para este problema es elegir las zonas de estudio que no posean discontinuidades. La idea que vamos a desarrollar no es exactamente esa, sino la construcción de SPLINES que no crucen las discontinuidades para poder facilitar su estudio.

6.2.2- Discretización del entorno de datos en los puntos característicos y en la dimensión de la celda

El análisis de la función f(x) en los sitios donde x = xi sólo es útil para detector y localizar la posición de discontinuidades de esquina.

El proceso de muestreo que permite obtener los pares de datos $(x;f(x))$, intrínsecamente produce la pérdida de información de la posición de las discontinuidades de salto. A pesar de que, si asumimos que los datos provienen de una discretización mediante promedios locales de la función f (x).entonces es posible detectar discontinuidades de salto y esquina

en f (x). Nosotros nos centraremos en la discretización de puntos característicos.

6.2.3- Discretización de los datos de ajuste en puntos característicos.

Consideremos el conjunto de funciones continuas por partes en el intervalo [a,b], el espacio de secuencias finitas V de longitud N = J + 1 y que X sea una partición uniforme o no uniforme del intervalo [a, b] en subintervalos J:

$$X = \{x_i\}_{i=1}^{J+1}, \qquad X_0 = a, \qquad X_i = ih_i$$

Consideremos ahora la operación de discretización en el ajuste del punto característico $\boldsymbol{f} = \boldsymbol{D}\boldsymbol{f}$, que definiremos como,

$$f_i = (D\,f)_i = f(x_i), \qquad f = \{f_i\}_{i=1}^{J+1},$$

Es fácil ver que la discretización que se describe sólo conserva la información local de los puntos X_i. Es posible localizar las discontinuidades de esquina al realizar esto, sin embargo, es imposible la localización de las discontinuidades de salto. Este problema se tratará en los siguientes apartados mediante la discretización por promedios locales.

6.3- Construcción de SPLINES cúbicos adaptados a discontinuidades.

Vamos a realizar una introducción a la construcción habitual de SPLINES cúbicos y posteriormente introduciremos las nuevas técnicas para adaptar los SPLINES clásicos a la presencia de discontinuidades de esquina y de salto. El objetivo es la obtención de una reconstrucción aguda o precisa de las discontinuidades de esquina y evitar la aparición del fenómeno de Gibbs y difusión en las discontinuidades de salto.

6.3.1- SPLINES Cúbicos Clásicos

En este trabajo vamos a intentar hacer algo distinto a lo que se suele encontrar en la literatura del tema, que será la realización de una secuencia de polinomios de tercer grado con el fin de obtener una función polinómica por trozos que vendrá nombrada como C^2*: SPLINES cúbicos,* que será capaz de localizar con precisión y recrear gráficos que posean discontinuidades de esquina. Vamos a comenzar con n+1 conjuntos de valores de (x_i, y_i) y la expresión del polinomio en un intervalo particular [x_i, x_{i+1}]

$$g_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i, \text{ para } i = 0, 1, \dots, n - 1.$$

Por lo que tenemos las siguientes condiciones:

$$g_i(x,) = y_i \cdot i = 0, 1, \dots, n - 1. \text{ y } g_{n-1}\, t(x_n) = y_n$$

$$g_i(x_{i+1}) = g_{i+1}(x_{i+1}),\ i = 0, 1, \dots, n - 2,$$

$$g_i'(x_{i+1}) = g_{i+1}'(x_{i+1}), i = 0, 1, \dots, n - 2,$$

$$g_i''(x_{i+1}) = g_{i+1}''(x_{i+1}),\quad i = 0, 1, \dots, n - 2.$$

Estas condiciones nos indican la adaptación del Spline cúbico a los pares de valores iniciales de n+1 (x,y) y que es continuo en la región determinada. Por tanto con n+1 pares de datos, tendremos n intervalos, n funciones polinómicas y 4*n incógnitas que serán la a,b,c,d… de la ecuación del polinomio en el intervalo particular. Con las ecuaciones anteriores tendremos la siguiente ecuación:

$$d_i = y_i, \qquad i = 0,1,\dots,n-1$$

Y de la ecuación del punto próximo sin derivar obtendremos:

$$y_{i+1} = a_i(x_{i+1} - x_i)^3 + b_i(x_{i+1} - x_i)^2 + c_i(x_{i+1} - x_i) + y_i$$
$$= a_i h_i^3 + b_i h_i^2 + c_i h_i + y_i$$

Donde hemos utilizado $h_i = (x_{i+1} - x_i)$; así al derivar la ecuación anterior tendremos:

$$g_i'(x) = 3a_i(x - x_i)^2 + 2b_i(x - x_i) + c_i$$
$$g_i''(x) = 6a_i(x - x_i) \quad + 2b_i, \qquad i = 0,1,\dots,n-1$$

Si enunciamos las ecuaciones según su segunda derivada $S_i = g_i''(x_i)$ *para i=0,1,…, n-1* y $S_n = g_{n-1}''(x_n)$ el sistema se transforma en un sistema tridiagonal que siempre tiene solución.

$$S_i(x_i) = 6a_i(x_i - x_i) + 2b_i = 2b_i$$
$$S_i(x_{i+1}) = 6a_i(x_{i+1} - x_i) + 2b_i = 6a_i h_i + 2b_i$$

Operando en estas ecuaciones podremos deducir el valor de a y b que será:

$$b_i = \frac{S_i}{2}$$
$$a_i = \frac{S_{i+1} - S_i}{6h_i}$$

Y finalmente introduciendo estos valores para operar en nuestra ecuación inicial tendremos el valor de c:

$$c_i = \frac{y_{i+1} - y_i}{h_i} - \frac{2h_i S_i + h_i + S_{i+1}}{6}$$

Las pendientes del polinomio en ambos lados del punto (x_i, y_i) serán iguales. Con todas estas ecuaciones podremos deducir dos expresiones para la derivada de y (y'):

$$y_i' = y_i'(x_i) = 3a_i(x_i - x_i)^2 + 2b_i(x_i - x_i) + c_i = c_i$$
$$y_{i-1}' = y_{i-1}'(x_i) = 3a_{i-1}(x_i - x_{i-1})^2 + 2b_{i-1}(x_i - x_{i-1}) + c_{i-1}$$
$$= 3a_{i-1}(h_{i-1})^2 + 2b_{i-1}h_{i-1} + c_{i-1}$$

Con todo esto podemos expresar el sistema de ecuaciones en función de la segunda derivada del Spline, S:

$$h_{i-1}S_{i-1} + 2(h_{i-1} + h_i)S_i + h_iS_{i+1} = 6(\frac{y_{y+1} - y_i}{h_i} - \frac{y_i - y_{y-1}}{h_{i-1}})$$

Esta ecuación será un sistema tridiagonal en el que las diferencias están divididas en el lado derecho. Con esto tendremos varias opciones para modelizar las condiciones de contorno, por ejemplo tomando $S_0 = S_n = 0$ y no perderíamos la generalidad de la zona.

6.3.2- SPLINES Cúbicos Adaptados

Vamos a empezar con el análisis de un punto característico y suponemos que hay una discontinuidad de esquina en *x** en el intervalo $[x_i, x_{i+1}]$. Como veremos más adelante, es fácil obtener una aproximación de *x** y el valor de *f(x*)* y sus derivadas en *x** en ambos lados de la discontinuidad usando el método que vamos a proponer en las páginas sucesivas. Por el momento, supongamos que podemos conocer la posición de la discontinuidad hasta la exactitud deseada y que podemos obtener una aproximación de la función y sus derivados de ambos lados de la discontinuidad. Etiquetaremos toda la información sobre la función a la izquierda de la discontinuidad como posititvas (+). La información a la derecha se etiquetará como negativa (-).

Así, teniendo una aproximación del valor de ***f*** y de sus derivadas en x *, es fácil asegurar la adaptación de la Spline a la discontinuidad.

$$h_{i-1}S_{i-1} + 2(h_{i-1} + h_i)S_i + h_iS_{i+1} = 6(\frac{y_{y+1} - y_i}{h_i} - \frac{y_i - y_{y-1}}{h_{i-1}})$$

Este sistema puede ser dividido en dos sistemas donde la información que hemos obtenido sobre la segunda derivada de los dos lados de la discontinuidad se transformará en condiciones de contorno para estos dos sistemas. En concreto, si hemos encontrado $f^-(x^*), f_x^-(x^*), f_{xx}^-(x^*), f^+(x^*), f_x^+(x^*)$ que definiremos posteriormente dividiremos el sistema anterior en dos partes.

El primer sistema estará compuesto por una diagonal con la ecuación $2\ (h_0 + h_1)$ por la cual empezará y acabará en $2\ (h_{i-1} + h_i)$, estando esta siempre rodeada por los h_n que están dentro del paréntesis, excepto en los extremos que solo se verán afectados por el término mayor en el primer dato y el término menor en el último. A esta matriz se le multiplica la de sus propias derivadas S_i y se acabará formando una expresión equivalente de la siguiente forma:

$$6\,x \begin{bmatrix} \frac{f_2 - f_1}{h_1} - \frac{f_1 - f_0}{h_0} \\ \frac{f_3 - f_2}{h_2} - \frac{f_2 - f_1}{h_1} \\ \dots \\ \dots \\ \frac{f^+(x^*) - f_i}{h_i} - \frac{f_i - f_{i-1}}{h_{i-1}} \end{bmatrix}$$

6.4 - DETECCIÓN DE LA POSICIÓN DE LAS DISCONTINUIDADES

Cuando trabajamos con una serie de datos que pueden presentar o presentan discontinuidades, es posible que podamos localizarlas. Si la función es de tramos lisos, sin discontinuidades de esquina, con una discontinuidad de salto aislada en dicha función y la reconstrucción se realiza usando una función similar $f_i = f(x_i)$, entonces las posibilidades de encontrar dicha discontinuidad serán nulas dentro del intervalo donde está contenida así como su posición exacta. Para detectar discontinuidades de salto, el muestreo de los puntos característicos debe reemplazarse por un promedio local, como los promedios de celdas.

Vamos a describir cómo se plante la resolución para la detección de discontinuidades. Siendo h el espaciamiento de la cuadrícula, utilizaremos el algoritmo de diferencias de segundo orden:

$$\Delta_h^2 f(x) = f(x) - 2f(x+h) + f(x+2h)$$

Como indicador de intervalos donde es posible que pueda estar la discontinuidad. A los intervalos con alta probabilidad de aparición de discontinuidades se les asignara la letra B. Estos candidatos serán los que cumplan:

-Si

$$\left|\Delta_h^2 f\big((k-1)h\big)\right| > \left|\Delta_h^2 f\big((k-1\pm n)h\big)\right|, \qquad n = 1, \dots, m.$$

Entonces los intervalosI_{k-1} e I_k serán denominados con la letra B. Este caso suele considerar que la situación de las discontinuidades está en un punto del intervalo, sin embargo, no sabremos él intervalo en el cual aparecerá la discontinuidad por lo que todos son sospechosos de poseerla.

-Si

$$\left|\Delta_h^2 f(kh)\right| > \left|\Delta_h^2 f\big((k+n)h\big)\right|, \qquad n = 1, \dots, m-1.$$

Y se cumple que:

$$\left|\Delta_h^2 f\big((k-1)h\big)\right| > \left|\Delta_h^2 f\big((k-1-n)h\big)\right|, \qquad n = 1, \dots, m-1.$$

Entonces el intervalo se denominará B y es en este caso que se cumple que las diferencias divididas más grandes incluirán el intervalo I_k, que será el candidato para contener la discontinuidad. El resto de intervalos se denominan con la letra G y se supone que en ellos no se encuentra ninguna singularidad.

El algoritmo se diseña para que cumpla su función con diferencias de h suficientemente pequeñas. Puede darse el caso de que el intervalo I_k denominado como B se encuentre en una región suave. Para conseguir una aproximación de la situación de la discontinuidad usamos una aproximación simple por medio de un polinomio de Lagrange de tercer orden tanto por la izquierda como por la derecha del intervalo que consideramos sospechoso de poseer la discontinuidad, estos se denominaran por BB.

Entonces, definimos una función G que será la diferencia entre los dos polinomios y suponemos que solo posee una sola raíz en el intervalo que estamos analizando. Con esto, la posición de la discontinuidad puede hallarse de una manera

mucho más sencilla encontrando las raíces de un polinomio de grado dos. Si bien es cierto que la precisión de dicho polinomio aumenta conforme aumenta el grado de este. Podemos usar el algoritmo de Newton poniendo como condición inicial mediante el método de bisección para encontrar, con la precisión adecuada, la situación de la discontinuidad. Podemos hallar las discontinuidades de esquina por medio de puntos característicos, las discontinuidades de salto y esquina en la configuración de celda puede ser detectada y localizada usando la función primitiva. El uso de SPLINES es deseable debido a que permiten la detección de discontinuidades a la vez que su posición usando localizaciones arbitrarias. Este método se podría utilizar para diferencias divididas no uniformes.

6.5 CÁLCULOS EN LA POSICIÓN DE LAS DISCONTINUIDADES

Aquí vamos a mostrar un método para la obtención del valor de la función y sus derivadas en los puntos cercanos a la discontinuidad tanto para puntos característicos como para intervalos definidos.

6.5.1- Obtención de los valores de f(x) y sus derivadas para discontinuidades en puntos característicos.

Una vez conocida la posición de la discontinuidad, podemos intentar obtener información acerca de la función y sus derivadas cerca de los puntos en los cuales hemos encontrado discontinuidades. Tendremos que asumir que solo vamos a encontrarnos con discontinuidades de esquina.

Si trabajamos con *Splines cúbicos* es conveniente el uso de al menos cuatro puntos para poder obtener el valor de la función y de sus derivadas en un punto X para asegurar la precisión en zonas suaves si se produce un fallo en la detección

de la discontinuidad. Vamos a suponer que conocemos la localización de la discontinuidad, y que esta estará en X, a una distancia α de X_j en el intervalo $\{X_j, X_{j+1}\}$ y supongamos que estamos trabajando con el stencil: $\{f_{j-3}^+, f_{j-2}^+, f_{j-1}^+, f_j^+, f_{j+1}^+, f_{j+2}^+, f_{j+3}^+, f_{j+4}^+\}$ que ocupan las posiciones respectivas de $\{x_{j-3}^+, x_{j-2}^+, x_{j-1}^+, x_j^+, x_{j+1}^+, x_{j+2}^+, x_{j+3}^+, x_{j+4}^+\}$. Entonces podremos obtener los valores de las derivadas de f a ambos lados de la discontinuidad mediante el uso de Taylor alrededor de X y combinándolo con el siguiente sistema de ecuaciones:

$$f_j^+ = f^+(X) - f_x^+(X)\alpha + \frac{1}{2}f_{xx}^+(X)\alpha^2 - \frac{1}{3!}f_{xxx}^+(X)\alpha^3$$

$$f_{j-1}^+ = f^+(X) - f_x^+(X)(h+\alpha) + \frac{1}{2}f_{xx}^+(X)(h+\alpha)^2 - \frac{1}{3!}f_{xxx}^+(X)(h+\alpha)^3$$

$$f_{j-2}^+ = f^+(X) - f_x^+(X)(2h+\alpha) + \frac{1}{2}f_{xx}^+(X)(2h+\alpha)^2 - \frac{1}{3!}f_{xxx}^+(X)(2h+\alpha)^3$$

$$f_{j-3}^+ = f^+(X) - f_x^+(X)(3h+\alpha) + \frac{1}{2}f_{xx}^+(X)(3h+\alpha)^2 - \frac{1}{3!}f_{xxx}^+(X)(3h+\alpha)^3$$

$$f_{j+1}^+ = f^-(X) + f_x^-(X)(h-\alpha) + \frac{1}{2}f_{xx}^-(X)(h-\alpha)^2 - \frac{1}{3!}f_{xxx}^-(X)(h-\alpha)^3$$

$$f_{j+2}^+ = f^-(X) + f_x^-(X)(2h-\alpha) + \frac{1}{2}f_{xx}^-(X)(2h-\alpha)^2 - \frac{1}{3!}f_{xxx}^-(X)(2h-\alpha)^3$$

$$f_{j+3}^+ = f^-(X) + f_x^-(X)(3h-\alpha) + \frac{1}{2}f_{xx}^-(X)(3h-\alpha)^2 - \frac{1}{3!}f_{xxx}^-(X)(3h-\alpha)^3$$

$$f_{j+4}^+ = f^-(X) + f_x^-(X)(4h-\alpha) + \frac{1}{2}f_{xx}^-(X)(4h-\alpha)^2 - \frac{1}{3!}f_{xxx}^-(X)(4h-\alpha)^3$$

Siendo α la diferencia entre $X\ y\ X_j$

Vamos a suponer, para simplificar los cálculos, que las diferencias serán uniformes, es decir, la distancia entre un punto del espacio y el siguiente será igual a la del primero con el anterior y así sucesivamente. Trabajando en ambos sistemas donde $f^+(X)f_x^+(X)f_{xx}^+(X)f_{xxx}^+(X)f^-(X)f_x^-(X)f_{xx}^-(X)f_{xxx}^-(X)$ son las incógnitas, el valor de la función y sus derivadas puede ser obtenido fácilmente.

Ahora bien, debemos comprobar el error que o más bien la precisión de la aproximación. Este error en la aproximación, para nuestro caso que utilizamos un espacio uniforme. Con esto, podemos expresar de forma matricial:

$$\begin{pmatrix} f_j^+ \\ f_{j-1}^+ \\ f_{j-2}^+ \\ f_{j-3}^+ \end{pmatrix} = \begin{pmatrix} 1 & \alpha & \frac{1}{2}\alpha^2 & -\frac{1}{3!}\alpha^3 \\ 1 & (h+\alpha) & \frac{1}{2}(h+\alpha)^2 & -\frac{1}{3!}(h+\alpha)^3 \\ 1 & (h+\alpha) & \frac{1}{2}(2h+\alpha)^2 & -\frac{1}{3!}(2h+\alpha)^3 \\ 1 & (h+\alpha) & \frac{1}{2}(3h+\alpha)^2 & -\frac{1}{3!}(3h+\alpha)^3 \end{pmatrix} \begin{pmatrix} f^+(X) \\ f_x^+(X) \\ f_{xx}^+(X) \\ f_{xxx}^+(X) \end{pmatrix} + \begin{pmatrix} O(h)^4 \\ O(h)^4 \\ O(h)^4 \\ O(h)^4 \end{pmatrix}$$

Y realizamos la inversa del Sistema de matrices:

$$A = \begin{pmatrix} \frac{6h^3+11h^2\alpha+6h\alpha^2+\alpha^3}{6h^3} & -\frac{\alpha(6h^2+5h\alpha+\alpha^2)}{2h^3} & \frac{\alpha(3h^2+4h\alpha+\alpha^2)}{2h^3} & -\frac{\alpha(2h^2+3h\alpha+\alpha^2)}{6h^3} \\ \frac{11h^2+12h\alpha+3\alpha^2}{6h^3} & -\frac{6h^2+10h\alpha+3\alpha^2}{2h^3} & \frac{3h^2+8h\alpha+3\alpha^2}{2h^3} & -\frac{2h^2+6h\alpha+3\alpha^2}{2h^3} \\ \frac{2h+\alpha}{h^3} & -\frac{5h+3\alpha}{h^3} & \frac{4h+3\alpha}{h^3} & -\frac{h+\alpha}{h^3} \\ h^{-3} & -3h^{-3} & -3h^{-3} & -h^{-3} \end{pmatrix}$$

Como $\alpha = O(h)$ podremos obtener la precisión adquirida:

$$\begin{pmatrix} f^+(X) \\ f_x^+(X) \\ f_{xx}^+(X) \\ f_{xxx}^+(X) \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{pmatrix} \begin{pmatrix} f_j^+ \\ f_{j-1}^+ \\ f_{j-2}^+ \\ f_{j-3}^+ \end{pmatrix} + \begin{pmatrix} O(h)^4 \\ O(h)^4 \\ O(h)^4 \\ O(h)^4 \end{pmatrix} \qquad (20)$$

Realizando la multiplicación, llegaremos a la expresión de la función y sus derivadas en la discontinuidad para $x = X$. Para solucionar el otro sistema podemos proceder manera similar a la anterior:

$$\begin{pmatrix} f_{j+1}^- \\ f_{j+2}^- \\ f_{j+3}^- \\ f_{j+4}^- \end{pmatrix} = \begin{pmatrix} 1 & h-\alpha & \frac{(h-\alpha)^2}{2} & \frac{(h-\alpha)^3}{6} \\ 1 & 2h-\alpha & \frac{(2h-\alpha)^2}{2} & \frac{(2h-\alpha)^3}{6} \\ 1 & 3h-\alpha & \frac{(3h-\alpha)^2}{2} & \frac{(3h-\alpha)^3}{6} \\ 1 & 4h-\alpha & \frac{(4h-\alpha)^2}{2} & \frac{(4h-\alpha)^3}{6} \end{pmatrix} \begin{pmatrix} f^-(X) \\ f_x^-(X) \\ f_{xx}^-(X) \\ f_{xxx}^-(X) \end{pmatrix} + \begin{pmatrix} O(h)^4 \\ O(h)^4 \\ O(h)^4 \\ O(h)^4 \end{pmatrix}$$

Proseguimos mediante la realización de la inversa:

$$B = \begin{pmatrix} -\frac{24h^3 + 26h^2\alpha - 9h\alpha^2 + \alpha^3}{6h^3} & \frac{(-h+\alpha)(12 + 7h\alpha + \alpha^2)}{2h^3} & -\frac{(-h+\alpha)(8h^2 + 6h\alpha + \alpha^2)}{2h^3} & -\frac{\alpha(2h^2 + 3h\alpha + \alpha^2)}{6h^3} \\ -\frac{26h^2 - 18h\alpha + 3\alpha^2}{6h^3} & \frac{19h^2 - 16h\alpha + 3\alpha^2}{2h^3} & -\frac{14h^2 - 14h\alpha + 3\alpha^2}{2h^3} & \frac{11h^2 - 12h\alpha + 3\alpha^2}{6h^3} \\ -\frac{3h+\alpha}{h^3} & -\frac{8h+3\alpha}{h^3} & -\frac{-7h+3\alpha}{h^3} & -\frac{2h+\alpha}{h^3} \\ -h^{-3} & 3h^{-3} & -3h^{-3} & h^{-3} \end{pmatrix}$$

Como hemos hecho anteriormente:

$$\begin{pmatrix} f^-(X) \\ f_x^-(X) \\ f_{xx}^-(X) \\ f_{xxx}^-(X) \end{pmatrix} = \begin{pmatrix} B_{11} & B_{12} & B_{13} & B_{14} \\ B_{21} & B_{22} & B_{23} & B_{24} \\ B_{31} & B_{32} & B_{33} & B_{34} \\ B_{41} & B_{42} & B_{43} & B_{44} \end{pmatrix} \begin{pmatrix} f_j^- \\ f_{j-1}^- \\ f_{j-2}^- \\ f_{j-3}^- \end{pmatrix} + \begin{pmatrix} O(h)^4 \\ O(h)^4 \\ O(h)^4 \\ O(h)^4 \end{pmatrix} \qquad (30)$$

Y con esto, tendremos las aproximaciones de la función y sus derivadas a ambos lados de la discontinuidad.

6.5.2- Obtención de los valores de f(x) y sus derivadas para discontinuidades en intervalos.

Podemos estar trabajando en un sistema de intervalos, por lo que tendremos dos opciones; podemos trabajar con la función primitiva o bien trabajar con los valores de los intervalos. El primer caso es directo ya que solo deberíamos hallar la función primitiva y posteriormente aplicando los sistemas (20) y (30) que hemos hallado en el apartado anterior podremos obtener los valores de la función y sus derivadas a ambos lados de la discontinuidad.

Para obtener la función que trabaje directamente en los intervalos tendremos que suponer que la discontinuidad se encuentra en el intervalo (x_j, x_{j+1}) cuya separación será α. Empezaremos trabajando en los puntos característicos con un espacio uniforme *h,* las expresiones para espacios no uniformes se mostrarán más adelante. Por tanto tendremos un sistema similar al anterior de puntos característicos salvo que introducimos las los puntos característicos de las primitivas:

$$\begin{pmatrix} F_j^+ \\ F_{j-1}^+ \\ F_{j-2}^+ \\ F_{j-3}^+ \\ \dots \end{pmatrix} = \begin{pmatrix} 1 & -\alpha & \frac{1}{2}\alpha^2 & -\frac{1}{3!}\alpha^3 & \dots \\ 1 & -(h+\alpha) & \frac{1}{2}(h+\alpha)^2 & -\frac{1}{3!}(h+\alpha)^3 & \dots \\ 1 & -(h+\alpha) & \frac{1}{2}(2h+\alpha)^2 & -\frac{1}{3!}(2h+\alpha)^3 & \dots \\ 1 & -(h+\alpha) & \frac{1}{2}(3h+\alpha)^2 & -\frac{1}{3!}(3h+\alpha)^3 & \dots \\ \dots & \dots & \dots & \dots & \dots \end{pmatrix} \begin{pmatrix} F^+(X) \\ F_x^+(X) \\ F_{xx}^+(X) \\ F_{xxx}^+(X) \\ \dots \end{pmatrix}$$

Suprimimos $O(h)^n$para simplificar la ecuación ya que no estamos interesados en analizar la precisión de esta.

La inversa del sistema en términos de intervalos será:

$$\begin{pmatrix} F^+(X) \\ F_x^+(X) \\ F_{xx}^+(X) \\ F_{xxx}^+(X) \\ \dots \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} & \dots \\ A_{21} & A_{22} & A_{23} & A_{24} & \dots \\ A_{31} & A_{32} & A_{33} & A_{34} & \dots \\ A_{41} & A_{42} & A_{43} & A_{44} & \dots \\ \dots & \dots & \dots & \dots & \dots \end{pmatrix} \begin{pmatrix} h\sum_{i=1}^{j} \bar{f}_i^+ \\ h\sum_{i=1}^{j-1} \bar{f}_i^+ \\ h\sum_{i=1}^{j-2} \bar{f}_i^+ \\ h\sum_{i=1}^{j-3} \bar{f}_i^+ \\ \dots \end{pmatrix}$$

Podemos hacer lo mismo con el sistema inicial para trabajar con la función primitiva:

$$\begin{pmatrix} F_{j+1}^+ \\ F_{j+2}^+ \\ F_{j+3}^+ \\ F_{j+4}^+ \\ \dots \end{pmatrix} = \begin{pmatrix} 1 & h-\alpha & \frac{1}{2}(h-\alpha)^2 & \frac{1}{6}(h-\alpha)^3 & \dots \\ 1 & 2h-\alpha & \frac{1}{2}(2h-\alpha)^2 & \frac{1}{6}(2h-\alpha)^3 & \dots \\ 1 & 3h-\alpha & \frac{1}{2}(3h-\alpha)^2 & \frac{1}{6}(3h-\alpha)^3 & \dots \\ 1 & 4h-\alpha & \frac{1}{2}(4h-\alpha)^2 & \frac{1}{6}(4h-\alpha)^3 & \dots \\ \dots & \dots & \dots & \dots & \dots \end{pmatrix} \begin{pmatrix} F^-(X) \\ F_x^-(X) \\ F_{xx}^-(X) \\ F_{xxx}^-(X) \\ \dots \end{pmatrix}$$

Como hemos hecho anteriormente con el otro “lado”, expresaremos la inversa en términos de intervalos mediante la expresión de los valores primitivos:

$$\begin{pmatrix} F^{-}(X) \\ F_x^{-}(X) \\ F_{xx}^{-}(X) \\ F_{xxx}^{-}(X) \\ \dots \end{pmatrix} = \begin{pmatrix} B_{11} & B_{12} & B_{13} & B_{14} & \dots \\ B_{21} & B_{22} & B_{23} & B_{24} & \dots \\ B_{31} & B_{32} & B_{33} & B_{34} & \dots \\ B_{41} & B_{42} & B_{43} & B_{44} & \dots \\ \dots & \dots & \dots & \dots & \dots \end{pmatrix} \begin{pmatrix} h\sum_{i=1}^{j+1} \bar{f}_i^{-} \\ h\sum_{i=1}^{j+2} \bar{f}_i^{-} \\ h\sum_{i=1}^{j+3} \bar{f}_i^{-} \\ h\sum_{i=1}^{j+4} \bar{f}_i^{-} \\ \dots \end{pmatrix}$$

Con esto, mediante una serie de operaciones, obtendremos las siguientes ecuaciones

$$\begin{pmatrix} F^{-}(X) \\ F_x^{-}(X) \\ F_{xx}^{-}(X) \\ F_{xxx}^{-}(X) \end{pmatrix} = \begin{pmatrix} 1 & \frac{(2h^2\alpha + 6h^3 + \alpha^3 + 3h\alpha^2)}{6h^2} & -\frac{(-2\alpha^3 - 9h\alpha^2 - 7h^2\alpha + 6h^3)}{2h^2} & -\frac{(6h^3 + 11h^2\alpha + 6h\alpha^2 + \alpha^3)}{6h^2} \\ 0 & \frac{2h^2 + 6h\alpha + 3\alpha^2}{6h^2} & \frac{-6\alpha^2 - 7h^2 - 18h\alpha}{6h^2} & \frac{11h^2 - 12h\alpha + 3\alpha^2}{6h^3} \\ 0 & \frac{h + \alpha}{6h^2} & \frac{-3h - 2\alpha}{6h^2} & \frac{-2h + \alpha}{6h^2} \\ 0 & \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} \end{pmatrix} \begin{pmatrix} h\sum_{i=1}^{j-3} \bar{f}_i \\ \bar{f}_{j-2} \\ \bar{f}_{j-1} \\ \bar{f}_j \end{pmatrix}$$

$$\begin{pmatrix} F^{+}(X) \\ F_x^{+}(X) \\ F_{xx}^{+}(X) \\ F_{xxx}^{+}(X) \end{pmatrix} = \begin{pmatrix} 1 & \frac{(-9h^2\alpha - 18h^3 + \alpha^3 + 26h\alpha^2)}{6h^2} & -\frac{(-2\alpha^3 - 31h^2\alpha + 15h\alpha^2\alpha + 18h^3)}{6h^2} & -\frac{(2h^2 + 3h\alpha + \alpha^2)}{6h^3} \\ 0 & \frac{26h^2 - 18h\alpha + 3\alpha^2}{6h^2} & \frac{-6\alpha^2 - 31h^2 + 30h\alpha}{6h^2} & \frac{11h^2 - 12h\alpha + 3\alpha^2}{6h^3} \\ 0 & \frac{-3h + \alpha}{h^2} & \frac{-3h - 2\alpha}{6h^2} & \frac{2h + \alpha}{h^2} \\ 0 & \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} \end{pmatrix} \begin{pmatrix} h\sum_{i=1}^{j+1} \bar{f}_i \\ \bar{f}_{j+2} \\ \bar{f}_{j+3} \\ \bar{f}_{j+4} \end{pmatrix}$$

Con todo esto y realizando una serie de operaciones algebraicas finalmente podremos obtener $\bar{f}^{-}(X)\ y\ \bar{f}^{+}(X)$:

$$\bar{f}^{-}(X) = \left(\frac{F^{-}(X) - F_j^k}{h - \alpha}\right) = \frac{\alpha(2h^2 + 3h\alpha + \alpha^2)}{6h^2(h - \alpha)}\bar{f}_{j-2} + \frac{\alpha(-2\alpha^2 - 9h\alpha - 7h^2)}{6h^2(h - \alpha)}\bar{f}_{j-1} + \frac{\alpha(\alpha^2 + 6h\alpha + 11h^2)}{6h^2(h - \alpha)}\bar{f}_j$$

$$\bar{f}^{+}(X) = \left(\frac{F^{+}(X) - F_j}{\alpha}\right) = \frac{h}{\alpha}\bar{f}_{j+1}\frac{(-9h\alpha^2 - 18h^3 + 26h^2\alpha + \alpha^3)}{6h^2\alpha}\bar{f}_{j+2} + \frac{(-2\alpha^2 + 15h\alpha^2 - 31h^2\alpha + 18h^3)}{6h^2\alpha}\bar{f}_{j+3} + \frac{(-6h^3 + 11h^2\alpha - 6h\alpha^2 + \alpha^3)}{6h^2\alpha}\bar{f}_{j+4}$$

Con todas estas expresiones en términos de intervalos, queda demostrado que es posible trabajar directamente con los valores de los intervalos cuando la discontinuidad se da en X. Podemos ver que en las ecuaciones de los SPLINES adaptados anteriores, para los valores hallados, solo serán necesarias las expresiones $\bar{f}^{+}(X)\ , \bar{f}^{-}(X), F_{xx}^{+}(X)\ y\ F_{xx}^{-}(X)$. Siguiendo un proceso similar, podemos obtener los valores de la segunda derivada en los límites $S_0\ y\ S_n$ que aparecen en la ecuación de los SPLINES.

6.6 Orden de Aproximación cerca de las discontinuidades

En este apartado vamos a analizar el efecto de las aproximaciones mediante diferencias finitas en los STENCILS que cruzan la discontinuidad. Las expresiones de los SPLINES cúbicos adaptados están diseñadas para trabajar en conjuntos de datos suaves. Cuando el STENCIL contiene una discontinuidad, el orden de precisión obtenido mediante SPLINES lineales clásicos se pierde. Esto tiene lógica si tenemos en cuenta el teorema de Weirstrass, que nos dice lo siguiente:

- Si una función *f (x)* es continua en un intervalo [a,b] finito, entonces existirá un polinomio $P_n(x)$ de grado n tal que:

$$|f(x) - P_n(x)| < e$$

Para cada intervalo [a,b], con un error e > 0. El grado de P_n es función de e.

Este teorema solo es válido para funciones continuas ya que dichos polinomios y sus derivadas son funciones continuas. Está claro que si queremos aproximar una parte concreta de una función continua usando SPLINES, el teorema de aproximación de Weierstrass nos dice que solo tomaremos información a uno de los lados de la discontinuidad para construir el SPLINE. Esta es la idea que proponemos en este trabajo. Si asumimos que podemos conocer con certeza la posición de los saltos en la función y sus derivadas con suficiente precisión es natural escoger los datos de uno de los lados para poder evitar el efecto de la discontinuidad cuando reconstruyamos la función por partes. Siguiendo esta idea, podemos dividir los SPLINES que contengan la o las discontinuidades usando puntos adicionales en la posición donde hemos localizado esta discontinuidad.

En las ecuaciones de los SPLINES cúbicos adaptados se muestra que el límite superior de la localización obtenida mediante la interpolación de Lagrange es $O(h^{\alpha})$

para cada α < 1, si el STENCIL toca una discontinuidad de esquina. Vamos a empezar pensando acerca del intervalo [a,b] y vamos a suponer que la función es continua en dicho intervalo. También suponemos que $f(x)$ ha sido reconstruido utilizando un SPLINE cúbico en el intervalo y finalmente supondremos que estamos trabajando con puntos característicos. Para hacer esto las diferencias de segundo orden del conjunto de datos discreto para todo el dominio donde queremos reconstruir son necesarios. Vamos a intentar obtener una aproximación de $f(x)\ para\ x_0 \in [a,b]$, usando el SPLINE de orden n $g(x)$, que podremos hallarlo mediante las fórmulas que hemos dado anteriormente en este capítulo. Una vez tenemos la expresión de $g(x)$, podemos particularizarla para cada punto del intervalo [a,b]. Si tomamos valores del SPLINE en *n* puntos distintos $x_j \in [a,b]$,podremos expresarlo usando la forma de Newton para la interpolación polinómica para puntos característicos.

$$p_n(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots c_n(x - x_0)(x - x_1) \dots (x - x_{n-1})$$

cuando $x_j \to x_0$ para todo *j.* Los coeficientes c_i se calculan fácilmente mediante el uso de diferencias dividias. Podemos denotar las diferencias dividas como:

$$g[x_j \dots, x_{j+k}] = \frac{g\left[x_{j+1}, \dots, x_{j+k}\right] - g[x_j, \dots, x_{j+k-1}]}{x_{j+k} - x_j}$$

Entonces podremos introducirlo en la ecuación de Newton que quedará de la siguiente manera:

$$p_n(x) = g[x_0] + g[x_0, x_1](x - x_0) + g\left[x_{0,} x_1, x_2\right](x - x_0)(x - x_1) + \cdots$$
$$+ g\left[x_j, \dots, x_{j+n}\right](x - x_0)(x - x_1) \dots (x - x_{n-1})$$

Podemos ver que en este paso hemos añadido un nuevo término que hace desaparecer todos los puntos de interpolación anteriores y hace que la función interpole por un nuevo punto. Los coeficientes serán constantes en el STENCIL en el que estemos trabajando. Si suponemos que $g_i = g(x_i)$ vienen de una función suave $f(x)$, tal que $g_i = f(x_i)$, entonces la diferencia dividida en este caso será:

$$g[x_j, \dots, x_{j+1}] = \frac{g(x_{j+1}) - g(x_j)}{x_{j+1} - x_j}$$

Que aproximará la derivada $f'(x)$. Por el mismo medio, si $x_j, \dots, x_{j+k}$ están lo suficientemente cerca, entonces tendremos que:

$$g[x_j, \dots, x_{j+1}] \approx \frac{1}{k!} f^{(k)}(x_j),$$

donde $f^{(k)}(x_j)$ es la K-derivada de $f(x)$, y k viene limitado por el orden del SPLINE. Para una función lo suficientemente suave f, es posible comprobar que

$$F[x_j, \dots, x_{j+k}] \approx \frac{1}{k!} f^{(k)}(\varepsilon), con\ k \leq n - 1$$

para todo $\mathcal{E}$ contenido en el intervalo (x_j, x_{j+k}). Esto será correcto cuando $k \leq n - 1$ veces diferenciable en dicho intervalo. Entonces, la forma de Newton quedará de una manera similar a la serie de Taylor:

$$g(x) = g(x_0) + g'(x_0)(x - x_0) + \frac{1}{2} g''(x_0)(x - x_0)^2 + \frac{1}{3!} g'''(x_0)(x - x_0)^3 + \cdots,$$

y reduce la serie de Taylor cuando $x_j \to x_0$ para todo j.

Ahora vamos a echar un vistazo al error de interpolación. Vamos a suponer que la función $f(x)$ que queremos aproximar a través del SPLINE $g(x)$, es una función suave. Evaluaremos la función en los puntos $f_i = f(x_i), i = 0,1, \dots, n\ para\ x_i \in [a, b]$. Podemos preguntarnos como de bien interpola nuestro SPLINE $g(\tilde{x})$ de orden n la función $f(x)$ si entre x_ny x_{n+1} hay una discontinuidad. Podemos expresar $p_n(x)$ en términos de un polinomio de orden inferior denominado $p_{n-1}(x)$ como:

$$p_n(x) = p_{n-1}(x) + F[x_0, \dots, x_{n-1}](x - x_0)(x - x_1) \dots (x - x_{n-1})$$

Usando la ecuación $g[x_j, \dots, x_{j+1}] \approx \frac{1}{k!} f^{(k)}(x_j)$, podemos obtener una fórmula para el error entre $p_n(x)$ y $p_{n-1}(x)$ que nos recuerda a la fórmula para series de Taylor:

$$p_n(\tilde{x}) - p_{n-1}(\tilde{x}) = \frac{1}{(n-1)!} F^{(n-1)}(\varepsilon)(x - x_0)(x - x_1) \dots (x - x_{n-1}),$$

donde ε es un punto contenido en el intervalo $(x_0, x_1, \dots, x_n) \in [a, b]$. Como sea el error dependerá de la distancia del punto *x* a los puntos de interpolación $x_0, \dots, x_n$ y como sea de pequeña la derivada $F^{(n-1)}(\varepsilon)$en el intervalo o lo que es lo mismo, como de suave es la función $F(x)$.

Hemos supuesto que $f(x)$ es continua por trozos, lo que implica que $(x_0, x_1, \dots, x_{n-1})$ que hemos usado en la ecuación de Newton anterior están a la izquierda de una hipotética discontinuidad y que x_n se sitúa al lado derecho de esta, por lo que $f(x)$ presenta una discontinuidad en el intervalo (x_{n-1}, x_n). En este caso, la fórmula del error será igual a la dada anteriormente, pero las diferencias divididas que aparecen aquí serán $O\left(\frac{h^l}{h^{n+1}}\right)$, donde *l* será el orden de la discontinuidad, *l=0* si se presenta una discontinuidad de salto, *l=1* si la discontinuidad de salto está en la primera derivada $f'(x)$, *l=2* si el salto se encuentra en la segunda derivada $f''(x)$ y así sucesivamente. Esto quiere decir que si el Stencil toca una discontinuidad de salto, el SPLINE solo tendrá $O(1)$ aproximaciones, ya que la discontinuidad de salto está contenida en la función. Si el Stencil toca una discontinuidad de esquina, el SPLINE solo será capaz de alcanzar $O(h)$aproximaciones, así como derivadas haya hasta que se dé el salto contenido en la función. Con esto y las ecuaciones anteriores vemos que la precisión de la aproximación obtenida por el SPLINE cuando el Stencil toca una discontinuidad está limitada a $O(h^l)$. Esto significa que aumentar el orden del SPLINE no mejorará la precisión. Localizando la discontinuidad en el panel que la contiene, podemos dividirlo en dos paneles, dividiendo el dominio en dos subdominios donde sabemos que la función que queremos reconstruir es continua.

6.7 Obtención de los valores F(x) y sus derivadas para discontinuidades en espaciados no uniformes

Ahora vamos a analizar la expresión de la función y sus derivadas cerca de la discontinuidad. Estas expresiones son difíciles de obtener, sin embargo, al querer introducir estos cálculos en nuestra programación de MATLAB, he creído necesario hacer una leve introducción a ellas. Estas ecuaciones demuestran que es posible trabajar directamente con intervalos sin tener que usar la primitiva de la función incluso para casos en los que los espaciados son no uniformes.

Empezaremos adaptando las ecuaciones para sistemas uniformes:

$$f_j^+ = f^+(X) - f_x^+(X)\alpha + \frac{1}{2}f_{xx}^+(X)\alpha^2 - \frac{1}{3!}f_{xxx}^+(X)\alpha^3$$

$$f_{j-1}^+ = f^+(X) - f_x^+(X)(h_{j-1}+\alpha) + \frac{1}{2}f_{xx}^+(X)(h_{j-1}+\alpha)^2 - \frac{1}{3!}f_{xxx}^+(X)(h_{j-1}+\alpha)^3$$

$$f_{j-2}^+ = f^+(X) - f_x^+(X)(h_{j-1}+h_{j-2}+\alpha) + \frac{1}{2}f_{xx}^+(X)(h_{j-1}+h_{j-2}+\alpha)^2 - \frac{1}{3!}f_{xxx}^+(X)(h_{j-1}+h_{j-2}+\alpha)^3$$

$$f_{j-3}^+ = f^+(X) - f_x^+(X)(h_{j-1}+h_{j-2}+h_{j-3}+\alpha) + \frac{1}{2}f_{xx}^+(X)(h_{j-1}+h_{j-2}+h_{j-3}+\alpha)^2 - \frac{1}{3!}f_{xxx}^+(X)(h_{j-1}+h_{j-2}+h_{j-3}+\alpha)^3$$

Y

$$f_{j+1}^+ = f^-(X) + f_x^-(X)(h_J-\alpha) + \frac{1}{2}f_{xx}^-(X)(h_J-\alpha)^2 - \frac{1}{3!}f_{xxx}^-(X)(h_J-\alpha)^3$$

$$f_{j+2}^+ = f^-(X) + f_x^-(X)(h_J+h_{J+1}-\alpha) + \frac{1}{2}f_{xx}^-(X)(h_J+h_{J+1}-\alpha)^2 - \frac{1}{3!}f_{xxx}^-(X)(h_J+h_{J+1}-\alpha)^3$$

$$f_{j+3}^+ = f^-(X) + f_x^-(X)(h_J+h_{J+1}+h_{J+2}-\alpha) + \frac{1}{2}f_{xx}^-(X)(h_J+h_{J+1}+h_{J+2}-\alpha)^2 - \frac{1}{3!}f_{xxx}^-(X)(h_J+h_{J+1}+h_{J+2}-\alpha)^3$$

$$f_{j+4}^+ = f^-(X) + f_x^-(X)(h_J+h_{J+1}+h_{J+2}+h_{J+3}-\alpha) + \frac{1}{2}f_{xx}^-(X)(h_J+h_{J+1}+h_{J+2}+h_{J+3}-\alpha)^2 - \frac{1}{3!}f_{xxx}^-(X)(h_J+h_{J+1}+h_{J+2}+h_{J+3}-\alpha)^3$$

Con esto, podemos obtener para puntos característicos:

$$\begin{pmatrix} f^+(X) \\ f_x^+(X) \\ f_{xx}^+(X) \\ f_{xxx}^+(X) \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} & C_{13} & C_{14} \\ C_{21} & C_{22} & C_{23} & C_{24} \\ C_{31} & C_{32} & C_{33} & C_{34} \\ C_{41} & C_{42} & C_{43} & C_{44} \end{pmatrix} \begin{pmatrix} f_j^+ \\ f_{j-1}^+ \\ f_{j-2}^+ \\ f_{j-3}^+ \end{pmatrix}$$

$$\begin{pmatrix} f^{-}(X) \\ f_{x}^{-}(X) \\ f_{xx}^{-}(X) \\ f_{xxx}^{-}(X) \end{pmatrix} = \begin{pmatrix} D_{11} & D_{12} & D_{13} & D_{14} \\ D_{21} & D_{22} & D_{23} & D_{24} \\ D_{31} & D_{32} & D_{33} & D_{34} \\ D_{41} & D_{42} & D_{43} & D_{44} \end{pmatrix} \begin{pmatrix} f_{j}^{-} \\ f_{j-1}^{-} \\ f_{j-2}^{-} \\ f_{j-3}^{-} \end{pmatrix}$$

Y para intervalos tendremos las siguientes ecuaciones:

$$\begin{pmatrix} F^{+}(X) \\ F_{x}^{+}(X) \\ F_{xx}^{+}(X) \\ F_{xxx}^{+}(X) \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} & C_{13} & C_{14} \\ C_{21} & C_{22} & C_{23} & C_{24} \\ C_{31} & C_{32} & C_{33} & C_{34} \\ C_{41} & C_{42} & C_{43} & C_{44} \end{pmatrix} \begin{pmatrix} \sum_{i=1}^{j} h_i \bar{f}_i^{+} \\ \sum_{i=1}^{j-1} h_i \bar{f}_i^{+} \\ \sum_{i=1}^{j-2} h_i \bar{f}_i^{+} \\ \sum_{i=1}^{j-3} h_i \bar{f}_i^{+} \end{pmatrix}$$

$$\begin{pmatrix} F^{-}(X) \\ F_{x}^{-}(X) \\ F_{xx}^{-}(X) \\ F_{xxx}^{-}(X) \end{pmatrix} = \begin{pmatrix} D_{11} & D_{12} & D_{13} & D_{14} \\ D_{21} & D_{22} & D_{23} & D_{24} \\ D_{31} & D_{32} & D_{33} & D_{34} \\ D_{41} & D_{42} & D_{43} & D_{44} \end{pmatrix} \begin{pmatrix} \sum_{i=1}^{j+1} h_i \bar{f}_i^{-} \\ \sum_{i=1}^{j+2} h_i \bar{f}_i^{-} \\ \sum_{i=1}^{j+3} h_i \bar{f}_i^{-} \\ \sum_{i=1}^{j+4} h_i \bar{f}_i^{-} \end{pmatrix}$$

Donde estas pueden ser obtenidas usando softwares matemáticos, como Maple, y solo dependerán de $h_i y\ \alpha$. Los distintos elementos de la matriz tomarán la siguiente forma:

$$C_{11} = \frac{1}{(h_{j-3}+h_{j-1}+h_{j-2})(h_{j-1}+h_{j-2})h_{j-1}} (h_{j-3}h_{j-1}^{2} + h_{j-3}h_{j-1}h_{j-2} + h_{j-1}^{3} + 2h_{j-1}{}^{2}h_{j-2} + h_{j-1}h_{j-2}{}^{2} + 2h_{j-3}h_{j-1}\alpha + h_{j-3}h_{j-2}\alpha + 3h_{j-1}{}^{2}\alpha + 4h_{j-1}h_{j-2}\alpha + h_{j-2}{}^{2}\alpha + 3h_{j-1}\alpha^{2} + 2h_{j-2}\alpha^{2} + h_{j-3}\alpha^{2} + \alpha^{3})$$

$$C_{12} = -\frac{\alpha(h_{j-3}h_{j-2} + h_{j-2}{}^{2} + h_{j-1}{}^{2} + 2h_{j-1}h_{j-2} + h_{j-3}h_{j-1} + 2h_{j-1}\alpha + 2h_{j-2}\alpha + h_{j-3}\alpha + \alpha^{2})}{(h_{j-3} + h_{j-2})h_{j-1}h_{j-2}}$$

$$C_{13} = \frac{\alpha(h_{j-1}{}^{2} + h_{j-3}h_{j-1} + h_{j-2}h_{j-1} + 2h_{j-1}\alpha + h_{j-3}\alpha + h_{j-2}\alpha + \alpha^{2})}{h_{j-2}(h_{j-1} + h_{j-2})h_{j-3}}$$

$$C_{14} = -\frac{\alpha({h_{j-1}}^2 + h_{j-1}h_{j-2} + 2h_{j-1}\alpha + h_{j-2}\alpha + \alpha^2)}{({h_{j-3}}^2 + h_{j-3}h_{j-1} + 2h_{j-3}h_{j-2} + h_{j-1}h_{j-2} + {h_{j-2}}^2)h_{j-3}}$$

$$C_{21} = \frac{2h_{j-3}h_{j-1} + h_{j-3}h_{j-2} + 3{h_{j-1}}^2 + 4h_{j-1}h_{j-2} + {h_{j-2}}^2 + 6h_{j-1}\alpha + 4h_{j-2}\alpha + 2h_{j-3}\alpha + 3\alpha^2}{\left(h_{j-3} + h_{j-1} + h_{j-2}\right)\left(h_{j-1} + h_{j-2}\right)h_{j-1}}$$

$$C_{22} = \frac{h_{j-3}h_{j-2} + {h_{j-2}}^2 + {h_{j-1}}^2 + 2h_{j-1}h_{j-2} + h_{j-3}h_{j-1} + 4h_{j-1}\alpha + 4h_{j-2}\alpha + 2h_{j-3}\alpha + 3\alpha^2}{(h_{j-3} + h_{j-2})h_{j-1}h_{j-2}}$$

$$C_{23} = \frac{{h_{j-1}}^2 + h_{j-3}h_{j-1} + h_{j-1}h_{j-2} + 4h_{j-1}\alpha + 2h_{j-3}\alpha + 2h_{j-2}\alpha + 3\alpha^2}{h_{j-2}(h_{j-1} + h_{j-2})h_{j-3}}$$

$$C_{24} = -\frac{{h_{j-1}}^2 + h_{j-1}h_{j-2} + 4h_{j-1}\alpha + 2h_{j-2}\alpha + 3\alpha^2}{({h_{j-3}}^2 + h_{j-3}h_{j-1} + 2h_{j-3}h_{j-2} + h_{j-1}h_{j-2} + {h_{j-2}}^2)h_{j-3}}$$

$$C_{31} = 2\frac{3h_{j-1} + 2h_{j-2} + h_{j-3} + 3\alpha}{(h_{j-3} + h_{j-2} + h_{j-1})(h_{j-1} + h_{j-2})h_{j-1}}$$

$$C_{32} = -2\frac{2h_{j-1} + 2h_{j-2} + h_{j-3} + 3\alpha}{(h_{j-3} + h_{j-2})h_{j-1}h_{j-2}}$$

$$C_{33} = 2\frac{2h_{j-1} + h_{j-2} + h_{j-3} + 3\alpha}{h_{j-2}(h_{j-1} + h_{j-2})h_{j-3}}$$

$$C_{34} = -2\frac{2h_{j-1} + h_{j-2} + 3\alpha}{\left({h_{j-3}}^2 + h_{j-3}h_{j-1} + 2h_{j-3}h_{j-2} + h_{j-1}h_{j-2} + {h_{j-2}}^2\right)h_{j-3}}$$

$$C_{41} = 6\frac{1}{(h_{j-3} + h_{j-2} + h_{j-1})(h_{j-1} + h_{j-2})h_{j-1}}$$

$$C_{42} = -6\frac{1}{(h_{j-3} + h_{j-2})h_{j-1}h_{j-2}}$$

$$C_{43} = 6\frac{1}{h_{j-2}(h_{j-1} + h_{j-2})h_{j-3}}$$

$$C_{34} = -6\frac{1}{\left({h_{j-3}}^2 + h_{j-3}h_{j-1} + 2h_{j-3}h_{j-2} + h_{j-1}h_{j-2} + {h_{j-2}}^2\right)h_{j-3}}$$

Vamos a realizarlo solo a uno de los lados del intervalo, ya que las fórmulas son muy largas pero parecidas para ambos lados y con uno podemos demostrar lo que

estamos buscando. Una vez realizadas las expresiones anteriores, podemos obtener las expresiones para trabajar en intervalos:

$$\bar{f}^{+}(X) = \frac{F^{+}(X) - F_j}{\alpha} = \frac{\bar{f}_j}{h_{j-1}h_{j-2}(h_{j-1}+h_{j-2})(h_{j-3}^{\,2} + h_{j-3}h_{j-1} + 2h_{j-3}h_{j-2} + h_{j-1}h_{j-2} + h_{j-2}^{\,2})h_{j-3}} * (h_{j-3}^{\,2}h_{j-2}^{\,2}$$
$$+ 2h_{j-3}^{\,2}h_{j-2}^{\,3}h_j + h_{j-3}h_{j-2}^{\,4}h_j + h_{j-3}^{\,3}h_{j-2}\alpha h_j + 3h_{j-3}^{\,2}h_{j-2}^{\,2}\alpha h_j + 2h_{j-3}h_{j-2}^{\,3}\alpha h_j + \alpha^2 h_j h_{j-3}^{\,2}h_{j-2}$$
$$+ \alpha^2 h_j h_{j-3}h_{j-2}^{\,2} + 6h_{j-3}^{\,2}h_{j-2}^{\,2}h_j h_{j-1} + 4h_{j-3}h_{j-2}^{\,2}h_j h_{j-1} + 3h_j h_{j-3}^{\,2}h_{j-1}^{\,2}h_{j-2} + 3h_j h_{j-2}^{\,2}h_{j-1}^{\,2}h_{j-3}$$
$$+ 3h_{j-3}^{\,2}\alpha h_j h_{j-1}h_{j-2} + 3\alpha h_j h_{j-3}h_{j-2}^{\,2}h_{j-1}$$
$$+ \frac{\bar{f}_{j-1}}{h_{j-1}h_{j-2}(h_{j-1}+h_{j-2})(h_{j-3}^{\,2} + h_{j-3}h_{j-1} + 2h_{j-3}h_{j-2} + h_{j-1}h_{j-2} + h_{j-2}^{\,2})h_{j-3}}$$
$$* (-h_{j-1}^{\,3}h_{j-3}^{\,3} - 2h_{j-1}^{\,4}h_{j-3}^{\,2} - h_{j-1}^{\,5}h_{j-3} - 3\alpha h_{j-1}^{\,3}h_{j-3}^{\,2} - 2\alpha h_{j-1}^{\,4}h_{j-3} - \alpha h_{j-1}^{\,2}h_{j-3}^{\,3} - \alpha^2 h_{j-1}^{\,2}h_{j-3}^{\,2}$$
$$- \alpha^2 h_{j-1}^{\,3}h_{j-3} - 3h_{j-1}^{\,3}h_{j-3}^{\,2}h_{j-2} - 4h_{j-3}h_{j-1}^{\,4}h_{j-2} - 3h_{j-3}h_{j-1}^{\,3}h_{j-2}^{\,2} - 3\alpha h_{j-1}^{\,2}h_{j-3}^{\,2}h_{j-2}$$
$$- 6\alpha h_{j-1}^{\,3}h_{j-3}h_{j-2} - 3\alpha h_{j-1}^{\,2}h_{j-3}h_{j-2}^{\,2} - 2\alpha^2 h_{j-1}^{\,2}h_{j-3}h_{j-2})$$
$$+ \frac{\bar{f}_{j-2}}{h_{j-1}h_{j-2}(h_{j-1}+h_{j-2})(h_{j-3}^{\,2} + h_{j-3}h_{j-1} + 2h_{j-3}h_{j-2} + h_{j-1}h_{j-2} + h_{j-2}^{\,2})h_{j-3}} * (h_{j-2}^{\,4}h_{j-1}^{\,2}$$
$$+ 2h_{j-2}^{\,3}h_{j-1}^{\,3} + \alpha h_{j-2}^{\,4}h_{j-1} + h_{j-1}^{\,4}h_{j-2}^{\,2} + 3\alpha h_{j-2}^{\,3}h_{j-1}^{\,2} + 2\alpha h_{j-2}^{\,2}h_{j-1}^{\,3} + \alpha^2 h_{j-2}^{\,3}h_{j-1} + h_{j-2}^{\,2}\alpha^2 h_{j-1}^{\,2})$$

$$F_x^{+}(X) = \frac{\bar{f}_{j-2}}{h_{j-1}h_{j-2}(h_{j-1}+h_{j-2})(h_{j-3}^{\,2} + h_{j-3}h_{j-1} + 2h_{j-3}h_{j-2} + h_{j-1}h_{j-2} + h_{j-2}^{\,2})h_{j-3}}$$
$$* (2h_{j-2}^{\,3}h_{j-1}^{\,3} + h_{j-2}^{\,4}h_{j-1}^{\,2} + h_{j-2}^{\,2}h_{j-1}^{\,4} + 2\alpha h_{j-2}^{\,4}h_{j-1} + 6\alpha h_{j-2}^{\,3}h_{j-1}^{\,2} + 4\alpha h_{j-2}^{\,2}h_{j-1}^{\,3} + 3\alpha^2 h_{j-2}^{\,3}h_{j-1}$$
$$+ 3h_{j-2}^{\,2}\alpha^2 h_{j-1}^{\,2}) + \frac{\bar{f}_{j-1}}{h_{j-1}h_{j-2}(h_{j-1}+h_{j-2})(h_{j-3}^{\,2} + h_{j-3}h_{j-1} + 2h_{j-3}h_{j-2} + h_{j-1}h_{j-2} + h_{j-2}^{\,2})h_{j-3}}$$
$$* (h_{j-1}^{\,3}h_{j-3}^{\,3} - 2h_{j-1}^{\,4}h_{j-3}^{\,2} - 4h_{j-3}h_{j-1}^{\,4}h_{j-2} - 3h_{j-3}h_{j-1}^{\,3}h_{j-2}^{\,2} - 3h_{j-1}^{\,3}h_{j-3}^{\,2}h_{j-2} - 6\alpha h_{j-1}^{\,2}h_{j-3}^{\,2}h_{j-2}$$
$$- 6\alpha h_{j-1}^{\,2}h_{j-3}^{\,2}h_{j-2} - 6\alpha h_{j-1}^{\,2}h_{j-2}^{\,2}h_{j-3} - 12\alpha h_{j-1}^{\,3}h_{j-3}h_{j-2} - h_{j-1}^{\,5}h_{j-3} - 6\alpha h_{j-1}^{\,3}h_{j-3}^{\,2} - 4\alpha h_{j-1}^{\,4}h_{j-3}$$
$$- 2\alpha h_{j-1}^{\,2}h_{j-3}^{\,3} - 3\alpha^2 h_{j-1}^{\,2}h_{j-3}^{\,2} - 6\alpha^2 h_{j-1}^{\,2}h_{j-3}h_{j-2} - 3\alpha^2 h_{j-1}^{\,3}h_{j-3})$$
$$+ \frac{\bar{f}_j}{h_{j-1}h_{j-2}(h_{j-1}+h_{j-2})(h_{j-3}^{\,2} + h_{j-3}h_{j-1} + 2h_{j-3}h_{j-2} + h_{j-1}h_{j-2} + h_{j-2}^{\,2})h_{j-3}} * (6h_{j-3}^{\,2}\alpha h_j h_{j-1}h_{j-2}$$
$$+ 6\alpha h_j h_{j-3}h_{j-2}^{\,2}h_{j-1} + h_{j-3}h_{j-2}^{\,4}h_j + h_{j-3}^{\,3}h_{j-2}^{\,2}h_j + 2h_{j-3}^{\,2}h_{j-2}^{\,3}h_j + 2h_{j-3}^{\,3}h_{j-2}h_j h_{j-1}$$
$$+ 4h_{j-3}h_{j-2}^{\,3}h_j h_{j-1} + 6h_{j-3}^{\,3}h_{j-2}^{\,2}h_j h_{j-1} + 6h_{j-3}^{\,3}h_{j-2}^{\,2}\alpha h_j + 2h_{j-3}^{\,3}h_{j-2}\alpha h_j + 4h_{j-3}h_{j-2}^{\,3}\alpha h_j$$
$$+ 3h_{j-3}^{\,2}h_{j-1}^{\,2}h_j h_{j-2} + 3h_{j-2}^{\,2}h_{j-1}^{\,2}h_j h_{j-3} + 3\alpha^2 h_{j-3}^{\,2}h_j h_{j-2} + 3\alpha^2 h_{j-2}^{\,2}h_j h_{j-3}$$

$$F_{xx}^{+}(X) = 2\frac{\bar{f}_{j-2}}{h_{j-1}h_{j-2}(h_{j-1}+h_{j-2})(h_{j-3}^{\,2} + h_{j-3}h_{j-1} + 2h_{j-3}h_{j-2} + h_{j-1}h_{j-2} + h_{j-2}^{\,2})h_{j-3}}$$
$$* (h_{j-2}^{\,4}h_{j-1} + 3h_{j-2}^{\,3}h_{j-1}^{\,2} + 3\alpha h_{j-2}^{\,3}h_{j-1} + 3h_{j-2}^{\,2}\alpha h_{j-1}^{\,2} + 2h_{j-2}^{\,2}h_{j-1}^{\,3})$$
$$+ 2\frac{\bar{f}_{j-1}}{h_{j-1}h_{j-2}(h_{j-1}+h_{j-2})(h_{j-3}^{\,2} + h_{j-3}h_{j-1} + 2h_{j-3}h_{j-2} + h_{j-1}h_{j-2} + h_{j-2}^{\,2})h_{j-3}}$$
$$* (-3h_{j-1}^{\,2}h_{j-3}h_{j-2}^{\,2} - 6h_{j-1}^{\,3}h_{j-3}h_{j-2} - 3h_{j-3}^{\,2}h_{j-1}^{\,2}h_{j-2} - 6\alpha h_{j-3}h_{j-1}^{\,2}h_{j-2} - 2h_{j-1}^{\,4}h_{j-3}$$
$$- 3\alpha h_{j-1}^{\,2}h_{j-3}^{\,2} - h_{j-1}^{\,2}h_{j-3}^{\,3} - 3\alpha h_{j-1}^{\,3}h_{j-3} - 3h_{j-1}^{\,3}h_{j-3}^{\,2})$$
$$+ 2\frac{\bar{f}_j}{h_{j-1}h_{j-2}(h_{j-1}+h_{j-2})(h_{j-3}^{\,2} + h_{j-3}h_{j-1} + 2h_{j-3}h_{j-2} + h_{j-1}h_{j-2} + h_{j-2}^{\,2})h_{j-3}} * (2h_j h_{j-3}h_{j-2}^{\,3}$$
$$+ 3h_j h_{j-3}^{\,2}h_{j-2}^{\,2} + 3h_j h_{j-3}h_{j-2}^{\,2}h_{j-1} + 3\alpha h_j h_{j-3}h_{j-2}^{\,2} + 3h_j h_{j-3}^{\,2}h_{j-2}h_{j-1} + h_j h_{j-3}^{\,3}h_{j-2}$$
$$+ 3\alpha h_j h_{j-2}h_{j-3}^{\,2})$$

$$F_{xxx}^{+}(X) = 6\frac{\left(h_{j-2}{}^{2}h_{j-1}{}^{2} + h_{j-2}{}^{3}h_{j-1}\right)\bar{f}_{j-2}}{h_{j-1}h_{j-2}\left(h_{j-1}+h_{j-2}\right)\left(h_{j-3}{}^{2} + h_{j-3}h_{j-1} + 2h_{j-3}h_{j-2} + h_{j-1}h_{j-2} + h_{j-2}{}^{2}\right)h_{j-3}}$$
$$+ 6\frac{\left(-2h_{j-3}h_{j-1}{}^{2}h_{j-2} - h_{j-3}h_{j-1}{}^{3} - h_{j-3}{}^{2}h_{j-1}{}^{2}\right)\bar{f}_{j-1}}{h_{j-1}h_{j-2}\left(h_{j-1}+h_{j-2}\right)\left(h_{j-3}{}^{2} + h_{j-3}h_{j-1} + 2h_{j-3}h_{j-2} + h_{j-1}h_{j-2} + h_{j-2}{}^{2}\right)h_{j-3}}$$
$$+ 6\frac{\left(h_{j}h_{j-3}h_{j-2}{}^{2} + h_{j}h_{j-2}h_{j-3}{}^{2}\right)\bar{f}_{j}}{h_{j-1}h_{j-2}\left(h_{j-1}+h_{j-2}\right)\left(h_{j-3}{}^{2} + h_{j-3}h_{j-1} + 2h_{j-3}h_{j-2} + h_{j-1}h_{j-2} + h_{j-2}{}^{2}\right)h_{j-3}}$$

Podemos comprobar solo con las ecuaciones a uno de los lados que estas expresiones son largas y complicadas pero aun así demuestran que es posible trabajar directamente con la función primitiva en intervalos que no sean uniformes.

Con todo esto, hemos hecho una definición de lo que es un Spline, sus usos y su forma tanto para intervalos uniformes como no uniformes, por ello a continuación vamos a mostrar su aplicación en un programa informático, en nuestro caso MATLAB, así como su programación y otros programas necesarios para poder realizar los Splines para discontinuidades de esquina.

6.8 Finalidad del uso de SPLINES a trozos.

Bien, una vez visto como realizar los cálculos para hacer los SPLINES a un lado y a otro de la discontinuidad de esquina, debemos decir que finalidad tiene el realizar los cálculos a ambos lados de dicha discontinuidad o el realizar un SPLINE que recorra toda nuestra función sin tener en cuenta dicha discontinuidad.

Inicialmente, podemos comparar el error que comete cada uno de ellos para una serie de funciones que presentan una discontinuidad de esquina, con el mismo número de datos y mallado. Así pues podremos comparar el error que cometen ambas aproximaciones y así podremos comprobar el porqué de realizar SPLINES a un lado y otro de la discontinuidad.

Para realizar estas comparaciones, introduciremos en nuestro programa, que definiremos y explicaremos posteriormente, una serie de funciones, con unos mallados creados para cada una de ellas mediante otro programa, y calcularemos las reconstrucciones mediante SPLINES de una manera (como una función completa) o

de otra (como una función a trozos en la cual la discontinuidad queda definida en uno de los intervalos y con la realización de la reconstrucción a un lado y otro de la esquina).

Pues bien, vamos a iniciar la comparación mediante una serie de representaciones gráficas de la misma función para cada uno de los casos que hemos dicho anteriormente, donde se mostrará la función escogida así como la reconstrucción realizada, cada una en una imagen, de los casos propuestos.

- 1^{er} ***CASO***

Para un mallado de 150 puntos, comparamos primero realizando la interpolación de la función como un "todo", es decir, el SPLINE se realiza en toda la función completa, no a trozos. Los SPLINES realizados a trozos serán mostrados en la gráfica siguiente. Una vez mostradas las imágenes, procederemos a comparar su error.

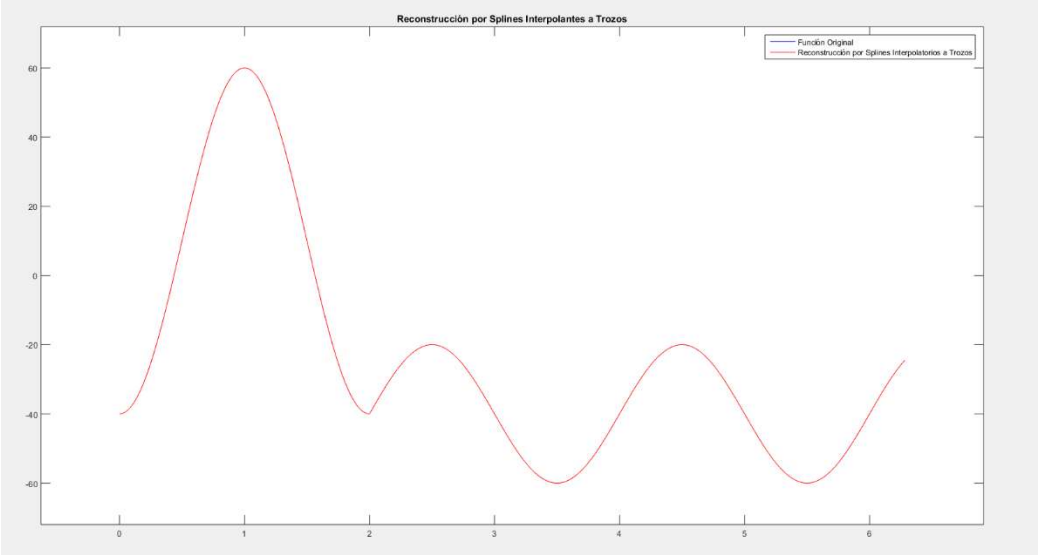


Fig.17 – Graficas SPLINES a trozos y Función superpuestas EJEMPLO 1

Vemos que la función original presenta una discontinuidad en la zona aproximada de X = 2, por lo que es ahí donde vamos a ver como se ha realizado la

reconstrucción:

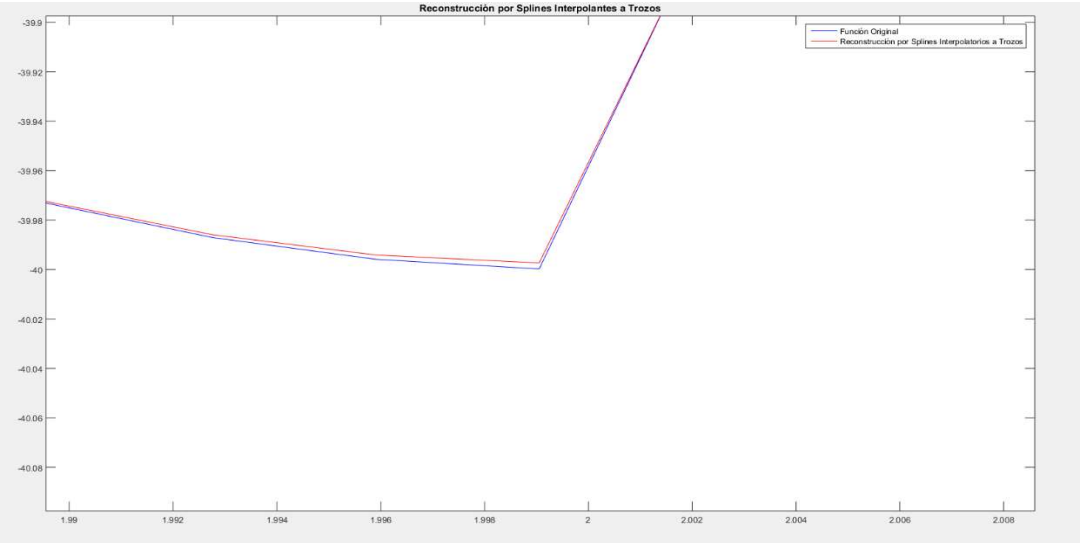


Fig.18 – Ampliación zona de esquina (N = 150)

Vemos pues, que el ajuste de la reconstrucción es muy pequeño para un mallado relativamente grande, por lo que como comprobaremos a continuación los errores que se irán cometiendo serán mucho menores que este.

- ***2º CASO***

Para este caso ampliaremos el número de puntos en el mallado a prácticamente el doble para comprobar así cuál de los errores se reduce más en función del número de puntos del mallado.

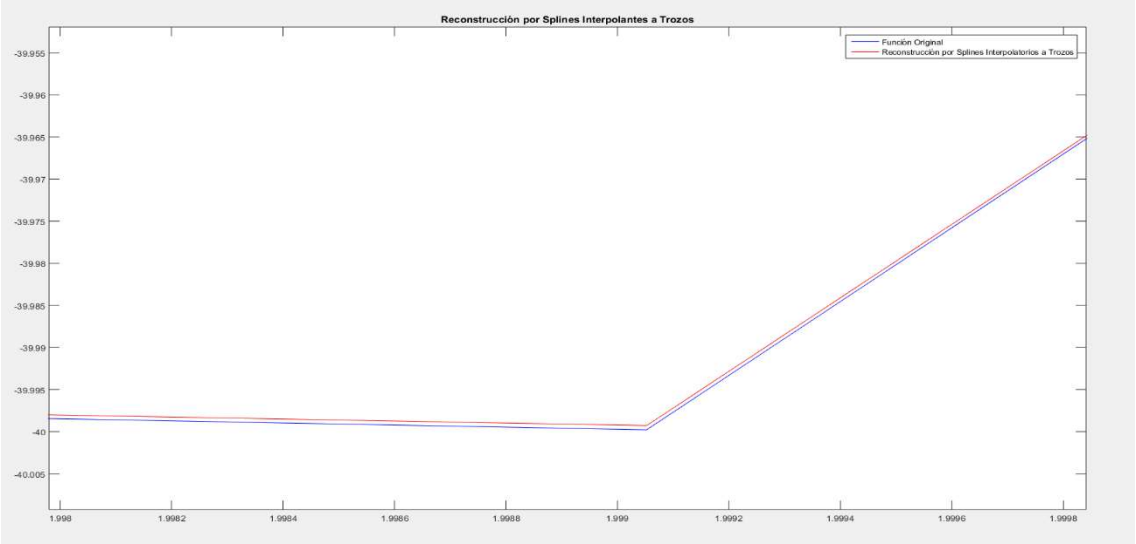


Fig.19 - Ampliación zona de esquina (N = 299)

Comprobamos que, para un mayor número de puntos, es decir, con un mallado más estrecho, el ajuste realizado mejora con respecto al anterior mostrado.

- **3^{er} CASO**

Volvemos a aumentar el número de puntos por intervalo a casi el doble para ver como el error cometido en cada uno de los casos es muy diferente y comprobar que el ajuste realizado mediante la realización de SPLINES a trozos es mucho mayor que el otro.

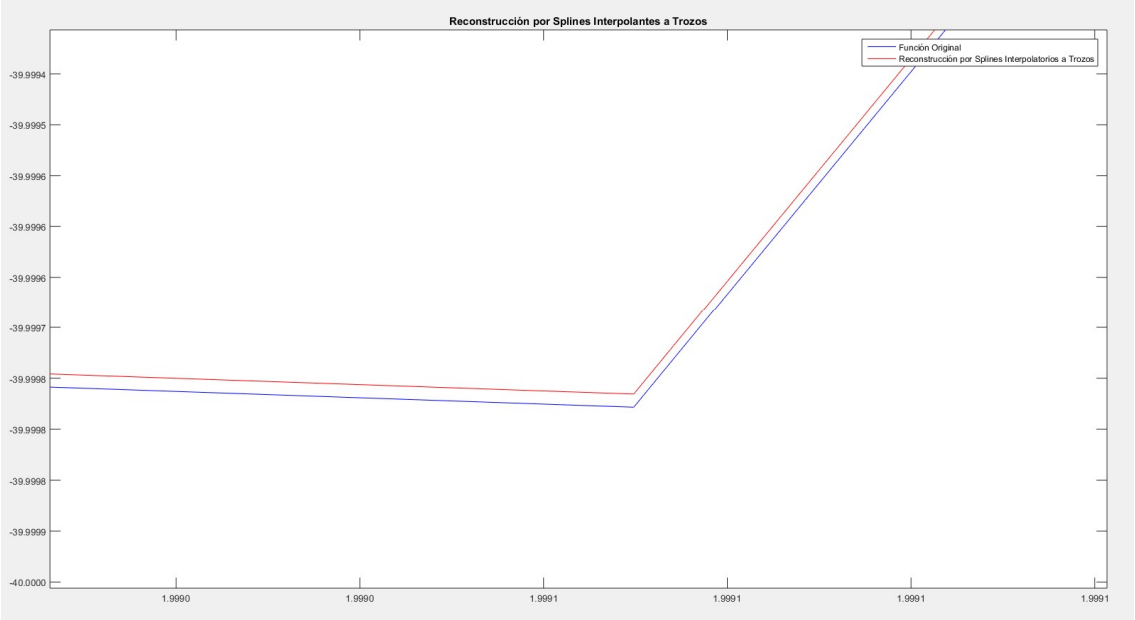


Fig.20 - Ampliación zona de esquina (N = 597)

Finalmente comprobamos aquí que para poder comprobar la diferencia entre ambas gráficas es necesario un zoom mucho mayor en la zona de la discontinuidad, lo que implica que el error cometido en este último caso es menor que el anterior, lo que nos lleva a deducir que, conforme aumente el número de puntos del mallado, aumentará así la precisión de la reconstrucción.

- ***4º CASO***

Vamos ahora a realizar el cálculo para otra función con más de una discontinuidad de esquina, para mostrar que es capaz de realizar los SPLINES cuando se tiene más de una discontinuidad.

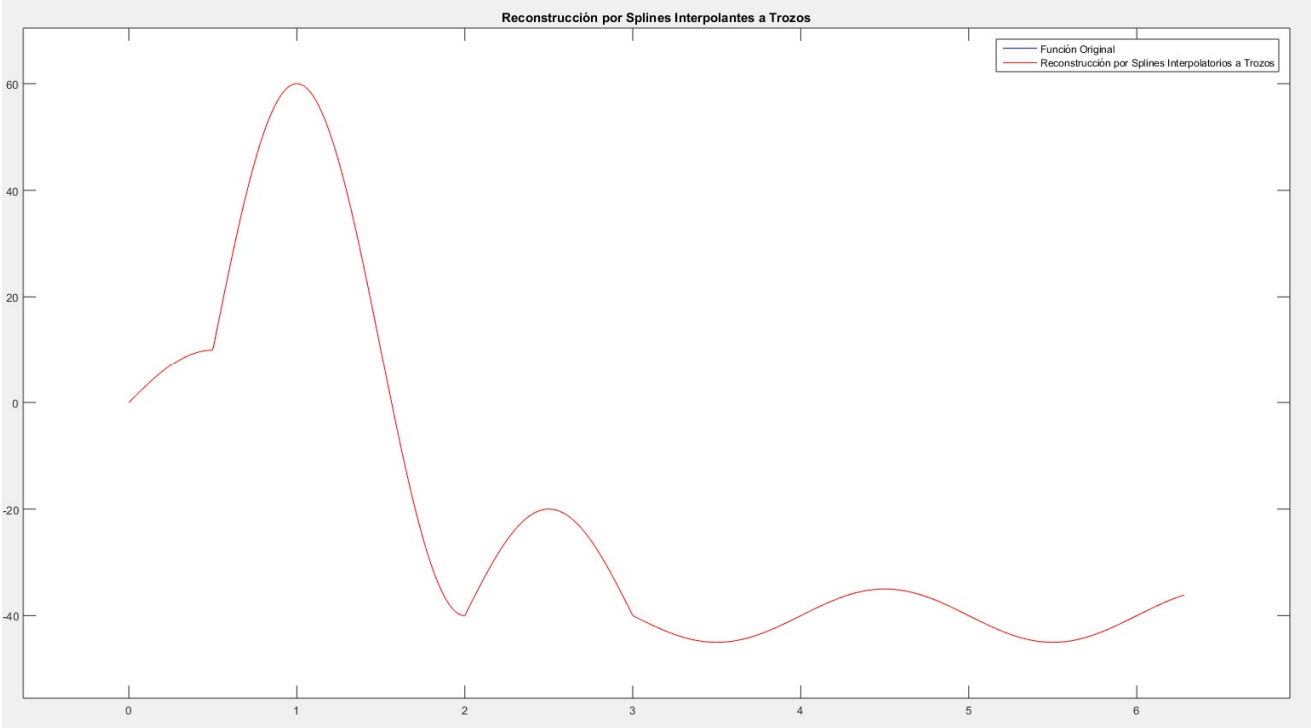


Fig.21 – Graficas SPLINES a trozos y Función superpuestas EJEMPLO 2

Vemos que ya para un mallado N = 150, no se aprecia ninguna diferencia entre la función original ni la gráfica construida mediante SPLINES a trozos, por lo que nuestro error será ya bastante bajo, para apreciar más de cerca la reconstrucción vamos a ampliar, como hemos hecho anteriormente, las zonas de las discontinuidades para poder ver cómo se va reduciendo el error cuando subdividimos el mallado.

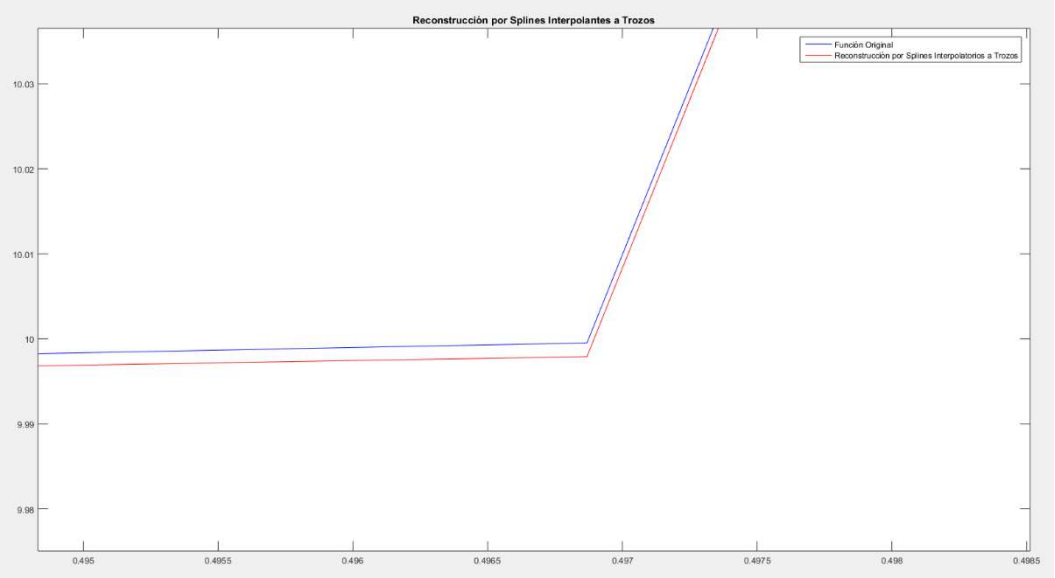


Fig.22 – Ampliación Primera Esquina (N=150)

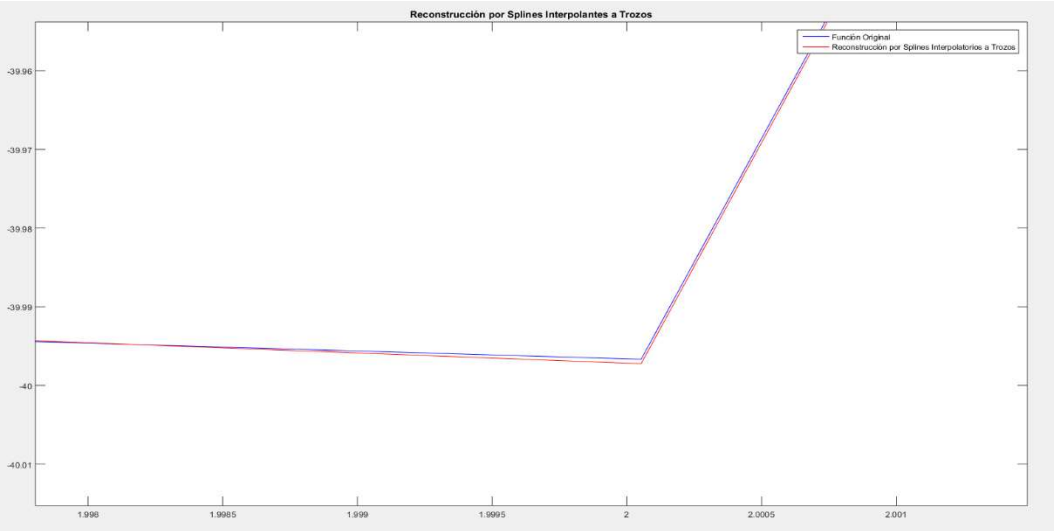


Fig.23 – Ampliación Segunda Esquina (N=150)

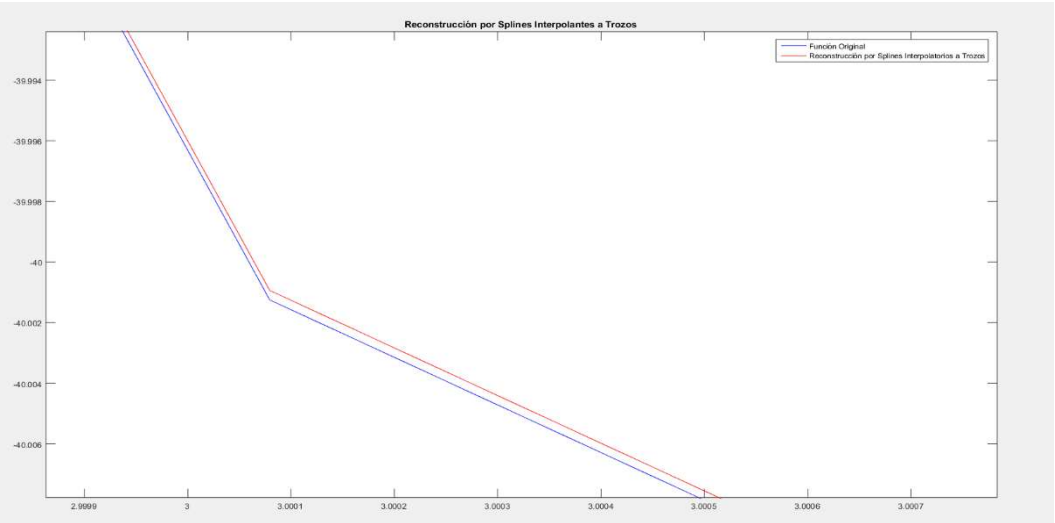


Fig.24 – Ampliación Tercera Esquina (N=150)

En estas tres gráficas vemos que ya el error es pequeño, pues bien, si aumentamos mucho el número de subdivisiones que realizamos al mallado, vamos a comprobar que el error que cometemos, como en el primer ejemplo, se reduce muchísimo. Vamos a hacer la comprobación para un mallado de N = 2385, para ver cómo se reduce el error drásticamente:

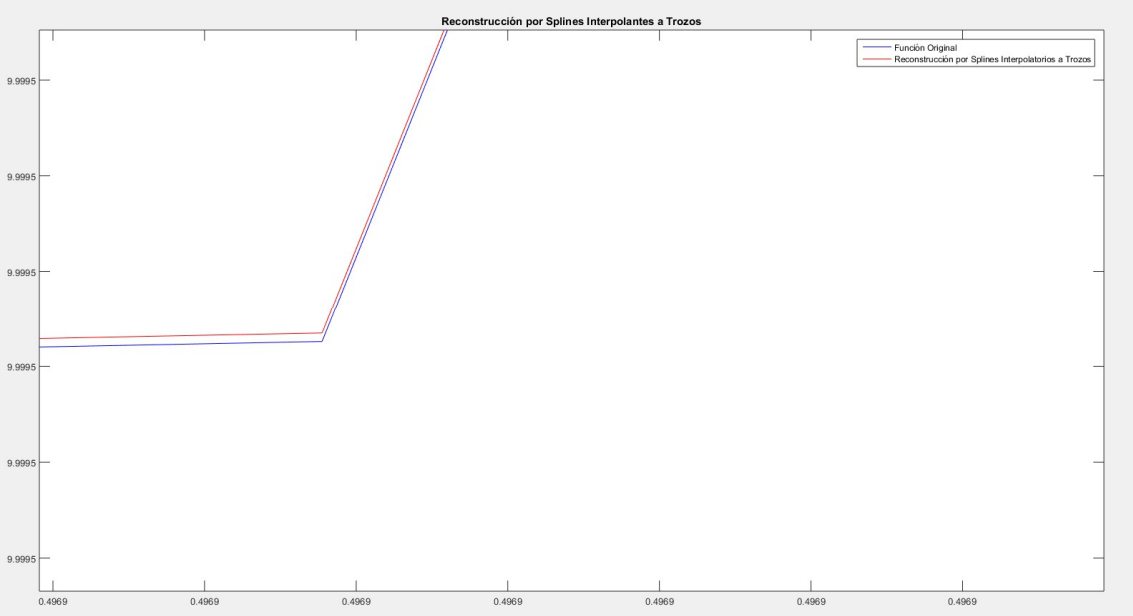


Fig.25 – Ampliación Primera Esquina (N = 2385)

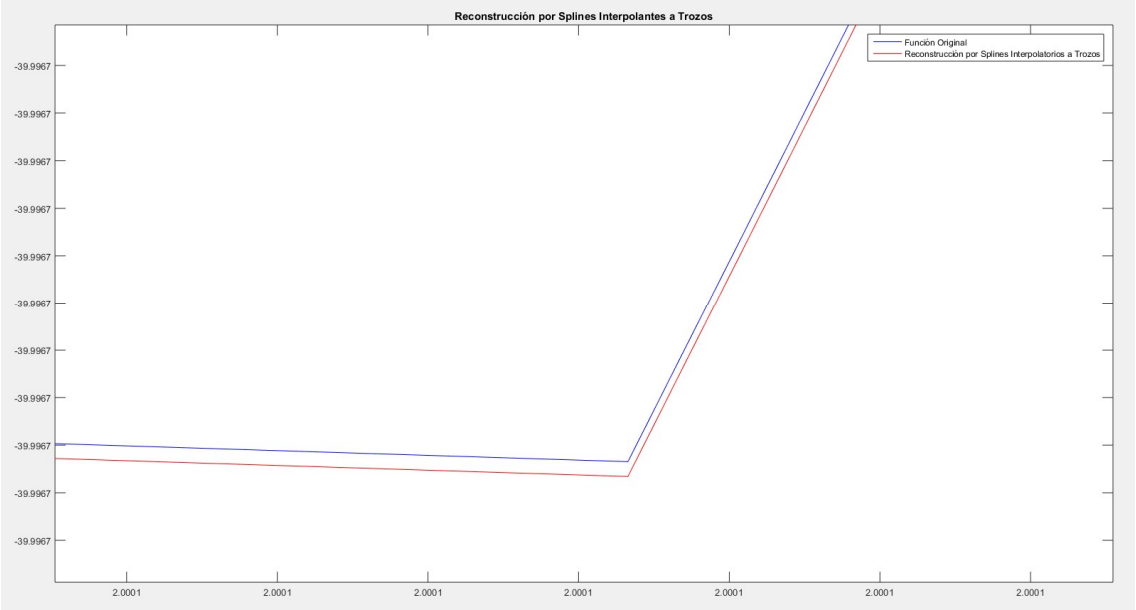


Fig.26 - Ampliación Segunda Esquina (N = 2385)

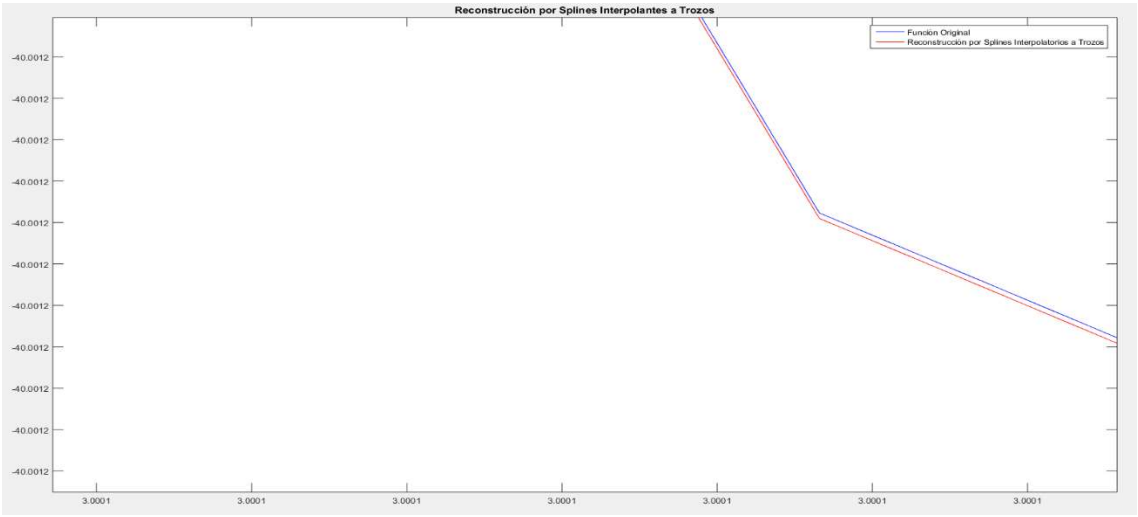


Fig.27 - Ampliación Tercera Esquina (N = 2385)

Podemos comprobar por el eje de Y que para poder ver la diferencia entre la función real y la función reconstruida es necesario ampliar la gráfica hasta valores mucho menos distantes, es decir, para puntos más unidos entre sí, lo que implica que el ajuste realizado es mejor por lo que se reduce, como se ha dicho anteriormente, drásticamente el error.

Ahora bien, hemos realizado la comparación con la gráfica de la función pero todavía podemos compararlo a la otra manera de realizar los SPLINES para una función, es decir podemos y vamos a comparar los SPLINES cúbicos para funciones completas con los SPLINES a trozos que estamos estudiando, analizando y realizando en este trabajo con nuestro programa.

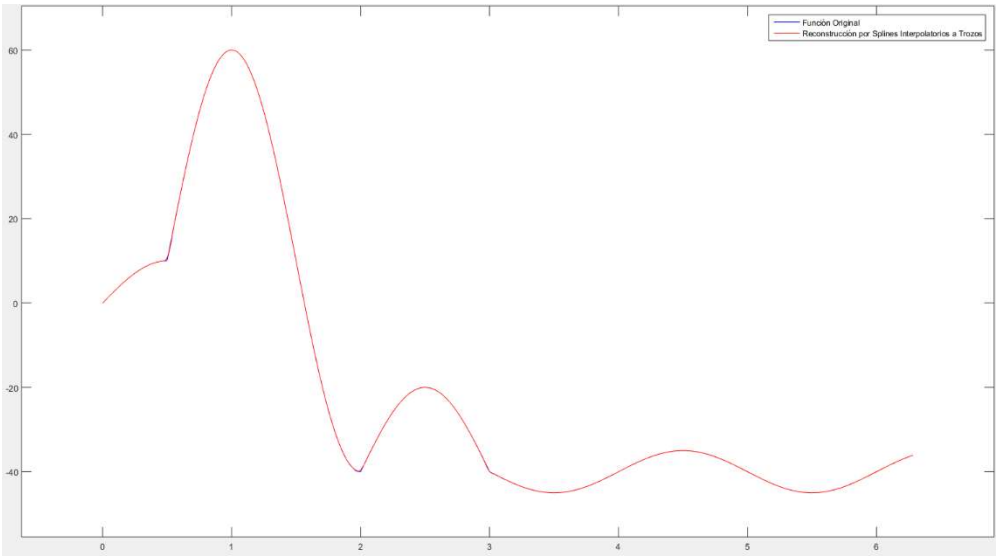


Fig.28 – Gráfica SPLINES y Función Superpuestas

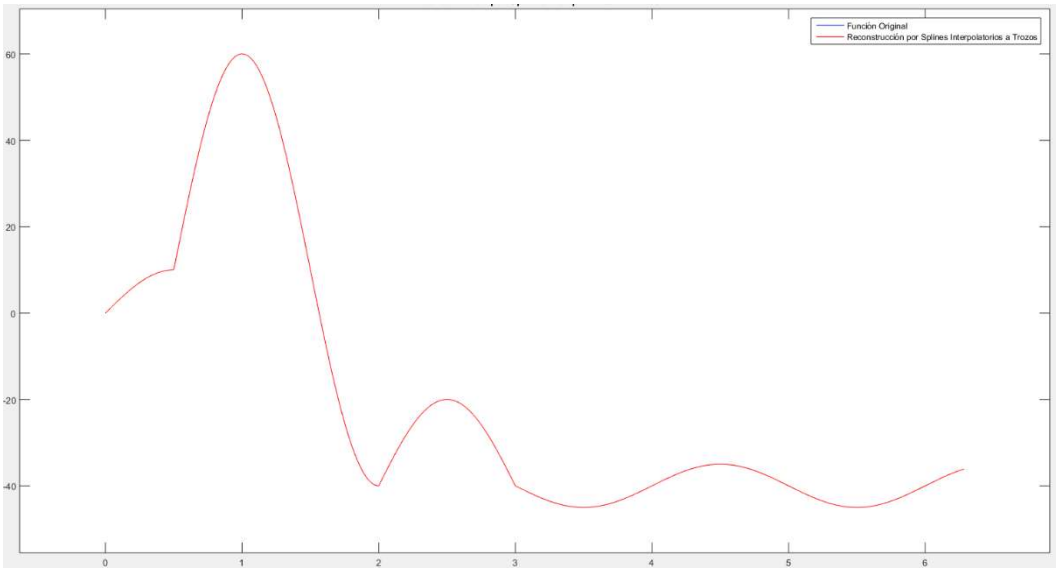


Fig.29 - Gráfica SPLINES a trozos y Función Superpuestas

A primera vista podemos comprobar que la superposición de la gráfica creada mediante SPLINES para toda la función no es tan buena como la gráfica de SPLINES realizados a trozos, sobre todo en las zonas de las discontinuidades de esquina, donde se comprueba a simple vista las diferentes gráficas en esos puntos. Aun así, vamos a ampliar las zonas donde se encuentran las discontinuidades, para así mostrarlo más claramente.

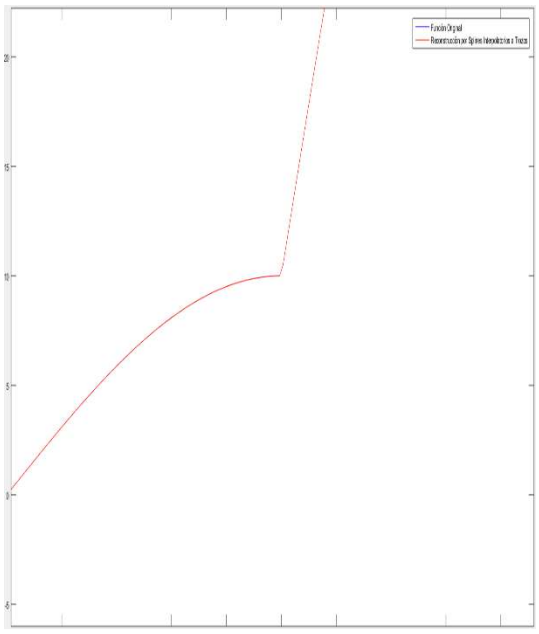


Fig.30 – Ampliación Primera Esquina a trozos

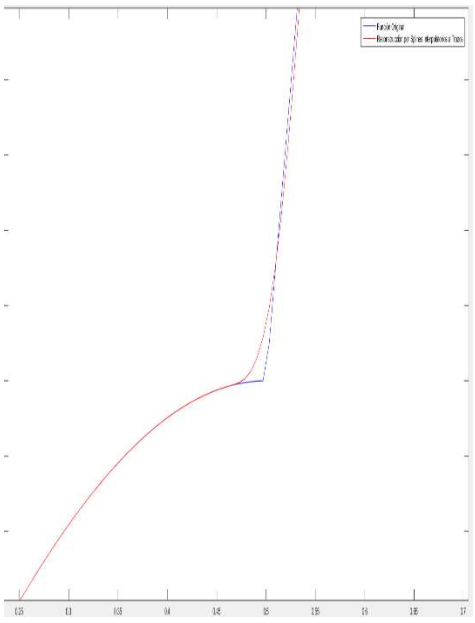


Fig.31 – Ampliación Primera Esquina Total

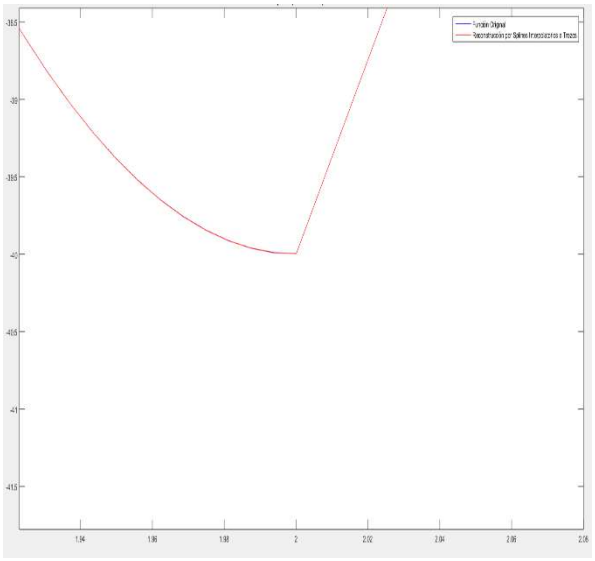


Fig.32 - Ampliación Segunda Esquina a trozos

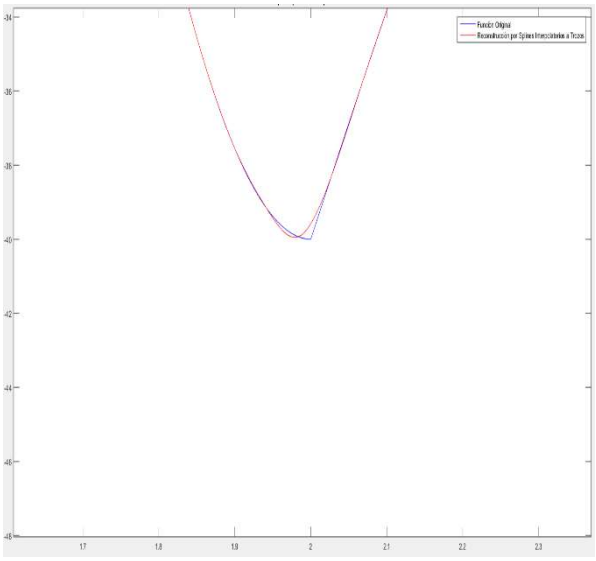


Fig.33 - Ampliación Segunda Esquina Total

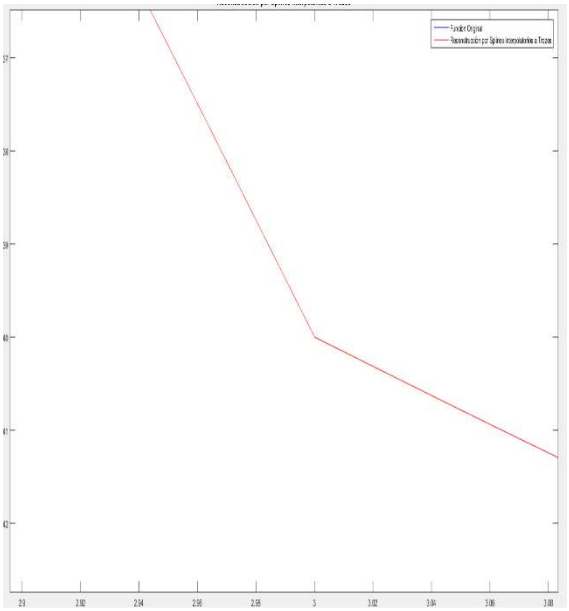


Fig.34 - Ampliación Tercera Esquina a trozos

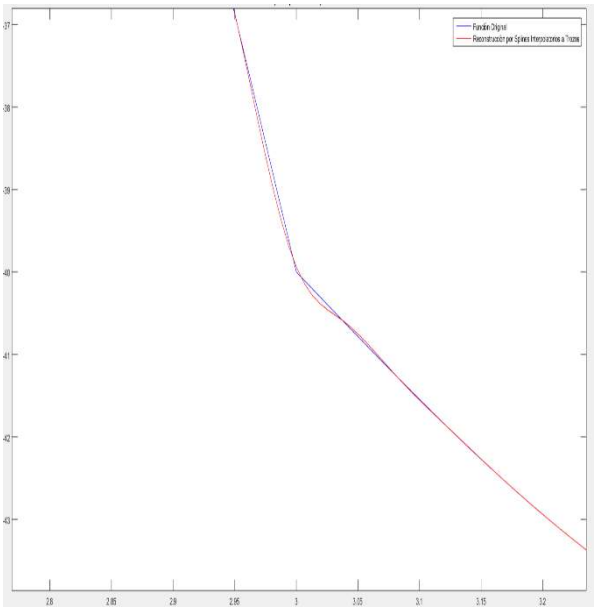


Fig.35 - Ampliación Tercera Esquina a Total

Podemos comprobar más fielmente mediante las imágenes ampliadas que la superposición de la gráfica de SPLINES a trozos es mucho más ajustada que la gráfica de los SPLINES para todo el intervalo. Esto se ve claramente en el error de ambas aproximaciones:

Error SPLINES a trozos = 0.0016

Error SPLINES para toda la función = 0.5739

Con esto concluimos por tanto que la aproximación realizada mediante SPLINES a trozos es más precisa que la realizada para toda la función, que es por lo que se está realizando este proyecto, para dar a conocer el uso de SPLINES a trozos en el mundo naval ya que generan un menor error de aproximación o realizan un mejor ajuste de la función.

6.9 Comprobación del Ajuste realizado

Una vez comprobado en el apartado anterior que el ajuste realizado mediante una reconstrucción por SPLINES a trozos a ambos lados de la discontinuidad es mucho más precisa que una simple reconstrucción mediante SPLINES en toda la función, tenemos que comprobar si nuestro ajuste está bien realizado para nuestro mallado.

Esto se realiza mediante la comprobación del error $Error_h = E_n$.

Supongamos $h_n = \frac{b-a}{n}$ como la distancia que hay entre los puntos de nuestros nodos. Se dice que el orden de aproximación numérico del algoritmo es *"p"* cuando:

$$E_n \approx Ch^p, donde\ C \in R$$

C será una constante.

Así pues, podremos comparar los errores obtenidos para cada división entre dos de nuestro mallado inicial, denominemos al error del mallado subdividido E_{2n}, por lo que tendremos el siguiente par de ecuaciones:

$$E_n \approx Ch_n^p$$

$$E_{2n} \approx C(\frac{h_n}{2})^p$$

Con esto, si comparamos los dos errores, por medio de la división del primero entre el segundo tendremos la siguiente forma:

$$\frac{E_n}{E_{2n}} = 2^p \rightarrow p_n = log_2(\frac{E_n}{E_{2n}})$$

Pues bien, una vez sabemos cómo conseguir nuestro orden de aproximación del algoritmo "*p",* podremos comprobar como es éste para la verificación de se está realizando un buen ajuste de la función cuando dividimos por la mitad el número de nodos. Puesto que nuestro SPLINE es cúbico, conforme aumentamos el número de divisiones de los intervalos, el valor de *p* debe de tender al orden del algoritmo real, que sería, para el caso de SPLINES cúbicos, de 4. Ahora bien, este valor de *p* cercano a 4 puede no encontrarse en la primera subdivisión, por lo que la precisión de esta no será la mejor, sin embargo, conforme reducimos a la mitad el tamaño del intervalo, el valor de *p* tenderá a 4.

Dicho de una manera más teórica, para medir numéricamente el orden *p* del algoritmo consideramos la sucesión de errores $E_n, E_{2n}, E_{4n}, \dots, E_{2^k n}$que nos arrojarán una sucesión de valores $p_n, p_{2n}, p_{4n}, \dots, p_{2^{k-1}n}$, que deberán converger en el orden teórico del algoritmo, es decir, en 4 ya que estamos usando SPLINES cúbicos.

Para ver esto, vamos a ver un ejemplo de una función que presenta varias discontinuidades de esquina las cuales vamos a analizar para un mallado tal que podamos ir subdividiendo, para así comprobar numéricamente lo explicado anteriormente.

- **1^{er} *MALLADO (N = 150)***

  Aquí hemos calculado el error para un mallado con 150 puntos para la función dada en el apartado anterior, el valor del error será:

  E1 = 0.0016

- ***2º MALLADO (N = 298)***

  El error para el segundo mallado será:

  E2 = 2.3656e-4

- 3^{er} ***MALLADO (N = 597)***

  El error para el tercer mallado, la segunda subdivisión de los intervalos será:

  E3 = 1.7718e-6

- ***4º MALLADO (N = 1193)***

  E4 = 5.2398e-8

- ***5º MALLADO (N = 2385)***

  E5 = 1.7808e-9

- ***6º MALLADO (N = 4769)***

  E6 = 5.1180e-11

- ***7º MALLADO (N = 9537)***

  E7 = 3.2330e-12

- ***8º MALLADO (N = 19073)***

  E8 = 1.7053e-13

Con esto, podemos pasar a calcular la relación de errores propuesta para comprobar así la veracidad del ajuste realizado. Como ya sabemos, esta relación debe de tender a 4 conforme subdividimos los intervalos por la mitad, aunque puede no cumplirse para mallados muy bajos.

- $log_2\left(\frac{E_1}{E_2}\right) = 2.7668$
- $log_2\left(\frac{E_2}{E_3}\right) = 7.0608$
- $log_2\left(\frac{E_3}{E_4}\right) = 5.0796$
- $log_2\left(\frac{E_4}{E_5}\right) =$*4.8789*
- $log_2\left(\frac{E_5}{E_6}\right) = 5.1208$
- $log_2\left(\frac{E_6}{E_7}\right) =$3.9847
- $log_2\left(\frac{E_7}{E_8}\right) =$4.2448

Vemos que, conforme aumentamos el número de subdivisiones en el mallado con el que analizamos la función, el error va acercándose más a 4 hasta que empieza a oscilar entre este con valores muy aproximados a él, lo que nos indica que el ajuste realizado es correcto.

Capítulo VII
MATLAB para SPLINES

Capítulo VII

Programación de SPLINES en MATLAB

Para el análisis mediante ordenador de los SPLINES y para su cálculo, se ha escogido un programa de programación matemática llamado MATLAB, cuyo nombre viene de "MATrix LABoratory" o lo que es lo mismo, laboratorio de matrices. Este programa ofrece un entorno de desarrollo integrado y un lenguaje de programación propio que nos facilitará la creación de nuestro código para los SPLINES.

Tiene diversas funciones básicas, entre las que cabe destacar: representación de datos y funciones, manipulación de matrices, la implementación de algoritmos, la creación de interfaces de usuario (GUI), y la comunicación con programas en otros lenguajes y con otros dispositivos hardware.

La programación en MATLAB es la escogida debido a lo dicho anteriormente ya que, en su lenguaje de programación las funciones matemáticas suelen estar incluidas (ya sea, por ejemplo, la función seno, coseno, derivadas, integrales…) lo que nos facilita la programación debido a que no es necesario programar también las funciones que se vayan a utilizar sino los comandos que las ligan y que queremos analizar. Esto implica que para la programación y la utilización de este programa se deben tener conocimientos en este entorno, tanto de su lenguaje como de su modo de programación, es decir, de sus comandos y de la manera en la que debemos introducirlos para que cumplan la función que nosotros esperamos. Para conocer más estos comandos y como utilizarlos podemos ver toda la información que hay acerca del programa en sus manuales, habiendo para usuarios que se inician en el mundo de la programación por MATLAB tanto así como para usuarios más avanzados que requieran

de herramientas más específicas.

Otra de las funciones que ofrece MATLAB y que es muy importante, es la capacidad de reproducir gráficamente, tanto en 2D como en 3D, las funciones y otros elementos precisan de una representación gráfica, ya sea para aclarar o para completar la definición del elemento así como para ver su forma y comprobar que coincide con nuestra idea de resolución gráfica del problema. También es capaz de crear una interfaz gráfica para poder realizar los cálculos y órdenes sin necesidad de ver las órdenes, sino solo con una simple introducción de datos en dicha interfaz para que comience a compilar y a realizar el programa.

Para el tema que nos atañe, los SPLINES, nos interesa sobre todo el control de matrices para la introducción de datos en una o varias dimensiones así como la representación gráfica del SPLINE que queremos calcular. Para ello, utilizaremos una serie de comandos llamados SCRIPTS.

7.1- SCRIPTS

Los SCRIPTS es un conjunto de órdenes guardadas en un archivo de texto, generalmente muy ligero y, que es ejecutado por lotes o línea a línea, en tiempo real por un intérprete que será el programa que vayamos a utilizar, en nuestro caso, el MATLAB. Para el tema que nos atañe, los SPLINES, nos interesa sobre todo el control de matrices para la introducción de datos en una o varias dimensiones así como la representación gráfica del SPLINE que queremos calcular.

Para la realización del programa que queremos usar para SPLINES es necesaria una serie de SCRIPTS que digan al programa las diferentes operaciones que se tienen que realizar para su cálculo así como los datos que vamos a utilizar para realizar dicho cálculo. Estos datos suelen darse en un archivo de texto .txt en el cual también se dará la solución del problema.

7.2- Datos de entrada y salida

Una vez realizado el programa, es decir, introducidos los comandos necesarios para que MATLAB calcule nuestro SPLINE o serie de estos, es necesario introducir una serie de datos para que estos cálculos se inicien. Esta serie de datos deberá contener los valores de nuestra función a analizar, en nuestro caso serán dos funciones, una formada por la forma de la cuaderna en estribor y la otra la formará esa misma cuaderna por babor, para así formar la quilla en la unión de ambas.

7.3- Programación necesaria para el Cálculo de SPLINES

Una vez conocidos ligeramente los diferentes datos iniciales que se usan en MATLAB (entradas y salidas) así como una breve descripción del programa, conviene explicar que programas hemos tenido que realizar y usar para nuestro caso de los Splines, ya que para este trabajo se usan varios programas realizados en MATLAB aparte del cálculo de los Splines.

Podremos comprobar que para introducir lo que queremos hacer en MATLAB, no solo es necesario el uso de SCRIPTS, sino también el uso de un lenguaje de programación propio del programa. Este lenguaje tiene una grandísima cantidad de variables dependiendo de las operaciones que queramos programar.

Los distintos programas que vamos a utilizar son:

1- **GET DERIVATIVES:** este programa realiza las derivadas en la aproximación de la discontinuidad a ambos lados de esta (derecha e izquierda). El programa será de la siguiente forma:

```
function df=get_derivatives(x,fx,d)

% Esta función obtiene las derivadas por la izquierda o por la derecha en la aproximación
% calculada a la discontinuidad
%
% df=get_derivatives(x,fx,d)
%
% Variables de entrada:
%
% x abscisas donde se plantean los desarrollos de Taylor
%   [x_j,x_{j-1}, ...,x_{j-N-1}]  si es por el lado izquierdo
%   [x_{j+1},x_{j+2},...,x_{j+N}] si es por el lado derecho
% fx valores conocidos de la función en las abscisas x
% d aproximación calculada a la discontinuidad
%
% Variables de salida:
%
% df aproximación de orden N a las derivadas por un lado en la
%    discontinuidad


% Orden de aproximación
N=length(x);

% x-x*

c=x-d;

% Matriz de Vandermonde

A=fliplr(vander(c));

for i=2:N
    A(:,i)=A(:,i)/factorial(i-1);
end

% Vector de terminos independientes
b=fx';


% Resolvemos el sistema lineal

df=A\b;
```

Para MATLAB, todo lo que tenga en su principio el símbolo % será reconocido como una aclaración para los usuarios del programa, es decir, no son líneas de programación propiamente dicha, sino de aclaración y no interfieren con la programación del programa.

2- **LAGRANGE:** este programa va a realizar la interpolación de Lagrange para un conjunto de puntos dado, y tendrá la siguiente forma:

```
function y=lagrange(f,nod,x)

% este programa calcula la interpolacion de lagrange con n puntos
% function y=lagrange(f,nod,x)
% f valores de la funcion en los nodos
% nod nodos para interpolar
% x vector de abcisas donde evaluar el polinomio interpolador

n=length(f);
m=length(x);
b=zeros(n,m);
for i=1:n
   b(i,1:m)=ones(1,m);
   for j=1:n
       if(j~=i)
       b(i,1:m)=b(i,1:m).*(x-nod(j)*ones(1,m))/(nod(i)-nod(j));
       end
   end
end

y=zeros(1,length(x));
for i=1:n
    y(1:m)=y(1:m)+b(i,1:m)*f(i);
end
return


```

3- **MU:** con este programa compararemos los distintos polinomios de Lagrange hallados anteriormente hallando las raíces de la función que diferencia los polinomios de Lagrange a comparar.

```
function [raiz]=Mu(nodosI,yI,nodosD,yD)

% Esta función aproxima la raíz de la función diferencia de los
% dos polinomios de Lagrange construidos a partir de los datos
% (nodosI,yI) y (nodosD,yD) respectivamente.
%
% [raiz]=Mu(nodosI,yI,nodosD,yD)
%
```

```
%
% Variables de entrada:
% nodosI valores de las abscisas para construir el polinomio de Lagrange
%        de orden N de la izquierda
% yI valores de la función en los nodosI
% nodosD valores de las abscisas para construir el polinomio de Lagrange
%        de orden N de la derecha
% yD valores de la función en los nodosD
%
% Variables de salida:
% raiz aproximación obtenida de la raíz por el método de Bisección



 tol=10^(-15);
 tolf=10^(-15);


%Condiciones de convergencia.
%1.f(a)*f(b)<0

n=length(nodosI);


a=nodosI(n);
b=nodosD(1);

pa=yI(n);
qb=yD(1);







fa=pa-lagrange(yD,nodosD,a);
fb=lagrange(yI,nodosI,b)-qb;




if abs(fa)<tolf
    raiz=a;
    return;
end

if abs(fb)<tolf
    raiz=b;
    return;
end

if fa*fb>0
    error ('La raiz no se encuentra en el intervalo(a,b)')
end



%Comenzamos las iteraciones

n=ceil(log(abs(b-a)/tol)/log(2.0));
err=(b-a)/(2^(n));


for i=1:n

    c=0.5*(a+b);
    fc=lagrange(yI,nodosI,c)-lagrange(yD,nodosD,c);


    if abs(fc)<tolf
        raiz=c;
        return
    else
        if fb*fc<0
```

```
                a=c;
                fa=fc;
            else
                b=c;
                fb=fc;
            end
        end

    end

    raiz=(a+b)/2;
```

4- **BB:** sirve para localizar los intervalos que puedan contener una discontinuidad.

```
function candidatos=BB(d2,N)

% Este programa marca la posición de los posibles candidatos a contener una
% discontinuidad
% candidatos=BB(d2,N)
% Variables de entrada:
% d2 vector que contiene las diferencias segundas de los datos
% N indica el orden N+1 de aproximación requerido
% Variables de salida:
% Candidatos vector que contiene en cada posición:
%                       0 si ese intervalo no es candidato
%                       1 si ese intervalo contiene una
%                         discontinuidad justo en el nodo inicial
%                       2 si ese intervalor contiene una
%                         discontinuidad intermedia



n=length(d2);
candidatos=zeros(size(d2));
peso=1.5;


for j=N+2:n-N-1

    M=max(abs([d2(j-N-1:j-1),d2(j+1:j+N+1)]));

    if abs(d2(j))>peso*M
        candidatos(j)=1;
    end

    Mi=max(abs(d2(j-N-1:j-1)));

    Md=max(abs(d2(j+2:j+N+1)));

    if abs(d2(j+1))>peso*Md & abs(d2(j))>peso*Mi
        candidatos(j)=2;
    end

end
```

5- SPLINES INTERPOLANTES: este es nuestro segundo programa más importante, no con ello se puede decir que el programa final pudiera funcionar sin los anteriores, sino que este programa es el que realiza los cálculos para hallar las ecuaciones de los SPLINES, además de mostrar las gráficas, primera y segunda derivada y valores obtenidos al interpolar:

```
function [Si]=Splines_Interpolantes(t,y,CI,S1a,S2a,CD,S1b,S2b,opx,x,opg1,opg2,opg3,opg4)


% Esta función busca hallar las ecuaciones que definen el spline y dibujar
% las gráficas de la curva, su primera y segunda derivada, asi como los
% valores obtenidos al interpolar y su gráfica. Creando ficheros que
% recojan la información.
%
% Splines_Interpolantes(t,y,CI,S1a,S2a,CD,S1b,S2b,opx,x,opg1,opg2,opg3,opg4)
%
% Variables de entrada:
% t:abscisas de la tabla de valores de los nodos
% y:ordenadas de la tabla de valores de los nodos
% CI:condición de contorno en el extremo izquierdo
% CD:condición de contorno en  el extremo derecho
%
% Posibles condiciones de contorno:
% 1- De primer tipo: valores de la primera derivada
% 2- De segundo tipo: valores de la segunda derivada
% 3- De tercer tipo: periódicas de periodo T=b-a
%    S'(a)=S'(b); S''(a)=S''(b). Si CI=3->CD=3
% 4- De cuarto tipo: Tres veces diferenciable en los puntos t2 y/o tm-1
%
% S1a:valor de la primera derivada en el extremo izquierdo
% S2a:valor de la segunda derivada en el extremo izquierdo
% S1b:valor de la primera derivada en el extremo derecho
% S2b:valor de la segunda derivada en el extremo derecho
%
% opx:parámetro que nos indica si debemos calcular o no los valores interpolados
% 's':se calculan; 'n': no se calculan
% x:retículo de puntos donde queremos evaluar la función polinómica a trozos
% si opgI=1 entonces dibujará la gráfica
% opg1:parámetro que nos indica se debemos dibujar o no la gráfica de los valoes interpolados
% opg2:parámetro que nos indica se debemos dibujar o no la gráfica de los splines
% opg3:parámetro que nos indica se debemos dibujar o no la gráfica de las primeras derivadas
% opg4:parámetro que nos indica se debemos dibujar o no la gráfica de las segundas derivadas
%
%EJEMPLO:
%t=[1,3,4,6]
%y=[2,-1,3,1]
%CI=2 ; CD=1
%S1a=0; S2b=0 (no se van a utilizar, da igual el dato)
%S2a=2; S1b=2
```

```
%opx='s'
%x=[0:0.2:6]
%
%Splines_Interpolantes([0,3,4,6],[2,-1,3,1],2,0,2,1,2,0,'s',[0:0.2:6],1,1,1,1)




% Número de puntos en la tabla
m=length(t);

% Comprobamos que las entradas son correctas

if m<3
    uiwait(msgbox('El número de puntos de control debe ser mayor que 2', 'Mensaje de error',...
        'error','modal'));
    return;
elseif CI==0 | CD==0
    uiwait(msgbox('Falta especificar alguna de las condiciones de frontera', 'Mensaje de error',...
        'error','modal'));
    return;
elseif CI==3 & CD~=3
    uiwait(msgbox('Si CI=3 entonces CD debe ser también 3', 'Mensaje de error',...
        'error','modal'));
    return;
elseif CD==3 & CI~=3
    uiwait(msgbox('Si CD=3 entonces CI debe ser también 3', 'Mensaje de error',...
        'error','modal'));
    return;
elseif CD==3 & y(1)~=y(m)
    uiwait(msgbox('Para que la función sea periódica debe valer lo mismo en los extremos', 'Mensaje de error',...
        'error','modal'));
    return;
elseif m==3 & CI==4 & CD==4
    uiwait(msgbox('Con 3 nodos la condición CI=4 ya implica CD=4 y nos falta una condición', 'Mensaje de error',...
        'error','modal'));
    return;
end


% Definimos el vector h de distancias entre las abscisas
h=diff(t,1);
```

```
% Inicializamos
z=zeros(1,m);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Cálculo de las derivadas segundas
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Definimos la matriz de coeficientes del sistema lineal
A=zeros(m,m);
% Definimos el término independiente
B=zeros(m,1);



% Definimos la primera ecuación del sistema dependiendo de CI

if CI==1
    A(1,1)=-h(1)/3;
    A(1,2)=-h(1)/6;
    B(1)=S1a+(y(1)-y(2))/h(1);
elseif CI==2
    B(1)=S2a;
    A(1,1)=1;
elseif CI==3
    % primera ecuación
    A(1,1)=h(1)/3; A(1,2)=h(1)/6; A(1,m-1)=h(m-1)/6; A(1,m)=h(m-1)/3;
    B(1)=(y(m-1)-y(m))/h(m-1)+(y(2)-y(1))/h(1);
    % última ecuación
    A(m,1)=1; A(m,m)=-1;
elseif CI==4
    A(1,1)=-h(2); A(1,2)=h(1)+h(2); A(1,3)=-h(1);
end


% Definimos las m-2 ecuaciones intermedias


for i=2:m-1
    A(i,i-1)=h(i-1);
    A(i,i)=2*(h(i-1)+h(i));
    A(i,i+1)=h(i);
```

```
130 -   end
131
132     % A continuación definimos la matriz B
133
134 -   for i=2:m-1
135 -       B(i)=6/h(i)*(y(i+1)-y(i))-6/h(i-1)*(y(i)-y(i-1));
136 -   end
137
138
139
140     % Definimos la última ecuación del sistema dependiendo de CD
141
142 -   if CD==1
143 -       A(m,m-1)=h(m-1)/6; A(m,m)=h(m-1)/3;
144 -       B(m)=S1b+(y(m-1)-y(m))/h(m-1);
145 -   elseif CD==2
146 -       B(m)=S2b;
147 -       A(m,m)=1;
148 -   elseif CD==4
149 -       A(m,m-2)=-h(m-1); A(m,m-1)=h(m-1)+h(m-2); A(m,m)=-h(m-2);
150 -   end
151
152
153     % Resolución del sistema
154
155 -   z(1:m)=A\B;
156
157
158
159     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
160     % Evaluación de los splines en x
161     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
162 -   if opx=='s'
163
164         % Encuentra el trozo polinómico a evaluar según cada entrada de x
165 -       if x(1)==t(1)
166 -           minim=y(1); maxim=y(1);
167 -           Si(1)=y(1);
168 -       else
169 -           ind=find((x(1)>t)==0);
170 -           ind=ind(1)-1;
171 -           ind=ind(1);
172             % Aplica la fórmula para calcular S_ind(x)
173
174 -           Si(1)=z(ind+1)/(6*h(ind))*(x(1)-t(ind))^3+...
175                 z(ind)/(6*h(ind))*(t(ind+1)-x(1))^3+(y(ind+1)/h(ind)-...
176                 z(ind+1)*h(ind)/6)*(x(1)-t(ind))+(y(ind)/h(ind)-...
177                 z(ind)*h(ind)/6)*(t(ind+1)-x(1));
178 -           minim=Si; maxim=Si;
179 -       end
180
181 -       for i=2:length(x)
182 -           ind=find((x(i)>t)==0);
183 -           ind=ind(1)-1;
184 -           ind=ind(1);
185             % Aplica la fórmula para calcular S_ind(x)
186 -           Si(i)=z(ind+1)/(6*h(ind))*(x(i)-t(ind))^3+...
187                 z(ind)/(6*h(ind))*(t(ind+1)-x(i))^3+(y(ind+1)/h(ind)-...
188                 z(ind+1)*h(ind)/6)*(x(i)-t(ind))+(y(ind)/h(ind)-...
189                 z(ind)*h(ind)/6)*(t(ind+1)-x(i));
190 -           if Si(i)<minim
191 -               minim=Si(i);
192 -           elseif Si(i)>maxim
193 -               maxim=Si(i);
```

```
        end
    end

    if opg1==1
        figure(1);
        %Dibuja los puntos de control en la gráfica
        b1=plot(t,y,'ko','MarkerSize',2,'LineWidth',3);
        hold on;
        %Dibuja los valores interpolados en la gráfica
        b2=plot(x,Si,'ro','MarkerSize',3);
        hold on;
        axis([t(1)-0.1*(t(2)-t(1)) t(length(t))+...
            0.1*(t(2)-t(1)) min(min(y),minim)-...
            0.1*(max(y)-min(y)) max(max(y),maxim)+0.1*(max(y)-min(y))]);
        axis equal;
        legend([b1,b2],'Puntos de Control','Valores Interpolados');
    end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Polinomios S_i(x)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

syms r;



for i=1:m-1
    S(i)=z(i+1)/(6*h(i))*(r-t(i))^3+z(i)/(6*h(i))*(t(i+1)-r)^3+...
        (y(i+1)/h(i)-z(i+1)*h(i)/6)*(r-t(i))+(y(i)/h(i)-...
        z(i)*h(i)/6)*(t(i+1)-r);
end


if opg2==1
    %Dibuja los puntos de control en la gráfica
    figure(2);
```

```
  b1=plot(t,y,'ko','MarkerSize',2,'LineWidth',3);
  hold on;

  %Dibuja cada uno de los polinomios
  Ms=0;
  ms=0;
  for i=1:m-1
      b3=ezplot(S(i),[t(i),t(i+1)]);
      auxX=t(i):(t(i+1)-t(i))/10:t(i+1);
      auxY=subs(S(i),auxX); MauxY=max(auxY); mauxY=min(auxY);
      if MauxY>Ms
          Ms=MauxY;
      end
      if mauxY<ms
          ms=mauxY;
      end
  end
  axis(double([t(1)-0.1*(t(2)-t(1)) t(length(t))+0.1*(t(2)-t(1))  ms-0.1*(Ms-ms)  Ms+0.1*(Ms-ms)]));
  axis equal;
  legend([b1,b3],'Puntos de Control','Splines Cúbicos');
  title('Ajuste por Splines Cúbicos Interpolantes');
```

```
end


if opg3==1
    %Dibuja cada una de las primeras derivadas
    figure(3);
    hold on
    Ms=0;
    ms=0;
    S1=diff(S,1);
    for i=1:m-1
        b4=ezplot(S1(i),[t(i),t(i+1)]);
        auxX=t(i):(t(i+1)-t(i))/10:t(i+1);
        auxY=subs(S1(i),auxX); MauxY=max(auxY); mauxY=min(auxY);
        if MauxY>Ms
            Ms=MauxY;
        end
        if mauxY<ms
            ms=mauxY;
        end
    end
    axis(double([t(1)-0.1*(t(2)-t(1)) t(length(t))+...
        0.1*(t(2)-t(1))  ms-0.1*(Ms-ms)  Ms+0.1*(Ms-ms)]));
    legend(b4,'Primera Derivada Splines Cúbicos');
    title('Primera Derivada de los Splines Cúbicos Interpolantes');
end


if opg4==1
    %Dibuja cada una de las segundas derivadas
    figure(4)
    hold on
    S2=diff(S,2);
    Msd=subs(S2(1),t(1));
    msd=subs(S2(1),t(1));
    str=[];
    for i=1:m-1
        b5=ezplot(S2(i),[t(i),t(i+1)]);
        aux=subs(S2(i),t(i+1));
        if aux>Msd
            Msd=aux;
        elseif aux<msd
            msd=aux;
        end
    end
    axis(double([t(1)-0.1*(t(2)-t(1)) t(length(t))+...
        0.1*(t(2)-t(1)) msd-0.1*(Msd-msd)  Msd+0.1*(Msd-msd)]));
    legend(b5,'Segunda Derivada Splines Cúbicos');
    title('Segunda Derivada de los Splines Cúbicos Interpolantes');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  SALIDA DE RESULTADOS A UN FICHERO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%  Se abre o crea un archivo y se escribe en él para los resultados de valores interpolados

if opx=='s'
    fid=fopen('resul_valores_interpolados.txt','w');
    fprintf(fid,'*************************************************************************\n');
    hora=clock;
    fprintf(fid,'Resultado ejecutado el dia %d-%d-%d a las %d : %d : %d\n',...
        hora(3),hora(2),hora(1),hora(4),hora(5),fix(hora(6)));
```

```
317 -       fprintf(fid,'*************************************************************************************\n\n');
318 -       for i=1:length(x)
319 -           fprintf(fid,'%f %f \n',x(i),Si(i));
320 -       end
321 -       fclose(fid);
322 -   end
323
324     %  Se abre o crea un archivo y se escribe en él para los resultados de los polinomios del Spline
325
326 -   fid=fopen('polinomios_splines.txt','w');
327 -   fprintf(fid,'*************************************************************************************\n');
328 -   hora=clock;
329 -   fprintf(fid,'Resultado ejecutado el dia %d-%d-%d a las %d : %d : %d\n',...
330           hora(3),hora(2),hora(1),hora(4),hora(5),fix(hora(6)));
331 -   fprintf(fid,'*************************************************************************************\n\n');
332 -   signos='----';
333 -   for i=1:length(t)-1
334 -       coefi=sym2poly(expand(S(i)));
335 -       for j=1:4
336 -           if coefi(j)>=0
337 -               signos(j)='+';
```

```
338 -               end
339 -           end
340 -           fprintf(fid,' %c %f x^3 %c %f x^2 %c %f x %c %f  \n',...
341                 signos(1),abs(coefi(1)),signos(2),abs(coefi(2)),signos(3),...
342                 abs(coefi(3)),signos(4),abs(coefi(4)));
343 -       end
344 -       fprintf(fid,'\n');
345 -       fclose(fid);
346
```

Sin embargo, este programa lo usaremos para el cálculo de los SPLINES, no para su representación gráfica, ya que en realidad nosotros no queremos hacer SPLINES suavizantes, sino comparar los que hemos calculado a un lado y a otro de la discontinuidad para saber su estado y comparar ambas funciones a los lados comprobando la continuidad de las formas.

6- **SPLINES_NU:** este es el programa final que vamos a utilizar, como nuestro programa final este engloba, es decir, usa todos los programas que hemos definido anteriormente en alguno de sus pasos para la obtención final de la reconstruccion del SPLINE a trozos de una función que posee discontinuidades de esquina. El NU que lleva en el nombre significa “Non-Uniform”, lo que quiere decir que nuestro programa podra trabajar tanto para mallados uniformes como no uniformes, dependiendo de los datos de dicho mallado.

```
function Spline_nu_D(fcell,Dcell,n,mev)


% Esta función calcula la reconstrucción de Spline a trozos de una función
% discontinua dada, partiendo justo en las discontinuidades para no
% cruzarlas
% Spline_nu_D(fcell,Dcell,n,mev)
%
% Variables de entrada:
% fcell variable de celda conteniendo las expresiones de las funciones
%       entre discontinuidades
% Dcell variable de celda conteniendo las posiciones de las
%       discontinuidades. Dcell empieza por el extremo inicial a y
%       acaba en el extremo final b
% n número de puntos en que discretizamos el intervalo [a,b]
% mev número de puntos del mallado fino (m >> n) donde discretizamos
%     el intervalo [a,b]
%
% Ejemplo 1:
% Spline_nu_D({10*sin(pi*x),-50*cos(pi*x)+10,20*sin(pi*x)-40,-40+5*sin
% (pi*x)},{0,0.5,2,3,2*pi},150,1000);
% Ejemplo 2:
% Spline_nu_D({-50*cos(pi*x)+10,20*sin(pi*x)-40},{0,2,2*pi},150,1000);

syms x

%%%%%%%%%%%%%%%%%%%%%%%%%
% Construimos la función
%%%%%%%%%%%%%%%%%%%%%%%%%

m=length(Dcell);  % número de discontinuidades más los dos extremos a y b

a=Dcell{1};   % extremo izquierdo
b=Dcell{m};   % extremo derecho

xa=linspace(a,b,n);   % mallado usado para aproximar
hxa=(b-a)/n;
rxa=0.1*hxa*rand(size(xa));
xa=xa+rxa;

h=diff(xa);   % vector de espaciados no uniformes


xev=linspace(a,b,mev);   % mallado usado para evaluar y medir errores de
% aproximación


y=[];    % inicializamos el vector con los valores numéricos de la función

raiz=[]; % inicializamos el vector de raíces encontradas


dizq=Dcell{1};


n_nue=0;

for i=2:m

    dfaux{i-1}=fcell{i-1};
    dder=Dcell{i};
    ind_aux=find( (dizq <=xa) & (xa< dder));
    x_aux=xa(ind_aux);
    n_ant=n_nue;
    n_nue=n_nue+length(x_aux);
    faux=dfaux{i-1};
    y_aux=subs(faux,{x},x_aux);
    y(n_ant+1:n_nue)=y_aux;
    dizq=dder;

end
y(n)=double(subs(faux,{x},b));
```

```
y(n)=double(subs(faux,{x},b));



% evaluación de la función en los nodos de evaluación

dizq=Dcell{1};

n_nue=0;

for i=2:m

    dder=Dcell{i};
    ind_aux=find( (dizq <=xev) & (xev< dder));
    x_aux=xev(ind_aux);
    n_ant=n_nue;
    n_nue=n_nue+length(x_aux);
    faux=fcell{i-1};
    y_aux=double(subs(faux,{x},x_aux));
    fevE(n_ant+1:n_nue)=y_aux;
    dizq=dder;

end

yevE(mev)=double(subs(faux,{x},b));



% Cálculo de las condiciones iniciales en los extremos

% Segunda derivada en el extremo a

dfaux2=diff(dfaux{1},x,2);
dfdera=double(subs(dfaux2,x,a));


dfaux2=diff(dfaux{m-1},x,2);
dfizqb=double(subs(dfaux2,x,b));


% Cálculo del valor máximo y mínimo de la función
```

```
minY=min(y);
maxY=max(y);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Detectamos dónde están las discontinuidades
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



% Primero marcamos los intervalos sospechosos

% calculamos las diferencias divididas de segundo orden

for i=2:n-1
    d2(i)=1/(h(i-1)*(h(i-1)+h(i)))*y(i-1)+1/(h(i-1)*h(i))*y(i)+1/(h(i)*(h(i-1)+h(i)))*y(i+1);
end

N=4;
candidatos=BB(d2,N-1);


% Establecemos las condiciones del artículo de Rosa Donat et. al

i=0;
for j=1:length(d2)

    if candidatos(j)==1
        i=i+1;
        nodosI=xa(j-N+1:j);
        yI=y(j-N+1:j);
        nodosD=xa(j+1:j+N);
        yD=y(j+1:j+N);
        indice_raiz(i)=j;
        [raiz(i)]=Mu(nodosI,yI,nodosD,yD);
        candidatos(j+1)=0;
    elseif candidatos(j)==2
        i=i+1;
        nodosI=xa(j-N+2:j+1);
        yI=y(j-N+2:j+1);
        nodosD=xa(j+2:j+N+1);
```

```
            yD=y(j+2:j+N+1);
            indice_raiz(i)=j+1;
            [raiz(i)]=Mu(nodosI,yI,nodosD,yD);
            candidatos(j+1)=0;
        end

end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Aproximación de la función por trozos entre discontinuidades si las hay
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if ~isempty(raiz)

    % Inicializamos el vector que contendrá los valores de la función
    % reconstruida para ir completándolos

    yevD=[];

    % Número de raíces intermedias encontradas

    nr=length(raiz);


    % Partimos nuestro intervalo en trozos en cada raíz

    % Primer trozo: desde el inicio a la primera raíz

    % Cálculo del valor de la función y las derivadas en la primera raíz por la
    % izquierda


    dfizq=get_derivatives(xa(indice_raiz(1)-3:indice_raiz(1)),y(indice_raiz(1)-3:indice_raiz(1)),raiz(1));
    dfizq=dfizq(1:4)';

    xa_aux=[xa(1:indice_raiz(1)),raiz(1)];
    y_aux=[y(1:indice_raiz(1)),dfizq(1)];
    ind_xev_aux=find(xev<=raiz(i));
    xev_aux=xev(ind_xev_aux);


    yevD_aux=Splines_Interpolantes(xa_aux,y_aux,2,0,dfdera,2,0,dfizq(3),'s',xev_aux,0,0,0,0);


    yevD=[yevD,yevD_aux];


    % trozos centrales entre raíces

    for i=1:nr-1

        % Cálculo del valor de la función y las derivadas en la raíz i por la
        % derecha y de la raíz i+1 por la izquierda

        dfder=get_derivatives(xa(indice_raiz(i)+1:indice_raiz(i)+4),y(indice_raiz(i)+1:indice_raiz(i)+4),raiz(i));
        dfder=dfder(1:4)';

        dfizq=get_derivatives(xa(indice_raiz(i+1)-3:indice_raiz(i+1)),y(indice_raiz(i+1)-3:indice_raiz(i+1)),raiz(i+1));
        dfizq=dfizq(1:4)';


        xa_aux=[raiz(i),xa(indice_raiz(i)+1:indice_raiz(i+1)),raiz(i+1)];
        y_aux=[dfder(1),y(indice_raiz(i)+1:indice_raiz(i+1)),dfizq(1)];

        ind_xev_aux=find((xev>raiz(i)) & (xev <= raiz(i+1)));
        xev_aux=xev(ind_xev_aux);
```

```
227
228 -           yevD_aux=Splines_Interpolantes(xa_aux,y_aux,2,0,dfder(3),2,0,dfizq(3),'s',xev_aux,0,0,0,0);
229
230 -           yevD=[yevD,yevD_aux];
231
232
233 -       end
234
235         % Último trozo: entre la última raíz y el extremo b
236
237         % Cálculo del valor de la función y las derivadas en la última raíz por la
238         % derecha
239
240
241 -       dfder=get_derivatives(xa(indice_raiz(nr)+1:indice_raiz(nr)+4),y(indice_raiz(nr)+1:indice_raiz(nr)+4),raiz(nr));
242 -       dfder=dfder(1:4)';
243
244 -       xa_aux=[raiz(nr),xa(indice_raiz(nr)+1:end)];
245 -       y_aux=[dfder(1),y(indice_raiz(nr)+1:end)];
246 -       ind_xev_aux=find(xev>raiz(nr));
247 -       xev_aux=xev(ind_xev_aux);
248
249 -       yevD_aux=Splines_Interpolantes(xa_aux,y_aux,2,0,dfder(3),2,0,dfizqb,'s',xev_aux,0,0,0,0);
250
251 -       yevD=[yevD,yevD_aux];
252
253
254
255 -   else   % Hace Splines
256
257
258 -       size(xa)
259 -       size(y)
260 -       plot(xa,y);
261 -       pause
262
263 -       yevD=Splines_Interpolantes(xa,y,2,0,dfdera,2,0,dfizqb,'s',xev,0,0,0,0);
264
265 -   end
266
267
268     % Dibujamos la función y la reconstrucción obtenida
```

```
269
270 -   figure
271 -   plot(xev,yevE,'b',xev,fevD,'r')
272 -   axis([xa(1)-0.1*(xa(end)-xa(1)) xa(end)+0.1*(xa(end)-xa(1)) minY-0.1*(maxY-minY) maxY+0.1*(maxY-minY)]);
273 -   title('Reconstrucción por Splines Interpolantes a Trozos');
274 -   legend('Función Original','Reconstrucción por Splines Interpolatorios a Trozos');
275
276     % Imprimimos la norma del error
277
278
279 -   norm(yevE-yevD,inf)
280
281
282
283 -   end
```

Con esto, solo no faltaría crear una interfaz gráfica que nos permita la introducción de los datos en nuestro programa, asi como los distintos inputs o entradas que pueda tener el programa para que asi su uso sea mas sencillo y visible para cualquier usuario que desee realizar algún cálculo con dicho programa.

Esta matriz se compone de comandos que llaman a los diferentes programas, o más bien, lineas de un mismo programa en las cuales se encuentra la programación necesaria para el cálculo que deseemos. Por tanto, para la hora de su creación debemos tener claros los cálculos que puede hacer mi programa, así como los cálculos que podrían desear conocer los usuarios para así no dejar que la interfaz minimice las operaciones que puede hacer el programa pero tampoco nos muestre cálculos que se realizan para llegar a una conclusión dada, que serán los datos que interesen a los usuarios.

Capítulo VIII
Interfaz Gráfica

Capítulo VIII

Interfaz Gráfica

En este capítulo vamos a mostrar la interfaz gráfica realizada para nuestro programa así como una breve definición de los diferentes elementos que vamos a encontrar en ella.

8.1- Creación de la Interfaz Gráfica

Para la realización de esta interfaz vamos a utilizar el entorno gráfico de MATLAB que le permita a cualquiera de los usuarios realizar el cálculo de la reconstrucción del SPLINE a trozos para una función discontinua conocida. A continuación hacemos un repaso de los elementos que compone esta interfaz y cómo usarla.

Lo primero que debemos saber es que dentro de la carpeta en la cual creemos la interfaz, tendremos que tener todos y cada uno de los elementos que se vayan a usar en ella, es decir, cualquiera de los "botones" o de las posibles elecciones que podamos hacer una vez abierto el programa así como las imágenes que vayan a mostrarse y que no son parte del cálculo realizado por el programa, ya que sino nuestra interfaz no se mostraría como deseamos ni realizaría los cálculos para los que ha sido programada. Los programas e imágenes utilizados en esta interfaz serán los siguientes:

- **Programas**
  - crear_datos
  - crear_datosx
  - crear_datosy
  - crear_evaluacion
  - crear_funcion
  - crear mallado
  - Spline_nu_D
  - Spline_nu_D_datos → Creado para que el programa inicial pueda leer ficheros de datos
  - Splines_Interpolantes
  - sci → programa de la interfaz gráfica

- **Imágenes**
  - escudo.jpg
  - escudo_navales.jpg
  - escudo_upct.jpg

Para poder ejecutar nuestra interfaz, tendremos que ir a la ventana de comandos de Matlab e introducir el comando sci.

8.2- Partes de la Interfaz Gráfica

Antes de entrar en la interfaz gráfica que nos permite realizar los cálculos, tendremos una pequeña interfaz con tres botones que vamos a definir ahora:

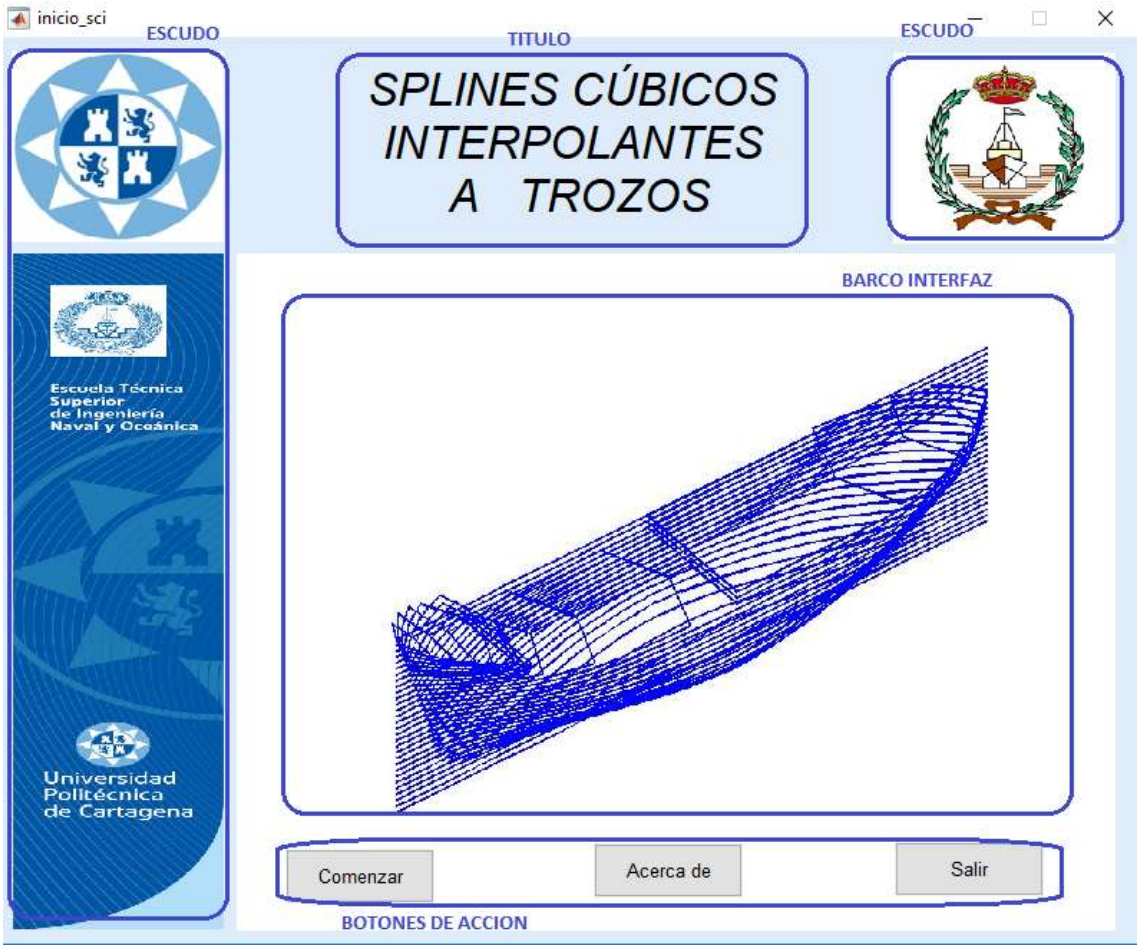
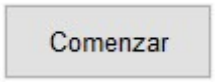
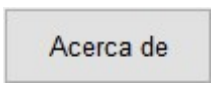


Fig.36 – Inicio Programa SCI

En este caso vamos e explicar que realizan los 3 botones de acción:



Al pulsar este botón se nos abrirá la interfaz gráfica principal que nos permite realizar los cálculos programados para SPLINES que hemos propuesto en el trabajo.



Nos muestra la información acerca del Trabajo Fin de Grado realizado así como de su autor y directores.

Nos da la opción de salir del programa mediante la creación de una ventana emergente la cual nos pregunta si deseamos o no salir y será de la siguiente forma:

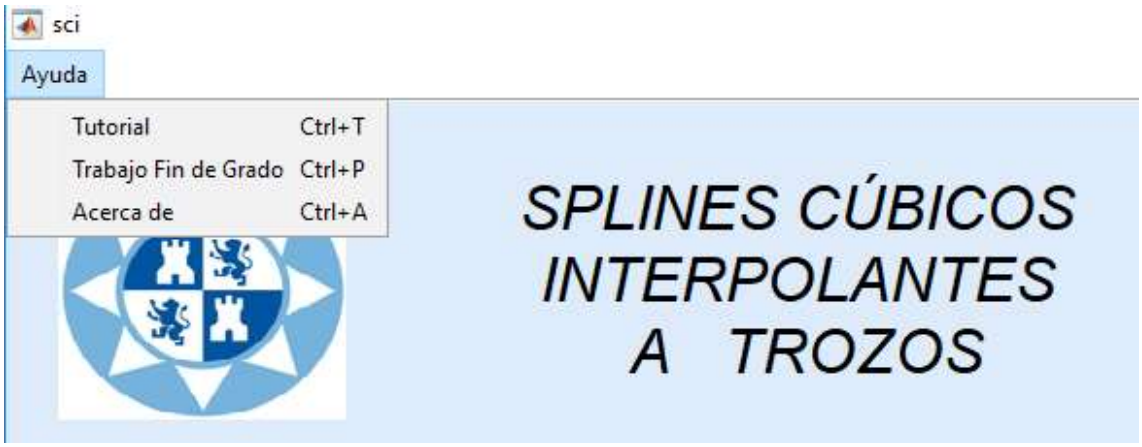
Tras esto vamos a ver que partes componen nuestra interfaz gráfica de cálculo, haciendo una breve definición de cada una de las partes así como una pequeña explicación de cómo debe usarse así como para que se usa.

Una vez ejecutamos la interfaz gráfica, se nos mostrara en pantalla la siguiente imagen:

Fig.37 – Interfaz Gráfica del Programa SCI

Como podemos observar, en la imagen podemos ver varias secciones, tres para ser concretos, cada una de ellas con sus distintos pulsadores, así como una serie de pulsadores que son los que le dirán al programa que está pidiendo el usuario. En la siguiente imagen veremos nuestra interfaz subdividida para poder explicar cada una de las partes:

En esta imagen vemos cada una de las partes de la interfaz gráfica, además de una pequeña información de las imágenes que contiene. A continuación vamos a explicar cada una de las partes indicadas:

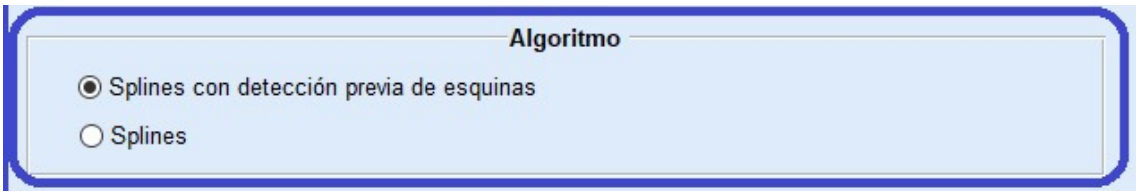


Inicialmente vamos a definir brevemente el cuadro de ayuda que puede ofrecer nuestro programa. Vemos como la ventana de ayuda está compuesta por tres apartados:

- Tutorial: si pulsamos aquí se nos mostrará un tutorial que nos ayuda a comprender y usar el programa que hemos creado.
- Trabajo Fin de Grado: aquí se muestra una pequeña definición del trabajo fin de grado.
- Acerca de: se nos muestra una pantalla con información acerca de la realización del programa y los autores.

Una vez definida la ayuda del programa pasamos a definir cada una de las partes que este tiene y que es necesario conocer para un correcto uso del programa:

1- Algoritmo:

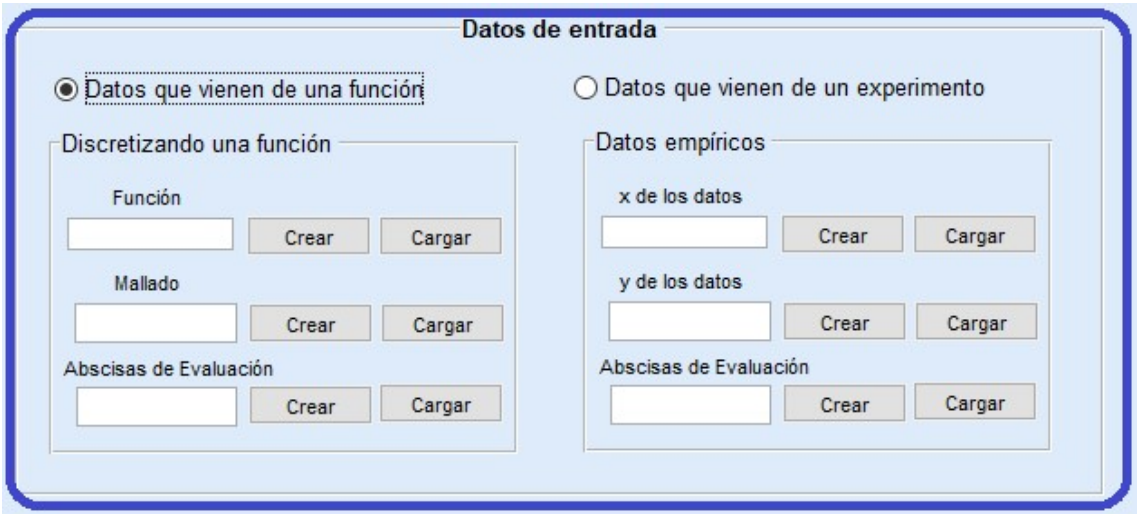


En este primer apartado podremos escoger el modo de trabajar de nuestro programa, como vemos en la imagen, podremos escoger entre dos opciones distintas, la primera es la realización del SPLINE con detección previa de esquinas y la otra opción es la realización del SPLINE normal. Podremos escoger la que más se adecúe a nuestras necesidades.

En cualquier caso, no podremos seleccionar los dos tipos de algoritmos a la vez ya que el programa no nos lo permitirá. Para comparar los resultados podremos realizar primero el cálculo mediante un algoritmo y posteriormente realizándolo con el otro tipo.

Cuando hayamos escogido el tipo de algoritmo a aplicar, tendremos que escoger ahora el tipo de datos de entrada que usara el programa.

2- Datos de entrada:



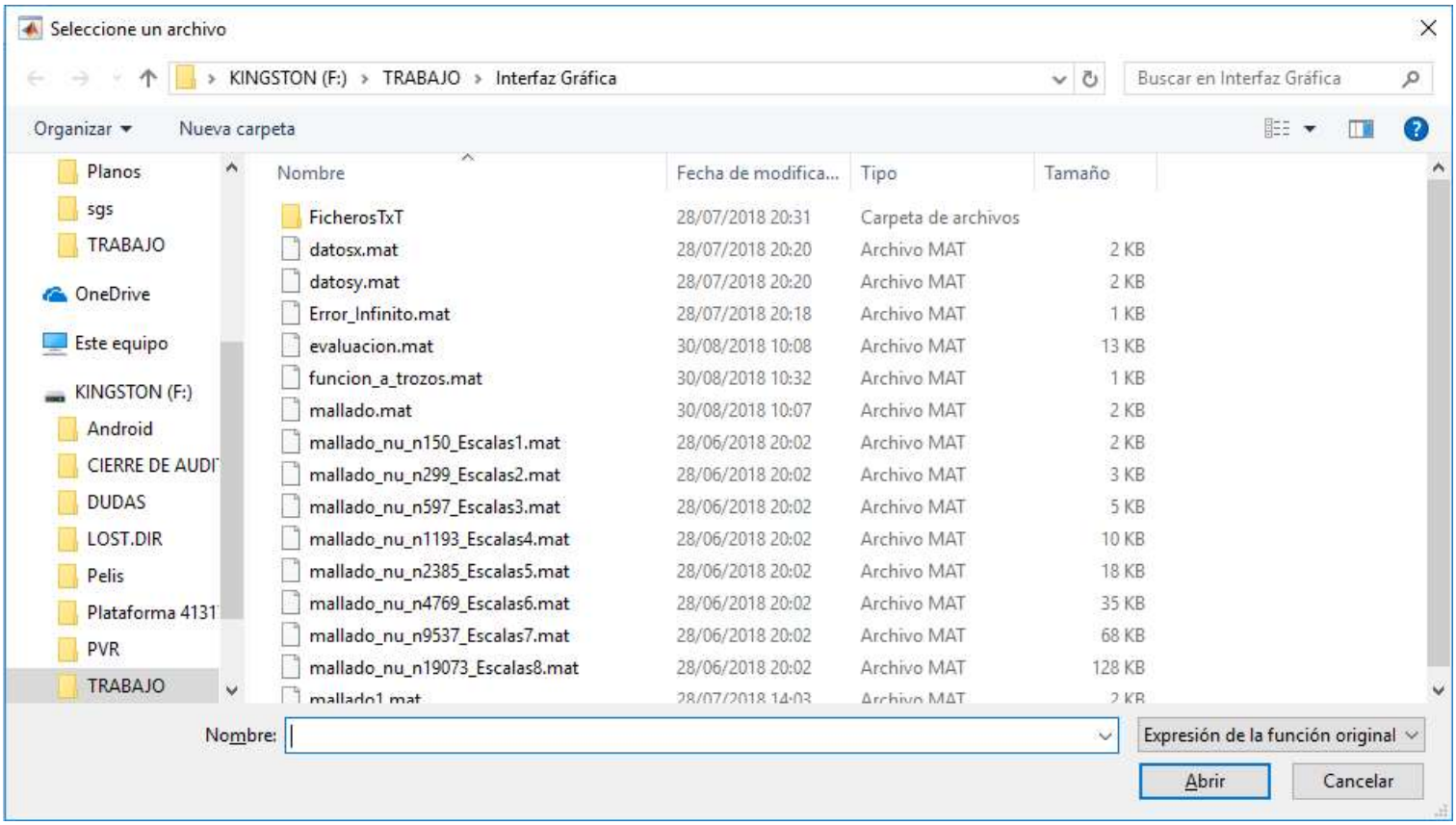
Aquí podemos escoger el tipo de datos de entrada que queremos usar, como vemos hay dos opciones, una en la cual los datos que introducimos serán una función y otra que se basa en datos tomados de un experimento, es decir, una serie de datos, no una función. Para el primer caso, datos que vienen de una función, tendremos tres series de datos:

- **Función**: aquí introducimos la función que queremos analizar, puede ser creada por nosotros o cargada desde una base de datos externa.
  - Si pulsamos el botón crear se abrirá MatLab, el programa crear_funcion.mat para que introduzcamos los datos en este, será de la siguiente forma, donde explica como introducir la función deseada:

```
function [fichero_funcion]=crear_funcion()

% Esta función crea un fichero de Matlab .mat que contiene dos variables
% tipo cell, una con la expresión de cada función en cada trozo entre discontinuidades
% y otra con la posición real de las discontinuidades presentes en la
% función a trozos
%
% [fichero_funcion]=crear_funcion();
%
% Variables de entrada:
% No necesita
%
% Variables de salida:
% fichero_funcion, cadena de caracteres que contiene el nombre del fichero
%                  .mat que guarda las dos variables tipo celda fcell, Dcell
%                  con las expresiones de las funciones y de las posiciones
%                  de las discontinuidades respectivamente


% Declaramos la variable x como simbólica

syms x


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Completar con las expresiones de las funciones en cada trozo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Ejemplo 1

fcell={10*sin(pi*x),-50*cos(pi*x)+10,20*sin(pi*x)-40,-40+5*sin(pi*x)};
Dcell={0,0.5,2,3,2*pi};


% % Ejemplo 2
%
% fcell={-50*cos(pi*x)+10,20*sin(pi*x)-40};
% Dcell={0,2,2*pi};



% Se guardan las variables fcell, Dcell en un fichero llamado
```

```
44      % funcion_a_trozos.mat
45
46 -    fichero_funcion='funcion_a_trozos.mat';
47
48 -    save(fichero_funcion,'fcell','Dcell');
49
```

- Si pulsamos el botón cargar se abrirá una ventana en la cual tendremos que escoger la función que hallamos guardado con anterioridad y que deseamos analizar. La ventana tendrá la siguiente forma:



- **Mallado**: ahora debemos escoger el mallado a utilizar, podemos escoger tanto crearlo nosotros como escoger un mallado ya creado.

- Si escogemos la opción de crear nuestro mallado se nos abrirá MatLab, el programa crear_mallado.m, que nos indica como introducir los datos para crear el mallado según lo deseemos:

```
function [fichero_mallado]=crear_mallado()


% Esta función crea un fichero de Matlab .mat que contiene una variable
% tipo vector de doubles que contiene las posiciones x  de los nodos usados
% para construir la reconstrucción
%
% [fichero_mallado]=crear_mallado();
%
% Variables de entrada:
% No necesita
%
% Variables de salida:
% fichero_mallado, cadena de caracteres que contiene el nombre del fichero
%                  .mat contiene una variable tipo vector de doubles que
%                  contiene las posiciones x  de los nodos usados para
%                  construir la reconstrucción




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Introducimos el mallado
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Mallado igualmente espaciado
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



% Introducir correctamente los extremos del intervalo a considerar

a=0;
b=2*pi;

% Número de puntos en el mallado

n=150;

% Generamos el mallado

xa=linspace(a,b,n);


% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Mallado no igualmente espaciado
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
% % Podemos generar la variable xa usando el programa  mallado_nu.m
%
% n=150;        % número de puntos en el mallado no uniforme
% lambda=0.1;   % indica lo que varia del mallado uniforme
% escalas=1;    % indica si se quieren también mallados refinados sucesivamente por la mitad
%
% [xa,fichero_mallado]=mallado_nu(a,b,n,lambda,escalas);
%
%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Definiendo directamente el vector xa
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
% xa=[0,0.2,0.4,0.7,1,1.3,1.5,2,pi];
%
%
%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Leyendo el vector xa de un fichero excel
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% xa= xlsread(filename,sheet,xlRange);



% Se guarda la variable xa en un fichero llamado
% mallado.mat

fichero_mallado='mallado.mat';

save(fichero_mallado,'xa');
```

- Si por el contrario, queremos escoger un mallado ya creado, pulsaremos el botón cargar, el cual nos abrirá una ventana similar a la anterior para la función donde buscaremos nuestro mallado.

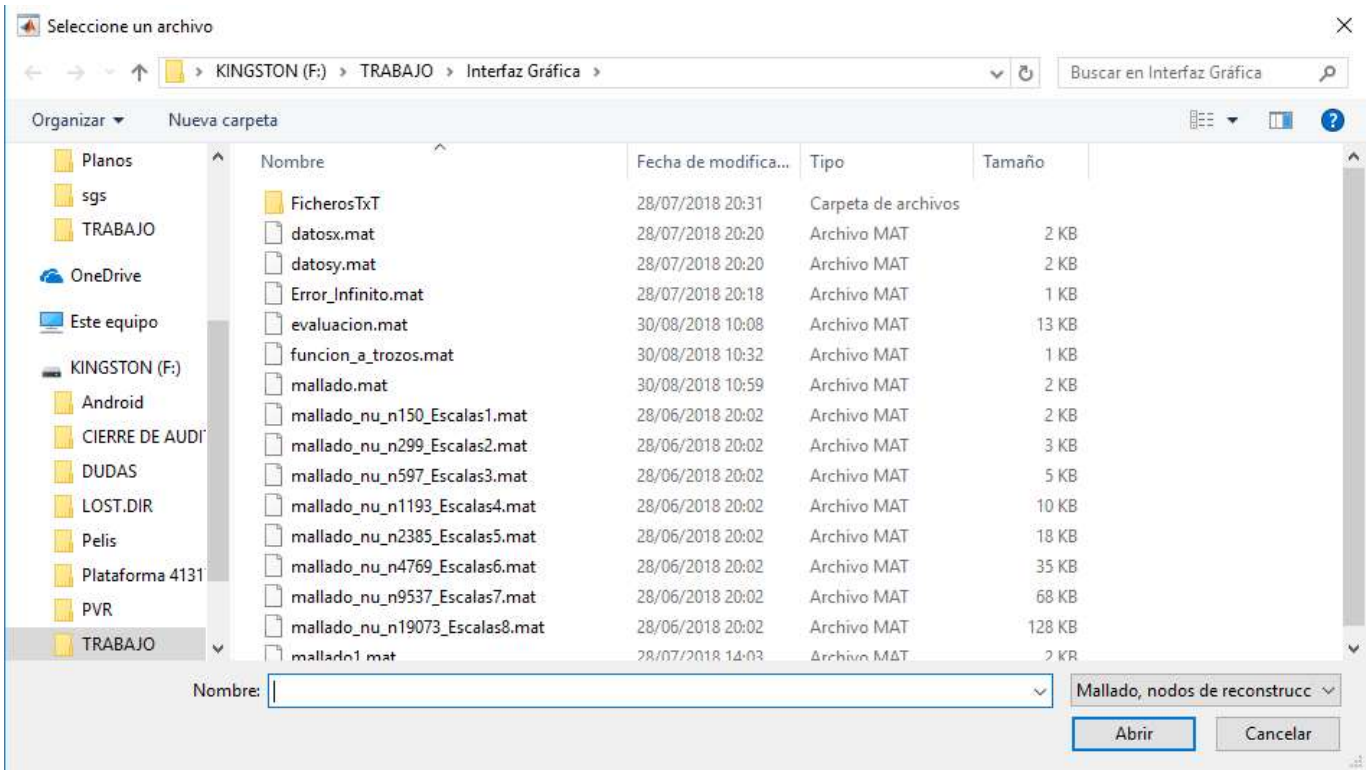


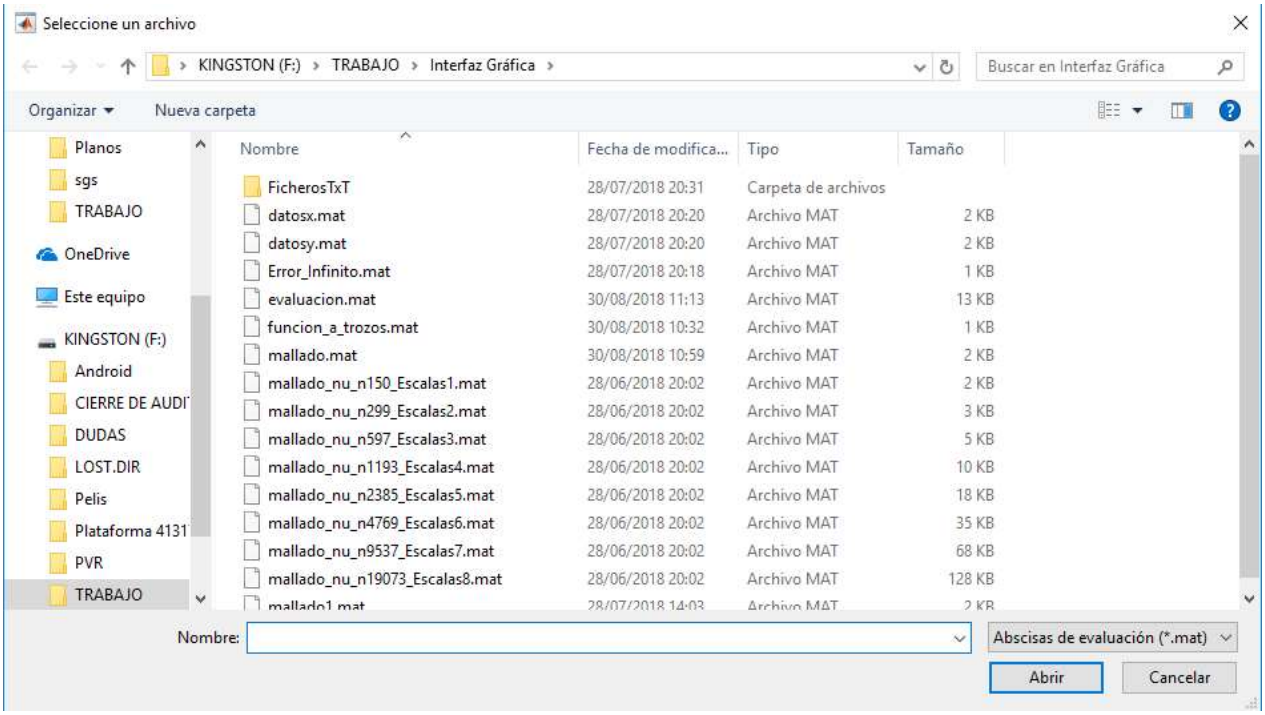
Fig.38 – Ventana entrada datos

- **Abscisas de Evaluación**: aquí podemos escoger las abscisas del intervalo que vamos a analizar, al igual que en los casos anteriores, podremos crearlo nosotros o bien escoger un fichero ya creado.

  - Al igual que en los casos anteriores, si escogemos la opción de crear, nos abrirá en MatLab el programa crear_evaluacion que nos permitirá crear las abscisas de evaluación como deseemos, explicando la manera de hacerlo dentro del mismo programa:

```
function [fichero_evaluacion]=crear_evaluacion()

% Esta función crea un fichero de Matlab .mat que contiene la variable
% xev, que contiene las abscisas, normalmente en el intervalo [a,b], donde se quiere
% evaluar la reconstrucción
%
% [fichero_evaluacion]=crear_evaluacion();
%
% Variables de entrada:
% No necesita
%
% Variables de salida:
% fichero_evaluacion, cadena de caracteres que contiene el nombre del fichero
%                   .mat que contiene la variable xev, que contiene las abscisas,
%                    normalmente en el intervalo [a,b], donde se quiere evaluar
%                    la reconstrucción




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Introducimos las abscisas de evaluación
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Mallado igualmente espaciado
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


xev=linspace(a,b,n);


% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Mallado no igualmente espaciado
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
% % Podemos generar la variable xa usando el programa  mallado_nu.m
%
% n=2000;       % número de puntos en el mallado no uniforme
% lambda=0.1;  % indica lo que varia del mallado uniforme
% escalas=1;   % indica si se quieren también mallados refinados sucesivamente por la mitad
%
% [xev,fichero_evaluacion]=mallado_nu(a,b,n,lambda,escalas);
%
%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Definiendo directamente el vector xev
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
% xev=[0.25,0.5];
%
%
%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Leyendo el vector xev de un fichero excel
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% xev= xlsread(filename,sheet,xlRange);



% Se guarda la variable xev en un fichero llamado
% evaluacion.mat

fichero_evaluacion='evaluacion.mat';

save(fichero_evaluacion,'xev');
```

- Y para finalizar el caso de que usemos una función en lugar de datos empíricos podemos escoger cargar las abscisas de evaluación al darle al botón de cargar, que nos abrirá la siguiente ventana para escoger el fichero deseado.



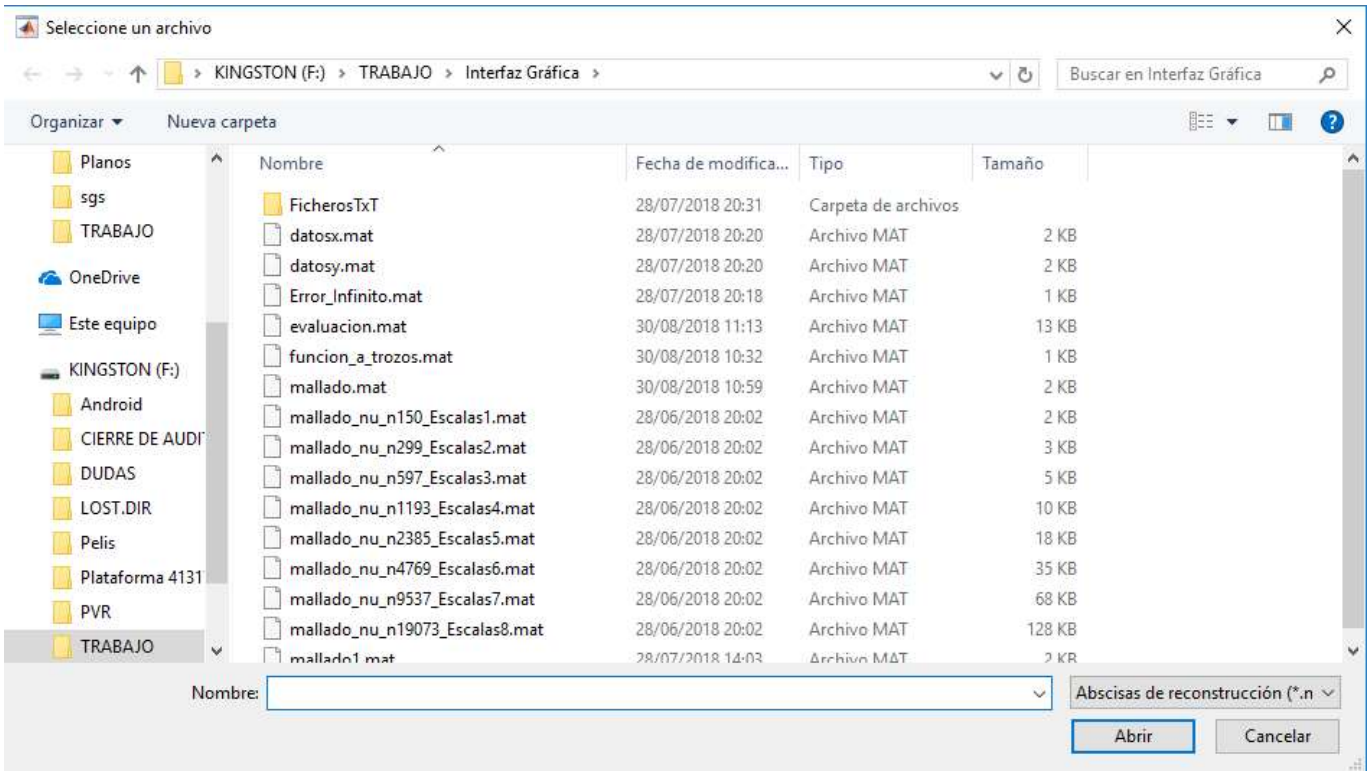
Para el caso de que escojamos **“Datos que vienen de un experimento”**, tendremos un desarrollo similar al anterior, salvo que no introduciremos funciones, sino grupos de datos que el programa analizará. Para este caso tendremos igual otras tres elecciones que realizar:

- **X de los datos**: introducimos o cargamos una serie de datos que serán los valores X para nuestro programa.
  - Podemos crear los datos, que igual que en los casos anteriores nos mandará al programa de MatLab crear_datosx.m que será de la siguiente forma:

```
function [fichero_datosx]=crear_datosx()


% Esta función crea un fichero de Matlab .mat que contiene una variable
% tipo vector de doubles xa que contiene las posiciones x  de los nodos usados
% para construir la reconstrucción
%
% [fichero_datosx]=crear_datosx();
%
% Variables de entrada:
% No necesita
%
% Variables de salida:
% fichero_datosx, cadena de caracteres que contiene el nombre del fichero
%                   .mat contiene una variable tipo vector de doubles xa que
%                   contiene las posiciones x  de los nodos usados para
%                   construir la reconstrucción




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Introducimos el mallado
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Mallado igualmente espaciado
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



% Introducir correctamente los extremos del intervalo a considerar

a=0;
b=2*pi;

% Número de puntos en el mallado

n=150;

% Generamos el mallado

```

```
xa=linspace(a,b,n);


% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Mallado no igualmente espaciado
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
% % Podemos generar la variable xa usando el programa  mallado_nu.m
%
% n=150;        % número de puntos en el mallado no uniforme
% lambda=0.1;   % indica lo que varia del mallado uniforme
% escalas=1;    % indica si se quieren también mallados refinados sucesivamente por la mitad
%
% [xa,fichero_mallado]=mallado_nu(a,b,n,lambda,escalas);
%
%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Definiendo directamente el vector xa
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
% xa=[0,0.2,0.4,0.7,1,1.3,1.5,2,pi];
%
%
%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Leyendo el vector xa de un fichero excel
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% xa= xlsread(filename,sheet,xlRange);



% Se guarda la variable xa en un fichero llamado
% mallado.mat

fichero_datosx='datosx.mat';

save(fichero_datosx,'xa');
```

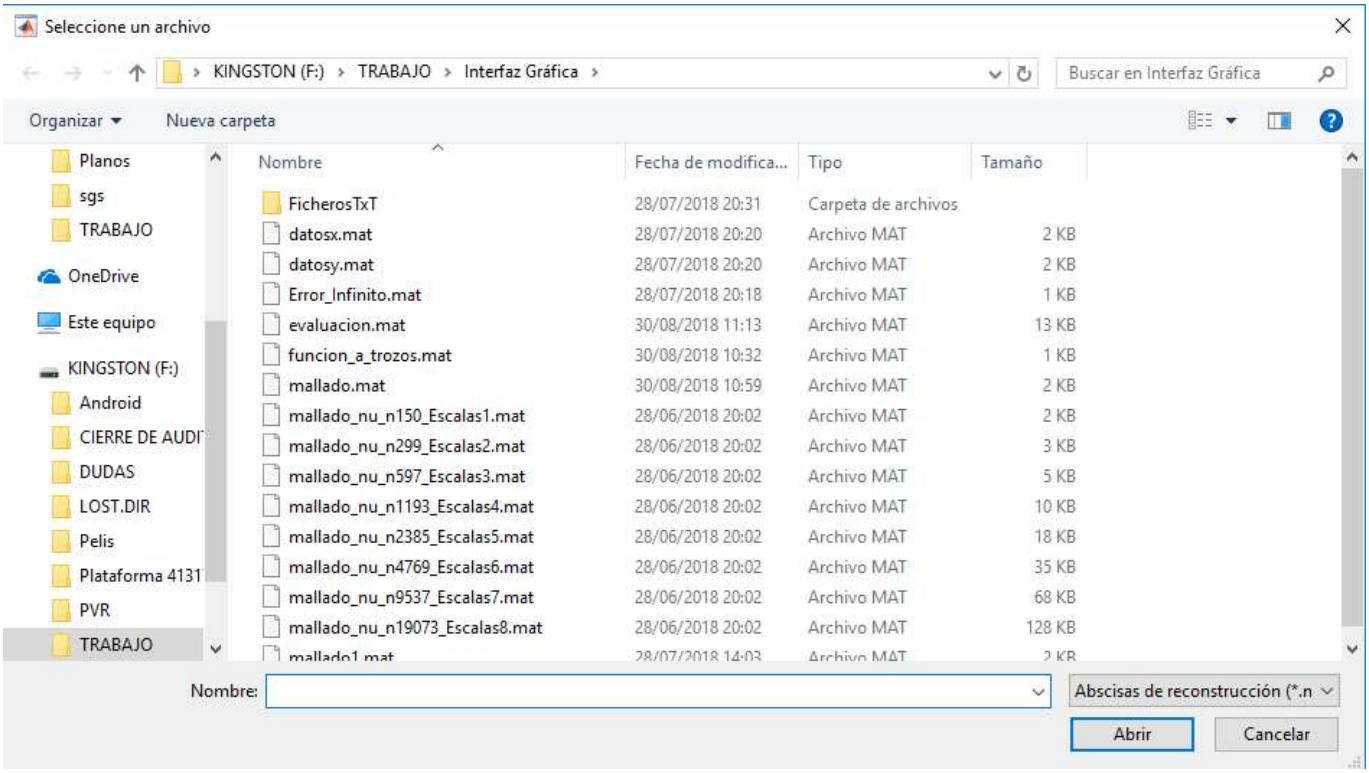
- También podemos cargar los datos de una base existente.



- **Y de los datos**: introducimos o cargamos una serie de datos que serán los valores Y para nuestro programa.
  - Podemos crear los datos, que igual que en los casos anteriores nos mandará al programa de MatLab crear_datosx.m que será de la siguiente forma

```
function [fichero_datosy]=crear_datosy()


% Esta función crea un fichero de Matlab .mat que contiene una variable
% tipo vector de doubles y que contiene las ordenadas y de los datos usados
% para construir la reconstrucción
%
% [fichero_datosy]=crear_datosy();
%
% Variables de entrada:
% No necesita
%
% Variables de salida:
% fichero_datosy, cadena de caracteres que contiene el nombre del fichero
%                   .mat contiene una variable tipo vector de doubles y que
%                   contiene las ordenadas y de los datos usados para construir
%                   la reconstrucción





%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Evaluamos la función en un mallado que se carga de un archivo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% cargamos el archivo, y por tanto la variable xa

load('datosx.mat');

% evaluamos una función en el mallado

y=(xa.^2).*sin(xa);



% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Definiendo directamente el vector y
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
% y=[10,9.2,7.7,7.5,7,6.3,6.5,6.9];
%
%
%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Leyendo el vector xa de un fichero excel
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% y= xlsread(filename,sheet,xlRange);



% Se guarda la variable y en un fichero llamado
% datosy.mat

fichero_datosy='datosy.mat';

save(fichero_datosy,'y');

```

- También podemos cargar los datos de una base existente como hemos hecho anteriormente.

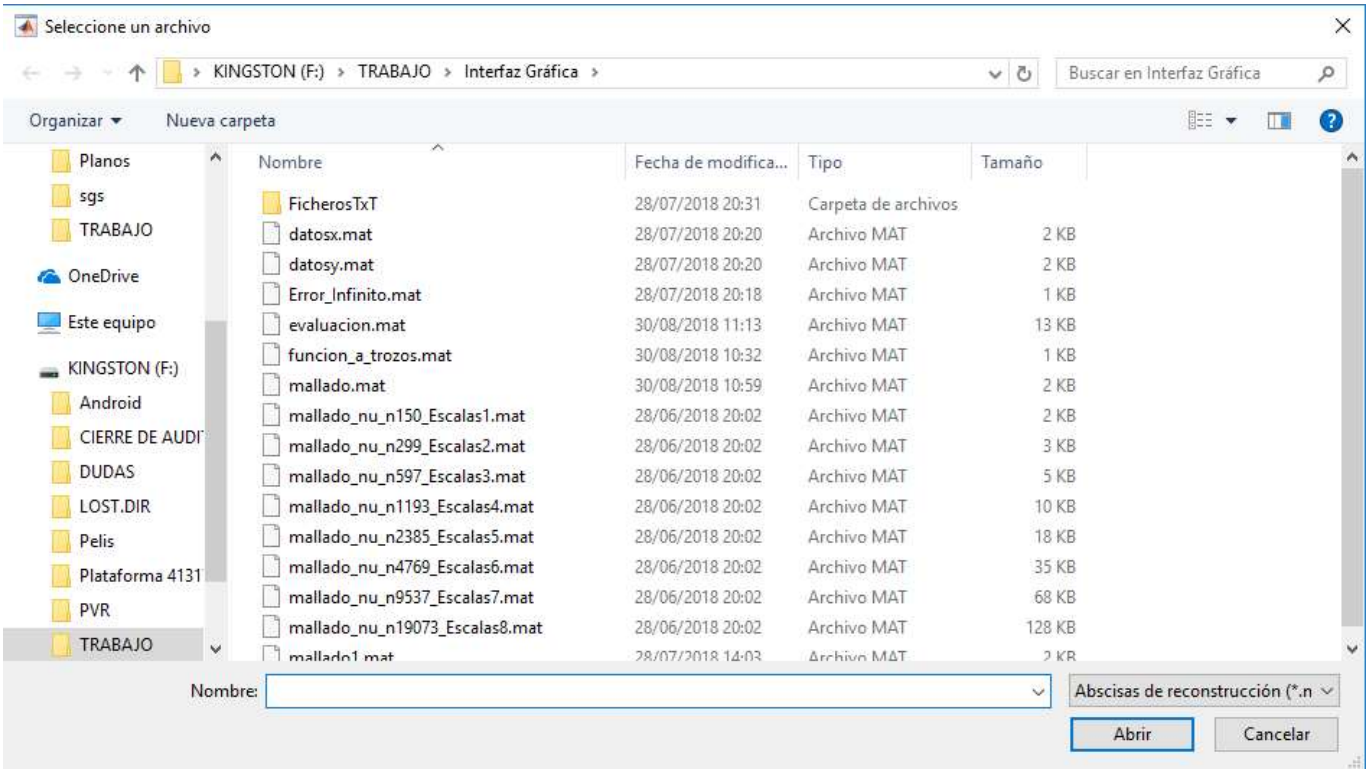


- **Abscisas de Evaluación**: finalmente y al igual que en el caso para funciones, podremos introducir las abscisas del intervalo a evaluar o bien cargarlas desde una base de datos externa.

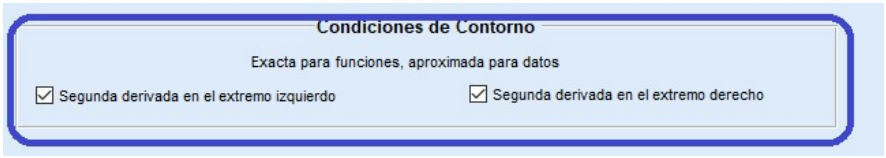
  - Para crearlas pulsaremos el botón Crear que abrirá el programa crear_datos.m el cual nos explica como introducir los datos y será de la siguiente forma:

```
function x=crear_datos()

% completar las abscisas donde se quiere evaluar la reconstrucción

%x=[0:0.5:1.5];

x=2.5;
```

- Y como hemos mostrado para los casos anteriores, también podremos cargar los datos desde una base externa mediante el botón “Cargar”:

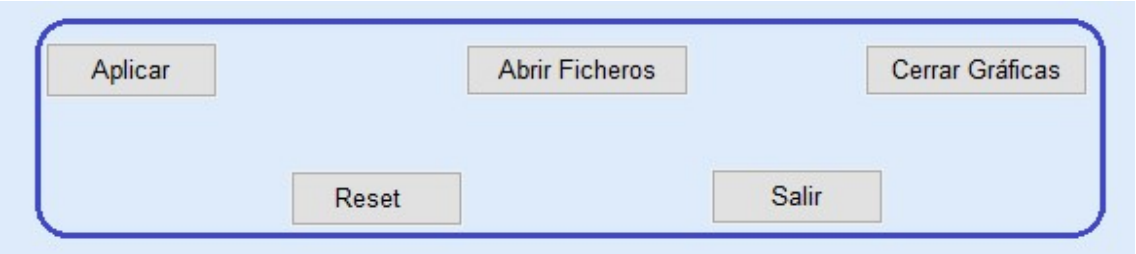


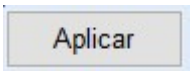
3- Datos de entrada:



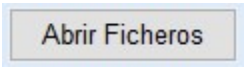
En este caso, los marcadores están activados ya que es una parte informativa, es decir, que se usará la segunda derivada tanto en el extremo izquierdo como en el derecho en cualquiera de los casos que introduzcamos.

4- Botones de acción:



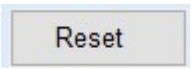


Al pulsar este botón se realizara el cálculo que hemos propuesto.



Al pulsar este botón abrirá los ficheros que contienen los valores que hemos calculado.

Al pulsar este botón cerraremos las gráficas que ha creado el programa mediante el cálculo que ha realizado.



Al pulsar este botón borraremos los valores que hemos introducido para poder realizar nuevos cálculos.

Al pulsar este botón se nos mostrara una ventana emergente preguntando si deseamos salir del programa, si se pulsa SI el programa se cerrara mientras que si pulsamos NO continuaremos en este. La ventana emergente será de la siguiente manera:

Capítulo IX
Caso Práctico

Capítulo IX

Caso Práctico

En este capítulo vamos a enseñar un posible uso naval para el programa que hemos propuesto, es decir, para el cálculo de la reconstrucción mediante SPLINES de una función discontinua dada. En este caso, vamos a analizar una unión entre dos semicuadernas para comprobar la precisión y la localización exacta de la unión entre estas, ya que para el diseño de buques sin quilla la unión directa de cuadernas debe estar bien definida para poder así realizar los cálculos con una mayor precisión en zonas donde la resistencia que ofrece el buque varía de manera importante, más si contamos que en ese punto exacto se partirá la corriente de agua que choca con nuestro caso. Además vamos a analizar un perfil Naca recreando su gráfica y sabiendo con precisión donde se encuentran los puntos donde se producen grandes cambios para poder anticiparse a los efectos hidrodinámicos que causan.

Todo esto se ve reducido en la capacidad de hacer al buque más eficiente partiendo de la unión entre sus cuadernas o de la forma de su perfil Naca.

También vamos a analizar una línea de costa para poder aproximarla con mucha mayor precisión que de una manera alisada lo que beneficiará a nuestro buque o al diseño de nuestras rutas de transporte o paso de buques.

9.1- Casos a analizar

El primer caso que vamos a analizar es el de la línea de costa, la cual vamos a analizar con nuestro algoritmo de interpolación, para así crear esta lo más fiel posible a la realidad, es decir con el mínimo error posible.

Tras este primer caso, vamos a analizar dos formas del buque: en primer lugar la unión entre dos semicuadernas y después analizaremos el perfil NACA como hemos indicado anteriormente. Para el primer caso, al ser las cartillas de trazado alisadas, no podemos tomar los valores de estas ya que no presentarían discontinuidad alguna, por ello debemos hacer las formas en Rhinoceros y de ahí, como se ha dicho anteriormente, sacar las coordenadas. Esto no pasara para la línea de costa ni para el análisis del perfil, que podremos analizar por puntos para poder interpolarlos posteriormente.

Una vez obtenidos los datos de entrada a modo de coordenadas, podemos comenzar a utilizar la interfaz gráfica.

Estos datos, para que puedan ser utilizados por la interfaz, deberán tener la siguiente forma:

LINEA DE COSTA

Para poder trabajar mediante SPLINES a trozos, es necesario un número de puntos muy elevado para que así el programa pueda localizar las discontinuidades, para nuestro caso hemos escogido 80 puntos del primer tramo de la línea de costa.

SEMICUADERNA

Vamos a tomar todas las medidas necesarias para el análisis de esta gráficamente, es decir, por medio de puntos de una cuaderna unida, lo cual nos definirá la zona de la discontinuidad y la forma de la cuaderna

PERFIL NACA

En este caso tomaremos los valores aproximados mediante una

9.2- Resolución de los casos propuestos

Pues bien, para la realización de los cálculos vamos a utilizar nuestra Interfaz Gráfica que hemos estado desarrollando. Esta reconstruirá el polinomio además de mostrar las potenciales discontinuidades de esquina que podamos tener en nuestro conjunto de datos a analizar. Para empezar, comenzamos iniciando nuestra interfaz gráfica mediante el comando “splines_cubicos_interpolantes_a_trozos”. Cuando la tengamos abierta, pulsaremos al botón “Comenzar” para que se nos muestre la ventana del programa que nos permite realizar el cálculo.

Una vez estemos en esta ventana, tendremos que indicar si son datos empíricos o de una función, en nuestro caso serán datos empíricos ya que han sido tomados de las formas del buque. Cuando hayamos escogido esta opción, podremos escoger si cargar o crear los datos con los que va a trabajar el programa, en nuestro caso le daremos a "Crear" y en la ventana emergente introduciremos los datos que hemos mostrado anteriormente. Las condiciones de contorno son siempre las mismas y por ello el programa las marca al inicio de este.

1- LINEA DE COSTA

Para este caso, al igual que en el anterior, escogeremos otra vez los datos que vienen de un experimento, ya que los valores de la línea de costa se tomarán en campo para una serie de puntos que vamos a interpolar. Una vez introducidos los datos como nos pide el programa, podremos ejecutarlo y ver la recreación de la línea de costa según los distintos polinomios interpolantes a trozos que se hayan creado, teniendo distintas ecuaciones en cada tramo. La línea de costa quedara de la siguiente manera:

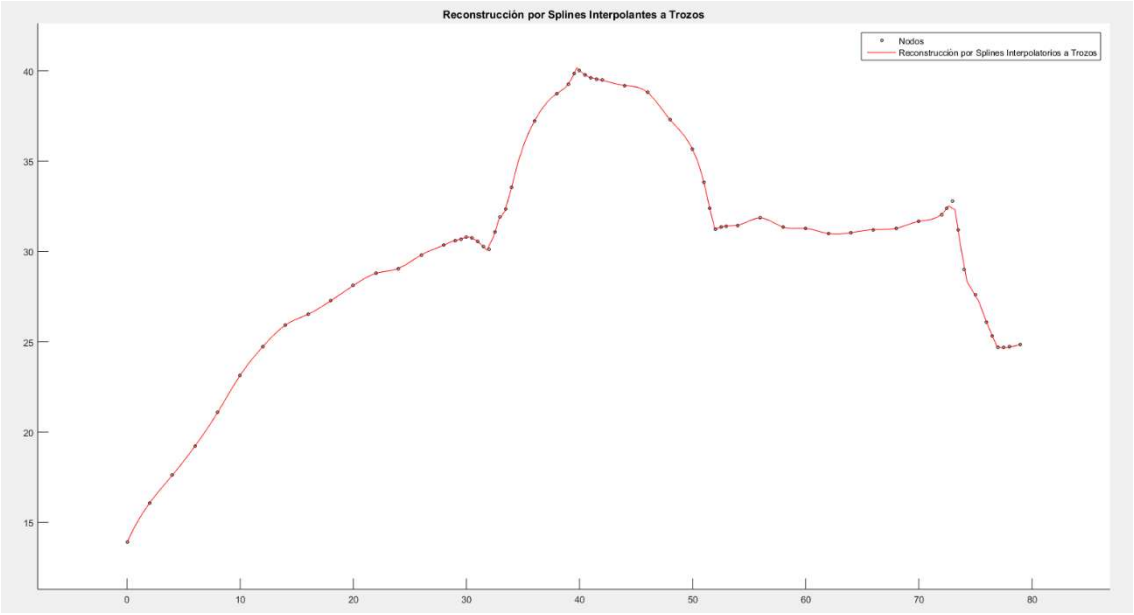
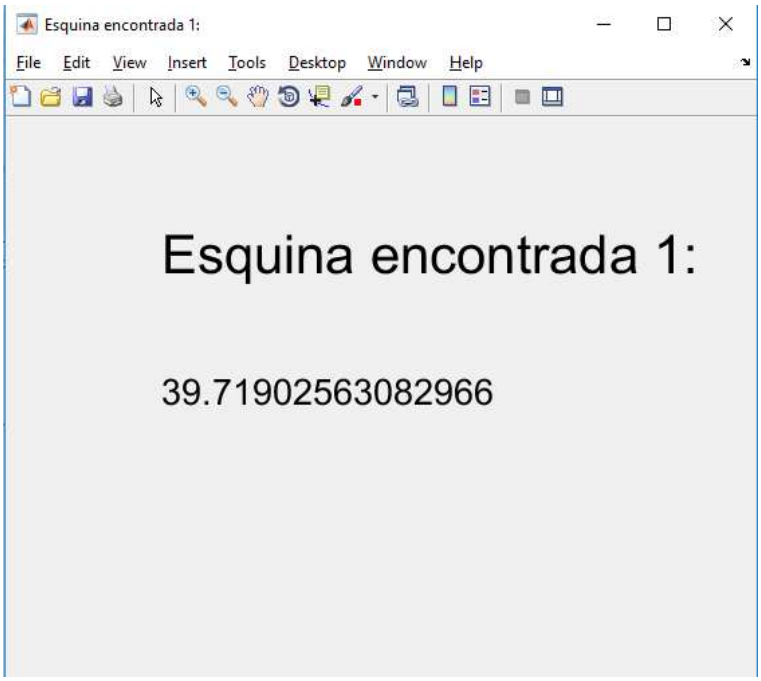


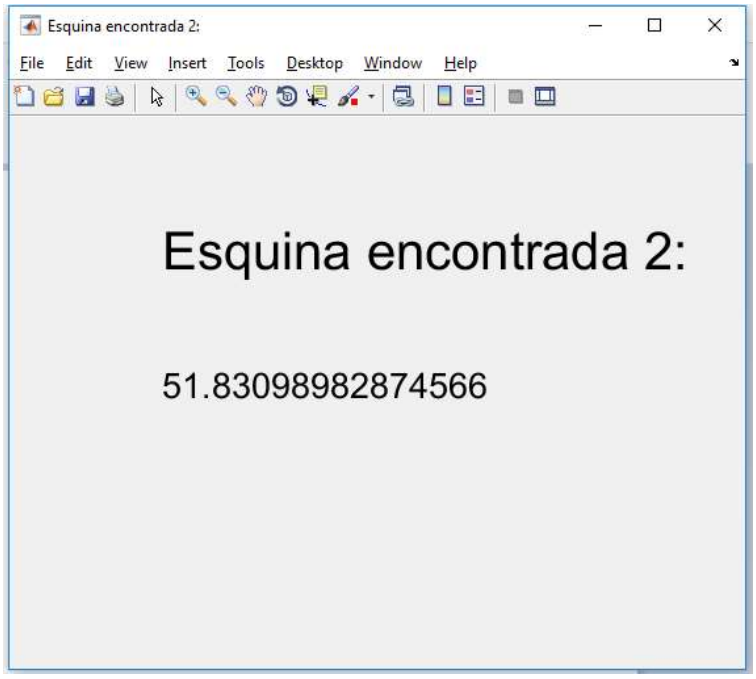
Fig.39 – Recreación Linea de Costa

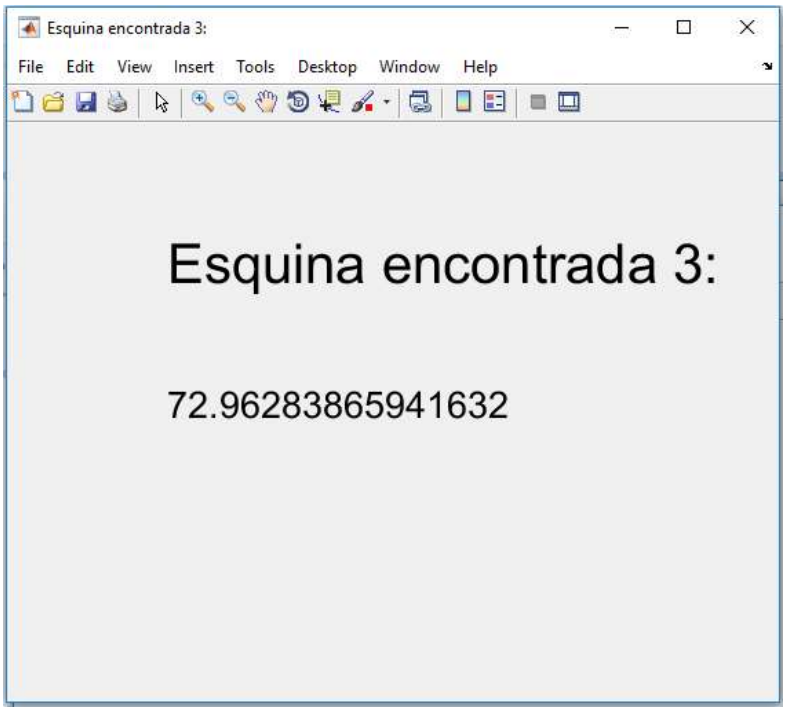
Como se puede comprobar, la precisión en las esquinas es muy elevada, haciendo una reconstrucción muy precisa de la zona de costa que hemos analizado.

Este programa además de darnos la reconstrucción nos indica donde se encuentran las discontinuidades presentes en la serie de datos que hemos introducido

por medio de una ventana emergente, en nuestro caso ha localizado tres discontinuidades notables:







Esto nos servirá ya que sabremos de manera precisa donde se encuentran las zonas más abruptas de la costa que hemos analizado.

Además, el programa nos mostrará los diversos polinomios que se han usado para recrear la línea de costa si usamos el botón Abrir Ficheros, que a su vez vienen divididos por cada uno de los tramos en los que las discontinuidades dividen la función. Estos polinomios son los que mostramos a continuación:

1- Primer Tramo:

```
+ 0.014162 x^3 - 0.170000 x^2 + 1.368351 x + 13.910000
+ 0.019189 x^3 - 0.200162 x^2 + 1.428675 x + 13.869784
+ 0.000919 x^3 + 0.041134 x^2 + 0.463489 x + 15.156699
+ 0.005736 x^3 + 0.078659 x^2 + 1.182247 x + 13.719183
+ 0.027026 x^3 + 0.707644 x^2 + 5.108175 x + 30.493642
+ 0.014869 x^3 + 0.549215 x^2 + 7.460413 x + 11.401653
+ 0.018700 x^3 + 0.659254 x^2 + 7.041206 x + 46.604825
+ 0.031179 x^3 + 1.435663 x^2 + 22.287622 x + 90.263037
+ 0.009768 x^3 + 0.529806 x^2 + 9.159882 x + 77.456982
+ 0.000858 x^3 + 0.048660 x^2 + 0.499243 x + 25.493148
+ 0.018051 x^3 + 1.080253 x^2 + 21.131106 x + 163.038903
+ 0.041812 x^3 + 2.870689 x^2 + 65.789614 x + 474.379714
+ 0.037946 x^3 + 2.871873 x^2 + 72.031866 x + 628.192132
+ 0.023722 x^3 + 1.938266 x^2 + 53.031751 x + 455.692552
+ 0.078358 x^3 + 6.636519 x^2 + 187.062233 x + 1785.184632
+ 0.290521 x^3 + 25.455998 x^2 + 743.620757 x + 7211.417602
+ 0.409969 x^3 + 36.537404 x^2 + 1085.184608 x + 10771.835155
+ 0.090644 x^3 + 7.798129 x^2 + 223.006358 x + 2150.052651
+ 0.372545 x^3 + 34.583693 x^2 + 1069.639205 x + 10991.843910
+ 0.359537 x^3 + 33.499981 x^2 + 1040.954679 x + 10817.626233
+ 2.665604 x^3 + 252.375819 x^2 + 7964.133027 x + 83735.794688
+ 2.782877 x^3 + 270.678305 x^2 + 8773.598943 x + 94800.012996
+ 0.814096 x^3 + 78.722148 x^2 + 2535.023853 x + 27215.449526
+ 3.239260 x^3 + 322.560104 x^2 + 10707.290479 x + 118450.008127
+ 2.382945 x^3 + 242.471542 x^2 + 8221.269674 x + 92918.913576
+ 0.064107 x^3 + 7.127825 x^2 + 265.108805 x + 3260.042520
+ 0.005829 x^3 + 0.833780 x^2 + 38.523174 x + 541.014947
+ 0.261056 x^3 + 29.929639 x^2 + 1144.165833 x + 14545.821956
+ 0.080693 x^3 + 8.827182 x^2 + 321.169999 x + 3846.876123
+ 0.199141 x^3 + 22.863301 x^2 + 875.596705 x + 11146.827752
```

2- Segundo Tramo:

```
- 0.022077 x^3 + 2.893097 x^2 - 126.091245 x + 1867.636197
- 0.037303 x^3 + 4.720238 x^2 - 199.176869 x + 2842.111188
+ 0.011040 x^3 + 1.153457 x^2 + 38.707783 x + 369.331613
+ 0.166857 x^3 + 20.727923 x^2 + 858.428828 x + 11891.535396
+ 0.016389 x^3 + 2.086259 x^2 + 88.359757 x + 1205.706697
+ 0.013042 x^3 + 1.664490 x^2 + 70.645430 x + 957.706113
+ 0.053981 x^3 + 7.182501 x^2 + 318.622184 x + 4751.552228
+ 0.069132 x^3 + 9.807057 x^2 + 462.897491 x + 7231.749456
+ 0.093796 x^3 + 13.654619 x^2 + 663.262920 x + 10786.817111
+ 0.120634 x^3 + 17.680318 x^2 + 864.547916 x + 14141.567044
+ 0.102015 x^3 + 14.831511 x^2 + 719.258753 x + 11671.651280
+ 0.112147 x^3 + 16.396960 x^2 + 799.879357 x + 13055.638317
```

3- Tercer Tramo:

```
+ 4.535186 x^3 - 996.498460 x^2 + 72983.018137 x - 1781651.609528
+ 0.913120 x^3 - 203.266109 x^2 + 15077.056550 x - 372606.544247
+ 3.725493 x^3 - 823.394404 x^2 + 60656.486234 x - 1489302.571499
+ 1.596347 x^3 + 358.054178 x^2 + 26770.708872 x + 667234.907785
+ 0.521441 x^3 + 118.448149 x^2 + 8966.965701 x + 226206.956527
+ 0.476812 x^3 + 109.153453 x^2 + 8330.756107 x + 212001.995927
+ 1.582340 x^3 + 363.421860 x^2 + 27821.255371 x + 709874.296756
+ 1.692549 x^3 + 393.077593 x^2 + 30429.202496 x + 785220.788498
+ 0.387857 x^3 + 90.616786 x^2 + 7057.111892 x + 183175.666523
+ 0.045561 x^3 + 10.803003 x^2 + 853.631665 x + 22503.665948
```

No solo nos da estos valores, si no los resultados de las interpolaciones en cada punto:

1- Primer Tramo:

```
0.000000 13.910000
0.556560 14.621352
1.085213 15.212846
1.593744 15.716329
2.156723 16.212500
2.653241 16.609741
3.184971 17.009595
3.738883 17.416276
4.246705 17.796467
4.814901 18.239417
5.345071 18.668967
5.870263 19.109119
6.370309 19.541339
6.927357 20.041273
7.450131 20.533230
8.004263 21.084371
8.517402 21.622433
9.055574 22.196014
9.567524 22.727745
10.096598 23.239545
10.647493 23.717374
11.138624 24.104905
11.671443 24.497105
12.203763 24.872440
12.745422 25.238293
13.298820 25.578158
13.827545 25.853476
14.318621 26.052834
14.866665 26.221761
15.403588 26.360807
15.927992 26.499265
16.470836 26.670059
16.999516 26.863170
17.512022 27.067351
18.049579 27.290767
18.557863 27.504185
19.139076 27.748758
19.626254 27.953495
20.153245 28.173898
20.697466 28.393924
21.218488 28.582282
21.764045 28.739225
22.286336 28.836269
22.848777 28.896768
23.377330 28.950126
23.861835 29.028394
24.428122 29.188500
24.933637 29.381801
25.471934 29.605748
26.008720 29.813054
26.559718 29.983048
27.062270 30.112028
27.622244 30.249225
28.116548 30.383708
28.667797 30.540547
29.196168 30.636177
29.719669 30.730785
30.258243 30.801261
30.786782 30.646210
31.291261 30.390359
31.818823 30.072164
32.394900 30.819592
32.881496 31.778100
33.404402 32.226950
33.962442 33.466339
34.509532 34.822379
35.028532 35.836322
35.533519 36.620209
36.073121 37.293374
36.608158 37.838278
37.165794 38.288322
37.652533 38.587489
38.219554 38.834224
38.738656 39.064380
39.254716 39.508404
```

2- Segundo Tramo:

39.775156	40.170321
40.317857	39.848719
40.850890	39.654410
41.362057	39.566284
41.916953	39.492692
42.428020	39.410914
42.966565	39.322406
43.507214	39.243248
44.019973	39.188362
44.552209	39.152398
45.099614	39.095774
45.611287	38.978294
46.170934	38.726602
46.709472	38.353639
47.226379	37.929523
47.736232	37.504389
48.279083	37.102774
48.784199	36.764904
49.356457	36.333395
49.885255	35.800924
50.412196	35.055716
50.913061	34.058611
51.460833	32.538590

3- Tercer Tramo:

73.203394	32.307172
73.713211	30.231258
74.263570	28.328479
74.795000	27.768718
75.292172	27.253140
75.832211	26.351924
76.360793	25.520089
76.914367	24.738022
77.453867	24.659861
77.957739	24.707947
78.510906	24.768433
79.000000	24.860000

Con todo esto, tenemos definida de una manera precisa y concreta toda la costa en la que pudiéramos operar con mayor facilidad, además de poder comprobar mejor las distintas geografías que compondrán nuestras rutas de comercio marítimo.

2- ANALISIS DE UNION DE CUADERNAS

En este apartado vamos a realizar el análisis de la unión de dos semicuadernas dadas en un plano de formas completo, es decir, analizaremos la unión que crea la cuaderna en la zona de la quilla, para buques que no presentan esta y que poseen una discontinuidad de esquina en esta zona que será la unión propiamente dicha.

Para nuestro caso, hemos tomado los valores de un plano de formas por medio de la introducción de este en rhino y posteriormente tomando puntos en la cuaderna

que hemos escogido, por ello, la unión no nos sale precisamente en 0, porque estamos analizando datos experimentales que no provienen del cálculo directo. Sin embargo, con esto demostramos que si introducimos los Splines Interpolantes a Trozos en nuestro método de cálculo para el diseño de nuestras cuadernas, tendremos que la zona en la cual se unen las cuadernas quedara totalmente definida, lo que significa que no tiene por qué ser necesario la creación de las cuadernas mediante espejo, se realizaba de esta manera ya que los Splines Suavizantes no detectan las discontinuidades que se generan en el modelado y por ello no son capaces de realizar la unión entre cuadernas de manera precisa, cosa que como vamos a mostrar con los cálculos siguientes es que con Splines Cúbicos Interpolantes a Trozos si se precisa la zona de unión.

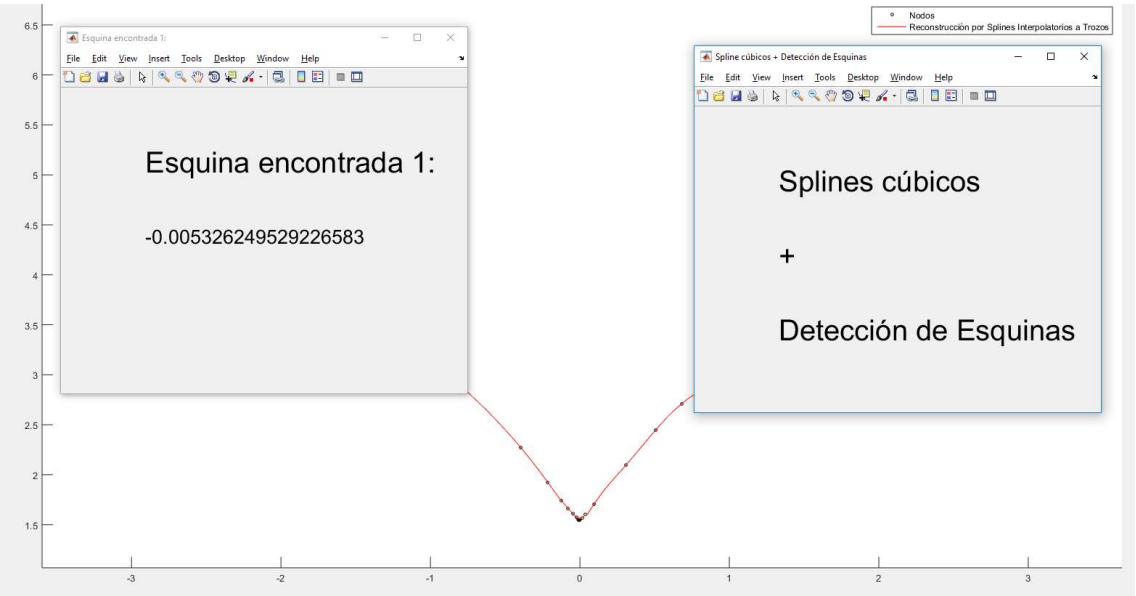


Fig.40 – Unión entre semicuadernas

Aquí vemos la unión gráficamente al igual que la localización exacta del punto en el cual se unen ambas partes de la cuaderna. A continuación vamos a mostrar los valores de los polinomios que ha hallado el programa para la creación de dicha cuaderna a base de puntos:

1- Primer Tramo:

```
 - 0.312504 x^3 - 1.157631 x^2 - 2.572764 x + 1.413374   - 0.299592 x^3 - 1.107726 x^2 -
2.508467 x + 1.440988   - 0.334939 x^3 - 1.218141 x^2 - 2.623439 x + 1.401083   - 0.148941 x^3
- 0.863666 x^2 - 2.398251 x + 1.448768   - 0.477362 x^3 - 1.251795 x^2 - 2.551149 x + 1.428690
 + 16.474745 x^3 + 9.668685 x^2 - 0.206172 x + 1.596537   + 20.675725 x^3 + 4.122424 x^2 -
1.912698 x + 1.526148   + 3.259180 x^3 + 1.549825 x^2 - 1.464616 x + 1.537947   + 4.810874 x^3
+ 0.406910 x^2 - 1.518571 x + 1.537098   + 0.755621 x^3 + 0.684918 x^2 - 1.512218 x + 1.537147
 + 3.382578 x^3 + 0.599491 x^2 - 1.513144 x + 1.537143
```

2- Segundo Tramo:

```
16.474745 x^3 + 9.668685 x^2 - 0.206172 x + 1.596537   + 20.675725 x^3 + 4.122424 x^2 - 1.912698 x
+ 1.526148   + 3.259180 x^3 + 1.549825 x^2 - 1.464616 x + 1.537947   + 4.810874 x^3 + 0.406910 x^2
- 1.518571 x + 1.537098   + 0.755621 x^3 + 0.684918 x^2 - 1.512218 x + 1.537147   + 3.382578 x^3 +
0.599491 x^2 - 1.513144 x + 1.537143
```

Con esto comprobamos que nuestro programa es capaz de diseñar una cuaderna completa por medio de una serie de puntos, a mayor precisión de los puntos mayor precisión en la reconstrucción de la cuaderna y mayor precisión en la localización de la esquina.

Esto implica que nuestro programa podría usarse para el diseño y el cálculo de formas del buque y qué nuestro algoritmo será mucho más preciso del que se suele usar en el diseño naval.

Con estos dos ejemplos vemos la funcionalidad de los Splines Cúbicos Interpolantes a trozos, pero no solo tienen estos dos usos sino que pueden usarse para facilitar y mejorar los cálculos en cualquier lugar en el que se presenten discontinuidades de esquina.

CONCLUSIONES

En conclusión con este trabajo se ha dado a conocer el uso de SPLINES de una manera distinta a la manera tradicional que es realizando el SPLINE para toda la función en lugar de a trozos. Esto nos ha permitido introducir funciones o series de datos que sabemos que pueden contener una discontinuidad, localizarla y realizar un ajuste mucho mejor que con el método tradicional. A parte de esto, se ha realizado diversos programas los cuales acaban convergiendo en una interfaz gráfica que te permite realizar los cálculos que hemos programado de una manera sencilla e intuitiva.

Uno de los objetivos de este Trabajo era la de la programación en MatLab de los SPLINES Cúbicos Interpolantes a Trozos para su correcta utilización a través de la Interfaz, esto nos ha permitido adquirir conocimientos tanto de la manera en la que se va a realizar la interpolación así como de los comandos que se requieren en MatLab para poder traducir la manera teórica a lenguaje de programación de este programa.

Con esto comprobamos que los SPLINES Cúbicos Interpolantes a Trozos tienen diversos usos en el ámbito naval, como hemos visto en los casos prácticos, ya sea para reconstrucción, para diseño, así como para elementos que nos afectan en el mundo náutico, como hemos mostrado, calculando de manera muy precisa una línea de costa de la cual solo tenemos sus puntos como referencia, esto nos puede permitir mejorar la llegada del Buque a puerto así como el propio diseño de los puertos que vaya a haber en dicha costa. Para el caso del diseño tenemos nuestro análisis de la unión de la cuaderna en una zona de esquina, pudiendo realizar diseños sin necesidad de semicuadernas. Por último hemos demostrado la utilizad de estos SPLINES para cálculos que pueden darnos más información acerca del perfil NACA escogido, lo que implica un mayor conocimiento del comportamiento de dicho perfil según las circunstancias que lo atañen.

BIBLIOGRAFÍA

FICATIER, Bernard; ROCHE, Hugues; ANGELI, Jean (2004). Concevoir, reveler et dessiner des plans de voiliers classiques et traditionnels. Le Chasse-Marée.

Programas Informáticos Navales. [Apuntes] José Alfonso Martínez García. Departamento de Tecnología Naval (ETSINO-Universidad Politécnica de Cartagena).

Matemáticas Asistidas por Computador. [Apuntes] Antonio Escudero Vergara. Departamento de Matemática Aplicada y Estadística (ETSINO-Universidad Politécnica de Cartagena).

CORDERO BARBERO, A.; MOLADA MARTÍNEZ, E.; HUESO PAGOAGA, J.L.; TORREGROSA SÁNCHEZ, J.R. Métodos numéricos con Matlab. Editorial Universidad Politécnica de Valencia.

S. Amat, K. Dadourian, J . Liandrat, J. Ruiz, J. C. Trillo, On a class of LI-stable nonlinear cell-average multiresolution schemes, J. Comput. Appl. Math. 234 (4) (2010) 1129--1139.

S. Amat, J . Liandrat, J. Ruiz, J. 'Ih llo, On a family of nonlinear cell-average multiresolution schemes for image processing, Math. Comput. Simul. 118 (C) (2015) 30-48.

Dibujo Naval. [Apuntes] Leandro Ruíz Peñalver. Departamento de Tecnología Naval (ETSINO-Universidad Politécnica de Cartagena).