APROXIMACIONES POR MÍNIMOS CUADRADOS PONDERADOS EN EL ENTORNO DE MULTIRRESOLUCIÓN POR VALORES PUNTUALES.

Autor: Karima Torkzi
Director: Juan Carlos Trillo Moya

Noviembre del 2011

A toda mi familia, en especial a mis padres, por su paciencia, colaboración y apoyo,

A esas personas que han puesto su granito de arena en estos años, por pequeño que haya sido,

A mi director Juan Carlos, por su ayuda y predisposición,

A todos vosotros,

¡Mil Gracias!

Autor	Karima Torkzi
E-mail del Autor	koi_faire@hotmail.com
Directores	Juan Carlos Trillo Moya
E-mail del Director	jctrillo@upct.es
Título del PFC	Aproximaciones por Mínimos Cuadrados Ponderados en el entorno de Multirresolución por Valores Puntuales.
Descritores	Compresión de datos, reconstrucciones, multirresolución
Resúmen	Estudio de los algoritmos de multirresolución por valores puntuales. Estudio de las aproximaciones por mínimos cuadrados ponderados. Incorporación de estas aproximaciones como operadores de reconstrucción y predicción en la multirresolución de Harten. Programación en Matlab de los algoritmos mencionados anteriormente. Análisis del comportamiento de las aproximaciones por mínimos cuadrados ponderados cuando se usan como operadores de reconstrucción-predicción en la multirresolución de Harten. Aplicación de los algoritmos resultantes a diferentes problemas en 1D y en 2D (compresión de señales).
Titulación	Ingeniero Técnico de Telecomunicaciones, Especialidad Telemática
Departamento	Matemática Aplicada y Estadística
Fecha de Presentación	Noviembre de 2011

Índice general

Índice de Tablas

Índice de Figuras

Capítulo 1

Introducción.

Las transformaciones de multirresolución son una de las herramientas más eficaces para la compresión de señales, de imágenes, y de datos en general. Proporcionan algoritmos rápidos y buenos resultados en comparación con otras aproximaciones clásicas como los métodos basados en la transformada de Fourier TF.

En este proyecto se lleva a cabo el estudio de los algoritmos de Multirresolución de Harten para el caso de discretizacón por valores puntuales.

El objetivo del enfoque propuesto por Harten es la construcción de esquemas de multirresolución adaptados a cada proceso de discretización. El paso fundamental en la construcción de un esquema de multirresolución es la definición de un operador de reconstrucción apropiado para la discretización que se está considerando.

Al utilizar técnicas de interpolación independientes de los datos, es decir, lineales, la capacidad de compresión del esquema de multirresolución se verá reducida. Por otra parte, al utilizar técnicas de interpolación que dependan de los datos, es decir, no lineales, la capacidad de compresión del esquema de multirresolución mejora.

Estudiaremos el operador de predicción lineal basado en la interpolación de Lagrange, así como otros operadores de predicción que vendrán de considerar distintas funciones de ponderación. En particular estudiaremos los filtros que vienen de las funciones Bisquare, Exponencial, Gaussiana, Huber, Spline Cuártico, Spline Cúbico, Uniforme y Lagrange. Analizaremos también cómo puede influir un cambio de datos no lineal en los resultados obtenidos al trabajar con datos que presentan discontinuidades. En particular realizaremos un cambio de datos basado en el operador local no lineal de reconstrucción PPH (ver [11]).

La memoria está organizada como sigue. En la Sección 2 se desarrollan los contenidos acerca de la Multirresolución de Harten en el entorno de valores

puntuales. También se estudian los métodos de reconstrucción de Lagrange y PPH, el paso a $2D$ a través del producto tensor, y cómo hacer compresión de datos. En la Sección 3 se definen las funciones de ponderación utilizadas y se obtienen los correspondientes filtros numéricos. En la Sección 4 se detallan los códigos programados en Matlab que han sido utilizados. En la Sección 5 se desarrolla un tutorial de la Interfaz Gráfica, así como algunos ejemplos acerca del uso de la misma. En la Sección 6 se llevan a cabo experimentos numéricos tanto en 1D como en 2D. Y por último, en la Sección 7 exponemos las conclusiones obtenidas en este proyecto.

Capítulo 2

Multirresolución de Harten.

El objetivo de la multirresolución es obtener una reordenación multiescala de la información contenida en un conjunto de datos discretos a una cierta resolución. Por ejemplo, esta información puede ser el resultado de discretizar una función, denotada f, en un cierto espacio vectorial V^k, en el que k indica el nivel de resolución. Un mayor valor de k indica una mayor resolución. Para realizar la transición entre distintos niveles de resolución se utilizan dos operadores llamados decimación y predicción. El operador decimación proporciona información discreta a un nivel de resolución $k-1$ a partir de la información contenida en el nivel k:

$$D_k^{k-1} : V^k \to V^{k-1},$$

y debe ser lineal y sobreyectivo.

El operador predicción actúa en sentido opuesto, dando una aproximación a la información discreta en el nivel k a partir de la información contenida en el nivel $k-1$:

$$P_{k-1}^k : V^{k-1} \to V^k.$$

Además, al operador predicción no se le exige que sea lineal.

Los datos discretos se obtienen a partir de la discretización de una función f, para lo cual existen distintos operadores. Dependiendo del operador discretización utilizado, la secuencia de datos f^k que resulta es diferente. El objetivo del enfoque propuesto por Harten es la construcción de esquemas de multirresolución adaptados a cada proceso de discretización. Esto se consigue definiendo un operador reconstrucción apropiado. Estos operadores, discretización y reconstrucción, son los elementos a partir de los cuales se construyen los operadores decimación y predicción del esquema de multirresolución.

Formalmente, sea F un espacio de funciones:

$$F \subset \{f | f : \Omega \subset \mathbb{R}^m \longrightarrow \mathbb{R}\}$$

El operador discretización asigna a cada elemento de este espacio, $f \in F$, una secuencia f^k de datos discretos perteneciente al espacio V^k. De este modo se define el operador discretización:

$$D_k : F \to V^k$$

que ha de ser lineal y sobreyectivo y que a cada $f \in F$ le asocia:

$$f^k = D_k(f).$$

La reconstrucción opera en sentido inverso, tomando una secuencia de datos discretos para reconstruir, a partir de la información proporcionada por dichos datos, la función de la que provienen:

$$R_k : V^k \to F$$

La principal novedad introducida por Harten consiste en que a este operador reconstrucción no se le exige que sea lineal.

Por motivos de consistencia, se requiere que los operadores discretización y reconstrucción satisfagan la siguiente condición:

$$D_k R_k f^k = f^k, \forall f^k \in V^k$$

o expresado de otro modo:

$$D_k R_k = I_{V^k}.$$

es decir, si tomamos la información reconstruida a partir de unos datos discretos con una cierta resolución y la discretizamos a ese mismo nivel de resolución, la información discreta obtenida coincide con la original.

En la Figura 2.1 se muestran las relaciones existentes entre los operadores discretización y reconstrucción, y los operadores decimación y predicción. Según estas relaciones, el operador decimación se define del siguiente modo:

$$D_k^{k-1} := D_{k-1} R_k,$$

Aunque aparentemente el operador decimación depende de la elección del operador reconstrucción, en realidad no es así si (y sólo si) la sucesión de operadores discretización $\{D_k\}$ es anidada, es decir, si se tiene:

$$D_k f = 0 \Longrightarrow D_{k-1} f = 0, \ \forall f \in F$$

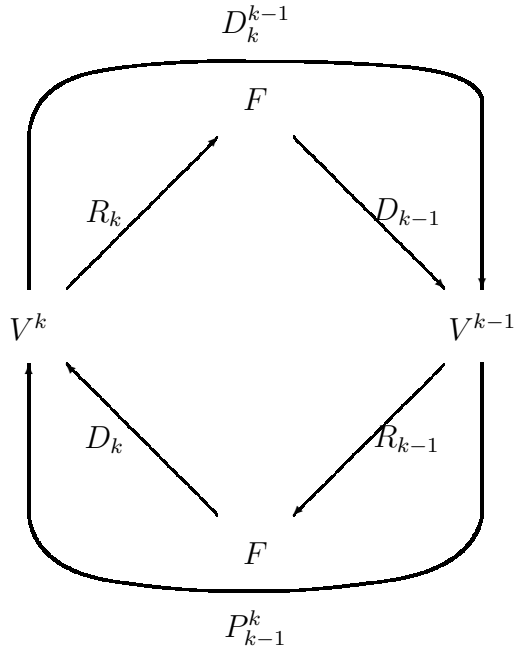

Figura 2.1: Definición de operadores.

La propiedad de anidamiento significa que la información contenida en los datos a un cierto nivel de resolución k no será nunca mayor que la información contenida en un nivel de resolución superior.
De forma similar, el operador predicción se construye según la expresión:

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

A partir de estas definiciones se obtiene la siguiente relación de consistencia para los operadores decimación y predicción:

$$D_k^{k-1} P_{k-1}^k = D_{k-1} R_k D_k R_{k-1} = D_{k-1} R_{k-1} = I_{V^{k-1}}.$$

Esta última relación lo que significa es que cuando utilizamos estos operadores no inventamos información, es decir, si decimamos la información obtenida a partir de la predicción realizada sobre una información con resolución dada por V^{k-1}, obtenemos exactamente la misma información de partida, sin introducir ningún elemento nuevo.
Consideremos ahora f^k, la información discreta en el nivel de resolución k. Si aplicamos el operador decimación sobre f^k obtenemos f^{k-1}, es decir, la información contenida en el nivel de resolución $k-1$:

$$f^{k-1} = D_k^{k-1} f^k$$

En este caso, podemos interpretar que $P_{k-1}^k f^{k-1}$ es una aproximación a f^k, con un error:

$$e^k := \left(I_{V^k} - P_{k-1}^k D_k^{k-1}\right) f^k =: Q_k f^k \in V^k$$

De esta forma podemos representar la información contenida en f^k en la forma ya descrita, y recíprocamente, conociendo f^{k-1} y e^k se puede calcular f^k mediante la expresión $P_{k-1}^k f^{k-1} + e^k = f^k$.

El problema es que haciendo esto incluimos información redundante, ya que, si suponemos que V^k es un espacio de dimensión finita (como lo es habitualmente en la práctica), $dim\ V^k = N_k$, tenemos por un lado f^k, que contiene la información codificada en N_k elementos, mientras que $\left\{f^{k-1}, e^k\right\}$ contine la misma información codificada en $N_{k-1} + N_k$ elementos. Esta información redundante puede ser eliminada, como consecuencia del siguiente resultado:

$$\begin{aligned} D_k^{k-1} e^k &= D_k^{k-1}\left(I_{V^k} - P_{k-1}^k D_k^{k-1}\right) v^k, && (2.1)\\ &= D_k^{k-1} v^k - D_k^{k-1} P_{k-1}^k D_k^{k-1} v^k, && (2.2)\\ &= D_k^{k-1} v^k - D_k^{k-1} v^k = 0. && (2.3) \end{aligned}$$

es decir, $e^k \in N\left(D_k^{k-1}\right) = \left\{f^k \in V_k : D_k^{k-1} f^k = 0\right\}$, cuya dimensión es $dimN\left(D_k^{k-1}\right) = dimV^k - dimV^{k-1} = N_k - N_{k-1}$.

Sea $\left\{\mu_i^k\right\}$ el conjunto de elementos que generan el espacio $N\left(D_k^{k-1}\right)$. Podemos expresar el error e^k como:

$$e^k = \sum d_i^k \mu_i^k,$$

Si definimos un operador G_k que asocie a cada elemento $e^k \in N\left(D_k^{k-1}\right)$ su correspondiente conjunto de coeficientes $\left\{d_i^k\right\}$, podemos establecer la siguiente equivalencia:

$$f^k \equiv \left\{f^{k-1}, d^k\right\}$$

mediante las relaciones:

$$\begin{aligned} f^{k-1} &= D_k^{k-1} f^k, \\ d^k &= G_k\left(I - P_{k-1}^k D_k^{k-1}\right) f^k, \end{aligned}$$

para obtener $\left\{f^{k-1}, d^k\right\}$ a partir de f^k, y :

$$P_{k-1}^k f^{k-1} + E_k d^k = f^k.$$

en sentido inverso.

En este caso la equivalencia de información lo es también en cuanto a número de elementos utilizados para codificar dicha información, ya que en $\{f^{k-1}, d^k\}$ tenemos $N_{k-1}+(N_k - N_{k-1}) = N_k$ elementos, los mismos que hay en f^k. Decimos entonces que los coeficientes $\{d_i^k\}$ contienen la información no redundante del error de predicción, y serán llamados detalles.

Iterando este procedimento en cada nivel de resolución, se consigue la descomposición multiescala que se muestra en la Figura 2.2, y que permite establecer la siguiente equivalencia:

$$f^L \equiv \{f^0, d^L, \ldots, d^1\}$$

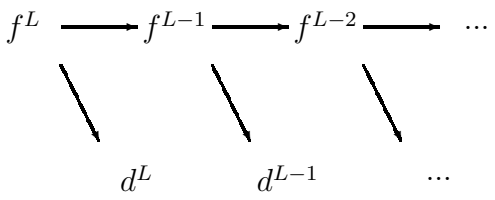

Figura 2.2: Descomposición multiescala.

Por tanto, y para resumir, los algoritmos para las transformaciones directa e inversa de la multirresolución son los siguientes:

(Directa)

$$f^L \to Mf^L = \{f^0, d^1, \ldots, d^L\} \begin{cases} \text{Do}, \quad k = L, \ldots, 1 \\ f^{k-1} = D_k^{k-1} f^k, \\ d^k = G_k(f^k - P_{k-1}^k f^{k-1}). \end{cases} \tag{2.4}$$

e

(Inversa)

$$Mf^L \to M^{-1}Mf^L \begin{cases} \text{Do}, \quad k = 1, \ldots, L \\ f^k = P_{k-1}^k f^{k-1} + E_k d^k. \end{cases} \tag{2.5}$$

El paso fundamental en la construcción de un esquema de multirresolución es la definición de un operador reconstrucción apropiado para la discretización que se está considerando.

2.1. MR para la Discretización por Valores Puntuales en [0,1].

La multirresolución para la discretización por valores puntuales en el intervalo [0,1], no es más que un caso particular de

Se considera el conjunto de redes anidadas en el intervalo [0,1] dado por:

$$X^k = \{x_j^k\}_{j=0}^{J_k}, \quad x_j^k = jh_k, \quad h_k = 2^{-k}/J_0, \quad J_k = 2^k J_0,$$

donde J_0 es un entero fijo y X^k una partición uniforme en el intervalo unidad cerrado. La discretización por valores puntuales viene dada por:

$$D_k : \begin{cases} C([0,1]) & \to V^k \\ f & \mapsto f^k = (f_j^k)_{j=0}^{J_k} = (f(x_j^k))_{j=0}^{J_k} \end{cases} \tag{2.6}$$

donde V^k es el espacio de las secuencias reales de dimensión $J_k + 1$. Un operador de reconstrucción para esta discretización es cualquier operador R_k tal que:

$$R_k : V^k \to C([0,1]); \quad \text{y satisface} \quad D_k R_k f^k = f^k, \tag{2.7}$$

lo cual significa que:

$$(R_k f^k)(x_j^k) = f_j^k = f(x_j^k). \tag{2.8}$$

En otras palabras, $(R_k f^k)(x)$ es una función continua que interpola los datos f^k en X^k.

Si se escribe $(R_k f^k)(x) = I_k(x; f^k)$, entonces uno puede definir las transformadas directa (2.4) e inversa (2.5) de la multirresolución como:

$$f^L \to Mf^L \begin{cases} \text{Do} \quad k = L, \ldots, 1, & \\ \quad f_j^{k-1} = f_{2j}^k & 0 \le j \le J_{k-1}, \\ d_j^k \quad = f_{2j-1}^k - I_{k-1}(x_{2j-1}^k; f^{k-1}) & 1 \le j \le J_{k-1}. \end{cases} \tag{2.9}$$

y

$$Mf^L \to M^{-1}Mf^L \begin{cases} \text{Do} \quad k = 1, \ldots, L \\ f_{2j}^k = f_j^{k-1} & 1 \le j \le J_{k-1}, \\ \quad f_{2j-1}^k = I_{k-1}(x_{2j-1}^k; f^{k-1}) + d_j^k & 0 \le j \le J_{k-1}. \end{cases} \tag{2.10}$$

Podemos pensar en el análisis de multirresolución como un análisis de la regularidad de una función. Los mayores coeficientes d_j^k van asociados a las singularidades de la función, lo que significa que no podemos predecir la información contenida en esas regiones. Más importante aún es el hecho de que si utilizamos una técnica de interpolación independiente de los datos, el conjunto de los intervalos afectados por una singularidad no se reduce únicamente a aquel intervalo en el que se localiza dicha singularidad, sino a todos los intervalos cuyo stencil contenga el intervalo donde se encuentra la singularidad, por lo que habrá una mayor cantidad de coeficientes con valores significativos, y la capacidad de compresión del esquema de multirresolución se verá reducida. Esto es lo que ocurre cuando se utiliza interpolación lineal centrada.

Por otra parte, al utilizar técnicas de interpolación que dependan de los datos, es decir, no lineales, se minimiza la zona afectada por cada singularidad, y la capacidad de compresión del esquema de multirresolución mejora. Esto ocurre cuando se utilizan las técnicas no lineales.

Las técnicas de interpolación más usuales son los polinomios.

2.2. Reconstrucciones Lineales y No Lineales.

En esta sección vamos a presentar algunas reconstrucciones lineales y no lineales usadas en el entorno de multirresolución de Harten. Las presentamos dentro del entorno de valores puntuales. Es en este marco de valores puntuales donde la definición de la reconstrucción es más clara y por eso hemos preferido esta presentación.
En primer lugar presentamos un operador de reconstrucción lineal basado en interpolación de Lagrange, para seguidamente presentar un operador no

lineal cuyo objetivo es la mejora de los puntos débiles de esta reconstrucción.

2.2.1. Reconstrucción Lineal: Lagrange.

Consideremos los valores de la función $f_{j-1}, f_j, f_{j+1}, f_{j+2}$ que se corresponden con las abscisas $x_{j-1}, x_j, x_{j+1}, x_{j+2}$ de una malla regular X y vamos a describir cómo construir un trozo polinómico del operador de reconstrucción y la predicción $\hat{f}_{2j+1}$ en el punto medio $\frac{x_j+x_{j+1}}{2} = x_{j+\frac{1}{2}}$. La reconstrucción en el intervalo $[x_j, x_{j+1}]$ viene dada por

$$P_j(x) = f_{j-1}L_{-1}(x) + f_jL_0(x) + f_{j+1}L_1(x) + f_{j+2}L_2(x) \tag{2.11}$$

donde $L_i(x)$ $i = -1, 0, 1, 2$ son los polinomios de Lagrange dados por

$$L_i(x) = \prod_{l=-1, l\neq i}^{2} \frac{(x - x_{j+l})}{(x_{j+i} - x_{j+l})}. \tag{2.12}$$

$\hat{f}_{2j+1}$ se define como la evaluación en $x_{j+\frac{1}{2}}$ del polinomio de Lagrange $P_j(x)$, es decir

$$P_j(x_{j+\frac{1}{2}}) = f_{j-1}L_{-1}(x_{j+\frac{1}{2}}) + f_jL_0(x_{j+\frac{1}{2}}) + f_{j+1}L_1(x_{j+\frac{1}{2}}) + f_{j+2}L_2(x_{j+\frac{1}{2}}), \tag{2.13}$$

y haciendo algunos cálculos

$$P_j(x_{j+\frac{1}{2}}) = -\frac{1}{16}f_{j-1} + \frac{9}{16}f_j + \frac{9}{16}f_{j+1} - \frac{1}{16}f_{j+2}. \tag{2.14}$$

Al conjunto de valores $\{-\frac{1}{16}, \frac{9}{16}, \frac{9}{16}, -\frac{1}{16}\}$ se le denomina máscara del operador de predicción basado en interpolación segmentaria de Lagrange centrada.

En el caso general, podemos llegar fácilmente a una expresión similar a (2.11). Para ello, vamos a considerar el polinomio interpolador de Lagrange de grado $r = 2s - 1$ basado en el conjunto de $2s$ puntos dado por $\{x_{j-s+1}, \ldots, x_j, x_{j+1}, \ldots, x_{j+s}\}$ y sus correspondientes valores de función $\{f_{j-s+1}, \ldots, f_j, f_{j+1}, \ldots, f_{j+s}\}$. En este caso el operador de reconstrucción estará formado por la unión de trozos polinómicos del tipo

$$P_j(x) = \sum_{i=-s+1}^{s} f_{j+i}L_i(x_{j+\frac{1}{2}}), \quad x \in [x_j, x_{j+1}], \tag{2.15}$$

donde los polinomios de Lagrange $L_i(x)$ se definen como en (2.12) por

$$L_i(x) = \prod_{l=-s+1, l\neq i}^{s} \frac{(x - x_{j+l})}{(x_{j+i} - x_{j+l})}. \tag{2.16}$$

Tendremos que el operador de predicción en el punto medio toma la forma

$$P_j(x_{j+\frac{1}{2}}) = \sum_{i=-s+1}^{s} f_{j+i} L_i(x_{j+\frac{1}{2}}). \tag{2.17}$$

La máscara del operador de predicción en este caso es $\{L_i(x_{j+\frac{1}{2}})\}_{i=-s+1}^{i=s}$. En la Tabla 2.1 podemos ver los valores de las máscaras para $s = 2, 3, 4$.

	Máscaras
$s = 2$	$\{\frac{-1}{16}, \frac{9}{16}, \frac{9}{16}, \frac{-1}{16}\}$
$s = 3$	$\{\frac{3}{256}, \frac{-25}{256}, \frac{150}{256}, \frac{150}{256}, \frac{-25}{256}, \frac{3}{256}\}$
$s = 4$	$\{\frac{-5}{2048}, \frac{49}{2048}, \frac{-245}{2048}, \frac{1225}{2048}, \frac{1225}{2048}, \frac{-245}{2048}, \frac{49}{2048}, \frac{-5}{2048}\}$

Tabla 2.1: Máscaras del operador de predicción basado en interpolación segmentaria de Lagrange centrada con $2s$ puntos para $s = 2, 3, 4$.

Para más detalles sobre esta reconstrucción y sobre las propiedades de los esquemas de subdivisión y multirresolución asociados se puede consultar [10].

2.2.2. Reconstrucción No Lineal: PPH.

Vamos a definir como construir un trozo polinómico PPH de orden $2s$ para el intervalo $[x_j, x_{j+1}]$. Partimos de los $2s$ datos

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

y lo que vamos a hacer es modificar algunos de ellos para suavizar la función de manera que luego podamos aplicar interpolación de Lagrange sobre datos sin discontinuidades apreciables. Esta modificación es hecha también teniendo en cuenta que nuestro objetivo es dar una aproximación al valor de la función en el punto medio.

Definimos los coeficientes B_{s-1}^i por medio de la siguiente recurrencia

$$B_{s-1}^{s-1} = 2, \tag{2.18}$$

$$B_{s-1}^{q} = \sum_{j=1}^{s-1-q} (-1)^j \left(\binom{2(q+j)}{(j-1)} - \binom{2(q+j)}{j} \right) B_{s-1}^{q+j}, \tag{2.19}$$

para $q = s-2, \ldots, 1$.

En la Tabla 2.2.2 podemos ver el valor de estos coeficientes para $s = 2, 3, 4$.

	B^i_{s-1}
$s=2$	$\{2\}$
$s=3$	$\{2, 6\}$
$s=4$	$\{2, 10, 12\}$

Tabla 2.2: Coeficientes B^i_{s-1}, $s = 2, 3, 4$.

También vamos a necesitar las medias p-power $power_p(x, y)$, que se introdujeron en [28] para cualquier número entero $p \geq 1$, y cualquier pareja (x, y) como:

$$power_p(x,y) = \frac{sign(x)+sign(y)}{2}\frac{|x+y|}{2}\left(1-\left|\frac{x-y}{x+y}\right|^p\right). \quad (2.20)$$

Igualmente necesitaremos usar las diferencias divididas, que es conocido que actúan como indicadores de zonas de suavidad de la función. Las diferencias divididas en el caso de nodos igualmente espaciados pueden calcularse haciendo uso del famoso triángulo de Tartaglia. En nuestro caso, donde nos interesa comparar el tamaño absoluto de dichas diferencias para detectar las zonas de suavidad, podremos prescindir de los denominadores. Y por tanto, su cálculo se reduce al de las diferencias finitas del mismo orden.

Llevaremos a cabo una modificación progresiva de los datos de la siguiente manera (ver [11] y [9] para más detalles). Observamos que esta modificación está diseñada para mantener el orden de interpolación en las regiones convexas suaves en el momento de aplicar la interpolación de Lagrange.

Modificación de datos de entrada $f \to \tilde{f}$

- Paso 1

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

Si $|\Delta^2_{j-1}f| \leq |\Delta^2_j f|$ entonces

$$\tilde{f}_{j+2} := f_{j+1} + f_j - f_{j-1} + B^1_1 pow_{2s-2}(\Delta^2_{j-1}f, \Delta^2_j f),$$

en otro caso

$$\tilde{f}_{j-1} := f_{j+1} + f_j - f_{j+2} + B^1_1 pow_{2s-2}(\Delta^2_{j-1}f, \Delta^2_j f).$$

Así, definimos

$$\tilde{f} := \begin{cases} (\ldots, f_{j-2}, f_{j-1}, f_j, f_{j+1}, \tilde{f}_{j+2}, f_{j+3}, \ldots) & \text{si } |\Delta^2_{j-1}f| \leq |\Delta^2_j f|, \\ (\ldots, f_{j-2}, \tilde{f}_{j-1}, f_j, f_{j+1}, f_{j+2}, f_{j+3}, \ldots) & \text{en otro caso.} \end{cases} \tag{2.21}$$

- Paso 2

  Consideramos $\{f_{j-2}, \tilde{f}_{j-1}, \tilde{f}_j, \tilde{f}_{j+1}, \tilde{f}_{j+2}, f_{j+3}\}$.
  Si $|\Delta^4_{j-1}\tilde{f}| \leq |\Delta^4_j \tilde{f}|$ entonces

  $$\tilde{f}_{j+3} := f_{j+1} + f_j - f_{j-2} + B^1_2 pow_{2s-2}(\Delta^2_{j-1}f, \Delta^2_j f) + B^2_2 pow_{2s-4}(\Delta^4_{j-2}\tilde{f}, \Delta^4_{j-1}\tilde{f}),$$

  en otro caso

  $$\tilde{f}_{j-2} := f_{j+1} + f_j - f_{j+3} + B^1_2 pow_{2s-2}(\Delta^2_{j-1}f, \Delta^2_j f) + B^2_2 pow_{2s-4}(\Delta^4_{j-2}\tilde{f}, \Delta^4_{j-1}\tilde{f}).$$

  Así, definimos

  $$f = \begin{cases} (\ldots, f_{j-2}, f_{j-1}, f_j, f_{j+1}, \tilde{f}_{j+2}, f_{j+3}, \ldots) & \text{si } |\Delta^2_{j-1}f| \leq |\Delta^2_j f|, \\ (\ldots, f_{j-2}, \tilde{f}_{j-1}, f_j, f_{j+1}, f_{j+2}, f_{j+3}, \ldots) & \text{en otro caso.} \end{cases} \tag{2.22}$$

  y definimos

  $$\tilde{f} := \begin{cases} (\ldots, f_{j-3}, f_{j-2}, \tilde{f}_{j-1}, \tilde{f}_j, \tilde{f}_{j+1}, \tilde{f}_{j+2}, \tilde{f}_{j+3}, f_{j+4} \ldots) & \text{si } |\Delta^4_{j-2}\tilde{f}| \leq |\Delta^4_{j-1}\tilde{f}|, \\ (\ldots, f_{j-3}, \tilde{f}_{j-2}, \tilde{f}_{j-1}, \tilde{f}_j, \tilde{f}_{j+1}, \tilde{f}_{j+2}, f_{j+3}, f_{j+4} \ldots) & \text{en otro caso.} \end{cases} \tag{2.23}$$

  Y así se realizarán sucesivos pasos hasta completar la modificación de los $2s$ puntos.

Aplicando la interpolación de Lagrange de orden $2s$ con $\tilde{f}$, a los datos de entrada modificados en el apartado de arriba, obtenemos la interpolación no lineal deseada.

Por contrucción, esta técnica de interpolación no lineal nos lleva a un operador de reconstrucción con muchas características deseables.Primero, cada

pieza polinómica se construye con un stencil de $2s$ puntos. Segundo, la reconstrucción es tan precisa como su equivalente lineal en las regiones convexas suaves. Tercero, la precisión se reduce según se acerca a las singularidades, pero no se pierde totalmente como ocurre en su contraparte lineal. En particular, las reconstrucciones están libres de los efectos de Gibbs.

Por claridad vamos a estudiar unos casos partículares. Supongamos que tenemos cuatro puntos dispuestos como en la Figura 2.3. En el primer paso miraremos qué diferencia dividida de segundo orden es mayor, y a continuación cambiaremos el valor correspondiente. Sólo nos quedará entonces llevar a cabo una interpolación de Lagrange con estos datos como queda representado en la Figura 2.4.

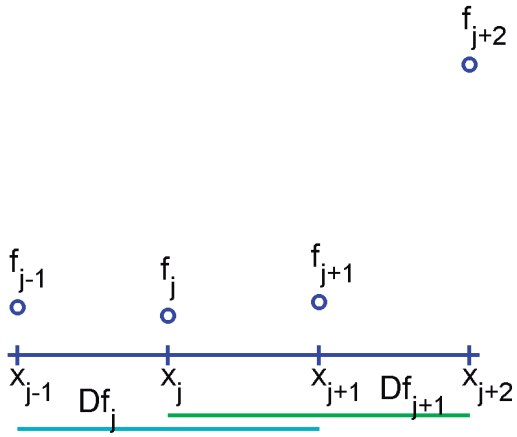

Figura 2.3: Primer paso de la reconstrucción PPH con 4 puntos.

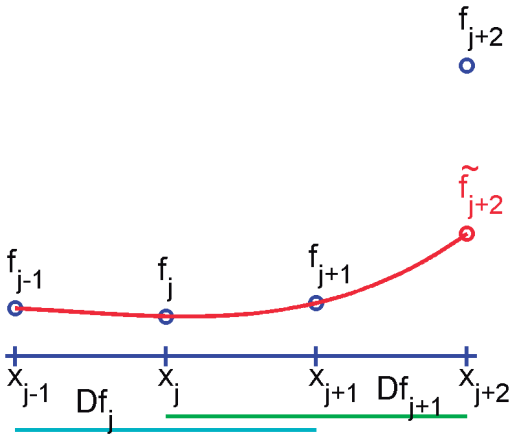

Figura 2.4: Construcción de la reconstrucción PPH con 4 puntos con los valores modificados.

En el siguiente ejemplo vamos a ilustrar de manera gráfica como se realizará la modificación de los datos en el caso de tener los valores de la Figura 2.5. En primer lugar se mirarán las diferencias divididas de segundo orden, que nos servirán para decidir que el valor a cambiar es f_{j-1} como aparece en la Figura 2.6. A continuación se considerarán las diferencias divididas de tercer orden indicadas en la Figura 2.7 y por tanto se cambiará el valor f_{j-2} tal y como se ve en la Figura 2.8.

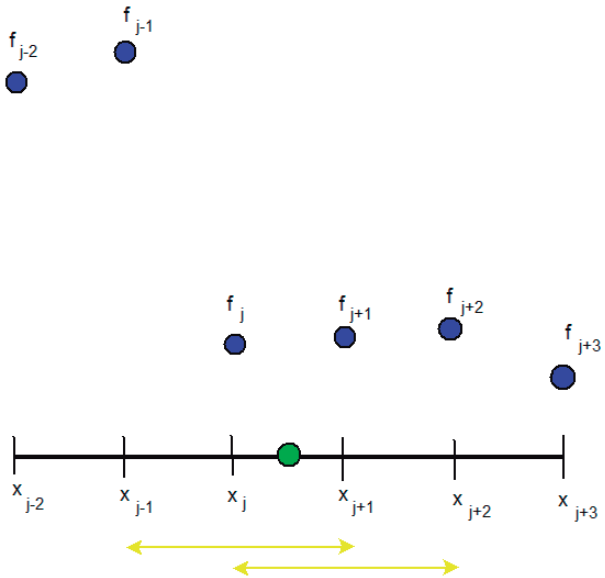

Figura 2.5: Primer paso de la reconstrucción PPH con 6 puntos.

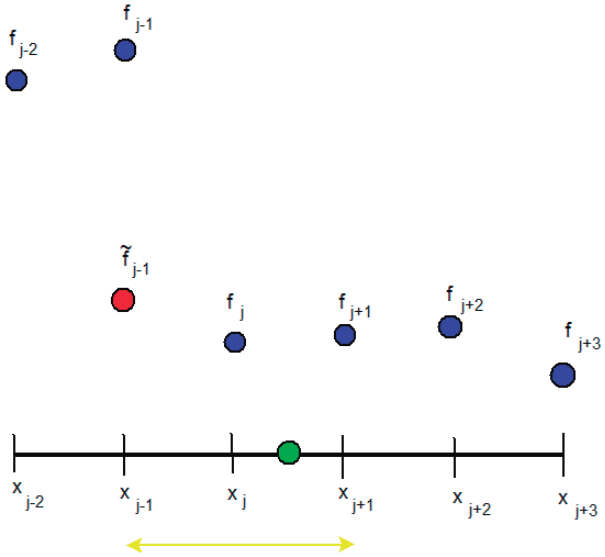

Figura 2.6: Modificación del primer valor en la construcción de la reconstrucción PPH con 6 puntos.

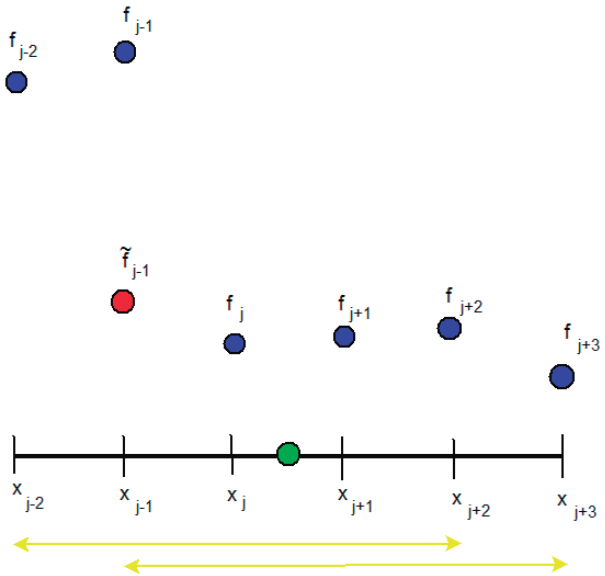

Figura 2.7: Segundo paso de la reconstrucción PPH con 6 puntos.

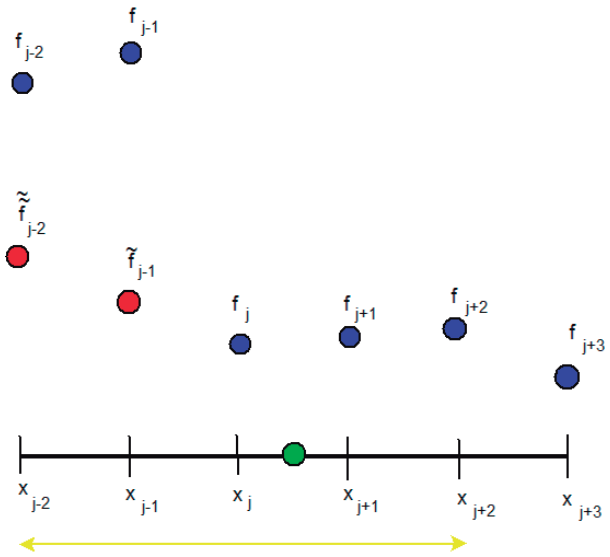

Figura 2.8: Modificación del segundo valor en la construcción de la reconstrucción PPH con 6 puntos.

2.3. Tratamiento en la frontera.

En la frontera no disponemos de datos suficientes para aplicar los filtros de manera centrada, como sería deseable, y entonces hemos de seguir otra alternativa. Hay diferentes formas de trabajar en la frontera. Explicaremos el caso de la frontera izquierda, ya que el procedimiento para la frontera derecha es similar. Para ilustrar de manera clara y sencilla cada tratamiento nos centraremos en el caso de filtros de 6 puntos.
Denotamos $f_i,\ i = 1 \ldots, n$ a los datos disponibles.
Entre los modos de tratar la frontera hemos considerado las siguientes:

- Periódico, como su nombre indica considera que los datos provienen de una función periódica y completa los datos que le faltan para la aplicación del filtro en la frontera haciendo uso de los datos de la otra frontera. Por ejemplo en el caso de 6 puntos hay problemas en la frontera sólo en dos intervalos. Para aplicar el filtro con 6 puntos en el primer intervalo afectado faltan dos datos. Y por tanto consideraremos los dos últimos datos en su lugar, es decir aplicaremos el filtro a $\{f_{n-1}, f_n, f_1, f_2, f_3, f_4\}$. Para el segundo intervalo de la frontera izquierda aplicaremos el filtro a $\{f_n, f_1, f_2, f_3, f_4, f_5\}$.

- Simétrico, en este caso los valores que faltan se completan suponiendo la función simétrica respecto del primer valor. Los filtros se aplicarán en este caso a $\{f_3, f_2, f_1, f_2, f_3, f_4\}$ para el primer intervalo y a $\{f_2, f_1, f_2, f_3, f_4, f_5\}$ en el segundo intervalo.

- Extensión con ceros, es una de las posibilidades más sencillas pero tiene el problema de generar discontinuidades inexistentes cuando la función y sus derivadas no empiezan valiendo cero. Los filtros se aplicarán en este caso a $\{0, 0, f_1, f_2, f_3, f_4\}$ para el primer intervalo y a $\{0, f_1, f_2, f_3, f_4, f_5\}$ en el segundo intervalo.

- Bajo orden, significa reducir el número de puntos del filtro conforme nos acercamos a la frontera para tener siempre suficientes datos. Los filtros se aplicarán en este caso a $\{f_1, f_2\}$ para el primer intervalo y a $\{f_1, f_2, f_3, f_4\}$ en el segundo intervalo.

- Hacia el interior, quiere indicar que se toman las máscaras descentradas siempre con los datos disponibles y sin reducir el número de puntos del filtro. Por simplicidad se ha tomado siempre el caso de interpolación de Lagrange. Los filtros de Lagrange descentrados se aplicarán en este caso a $\{f_1, f_2, f_3, f_4, f_5, f_6\}$ para el primer intervalo y a $\{f_1, f_2, f_3, f_4, f_5, f_6\}$ en el segundo intervalo.

- Antisimétrico, quiere indicar que la función es simétrica respecto del punto (x_1, f_1). Los filtros se aplicarán en este caso a $\{2f_1 - f_3, 2f_1 - f_2, f_1, f_2, f_3, f_4\}$ para el primer intervalo y a $\{2f_1 - f_2, f_1, f_2, f_3, f_4, f_5\}$ en el segundo intervalo.

2.4. Producto Tensor en 2D.

Explicaremos de manera gráfica para que sea más comprensible como se realiza el proceso de multirresolución que se aplica a la matriz original dependiendo del número de niveles aplicados.
Supongamos que ya disponemos de un algoritmo en una dimensión, el cual dado un vector discreto lo descompone en una parte correspondiente a valores significativos y otra a detalles. Esta descomposición contiene la misma información que el vector original. Aplicando el algoritmo 1-dimensional primero a cada una de las filas y después a cada una de las columnas tenemos una descomposición 2-dimensional ([4]).
Para la matriz original, el primer nivel de multirresolución se realiza de la siguiente manera:

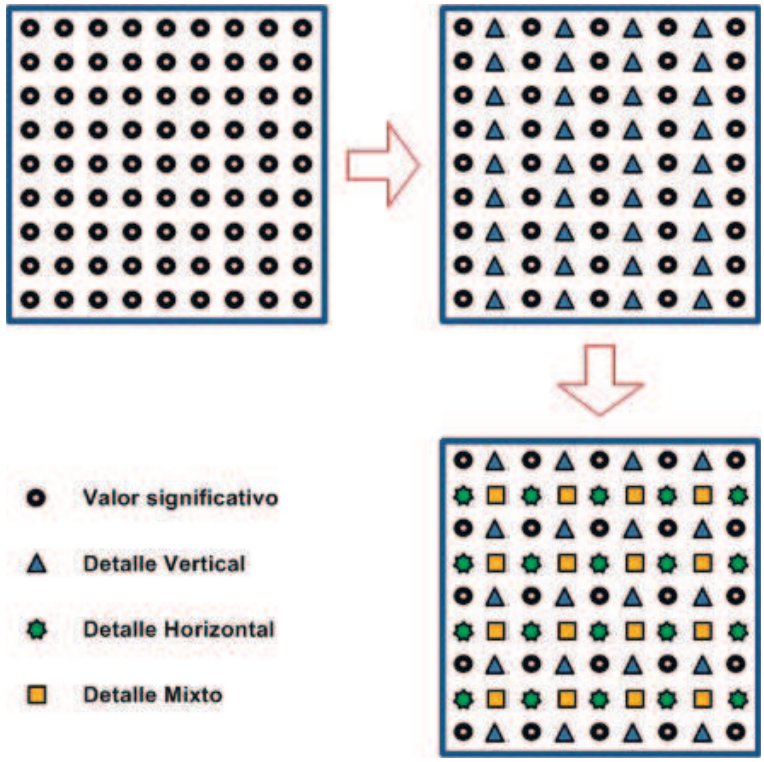

Figura 2.9: Pasos del algoritmo de multirresolución de valores puntuales: datos 2D originales, algoritmo 1D aplicado a las filas y algoritmo 1D aplicado a las columnas.

La matriz en la esquina superior izquierda corresponde a la matriz original, la segunda matriz corresponde a la aplicación del algoritmo 1-dimensional sobre cada una de las filas. Hemos representado mediante círculos negros los valores significativos y con triángulos azules los detalles verticales.
La tercera matriz, situada debajo, corresponde con el resultado de aplicar el algoritmo 1-dimensional a cada una de las columnas del resultado del paso anterior. Quedan representados con círculos negros los valores significativos, con triángulos azules los detalles verticales, con estrellas verdes los detalles horizontales y con cuadrados amarillos los detalles mixtos. Por tanto la última matriz está formada por los valores significativos, los detalles verticales, detalles horizontales y detalles mixtos aunque se encuentren todos entremezclados (ver Figura 2.9).

El siguiente paso sería recolocar la matriz agrupando por tipos cada uno de los valores y detalles. El resultado de la matriz reordenada puede verse en la Figura 2.10.

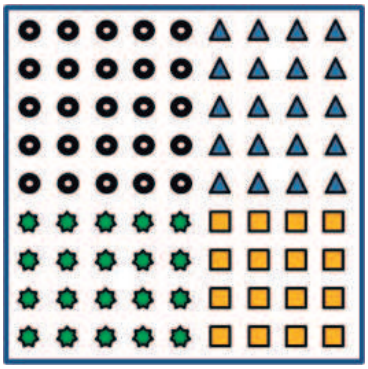

Figura 2.10: Versión de multirresolución de los datos para valores puntuales una vez reorganizados los coeficientes.

Para realizar el siguiente nivel de multirresolución procederíamos a realizar los mismos pasos tomando como matriz inicial la submatriz superior izquierda, es decir, la que está formada por los valores significativos obtenidos. Volveríamos a localizar los detalles verticales (de esa submatriz), a continuación los horizontales y por tanto los mixtos y reordenaríamos de nuevo la nueva matriz obtenida.
Este proceso se repite tantas veces como niveles de multirresolución hayamos pedido que realice.

El resultado de la matriz reordenada puede verse en la Figura 2.12.

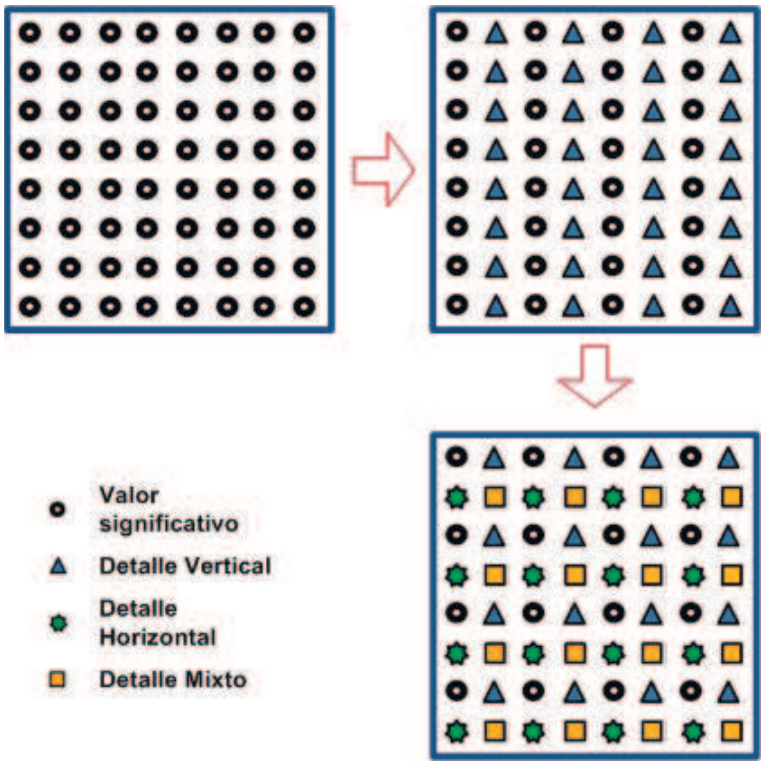

Figura 2.11: Pasos del algoritmo de multirresolución de medias en celda: datos 2D originales, algoritmo 1D aplicado a las filas y algoritmo 1D aplicado a las columnas.

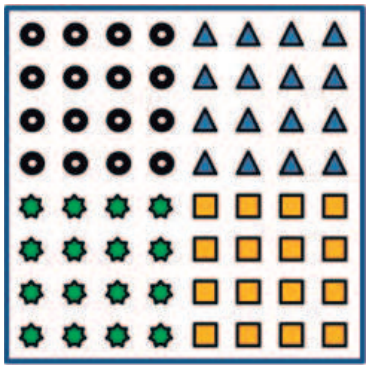

Figura 2.12: Versión de multirresolución de los datos para medias en celda una vez reorganizados los coeficientes.

Este procedimiento basado en producto tensorial se puede realizar también para trabajar con dimensiones más altas. La estrategia de producto tensorial no es específica de los algoritmos de multirresolución, sino que se

puede llevar a cabo para trabajar con datos en varias dimensiones cuando se dispone de un algoritmo 1-dimensional.

2.5. Compresión de Datos Bidimensionales.

Hemos visto que, dados unos datos bidimensionales, una representación de un nivel de multirresolución de dichos datos viene dada en las Figuras 2.10 y 2.12. Este proceso de descenso de un nivel de multirresolución puede expresarse matricialmente en la forma

$$A^k \leftrightarrow \left(\begin{array}{c|c} A^{k-1} & \Delta_2^k \\ \hline \Delta_3^k & \Delta_1^k \end{array}\right) \tag{2.24}$$

donde A^k representa la matriz de datos discretos en el nivel k, y Δ_1^k, Δ_2^k y Δ_3^k son los respectivos detalles necesarios para obtener A^k a partir de su versión decimada A^{k-1}.

Para llevar a cabo la compresión de los datos bidimensionales podemos llevar a cabo dos procedimientos sobre la versión de multirresolución:

1) Dado un parámetro de truncación ϵ, definimos el operador de truncación $\mathbf{tr}^\epsilon$ como

$$\mathbf{tr}^\epsilon(A^0, \Delta) = (A^0, \hat{\Delta})$$

con

$$\hat{\Delta}_i^k = \begin{cases} 0 & |\Delta_i^k| \leq \epsilon \\ \Delta_i^k & sino. \end{cases}$$

Notar que cuando se trabaja en el entorno de multirresolución por valores puntuales, usualmente se usa el mismo valor de ϵ para todos los niveles de multirresolución y que en medias en celda es habitual dividir por 2 dicho valor de ϵ cuando se desciende una escala.

Una vez aplicado este preproceso a los detalles de la versión multirresolutiva de los datos, tendremos muchas entradas de la matriz con el valor cero, y por tanto fáciles de almacenar en el ordenador.

2) Dado un porcentaje de compresión de detalles, nos quedamos justamente con este número de detalles. Lo único que debemos hacer en este caso es escoger aquellos más grandes en valor absoluto, pues son los más necesarios para construir una buena aproximación a los datos originales una vez que prescindamos del resto.

Entre los dos procedimientos propuestos nosotros preferiremos centrar nuestros experimentos numéricos en la segunda opción, ya que en la primera no hay un control claro a priori de la tasa de compresión que se va a lograr.

Después de procesar los detalles y quedarnos sólo con aquellos que no han sido truncados, utilizamos la transformación de multirresolución inversa para obtener una aproximación $\hat{A}^L$ a la información original A^L, es decir

$$\hat{A}^L = M^{-1}\mathbf{tr}^{\epsilon}(MA^L).$$

Comparamos entonces ambas secuencias de datos mediante las siguientes normas

$$\begin{aligned} ||A^L - \hat{A}^L||_p &= \left(\frac{1}{dim(A^L)}\sum_i |A_i^L - \hat{A}_i^L|^p\right)^{1/p}, \quad p = 1, 2, \\ ||A^L - \hat{A}^L||_\infty &= \max_i(|A_i^L - \hat{A}_i^L|). \end{aligned}$$

Este procedimiento de compresión se lleva a cabo de la misma forma y sin mayor complicación para dimensiones mayores.
Siguiendo los algoritmos de multirresolución explicados en las secciones anteriores y quedándonos con el porcentaje de detalles deseado, según se explica en el segundo procedimiento de truncación arriba mencionado, podemos controlar la tasa de compresión de los datos. Ahora bien, en algunas aplicaciones puede ser interesante o imprescindible controlar la calidad de la reconstrucción. Mediante los algoritmos propuestos sí estamos controlando la tasa de compresión, pero no la calidad. Es por eso, que se desarrollaron (ver [21]) algoritmos que permiten dicho control del error.

Capítulo 3

Filtros basados en funciones de ponderación.

3.1. Mínimos cuadrados ponderados.

Dado un conjunto de pares de puntos $(x_1, y_1), \ldots, (x_n, y_n)$ buscamos aquella recta $y = a \cdot x + b$ de tal manera que se minimiza la suma de los cuadrados de los errores que se obtienen al realizar la aproximación de los valores y_i por los obtenidos por la recta $\hat{y}_i = a \cdot x_i + b$ ponderados por unos factores de peso $w_i (w_i \geq 0)$. Por lo tanto, el objetivo es minimizar:

$$\sum_i w_i (\hat{y}_i - y_i)^2$$

donde w_i son unos pesos seleccionados para reducir/amplificar la influencia de las observaciones a la hora de encontrar la recta ajustada". La técnica usual es asociar a cada observación i un peso que sea inversamente proporcional a la variabilidad asociada al valor observado elevada al cuadrado, es decir:

$$w_i = \frac{k}{Var(y_i)}$$

Ahora bien, en la práctica resulta imposible conocer el valor $Var(y_i)$ por lo que se suele asumir que esta es proporcional al valor de x_i

$$Var(y_i) = c \cdot x_i$$

Por lo tanto se suponen que los pesos deben ser funciones cuyo valor sea proporcional a:

$$w_i \simeq \frac{1}{(x_i)^\alpha},$$

siendo α una constante positiva.
La solución al problema:

$$\arg\min_{a,b} \sum_i w_i(\hat{y}_i - y_i)^2 = \arg\min_{a,b} \sum_i w_i((a \cdot x_i + b) - y_i)^2$$

viene dado por:

$$a^* = \frac{(\sum_{i=1}^n w_i)(\sum_{i=1}^n w_i x_i y_i) - (\sum_{i=1}^n w_i y_i)(\sum_{i=1}^n w_i x_i)}{(\sum_{i=1}^n w_i)(\sum_{i=1}^n w_i x_i^2) - (\sum_{i=1}^n w_i x_i)^2},$$

$$b^* = \frac{(\sum_{i=1}^n w_i y_i)(\sum_{i=1}^n w_i x_i^2) - (\sum_{i=1}^n w_i x_i y_i)(\sum_{i=1}^n w_i x_i)}{(\sum_{i=1}^n w_i)(\sum_{i=1}^n w_i x_i^2) - (\sum_{i=1}^n w_i x_i)^2}.$$

Ahora bien, si asumimos que $\sum_{i=1}^n w_i = 1$, se tiene:

$$a^* = \frac{(\sum_{i=1}^n w_i x_i y_i) - (\sum_{i=1}^n w_i y_i)(\sum_{i=1}^n w_i x_i)}{(\sum_{i=1}^n w_i x_i^2) - (\sum_{i=1}^n w_i x_i)^2},$$

$$b^* = \frac{(\sum_{i=1}^n w_i y_i)(\sum_{i=1}^n w_i x_i^2) - (\sum_{i=1}^n w_i x_i y_i)(\sum_{i=1}^n w_i x_i)}{(\sum_{i=1}^n w_i x_i^2) - (\sum_{i=1}^n w_i x_i)^2}.$$

Por tanto, para:

$$\hat{x} = \sum_{i=1}^n w_i x_i$$

se tiene:

$$\hat{y} = a^* \cdot \hat{x} + b$$

$$= \frac{(\sum_{i=1}^n w_i x_i y_i) - (\sum_{i=1}^n w_i y_i)(\sum_{i=1}^n w_i x_i)}{(\sum_{i=1}^n w_i x_i^2) - (\sum_{i=1}^n w_i x_i)^2}(\sum_{i=1}^n w_i x_i) +$$

$$\frac{(\sum_{i=1}^n w_i y_i)(\sum_{i=1}^n w_i x_i^2) - (\sum_{i=1}^n w_i x_i y_i)(\sum_{i=1}^n w_i x_i)}{(\sum_{i=1}^n w_i x_i^2) - (\sum_{i=1}^n w_i x_i)^2}$$

$$= \sum_{i=1}^n w_i x_i$$

por tanto, la recta ajustada pasa siempre por el punto:

$$\left(\sum_{i=1}^{n} w_i x_i, \sum_{i=1}^{n} w_i y_i\right)$$

Por otro lado, si se desea realizar la predicción en el punto $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, si en el conjunto de valores $\{x_i\}$ se encuentran dispuestos de manera equidistante y los pesos w_i dependen únicamente de la distancia del punto x_i al punto $\bar{x}$, se tiene:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} w_i x_i$$

lo que indica, que la estimación que se realiza en el punto medio de los valores observados x_i utilizando mínimos cuadrados ponderados por una función peso w que dependa directamente de la distancia entre la observación x_i y $\bar{x}$, coincide con la media ponderada por w de los valores observados y_i.

$$\hat{y}(x = \bar{x}) = \sum_{i=1}^{n} w_i y_i$$

Así pues, utilizando distintas funciones de ponderación obtendremos distintas predicciones $\hat{y}$ en el punto $\bar{x}$.

3.2. Funciones de ponderación.

En toda esta sección asumiremos las siguientes coordenadas x_i

- 4 puntos $\rightarrow \left[1, 2, 3, 4\right] \rightarrow$ estimación en $\bar{x} = 2,5$
- 6 puntos $\rightarrow \left[1, 2, 3, 4, 5, 6\right] \rightarrow$ estimación en $\bar{x} = 3,5$
- 8 puntos $\rightarrow \left[1, 2, 3, 4, 5, 6, 7, 8\right] \rightarrow$ estimación en $\bar{x} = 4,5$

Veamos los resultados que se obtienen con las funciones de ponderación usuales.

3.2.1. Función de ponderación Uniforme.

$$w = \frac{1}{n}$$

Todos los puntos tienen la misma ponderación $\rightarrow$ Mínimos cuadrados ordinarios.

Coeficientes :

- 4 puntos $\rightarrow \left[1/4, 1/4, 1/4, 1/4\right]$
- 6 puntos $\rightarrow \left[1/6, 1/6, 1/6, 1/6, 1/6, 1/6\right]$
- 8 puntos $\rightarrow \left[1/8, \ldots, 1/8\right]$

3.2.2. Función de ponderación Bisquare.

$$w = \begin{cases} \left(1 - (\frac{(x_i - \bar{x})}{d\ max})^2\right)^2, & si\ |x_i - \bar{x}| < d\ max \\ \\ 0 & \text{en otro caso.} \end{cases}$$

siendo dmax, la distancia máxima permitida para que la función peso tome valores no nulos.

Representación gráfica para $dmax = 2$ (figura 3.1):

$$(1 - (x/2)^2)^2$$

Coeficientes :

- 4 puntos (dmax=2)$\rightarrow$ $\begin{bmatrix} \frac{(1-(\frac{3/2}{2})^2)^2}{137/64} \\ \frac{(1-(\frac{1/2}{2})^2)^2}{137/64} \\ \frac{(1-(\frac{1/2}{2})^2)^2}{137/64} \\ \frac{(1-(\frac{3/2}{2})^2)^2}{137/64} \end{bmatrix} = \begin{bmatrix} 8{,}9416 \times 10^{-2} \\ 0{,}410\,58 \\ 0{,}410\,58 \\ 8{,}9416 \times 10^{-2} \end{bmatrix}$
- 6 puntos (dmax=3)$\rightarrow$ $\begin{bmatrix} \frac{(1-(\frac{5/2}{3})^2)^2}{2075/648} \\ \frac{(1-(\frac{3/2}{3})^2)^2}{2075/648} \\ \frac{(1-(\frac{1/2}{3})^2)^2}{2075/648} \\ \frac{(1-(\frac{1/2}{3})^2)^2}{2075/648} \\ \frac{(1-(\frac{3/2}{3})^2)^2}{2075/648} \\ \frac{(1-(\frac{5/2}{3})^2)^2}{2075/648} \end{bmatrix} = \begin{bmatrix} 2{,}9157 \times 10^{-2} \\ 0{,}175\,66 \\ 0{,}295\,18 \\ 0{,}295\,18 \\ 0{,}175\,66 \\ 2{,}9157 \times 10^{-2} \end{bmatrix}$

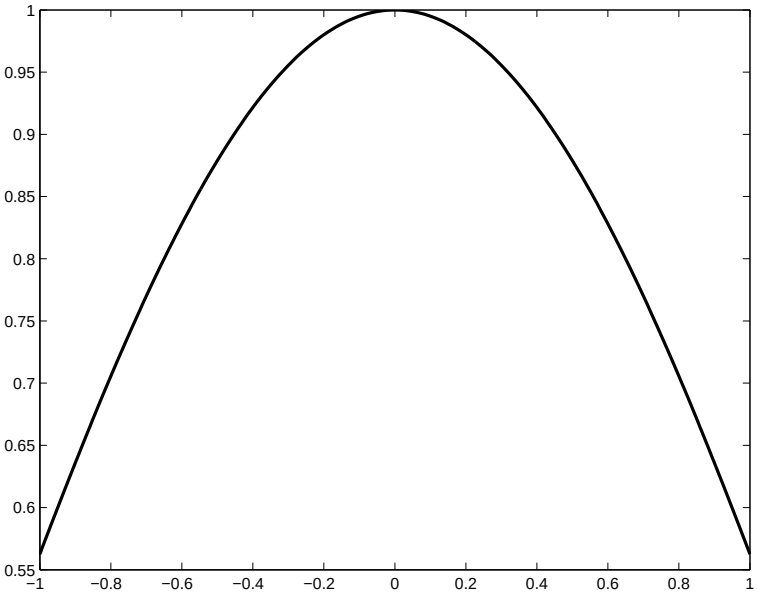

Figura 3.1: Representación gráfica de la función de ponderación Bisquare.

- 8 puntos (dmax=4)→ $\begin{bmatrix} \frac{(1-(\frac{7/2}{4})^2)^2}{2185/512} \\ \frac{(1-(\frac{5/2}{4})^2)^2}{2185/512} \\ \frac{(1-(\frac{3/2}{4})^2)^2}{2185/512} \\ \frac{(1-(\frac{1/2}{4})^2)^2}{2185/512} \\ \frac{(1-(\frac{1/2}{4})^2)^2}{2185/512} \\ \frac{(1-(\frac{3/2}{4})^2)^2}{2185/512} \\ \frac{(1-(\frac{5/2}{4})^2)^2}{2185/512} \\ \frac{(1-(\frac{7/2}{4})^2)^2}{2185/512} \end{bmatrix} = \begin{bmatrix} 1{,}2872 \times 10^{-2} \\ 8{,}7014 \times 10^{-2} \\ 0{,}173\,05 \\ 0{,}227\,06 \\ 0{,}227\,06 \\ 0{,}173\,05 \\ 8{,}7014 \times 10^{-2} \\ 1{,}2872 \times 10^{-2} \end{bmatrix}$

3.2.3. Función de ponderación Huber.

$$w = \frac{1}{|x_i - \bar{x}|}$$

Representación gráfica (figura 3.2):

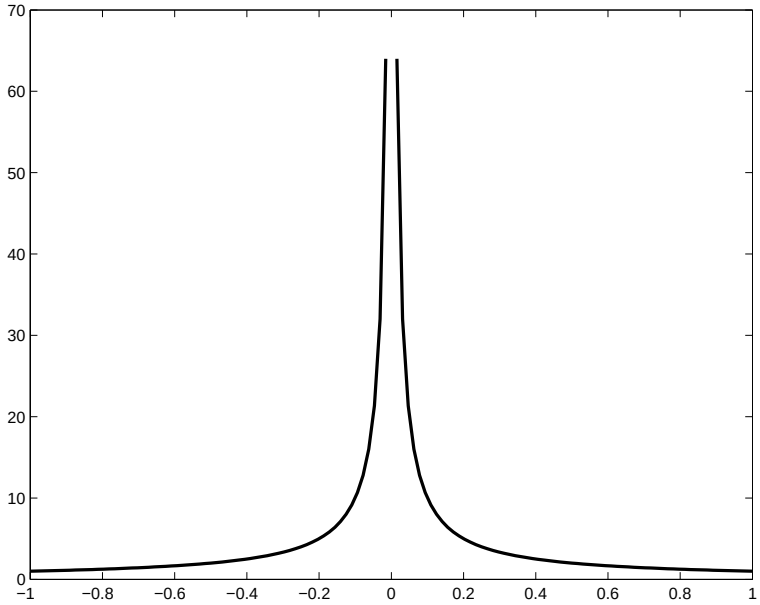

Figura 3.2: Representación gráfica de la función de ponderación Huber.

Coeficientes :

- 4 puntos $\rightarrow \begin{bmatrix} \frac{1}{3/2} \\ \frac{16/3}{1} \\ \frac{1}{1/2} \\ \frac{16/3}{1} \\ \frac{1}{1/2} \\ \frac{16/3}{1} \\ \frac{1}{3/2} \\ \frac{16/3}{} \end{bmatrix} = \begin{bmatrix} 0,125 \\ 0{,}375 \\ 0{,}375 \\ 0{,}125 \end{bmatrix}$

- 6 puntos $\rightarrow \begin{bmatrix} \frac{1}{5/2} \\ \frac{92/15}{1} \\ \frac{1}{3/2} \\ \frac{92/15}{1} \\ \frac{1}{1/2} \\ \frac{92/15}{1} \\ \frac{1}{1/2} \\ \frac{92/15}{1} \\ \frac{1}{3/2} \\ \frac{92/15}{1} \\ \frac{1}{5/2} \\ \frac{92/15}{} \end{bmatrix} = \begin{bmatrix} 6{,}5217 \times 10^{-2} \\ 0{,}108\,70 \\ 0{,}326\,09 \\ 0{,}326\,09 \\ 0{,}108\,70 \\ 6{,}5217 \times 10^{-2} \end{bmatrix}$

- 8 puntos $\rightarrow$ $\begin{bmatrix} \frac{1}{7/2} \\ \overline{704/105} \\ \frac{1}{5/2} \\ \overline{704/105} \\ \frac{1}{3/2} \\ \overline{704/105} \\ \frac{1}{1/2} \\ \overline{704/105} \\ \frac{1}{1/2} \\ \overline{704/105} \\ \frac{1}{3/2} \\ \overline{704/105} \\ \frac{1}{5/2} \\ \overline{704/105} \\ \frac{1}{7/2} \\ \overline{704/105} \end{bmatrix} = \begin{bmatrix} 4{,}2614 \times 10^{-2} \\ 5{,}9659 \times 10^{-2} \\ 9{,}9432 \times 10^{-2} \\ 0{,}298\,30 \\ 0{,}298\,30 \\ 9{,}9432 \times 10^{-2} \\ 5{,}9659 \times 10^{-2} \\ 4{,}2614 \times 10^{-2} \end{bmatrix}$

3.2.4. Función de ponderación Exponencial.

$$w = \begin{cases} \exp\left(-\left(\frac{|x_i - \bar{x}|}{\alpha . r}\right)\right) & si \;\; 0 \leq |x_i - \bar{x}| < r \qquad , \text{con} \;\; \alpha > 0 \\ \\ 0 & \text{en otro caso.} \end{cases}$$

Representación gráfica (figura 3.3):

$$\alpha = 1, r = 2 \longrightarrow \exp\left(-\left(\frac{\sqrt{(x^2)}}{2}\right)\right)$$

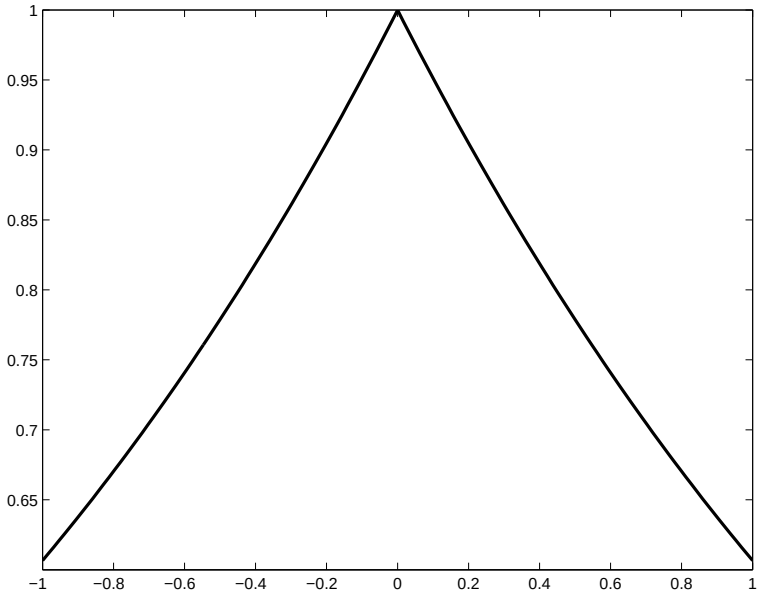

Figura 3.3: Representación gráfica de la función de ponderación Exponencial.

Coeficientes :

- 4 puntos $(\alpha = 1, r = 2) \rightarrow$ $\begin{bmatrix} 0{,}188\,77 \\ 0{,}311\,23 \\ 0{,}311\,23 \\ 0{,}188\,77 \end{bmatrix}$
- 6 puntos $(\alpha = 1, r = 3) \rightarrow$ $\begin{bmatrix} 0{,}115\,12 \\ 0{,}160\,66 \\ 0{,}224\,22 \\ 0{,}224\,22 \\ 0{,}160\,66 \\ 0{,}115\,12 \end{bmatrix}$
- 8 puntos $(\alpha = 1, r = 4) \rightarrow$ $\begin{bmatrix} 8{,}2648 \times 10^{-2} \\ 0{,}106\,12 \\ 0{,}136\,26 \\ 0{,}174\,97 \\ 0{,}174\,97 \\ 0{,}136\,26 \\ 0{,}106\,12 \\ 8{,}2648 \times 10^{-2} \end{bmatrix}$

3.2.5. Función de ponderación Gaussiana.

$$w = \begin{cases} \dfrac{\exp\left(-\left(\frac{|x_i-\bar{x}|}{c}\right)^{2k}\right)-\left(\exp-(\frac{r}{c})^{2k}\right)}{1-\left(\exp-(\frac{r}{c})^{2k}\right)} & si \;\; 0 \leq |x_i-\bar{x}| < r \qquad , \text{con} \;\; c,k>0 \\ \\ 0 & \text{en otro caso} \end{cases}$$

Representación gráfica (r=2,k=1,c=r) (figura 3.4):

$$\frac{\exp\left(-\left(\frac{\sqrt{x^2}}{2}\right)^2\right)-(\exp(-1))}{1-(\exp(-1))}$$

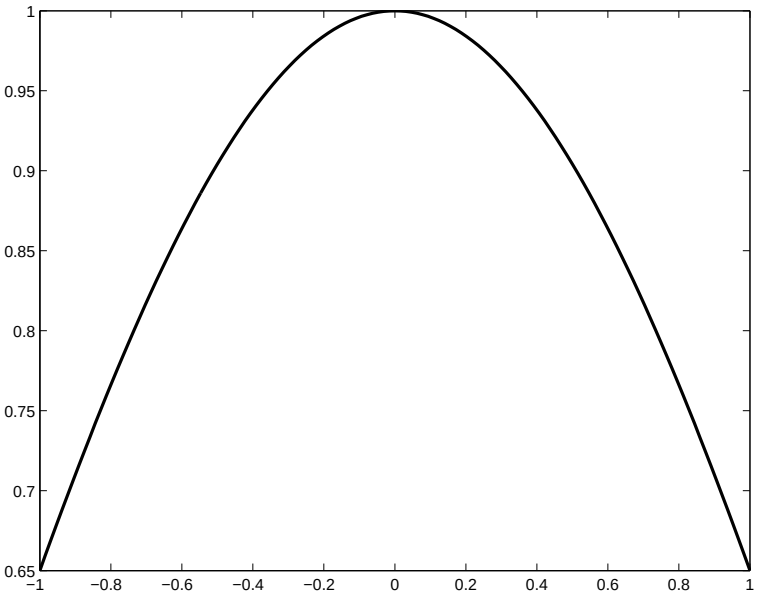

Figura 3.4: Representación gráfica de la función de ponderación Gaussiana.

Coeficientes :

- 4 puntos $(r = 2, k = 1, c = r) \rightarrow \begin{bmatrix} 0{,}130\,52 \\ 0{,}369\,48 \\ 0{,}369\,48 \\ 0{,}130\,52 \end{bmatrix}$
- 6 puntos $(r = 3, k = 1, c = r) \rightarrow \begin{bmatrix} 5{,}7305 \times 10^{-2} \\ 0{,}179\,11 \\ 0{,}263\,58 \\ 0{,}263\,58 \\ 0{,}179\,11 \\ 5{,}7305 \times 10^{-2} \end{bmatrix}$
- 8 puntos $(r = 4, k = 1, c = r) \rightarrow \begin{bmatrix} 3{,}1889 \times 10^{-2} \\ 0{,}101\,33 \\ 0{,}164\,41 \\ 0{,}202\,37 \\ 0{,}202\,37 \\ 0{,}164\,41 \\ 0{,}101\,33 \\ 3{,}1889 \times 10^{-2} \end{bmatrix}$

3.2.6. Función de ponderación Spline Cúbico.

$$w = \begin{cases} \frac{2}{3} - 4\left(\frac{|x_i - \bar{x}|}{r}\right)^2 + 4\left(\frac{|x_i - \bar{x}|}{r}\right)^3 & si \;\; 0 \leq |x_i - \bar{x}| < \frac{r}{2} \quad , \text{con} \;\; r > 0 \\ \frac{4}{3} - 4(\frac{|x_i - \bar{x}|}{r}) + 4\left(\frac{|x_i - \bar{x}|}{r}\right)^2 - \frac{4}{3}\left(\frac{|x_i - \bar{x}|}{r}\right)^3 & si \;\; \frac{r}{2} < |x_i - \bar{x}| \leq r \quad , \text{con} \;\; r > 0 \\ 0 & \text{en otro caso} \end{cases}$$

Representación gráfica (figura 3.5):

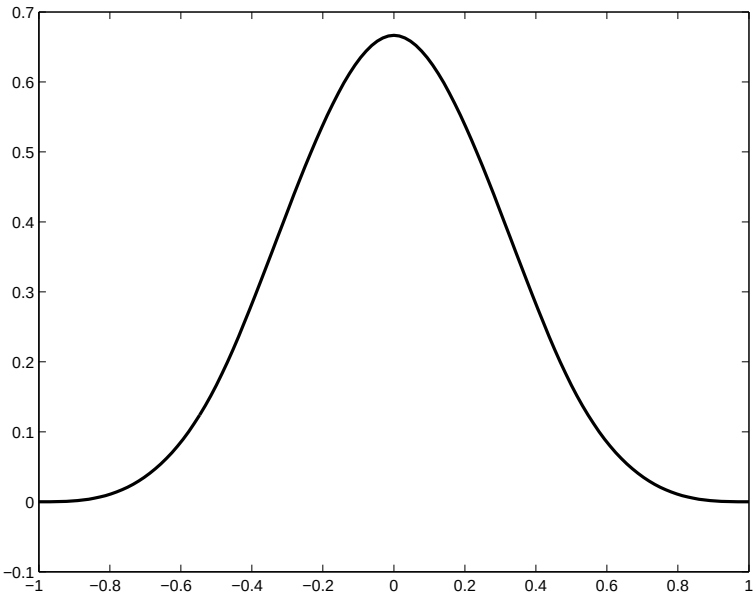

Figura 3.5: Representación gráfica de la función de ponderación Spline Cúbico.

Coeficientes :

- 4 puntos (r=2)→

$$w = \begin{cases} \frac{2}{3} - 4\left(\frac{\sqrt{x^2}}{2}\right)^2 + 4\left(\frac{\sqrt{x^2}}{2}\right)^3 & si \;\; 0 \leq |x| < 1 \\ \frac{4}{3} - 4\frac{\sqrt{x^2}}{2} + 4\left(\frac{\sqrt{x^2}}{2}\right)^2 - \frac{4}{3}\left(\frac{\sqrt{x^2}}{2}\right)^3 & si \;\; 1 < |x| \leq 2 \\ 0 & \text{en otro caso} \end{cases}$$

$$\begin{bmatrix} \frac{1}{48} \\ \frac{23}{48} \\ \frac{23}{48} \\ \frac{1}{48} \end{bmatrix} = \begin{bmatrix} 2{,}0833 \times 10^{-2} \\ 0{,}479\,17 \\ 0{,}479\,17 \\ 2{,}0833 \times 10^{-2} \end{bmatrix} \text{ (se ha forzado a que sumen 1)}$$

- 6 puntos (r=3)→

$$w = \begin{cases} \frac{2}{3} - 4\left(\frac{\sqrt{x^2}}{3}\right)^2 + 4\left(\frac{\sqrt{x^2}}{3}\right)^3 & si \;\; 0 \leq |x| < \frac{3}{2} \\ \frac{4}{3} - 4\frac{\sqrt{x^2}}{3} + 4\left(\frac{\sqrt{x^2}}{3}\right)^2 - \frac{4}{3}\left(\frac{\sqrt{x^2}}{3}\right)^3 & si \;\; \frac{3}{2} < |x| \leq 3 \\ 0 & \text{en otro caso} \end{cases}$$

$$\begin{bmatrix} \frac{1}{242} \\ \frac{27}{242} \\ \frac{93}{242} \\ \frac{93}{242} \\ \frac{27}{242} \\ \frac{1}{242} \end{bmatrix} = \begin{bmatrix} 4{,}1322 \times 10^{-3} \\ 0{,}111\,57 \\ 0{,}384\,30 \\ 0{,}384\,30 \\ 0{,}111\,57 \\ 4{,}1322 \times 10^{-3} \end{bmatrix} \text{ (se ha forzado a que sumen 1)}$$

- 8 puntos (r=4)→

$$w = \begin{cases} \frac{2}{3} - 4\left(\frac{\sqrt{x^2}}{4}\right)^2 + 4\left(\frac{\sqrt{x^2}}{4}\right)^3 & si \;\; 0 \leq |x| < 2 \\ \frac{4}{3} - 4\frac{\sqrt{x^2}}{4} + 4\left(\frac{\sqrt{x^2}}{4}\right)^2 - \frac{4}{3}\left(\frac{\sqrt{x^2}}{4}\right)^3 & si \;\; 2 < |x| \leq 4 \\ 0 & \text{en otro caso} \end{cases}$$

$$\begin{bmatrix} \frac{1}{768} \\ \frac{9}{256} \\ \frac{121}{768} \\ \frac{235}{768} \\ \frac{235}{768} \\ \frac{121}{768} \\ \frac{9}{256} \\ \frac{1}{768} \end{bmatrix} = \begin{bmatrix} 1{,}3021 \times 10^{-3} \\ 3{,}5156 \times 10^{-2} \\ 0{,}157\,55 \\ 0{,}305\,99 \\ 0{,}305\,99 \\ 0{,}157\,55 \\ 3{,}5156 \times 10^{-2} \\ 1{,}3021 \times 10^{-3} \end{bmatrix} \text{ (se ha forzado a que sumen 1)}$$

3.2.7. Función de ponderación Spline Cuártico.

$$w = \begin{cases} 1 - 6\left(\frac{|x_i - \bar{x}|}{r}\right)^2 + 8\left(\frac{|x_i - \bar{x}|}{r}\right)^3 - 3\left(\frac{|x_i - \bar{x}|}{r}\right)^4 & si \;\; 0 \leq |x_i - \bar{x}| < r \qquad , \text{con} \;\; r > 0 \\ \\ 0 & \text{en otro caso} \end{cases}$$

Representación gráfica $r = 2$ (figura 3.6):

$$1 - 6\left(\frac{\sqrt{x^2}}{2}\right)^2 + 8\left(\frac{\sqrt{x^2}}{2}\right)^3 - 3\left(\frac{\sqrt{x^2}}{2}\right)^4$$

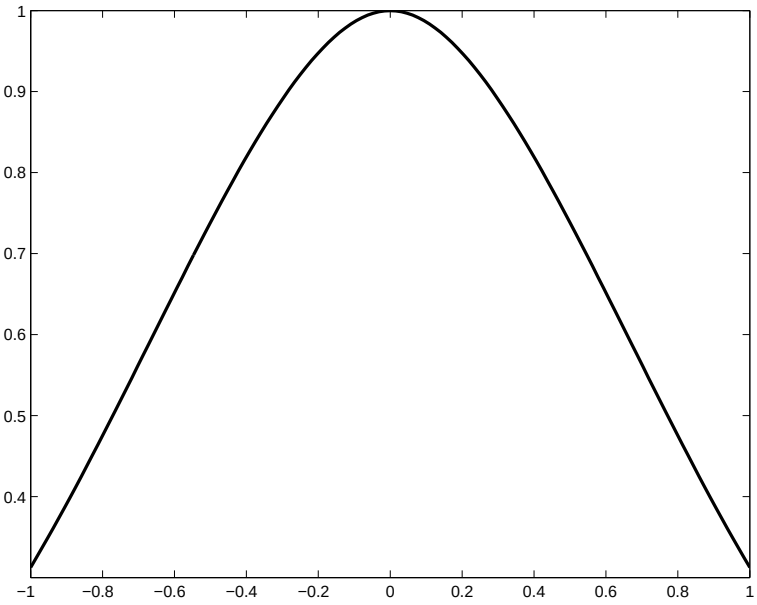

Figura 3.6: Representación gráfica de la función de ponderación Spline Cuártico.

Coeficientes :

- 4 puntos (r=2)$\rightarrow$ $\begin{bmatrix} \frac{13}{404} \\ \frac{189}{404} \\ \frac{189}{404} \\ \frac{13}{404} \end{bmatrix} = \begin{bmatrix} 3{,}2178 \times 10^{-2} \\ 0{,}467\,82 \\ 0{,}467\,82 \\ 3{,}2178 \times 10^{-2} \end{bmatrix}$

  (se ha forzado a que sumen 1)

- 6 puntos (r=3)$\rightarrow$ $\begin{bmatrix} \frac{7}{1034} \\ \frac{135}{1034} \\ \frac{375}{1034} \\ \frac{375}{1034} \\ \frac{135}{1034} \\ \frac{7}{1034} \end{bmatrix} = \begin{bmatrix} 6{,}7698 \times 10^{-3} \\ 0{,}130\,56 \\ 0{,}362\,67 \\ 0{,}362\,67 \\ 0{,}130\,56 \\ 6{,}7698 \times 10^{-3} \end{bmatrix}$
  (se ha forzado a que sumen 1)

- 8 puntos (r=4)$\rightarrow$ $\begin{bmatrix} \frac{29}{13\,096} \\ \frac{626}{13\,096} \\ \frac{2125}{13\,096} \\ \frac{3773}{13\,096} \\ \frac{3773}{13\,096} \\ \frac{2125}{13\,096} \\ \frac{621}{13\,096} \\ \frac{29}{13\,096} \end{bmatrix} = \begin{bmatrix} 2{,}2144 \times 10^{-3} \\ 4{,}7419 \times 10^{-2} \\ 0{,}162\,26 \\ 0{,}288\,1 \\ 0{,}288\,1 \\ 0{,}162\,26 \\ 4{,}7419 \times 10^{-2} \\ 2{,}2144 \times 10^{-3} \end{bmatrix}$
  (se ha forzado a que sumen 1)

Capítulo 4

Algoritmos codificados en Matlab.

A continuación se presenta una generalización de los algoritmos de multirresolución de Harten en valores puntuales con la reconstrucción de Lagrange para permitir el uso de otros filtros. Nosotros consideraremos en particular los filtros Bisquare, Exponencial, Huber, Lagrange, Spline Cuártico, Spline Cúbico y Uniforme que vienen de las correspondientes funciones de ponderación del mismo nombre. Se considera tanto el caso 1D como el 2D. Los algoritmos han sido codificados en Matlab .

4.1. Algoritmos de Multirresolución (MR) basados en Mínimos Cuadrados Ponderados (MCP) en 1D.

```
function [a,mra]=valpun(fichero,l,trun,fil,num,fron,met,str1)

% Este programa comprime un vector mediante algoritmos de multirresolución
% [a,mra]=valpun(fichero,l,trun,fil,num,fron,met,str1);
% Variables de entrada:
% fichero guarda los datos de la función a la que se le aplica la multirresolución
% l son los niveles de multirresolución
% trun indica el método de truncamiento:
% trun=[1,com] se queda con los  %com detalles mayores
% trun=[0,tol] indica la tolerancia de truncamiento, se queda con los
```

```
% detalles mayores en valor absoluto
% fil filtro a utilizar:
% 'Bisquare' 'Exponencial'
% 'Gaussiana' 'Huber'
% 'Lagrange' 'SplineCuartico'
% 'SplineCubico' 'Uniforme'
% num número de puntos del filtro
% fron tipo de tratamiento en la frontera:
% 'p' periódico 'i' hacia el interior usando Lagrange
% 's' simétrico, 'a' antisimétrico
% 'c' extendido con ceros 'b' bajo orden
% met indica si se hace un cambio de los datos:
% 0 sin cambio
% 1 con cambio PPH
% str1 contiene el nombre del fichero donde se guardan los datos de salida
% Variables de salida:
% a aproximación al vector despúes de truncar y decodificar
% mra versión de multirresolución procesada del vector original


% leemos del fichero los datos de la función
% en particular se cargan x1, a1,nombre_fun,tipo_fun

fichero=strcat('./ficheros de funciones/',fichero);
load(fichero);


n1=length(a1); % calcula la longitud del vector a1

t0=clock; % ordenes que sirven para medir el tiempo de computación
t=cputime;


% descendemos por la pirámide de multirresolución

mra=descenvp(a1,l,fil,num,fron,met);

% trunca los detalles

[mra,rat]=truncarpv(mra,l,trun);
```

```
% ascendemos por la pirámide de multirresolución

a=ascenvp(mra,l,fil,num,fron,met);

% medimos el tiempo de computación

tiempo=etime(clock,t0);
tiempo1=cputime-t;

% calculamos los errores

norma1=sum(abs(a-a1))/n1;
infin=max(abs(a-a1));
norma2=sum((a-a1).^2)/n1

% salida de resultados: en gráficas, por la pantalla y a un fichero

% dibujos

% dibujo de la función original

figure('Tag','orig_fun','Name',[blanks(6),'Función original'],'Position',...
[6 100 1270 690])

axes('Position',[0.05 0.05 0.9 0.9]);

ym=min(a1);
yM=max(a1);
rel=max([abs(ym),abs(yM)]);
plot(x1,a1,'k-','LineWidth',3,'MarkerSize',10);
axis([x1(1)-0.1 x1(n1)+0.1 ym-0.1*rel yM+0.1*rel]);

% dibujo de la descomposición piramidal de los detalles

figure('Tag','sub_fig','Name',[blanks(6),'Descomposición piramidal...
de los detalles',blanks(2),met, blanks(6),'Escalas:',blanks(2),...
num2str(l)],'Position',[6 20 1270 690])

axes('Position',[0.05 0.05 0.9 0.9]);

[multi_detail,multi_per_org]=pyramid(mra,l);
```

```
plot(x1,multi_detail,'ko','LineWidth',3,'MarkerSize',10);
axis([0 1 0 l+1])


% dibujo de la función reconstruida después de la compresión

% incluir el nombre del filtro fil, la tolerancia el método met

figure('Tag','reco_fig','Name',[blanks(6),'Función Reconstruida',...
blanks(6),'Escalas:',blanks(2),num2str(l)],'Position',[6 0 1270 690])

axes('Position',[0.05 0.05 0.9 0.9]);

ym=min(a);
yM=max(a);
rel=max([abs(ym),abs(yM)]);
plot(x1,a,'k–','LineWidth',3);
axis([x1(1)-0.1 x1(n1)+0.1 ym-0.1*rel yM+0.1*rel])



% Mandamos los datos de salida a una figura en la pantalla

figure('Tag','sub_fig','Name',[blanks(6),'Datos de Salida'],'Position',[6 0 1270
690]);
axis off


if fron=='p'
    fron='periódicas';
elseif fron=='i'
    fron='en el intervalo';
elseif fron=='s'
    fron='simétricas';
elseif fron=='a'
    fron='antisimétricas';
elseif fron=='c'
    fron='extensión con ceros';
```

```
elseif fron=='l'
    fron='bajando el orden'
end


if met==0
    met='sin cambio de datos';
elseif met==1
    met='con cambio PPH';
end



str={['Tipo de Función:',blanks(2),tipo_fun],['Nombre de la función:',...
blanks(2),nombre_fun],['Discontinuidades:',blanks(2),num2str(dis)],...
['Niveles de Multirresolución:',blanks(2),num2str(l)],...
['Filtro:',blanks(2),fil],['Tamaño del filtro:',blanks(2),num2str(num)],
['Método:',blanks(2),met],['Tratamiento en la frontera:',blanks(2),fron],
['Tasa de Compresión:',blanks(2),num2str(rat)],['Norma Infinito del Error:',...
blanks(2),num2str(infin)],['Norma 1 del Error:',blanks(2),num2str(norma1)],...
['Norma 2 del Error:',blanks(2),num2str(norma2)],...
['Tiempo de cálculo:',blanks(2),num2str(tiempo),blanks(3),'segundos'],...
['Tiempo de CPU:',blanks(2),num2str(tiempo1),blanks(3),'segundos'],
['Datos guardados en el fichero Datos_salida.txt' ]};



h1=uicontrol('Style','Text','String',str,'FontSize',16,'FontWeight','bold','Units',...
'normalized','Position',[0 0 1 0.95],'Background','White','HorizontalAlignment','left');


h2=uicontrol('Style','Text','String', {'Esquemas de Multirresolución en 1D'},
'FontSize',18,...
'FontWeight','bold','Units','normalized','Position',[0 0.95 1 0.05],'Background','green',...
'HorizontalAlignment','center','ForegroundColor','blue');

% fichero para generar las gráficas

format long;
```

```
fid=fopen(str1,'a');
fprintf(fid,' %d %d %f %4.16f %4.16f %4.16f %f %f \n',l,num,rat,infin,norma1,...
norma2,tiempo,tiempo1);
fclose(fid);
```

```
function c=descenvp(a1,l,fil,num,fron,met)

% Este programa desciende en la pirámide de multirresolución
% c=descenvp(a1,l,fil,num,fron,met);
% Variables de entrada:
% a1 vector al que se le aplica la multirresolución
% l son los niveles de multirresolución
% fil filtro a utilizar:
% 'Bisquare' 'Exponencial'
% 'Gaussiana' 'Huber'
% 'Lagrange' 'SplineCuartico'
% 'SplineCubico' 'Uniforme'
% num número de puntos del filtro
% fron tipo de tratamiento en la frontera:
% 'p' periódico 'i' hacia el interior usando Lagrange
% 's' simétrico, 'a' antisimétrico
% 'c' extendido con ceros 'b' bajo orden
% met indica si se hace un cambio de los datos:
% 0 sin cambio
% 1 con cambio PPH
% Variables de salida:
% c versión de multirresolución del vector a1


% calcula las máscaras del filtro

cd './filtros numéricos'
fil=strcat(fil,'.mat');
load(fil);
cd ..
```

```
% máscaras para la frontera bajando el orden

masf={mas2,mas4,mas6,mas8};


n=length(a1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SIN TRANSFORMACIÓN DE LOS DATOS. CASO LINEAL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


if met==0

   c=a1;


   %%%%%%%%%%%%%%
   % LINEAL CON 4 PUNTOS
   %%%%%%%%%%%%%%

   if num==4


      % máscaras de Lagrange para la frontera izquierda con los datos disponibles

      for i=1:num/2-1
         mas(1:num,i)=maskslr(i,num-i);
      end


      % bucle para los niveles de multirresolución

      for k=1:l

         % valores significativos de la escala inferior

         f1=c(1:2:n);
```

```
nk1=(n-1)/2+1; % dimensión de una escala inferior


% cálculo de los primeros detalles en la frontera izquierda
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if fron=='p'
    for i=1:num/2-1
        f2(i)=c(2*i)-[f1(nk1-num/2+i:nk1-1),f1(1:i+num/2)]*mas4';
    end
elseif fron=='i'
    f2(1:num/2-1)=c(2:2:num-2)-f1(1:num)*mas;
elseif fron=='s'
    for i=1:num/2-1
        f2(i)=c(2*i)-[f1(num/2-i+1:-1:2),f1(1:i+num/2)]*mas4';
    end
elseif fron=='a'
    for i=1:num/2-1
        f2(i)=c(2*i)-[2*f1(1)-f1(num/2-i+1:-1:2),f1(1:i+num/2)]*mas4';
    end
elseif fron=='c'
    for i=1:num/2-1
        f2(i)=c(2*i)-[zeros(1,num/2-i),f1(1:i+num/2)]*mas4';
    end
elseif fron=='b'
    for i=1:num/2-1
        f2(i)=c(2*i)-[f1(1:2*i)]*masf{i}';

    end

end


% cálculo de los detalles intermedios
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for j=num/2:nk1-num/2

    % calcula la predicción

    aux=f1(j-num/2+1:j+num/2);
```

```
            q=aux*mas4';

            % calcula el detalle

            f2(j)=c(2*j)-q;

        end


        % cálculo de los últimos detalles en la frontera derecha
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        if fron=='p'
            for i=nk1-1:-1:nk1-num/2+1
                f2(i)=c(2*i)-[f1(num/2+i-nk1+1:-1:2),f1(nk1:-1:i-num/2+1)]*mas4';
            end
        elseif fron=='i'
            f2(nk1-1:-1:nk1-num/2+1)=c(n-1:-2:n-num+3)-f1(nk1:-1:nk1-num+1)*mas;
        elseif fron=='s'
            for i=nk1-1:-1:nk1-num/2+1
                f2(i)=c(2*i)-[f1(2*nk1-num/2-i:nk1-1),f1(nk1:-1:i-num/2+1)]*mas4';
            end
        elseif fron=='a'
            for i=nk1-1:-1:nk1-num/2+1
                f2(i)=c(2*i)-[2*f1(nk1)-f1(2*nk1-num/2-i:nk1-1),f1(nk1:-1:i-
num/2+1)]*mas4';
            end
        elseif fron=='c'
            for i=nk1-1:-1:nk1-num/2+1
                f2(i)=c(2*i)-[zeros(1,num/2+i-nk1),f1(nk1:-1:i-num/2+1)]*mas4';
            end
        elseif fron=='b'
            for i=nk1-1:-1:nk1-num/2+1
                f2(i)=c(2*i)-[f1(nk1:-1:2*i-nk1+1)]*masf{nk1-i}';
            end
        end


        c(1:n)=[f1,f2]; clear f1; clear f2;
        n=nk1;

        % se cierra el bucle de los niveles de multirresolución
```

```
    end


    %%%%%%%%%%%%%%
    % LINEAL CON 6 PUNTOS
    %%%%%%%%%%%%%%


elseif num==6

    % máscaras de Lagrange para la frontera izquierda con los datos disponibles

    for i=1:num/2-1
        mas(1:num,i)=maskslr(i,num-i);
    end


    % bucle para los niveles de multirresolución

    for k=1:l

        % valores significativos de la escala inferior

        f1=c(1:2:n);

        nk1=(n-1)/2+1; % dimensión de una escala inferior


        % cálculo de los primeros detalles en la frontera izquierda
        %%%%%%%%%%%%%%%%%%%%%%%%%%

        if fron=='p'
            for i=1:num/2-1
                f2(i)=c(2*i)-[f1(nk1-num/2+i:nk1-1),f1(1:i+num/2)]*mas6';
            end
        elseif fron=='i'
            f2(1:num/2-1)=c(2:2:num-2)-f1(1:num)*mas;
        elseif fron=='s'
            for i=1:num/2-1
                f2(i)=c(2*i)-[f1(num/2-i+1:-1:2),f1(1:i+num/2)]*mas6';
```

```
    end
elseif fron=='a'
    for i=1:num/2-1
        f2(i)=c(2*i)-[2*f1(1)-f1(num/2-i+1:-1:2),f1(1:i+num/2)]*mas6';
    end
elseif fron=='c'
    for i=1:num/2-1
        f2(i)=c(2*i)-[zeros(1,num/2-i),f1(1:i+num/2)]*mas6';
    end
elseif fron=='b'
    for i=1:num/2-1
        f2(i)=c(2*i)-[f1(1:2*i)]*masf{i}';

    end

end


% cálculo de los detalles intermedios
%%%%%%%%%%%%%%%%%%%%%

for j=num/2:nk1-num/2

    % calcula la predicción

    aux=f1(j-num/2+1:j+num/2);
    q=aux*mas6';

    % calcula el detalle

    f2(j)=c(2*j)-q;

end

% cálculo de los últimos detalles en la frontera derecha
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if fron=='p'
    for i=nk1-1:-1:nk1-num/2+1
        f2(i)=c(2*i)-[f1(num/2+i-nk1+1:-1:2),f1(nk1:-1:i-num/2+1)]*mas6';
    end
```

```
            elseif fron=='i'
                f2(nk1-1:-1:nk1-num/2+1)=c(n-1:-2:n-num+3)-f1(nk1:-1:nk1-num+1)*mas;
            elseif fron=='s'
                for i=nk1-1:-1:nk1-num/2+1
                    f2(i)=c(2*i)-[f1(2*nk1-num/2-i:nk1-1),f1(nk1:-1:i-num/2+1)]*mas6';
                end
            elseif fron=='a'
                for i=nk1-1:-1:nk1-num/2+1
                    f2(i)=c(2*i)-[2*f1(nk1)-f1(2*nk1-num/2-i:nk1-1),f1(nk1:-1:i-
num/2+1)]*mas6';
                end
            elseif fron=='c'
                for i=nk1-1:-1:nk1-num/2+1
                    f2(i)=c(2*i)-[zeros(1,num/2+i-nk1),f1(nk1:-1:i-num/2+1)]*mas6';
                end
            elseif fron=='b'
                for i=nk1-1:-1:nk1-num/2+1
                    f2(i)=c(2*i)-[f1(nk1:-1:2*i-nk1+1)]*masf{nk1-i}';
                end
            end


            c(1:n)=[f1,f2]; clear f1; clear f2;
            n=nk1;

             % se cierra el bucle de los niveles de multirresolución
        end


        %%%%%%%%%%%%%
        % LINEAL CON 8 PUNTOS
        %%%%%%%%%%%%%

    elseif num==8

        % máscaras de Lagrange para la frontera izquierda con los datos dis-
ponibles

        for i=1:num/2-1
            mas(1:num,i)=maskslr(i,num-i);
        end
```

```
% bucle para los niveles de multirresolución

for k=1:l

    % valores significativos de la escala inferior

    f1=c(1:2:n);

    nk1=(n-1)/2+1; % dimensión de una escala inferior


    % cálculo de los primeros detalles en la frontera izquierda
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    if fron=='p'
        for i=1:num/2-1
            f2(i)=c(2*i)-[f1(nk1-num/2+i:nk1-1),f1(1:i+num/2)]*mas8';
        end
    elseif fron=='i'
        f2(1:num/2-1)=c(2:2:num-2)-f1(1:num)*mas;
    elseif fron=='s'
        for i=1:num/2-1
            f2(i)=c(2*i)-[f1(num/2-i+1:-1:2),f1(1:i+num/2)]*mas8';
        end
    elseif fron=='a'
        for i=1:num/2-1
            f2(i)=c(2*i)-[2*f1(1)-f1(num/2-i+1:-1:2),f1(1:i+num/2)]*mas8';
        end
    elseif fron=='c'
        for i=1:num/2-1
            f2(i)=c(2*i)-[zeros(1,num/2-i),f1(1:i+num/2)]*mas8';
        end
    elseif fron=='b'
        for i=1:num/2-1
            f2(i)=c(2*i)-[f1(1:2*i)]*masf{i}';

        end

    end
```

```
% cálculo de los detalles intermedios
%%%%%%%%%%%%%%%%%%%%

for j=num/2:nk1-num/2

    % calcula la predicción

    aux=f1(j-num/2+1:j+num/2);
    q=aux*mas8';

    % calcula el detalle

    f2(j)=c(2*j)-q;

end

% cálculo de los últimos detalles en la frontera derecha
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if fron=='p'
    for i=nk1-1:-1:nk1-num/2+1
        f2(i)=c(2*i)-[f1(num/2+i-nk1+1:-1:2),f1(nk1:-1:i-num/2+1)]*mas8';
    end
elseif fron=='i'
    f2(nk1-1:-1:nk1-num/2+1)=c(n-1:-2:n-num+3)-f1(nk1:-1:nk1-num+1)*mas;
elseif fron=='s'
    for i=nk1-1:-1:nk1-num/2+1
        f2(i)=c(2*i)-[f1(2*nk1-num/2-i:nk1-1),f1(nk1:-1:i-num/2+1)]*mas8';
    end
elseif fron=='a'
    for i=nk1-1:-1:nk1-num/2+1
        f2(i)=c(2*i)-[2*f1(nk1)-f1(2*nk1-num/2-i:nk1-1),f1(nk1:-1:i-
num/2+1)]*mas8';
    end
elseif fron=='c'
    for i=nk1-1:-1:nk1-num/2+1
        f2(i)=c(2*i)-[zeros(1,num/2+i-nk1),f1(nk1:-1:i-num/2+1)]*mas8';
    end
elseif fron=='b'
```

```
        for i=nk1-1:-1:nk1-num/2+1
            f2(i)=c(2*i)-[f1(nk1:-1:2*i-nk1+1)]*masf{nk1-i}';
        end
    end


    c(1:n)=[f1,f2]; clear f1; clear f2;
    n=nk1;

    % se cierra el bucle de los niveles de multirresolución
  end

end


%%%%%%%%%%%%%%%%%%%%%%%%
% CON TRANSFORMACIÓN DE LOS DATOS. PPH
%%%%%%%%%%%%%%%%%%%%%%%%


elseif met==1

c=a1;

%%%%%%%%%%%%%%%%%%%%%%%%
% CAMBIO PPH + FILTRO LINEAL CON 4 PUNTOS
%%%%%%%%%%%%%%%%%%%%%%%%


if num==4

    % máscaras de Lagrange para la frontera izquierda con los datos disponibles

  for i=1:num/2-1
      mas(1:num,i)=maskslr(i,num-i);
  end


    % bucle para los niveles de multirresolución
```

```
for k=1:l

    % valores significativos de la escala inferior

    f1=c(1:2:n);

    nk1=(n-1)/2+1; % dimensión de una escala inferior


    % cálculo de los primeros detalles en la frontera izquierda
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    if fron=='p'
        for i=1:num/2-1
            f2(i)=c(2*i)-[f1(nk1-num/2+i:nk1-1),f1(1:i+num/2)]*mas4';
        end
    elseif fron=='i'
        f2(1:num/2-1)=c(2:2:num-2)-f1(1:num)*mas;
    elseif fron=='s'
        for i=1:num/2-1
            f2(i)=c(2*i)-[f1(num/2-i+1:-1:2),f1(1:i+num/2)]*mas4';
        end
    elseif fron=='a'
        for i=1:num/2-1
            f2(i)=c(2*i)-[2*f1(1)-f1(num/2-i+1:-1:2),f1(1:i+num/2)]*mas4';
        end
    elseif fron=='c'
        for i=1:num/2-1
            f2(i)=c(2*i)-[zeros(1,num/2-i),f1(1:i+num/2)]*mas4';
        end
    elseif fron=='b'
        for i=1:num/2-1
            f2(i)=c(2*i)-[f1(1:2*i)]*masf{i}';

        end

    end


    % cálculo de los detalles intermedios
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
for j=num/2:nk1-num/2

    % calcula la predicción

    aux=f1(j-num/2+1:j+num/2);
    aux=f_modify(aux);
    q=aux*mas4';

    % calcula el detalle

    f2(j)=c(2*j)-q;

end


% cálculo de los últimos detalles en la frontera derecha
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if fron=='p'
    for i=nk1-1:-1:nk1-num/2+1
        f2(i)=c(2*i)-[f1(num/2+i-nk1+1:-1:2),f1(nk1:-1:i-num/2+1)]*mas4';
    end
elseif fron=='i'
    f2(nk1-1:-1:nk1-num/2+1)=c(n-1:-2:n-num+3)-f1(nk1:-1:nk1-num+1)*mas;
elseif fron=='s'
    for i=nk1-1:-1:nk1-num/2+1
        f2(i)=c(2*i)-[f1(2*nk1-num/2-i:nk1-1),f1(nk1:-1:i-num/2+1)]*mas4';
    end
elseif fron=='a'
    for i=nk1-1:-1:nk1-num/2+1
        f2(i)=c(2*i)-[2*f1(nk1)-f1(2*nk1-num/2-i:nk1-1),f1(nk1:-1:i-
num/2+1)]*mas4';
    end
elseif fron=='c'
    for i=nk1-1:-1:nk1-num/2+1
        f2(i)=c(2*i)-[zeros(1,num/2+i-nk1),f1(nk1:-1:i-num/2+1)]*mas4';
    end
elseif fron=='b'
    for i=nk1-1:-1:nk1-num/2+1
        f2(i)=c(2*i)-[f1(nk1:-1:2*i-nk1+1)]*masf{nk1-i}';
```

```
            end
        end

        c(1:n)=[f1,f2]; clear f1; clear f2;
        n=nk1;

         % se cierra el bucle de los niveles de multirresolución
    end

    %%%%%%%%%%%%%%%%%%%%%%%%
    % CAMBIO PPH + FILTRO LINEAL CON 6 PUNTOS
    %%%%%%%%%%%%%%%%%%%%%%%%

elseif num==6

    % máscaras de Lagrange para la frontera izquierda con los datos disponibles

    for i=1:num/2-1
        mas(1:num,i)=maskslr(i,num-i);
    end

    % bucle para los niveles de multirresolución

    for k=1:l

        % valores significativos de la escala inferior

        f1=c(1:2:n);

        nk1=(n-1)/2+1; % dimensión de una escala inferior

        % cálculo de los primeros detalles en la frontera izquierda
        %%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
if fron=='p'
    for i=1:num/2-1
        f2(i)=c(2*i)-[f1(nk1-num/2+i:nk1-1),f1(1:i+num/2)]*mas6';
    end
elseif fron=='i'
    f2(1:num/2-1)=c(2:2:num-2)-f1(1:num)*mas;
elseif fron=='s'
    for i=1:num/2-1
        f2(i)=c(2*i)-[f1(num/2-i+1:-1:2),f1(1:i+num/2)]*mas6';
    end
elseif fron=='a'
    for i=1:num/2-1
        f2(i)=c(2*i)-[2*f1(1)-f1(num/2-i+1:-1:2),f1(1:i+num/2)]*mas6';
    end
elseif fron=='c'
    for i=1:num/2-1
        f2(i)=c(2*i)-[zeros(1,num/2-i),f1(1:i+num/2)]*mas6';
    end
elseif fron=='b'
    for i=1:num/2-1
        f2(i)=c(2*i)-[f1(1:2*i)]*masf{i}';

    end

end


% cálculo de los detalles intermedios
%%%%%%%%%%%%%%%%%%%

for j=num/2:nk1-num/2

    % calcula la predicción

    aux=f1(j-num/2+1:j+num/2);
    aux=f_modify(aux);
    q=aux*mas6';

    % calcula el detalle

    f2(j)=c(2*j)-q;
```

```
    end


    % cálculo de los últimos detalles en la frontera derecha
    %%%%%%%%%%%%%%%%%%%%%%

    if fron=='p'
        for i=nk1-1:-1:nk1-num/2+1
            f2(i)=c(2*i)-[f1(num/2+i-nk1+1:-1:2),f1(nk1:-1:i-num/2+1)]*mas6';
        end
    elseif fron=='i'
        f2(nk1-1:-1:nk1-num/2+1)=c(n-1:-2:n-num+3)-f1(nk1:-1:nk1-num+1)*mas;
    elseif fron=='s'
        for i=nk1-1:-1:nk1-num/2+1
            f2(i)=c(2*i)-[f1(2*nk1-num/2-i:nk1-1),f1(nk1:-1:i-num/2+1)]*mas6';
        end
    elseif fron=='a'
        for i=nk1-1:-1:nk1-num/2+1
            f2(i)=c(2*i)-[2*f1(nk1)-f1(2*nk1-num/2-i:nk1-1),f1(nk1:-1:i-
num/2+1)]*mas6';
        end
    elseif fron=='c'
        for i=nk1-1:-1:nk1-num/2+1
            f2(i)=c(2*i)-[zeros(1,num/2+i-nk1),f1(nk1:-1:i-num/2+1)]*mas6';
        end
    elseif fron=='b'
        for i=nk1-1:-1:nk1-num/2+1
            f2(i)=c(2*i)-[f1(nk1:-1:2*i-nk1+1)]*masf{nk1-i}';
        end
    end


    c(1:n)=[f1,f2]; clear f1; clear f2;
    n=nk1;

    % se cierra el bucle de los niveles de multirresolución
end


%%%%%%%%%%%%%%%%%%%%%%%%
```

```
    % CAMBIO PPH + FILTRO LINEAL CON 8 PUNTOS
    %%%%%%%%%%%%%%%%%%%%%%%%%


elseif num==8

    % máscaras de Lagrange para la frontera izquierda con los datos disponibles

    for i=1:num/2-1
        mas(1:num,i)=maskslr(i,num-i);
    end


    % bucle para los niveles de multirresolución

    for k=1:l

        % valores significativos de la escala inferior

        f1=c(1:2:n);

        nk1=(n-1)/2+1; % dimensión de una escala inferior


        % cálculo de los primeros detalles en la frontera izquierda
        %%%%%%%%%%%%%%%%%%%%%%%%%%%

        if fron=='p'
            for i=1:num/2-1
                f2(i)=c(2*i)-[f1(nk1-num/2+i:nk1-1),f1(1:i+num/2)]*mas8';
            end
        elseif fron=='i'
            f2(1:num/2-1)=c(2:2:num-2)-f1(1:num)*mas;
        elseif fron=='s'
            for i=1:num/2-1
                f2(i)=c(2*i)-[f1(num/2-i+1:-1:2),f1(1:i+num/2)]*mas8';
            end
        elseif fron=='a'
            for i=1:num/2-1
                f2(i)=c(2*i)-[2*f1(1)-f1(num/2-i+1:-1:2),f1(1:i+num/2)]*mas8';
```

```
    end
elseif fron=='c'
    for i=1:num/2-1
        f2(i)=c(2*i)-[zeros(1,num/2-i),f1(1:i+num/2)]*mas8';
    end
elseif fron=='b'
    for i=1:num/2-1
        f2(i)=c(2*i)-[f1(1:2*i)]*masf{i}';

    end

end


% cálculo de los detalles intermedios
%%%%%%%%%%%%%%%%%%%%

for j=num/2:nk1-num/2

    % calcula la predicción

    aux=f1(j-num/2+1:j+num/2);
    aux=f_modify(aux);
    q=aux*mas8';

    % calcula el detalle

    f2(j)=c(2*j)-q;

end


% cálculo de los últimos detalles en la frontera derecha
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if fron=='p'
    for i=nk1-1:-1:nk1-num/2+1
        f2(i)=c(2*i)-[f1(num/2+i-nk1+1:-1:2),f1(nk1:-1:i-num/2+1)]*mas8';
    end
elseif fron=='i'
    f2(nk1-1:-1:nk1-num/2+1)=c(n-1:-2:n-num+3)-f1(nk1:-1:nk1-num+1)*mas;
```

```
            elseif fron=='s'
                for i=nk1-1:-1:nk1-num/2+1
                    f2(i)=c(2*i)-[f1(2*nk1-num/2-i:nk1-1),f1(nk1:-1:i-num/2+1)]*mas8';
                end
            elseif fron=='a'
                for i=nk1-1:-1:nk1-num/2+1
                    f2(i)=c(2*i)-[2*f1(nk1)-f1(2*nk1-num/2-i:nk1-1),f1(nk1:-1:i-
num/2+1)]*mas8';
                end
            elseif fron=='c'
                for i=nk1-1:-1:nk1-num/2+1
                    f2(i)=c(2*i)-[zeros(1,num/2+i-nk1),f1(nk1:-1:i-num/2+1)]*mas8';
                end
            elseif fron=='b'
                for i=nk1-1:-1:nk1-num/2+1
                    f2(i)=c(2*i)-[f1(nk1:-1:2*i-nk1+1)]*masf{nk1-i}';
                end
            end


            c(1:n)=[f1,f2]; clear f1; clear f2;
            n=nk1;

            % se cierra el bucle de los niveles de multirresolución
        end

    end


end


function [c,rat]=truncarpv(a,l,trun)

% funcion que trunca los detalles del vector multirresolucionado a
% discretización por valores puntuales
% c=truncarpv(a,l,tol)
% Variables de entrada:
% a vector al que truncarle los detalles
% l niveles de multirresolución
% trun indica el método de truncamiento:
```

```
% trun=[1,com] se queda con los %com detalles mayores
% trun=[0,tol] indica la tolerancia de truncamiento, se queda con los
% detalles mayores en valor absoluto
% Variables de salida:
% c vector con los detalles truncados
% rat porcentaje de detalles no cero


n=length(a);
n1=(n-1)/2^l+1;

if trun(1)==0

    c=[a(1:n1),a(n1+1:n).*(abs(a(n1+1:n))>=trun(2))];

    nz=nnz(c(n1+1:n));

    rat=nz*100/(n-n1);

else


    % nos quedamos con los num_det mayores coeficientes de detalle

    num_det=floor(trun(2)*(n-n1)/100);

    c=a;
    c(1:n1)=0; % eliminamos en b los coeficientes significativos de la última
escala
    [y,ind]=sort(abs(c),'descend'); % ordenamos los el valor absoluto de los
coeficientes en una columna
    y=c(ind); % obtenemos los valores ordenados
    y(num_det+1:n)=0; % eliminamos los menos significativos
    c(ind)=y;
    c(1:n1)=a(1:n1); % restauramos los coeficientes significativos de la últi-
ma escala
    nz=nnz(c(n1+1:n));
    rat=nz*100/(n-n1);

end
```

```
function c=ascenvp(a,l,fil,num,fron,met)

% Decodificación en 1D para valores puntuales
% c=ascenvp(a,l,fil,num,fron,met)
% Variables de entrada:
% a vector a decodificar
% l número de escalas a subir
% fil filtro a utilizar:
% 'Bisquare' 'Exponencial'
% 'Gaussiana' 'Huber'
% 'Lagrange' 'SplineCuartico'
% 'SplineCubico' 'Uniforme'
% num número de puntos del filtro
% fron tipo de tratamiento en la frontera:
% 'p' periódico 'i' hacia el interior usando Lagrange
% 's' simétrico, 'a' antisimétrico
% 'c' extendido con ceros 'b' bajo orden
% met indica si se hace un cambio de los datos:
% 0 sin cambio
% 1 con cambio PPH
% Variables de salida:
% c aproximación al vector original después de la decodificación


% calcula las máscaras del filtro
cd './filtros numéricos'
fil=strcat(fil,'.mat');
load(fil);
cd ..


% máscaras para la frontera bajando el orden
masf={mas2,mas4,mas6,mas8};


n=length(a);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SIN TRANSFORMACIÓN DE LOS DATOS. CASO LINEAL
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if met==0

  c=a;

  %%%%%%%%%%%%%
  % LINEAL CON 4 PUNTOS
  %%%%%%%%%%%%%

  if num==4

    % máscaras de Lagrange para la frontera izquierda con los datos disponibles

    for i=1:num/2-1
      mas(1:num,i)=maskslr(i,num-i);
    end

    nl=(n-1)/(2^l)+1;

    % inicialización de las variables

    for k=l:-1:1

      % se trata del bucle para los niveles de multirresolución
      % reordenamos el vector haciendo el proceso inverso que en la codificación

      nb=2*nl-1;
      b=zeros(1,nl); b(1:nl)=c(1:nl);
      f=zeros(1,nb);
      f(1:2:nb)=c(1:nl);
      f(2:2:nb-1)=c(nl+1:nb);
```

```
% predicción del primer elemento de f

if fron=='p'
    for i=1:num/2-1
        f(2*i)=f(2*i)+[b(nl-num/2+i:nl-1),b(1:i+num/2)]*mas4';
    end
elseif fron=='i'
    f(2:2:num-2)=f(2:2:num-2)+b(1:num)*mas;
elseif fron=='s'
    for i=1:num/2-1
        f(2*i)=f(2*i)+[b(num/2-i+1:-1:2),b(1:i+num/2)]*mas4';
    end
elseif fron=='a'
    for i=1:num/2-1
        f(2*i)=f(2*i)+[2*b(1)-b(num/2-i+1:-1:2),b(1:i+num/2)]*mas4';
    end
elseif fron=='c'
    for i=1:num/2-1
        f(2*i)=f(2*i)+[zeros(1,num/2-i),b(1:i+num/2)]*mas4';
    end
elseif fron=='b'
    for i=1:num/2-1
        f(2*i)=f(2*i)+[b(1:2*i)]*masf{i}';
    end
end


% predicción de los valores intermedios de f

for i=num/2:nl-num/2

    aux=b(i-num/2+1:i+num/2)';
    q=mas4*aux;
    f(2*i)=f(2*i)+q;

end


% predicción del último elemento

if fron=='p'
```

```
            for i=nl-1:-1:nl-num/2+1
                f(2*i)=f(2*i)+[b(num/2+i-nl+1:-1:2),b(nl:-1:i-num/2+1)]*mas4';
            end
        elseif fron=='i'
            f(nb-1:-2:nb-num+3)=f(nb-1:-2:nb-num+3)+b(nl:-1:nl-num+1)*mas;
        elseif fron=='s'
            for i=nl-1:-1:nl-num/2+1
                f(2*i)=f(2*i)+[b(2*nl-num/2-i:nl-1),b(nl:-1:i-num/2+1)]*mas4';
            end
        elseif fron=='a'
            for i=nl-1:-1:nl-num/2+1
                f(2*i)=f(2*i)+[2*b(nl)-b(2*nl-num/2-i:nl-1),b(nl:-1:i-num/2+1)]*mas4';
            end
        elseif fron=='c'
            for i=nl-1:-1:nl-num/2+1
                f(2*i)=f(2*i)+[zeros(1,num/2+i-nl),b(nl:-1:i-num/2+1)]*mas4';
            end
        elseif fron=='b'
            for i=nl-1:-1:nl-num/2+1
                f(2*i)=f(2*i)+[b(nl:-1:2*i-nl+1)]*masf{nl-i}';
            end
        end

        c(1:nb)=f(1:nb);
        clear b; clear f;


        % se calcula el nl para subir al siguiente nivel

        nl=nb;

    end


    %%%%%%%%%%%%%
    % LINEAL CON 6 PUNTOS
    %%%%%%%%%%%%%


elseif num==6
```

```
% máscaras de Lagrange para la frontera izquierda con los datos disponibles

for i=1:num/2-1
    mas(1:num,i)=maskslr(i,num-i);
end


nl=(n-1)/(2^l)+1;


% inicialización de las variables

for k=l:-1:1

    % se trata del bucle para los niveles de multirresolución
    % reordenamos el vector haciendo el proceso inverso que en la codificación

    nb=2*nl-1;
    b=zeros(1,nl); b(1:nl)=c(1:nl);
    f=zeros(1,nb);
    f(1:2:nb)=c(1:nl);
    f(2:2:nb-1)=c(nl+1:nb);


    % predicción del primer elemento de f

    if fron=='p'
        for i=1:num/2-1
            f(2*i)=f(2*i)+[b(nl-num/2+i:nl-1),b(1:i+num/2)]*mas6';
        end
    elseif fron=='i'
        f(2:2:num-2)=f(2:2:num-2)+b(1:num)*mas;
    elseif fron=='s'
        for i=1:num/2-1
            f(2*i)=f(2*i)+[b(num/2-i+1:-1:2),b(1:i+num/2)]*mas6';
        end
    elseif fron=='a'
        for i=1:num/2-1
```

```
        f(2*i)=f(2*i)+[2*b(1)-b(num/2-i+1:-1:2),b(1:i+num/2)]*mas6';
    end
elseif fron=='c'
    for i=1:num/2-1
        f(2*i)=f(2*i)+[zeros(1,num/2-i),b(1:i+num/2)]*mas6';
    end
elseif fron=='b'
    for i=1:num/2-1
        f(2*i)=f(2*i)+[b(1:2*i)]*masf{i}';
    end
end


% predicción de los valores intermedios de f

for i=num/2:nl-num/2

    aux=b(i-num/2+1:i+num/2)';
    q=mas6*aux;
    f(2*i)=f(2*i)+q;

end


% predicción del último elemento

if fron=='p'
    for i=nl-1:-1:nl-num/2+1
        f(2*i)=f(2*i)+[b(num/2+i-nl+1:-1:2),b(nl:-1:i-num/2+1)]*mas6';
    end
elseif fron=='i'
    f(nb-1:-2:nb-num+3)=f(nb-1:-2:nb-num+3)+b(nl:-1:nl-num+1)*mas;
elseif fron=='s'
    for i=nl-1:-1:nl-num/2+1
        f(2*i)=f(2*i)+[b(2*nl-num/2-i:nl-1),b(nl:-1:i-num/2+1)]*mas6';
    end
elseif fron=='a'
    for i=nl-1:-1:nl-num/2+1
        f(2*i)=f(2*i)+[2*b(nl)-b(2*nl-num/2-i:nl-1),b(nl:-1:i-num/2+1)]*mas6';
    end
elseif fron=='c'
```

```
            for i=nl-1:-1:nl-num/2+1
                f(2*i)=f(2*i)+[zeros(1,num/2+i-nl),b(nl:-1:i-num/2+1)]*mas6';
            end
        elseif fron=='b'
            for i=nl-1:-1:nl-num/2+1
                f(2*i)=f(2*i)+[b(nl:-1:2*i-nl+1)]*masf{nl-i}';
            end
        end


        c(1:nb)=f(1:nb);
        clear b; clear f;


        % se calcula el nl para subir al siguiente nivel

        nl=nb;

    end


    %%%%%%%%%%%%%
    % LINEAL CON 8 PUNTOS
    %%%%%%%%%%%%%


elseif num==8


    % máscaras de Lagrange para la frontera izquierda con los datos disponibles

    for i=1:num/2-1
        mas(1:num,i)=maskslr(i,num-i);
    end


    nl=(n-1)/(2^l)+1;


    % inicialización de las variables
```

```
for k=l:-1:1

    % se trata del bucle para los niveles de multirresolución
    % reordenamos el vector haciendo el proceso inverso que en la co-
dificación

    nb=2*nl-1;
    b=zeros(1,nl); b(1:nl)=c(1:nl);
    f=zeros(1,nb);
    f(1:2:nb)=c(1:nl);
    f(2:2:nb-1)=c(nl+1:nb);


    % predicción del primer elemento de f

    if fron=='p'
        for i=1:num/2-1
            f(2*i)=f(2*i)+[b(nl-num/2+i:nl-1),b(1:i+num/2)]*mas8';
        end
    elseif fron=='i'
        f(2:2:num-2)=f(2:2:num-2)+b(1:num)*mas;
    elseif fron=='s'
        for i=1:num/2-1
            f(2*i)=f(2*i)+[b(num/2-i+1:-1:2),b(1:i+num/2)]*mas8';
        end
    elseif fron=='a'
        for i=1:num/2-1
            f(2*i)=f(2*i)+[2*b(1)-b(num/2-i+1:-1:2),b(1:i+num/2)]*mas8';
        end
    elseif fron=='c'
        for i=1:num/2-1
            f(2*i)=f(2*i)+[zeros(1,num/2-i),b(1:i+num/2)]*mas8';
        end
    elseif fron=='b'
        for i=1:num/2-1
            f(2*i)=f(2*i)+[b(1:2*i)]*masf{i}';
        end
    end
```

```
% predicción de los valores intermedios de f

for i=num/2:nl-num/2

    aux=b(i-num/2+1:i+num/2)';
    q=mas8*aux;
    f(2*i)=f(2*i)+q;

end


% predicción del último elemento

if fron=='p'
    for i=nl-1:-1:nl-num/2+1
        f(2*i)=f(2*i)+[b(num/2+i-nl+1:-1:2),b(nl:-1:i-num/2+1)]*mas8';
    end
elseif fron=='i'
    f(nb-1:-2:nb-num+3)=f(nb-1:-2:nb-num+3)+b(nl:-1:nl-num+1)*mas;
elseif fron=='s'
    for i=nl-1:-1:nl-num/2+1
        f(2*i)=f(2*i)+[b(2*nl-num/2-i:nl-1),b(nl:-1:i-num/2+1)]*mas8';
    end
elseif fron=='a'
    for i=nl-1:-1:nl-num/2+1
        f(2*i)=f(2*i)+[2*b(nl)-b(2*nl-num/2-i:nl-1),b(nl:-1:i-num/2+1)]*mas8';
    end
elseif fron=='c'
    for i=nl-1:-1:nl-num/2+1
        f(2*i)=f(2*i)+[zeros(1,num/2+i-nl),b(nl:-1:i-num/2+1)]*mas8';
    end
elseif fron=='b'
    for i=nl-1:-1:nl-num/2+1
        f(2*i)=f(2*i)+[b(nl:-1:2*i-nl+1)]*masf{nl-i}';
    end
end


c(1:nb)=f(1:nb);
clear b; clear f;
```

```
            % se calcula el nl para subir al siguiente nivel

            nl=nb;

        end

    end


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % CON TRANSFORMACIÓN DE LOS DATOS. PPH
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


elseif met==1

    c=a;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % CAMBIO PPH + FILTRO LINEAL CON 4 PUNTOS
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


    if num==4

        % máscaras de Lagrange para la frontera izquierda con los datos dis-
ponibles

        for i=1:num/2-1
            mas(1:num,i)=maskslr(i,num-i);
        end


        nl=(n-1)/(2^l)+1;


        % inicialización de las variables

        for k=l:-1:1
```

```
% se trata del bucle para los niveles de multirresolución
% reordenamos el vector haciendo el proceso inverso que en la co-
dificación

nb=2*nl-1;
b=zeros(1,nl); b(1:nl)=c(1:nl);
f=zeros(1,nb);
f(1:2:nb)=c(1:nl);
f(2:2:nb-1)=c(nl+1:nb);


% predicción del primer elemento de f

if fron=='p'
    for i=1:num/2-1
        f(2*i)=f(2*i)+[b(nl-num/2+i:nl-1),b(1:i+num/2)]*mas4';
    end
elseif fron=='i'
    f(2:2:num-2)=f(2:2:num-2)+b(1:num)*mas;
elseif fron=='s'
    for i=1:num/2-1
        f(2*i)=f(2*i)+[b(num/2-i+1:-1:2),b(1:i+num/2)]*mas4';
    end
elseif fron=='a'
    for i=1:num/2-1
        f(2*i)=f(2*i)+[2*b(1)-b(num/2-i+1:-1:2),b(1:i+num/2)]*mas4';
    end
elseif fron=='c'
    for i=1:num/2-1
        f(2*i)=f(2*i)+[zeros(1,num/2-i),b(1:i+num/2)]*mas4';
    end
elseif fron=='b'
    for i=1:num/2-1
        f(2*i)=f(2*i)+[b(1:2*i)]*masf{i}';
    end
end


% predicción de los valores intermedios de f

for i=num/2:nl-num/2
```

```
    aux=b(i-num/2+1:i+num/2);
    aux=f_modify(aux)';
    q=mas4*aux;
    f(2*i)=f(2*i)+q;

end


% predicción del último elemento

if fron=='p'
    for i=nl-1:-1:nl-num/2+1
        f(2*i)=f(2*i)+[b(num/2+i-nl+1:-1:2),b(nl:-1:i-num/2+1)]*mas4';
    end
elseif fron=='i'
    f(nb-1:-2:nb-num+3)=f(nb-1:-2:nb-num+3)+b(nl:-1:nl-num+1)*mas;
elseif fron=='s'
    for i=nl-1:-1:nl-num/2+1
        f(2*i)=f(2*i)+[b(2*nl-num/2-i:nl-1),b(nl:-1:i-num/2+1)]*mas4';
    end
elseif fron=='a'
    for i=nl-1:-1:nl-num/2+1
        f(2*i)=f(2*i)+[2*b(nl)-b(2*nl-num/2-i:nl-1),b(nl:-1:i-num/2+1)]*mas4';
    end
elseif fron=='c'
    for i=nl-1:-1:nl-num/2+1
        f(2*i)=f(2*i)+[zeros(1,num/2+i-nl),b(nl:-1:i-num/2+1)]*mas4';
    end
elseif fron=='b'
    for i=nl-1:-1:nl-num/2+1
        f(2*i)=f(2*i)+[b(nl:-1:2*i-nl+1)]*masf{nl-i}';
    end
end


c(1:nb)=f(1:nb);
clear b; clear f;


% se calcula el nl para subir al siguiente nivel
```

```
            nl=nb;

        end


        %%%%%%%%%%%%%%%%%%%%%%%%%
        % CAMBIO PPH + FILTRO LINEAL CON 6 PUNTOS
        %%%%%%%%%%%%%%%%%%%%%%%%%


    elseif num==6


        % máscaras de Lagrange para la frontera izquierda con los datos disponibles

        for i=1:num/2-1
            mas(1:num,i)=maskslr(i,num-i);
        end


        nl=(n-1)/(2^l)+1;


        % inicialización de las variables

        for k=l:-1:1

            % se trata del bucle para los niveles de multirresolución
            % reordenamos el vector haciendo el proceso inverso que en la codificación

            nb=2*nl-1;
            b=zeros(1,nl); b(1:nl)=c(1:nl);
            f=zeros(1,nb);
            f(1:2:nb)=c(1:nl);
            f(2:2:nb-1)=c(nl+1:nb);


            % predicción del primer elemento de f
```

```
if fron=='p'
    for i=1:num/2-1
        f(2*i)=f(2*i)+[b(nl-num/2+i:nl-1),b(1:i+num/2)]*mas6';
    end
elseif fron=='i'
    f(2:2:num-2)=f(2:2:num-2)+b(1:num)*mas;
elseif fron=='s'
    for i=1:num/2-1
        f(2*i)=f(2*i)+[b(num/2-i+1:-1:2),b(1:i+num/2)]*mas6';
    end
elseif fron=='a'
    for i=1:num/2-1
        f(2*i)=f(2*i)+[2*b(1)-b(num/2-i+1:-1:2),b(1:i+num/2)]*mas6';
    end
elseif fron=='c'
    for i=1:num/2-1
        f(2*i)=f(2*i)+[zeros(1,num/2-i),b(1:i+num/2)]*mas6';
    end
elseif fron=='b'
    for i=1:num/2-1
        f(2*i)=f(2*i)+[b(1:2*i)]*masf{i}';
    end
end


% predicción de los valores intermedios de f

for i=num/2:nl-num/2

    aux=b(i-num/2+1:i+num/2);
    aux=f_modify(aux)';
    q=mas6*aux;
    f(2*i)=f(2*i)+q;

end


% predicción del último elemento

if fron=='p'
```

```
            for i=nl-1:-1:nl-num/2+1
                f(2*i)=f(2*i)+[b(num/2+i-nl+1:-1:2),b(nl:-1:i-num/2+1)]*mas6';
            end
        elseif fron=='i'
            f(nb-1:-2:nb-num+3)=f(nb-1:-2:nb-num+3)+b(nl:-1:nl-num+1)*mas;
        elseif fron=='s'
            for i=nl-1:-1:nl-num/2+1
                f(2*i)=f(2*i)+[b(2*nl-num/2-i:nl-1),b(nl:-1:i-num/2+1)]*mas6';
            end
        elseif fron=='a'
            for i=nl-1:-1:nl-num/2+1
                f(2*i)=f(2*i)+[2*b(nl)-b(2*nl-num/2-i:nl-1),b(nl:-1:i-num/2+1)]*mas6';
            end
        elseif fron=='c'
            for i=nl-1:-1:nl-num/2+1
                f(2*i)=f(2*i)+[zeros(1,num/2+i-nl),b(nl:-1:i-num/2+1)]*mas6';
            end
        elseif fron=='b'
            for i=nl-1:-1:nl-num/2+1
                f(2*i)=f(2*i)+[b(nl:-1:2*i-nl+1)]*masf{nl-i}';
            end
        end


        c(1:nb)=f(1:nb);
        clear b; clear f;


        % se calcula el nl para subir al siguiente nivel

        nl=nb;

    end


    %%%%%%%%%%%%%%%%%%%%%%%%%%
    % CAMBIO PPH + FILTRO LINEAL CON 8 PUNTOS
    %%%%%%%%%%%%%%%%%%%%%%%%%%


elseif num==8
```

```
% máscaras de Lagrange para la frontera izquierda con los datos disponibles

for i=1:num/2-1
    mas(1:num,i)=maskslr(i,num-i);
end


nl=(n-1)/(2^l)+1;


% inicialización de las variables

for k=l:-1:1

    % se trata del bucle para los niveles de multirresolución
    % reordenamos el vector haciendo el proceso inverso que en la codificación

    nb=2*nl-1;
    b=zeros(1,nl); b(1:nl)=c(1:nl);
    f=zeros(1,nb);
    f(1:2:nb)=c(1:nl);
    f(2:2:nb-1)=c(nl+1:nb);


    % predicción del primer elemento de f

    if fron=='p'
        for i=1:num/2-1
            f(2*i)=f(2*i)+[b(nl-num/2+i:nl-1),b(1:i+num/2)]*mas8';
        end
    elseif fron=='i'
        f(2:2:num-2)=f(2:2:num-2)+b(1:num)*mas;
    elseif fron=='s'
        for i=1:num/2-1
            f(2*i)=f(2*i)+[b(num/2-i+1:-1:2),b(1:i+num/2)]*mas8';
        end
    elseif fron=='a'
        for i=1:num/2-1
```

```
        f(2*i)=f(2*i)+[2*b(1)-b(num/2-i+1:-1:2),b(1:i+num/2)]*mas8';
    end
elseif fron=='c'
    for i=1:num/2-1
        f(2*i)=f(2*i)+[zeros(1,num/2-i),b(1:i+num/2)]*mas8';
    end
elseif fron=='b'
    for i=1:num/2-1
        f(2*i)=f(2*i)+[b(1:2*i)]*masf{i}';
    end
end


% predicción de los valores intermedios de f

for i=num/2:nl-num/2

    aux=b(i-num/2+1:i+num/2);
    aux=f_modify(aux)';
    q=mas8*aux;
    f(2*i)=f(2*i)+q;

end


% predicción del último elemento

if fron=='p'
    for i=nl-1:-1:nl-num/2+1
        f(2*i)=f(2*i)+[b(num/2+i-nl+1:-1:2),b(nl:-1:i-num/2+1)]*mas8';
    end
elseif fron=='i'
    f(nb-1:-2:nb-num+3)=f(nb-1:-2:nb-num+3)+b(nl:-1:nl-num+1)*mas;
elseif fron=='s'
    for i=nl-1:-1:nl-num/2+1
        f(2*i)=f(2*i)+[b(2*nl-num/2-i:nl-1),b(nl:-1:i-num/2+1)]*mas8';
    end
elseif fron=='a'
    for i=nl-1:-1:nl-num/2+1
        f(2*i)=f(2*i)+[2*b(nl)-b(2*nl-num/2-i:nl-1),b(nl:-1:i-num/2+1)]*mas8';
    end
```

```
            elseif fron=='c'
                for i=nl-1:-1:nl-num/2+1
                    f(2*i)=f(2*i)+[zeros(1,num/2+i-nl),b(nl:-1:i-num/2+1)]*mas8';
                end
            elseif fron=='b'
                for i=nl-1:-1:nl-num/2+1
                    f(2*i)=f(2*i)+[b(nl:-1:2*i-nl+1)]*masf{nl-i}';
                end
            end


            c(1:nb)=f(1:nb);
            clear b; clear f;


            % se calcula el nl para subir al siguiente nivel

            nl=nb;

        end

    end

end



function [mas,numer,deno]=maskslr(l,r)

% Este programa calcula las máscaras del esquema de subdivisión de
% orden n de Lagrange con l puntos a la izquierda
% y r puntos a la derecha
% [mas,numer,deno]=maskslr(l,r);
% Variables de entrada:
% l número de nodos a la izquierda
% r número de nodos a la derecha
% Variables de salida:
% mas vector que contiene la máscara
% numer vector que contiene el numerador de la máscara
% deno vector que contiene el denominador de la máscara
```

```
alfas=-(2*l-1):2:(2*r-1);
n=l+r;

for i=0:(n-1)
    prod=1;
    prod2=1;
    for j=0:(n-1)
        if(j =i)
        prod=prod*alfas(j+1);
        prod2=prod2*(alfas(i+1)-alfas(j+1));
    end
end
numer(i+1)=(-1)^(n-1)*prod;
deno(i+1)=prod2;
mas(i+1)=numer(i+1)/deno(i+1);

end


function fm=f_modify(f)

% Esta función calcula los valores modificados para la reconstrucción
% pphpower partiendo de los valores iniciales de la función f
% fm=f_modify(f);
% Variables de entrada:
% f vector con los valores discretos de una función f (x)
% h espacio uniforme entre los puntos de la red
% Variables de salida:
% fm vector con los valores discretos modificados

n=length(f);


for i=1:n/2-1  % Bucle recursivo

    vl=f(n/2-i:n/2+i);
    dl=diff(vl,2*i);
    vr=f(n/2-i+1:n/2+i+1);
    dr=diff(vr,2*i);
    coef=comb(2*i);
    coef=[coef,fliplr(coef)];
```

```
    pmean=pwe(n-2*i,dl,dr);


    if abs(dl)<= abs(dr)

        f(n/2+i+1)= - f(n/2-i)+ sum(coef.*f(n/2-i+1:n/2+i))+2*pmean;

    else
        f(n/2-i)= - f(n/2+i+1) + sum(coef.*f(n/2-i+1:n/2+i))+2*pmean;

    end

end


fm=f;

return


function c=comb(n)

% Esta es una función auxiliar que da los coeficientes de la
% expansión de los valores modificados en la reconstrucción pphpower
% c=comb(n);
% Variables de entrada:
% n la longitud del vector de los coeficientes,debe ser par
% Variables de salida:
% c vector con los coeficientes de la expresión modificada

for i=1:n/2
    c(i)=(-1)^(i+1)*(combinatorio(n,i)-combinatorio(n,i-1));
end


function c=combinatorio(n,m)

% Esta función calcula el número combinatorio de n sobre m
% c=combinatorio(n,m);
```

```
if(n==m — m==0)
c=1;
else
c=combinatorio(n-1,m)+combinatorio(n-1,m-1);
end


function med=pwe(p,x,y)

% Esta función calcula la p-media entre dos cantidades
% med = pwe (p,x,y);
% Variables de entrada:
% p orden de la p-media
% x,y los números para calcular la p-media
% Variables de salida:
% med valor de la p-media

r=x-y;
s=x+y;

if(x*y>0)
med=s/2*(1-abs(r/s)^p);
else
med=0;
end


function [multi_detail,multi_per_org]=pyramid(multi_per,ls)

% Esta función crea un vector que contiene la estructura piramidal de los
% coeficientes de los detalles
% [multi_detail,multi_per_org]=pyramid(multi_per,ls);
% Variables de entrada:
% multi_per representación de la multirresolución truncada del vector
% ls niveles de multirresolución
% Variables de salida:
% multi_detail vector que contiene la estructura piramidal de los detalles
% multi_per_org representación de la multirresolución truncada del vector
% reorganizada según la posición
```

```
n=length(multi_per);
m=(n-1)/2^ls+1;
m1=2*m-1;

multi_per_org=abs(multi_per);
multi_per_org(1:m)=-10; % in order to plot the detail coefficients only % para sólo dibujar los coeficientes del detalle
multi_detail=zeros(1,n);
multi_detail(1:m)=-10; % in order to plot the detail coefficients only % para sólo dibujar los coeficientes del detalle


for i=1:ls
    multi_detail(m+1:m1)=i*(multi_per(m+1:m1) =0)-10*(multi_per(m+1:m1)==0);
    multi_per_org(m+1:m1)=multi_per_org(m+1:m1)-10*(multi_per(m+1:m1)==0);
    b(1:2:m1)=multi_detail(1:m); b(2:2:m1-1)=multi_detail(m+1:m1);
    multi_detail(1:m1)=b;
    b(1:2:m1)=multi_per_org(1:m); b(2:2:m1-1)=multi_per_org(m+1:m1);
    multi_per_org(1:m1)=b;

    m=m1;
    m1=2*m-1;
end


% Este guion automatiza la ejecución del programa principal 'valpun.m'

% para una función concreta(en este caso particular 'fsoftpol2a_m257')
% con diferentes tipos de filtros, diferentes valores de compresión,
% diferente tratamiento de la frontera (en este caso particular 'i'),
% diferentes tamaños del filtro (en este caso particular 4 puntos) y
% distintos niveles de multirresolución (en este caso particular 3 niveles).

funciones={'fhiperbola_m257'};
n=length(funciones);
%l=[2,3];
l=3;
nl=length(l);
com=[50 65 80 90 100];
m=length(com);
filtros={'Bisquare','Exponencial','Gaussiana','Huber','Lagrange',...
```

```
'SplineCuartico','SplineCubico','Uniforme'};
nf=length(filtros);
filtros1={'B','E','G','H','L','S4','S3','U'};
%puntos=[4 6 8];
puntos=6;
np=length(puntos);
%frontera={'p','s','c','b','i','a'};
frontera={'i'};
nfr=length(frontera);


t0=clock; % ordenes que sirven para medir el tiempo de computación
t=cputime;


for i=1:n
    str=[funciones{i},'.mat'];
    for j=1:nl
        for k=1:m
            for t=1:nf
                for q=1:np
                    for s=1:nfr
                        str1=['./Datos_salida/',num2str(l(j)),funciones{i},'_',filtros1{t},...
                        num2str(puntos(q)),frontera{s},'.dat'];
                        [a,mra]=valpun(str,l(j),[1,com(k)],filtros{t},puntos(q),frontera{s},0,str1);
                        [a,mra]=valpun(str,l(j),[1,com(k)],filtros{t},puntos(q),frontera{s},1,str1);

                    end
                end
            end
        end
    end
end



tiempo=etime(clock,t0) % miden el tiempo de computación
tiempo1=cputime-t


% Este programa genera las gráficas de la función en cuestion
```

```
% (en este caso particular 'fsoftpol2a_m257')
% con diferentes tipos de filtros, diferentes valores de compresión,
% diferente tratamiento de la frontera (en este caso particular 'i'),
% diferentes tamaños del filtro (en este caso particular 4 puntos) y
% distintos niveles de multirresolución (en este caso particular 3
% niveles).

funciones={'fsoftpol2_m257'};
n=length(funciones);
%l=[2,3];
l=3;
nl=length(l);
com=[50 65 80 90 100];
m=length(com);
filtros={'Bisquare','Exponencial','Gaussiana','Huber','Lagrange',...
'SplineCuartico','SplineCubico','Uniforme'};
nf=length(filtros);
filtros1={'B','E','G','H','L','S4','S3','U'};
%puntos=[4 6 8];
puntos=4;
np=length(puntos);
%frontera={'p','s','c','b','i','a'};
frontera={'i'};
nfr=length(frontera);

t0=clock; % ordenes que sirven para medir el tiempo de computación
t=cputime;

% estilos para el dibujo incluyendo diferentes colores

estilo={'r','b','g','k','r-.','k-.','c','m'};
%filtros1={'B','E','G','H','L','S4','S3','U'};


for i=1:n
    %str=['../funciones 2D/',funciones{i},'.mat'];
    for q=1:np
        for t=1:nf
            for j=1:nl
                for s=1:nfr
                    str1=[num2str(l(j)),funciones{i},'_',filtros1{t},...
```

```
            num2str(puntos(q)),frontera{s},'.dat'];
            a=load(str1);
            er_inf{t}=a(:,4);
            er_2{t}=a(:,6);
        end
    end
end


% automatiza la impresión de las gráficas

figure(1)
h=[];
for t=1:8
    h1=plot([0.5 0.65 0.8 0.9 1], er_inf{t},estilo{t});
    h=[h,h1];
    hold on
end

title('Errores en la norma infinito');
legend(h,'Bisquare','Exponencial','Gaussiana','Huber','Lagrange',...
'Spline Cuártico','Spline Cúbico','Uniforme');
str1=['./Gráficas para Latex/EPS/','Infin','_',funciones{i},'_',num2str(puntos(q)),'.eps'];
print('-depsc',str1);
str1=['./Gráficas para Latex/PNG/','Infin','_',funciones{i},'_',num2str(puntos(q)),'.png'];
print('-dpng',str1);
str1=['./Gráficas para Latex/JPEG/','Infin','_',funciones{i},'_',num2str(puntos(q)),'.jpeg'];
print('-djpeg',str1);
close(1)

figure(2)
for t=1:8
    plot([0.5 0.65 0.8 0.9 1], er_2{t},estilo{t});
    hold on
end
title('Errores en la norma 2');
legend('Bisquare','Exponencial','Gaussiana','Huber','Lagrange',...
'Spline Cuártico','Spline Cúbico','Uniforme');
str1=['./Gráficas para Latex/EPS/','Norma2','_',funciones{i},...
'_',num2str(puntos(q)),'.eps'];
print('-depsc',str1);
```

```
        str1=['./Gráficas para Latex/PNG/','Norma2','_',funciones{i},...
        '_',num2str(puntos(q)),'.png'];
        print('-dpng',str1);
        str1=['./Gráficas para Latex/JPEG/','Norma2','_',funciones{i},...
        '_',num2str(puntos(q)),'.jpeg'];
        print('-djpeg',str1);
        close(2)


    end
end
```

4.2. Algoritmos de MR basados en MCP en 2D.

```
    function [a,mra]=valpun2(fichero,l,trun,fil,num,fron,met,str1)

% Este programa comprime datos bidimensionales mediante algoritmos de
multirresolución
% [a,mra]=valpun2(fichero,l,trun,fil,num,fron,met,str1);
% Variables de entrada:
% fichero guarda los datos de la función a la que se le aplica la multirreso-
lución
% l son los niveles de multirresolución
% trun indica el método de truncamiento:
% trun=[1,com] se queda con los %com detalles mayores
% trun=[0,tol] indica la tolerancia de truncamiento, se queda con los
% detalles mayores en valor absoluto
% fil filtro a utilizar:
% 'Bisquare' 'Exponencial'
```

```
% 'Gaussiana' 'Huber'
% 'Lagrange' 'SplineCuartico'
% 'SplineCubico' 'Uniforme'
% num número de puntos del filtro
% fron tipo de tratamiento en la frontera:
% 'p' periódico 'i' hacia el interior usando Lagrange
% 's' simétrico, 'a' antisimétrico
% 'c' extendido con ceros 'b' bajo orden
% met indica si se hace un cambio de los datos:
% 0 sin cambio
% 1 con cambio PPH
% str1 contiene el nombre del fichero donde se guardan los datos de salida
% Variables de salida:
% a aproximación a los datos después de truncar y decodificar
% mra versión de multirresolución procesada del vector original


% leemos del fichero los datos de la función
% en particular se cargan x1, a1,nombre_fun,tipo_fun

fichero=strcat('./ficheros de funciones 2D/',fichero);
load(fichero);


[n1,m1]=size(a1); % calcula el tamaño de los datos a1

t0=clock; % ordenes que sirven para medir el tiempo de computación
t=cputime;


% descendemos por la pirámide de multirresolución

mra=descenvp2(a1,l,fil,num,fron,met);


% trunca los detalles

[mra,rat]=truncarpv2(mra,l,trun);


% ascendemos por la pirámide de multirresolución
```

```
a=ascenvp2(mra,l,fil,num,fron,met);


tiempo=etime(clock,t0); % miden el tiempo de computación
tiempo1=cputime-t;


% calculamos los errores

norma1=sum(sum(abs(a-a1)))/n1/m1;
norma2_cuadrado=sum(sum((abs(a-a1)).^2))/n1/m1;
infin=max(max(abs(a-a1)));


% salida de resultados: en gráficas, por la pantalla y a un fichero


% dibujos

% dibujo de la función original

figure('Tag','orig_fun','Name',[blanks(6),'Función original'],...
'Position',[6 40 1270 690])
axes('Position',[0.05 0.05 0.9 0.9]);
mesh(a1);


% dibujo de la descomposición piramidal de los detalles

figure('Tag','det_fig','Name',[blanks(6),'Descomposición piramidal ...
de los detalles',blanks(2),met,blanks(6),'Escalas:',blanks(2),...
num2str(l)],'Position',[6 15 1270 690])
axes('Position',[0.05 0.05 0.9 0.9]);
spy(mra);


% dibujo de la función reconstruida después de la compresión
% incluir el nombre del filtro fil, la tolerancia, el método met

if fron=='p'
```

```
    fron='periódicas';
elseif fron=='i'
    fron='en el intervalo';
elseif fron=='s'
    fron='simétricas';
elseif fron=='a'
    fron='antisimétricas';
elseif fron=='c'
    fron='extensión con ceros';
elseif fron=='l'
    fron='bajando el orden'
end


if met==0
    met='sin cambio de datos';
elseif met==1
    met='con cambio PPH';
end


if trun(1)==1
    trun=num2str(trun(2));
    trun=[trun,'%'];
else
    trun=num2str(trun(2));
end

figure('Tag','reco_fig','Name',[blanks(6),'Función Reconstruida',...
blanks(6), blanks(2),fil,blanks(2),met,blanks(2),fron,blanks(2),trun,blanks(2),
'Escalas:',blanks(2),num2str(l)],'Position',[6 -8 1270 690])
axes('Position',[0.05 0.05 0.9 0.9]);
mesh(a);


% Mandamos los datos de salida a una figura en la pantalla
figure('Tag','dat_fig','Name',[blanks(6),'Datos de Salida para 2D'],...
'Position',[6 -8 1270 690]);
axis off

str={['Tipo de Función:',blanks(2),tipo_fun],['Nombre de la función:',blanks(2),nombre_fun],...
```

```
['Discontinuidades:',blanks(2),dis],...
['Niveles de Multirresolución:',blanks(2),num2str(l)],...
['Filtro:',blanks(2),fil],['Tamaño del filtro:',blanks(2),num2str(num)],...
['Método:',blanks(2),met],['Tratamiento en la frontera:',blanks(2),fron],...
['Tasa de Compresión:',blanks(2),num2str(rat)],...
['Norma Infinito del Error:',blanks(2),num2str(infin)],...
['Norma 1 del Error:',blanks(2),num2str(norma1)],...
['Norma 2 del Error:',blanks(2),num2str(norma2_cuadrado)],...
['Tiempo de cálculo:',blanks(2),num2str(tiempo),blanks(3),'segundos'],...
['Tiempo de CPU:',blanks(2),num2str(tiempo1),blanks(3),'segundos'],...
['Datos guardados en el fichero datos.txt' ]};


h1=uicontrol('Style','Text','String',str,'FontSize',16,'FontWeight',...
'bold','Units','normalized', 'Position',[0 0 1 0.95],...
'Background','White','HorizontalAlignment','left');

h2=uicontrol('Style','Text','String',...
{'Esquemas de Multirresolución en 2D con Diferentes Filtros'},'FontSize',18,...
'FontWeight','bold','Units','normalized','Position',[0 0.95 1 0.05],...
'Background','green','HorizontalAlignment','center','ForegroundColor','blue');



format long;


% fichero para generar las gráficas

fid=fopen(str1,'a');
fprintf(fid,'%d %d %f %4.16f %4.16f %4.16f %f %f \n',l,num,rat,...
infin,norma1,norma2_cuadrado,tiempo,tiempo1);
fclose(fid);


function d=descenvp2(d,l,fil,num,fron,met)

% Función de codificación en 2D para la discretización por valores
% puntuales
% d=descenvp2(a1,l,fil,num,fron,met)
% Variables de entrada:
```

```
% d secuencia 2D original
% l número de niveles de multirresolución considerados
% fil filtro a utilizar:
% 'Bisquare' 'Exponencial'
% 'Gaussiana' 'Huber'
% 'Lagrange' 'SplineCuartico'
% 'SplineCubico' 'Uniforme'
% num número de puntos del filtro
% fron tipo de tratamiento en la frontera:
% 'p' periódico 'i' hacia el interior usando Lagrange
% 's' simétrico, 'a' antisimétrico
% 'c' extendido con ceros 'b' bajo orden
% met indica si se hace un cambio de los datos:
% 0 sin cambio
% 1 con cambio PPH
% Variables de salida:
% d versión de multirresolución de los datos originales


% Inicialización de las variables

[width,height]=size(d);
wk=width; hk=height;


% Bucle para los niveles de multirresolución

for k=1:l

    % Decimamos para obtener un nivel inferior de resolución

    deci=d(1:2:wk,1:2:hk);


    % Se llama al operador de predicción

    pred=zeros(wk,hk);
    pred=prediccion2d(deci,fil,num,fron,met);

    % Se calculan los errores de predicción
```

```
    errores=zeros(wk,hk);
    errores=d(1:wk,1:hk)-pred(1:wk,1:hk);

    % Se reordena la matriz

    e1=errores(1:2:wk,1:2:hk); % detalles que deben ser cero
    e2=errores(1:2:wk,2:2:hk);
    e3=errores(2:2:wk,1:2:hk);
    e4=errores(2:2:wk,2:2:hk);

    d(1:wk,1:hk)=[deci,e2;e3,e4];

    % Comprobación de que los errores pertenecen al núcleo del operador de

    % decimación

    nz(k)=nnz((abs(e1)>10^(-15)).*e1);

    % Se actualiza el tamaño de la matriz de valores significativos

    wk=(wk+1)/2; hk=(hk+1)/2;

    % Se borran las variables auxiliares

    clear deci; clear pred; clear errores;

end


% Da un mensaje por pantalla comprobando que los errores pertenecen al
% núcleo del operador decimación

if nnz(nz)==0
    display('Efectivamente los errores pertenecen al núcleo del operador de-
cimación');
end


function pred=prediccion1d(a,fil,num,fron,met)

% Esta función calcula la predicción en 1D correspondiente al filtro
```

```
% utilizado y al método de cambio local de valores utilizado.
% Entorno de valores puntuales
% pred=prediccion1d(a,fil,num,met);
% Variables de entrada:
% a vector del nivel inferior de resolución
% fil filtro utilizado como máscaras del esquema de subdivision
% num número de puntos del filtro
% fron tipo de tratamiento en la frontera:
% 'p' periódico 'i' hacia el interior usando Lagrange
% 's' simétrico, 'a' antisimétrico
% 'c' extendido con ceros 'b' bajo orden
% met indica si se hace un cambio de los datos:
% 0 sin cambio
% 1 con cambio PPH
% Variables de salida:
% pred vector 1D con la predicción del nivel superior de resolución


% Inicialización de variables

n=length(a);
n1=2*n-1;

pred=zeros(1,n1);

% predicción de los valores impares, simplemente por proyección hacia
% arriba

pred(1:2:n1)=a(1:n);

% calcula las máscaras del filtro

cd './filtros numéricos'
fil=strcat(fil,'.mat');
load(fil);
cd ..

% máscaras para la frontera bajando el orden

masf={mas2,mas4,mas6,mas8};
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SIN TRANSFORMACIÓN DE LOS DATOS. CASO LINEAL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if met==0

   %%%%%%%%%%%%%
   % LINEAL CON 4 PUNTOS
   %%%%%%%%%%%%%

   if num==4

      % máscaras de Lagrange para la frontera izquierda con los datos disponibles

      for i=1:num/2-1
         mas(1:num,i)=maskslr(i,num-i);
      end

      % predicción del primer elemento

      if fron=='p'
         for i=1:num/2-1
            pred(2*i)=[a(n-num/2+i:n-1),a(1:i+num/2)]*mas4';
         end
      elseif fron=='i'
         pred(2:2:num-2)=a(1:num)*mas;
      elseif fron=='s'
         for i=1:num/2-1
            pred(2*i)=[a(num/2-i+1:-1:2),a(1:i+num/2)]*mas4';
         end
      elseif fron=='a'
         for i=1:num/2-1
            pred(2*i)=[2*a(1)-a(num/2-i+1:-1:2),a(1:i+num/2)]*mas4';
         end
      elseif fron=='c'
```

```
    for i=1:num/2-1
        pred(2*i)=[zeros(1,num/2-i),a(1:i+num/2)]*mas4';
    end
elseif fron=='b'
    for i=1:num/2-1
        pred(2*i)=[a(1:2*i)]*masf{i}';
    end
end



% predicción de los valores intermedios

for i=num/2:n-num/2

    aux=a(i-num/2+1:i+num/2)';
    pred(2*i)=mas4*aux;

end



% predicción del último elemento

if fron=='p'
    for i=n-1:-1:n-num/2+1
        pred(2*i)=[a(num/2+i-n+1:-1:2),a(n:-1:i-num/2+1)]*mas4';
    end
elseif fron=='i'
    pred(n1-1:-2:n1-num+3)=a(n:-1:n-num+1)*mas;
elseif fron=='s'
    for i=n-1:-1:n-num/2+1
        pred(2*i)=[a(2*n-num/2-i:n-1),a(n:-1:i-num/2+1)]*mas4';
    end
elseif fron=='a'
    for i=n-1:-1:n-num/2+1
        pred(2*i)=[2*a(n)-a(2*n-num/2-i:n-1),a(n:-1:i-num/2+1)]*mas4';
    end
elseif fron=='c'
    for i=n-1:-1:n-num/2+1
        pred(2*i)=[zeros(1,num/2+i-n),a(n:-1:i-num/2+1)]*mas4';
```

```
        end
    elseif fron=='b'
        for i=n-1:-1:n-num/2+1
            pred(2*i)=[a(n:-1:2*i-n+1)]*masf{n-i}';
        end
    end



    %%%%%%%%%%%%%
    % LINEAL CON 6 PUNTOS
    %%%%%%%%%%%%%


elseif num==6


    % máscaras de Lagrange para la frontera izquierda con los datos disponibles

    for i=1:num/2-1
        mas(1:num,i)=maskslr(i,num-i);
    end


    % predicción del primer elemento

    if fron=='p'
        for i=1:num/2-1
            pred(2*i)=[a(n-num/2+i:n-1),a(1:i+num/2)]*mas6';
        end
    elseif fron=='i'
        pred(2:2:num-2)=a(1:num)*mas;
    elseif fron=='s'
        for i=1:num/2-1
            pred(2*i)=[a(num/2-i+1:-1:2),a(1:i+num/2)]*mas6';
        end
    elseif fron=='a'
        for i=1:num/2-1
            pred(2*i)=[2*a(1)-a(num/2-i+1:-1:2),a(1:i+num/2)]*mas6';
        end
```

```
elseif fron=='c'
    for i=1:num/2-1
        pred(2*i)=[zeros(1,num/2-i),a(1:i+num/2)]*mas6';
    end
elseif fron=='b'
    for i=1:num/2-1
        pred(2*i)=[a(1:2*i)]*masf{i}';
    end
end


% predicción de los valores intermedios

for i=num/2:n-num/2

    aux=a(i-num/2+1:i+num/2)';
    pred(2*i)=mas6*aux;

end


% predicción del último elemento

if fron=='p'
    for i=n-1:-1:n-num/2+1
        pred(2*i)=[a(num/2+i-n+1:-1:2),a(n:-1:i-num/2+1)]*mas6';
    end
elseif fron=='i'
    pred(n1-1:-2:n1-num+3)=a(n:-1:n-num+1)*mas;
elseif fron=='s'
    for i=n-1:-1:n-num/2+1
        pred(2*i)=[a(2*n-num/2-i:n-1),a(n:-1:i-num/2+1)]*mas6';
    end
elseif fron=='a'
    for i=n-1:-1:n-num/2+1
        pred(2*i)=[2*a(n)-a(2*n-num/2-i:n-1),a(n:-1:i-num/2+1)]*mas6';
    end
elseif fron=='c'
    for i=n-1:-1:n-num/2+1
        pred(2*i)=[zeros(1,num/2+i-n),a(n:-1:i-num/2+1)]*mas6';
    end
```

```
    elseif fron=='b'
        for i=n-1:-1:n-num/2+1
            pred(2*i)=[a(n:-1:2*i-n+1)]*masf{n-i}';
        end
    end


    %%%%%%%%%%%%
    % LINEAL CON 8 PUNTOS
    %%%%%%%%%%%%


elseif num==8


    % máscaras de Lagrange para la frontera izquierda con los datos disponibles

    for i=1:num/2-1
        mas(1:num,i)=maskslr(i,num-i);
    end


    % predicción del primer elemento

    if fron=='p'
        for i=1:num/2-1
            pred(2*i)=[a(n-num/2+i:n-1),a(1:i+num/2)]*mas8';
        end
    elseif fron=='i'
        pred(2:2:num-2)=a(1:num)*mas;
    elseif fron=='s'
        for i=1:num/2-1
            pred(2*i)=[a(num/2-i+1:-1:2),a(1:i+num/2)]*mas8';
        end
    elseif fron=='a'
        for i=1:num/2-1
            pred(2*i)=[2*a(1)-a(num/2-i+1:-1:2),a(1:i+num/2)]*mas8';
        end
    elseif fron=='c'
        for i=1:num/2-1
```

```
            pred(2*i)=[zeros(1,num/2-i),a(1:i+num/2)]*mas8';
        end
    elseif fron=='b'
        for i=1:num/2-1
            pred(2*i)=[a(1:2*i)]*masf{i}';
        end
    end


    % predicción de los valores intermedios

    for i=num/2:n-num/2

        aux=a(i-num/2+1:i+num/2)';
        pred(2*i)=mas8*aux;

    end


    % predicción del último elemento

    if fron=='p'
        for i=n-1:-1:n-num/2+1
            pred(2*i)=[a(num/2+i-n+1:-1:2),a(n:-1:i-num/2+1)]*mas8';
        end
    elseif fron=='i'
        pred(n1-1:-2:n1-num+3)=a(n:-1:n-num+1)*mas;
    elseif fron=='s'
        for i=n-1:-1:n-num/2+1
            pred(2*i)=[a(2*n-num/2-i:n-1),a(n:-1:i-num/2+1)]*mas8';
        end
    elseif fron=='a'
        for i=n-1:-1:n-num/2+1
            pred(2*i)=[2*a(n)-a(2*n-num/2-i:n-1),a(n:-1:i-num/2+1)]*mas8';
        end
    elseif fron=='c'
        for i=n-1:-1:n-num/2+1
            pred(2*i)=[zeros(1,num/2+i-n),a(n:-1:i-num/2+1)]*mas8';
        end
    elseif fron=='b'
        for i=n-1:-1:n-num/2+1
```

```
            pred(2*i)=[a(n:-1:2*i-n+1)]*masf{n-i}';
        end
    end



end


 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 % CON TRANSFORMACIÓN DE LOS DATOS. PPH
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%


elseif met==1


 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 % CAMBIO PPH + FILTRO LINEAL CON 4 PUNTOS
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%


if num==4


     % máscaras de Lagrange para la frontera izquierda con los datos dis-
ponibles

    for i=1:num/2-1
        mas(1:num,i)=maskslr(i,num-i);
    end


     % predicción del primer elemento

    if fron=='p'
        for i=1:num/2-1
            pred(2*i)=[a(n-num/2+i:n-1),a(1:i+num/2)]*mas4';
        end
    elseif fron=='i'
        pred(2:2:num-2)=a(1:num)*mas;
```

```
elseif fron=='s'
    for i=1:num/2-1
        pred(2*i)=[a(num/2-i+1:-1:2),a(1:i+num/2)]*mas4';
    end
elseif fron=='a'
    for i=1:num/2-1
        pred(2*i)=[2*a(1)-a(num/2-i+1:-1:2),a(1:i+num/2)]*mas4';
    end
elseif fron=='c'
    for i=1:num/2-1
        pred(2*i)=[zeros(1,num/2-i),a(1:i+num/2)]*mas4';
    end
elseif fron=='b'
    for i=1:num/2-1
        pred(2*i)=[a(1:2*i)]*masf{i}';
    end
end


% predicción de los valores intermedios


for i=num/2:n-num/2


    aux=a(i-num/2+1:i+num/2);
    aux=f_modify(aux)';
    pred(2*i)=mas4*aux;

end


% predicción del último elemento

if fron=='p'
    for i=n-1:-1:n-num/2+1
        pred(2*i)=[a(num/2+i-n+1:-1:2),a(n:-1:i-num/2+1)]*mas4';
    end
elseif fron=='i'
    pred(n1-1:-2:n1-num+3)=a(n:-1:n-num+1)*mas;
elseif fron=='s'
```

```
        for i=n-1:-1:n-num/2+1
            pred(2*i)=[a(2*n-num/2-i:n-1),a(n:-1:i-num/2+1)]*mas4';
        end
    elseif fron=='a'
        for i=n-1:-1:n-num/2+1
            pred(2*i)=[2*a(n)-a(2*n-num/2-i:n-1),a(n:-1:i-num/2+1)]*mas4';
        end
    elseif fron=='c'
        for i=n-1:-1:n-num/2+1
            pred(2*i)=[zeros(1,num/2+i-n),a(n:-1:i-num/2+1)]*mas4';
        end
    elseif fron=='b'
        for i=n-1:-1:n-num/2+1
            pred(2*i)=[a(n:-1:2*i-n+1)]*masf{n-i}';
        end
    end


    %%%%%%%%%%%%%%%%%%%%%%%%%%
    % CAMBIO PPH + FILTRO LINEAL CON 6 PUNTOS
    %%%%%%%%%%%%%%%%%%%%%%%%%%


elseif num==6


    % máscaras de Lagrange para la frontera izquierda con los datos disponibles

    for i=1:num/2-1
        mas(1:num,i)=maskslr(i,num-i);
    end


    % predicción del primer elemento

    if fron=='p'
        for i=1:num/2-1
            pred(2*i)=[a(n-num/2+i:n-1),a(1:i+num/2)]*mas6';
        end
    elseif fron=='i'
```

```
    pred(2:2:num-2)=a(1:num)*mas;
elseif fron=='s'
    for i=1:num/2-1
        pred(2*i)=[a(num/2-i+1:-1:2),a(1:i+num/2)]*mas6';
    end
elseif fron=='a'
    for i=1:num/2-1
        pred(2*i)=[2*a(1)-a(num/2-i+1:-1:2),a(1:i+num/2)]*mas6';
    end
elseif fron=='c'
    for i=1:num/2-1
        pred(2*i)=[zeros(1,num/2-i),a(1:i+num/2)]*mas6';
    end
elseif fron=='b'
    for i=1:num/2-1
        pred(2*i)=[a(1:2*i)]*masf{i}';
    end
end


% predicción de los valores intermedios

for i=num/2:n-num/2

    aux=a(i-num/2+1:i+num/2);
    aux=f_modify(aux)';
    pred(2*i)=mas6*aux;

end


% predicción del último elemento

if fron=='p'
    for i=n-1:-1:n-num/2+1
        pred(2*i)=[a(num/2+i-n+1:-1:2),a(n:-1:i-num/2+1)]*mas6';
    end
elseif fron=='i'
    pred(n1-1:-2:n1-num+3)=a(n:-1:n-num+1)*mas;
elseif fron=='s'
    for i=n-1:-1:n-num/2+1
```

```
                pred(2*i)=[a(2*n-num/2-i:n-1),a(n:-1:i-num/2+1)]*mas6';
            end
        elseif fron=='a'
            for i=n-1:-1:n-num/2+1
                pred(2*i)=[2*a(n)-a(2*n-num/2-i:n-1),a(n:-1:i-num/2+1)]*mas6';
            end
        elseif fron=='c'
            for i=n-1:-1:n-num/2+1
                pred(2*i)=[zeros(1,num/2+i-n),a(n:-1:i-num/2+1)]*mas6';
            end
        elseif fron=='b'
            for i=n-1:-1:n-num/2+1
                pred(2*i)=[a(n:-1:2*i-n+1)]*masf{n-i}';
            end
        end



        %%%%%%%%%%%%%%%%%%%%%%%%%
        % CAMBIO PPH + FILTRO LINEAL CON 8 PUNTOS
        %%%%%%%%%%%%%%%%%%%%%%%%%



    elseif num==8


        % máscaras de Lagrange para la frontera izquierda con los datos dis-
ponibles

        for i=1:num/2-1
            mas(1:num,i)=maskslr(i,num-i);
        end


        % predicción del primer elemento

        if fron=='p'
            for i=1:num/2-1
                pred(2*i)=[a(n-num/2+i:n-1),a(1:i+num/2)]*mas8';
            end
```

```
elseif fron=='i'
    pred(2:2:num-2)=a(1:num)*mas;
elseif fron=='s'
    for i=1:num/2-1
        pred(2*i)=[a(num/2-i+1:-1:2),a(1:i+num/2)]*mas8';
    end
elseif fron=='a'
    for i=1:num/2-1
        pred(2*i)=[2*a(1)-a(num/2-i+1:-1:2),a(1:i+num/2)]*mas8';
    end
elseif fron=='c'
    for i=1:num/2-1
        pred(2*i)=[zeros(1,num/2-i),a(1:i+num/2)]*mas8';
    end
elseif fron=='b'
    for i=1:num/2-1
        pred(2*i)=[a(1:2*i)]*masf{i}';
    end
end


% predicción de los valores intermedios

for i=num/2:n-num/2

    aux=a(i-num/2+1:i+num/2);
    aux=f_modify(aux)';
    pred(2*i)=mas8*aux;

end


% predicción del último elemento

if fron=='p'
    for i=n-1:-1:n-num/2+1
        pred(2*i)=[a(num/2+i-n+1:-1:2),a(n:-1:i-num/2+1)]*mas8';
    end
elseif fron=='i'
    pred(n1-1:-2:n1-num+3)=a(n:-1:n-num+1)*mas;
elseif fron=='s'
```

```
            for i=n-1:-1:n-num/2+1
                pred(2*i)=[a(2*n-num/2-i:n-1),a(n:-1:i-num/2+1)]*mas8';
            end
        elseif fron=='a'
            for i=n-1:-1:n-num/2+1
                pred(2*i)=[2*a(n)-a(2*n-num/2-i:n-1),a(n:-1:i-num/2+1)]*mas8';
            end
        elseif fron=='c'
            for i=n-1:-1:n-num/2+1
                pred(2*i)=[zeros(1,num/2+i-n),a(n:-1:i-num/2+1)]*mas8';
            end
        elseif fron=='b'
            for i=n-1:-1:n-num/2+1
                pred(2*i)=[a(n:-1:2*i-n+1)]*masf{n-i}';
            end
        end



    end


end


function pred=prediccion2d(a,fil,num,fron,met)

% Esta función calcula la predicción usando la técnica de producto tensor
% y la predicción en 1D correspondiente al filtro utilizado y al método
% de cambio local de valores utilizado. Entorno de valores puntuales
% pred=prediccion2d(a,fil,num,fron,met);
% Variables de entrada:
% a matriz del nivel inferior de resolución
% fil filtro utilizado como máscaras del esquema de subdivision
% num número de puntos del filtro
% fron tipo de tratamiento en la frontera:
% 'p' periódico 'i' hacia el interior usando Lagrange
% 's' simétrico, 'a' antisimétrico
% 'c' extendido con ceros 'b' bajo orden
% met indica si se hace un cambio de los datos:
```

```
% 0 sin cambio
% 1 con cambio PPH
% Variables de salida:
% pred matriz 2D con la predicción del nivel superior de resolución


% Inicialización de variables

[w,h]=size(a);
w1=2*w-1; h1=2*h-1;

pred=zeros(w1,h1);
pred(1:w,1:h)=a;


% Se hace la predicción para cada una de las filas

for i=1:h
    pred(1:w1,i)=prediccion1d(pred(1:w,i)',fil,num,fron,met)';
end

% Se hace la predicción para cada una de las columnas

for i=1:w1
    pred(i,1:h1)=prediccion1d(pred(i,1:h),fil,num,fron,met);
end



function [c,rat]=truncarpv2(a,l,trun)

% Función que trunca los detalles de la matriz multirresolucionada
% en el entorno de discretización por valores puntuales
% [c,rat]=truncarpv(a,l,trun)
% Variables de entrada:
% a matriz a la que truncarle los detalles
% l niveles de multirresolución
% trun indica el método de truncamiento:
% trun=[1,com] se queda con los  %com detalles mayores
% trun=[0,tol] indica la tolerancia de truncamiento, se queda con los
% detalles mayores en valor absoluto
```

```
% Variables de salida:
% c matriz con los detalles truncados
% rat porcentaje de detalles no cero


[n,m]=size(a);
n1=(n-1)/2^l+1; m1=(m-1)/2^l+1;

if trun(1)==0 % truncamos los detalles menores que una tolerancia tol

    c=[a(1:n1,1:m1), a(1:n1,m1+1:m).*(abs(a(1:n1,m1+1:m))>=trun(2));...
    a(n1+1:n,1:m1).*(abs(a(n1+1:n,1:m1))>=trun(2)),...
    a(n1+1:n,m1+1:m).*(abs(a(n1+1:n,m1+1:m))>=trun(2)) ];

    aux=c(1:n1,1:m1); c(1:n1,1:m1)=0;
    nz=nnz(c);

    rat=nz*100/(n*m-n1*m1);
    c(1:n1,1:m1)=aux;

else


    % Nos quedamos con los num_det mayores coeficientes de detalle

    num_det=floor(trun(2)*(n*m-n1*m1)/100);


    % Nos quedamos con los coeficientes significativos mayores

    c=a;
    c(1:n1,1:m1)=0; % eliminamos los coeficientes de la última escala,
    % que no son susceptibles de ser truncados
    aux=c(:); % obtenemos un vector columna
    [aux_Orden,ind]=sort(abs(aux),'descend'); % los ordenamos según el va-
lor absoluto
    % de los coeficientes en una columna
    aux_Orden=aux(ind); % obtenemos los valores ordenados
    % sin valor absoluto
    aux_Orden(num_det+1:end)=0; % eliminamos los menos significativos
    aux(ind)=aux_Orden; % llevamos los mayores coeficientes a su posición
```

```
    c(1:end)=aux; % rellenamos la matriz de multirresolución sólo con
     % los detalles más significativos
    nz=nnz(c);
    rat=nz*100/(n*m-n1*m1); % calcula el ratio de compresión según
     % la definición utilizada
    c(1:n1,1:m1)=a(1:n1,1:m1); % restauramos los coeficientes significativos
     % de la última escala


end


function a=ascenvp2(mra,l,fil,num,fron,met)

% Este programa asciende por la pirámide de multirresolución para datos
bidimensionales
% en el entorno de valores puntuales
% a=ascenvp2(mra,l,fil,num,fron,met)
% Variables de entrada:
% mra versión de multirresolución truncada de los datos originales
% l son los niveles de multirresolución
% fil filtro a utilizar:
% 'Bisquare' 'Exponencial'
% 'Gaussiana' 'Huber'
% 'Lagrange' 'SplineCuartico'
% 'SplineCubico' 'Uniforme'
% num número de puntos del filtro
% fron tipo de tratamiento en la frontera:
% 'p' periódico 'i' hacia el interior usando Lagrange
% 's' simétrico, 'a' antisimétrico
% 'c' extendido con ceros 'b' bajo orden
% met indica si se hace un cambio de los datos:
% 0 sin cambio
% 1 con cambio PPH
% Variables de salida:
% a aproximación a los datos originales despúes de truncar y decodificar


% Inicialización de las variables

[width,height]=size(mra);
```

```
wk=(width-1)/2^l+1; hk=(height-1)/2^l+1;


% Bucle para los niveles de multirresolución

for k=l:-1:1

    % Dimensiones de la escala superior

    wk1=2*wk-1; hk1=2*hk-1;

    % Calculamos la predicción desde el nivel inferior

    pred=zeros(wk1,hk1);
    pred=prediccion2d(mra(1:wk,1:hk),fil,num,fron,met);

    % Sumamos las predicciones más los correspondientes errores para los 3
    % tipos de errores

    escala_sup=zeros(wk1,hk1);
    % =pred(1:2:wk1,1:2:hk1);Consistencia
    escala_sup(1:2:wk1,1:2:hk1)=mra(1:wk,1:hk);
    % errores e2
    escala_sup(1:2:wk1,2:2:hk1)=pred(1:2:wk1,2:2:hk1)+mra(1:wk,hk+1:hk1);
    % errores e3
    escala_sup(2:2:wk1,1:2:hk1)=pred(2:2:wk1,1:2:hk1)+mra(wk+1:wk1,1:hk);
    % errores e4
    escala_sup(2:2:wk1,2:2:hk1)=pred(2:2:wk1,2:2:hk1)+mra(wk+1:wk1,hk+1:hk1);

    % Comprobación de que el operador de predicción cumple
    % la relación de consistencia (debe salir cero)

    pred_consis(k)=max(max(abs(pred(1:2:wk1,1:2:hk1)-mra(1:wk,1:hk))));

    % Actualización de la matriz de multirresolución

    mra(1:wk1,1:hk1)=escala_sup(1:wk1,1:hk1);

    % Actualizamos las variables

    wk=2*wk-1; hk=2*hk-1;
```

```
    clear pred; clear escala_sup;
end

if nnz((abs(pred_consis)>10^(-15)).*pred_consis)==0
    display('Operador de predicción consistente');
end

a=mra;


% Este guion automatiza la ejecución del programa principal 'valpun2.m'

% para una función concreta(en este caso particular 'fexponencialDisPara-
bola2D')
% con diferentes tipos de filtros, diferentes valores de compresión,
% diferente tratamiento de la frontera (en este caso particular 'i'),
% diferentes tamaños del filtro (en este caso particular 8 puntos) y
% distintos niveles de multirresolución (en este caso particular 3 niveles).

funciones={'fexponencialDisParabola2D_n257_m257'};
n=length(funciones);
%l=[2,3];
l=3;
nl=length(l);
com=[50 65 80 90 100];
m=length(com);
filtros={'Bisquare','Exponencial','Gaussiana','Huber','Lagrange',...
'SplineCuartico','SplineCubico','Uniforme'};
nf=length(filtros);
filtros1={'B','E','G','H','L','S4','S3','U'};
%puntos=[4 6 8];
puntos=8;
np=length(puntos);
%frontera={'p','s','c','b','i','a'};
frontera={'i'};
nfr=length(frontera);

t0=clock; % ordenes que sirven para medir el tiempo de computación
t=cputime;
```

```
for i=1:n
    str=[funciones{i},'.mat'];
    for j=1:nl
        for k=1:m
            for t=1:nf
                for q=1:np
                    for s=1:nfr
                        str1=['./Datos_salida 2D/',num2str(l(j)),funciones{i},'_',...
                        filtros1{t},num2str(puntos(q)),frontera{s},'.dat'];
                        [a,mra]=valpun2(str,l(j),[1,com(k)],filtros{t},puntos(q),...
                        frontera{s},0,str1);
                        [a,mra]=valpun2(str,l(j),[1,com(k)],filtros{t},puntos(q),...
                        frontera{s},1,str1);

                    end
                end
            end
        end
    end
end



tiempo=etime(clock,t0) % miden el tiempo de computación
tiempo1=cputime-t


% Este programa genera las gráficas 2D de la función en cuestion
%(en este caso particular 'fexponencialDisParabola2D_n257_m257')
% con diferentes tipos de filtros, diferentes valores de compresión,
% diferente tratamiento de la frontera(en este caso particular 'i'),
% diferentes tamaños del filtro (en este caso particular 8 puntos) y
% distintos niveles de multirresolución (en este caso particular 3
% niveles).

funciones={'fexponencialDisParabola2D_n257_m257'};
n=length(funciones);
%l=[2,3];
l=3;
nl=length(l);
com=[50 65 80 90 100];
```

```
m=length(com);
filtros={'Bisquare','Exponencial','Gaussiana','Huber','Lagrange',...
'SplineCuartico','SplineCubico','Uniforme'};
nf=length(filtros);
filtros1={'B','E','G','H','L','S4','S3','U'};
%puntos=[4 6 8];
puntos=8;
np=length(puntos);
%frontera={'p','s','c','b','i','a'};
frontera={'i'};
nfr=length(frontera);

t0=clock; % ordenes que sirven para medir el tiempo de computación
t=cputime;

% estilos para el dibujo incluyendo diferentes colores

estilo={'r','b','g','k','r-.','k-.','c','m'};
%filtros1={'B','E','G','H','L','S4','S3','U'};


for i=1:n
    for q=1:np
        for t=1:nf
            for j=1:nl
                for s=1:nfr
                    str1=[num2str(l(j)),funciones{i},'_',filtros1{t},...
                    num2str(puntos(q)),frontera{s},'.dat'];
                    a=load(str1);
                    er_inf{t}=a(:,4);
                    er_2{t}=a(:,6);
                end
            end
        end


        % automatiza la impresión de las gráficas

        figure(1)
        h=[];
        for t=1:8
```

```
        h1=plot([0.5 0.65 0.8 0.9 1], er_inf{t},estilo{t});
        h=[h,h1];
        hold on
    end

    title('Errores en la norma infinito');
    legend(h,'Bisquare','Exponencial','Gaussiana','Huber','Lagrange',...
    'Spline Cuártico','Spline Cúbico','Uniforme');
    str1=['./Gráficas para Latex 2D/EPS8/','Infin','_',funciones{i},...
    '_',num2str(puntos(q)),'.eps'];
    print('-depsc',str1);
    str1=['./Gráficas para Latex 2D/PNG8/','Infin','_',funciones{i},'_',...
    num2str(puntos(q)),'.png'];
    print('-dpng',str1);
    str1=['./Gráficas para Latex 2D/JPEG8/','Infin','_',funciones{i},'_',...
    num2str(puntos(q)),'.jpeg'];
    print('-djpeg',str1);
    close(1)

    figure(2)
    for t=1:8
        plot([0.5 0.65 0.8 0.9 1], er_2{t},estilo{t});
        hold on
    end
    title('Errores en la norma 2');
    legend('Bisquare','Exponencial','Gaussiana','Huber','Lagrange',...
    'Spline Cuártico','Spline Cúbico','Uniforme');
    str1=['./Gráficas para Latex 2D/EPS8/','Norma2','_',funciones{i},'_',...
    num2str(puntos(q)),'.eps'];
    print('-depsc',str1);
    str1=['./Gráficas para Latex 2D/PNG8/','Norma2','_',funciones{i},'_',...
    num2str(puntos(q)),'.png'];
    print('-dpng',str1);
    str1=['./Gráficas para Latex 2D/JPEG8/','Norma2','_',funciones{i},'_',...
    num2str(puntos(q)),'.jpeg'];
    print('-djpeg',str1);
    close(2)

  end
end
```

Capítulo 5

Interfaz Gráfica.

Se ha utilizado el entorno gráfico de Matlab (GUIDE) para codificar los algoritmos necesarios, con el objetivo de que el usuario pueda interactuar de una manera sencilla con el programa.

5.1. Ejecución de la Interfaz Gráfica

Una vez abierto Matlab, debemos situar el directorio de trabajo en el lugar adecuado. Esto es, en la ruta

"./PFC/Interfaz Gráfica".

Para ejecutar la interfaz gráfica, debemos introducir en el prompt de Matlab el siguiente comando

$>> inicio$

Aparece la pantalla de "Bienvenida" describiendo información general del proyecto (Figura 5.1).

Dándole a "CONTINUAR" nos lleva a la interfaz del programa principal de usuario descrita en la siguiente sección (Figura 5.2). En el caso de querer acceder directamente al programa principal de usuario, saltándonos la pantalla de bienvenida, deberemos ejecutar en el prompt

$>> main$

5.2. Documentación de la Interfaz Gráfica

A continuación procederemos a describir la interfaz principal de usuario (Figura 5.2), dividióndola en dos partes: Menú y Cuadro Central.

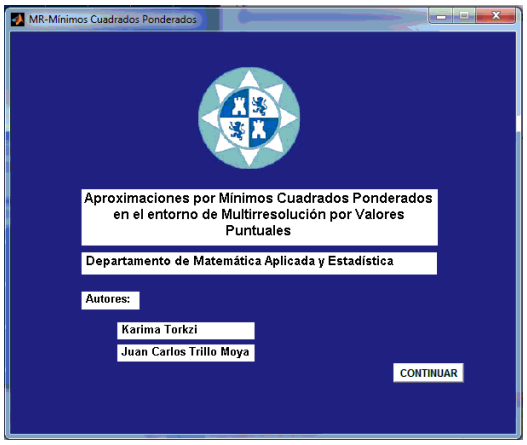

Figura 1.1: Pantalla de bienvenida.

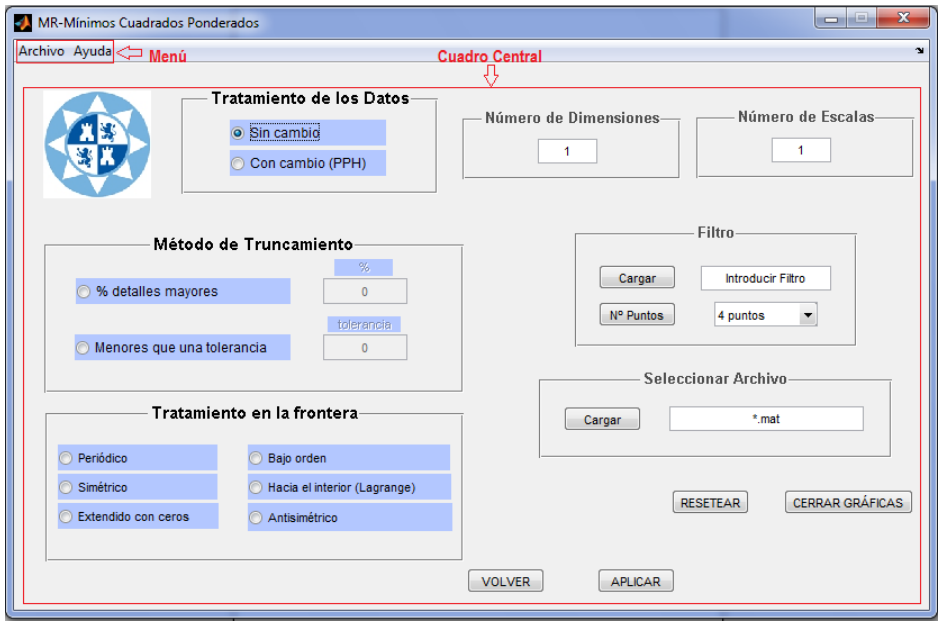

Figura 1.2: Interfaz gráfica principal de usuario.

1.2.1. Menú Principal

La interfaz principal de usuario cuenta con un menú en la parte superior izquierda (Figura 1.3) que consiste en una lista de opciones que puede desplegarse para llevar a cabo las diferentes configuraciones.

Vamos a describir cada submenú contenido en el menú de la interfaz.

- **Archivo**

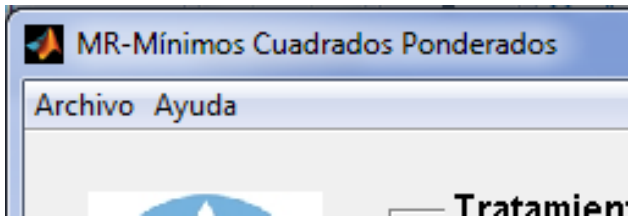

Figura 1.3: Menú de usuario.

En este menú podremos cargar el archivo de datos original sobre el que operaremos y el filtro de reconstrucción a aplicar (Figura 1.4). Se trata de una de las acciones más habituales a la hora de trabajar con cualquier aplicación. Generalmente, la operación consiste en acceder a un directorio y abrir un archivo.

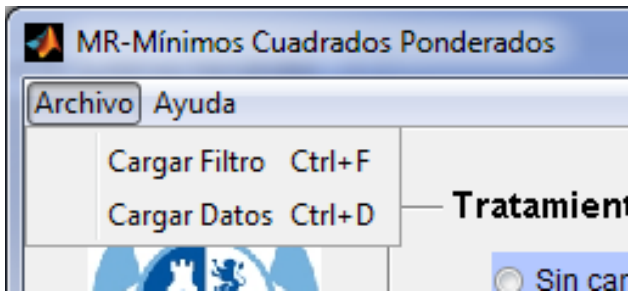

Figura 1.4: Submenús de archivo.

- **Cargar filtro**
  Deberemos buscar y cargar los archivos adecuados . Así, iremos accediendo a directorios hasta llegar a la carpeta de filtros correspondiente. Una vez dentro, seleccionaremos el filtro deseado (Figura 1.5).

  En este proyecto hemos considerado ocho posibles filtros: *Bisquare, Exponencial, Gaussiana, Huber, Lagrange, Spline Cuártico, Spline Cúbico* y *Uniforme* .

- **Cargar Datos**
  Deberemos buscar y cargar los archivos adecuados en función de la dimensión(1D o 2D). Teniendo en cuenta esto, avanzaremos por los directorios hasta llegar a la carpeta de datos que corresponda, seleccionando el archivo deseado (Figura 1.6).

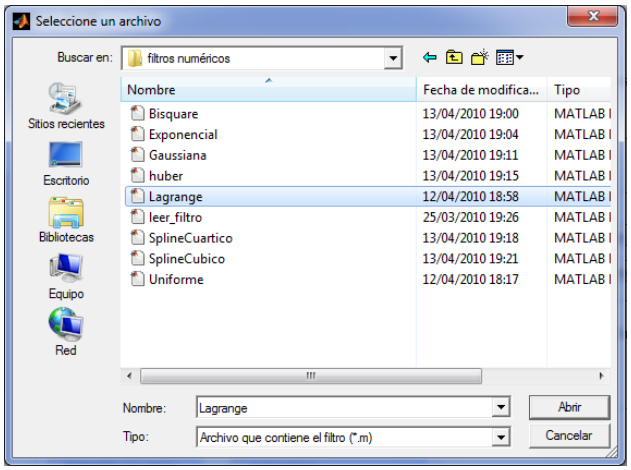

Figura 1.5: Cuadro de diálogo para cargar filtro de reconstrucción.

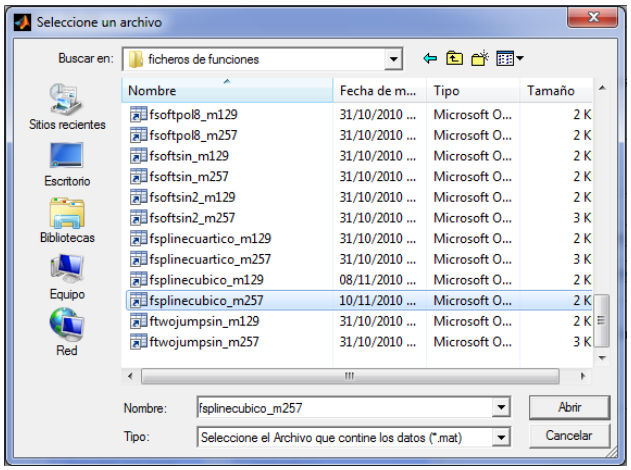

Figura 1.6: Cuadro de diálogo para cargar los datos.

- **Ayuda**
  Este menú contiene un tutorial del manejo de la interfaz gráfica e información general acerca del proyecto (Figura 1.7).

  - **Tutorial**
    Selecionando esta opción se abrirá un tutorial en formato pdf que contiene indicaciones sobre el manejo de la interfaz gráfica.
  - **Acerca de**
    Seleccionando esta opción recibiremos información general acerca del proyecto (Figura 1.8).

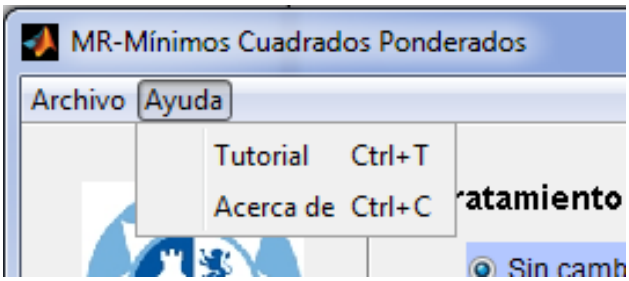

Figura 1.7: Submenús de ayuda.

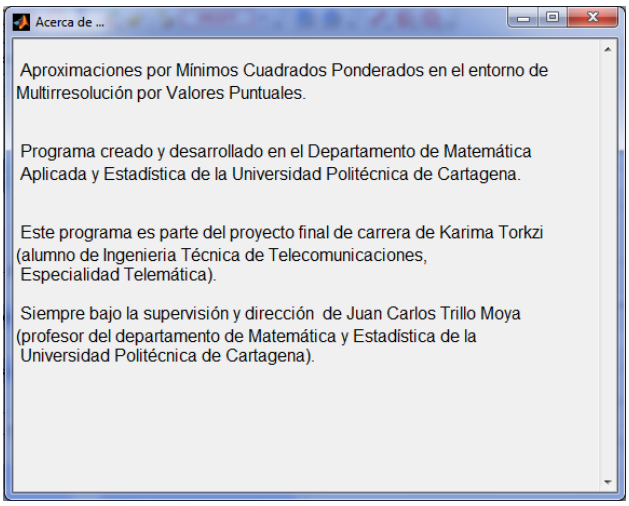

Figura 1.8: Información general acerca del proyecto.

1.2.2. Cuadro Central

Por "Cuadro Central" entendemos el resto de campos que forman la Interfaz Gráfica (Figura 1.2). Debemos configurar cada campo según los requerimientos. Iremos explicando campo a campo el significado de cada uno de ellos. Se aconseja ir accediendo y configurando los campos según el orden descrito a continuación.

- **Tratamiento de los Datos**
  Debemos elegir entre el tratamiento sin cambio de datos o con cambio de datos (PPH). Podremos marcar ambos simultáneamente (Figura 1.9).

- **Método de Truncamiento**
  Este cuadro hace referencia a la gestión del método de truncamiento. Debemos marcar una de las opciones dadas (Figura 1.10).

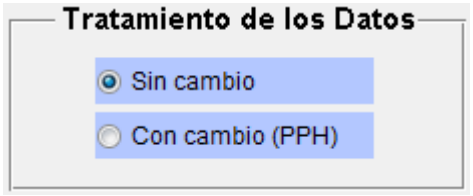

Figura 1.9: Cuadro del Tratamiento de los Datos.

Figura 1.10: Cuadro del Método de Truncamiento .

- % detalles mayores
  Esta opción lleva asociado el campo "%". En dicho campo se debe definir un valor que hace referencia al porcentaje de coeficientes que serán truncados. Hay que tener en cuenta que

$$0 \leq \text{compresión} \leq 100 \tag{1.1}$$

- Menores que une tolerancia
  Esta opción lleva asociado el campo "tolerancia". En dicho campo se debe definir un valor por debajo del cual se realiza el truncamiento en los algoritmos correspondientes. Hay que tener en cuenta que

$$\text{tolerancia} \geq 0 \tag{1.2}$$

- **Tratamiento en la frontera**
  Este cuadro hace referencia al tipo de tratamiento de la frontera que queremos seleccionar,nos da ocho opciones a elegir: *periódico, simétrico, extendido con ceros, bajo orden, hacia el interior (Lagrange)* y *antisimétrico*. Debemos marcar una sola opción(Figura 1.11).

- **Número de Dimensiones**
  Como el nombre del campo indica, se deben introducir las dimensiones deseadas (Figura 1.12). En nuestro caso, una o dos dimensiones. Luego hay que tener en cuenta que

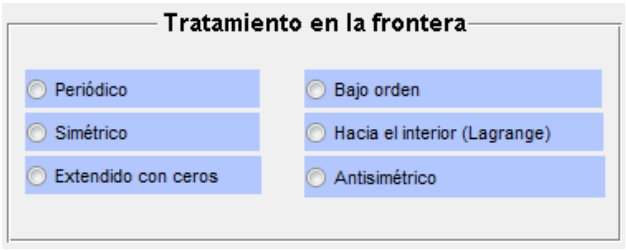

Figura 1.11: Cuadro del Tratamiento de la frontera.

$$0 < \text{número de dimensiones} \leq 2 \tag{1.3}$$

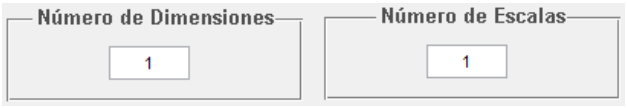

Figura 1.12: Número de dimensiones y número de escalas.

- **Número de Escalas**
  Este campo hace referencia al número de escalas que vamos a descender en el algoritmo de multirresolución (Figura 1.12). Se debe considerar

$$\text{número de escalas} \geq 1 \tag{1.4}$$

- **Filtro**
  La descripción de este campo es la misma que en la sección anterior, veáse apartado "Cargar Filtro" en la sección 1.2.1. La configuración de dicho campo se puede llevar a cabo de ambas maneras, quedando a disposición del usuario la elección de una u otra según su comodidad (Figura 1.13).

Figura 1.13: Filtro de reconstrucción.

Para escoger el número de puntos adecuado, simplemente desplegaremos el menú "Nº Puntos" y seleccionaremos el número de puntos deseado.

- **Seleccionar el Archivo**
  Al igual que sucede con el campo "Filtro", la descripción coincide con la sección anterior, veáse apartado "Cargar Datos" en la sección 1.2.1.

Figura 1.14: Seleccionar archivo.

- **Botones de la Interfaz**
  - **RESETEAR:** resetea todos los campos de la interfaz a sus valores de inicio, además, se cierran todas las gráficas que están abiertas.
  - **CERRAR GRÁFICAS:** se cierran todas las gráficas abiertas. A diferencia de "RESETEAR" no afecta a los valores de los campos de la interfaz.
  - **VOLVER:** volvemos a la pantalla de "Bienvenida" de la interfaz (Figura 1.1).
  - **APLICAR:** ejecuta el programa y muestra los resultados obtenidos.
    Es importante resaltar que siempre se muestra una ventana que recoge un resumen de la configuración y los resultados obtenidos. (Figura 1.15)
    Además, con cada ejecución del programa, se van guardando los resultados obtenidos en un fichero de texto con la siguiente ruta

    Para $1D$:
    "./datos.txt".
    Para $2D$:
    "./datos2D.txt".

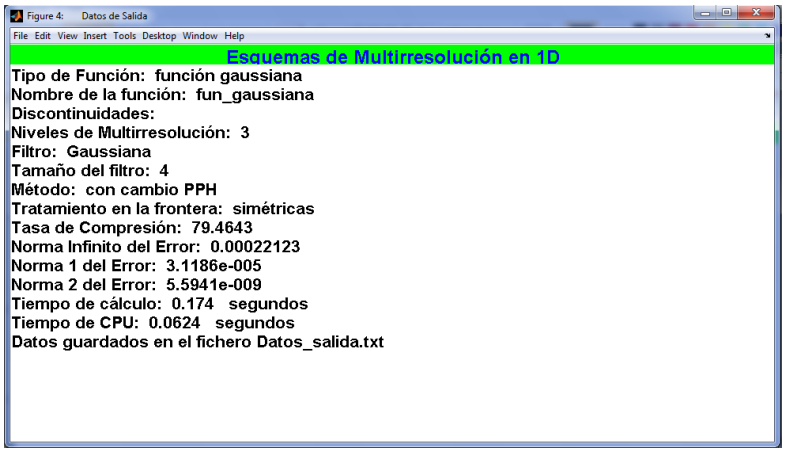

Figura 1.15: Ejemplo de ventana que contiene los resultados de salida.

1.3. Ejemplo de uso de la Interfaz Gráfica

A continuación se explica un ejemplo práctico del uso de la interfaz. Vamos completando los campos secuencialmente como se aconsejó en apartados anteriores, de esta manera, una de las posibles configuraciones del "Cuadro Central" podría ser la que se contempla en la Figura 1.16.

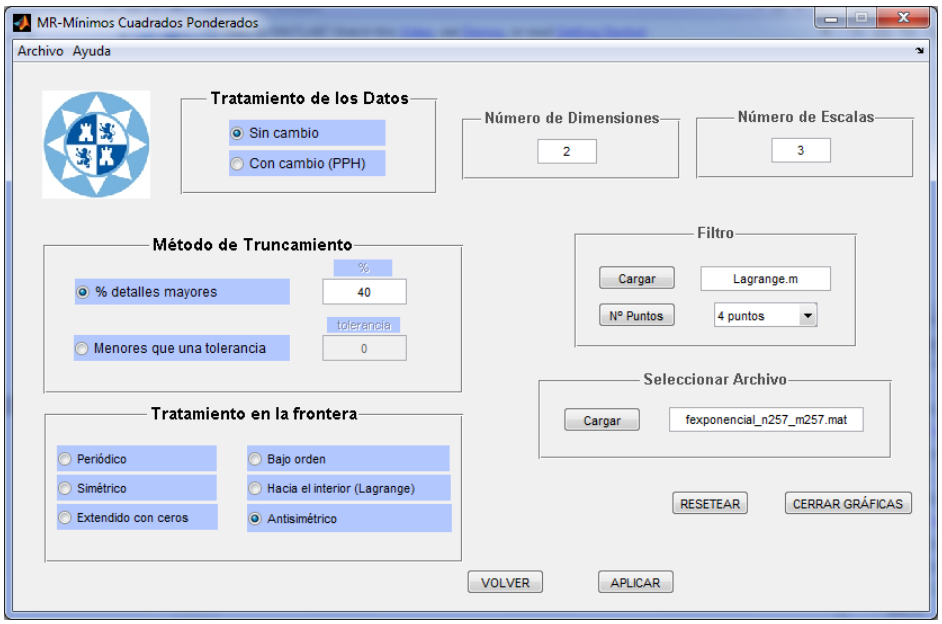

Figura 1.16: Ejemplo de configuración del cuadro central.

Una vez configurados todos los campos, le damos a “APLICAR”. Como hemos comentado con anterioridad, las pantallas mostrando los resultados varían de una configuración a otra, en este caso las pantallas resultantes se corresponden con las Figuras 1.17, 1.18, 1.19 y 1.20.

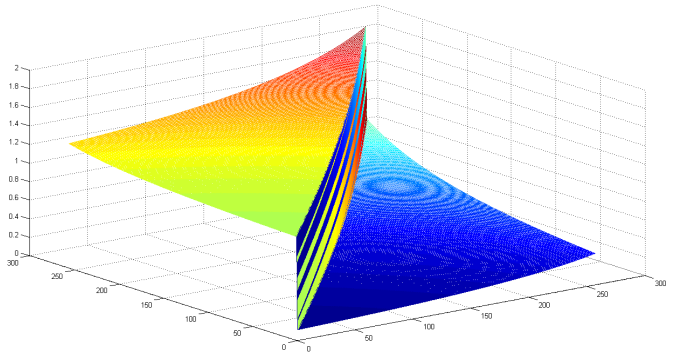

Figura 1.17: Representación de los datos originales.

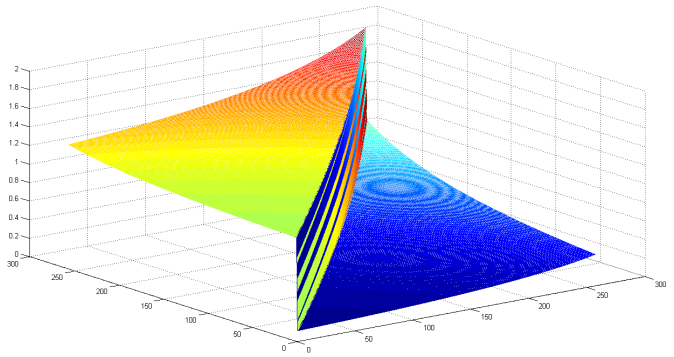

Figura 1.18: Representación de los datos reconstruidos.

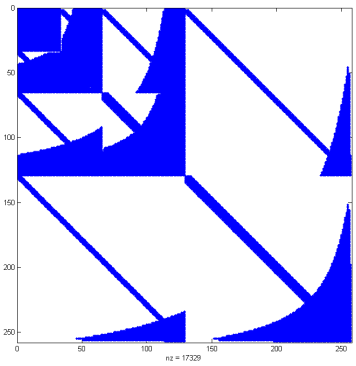

Figura 1.19: Representación de los detalles.

Esquemas de Multirresolución en 2D con Diferentes Filtros

Tipo de Función: función hipérbola 2D con discontinuidad una recta
Nombre de la función: fun_hiperbolaDisRecta2D
Discontinuidades: x1-x2
Niveles de Multirresolución: 3
Filtro: Gaussiana
Tamaño del filtro: 6
Método: sin cambio de datos
Tratamiento en la frontera: antisimétricas
Tasa de Compresión: 25
Norma Infinito del Error: 0.00024365
Norma 1 del Error: 6.2927e-005
Norma 2 del Error: 6.3287e-009
Tiempo de cálculo: 11.039 segundos
Tiempo de CPU: 11.0137 segundos
Datos guardados en el fichero datos.txt

Figura 1.20: Representación del resumen de resultados obtenidos.

Capítulo 6

Experimentos Numéricos.

En este capítulo vamos a aplicar los algoritmos estudiados a algunas funciones, tanto en $1D$ como en $2D$.

6.1. Desarrollo de los experimentos en $1D$.

Para el caso de las funciones en 1D, vamos a realizar el siguiente experimento. Partimos de unos datos originales f^L que provienen de la discretización por valores puntuales en un mallado uniforme en el intervalo $[-1, 1]$ de una cierta función f. En nuestro caso tomaremos 257 puntos en el mallado. Aplicamos los algoritmos de multirresolución con los diferentes filtros (primero con tamaño 4 y después con tamaño 6) fijando el número de escalas a 3, utilizando en la frontera máscaras laterales con los punto disponibles y variando el porcentaje de detalles guardados de 50 a 100. Para el cálculo de los errores cometidos entre la secuencia de datos original f^L (L representa el nivel de resolución) y la reconstruida $\tilde{f}^L$ hemos utilizado las normas

$$||f^L - \tilde{f}^L||_\infty = \max_i\{|f_i^L - \tilde{f}_i^L|\}, \tag{6.1}$$

$$||f^L - \tilde{f}^L||_2^2 = \sum_i (f_i^L - \tilde{f}_i^L)^2)/n, \tag{6.2}$$

donde n es el número de elementos de la secuencia. Es importante también resaltar que la norma infinito nos permite detectar errores locales grandes, mientras que la norma 2 es una medida del error más global.

Los resultados los analizamos entonces por medio de gráficas en las cuales se ha representado el tanto por ciento de detalles guardados versus el error cometido para cada una de las normas.

6.1.1. $1D$ Reconstrucción sin cambio de datos.

Vamos a trabajar con las siguientes funciones definidas en el intervalo $[-1, 1]$:

- Función de tipo Bisquare $\rightarrow$

$$f_b(x) = (1 - (x/2)^2)^2, \tag{6.3}$$

- Función de tipo Exponencial $\rightarrow$

$$f_e(x) = exp(-|x|/2), \tag{6.4}$$

- Función de tipo Gaussiana $\rightarrow$

$$f_g(x) = (exp - (|x|/2)^2 - exp(-1))/(1 - exp(-1)), \tag{6.5}$$

- Función de tipo Hipérbolica $\rightarrow$

$$f_h(x) = \frac{1}{x - 2}, \tag{6.6}$$

- Función de tipo Spline Cuártico $\rightarrow$

$$f_{s4}(x) = 1 - 6(|x|/2)^2 + 8(|x|/2)^3 - 3(|x|/2)^4, \tag{6.7}$$

- Función de tipo Spline Cúbico $\rightarrow$

$$f_{s3}(x) = \begin{cases} \frac{2}{3} - 4|x|^2 + 4|x|^3, & si \;\; 0 \leq |x| < 0{,}5 \\ \frac{4}{3} - 4|x| + 4|x|^2 - \frac{4}{3}|x|^3, & si \;\; 0{,}5 < |x| \leq 1 \\ 0 & \text{en otro caso.} \end{cases} \tag{6.8}$$

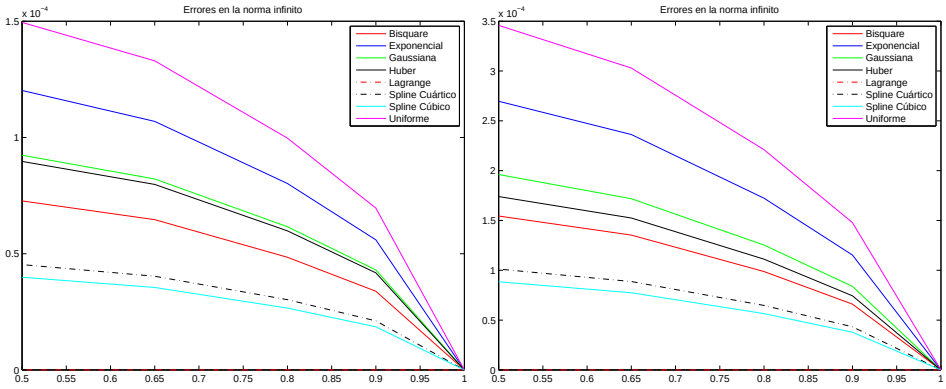

Figura 6.1: Función Bisquare: Representación gráfica del tanto por ciento de detalles guardados versus la norma infinito para diferentes tipos de filtros. Izquierda: tamaño 4, derecha: tamaño 6.

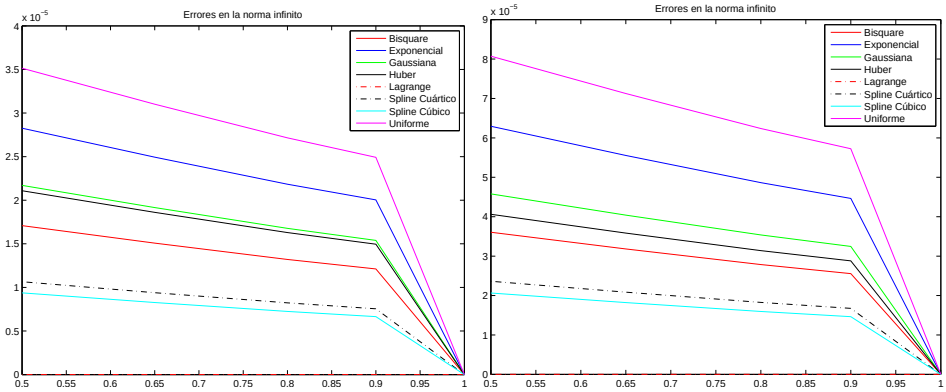

Figura 6.2: Función Exponencial: Representación gráfica del tanto por ciento de detalles guardados versus la norma infinito para diferentes tipos de filtros. Izquierda: tamaño 4, derecha: tamaño 6.

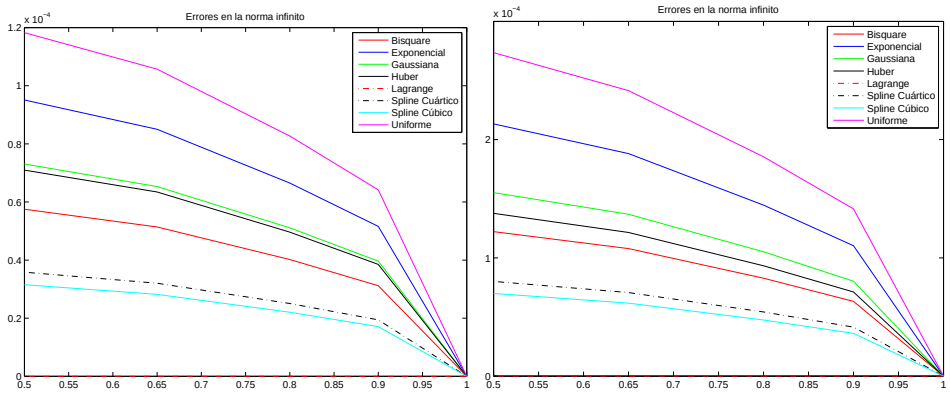

Figura 6.3: Función Gaussiana: Representación gráfica del tanto por ciento de detalles guardados versus la norma infinito para diferentes tipos de filtros. Izquierda: tamaño 4, derecha: tamaño 6.

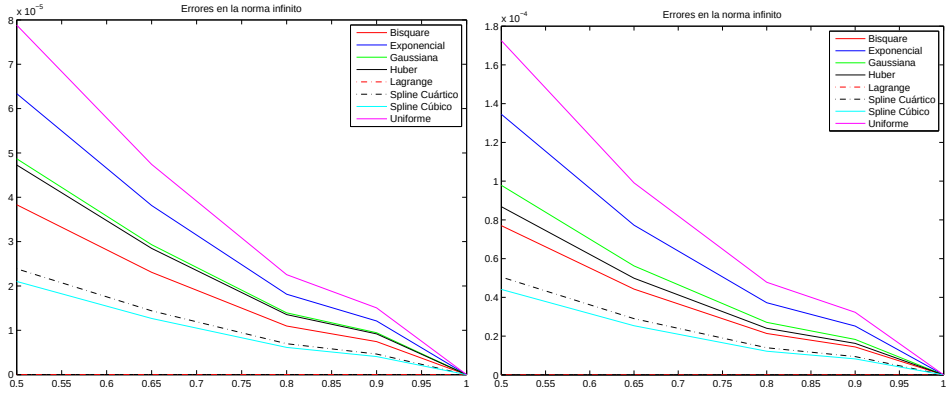

Figura 6.4: Función Hipérbola: Representación gráfica del tanto por ciento de detalles guardados versus la norma infinito para diferentes tipos de filtros. Izquierda: tamaño 4, derecha: tamaño 6.

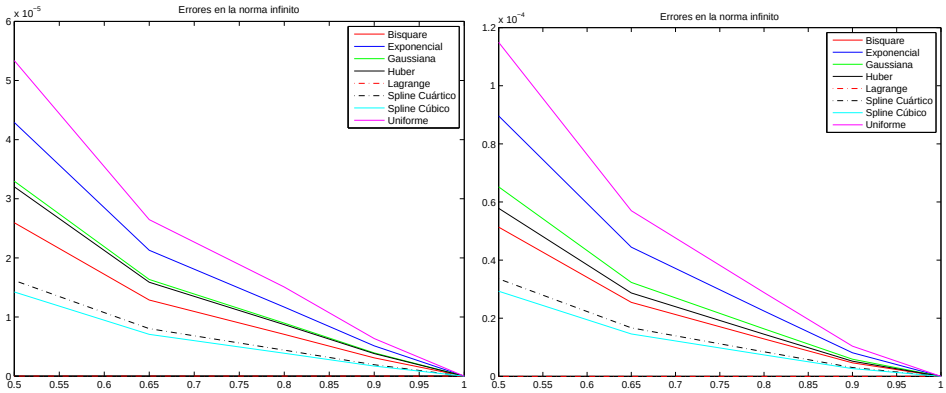

Figura 6.5: Función Spline Cuártico: Representación gráfica del tanto por ciento de detalles guardados versus la norma infinito para diferentes tipos de filtros. Izquierda: tamaño 4, derecha: tamaño 6.

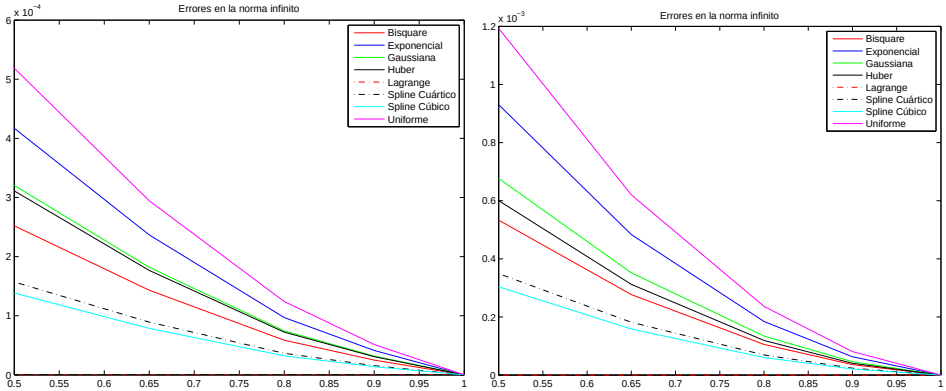

Figura 6.6: Función Spline cúbico: Representación gráfica del tanto por ciento de detalles guardados versus la norma infinito para diferentes tipos de filtros. Izquierda: tamaño 4, derecha: tamaño 6.

Tanto en el caso de trabajar con filtros de tamaño 4 o de tamaño 6, observamos que los filtros de recontrucción que cometen menor error, en norma infinito como en norma 2, para cualquiera de las funciones consideradas son el filtro de Lagrange, el Spline cúbico y el Spline cúartico en ese orden. En vista de estos resultados obtenidos, podemos confirmar que las reconstrucciones que aproximan mejor las funciones consideradas dentro de los algoritmos de multirresolución son las funciones polinomiales. Una posible explicación a este hecho sería que localmente las funciones pueden aproximarse bien por polinomios. También es conocido que los polinomios son densos dentro del espacio de las funciones continuas en un intervalo cerrado y acotado.

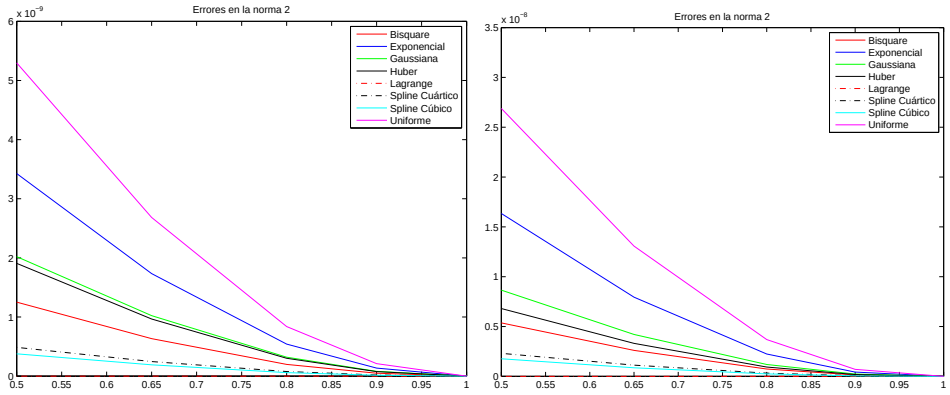

Figura 6.7: Función Bisquare: Representación gráfica del tanto por ciento de detalles guardados versus la norma 2 al cuadrado para diferentes tipos de filtros. Izquierda: tamaño 4, derecha: tamaño 6.

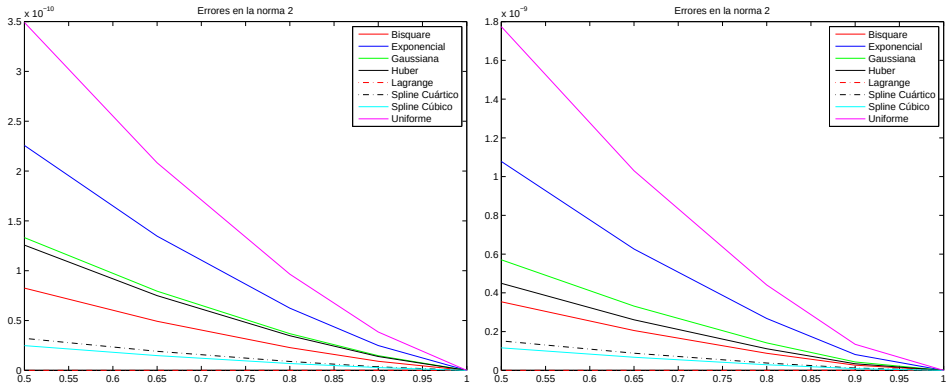

Figura 6.8: Función Exponencial: Representación gráfica del tanto por ciento de detalles guardados versus la norma 2 al cuadrado para diferentes tipos de filtros. Izquierda: tamaño 4, derecha: tamaño 6.

En las gráficas 6.1 y 6.7 vemos los resultados obtenidos al trabajar con la función $f_b(x)$ de tipo *Bisquare* dada en la fórmula (6.3). Como puede observarse los filtros que mejor funcionan en este caso, dando un error menor, podrían ordenarse de la siguiente forma: Lagrange, Spline Cúbico, Spline Cuártico, Bisquare, Huber, Gaussiana, Exponencial, Uniforme. Analizando los resultados obtenidos para las demás gráficas, es decir, gráficas 6.2 y 6.8 para la función $f_e(x)$ de tipo *Exponencial* dada en la fórmula (6.4), gráficas 6.3 y 6.9 para la función $f_g(x)$ de tipo *Gaussiana* dada en la fórmula (6.5), gráficas 6.4 y 6.10 para la función $f_h(x)$ de tipo *Hipérbola* dada en la fórmula (6.6), gráficas 6.5 y 6.11 para la función $f_{s4}(x)$ de tipo *Spline Cuártico* dada

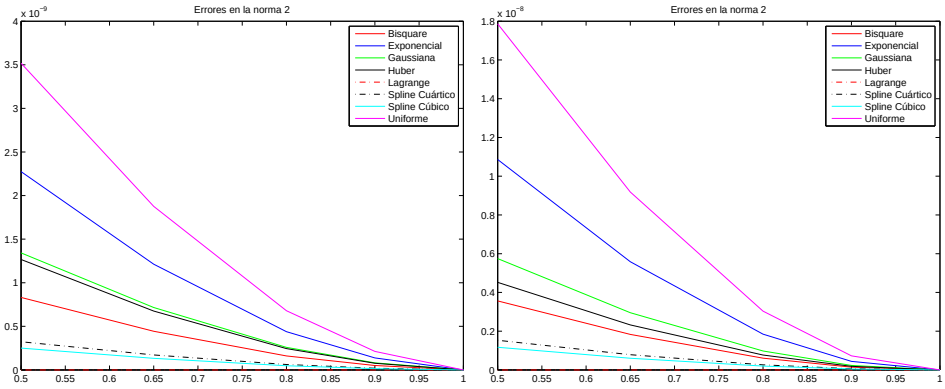

Figura 6.9: Función Gaussiana: Representación gráfica del tanto por ciento de detalles guardados versus la norma 2 al cuadrado para diferentes tipos de filtros. Izquierda: tamaño 4, derecha: tamaño 6.

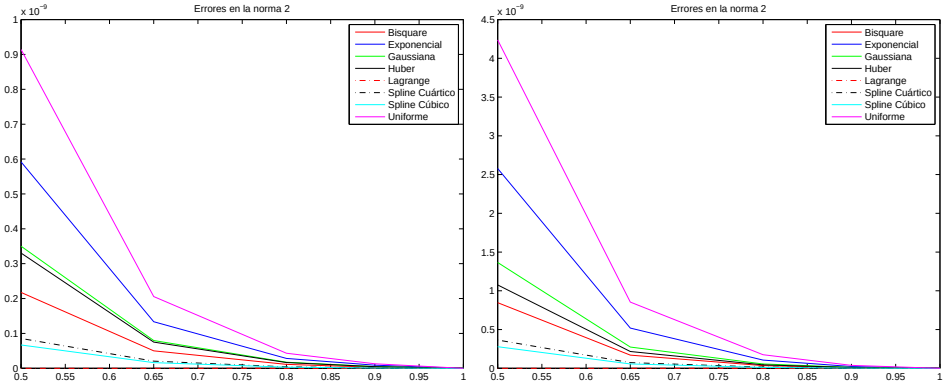

Figura 6.10: Función Hipérbola: Representación gráfica del tanto por ciento de detalles guardados versus la norma 2 al cuadrado para diferentes tipos de filtros. Izquierda: tamaño 4, derecha: tamaño 6.

en la fórmula (6.7) y las gráficas 6.6 y 6.12 para la función $f_{s3}(x)$ de tipo *Spline Cúbico* dada en la fórmula (6.8), vemos que se sigue cumpliendo el orden mencionado entre los filtros que mejores resultados obtienen. Es decir, que de menor error a mayor los filtros se ordenan: Lagrange, Spline Cúbico, Spline Cuártico, Bisquare, Huber, Gaussiana, Exponencial, Uniforme.
Es también reseñable y poco intuitivo que el filtro que proviene de la función de ponderación más parecida a la función de la que se tienen los datos no es en la mayoría de los casos el que mejor funciona, y ni siquiera lo es después de los filtros polinomiales.

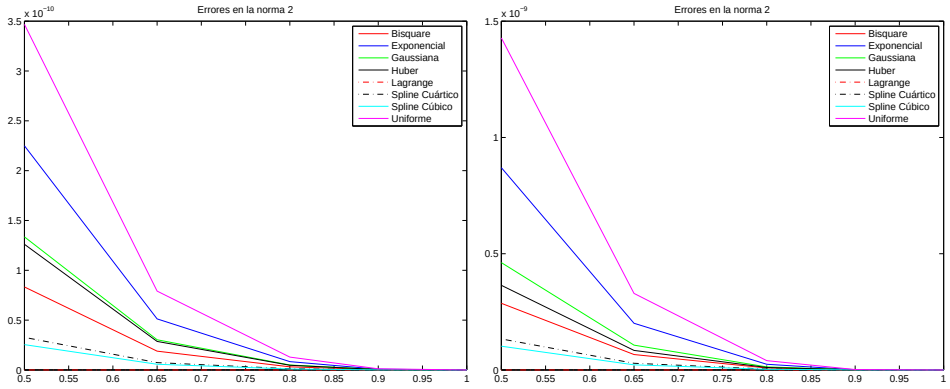

Figura 6.11: Función Spline Cuártico: Representación gráfica del tanto por ciento de detalles guardados versus la norma 2 al cuadrado para diferentes tipos de filtros. Izquierda: tamaño 4, derecha: tamaño 6.

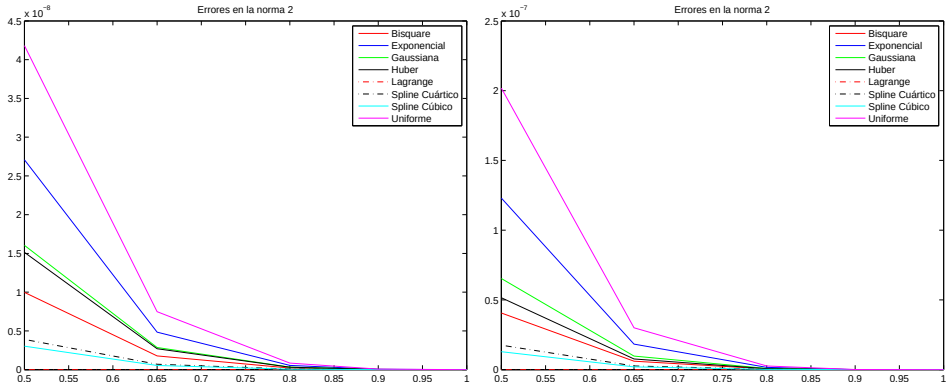

Figura 6.12: Función Spline cúbico: Representación gráfica del tanto por ciento de detalles guardados versus la norma 2 al cuadrado para diferentes tipos de filtros. Izquierda: tamaño 4, derecha: tamaño 6.

6.1.2. $1D$ Reconstrucción con cambio de datos: PPH.

En esta sección vamos a comparar el algoritmo de multirresolución sin cambio de datos y con cambio de datos *PPH*. Esta comparación la hemos llevado a cabo con respecto a la norma infinito dada en la fórmula (6.13) y la norma 2 dada en la fórmula (6.14), fijando el tamaño de los filtos a 4 puntos. Las funciones con discontinuidad consideradas, definidas en el intervalo $[-1, 1]$ son:

- Función de tipo Sinusoidal con una esquina $\rightarrow$

$$f_{cs}(x) = |100 \sin(\pi(x - dis))|, \tag{6.9}$$

donde $-1 < dis < 1$ es la posición de la discontinuidad de la función f_{cs} .

- Función de tipo Sinusoidal con un salto →

$$f_{js}(x) = \begin{cases} \sin(2\pi x), & si \;\; x \leq dis \\ \\ -10 - \sin(2\pi x), & si \;\; x > dis \end{cases} \tag{6.10}$$

donde $-1 < dis < 1$ es la posición de la discontinuidad de la función f_{js} .

- Función de tipo Sinusoidal con un salto y una esquina →

$$f_{jcs}(x) = \begin{cases} \sin(2\pi x), & si \;\; x \leq dis_1 \\ \\ -50 + \sin(2\pi x), & si \;\; dis_1 \leq x < dis_2 \\ -50 + \sin(2\pi dis_2) - 10\sin(2\pi(x - dis_2))), & si \;\; x > dis_2 \end{cases} \tag{6.11}$$

donde $-1 < dis_1 < dis_2 < 1$ indican la posición de la discontinuidad de salto de la función f_{jcs} y de su derivada respectivamente.

- Función de tipo Sinusoidal con dos saltos →

$$f_{tjs}(x) = \begin{cases} \sin(2\pi x), & si \;\; x \leq dis_1 \\ \\ -10 - \sin(2\pi x), & si \;\; dis_1 \leq x < dis_2 \\ 10 + \sin(2\pi x), & si \;\; x > dis_2 \end{cases} \tag{6.12}$$

donde $-1 < dis_1 < dis_2 < 1$ indican la posición de las discontinuidades de salto de la función f_{tjs}.

Como vímos en el apartado anterior, se puede asegurar que se mantiene el orden entre los filtros que proporcionan mejores resultados tanto para la reconstrucción con Lagrange de 4 puntos variando el porcentaje de detalles guardados de 50 a 100, como para la reconstrucción PPH. Para todas las gráficas, es decir, gráficas 6.13 y 6.17 para la función $f_{cs}(x)$ de tipo *Sinusoidal con una esquina* dada en la fórmula (6.9), gráficas 6.14 y 6.18 para la función $f_{js}(x)$ de tipo *Sinusoidal con un salto* dada en la fórmula (6.10), gráficas 6.15

y 6.19 para la función $f_{jcs}(x)$ de tipo *Sinusoidal con un salto y una esquina* dada en la fórmula (6.11), gráficas 6.16 y 6.20 para la función $f_{tjs}(x)$ de tipo *Sinusoidal con dos saltos* dada en la fórmula (6.12), observamos que los filtros de menor error a mayor se ordenan: Lagrange, Spline Cúbico, Spline Cuártico, Bisquare, Huber, Gaussiana, Exponencial, Uniforme. Entonces, se puede apreciar que las reconstrucciones que aproximan mejor las funciones que presentan una discontinuidad son las funciones polinomiales.

Otro punto destacado de este experimento es que debido a la presencia de una discontinuidad utilizando el método no lineal PPH se reduce el error cometido tanto para la norma infinito como para la norma 2, aunque principalmente se reduce para la norma 2.

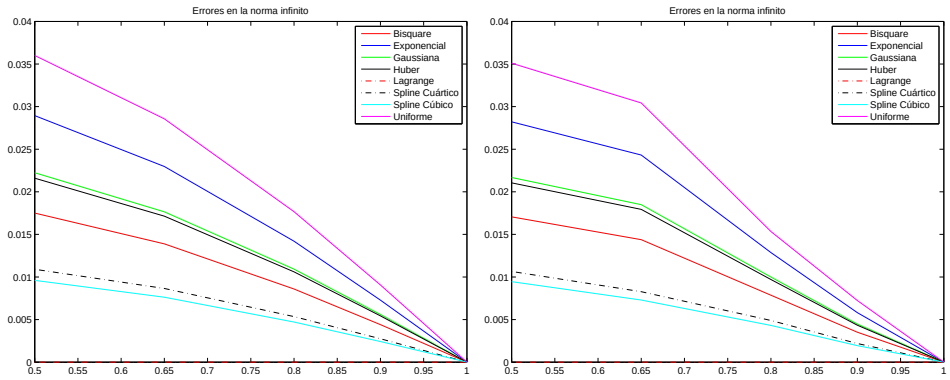

Figura 6.13: Función Sinusoidal con una esquina: Representación gráfica del tanto por ciento de detalles guardados versus la norma infinito para diferentes tipos de filtros. Izquierda: sin cambio, derecha: con cambio PPH.

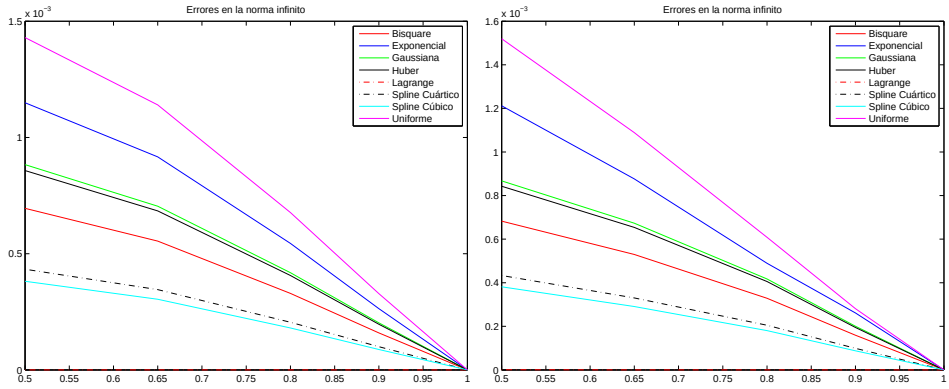

Figura 6.14: Función Sinusoidal con un salto: Representación gráfica del tanto por ciento de detalles guardados versus la norma infinito para diferentes tipos de filtros. Izquierda: sin cambio, derecha: con cambio PPH.

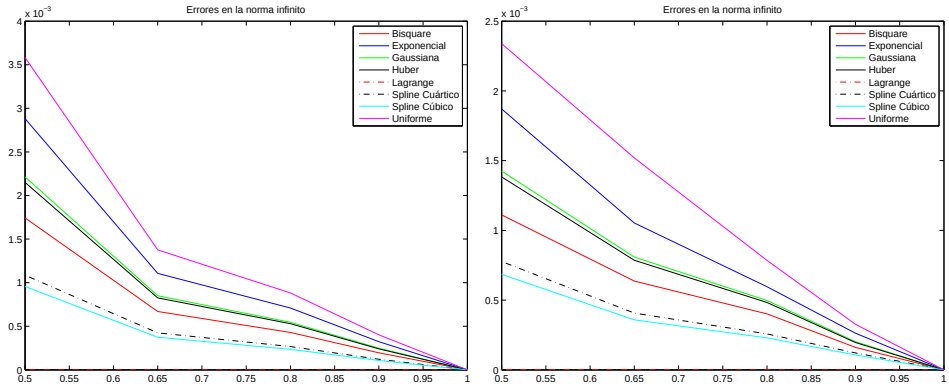

Figura 6.15: Función Sinusoidal con un salto y una esquina: Representación gráfica del tanto por ciento de detalles guardados versus la norma infinito para diferentes tipos de filtros. Izquierda: sin cambio, derecha: con cambio PPH.

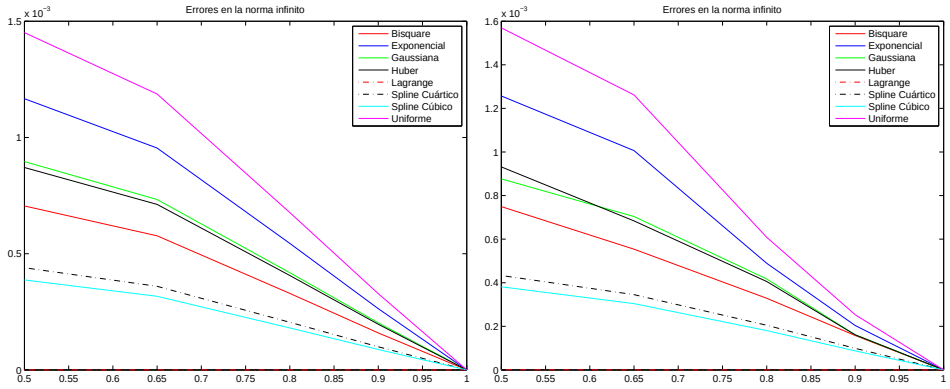

Figura 6.16: Función Sinusoidal con dos saltos: Representación gráfica del tanto por ciento de detalles guardados versus la norma infinito para diferentes tipos de filtros. Izquierda: sin cambio, derecha: con cambio PPH.

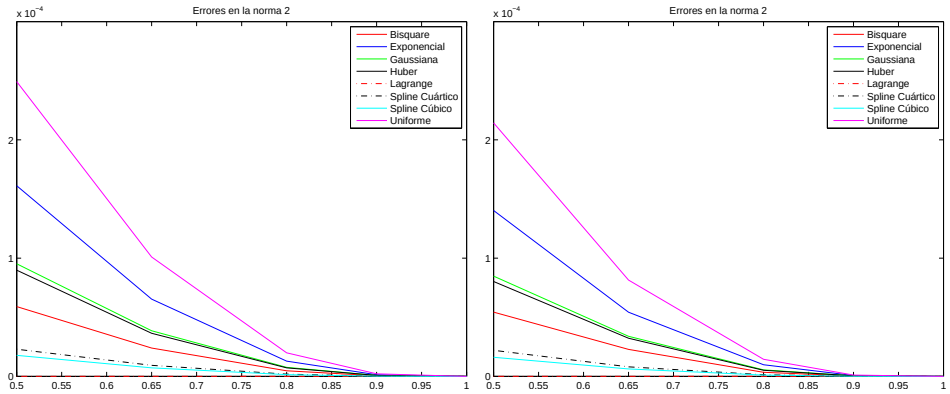

Figura 6.17: Función Sinusoidal con una esquina: Representación gráfica del tanto por ciento de detalles guardados versus la norma 2 al cuadrado para diferentes tipos de filtros. Izquierda: sin cambio, derecha: con cambio PPH.

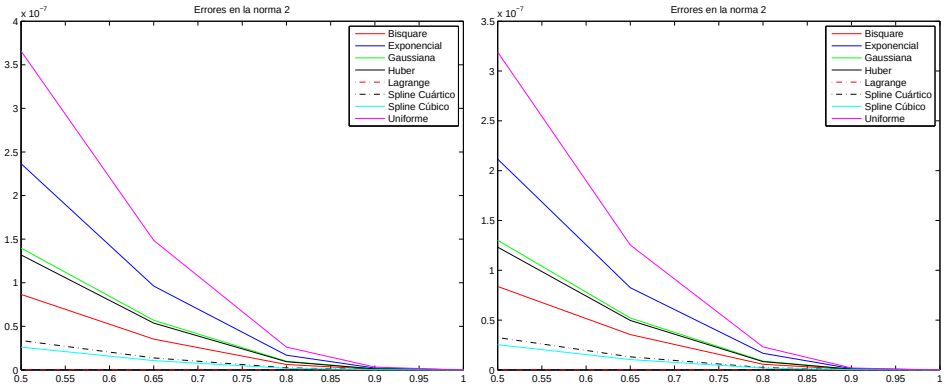

Figura 6.18: Función Sinusoidal con un salto: Representación gráfica del tanto por ciento de detalles guardados versus la norma 2 al cuadrado para diferentes tipos de filtros. Izquierda: sin cambio, derecha: con cambio PPH.

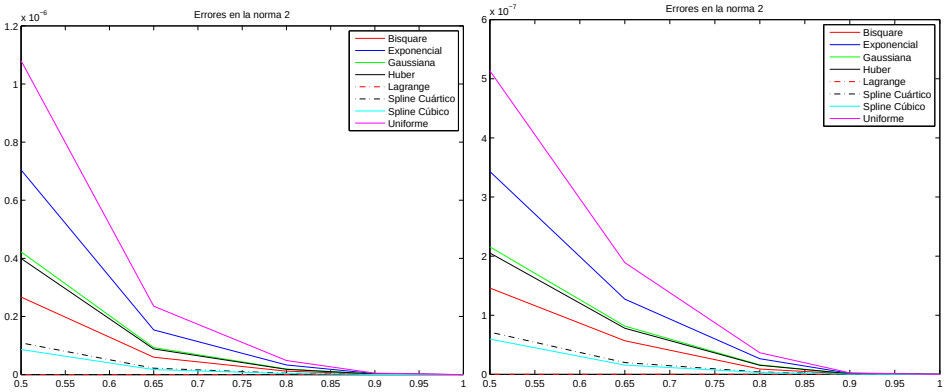

Figura 6.19: Función Sinusoidal con un salto y una esquina: Representación gráfica del tanto por ciento de detalles guardados versus la norma 2 al cuadrado para diferentes tipos de filtros. Izquierda: sin cambio, derecha: con cambio PPH.

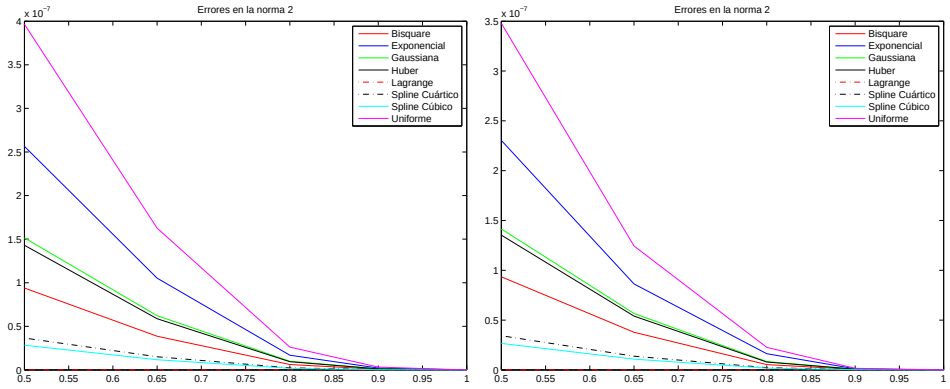

Figura 6.20: Función Sinusoidal con dos saltos: Representación gráfica del tanto por ciento de detalles guardados versus la norma 2 al cuadrado para diferentes tipos de filtros. Izquierda: sin cambio, derecha: con cambio PPH.

Para mostrar mejor el comportamiento de los filtros lineales y no lineales presentamos a continuación las reconstruccines obtenidas con una de las funciones en concreto y con el filtro de Lagrange de 4 puntos con y sin cambio previo de los datos. Vamos a centrarnos en la función sinusoidal con salto $f_{js}(x)$ que aparece en la expresión (6.10). En la figura 6.21 vemos la función original, la reconstrucción conseguida con Lagrange (4 puntos) sin cambio guardando 4 detalles y la reconstrucción también con Lagrange de 4 puntos guardando 4 detalles, pero esta vez con cambio previo de datos. Como puede observarse la discontinuidad está mucho mejor definida en el caso de realizar un tratamiento a los datos antes de aplicar el filtro localmente.

Lo que sucede con el filtro de Lagrange, sucede también con los demás filtros de reconstrucción. Como ejemplo se muestra en la figura (6.22) el comportamiento del filtro Exponenciel de tamaño 4 guardando 4 detalles.

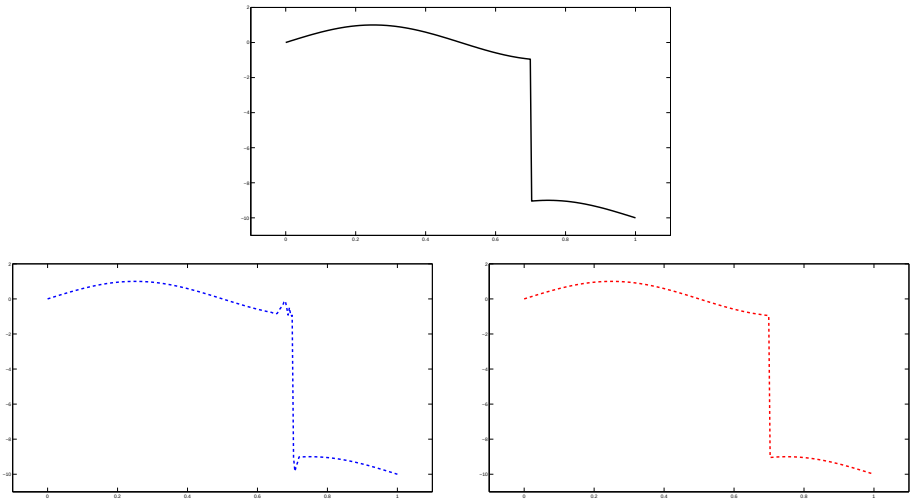

Figura 6.21: Parte superior: Función Original $f_{js}(x)$ definida en (6.10) con un salto, parte inferior izquierda: Aproximación obtenida por Lagrange con 4 puntos sin cambio guardando 4 detalles, parte inferior derecha: Lagrange con 4 puntos con cambio guardando 4 detalles. $L = 3$ niveles de multirresolución.

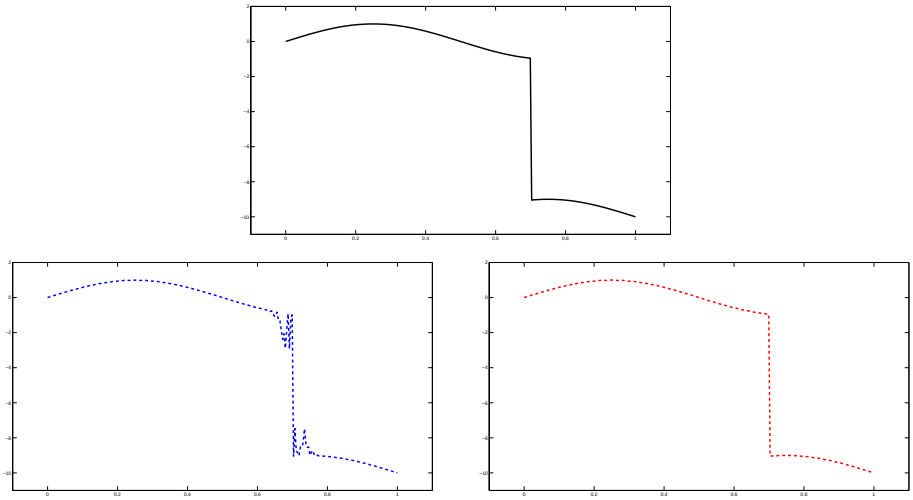

Figura 6.22: Parte superior: Función Original $f_{js}(x)$ definida en (6.10) con un salto, parte inferior izquierda: Aproximación obtenida por el filtro Exponencial con 4 puntos sin cambio guardando 4 detalles, parte inferior derecha: Filtro Exponencial con 4 puntos con cambio guardando 4 detalles. $L = 3$ niveles de multirresolución.

6.2. Desarrollo de los experimentos en $2D$.

Igualmente a como hemos procedido con el caso de $1D$, vamos a aplicar en $2D$ los algoritmos de multirresolución con los diferentes filtros (tamaño 4)

fijando el número de escalas a 3, utilizando en la frontera máscaras laterales con los puntos disponibles y variando el porcentaje de detalles guardados de 50 a 100. Para el cálculo de los errores cometidos entre la secuencia de datos original f^L (L representa el nivel de resolución), discretizada por valores puntuales en un mallado uniforme de 257×257 puntos en el rectángulo $[-1,1][-1,1]$, y la reconstruida $\tilde{f}^L$ hemos utilizado las normas

$$||f^L - \tilde{f}^L||_\infty = \max_{ij}\{|f^L_{ij} - \tilde{f}^L_{ij}|\}, \tag{6.13}$$

$$||f^L - \tilde{f}^L||_2^2 = \sum_{ij} \frac{(f^L_{ij} - \tilde{f}^L_{ij})^2}{nm}, \tag{6.14}$$

donde $n \times m$ es el tamaño de la secuencia de datos.

Los resultados los analizamos, de la misma manera que en el apartado anterior, por medio de gráficas en las cuales se ha representado el tanto por ciento de detalles guardados versus el error cometido para cada una de las normas (primero norma infinito y luego norma 2).

6.2.1. $2D$ Reconstrucción sin cambio de datos.

Vamos a estudiar el caso de las siguientes funciones definidas en el intervalo $[-1,1] \times [-1,1]$:

- Función de tipo Bisquare en $2D \rightarrow$

$$f_{b2}(x,y) = f_b(x).f_b(y), \tag{6.15}$$

  donde $f_b(x) = (1-(x/2)^2)^2$ y $f_b(y) = (1-(y/2)^2)^2$.

- Función de tipo Exponencial $2D \rightarrow$

$$f_{e2}(x,y) = f_e(x).f_e(y), \tag{6.16}$$

  donde $f_e(x) = exp(-|x|/2)$ y $f_e(y) = exp(-|y|/2)$.

- Función de tipo Gaussiana $2D \rightarrow$

$$f_{g2}(x,y) = f_g(x).f_g(y), \tag{6.17}$$

  donde $f_g(x) = (exp - (|x|/2)^2 - exp(-1))/(1 - exp(-1))$ y $f_g(y) = (exp - (|y|/2)^2 - exp(-1))/(1 - exp(-1))$.

- Función de tipo Hipérbolica $2D \rightarrow$

$$f_{h2}(x,y) = f_h(x).f_h(y), \tag{6.18}$$

donde $f_h(x) = \frac{1}{x-2}$ y $f_h(y) = \frac{1}{y-2}$.

A la vista de los resultados obtenidos en $2D$, podemos asegurar que al igual que en el caso de $1D$ los filtros de tamaño 4 presentan el mismo comportamiento.Es decir que en la gráficas, gráficas 6.23 para la función $f_{b2}(x,y)$ de tipo *Bisquare* $2D$ dada en la fórmula (6.15), gráficas 6.24 para la función $f_{e2}(x,y)$ de tipo *Exponencial* $2D$ dada en la fórmula (6.16), gráficas 6.25 para la función $f_{g2}(x,y)$ de tipo *Gaussiana* $2D$ dada en la fórmula (6.17), gráficas 6.26 para la función $f_{h2}(x,y)$ de tipo *Hipérbola* $2D$ dada en la fórmula (6.18), los filtros tienen el mismo orden mencionado en la sección anterior (de menor error a mayor): Lagrange, Spline Cúbico, Spline Cuártico, Bisquare, Huber, Gaussiana, Exponencial, Uniforme.

Podemos realizar observaciones semejantes a los experimentos anteriores, las reconstrucciones que aproximan mejor las funciones consideradas dentro de los algoritmos de multirresolución son las funciones polinomiales.

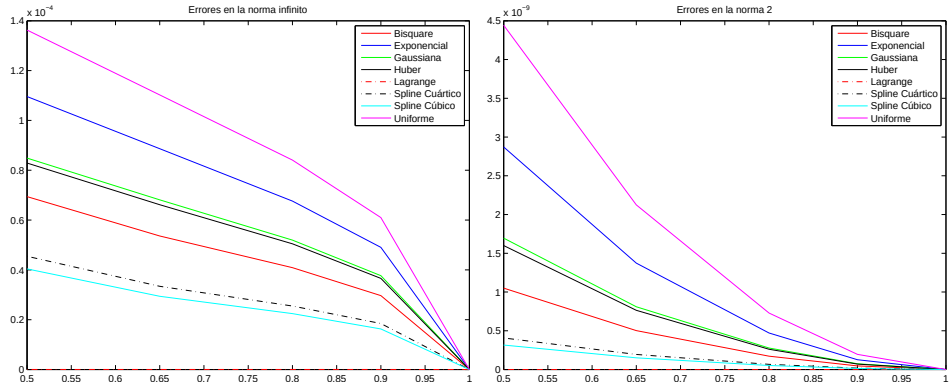

Figura 6.23: Función Bisquare 2D: Representación gráfica del tanto por ciento de detalles guardados versus el error cometido para diferentes tipos de filtros de tamaño 4. Izquierda: Norma infinito, derecha: Norma 2 al cuadrado.

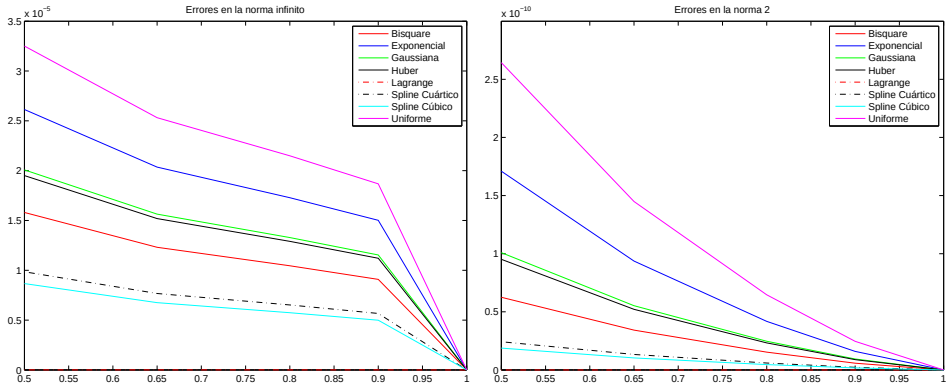

Figura 6.24: Función Exponencial 2D: Representación gráfica del tanto por ciento de detalles guardados versus el error cometido para diferentes tipos de filtros de tamaño 4. Izquierda: Norma infinito, derecha: Norma 2 al cuadrado.

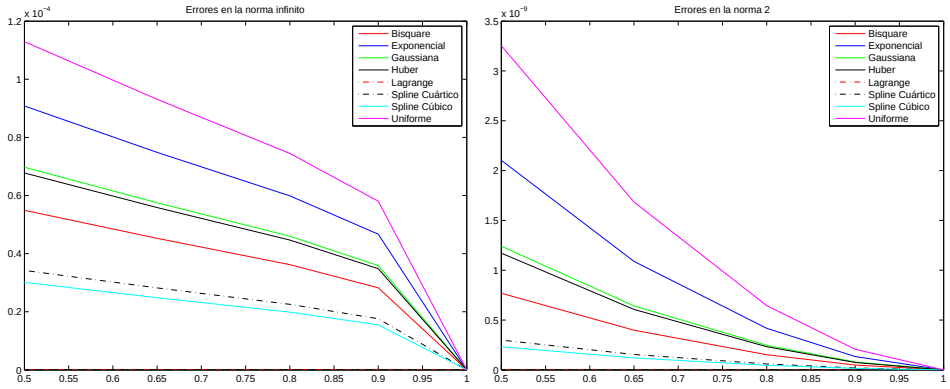

Figura 6.25: Función Gaussiana 2D: Representación gráfica del tanto por ciento de detalles guardados versus el error cometido para diferentes tipos de filtros de tamaño 4. Izquierda: Norma infinito, derecha: Norma 2 al cuadrado.

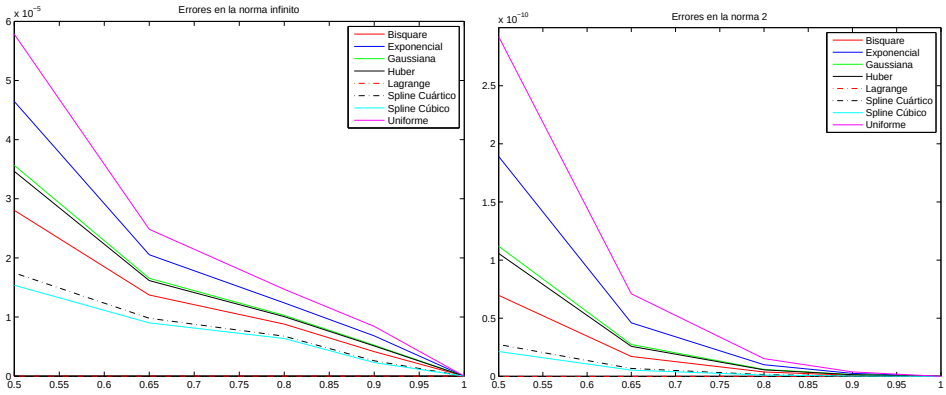

Figura 6.26: Función Hipérbola 2D: Representación gráfica del tanto por ciento de detalles guardados versus el error cometido para diferentes tipos de filtros de tamaño 4. Izquierda: Norma infinito, derecha: Norma 2 al cuadrado.

6.2.2. $2D$ Reconstrucción con cambio de datos: PPH.

A continuación, al igual que en el experimento realizado para $1D$, vamos a hacer una comparativa de la reconstrucción obtenida empleando algoritmos con y sin cambio de datos por el método de recontrucción PPH, variando el porcentaje de detalles guardados de 50 a 100 y fijando el tamaño de los filtros a 4 puntos y el número de escalas a 3.

Para este caso, donde las funciones presentan una discontinuidad, consideramos estas 3 discontinuidades :

1. Discontinuidad dada por la recta de ecuación : $x = y$.
2. Discontinuidad dada por la circunferencia de ecuación : $(x - 0{,}1)^2 + y^2 = 0{,}4$.
3. Discontinuidad dada por la parábola de ecuación : $y - x^2 = 0$.

Vamos a centrarnos en las siguientes funciones definidas en el intervalo $[-1, 1]x[-1, 1]$:

- Función de tipo Bisquare con discontinuidad en $2D \rightarrow$

$$f_{b2dis}(x, y) = \begin{cases} f_b(x).f_b(y), & si \quad dis \geq 0 \\ \\ f_b(x).f_b(y) + 1, & si \quad dis < 0. \end{cases} \tag{6.19}$$

donde $f_b(x) = (1-(x/2)^2)^2$ y $f_b(y) = (1-(y/2)^2)^2$, y dis viene definida con la expresión $dis = x - y$ para el caso de la discontinuidad dada por una recta,con la expresión $dis = (x - 0{,}1)^2 + y^2 - 0{,}4$ para el caso de la discontinuidad dada por una circunferencia y finalmente con la expresión $dis = y - x^2$ para el caso de la discontinuidad dada por una parábola.

- Función de tipo Exponencial con discontinuidad en $2D \rightarrow$

$$f_{e2dis}(x,y) = \begin{cases} f_e(x).f_e(y), & si \;\; dis \geq 0 \\ \\ f_e(x).f_e(y) + 1, & si \;\; dis < 0. \end{cases} \tag{6.20}$$

donde $f_e(x) = exp(-|x|/2)$ y $f_e(y) = exp(-|y|/2)$, y dis viene definida con la expresión $dis = x - y$ para el caso de la discontinuidad dada por una recta,con la expresión $dis = (x - 0{,}1)^2 + y^2 - 0{,}4$ para el caso de la discontinuidad dada por una circunferencia y finalmente con la expresión $dis = y - x^2$ para el caso de la discontinuidad dada por una parábola.

- Función de tipo Gaussiana con discontinuidad en $2D \rightarrow$

$$f_{g2dis}(x,y) = \begin{cases} f_g(x).f_g(y), & si \;\; dis \geq 0 \\ \\ f_g(x).f_g(y) + 1, & si \;\; dis < 0. \end{cases} \tag{6.21}$$

donde $f_g(x) = (exp - (|x|/2)^2 - exp(-1))/(1 - exp(-1))$ // y $f_g(y) = (exp - (|y|/2)^2 - exp(-1))/(1 - exp(-1))$, y dis viene definida con la expresión $dis = x - y$ para el caso de la discontinuidad dada por una recta,con la expresión $dis = (x - 0{,}1)^2 + y^2 - 0{,}4$ para el caso de la discontinuidad dada por una circunferencia y finalmente con la expresión $dis = y - x^2$ para el caso de la discontinuidad dada por una parábola.

- Función de tipo Hipérbolica con discontinuidad en $2D \rightarrow$

$$f_{h2dis}(x,y) = \begin{cases} f_h(x).f_h(y), & si \;\; dis \geq 0 \\ \\ f_h(x).f_h(y) + 1, & si \;\; dis < 0. \end{cases} \tag{6.22}$$

donde $f_h(x) = \frac{1}{x-2}$ y $f_h(y) = \frac{1}{y-2}$, y dis viene definida con la expresión $dis = x - y$ para el caso de la discontinuidad dada por una recta,con la expresión $dis = (x - 0{,}1)^2 + y^2 - 0{,}4$ para el caso de la discontinuidad dada por una circunferencia y finalmente con la expresión $dis = y - x^2$ para el caso de la discontinuidad dada por una parábola.

Con respecto a la cuestión de cómo se ordenan los filtros de recontrucción se pueden hacer observaciones semejantes a los demás experimentos llevados a cabo en este proyecto. Analizando los resultados obtenidos para las gráficas realizadas, es decir, gráficas 6.27, 6.28, 6.29, 6.30, 6.31 y 6.32 para la función $f_{b2dis}(x, y)$ de tipo *Bisquare en 2D definida en la fórmula (6.20) con discontinuidad dada por una recta, una circunferencia y una parábola*, gráficas 6.33 y 6.34 para la función $f_{e2dis}(x, y)$ de tipo *Exponencial en 2D definida en la fórmula (6.20) con discontinuidad dada por una recta*, gráficas 6.35 y 6.36 para la función $f_{g2dis}(x, y)$ de tipo *Gaussiana en 2D definida en la fórmula (6.21) con discontinuidad dada por una circunferencia* y gráficas 6.37 y 6.38 para la función $f_{h2dis}(x, y)$ de tipo *Hipérbola en 2D definida en la fórmula (6.22) con discontinuidad dada por una parábola*, vemos que se sigue cumpliendo el orden mencionado entre los filtros que mejores resultados obtienen. Es decir, que de menor error a mayor los filtros se ordenan: Lagrange, Spline Cúbico, Spline Cuártico, Bisquare, Huber, Gaussiana, Exponencial, Uniforme.

Observamos similar comportamiento de los operadores de reconstrucción. Las reconstrucciones que aproximan mejor las funciones consideradas dentro de los algoritmos de multirresolución son las funciones polinomiales.

Igual que anteriormente, notamos que debido a la presencia de una discontinuidad utilizando el operador no lineal PPH se reduce el error cometido tanto para la norma infinito como para la norma 2, aunque principalmente se reduce para la norma 2.

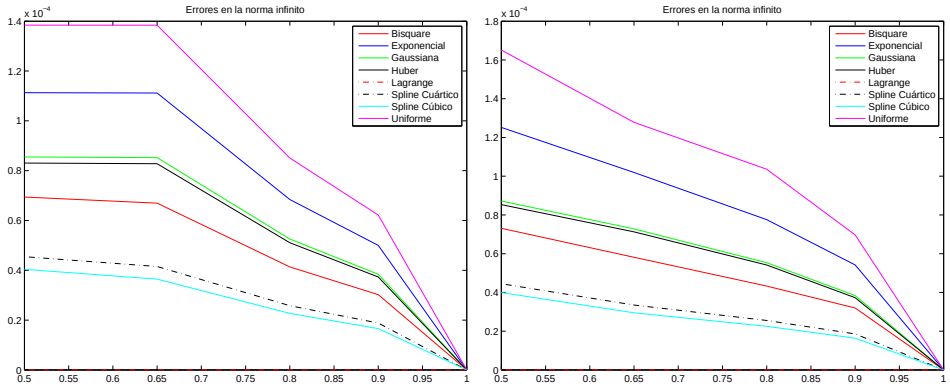

Figura 6.27: Función Bisquare en $2D$ con discontinuidad dada por una recta: Representación gráfica del tanto por ciento de detalles guardados versus la norma infinito para diferentes tipos de filtros. Izquierda: sin cambio, derecha: con cambio PPH.

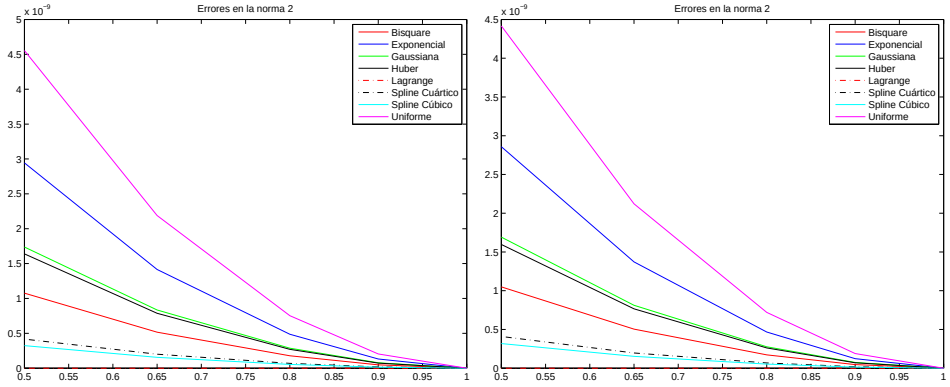

Figura 6.28: Función Bisquare en $2D$ con discontinuidad dada por una recta: Representación gráfica del tanto por ciento de detalles guardados versus la norma 2 al cuadrado para diferentes tipos de filtros. Izquierda: sin cambio, derecha: con cambio PPH.

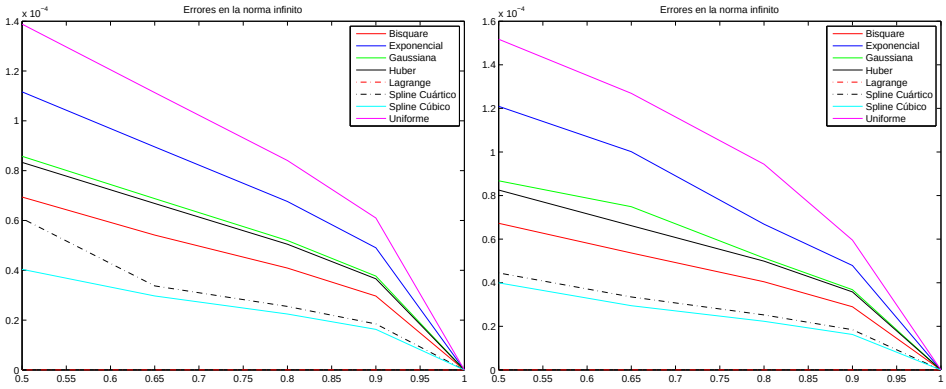

Figura 6.29: Función Bisquare en $2D$ con discontinuidad dada por una circunferencia: Representación gráfica del tanto por ciento de detalles guardados versus la norma infinito para diferentes tipos de filtros. Izquierda: sin cambio, derecha: con cambio PPH.

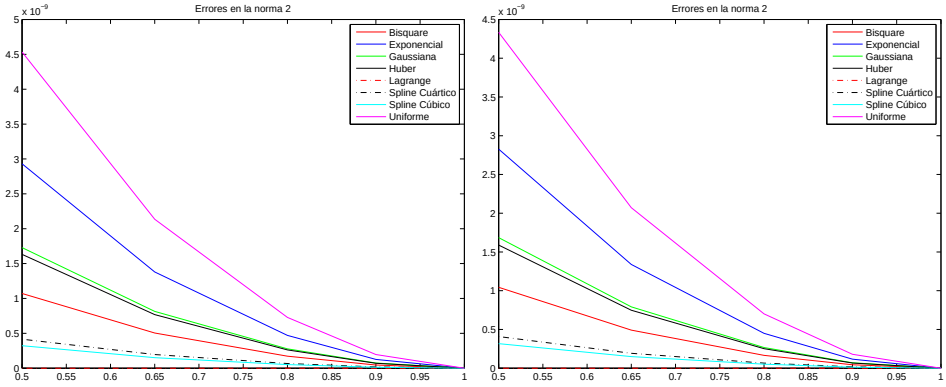

Figura 6.30: Función Bisquare en $2D$ con discontinuidad dada por una circunferencia: Representación gráfica del tanto por ciento de detalles guardados versus la norma 2 al cuadrado para diferentes tipos de filtros. Izquierda: sin cambio, derecha:con cambio PPH.

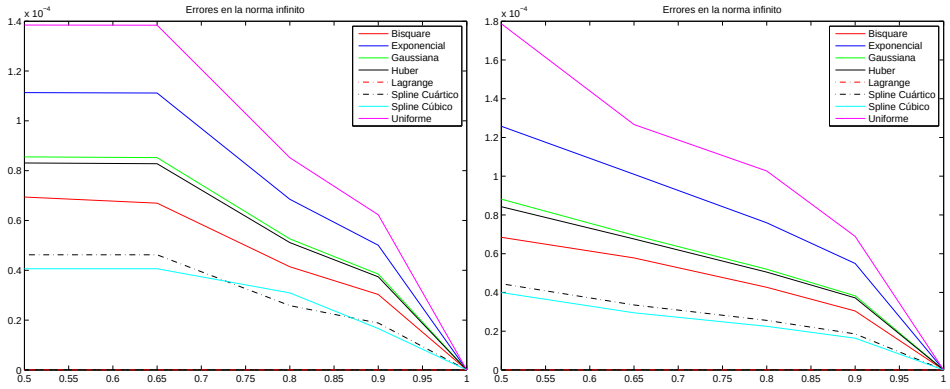

Figura 6.31: Función Bisquare en $2D$ con discontinuidad dada por una parábola: Representación gráfica del tanto por ciento de detalles guardados versus la norma infinito para diferentes tipos de filtros. Izquierda: sin cambio, derecha: con cambio PPH.

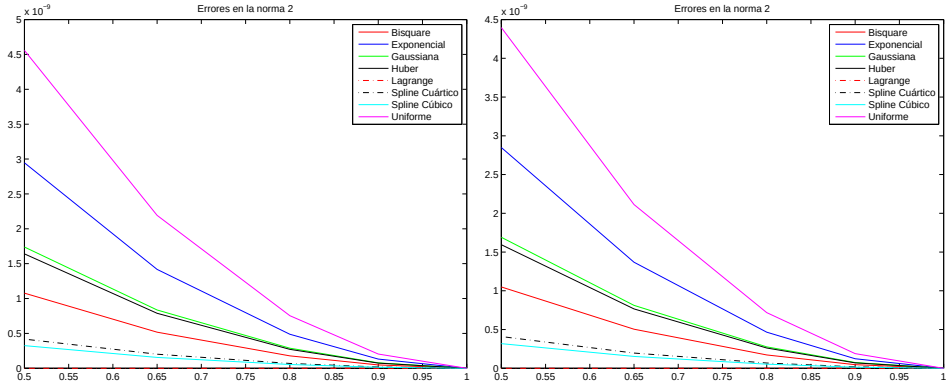

Figura 6.32: Función Bisquare en $2D$ con discontinuidad dada por una parábola: Representación gráfica del tanto por ciento de detalles guardados versus la norma 2 al cuadrado para diferentes tipos de filtros. Izquierda: sin cambio, derecha: con cambio PPH.

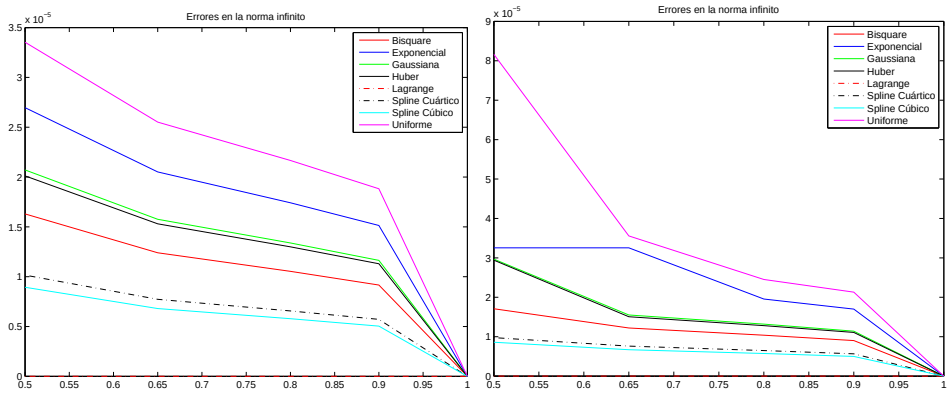

Figura 6.33: Función Exponencial en $2D$ con discontinuidad dada por una recta: Representación gráfica del tanto por ciento de detalles guardados versus la norma infinito para diferentes tipos de filtros. Izquierda: sin cambio, derecha: con cambio PPH.

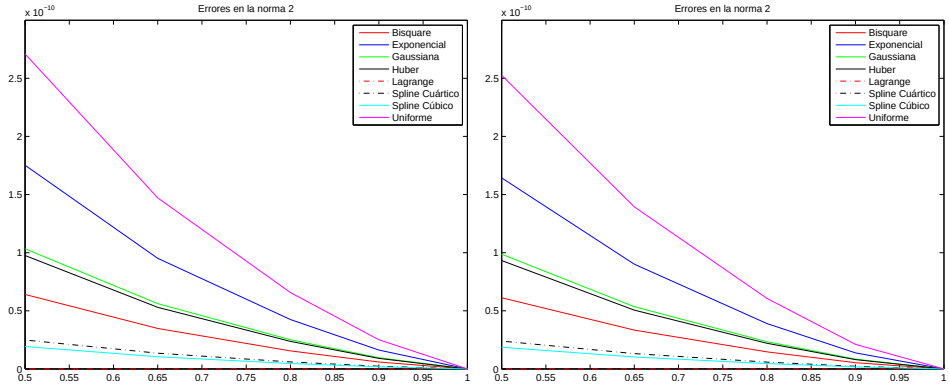

Figura 6.34: Función Exponencial en $2D$ con discontinuidad dada por una recta: Representación gráfica del tanto por ciento de detalles guardados versus la norma 2 al cuadrado para diferentes tipos de filtros. Izquierda: sin cambio, derecha: con cambio PPH.

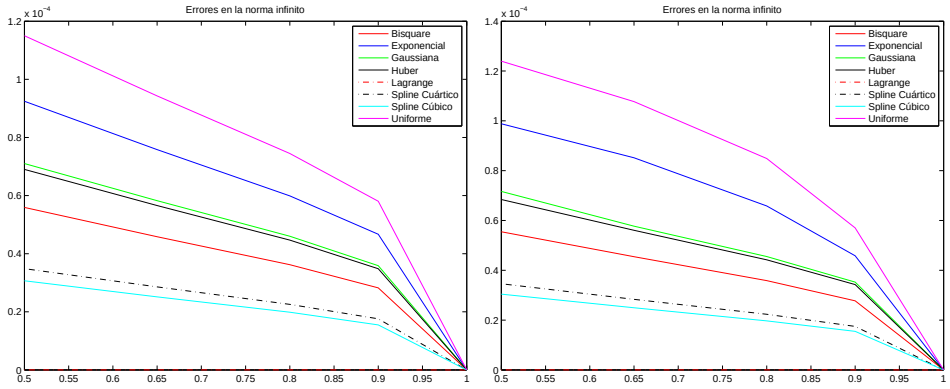

Figura 6.35: Función Gaussiana en $2D$ con discontinuidad una circunferencia: Representación gráfica del tanto por ciento de detalles guardados versus la norma infinito para diferentes tipos de filtros. Izquierda: sin cambio, derecha: con cambio PPH.

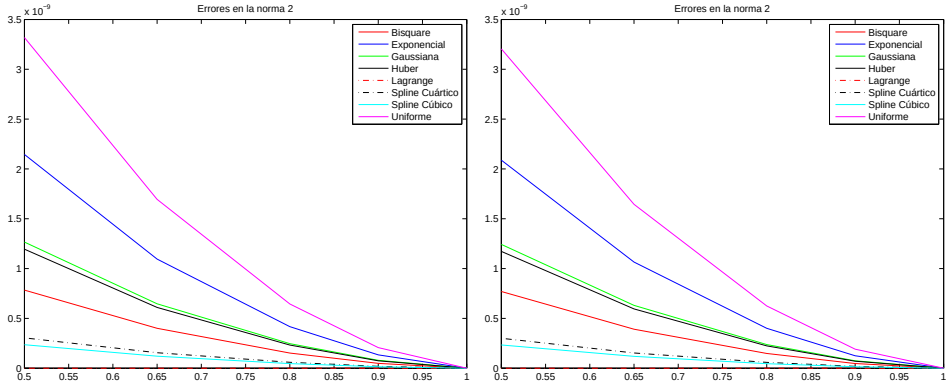

Figura 6.36: Función Gaussiana en $2D$ con discontinuidad una circunferencia: Representación gráfica del tanto por ciento de detalles guardados versus la norma 2 al cuadrado para diferentes tipos de filtros. Izquierda: sin cambio, derecha: con cambio PPH.

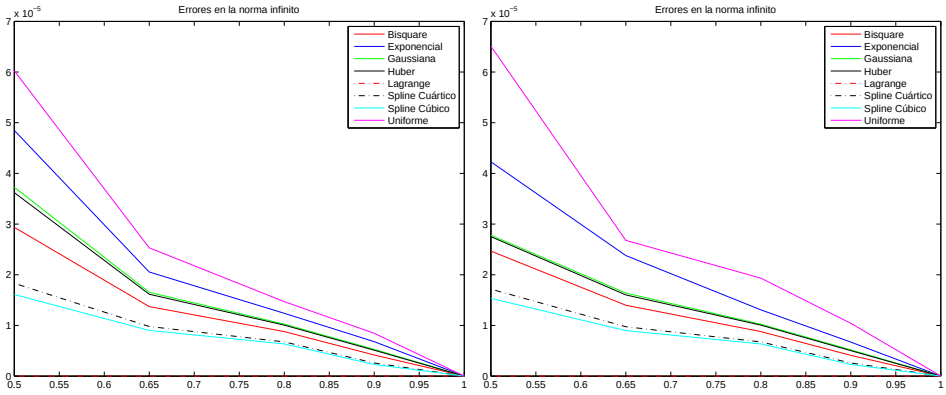

Figura 6.37: Función Hipérbola en $2D$ con discontinuidad una parábola: Representación gráfica del tanto por ciento de detalles guardados versus la norma infinito para diferentes tipos de filtros. Izquierda: sin cambio, derecha: con cambio PPH.

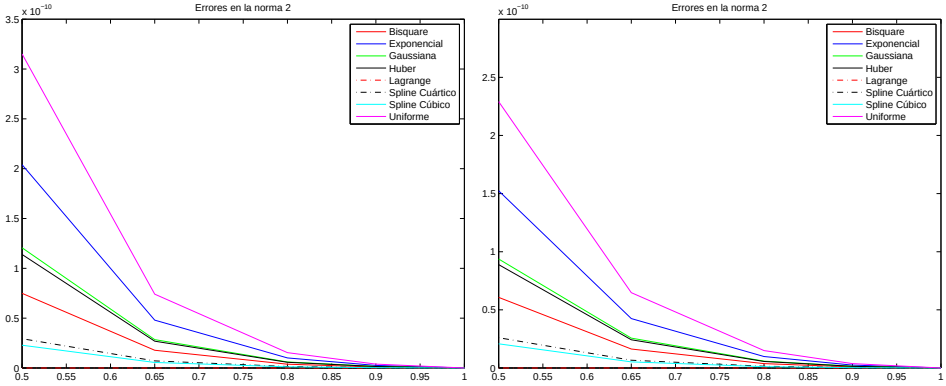

Figura 6.38: Función Hipérbola en $2D$ con discontinuidad una Parábola: Representación gráfica del tanto por ciento de detalles guardados versus la norma 2 al cuadrado para diferentes tipos de filtros. Izquierda: sin cambio, derecha: con cambio PPH.

Con el fin de mostrar de una manera más precisa las reconstrucciones obtenidas con los filtros lineales y no lineales en $2D$, consideramos una de las funciones en concreto y el filtro de Lagrange de 4 puntos con y sin cambio previo de los datos. Vamos a centrarnos en la función $f_{g2dis}(x, y)$ de tipo *Gaussiana en* $2D$ *con discontinuidad dada por la parábola de ecuación* $dis = y - x^2 = 0$ definida en la fórmula (6.21). En la figura 6.39 vemos la función original, la reconstrucción conseguida con el filtro de Lagrange de (4 puntos) sin cambio guardando el 2 % de detalles y la reconstrucción con el mismo filtro de 4 puntos guardando el 2 % de detalles, pero esta vez con cambio

previo de datos. Como puede observarse la discontinuidad está mucho mejor definida en el caso de realizar un tratamiento a los datos antes de aplicar el filtro localmente.

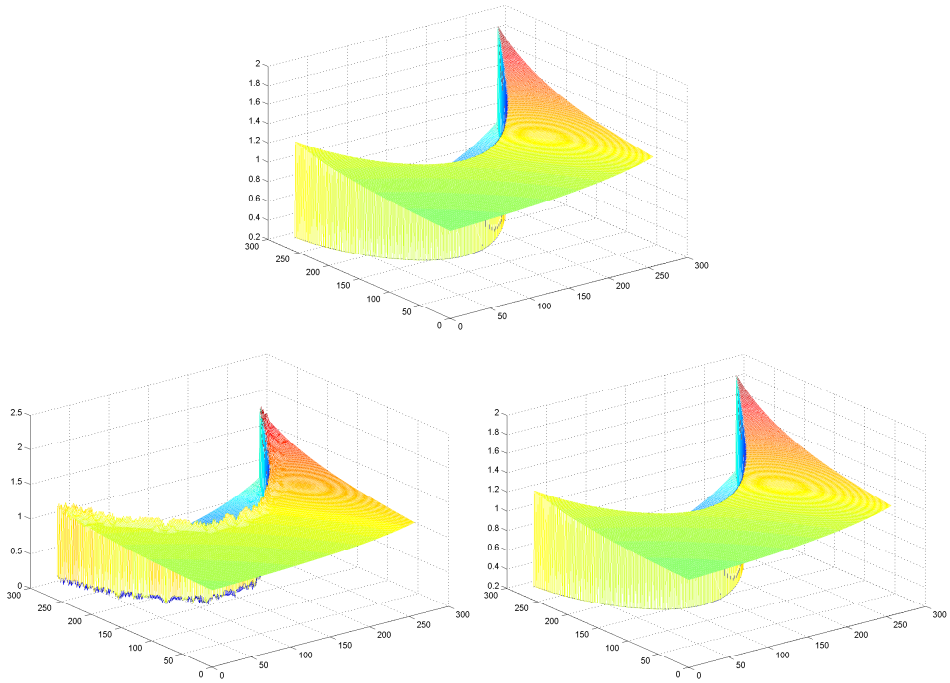

Figura 6.39: Parte superior: Función Original $f_{g2dis}(x, y)$ definida en (6.21) con la discontinuidad dada por una parábola, parte inferior izquierda: Aproximación obtenida por el filtro lagrange con 4 puntos sin cambio guardando el 2 % de detalles, parte inferior derecha: Lagrange con 4 puntos con cambio guardando el 2 % de detalles. $L = 3$ niveles de multirresolución.

Ahora vamos a proceder con la función $f_{e2dis}(x, y)$ de tipo *Exponencial en* $2D$ *con discontinuidad dada por una parábola de ecuación* $dis = y - x^2 = 0$ definida en la fórmula (6.20). Se muestra en la figura 6.40 la función original, la reconstrucción conseguida con el filtro Lagrange (4 puntos) sin cambio guardando el 2 % de detalles y la reconstrucción con el mismo filtro de 4 puntos guardando el 2 % de detalles, pero esta vez utilizando cambio previo de datos PPH. Al igual que el caso de trabajar con la función $f_{g2dis}(x, y)$ de tipo *Gaussiana en* $2D$ *con discontinuidad dada por una parábola*, la discontinuidad está mucho mejor definida en el caso de realizar un tratamiento a los datos antes de aplicar el filtro localmente.

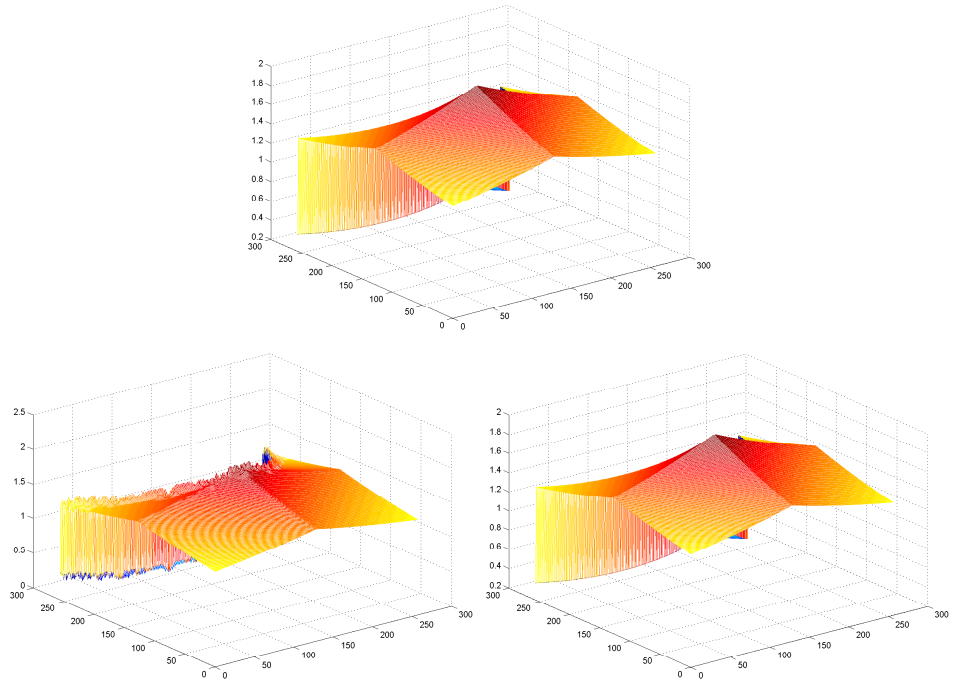

Figura 6.40: Parte superior: Función Original $f_{e2dis}(x, y)$ definida en (6.20) con la discontinuidad dada por una parábola, parte inferior izquierda: Aproximación obtenida por el filtro lagrange con 4 puntos sin cambio guardando el 2 % de detalles, parte inferior derecha: Lagrange con 4 puntos con cambio guardando el 2 % de detalles. $L = 3$ niveles de multirresolución.

Al guardar suficientes coeficientes de detalles, no se notan tanto las diferencias de mejora entre sin cambio de datos y con cambio previo de datos. Por esta razón, presentamos a continuación la gráfica 6.41 que muestra las reconstruccines obtenidas con la función $f_{e2dis}(x, y)$ de tipo *Exponencial en* $2D$ *con discontinuidad dada por una parábola de ecuación* $dis = y - x^2 = 0$ definida en la fórmula (6.20) utilizando varios filtros de reconstrucción de tamaño 4, variando los detalles guardados de 0 % a 20 % y fijando el número de escalas a 3. Es en estos casos en los que se guardan menos detalles en los que se aprecia mejor la diferencia entre realizar un cambio previo de los datos o no hacerlo.

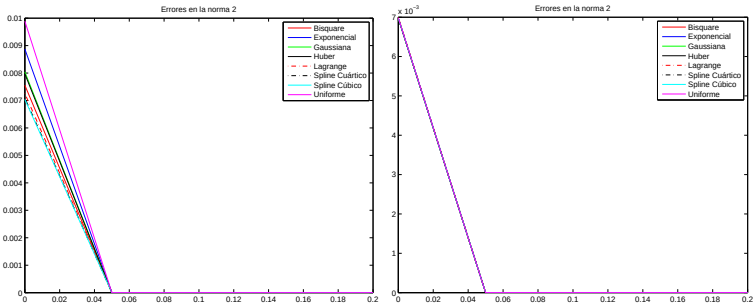

Figura 6.41: Función Exponencial en $2D$ con discontinuidad dada por una parábola: Representación gráfica del tanto por ciento de detalles guardados(de 0 % a 20 %) versus la norma 2 al cuadrado para diferentes tipos de filtros. Izquierda: sin cambio, derecha: con cambio PPH.

Capítulo 7

Conclusiones.

En este proyecto hemos realizado un estudio teórico práctico de los algoritmos de Multirresolución de Harten para el caso de discretizacón por valores puntuales utilizando diferentes operadores de predicción. En particular hemos estudiado el operador de predicción lineal basado en la interpolación de Lagrange y otros operadores de predicción correspondientes a distintas funciones de ponderación, así como hemos analizado cómo puede influir un cambio de datos con el operador no lineal de reconstrucción PPH en los resultados obtenidos al trabajar con datos que presentan discontinuidades.

Hemos desarrollado los programas Matlab necesararios para poner en práctica dichos algoritmos de multirresolución.

Son muy curiosas y a la vez poco intuitivas las conclusiones siguientes:

- Las reconstrucciones polinomiales son las que mejores resultados ofrecen.
- El filtro del tipo de la función de ponderación considerada no es en la mayoría de los casos el que mejor funciona, y ni siquiera lo es después de los filtros polinomiales.

Otra conclusión principal de este proyecto será que al utilizar técnicas de interpolación independientes de los datos, es decir, lineales, la capacidad de compresión del esquema de multirresolución se verá reducida en presencia de discontinuidades. Por otra parte, al utilizar técnicas de interpolación que dependan de los datos, es decir, no lineales, la capacidad de compresión del esquema de multirresolución mejora en estos casos. Utilizando el cambio de datos dado por el método PPH, se aprecia más la reducción del error cometido para la norma 2 que para la norma infinito, ya que continua existiendo un intervalo por discontinuidad donde la reconstrucción no se adapta y esto hace que el error máximo siga siendo alto en valor absoluto. Sin embargo el error

acumulado, expresado por la norma 2, sí va a decrecer ya que el número de intervalos afectados es menor que en el caso lineal.

Bibliografía

[1] S. Amat, *A review on the piecewise polynomial harmonic interpolation*, Appl. Num. Math., **58** (8), 1168-1185, (2008).

[2] S. Amat, *Nonseparable multiresolution with error control. Appl. Math. Comput.*, **145**(1), 117-132, (2003).

[3] S. Amat, F. Aràndiga, A. Cohen and R. Donat, *Tensor product multiresolution analysis with error control for compact image representation. Signal Processing*, **82**(4), 587-608, (2002).

[4] S. Amat, F. Aràndiga, A. Cohen, R. Donat, G.García and M.von Oehsen, *Data compression with ENO schemes: A case study*, Appl. Comp. Harm. Anal., **11**, 273-288, (2001).

[5] S. Amat, S. Busquier, and J.C. Trillo., *Stable Biorthogonal Multiresolution Transforms. Journal of Numerical Analysis, Industrial and Applied Mathematics*, **1**(3), 229-239, (2006).

[6] S. Amat, S. Busquier and V.F. Candela, *A polynomial approach to Piecewise Hyperbolic Method*, Int. J. Comput. Fluid Dyn., **17** (3), 205-217, (2003).

[7] S. Amat and J. Liandrat. *On the stability of PPH nonlinear multiresolution*, Appl. Comp. Harm. Anal., **18** (2), 198-206, (2005).

[8] S. Amat, K. Dadourian and J. Liandrat, *Analysis of a class of nonlinear subdivision schemes and associated multi-resolution transforms*, Adv. Comput. Math., **34**(3), 253-277,(2011).

[9] S. Amat, K. Dadourian, J.Liandrat and J.C. Trillo, *On a class of nonlinear interpolatory subdivision schemes*, submitted (2009).

[10] S. Amat, R. Donat and J.C. Trillo, *On specific stability bounds for linear multiresolution schemes based on piecewise polynomial Lagrange interpolation*, Journal of Math. Anal. and Appl., **1**, 18-27 , (2009).

[11] S.Amat, R.Donat, J.Liandrat and J.C.Trillo, *Analysis of a new nonlinear subdivision scheme. Applications in image proceszjsing*, Found. Comput. Math., **6** (2), 193-225, (2006).

[12] F. Aràndiga and R. Donat, *Nonlinear Multi-scale Decomposition: The Approach of A.Harten. Numerical Algorithms.*, **23**, 175-216, (2000).

[13] F. Aràndiga and R. Donat, *Stability through synchronization in nonlinear multiscale transformations*, SIAM J. Sci. Comput., **29**(1), 265-289,(2007).

[14] R.L. Claypoole, G. M. Davis, W. Sweldens, R.G. Baraniuk, *Nonlinear wavelet transforms for image coding via lifting.*, IEEE Trans. Image Process, **12**(12), 1449-1459, (2003).

[15] T. Chan and H.M.Zhou, *ENO-wavelet transforms for piecewise smooth functions. SIAM J. Numer. Anal*, **40** (4), 1369-1404, (2002).

[16] A. Cohen, N. Dyn and B. Matei, *Quasilinear subdivision schemes with applications to ENO interpolation*, Appl. Comp. Harm. Anal., **15**, 89-116, (2003).

[17] G. Delauries and S. Dubuc, *Symmetric Iterative Interpolation Scheme. Constructive Approximations*, SIAM J. Sci. Comput., **5**, 49-68,(1989).

[18] N. Dyn, *Subdivision schemes in computer aided geometric design*, Oxford University Press, **20**(4), 36-104, (1992).

[19] M.S. Floater and C.A. Micchelli, *Nonlinear stationary subdivision, Approximation theory: in memory of A.K. Varna*, edt: Govil N.K, Mohapatra N., Nashed Z., Sharma A., Szabados J., 209-224, (1998).

[20] A. Harten, *Discrete multiresolution analysis and generalized wavelets.*, J. Appl. Numer. Math., **12**, 153-192, (1993).

[21] A. Harten, *ENO schemes with subcell resolution.*, J. Comput. Phys., **83**, 148-184, (1989).

[22] A. Harten, *Multiresolution representation of data II.*, SIAM J. Numer. Anal., **33** (3), 1205-1256, (1996).

[23] A. Harten, B. Engquist, S.J. Osher and S.R. Chakravarthy, *Uniformly high order accurate essentially non-oscillatory schemes III.*, J. Comput. Phys., **71**, 231-303, (1987).

[24] F. Kuijt, *Convexity Preserving Interpolation - Stationary Nonlinear Subdivision and Splines.*, PhD Thesis, University of Twenty, Faculty of Mathematical Sciences, (1998).

[25] A. Marquina, *Local piecewise hyperbolic reconstruction of numerical fluxes for nonlinear scalar conservation laws.*, SIAM J. Sci. Comput., **15** (4), 892-915, (1994).

[26] M. Rabbani and P.W. Jones, *Digital Image Compression Techniques. Tutorial Text*, Society of Photo-Optical Instrumentation Engineers (SPIE), TT07, (1991).

[27] S. Serna and A.Marquina, *Power ENO methods: a fifth order accurate Weighted Power ENO method*, J. Comput. Phys., **194**, 632-658, (2004).

[28] W. Sweldens, *The lifting scheme: a custum-design construction of biorthogonal wavelets, Applied and Computational Harmonic Analysis*, **3** (2), 186-200, (1996).

[29] W. Sweldens, *The lifting scheme: a construction of second generation wavelets, SIAM Journal on Mathematical Analysis*, **29** (2), 511-546, (1998).

[30] W. Sweldens and P. Schröder, *Building your own wavelets at home, Wavelets in Computers Graphics, ACM SIGGRAPH Course notes*, 15-87, (1996).

[31] J.C. Trillo, *Nonlinear multiresolution and applications in image processing*, PhD in the University of Valencia, Spain, (2007).

[32] B. Zhang, J.M. Fadili, J.L. Starck, Wavelets, *ridgelets, and curvelets for Poisson noise removal.*, IEEE Trans. Image Process, **17** (7), 1093-1108, (2008).