Compresión y Zoom de Datos Geológicos utilizando Algoritmos de Multirresolución

Autor: Miguel Gálvez Rodríguez
Directores: Juan Carlos Trillo Moya
María José Moncayo Hormigo

Julio 2009

Autor	Miguel Gálvez Rodríguez
E-mail del Autor	m.galvez.rodriguez@hotmail.es
Directores	Maria José Moncayo Hormigo, Juan Carlos Trillo Moya
E-mail de los Directores	Maria.moncayo@upct.es,jctrillo@upct.es
Codirector	
Título del PFC	Compresión y Zoom de Datos Geológicos utilizando Algoritmos de Multirresolución
Descritores	zoom, compresión, multirresolución, datos geológicos
Resúmen	Métodos de multirresolución aplicados a la compresión y zoom de datos geológicos. Así por ejemplo dada una región montañosa de la cual poseemos información precisa de la altura de ciertos puntos, podremos estimar la altura en otros puntos que nos interesen sin volver a realizar nuevas mediciones. Esto es de especial importancia también cuando en vez de alturas tratamos de medir profundidades de cierta superficie en un entorno marino donde cada medición supone un gran coste económico. Igualmente es interesante guardar los datos de una manera precisa y a ser posible ocupando el mínimo espacio posible.
Titulación	Ingeniero Técnico de Telecomunicaciones, Especialidad Telemática
Departamento	Matemática Aplicada y Estadística
Fecha de Presentación	Julio de 2009

Quiero dedicarle este proyecto a todos esos momentos que me perdí por haber estado realizando este trabajo, aunque no me arrepiento de ello, por que me brindaron la oportunidad de aprender. A toda mi familia por haberme apoyado incondicionalmente y enseñarme que lo que uno se propone se logra. A mis amigos y compañeros que juntos recorrimos este camino. A mis directores ***María José*** *y* ***Juan Carlos*** *por la infinita paciencia y ayuda que han tenido conmigo en este proyecto, por sus consejos, por confiar en mi y por brindarme la oportunidad de poder realizarlo en el Departamento de Mátematica Aplicada y Estadística.*

¡Gracias!

Índice de Tablas

Índice de figuras

Índice general

Capítulo 1

Introducción

En las últimas décadas se han desarrollo técnicas de descomposición de datos en escalas como pueden ser las técnicas de multirresolución que explicaremos en este proyecto. Entre sus aplicaciones podemos citar muchas, desde el procesamiento de imágenes digitales, de sonido o de señales en general a la compresión de datos de otra naturaleza, pasando también por la capacidad de estos algoritmos para abaratar otros procesos que computacionalmente son muy costosos.

Nuestro objetivo va a ser el estudio de métodos de multirresolución aplicados a la compresión y zoom de datos geológicos. Así por ejemplo dada una región montañosa de la cual poseemos información precisa de la altura de ciertos puntos, podremos estimar la altura en otros puntos que nos interesen sin volver a realizar nuevas mediciones. Esto es de especial importancia también cuando en vez de alturas tratamos de medir profundidades de cierta superficie en un entorno marino donde cada medición supone un gran coste económico. Igualmente es interesante guardar los datos de una manera precisa y a ser posible ocupando el mínimo espacio posible. Éste será otro punto que tendremos en cuenta en nuestro estudio.

Por un lado utilizaremos los esquemas de subdivisión, que a groso modo, lo que hacen es lo siguiente: dado un conjunto discreto de datos, se va refinando este conjunto, añadiendo nuevos datos a partir de los datos anteriores mediante la aplicación de una regla adecuada. Este proceso se va repitiendo sucesivamente hasta obtener una red suficientemente tupida para la aplicación que llevemos en mente.

Intrinsecamente ligados a los esquemas de subdivisión aparecen los esquemas de multirresolución. Dichos esquemas están basados en dos operadores básicos llamados operador de discretización y operador de reconstrucción. A

partir de dichos operadores aparecen otros dos que son los que se encargan de conectar diferentes niveles de multirresolución, a saber, el operador de decimación y el operador de predicción. Los operadores de discretización y de decimación deben de ser lineales, mientras que el de reconstrucción y el de predicción pueden ser no lineales. Esta no linealidas permite mucha flexibilidad para adaptarse a diferentes situaciones prácticas (ver [9],[2],[20], [13] para más detalles).

En este proyecto vamos a estudiar dos operadores de reconstrucción particulares en el entorno de una discretización por valores puntuales y vamos a comparar sus resultados para diferentes ordenes de aproximación cuando son aplicados a la compresión y al zoom de datos geológicos. Para ello llevaremos a cabo una bateria de experimentos con diferentes tipos de mapas. El primero de dichos operadores es un operador lineal basado en la interpolación segmentaria de Lagrange centrada de orden $r = 2s$, donde $2s$ indica el número de puntos usados para construir cada uno de los trozos de la reconstrucción. En segundo lugar utilizaremos una modificación no lineal del operador anterior. La modificación tiene por objetivo el mejorar el rendimiento de los algoritmos en presencia de discontinuidades de salto en los datos (para una documentación más amplia sobre estos operadores consultar [7],[6],[5]).

Este proyecto se organiza de la manera siguiente. En la sección 2 explicamos brevemente la multirresolución de Harten en el entorno de valores puntuales y cómo aplicarla para realizar zoom o compresión de datos. En la sección 3 presentamos los dos tipos de operadores de reconstrucción, uno lineal y uno no lineal, que van a ser considerados durante este estudio. En la sección 4 detallamos los programas Matlab utilizados. A continuación en la sección 5 documentamos la interfaz de usuario creada para facilitar el manejo de los programas. Ya en la sección 6 ofrecemos una comparativa entre los algoritmos propuestos. Y finalmente en la sección 7 extraemos algunas conclusiones del trabajo.

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, y es un elemento perteneciente a un espacio, 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:

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

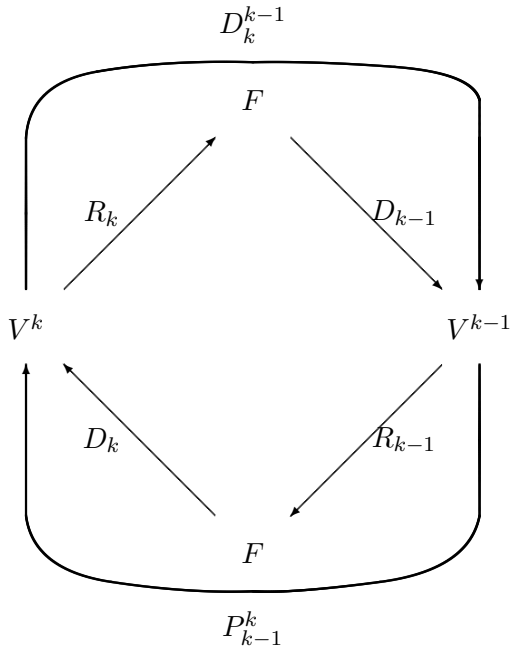

Figura 2.1: Definición de operadores.

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:

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

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:

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

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

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 $\left\{ f^{k-1}, d^k \right\}$ tenemos $N_{k-1} + (N_k - N_{k-1}) = N_k$ elementos, los mismos que hay en f^k. Decimos entonces que los coeficientes $\left\{ d_i^k \right\}$ 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:

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

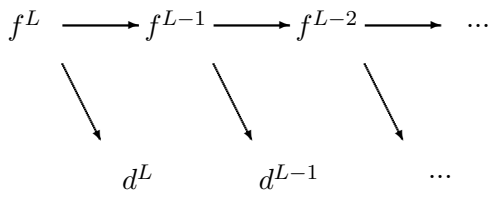

Figura 2.2: Descomposición multiescala.

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. Habitualmente se utilizan dos tipos de discretización: la discretización por valores puntuales y la discretización por medias en celda.

Para este proyecto en cuestión, se utilizará la discretización por valores puntuales, que pasamos a describir en el siguiente apartado.

2.1. Multirresolución para la discretización por valores puntuales en [0,1]

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

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

El valor de los coeficientes d_j^k varía en función de la singularidad, por lo que podemos pensar en el análisis de multirresolución como un análisis de la regularidad de una función. Los mayores coeficientes 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. Producto tensor en 2D

En primer lugar vamos a proceder a explicar de manera gráfica para que sea más fácilmente 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 ([2]).
Para la matriz original, el primer nivel de multirresolución se realiza de la siguiente manera:

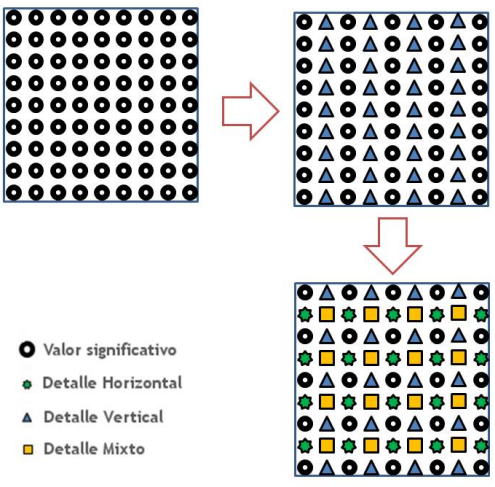

Figura 2.3: Pasos del algoritmo de multirresolución: mapa original, algoritmo 1D aplicado a las filas y algoritmo 1D aplicado a las columnas

La matriz en la esquina superior izquierda corresponde con la matriz original, la segunda matriz corresponde con la aplicación del algoritmo 1-dimensional a 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 estrellas verdes los detalles horizontales, con triángulos azules los detalles verticales 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 2.3).

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 2.4.

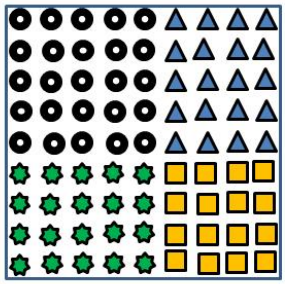

Figura 2.4: Versión de multirresolución del mapa una vez reorganizados los coeficientes

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.

2.3. Compresión de datos bidimensionales

Hemos visto que, dados unos datos bidimensionales, una representación de un nivel de multirresolución de dichos datos viene dada en la Figura 2.4. Este proceso de descenso de un nivel de multirresolución puede expresarse matricialmente en la forma

$$A^k \leftrightarrow \left(\begin{array}{c|c} A^{k-1} & \Delta_2^k \\ \hline \Delta_3^k & \Delta_1^k \end{array} \right) \tag{2.11}$$

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.

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) Dada 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 de detalles.

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

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

Y calcularemos tanto errores absolutos como errores relativos en las aproximaciones.

2.4. Zoom de datos bidimensionales

El zoom de datos bidimensionales consiste en aplicar primero un algoritmo de predicción en una dimensión a cada una de las filas de la imagen para así doblar el número de puntos y después realizar la misma operación por columnas.
La matriz en la esquina superior izquierda de la Figura 2.5 corresponde con el mapa original (donde el número de puntos se corresponden con el número de píxeles), la segunda matriz corresponde con la aplicación del algoritmo 1-dimensional a cada una de las filas. Hemos representado mediante circulos negros los valores iniciales y los triángulos azules son los valores de las predicciones hechas por filas. La tercera matriz, situada debajo de dichas matrices, corresponde con el resultado de aplicar el algoritmo 1-dimensional a cada una de las columnas del resultado del paso anterior. Los valores predecidos de los nuevos píxeles, quedan representados con estrellas verdes y con cuadrados amarillos.

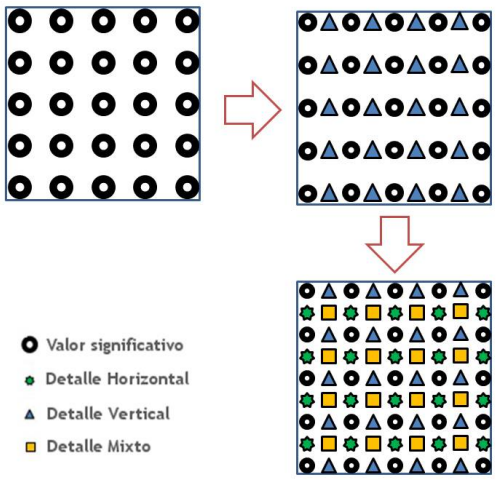

Figura 2.5: Pasos del algoritmo de zoom: mapa original, algoritmo 1D aplicado a las filas y algoritmo 1D aplicado a las columnas.

Capítulo 3

Reconstrucciones lineales y no lineales

En este capítulo vamos a presentar una reconstrucción lineal en el entorno de valores puntuales basada en una interpolación segmentaria de Lagrange centrada de orden $2s$, es decir, utilizando $2s$ puntos en el stencil de interpolación. También presentaremos una modificación de la misma, que trata de evitar el efecto Gibbs cerca de las discontinuidades, definiéndolas mejor. Dicha modificación es dependiente de los datos, y por tanto no lineal. Se conseguirá mantener el orden en las zonas de suavidad y no perderlo completamente en presencia de singularidades.

Pasamos primero a exponer la reconstrucción basada en interpolación de Lagrange. La describiremos en 1D y primero usando sólo 4 puntos para después dar la expresión general con $2s$ puntos.

3.1. 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

$$P_j(x) = f_{j-1}L_{-1}(x) + f_jL_0(x) + f_{j+1}L_1(x) + f_{j+2}L_2(x) \qquad (3.1)$$

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

$$L_i(x) = \prod_{l=-1, l\neq i}^{2} \frac{(x - x_{j+l})}{(x_{j+i} - x_{j+l})}. \tag{3.2}$$

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

$$P_j(x_{j+\frac{1}{2}}) = f_{j-1}L_{-1}(x_{j+\frac{1}{2}}) + f_jL_0(x_{j+\frac{1}{2}}) + f_{j+1}L_1(x_{j+\frac{1}{2}}) + f_{j+2}L_2(x_{j+\frac{1}{2}}), \tag{3.3}$$

y haciendo algunos cálculos

$$P_j(x_{j+\frac{1}{2}}) = -\frac{1}{16}f_{j-1} + \frac{9}{16}f_j + \frac{9}{16}f_{j+1} - \frac{1}{16}f_{j+2}. \tag{3.4}$$

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 (3.1). 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

$$P_j(x) = \sum_{i=-s+1}^{s} f_{j+i}L_i(x_{j+\frac{1}{2}}), \quad x \in [x_j, x_{j+1}], \tag{3.5}$$

donde los polinomios de Lagrange $L_i(x)$ se definen como en (3.2) por

$$L_i(x) = \prod_{l=-s+1, l\neq i}^{s} \frac{(x - x_{j+l})}{(x_{j+i} - x_{j+l})}. \tag{3.6}$$

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

$$P_j(x_{j+\frac{1}{2}}) = \sum_{i=-s+1}^{s} f_{j+i}L_i(x_{j+\frac{1}{2}}). \tag{3.7}$$

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 3.1 podemos ver los valores de las máscaras 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 [7].

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

3.2. 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

$$B_{s-1}^{s-1} = 2, \tag{3.8}$$

$$B_{s-1}^{q} = \sum_{j=1}^{s-1-q} (-1)^j \left(\binom{2(q+j)}{(j-1)} - \binom{2(q+j)}{j} \right) B_{s-1}^{q+j}, \tag{3.9}$$

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

En la Tabla 3.2 podemos ver el valor de estos coeficientes para $s = 2, 3, 4$.

También vamos a necesitar la medias p-power $power_p(x, y)$, que se introdujeron en [19] para cualquier número entero $p \geq 1$, y cualquier pareja (x, y) como:

$$power_p(x, y) = \frac{sign(x) + sign(y)}{2} \frac{x+y}{2} \left(1 - \left| \frac{x-y}{x+y} \right|^p \right). \tag{3.10}$$

Igualmente necesitaremos usar las diferencias divididas, que es conocido que actúan como indicadores de zonas de suavidad de la función. Es

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

Tabla 3.2: Coeficientes B^i_{s-1}, $s=2,3,4$.

conocido que 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 [6] y [5] 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

  $$\tilde{f} := \begin{cases} (\ldots, f_{j-2}, f_{j-1}, f_j, f_{j+1}, \tilde{f}_{j+2}, f_{j+3}, \ldots) & \text{si } |\Delta^2_{j-1}f| \leq |\Delta^2_j f|, \\ (\ldots, f_{j-2}, \tilde{f}_{j-1}, f_j, f_{j+1}, f_{j+2}, f_{j+3}, \ldots) & \text{en otro caso.} \end{cases} \tag{3.11}$$

- 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

$$f = \begin{cases} (\ldots, f_{j-2}, f_{j-1}, f_j, f_{j+1}, \tilde{f}_{j+2}, f_{j+3}, \ldots) & \text{si } |\Delta^2_{j-1}f| \leq |\Delta^2_j f|, \\ (\ldots, f_{j-2}, \tilde{f}_{j-1}, f_j, f_{j+1}, f_{j+2}, f_{j+3}, \ldots) & \text{en otro caso.} \end{cases} \tag{3.12}$$

y definimos

$$\tilde{f} := \begin{cases} (\ldots, f_{j-3}, f_{j-2}, \tilde{f}_{j-1}, \tilde{f}_j, \tilde{f}_{j+1}, \tilde{f}_{j+2}, \tilde{f}_{j+3}, f_{j+4} \ldots) & \text{si } |\Delta^4_{j-2}\tilde{f}| \leq |\Delta^4_{j-1}\tilde{f}|, \\ (\ldots, f_{j-3}, \tilde{f}_{j-2}, \tilde{f}_{j-1}, \tilde{f}_j, \tilde{f}_{j+1}, \tilde{f}_{j+2}, f_{j+3}, f_{j+4} \ldots) & \text{en otro caso.} \end{cases} \tag{3.13}$$

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 3.1. 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 3.2.

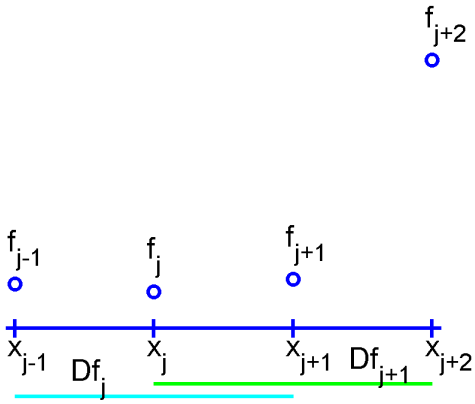

Figura 3.1: Primer paso de la reconstrucción PPH con 4 puntos

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

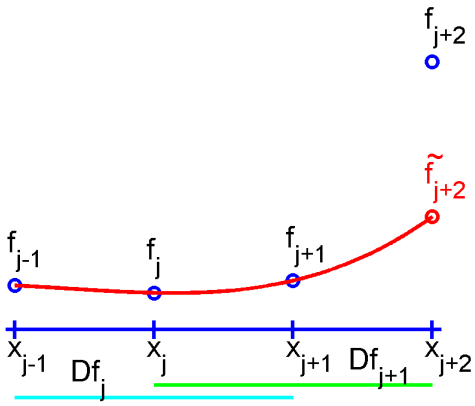

Figura 3.2: Construcción de la reconstrucción PPH con 4 puntos con los valores modificados

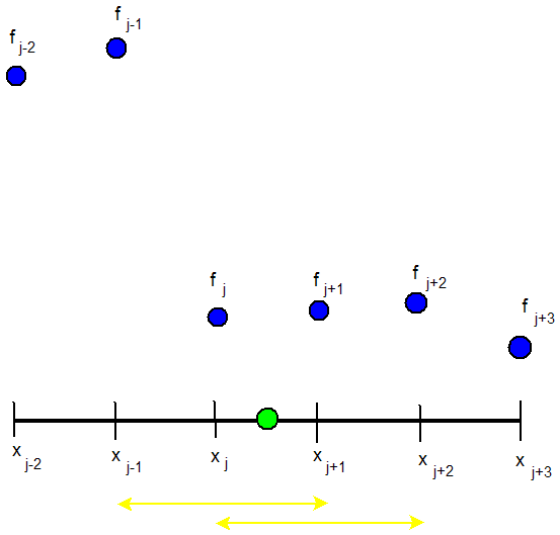

Figura 3.3: Primer paso de la reconstrucción PPH con 6 puntos

Figura 3.3. 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 3.4. A continuación se considerarán las diferencias di-

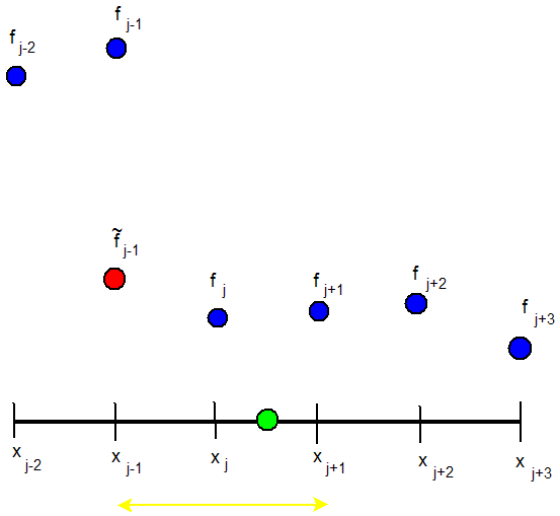

Figura 3.4: Modificación del primer valor en la construcción de la reconstrucción PPH con 6 puntos

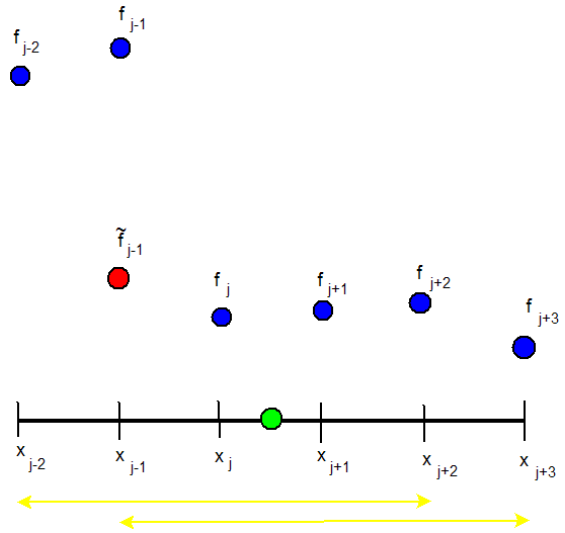

Figura 3.5: Segundo paso de la reconstrucción PPH con 6 puntos

vididas de tercer orden indicadas en la Figura 3.5 y por tanto se cambiará el valor f_{j-2} tal y como se ve en la Figura 3.6.

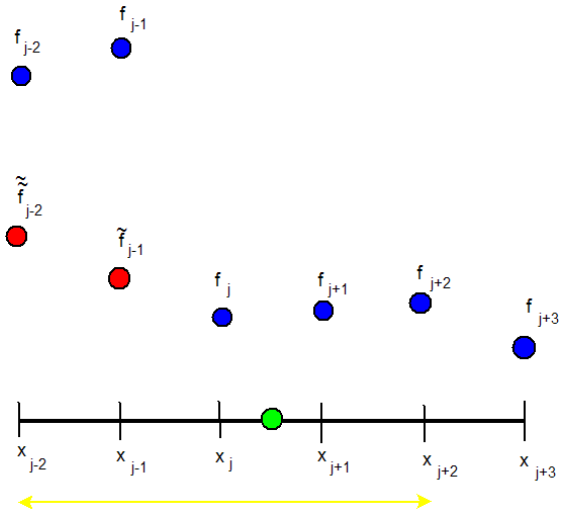

Figura 3.6: Modificación del segundo valor en la construcción de la reconstrucción PPH con 6 puntos

Capítulo 4

Códigos de los algoritmos en MATLAB

Se ha empleado el programa de cálculos científico MATLAB 7.0. A continuación aparecen los códigos de los distintos algoritmos vistos en capítulos anteriores, que han sido programados en este entorno. MATLAB se encargará de llevar a cabo todas las operaciones matemáticas necesarias para cada método.

4.1. Compresión de datos bidimensionales

4.1.1. *Diagrama de flujo*

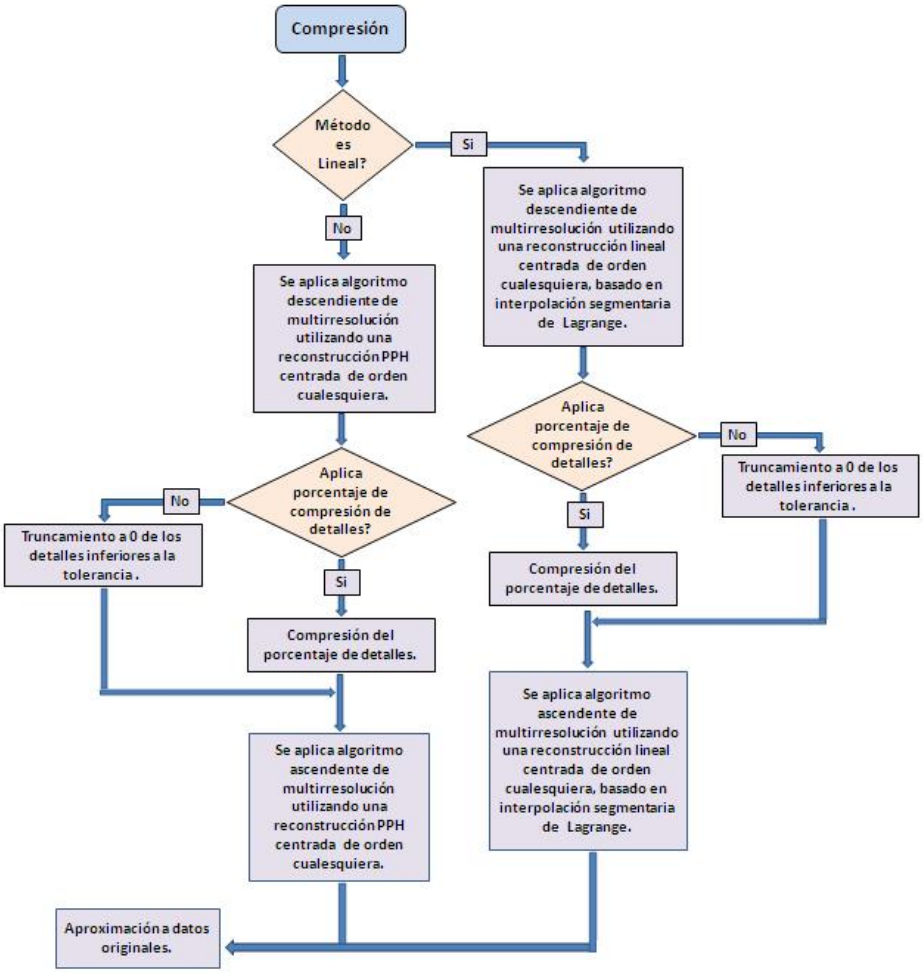

Figura 4.1: Diagrama de Flujo de Compresión de datos bidimensionales.

4.1.2. *Código*

```
function a = tensor(mapa,l,tp,com,met,or)


% Función que realiza compresión de datos geológicos mediante
% algoritmos de multirresolución basados en interpolación de
% Lagrange o en la reconstrucción PPH.
% a = tensor(mapa,l,tp,com,met,or)
% Variables de entrada:
% mapa matriz de datos.
% l niveles de multirresolución que desciendes.
% com indicador de tipo de compresión toma valor 0 ó 1.
% tp dependiendo del valor introducido en com:
% - 0.tolerancia de truncamiento.
% - 1.porcentaje de compresión de detalles.
% met método de reconstrucción utilizado.
% or orden de la reconstrucción.
% Variable de salida:
% a aproximación a los datos originales.



% Dimensión de la matriz de datos.

[n,m]=size(mapa);


% Descendemos por la pirámide de multiresolución.

[mra,rat,tanto_por_cien]=descender(mapa,l,com,tp,met,or);


% Ascendemos por la pirámide de multiresolución.

a=ascender(mra,l,met,or);


% Calculamos los errores.

norma1=sum(sum(abs(mapa-a)))/n/m;
infin=max(max(abs(mapa-a)));
norma2=sum(sum((mapa-a).^2))/n/m;
```

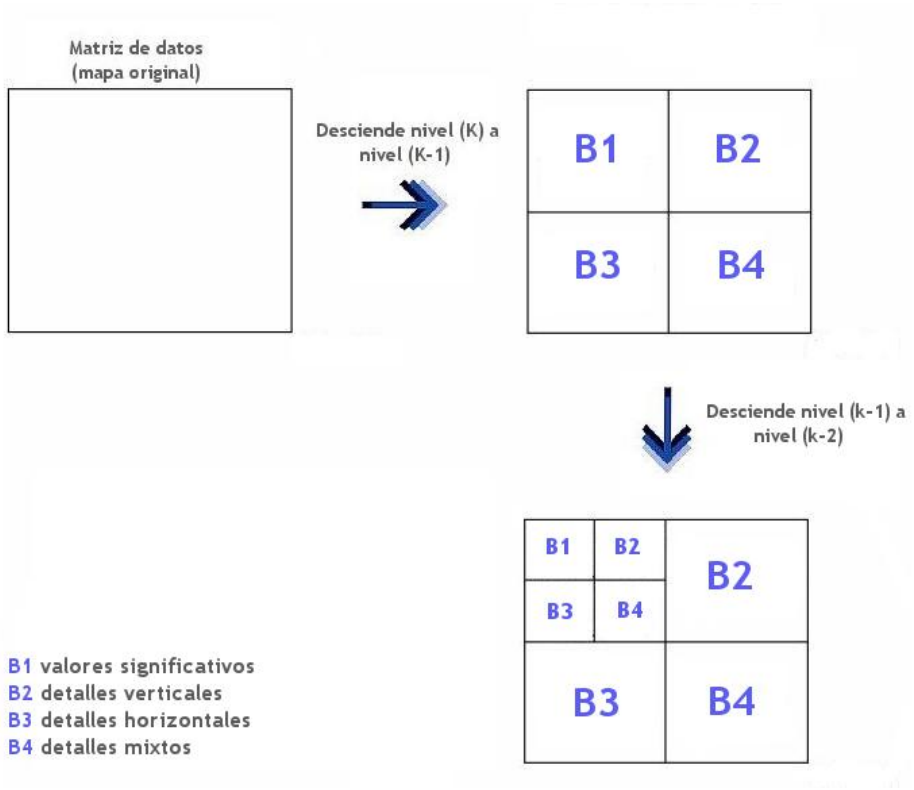

Figura 4.2: Proceso de codificación.

```
function [c,rat,tanto_por_cien]=descender(a,l,com,tp,met,or)


% Función que realiza el proceso de codificación del algoritmo
% de compresión, desciende k niveles de multirresolución.
% Dependiendo de la opción seleccionada, se aplica un
% determinado truncamiento al mapa codificado o bien comprime
% el porcentaje de detalles seleccionado.
% [c,rat,tanto_por_cien]=descender(a,l,com,tp,met,or)
% Variables de entrada:
% a matriz a la que le aplicamos el algorimo descendente de
% multirresolución.
% l niveles de multirresolución que desciende.
% com indicador de tipo de compresión toma valor 0 ó 1.
% tp dependiendo del valor introducido en com:
```

```
% - 0.tolerancia de truncamiento.
% - 1.porcentaje de compresión de detalles.
% met método de reconstrucción utilizado.
% or orden de la reconstrucción.
% Variables de salida:
% c versión de multirresolución de la matriz.
% rat tasa de compresión conseguida por el método.
% tanto_por_cien porcentaje del tamaño original empleado para
% almacenar la versión comprimida.



% Tamaño del mapa original.

[n,m]=size(a);
n1=n; m1=m;

% Dimensión de la matriz con resolución más baja.

sn=(n-1)/2^l+1;
sm=(m-1)/2^l+1;

% Inicialización de las variables.

nnceros=0;
c=a;

if met=='lin'

% Bucle para los niveles de multirresolución.

    for k=1:l

% Algoritmo de multirresolución por filas.

        for i=1:n
            [f1,f2]=lineale(c(i,1:m),m,or);
            c(i,1:2:m)=f1;
            c(i,2:2:m-1)=f2;
        end
        clear f1; clear f2;

% Algoritmo de multirresolución por columnas.
```

```
        for j=1:m
            [f1,f2]=lineale(c(1:n,j)',n,or);
            c(1:2:n,j)=f1';
            c(2:2:n-1,j)=f2';
        end
        clear f1; clear f2;

% Ordena la matriz.

        b1=c(1:2:n,1:2:m); b2=c(1:2:n,2:2:m-1);
        b3=c(2:2:n-1,1:2:m); b4=c(2:2:n-1,2:2:m-1);

        if com==0

% Pone a 0 los detalles menores a la tolerancia introducida.

            [I2,J2]=find(abs(b2)<tp);
            for i=1:length(I2)
                b2(I2(i),J2(i))=0;
            end
            clear I2; clear J2;


% Cuenta elementos no cero.

            nnceros=nnceros+nnz(b2);


% Pone a 0 los detalles menores a la tolerancia introducida.

            [I3,J3]=find(abs(b3)<tp);
            for i=1:length(I3)
                b3(I3(i),J3(i))=0;
            end
            clear I3; clear J3;


% Cuenta elementos no cero.

            nnceros=nnceros+nnz(b3);


% Pone a 0 los detalles menores a la tolerancia introducida.
```

```
            [I4,J4]=find(abs(b4)<tp);
            for i=1:length(I4)
                b4(I4(i),J4(i))=0;
            end
            clear I4; clear J4;


% Cuenta elementos no cero.

            nnceros=nnceros+nnz(b4);

        end

        c(1:n,1:m)=[b1,b2;b3,b4];
        clear b1; clear b2; clear b3; clear b4;

        n=(n-1)/2+1;
        m=(m-1)/2+1;
    end

elseif met=='pph'

% Bucle para los niveles de multirresolución.

    for k=1:l

% Algoritmo de multirresolución por filas.

        for i=1:n
            [f1,f2]=pphe(c(i,1:m),m,or);
            c(i,1:2:m)=f1;
            c(i,2:2:m-1)=f2;
        end
        clear f1; clear f2;

% Algoritmo de multirresolución por columnas.

        for j=1:m
            [f1,f2]=pphe(c(1:n,j)',n,or);
            c(1:2:n,j)=f1;
            c(2:2:n-1,j)=f2';
        end
        clear f1; clear f2;
```

```
% Ordena la matriz.

        b1=c(1:2:n,1:2:m); b2=c(1:2:n,2:2:m-1);
        b3=c(2:2:n-1,1:2:m);b4=c(2:2:n-1,2:2:m-1);

        if com==0


% Pone a 0 los detalles menores a la tolerancia introducida.

            [I2,J2]=find(abs(b2)<tp);
            for i=1:length(I2)
                b2(I2(i),J2(i))=0;
            end
            clear I2; clear J2;


% Cuenta elementos no cero.

            nnceros=nnceros+nnz(b2);


% Pone a 0 los detalles menores a la tolerancia introducida.

            [I3,J3]=find(abs(b3)<tp);
            for i=1:length(I3)
                b3(I3(i),J3(i))=0;
            end
            clear I3; clear J3;


% Cuenta elementos no cero.

            nnceros=nnceros+nnz(b3);


% Pone a 0 los detalles menores a la tolerancia introducida.

            [I4,J4]=find(abs(b4)<tp);
            for i=1:length(I4)
                b4(I4(i),J4(i))=0;
            end
            clear I4; clear J4;
```

```
% Cuenta elementos no cero.

            nnceros=nnceros+nnz(b4);

        end

        c(1:n,1:m)=[b1,b2;b3,b4];
        clear b1; clear b2; clear b3; clear b4;

        n=(n-1)/2+1;
        m=(m-1)/2+1;
    end
end

if com==1

% Compresión de detalles.

    b=c;
    b(1:n,1:m)=0;
    f=b(:);
    [y, ind]=sort(abs(f),1,'descend');
    y=f(ind);
    y(tp+1:end)=0;
    f(ind)=y;
    e=zeros(size(b));
    e(1:end)=f;
    e(1:n,1:m)=c(1:n,1:m);

    c=e;
    clear e;
    clear b;

    rat=(n1*m1)/(tp+sn*sm);
    tanto_por_cien=100*(tp+sn*sm)/n1/m1;

else
    rat=(n1*m1)/(nnceros+sn*sm);
    tanto_por_cien=100*(nnceros+sn*sm)/n1/m1;
end
```

```
function [f1,f2]=lineale(f,n,or)


% Función que desciende un nivel de multirresolución utilizando
% una reconstrucción lineal centrada de or puntos, basada en
% interpolación segmentaria de Lagrange.
% [f1,f2]=lineale(f,n,or)
% Variables de entrada:
% f vector de la escala superior.
% n dimension de f.
% or orden de la reconstrucción.
% Variables de salida:
% f1 valores significativos de la escala inferior.
% f2 detalles entre las escalas.

% Valores significativos de la escala inferior.

f1=f(1:2:n);

% Longitud del vector f1.

nk1=(n+1)/2;

% Cálculo de las máscaras necesarias.

masc=maskslr(or/2,or/2);
mas=zeros(or,or/2-1);

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

% Cálculo de los detalles en la frontera izquierda.

f2(1:or/2-1)=f(2:2:or-2)-f1(1:or)*mas;

% Cálculo de los detalles intermedios.

for i=or/2:nk1-or/2
    f2(i)=f(2*i)-masc*f1(i-or/2+1:i+or/2)';
end

% Cálculo de los detalles en la frontera derecha.
```

```
f2(nk1-1:-1:nk1-or/2+1)=f(n-1:-2:n-or+3)-f1(nk1:-1:nk1-or+1)*mas;

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


% Este programa calcula las máscaras del esquema de subdivisión
% basado en Lagrange de orden n 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 conteniendo las máscaras
% numer vector conteniendo el numerador de las máscaras
% deno vector conteniendo el denominador de las máscaras


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 [f1,f2]=pphe(f,n,or)

% Función que desciende un nivel de multirresolución
% utilizando una reconstrucción pph centrada de or puntos.
% [f1,f2]=pphe(f,n,or)
% Variables de entrada:
% f vector de la escala superior.
% n dimensión de f.
% or orden de la reconstrucción.
% Variables de salida:
% f1 valores significativos de la escala inferior.
% f2 detalles entre las escalas.


% Valores significativos de la escala inferior.
f1=f(1:2:n);
% Longitud del vector f1.
nk1=(n+1)/2;
% Cálculo de las máscaras necesarias.
masc=maskslr(or/2,or/2);
mas=zeros(or,or/2-1);

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

% Cálculo de los detalles en la frontera izquierda.
f2(1:or/2-1)=f(2:2:or-2)-f1(1:or)*mas;
% Cálculo de los detalles intermedios.
for i=or/2:nk1-or/2
    f1_modify=f_modify(f1(i-or/2+1:i+or/2));
    f2(i)=f(2*i)-masc*f1_modify';
end
% Cálculo de los detalles en la frontera derecha.
f2(nk1-1:-1:nk1-or/2+1)=f(n-1:-2:n-or+3)-f1(nk1:-1:nk1-or+1)*mas;
```

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


% Este programa calcula las máscaras del esquema de subdivisión
% basado en Lagrange de orden n 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 conteniendo las máscaras
% numer vector conteniendo el numerador de las máscaras
% deno vector conteniendo el denominador de las máscaras


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 de la función
% para aplicar la reconstrucción PPH de orden n. Estos valores modificados
% se obtienen a partir de los valores iniciales de la función
% fm=f_modify(f)
% Variables de entrada:
% f vector con los valores discretos de la función f
% h espaciado uniforme entre los puntos del mallado
% Variables de salida:
% fm vector con los valores modificados de la función


n=length(f);

% Bucle para las sucesivas modificaciones

for i=1:n/2-1


% Diferencia dividida a la izquierda

   vl=f(n/2-i:n/2+i);
   dl=diff(vl,2*i);

% Diferencia dividida a la derecha

   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 calcula los coeficientes necesarios
% para computar los valores modificados de la función en la
% reconstrucción PPH de orden n
% c=comb(n)
% Variables de entrada:
% n longitud del vector de 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 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
```

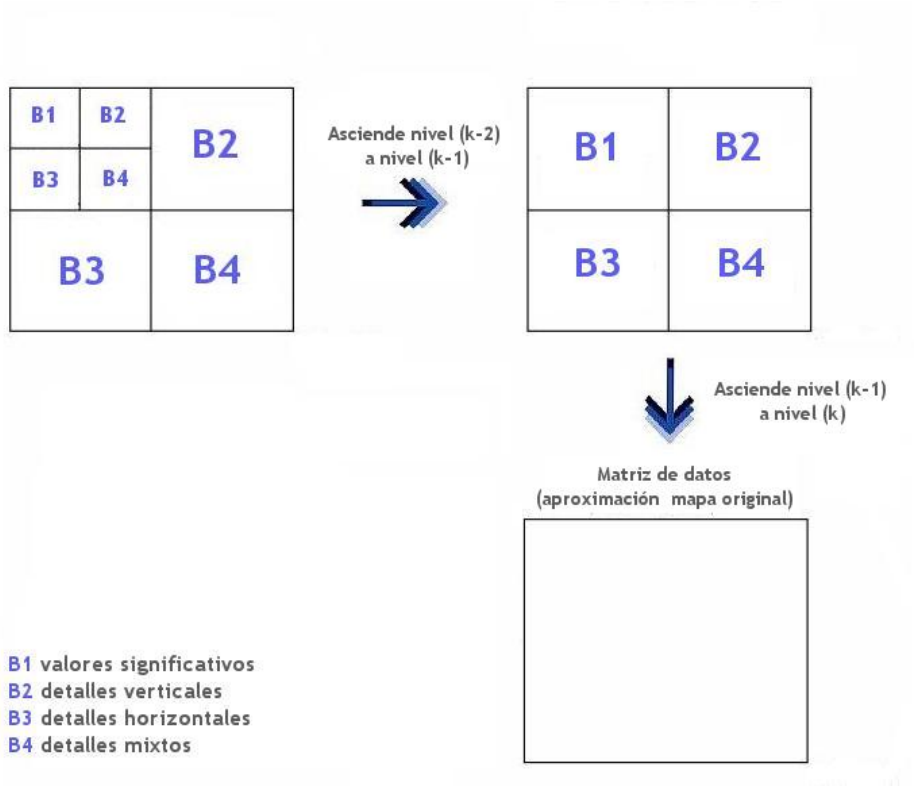

Figura 4.3: Proceso de decodificación.

```
function c=ascender(a,l,met,or)


% Función que relaiza proceso de decodificación del algoritmo
% de compresión, ascendiende k niveles de multirresolución.
% c=ascender(a,l,met,or)
% Variables de entrada:
% a matriz a la que le aplicamos el algorimo ascendente de
% multiresolución.
% l son los niveles de multiresolución que asciende.
% met es el método de reconstrucción utilizado.
% or orden de la reconstrucción.
% Variables de salida:
% c aproximación a los datos originales.
```

```
% Dimensiones de los datos.

[n,m]=size(a);

% Inicialización de las variables.

c=a;

% Dimensión de la matriz con resolución más baja.

nl=round((n-1)/(2^l));
ml=round((m-1)/(2^l));

if met=='lin'

% Bucle para los niveles de multirresolución.

    for k=l:-1:1


% Reordenamos la matriz haciendo el proceso inverso que en la codificación.

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

        c(1:nb,1:mb)=b;
        clear b;


% Algoritmo de multirresolución inverso para las columnas.

        for j=1:mb
            c(1:nb,j)=linealde(c(1:nb,j)',nb,or);
        end


% Algoritmo de multirresolución inverso para las filas.
```

```
        for i=1:nb
            c(i,1:mb)=linealde(c(i,1:mb),mb,or);
        end

        nl=2*nl; ml=2*ml;
    end


elseif met=='pph'

% Bucle para los niveles de multirresolución.

    for k=l:-1:1


% Reordenamos la matriz haciendo el proceso inverso que en la codificación.

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

        c(1:nb,1:mb)=b;
        clear b;


% Algoritmo de multirresolución inverso para las columnas.

        for j=1:mb
            c(1:nb,j)=pphde(c(1:nb,j)',nb,or);
        end


% Algoritmo de multirresolución inverso para las filas.

        for i=1:nb
            c(i,1:mb)=pphde(c(i,1:mb),mb,or);
        end

        nl=2*nl; ml=2*ml;
    end
end
```

```
function f=linealde(v,n,or)


% Función que sube un nivel de multirresolución utilizando una
% reconstrucción lineal centrada de or puntos basada en
% interpolación segmentaria de Lagrange.
% f=linealde(v,n,or);
% Variables de entrada:
% v vector que contiene los valores significativos de la escala
% inferior y los detalles para subir.
% n dimensión de v.
% or orden de la reconstrucción.
% Variables de salida:
% f vector de la escala superior.


f=zeros(size(v));
f(1:2:n)=v(1:2:n);
b=v(1:2:n);

% Cálculo de las máscaras necesarias.

masc=maskslr(or/2,or/2);
mas=zeros(or,or/2-1);

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

% Predicción de los valores en la frontera izquierda.

f(2:2:or-2)=v(2:2:or-2)+b(1:or)*mas;

% Predicción de los valores intermedios de f.

for i=or/2:(n+1)/2-or/2

    f(2*i)=v(2*i)+masc*b(i-or/2+1:i+or/2)';

end

% Predicción de los valores en la frontera derecha.

f(n-1:-2:n-or+3)=v(n-1:-2:n-or+3)+b((n+1)/2:-1:(n+1)/2-or+1)*mas;
```

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


% Este programa calcula las máscaras del esquema de subdivisión
% basado en Lagrange de orden n 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 conteniendo las máscaras
% numer vector conteniendo el numerador de las máscaras
% deno vector conteniendo el denominador de las máscaras


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 f=pphde(v,n,or)


% Programa que sube un nivel de multirresolución utilizando una
% reconstrucción pph centrada de or puntos.
% f=pphde(v,n,or)
% Variables de entrada:
% v vector que contiene los valores significativos de la escala
inferior
% y los detalles para subir.
% n dimensión de v.
% or orden de la reconstrucción.
% Variables de salida:
% f vector de la escala superior.


f=zeros(size(v));
f(1:2:n)=v(1:2:n);
b=v(1:2:n);

% Cálculo de las máscaras necesarias.

masc=maskslr(or/2,or/2);
mas=zeros(or,or/2-1);

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

% Predicción de los valores en la frontera izquierda.

f(2:2:or-2)=v(2:2:or-2)+b(1:or)*mas;

% Predicción de los valores intermedios.

for i=or/2:(n+1)/2-or/2
    b_modify=f_modify(b(i-or/2+1:i+or/2));
    f(2*i)=v(2*i)+masc*b_modify';
end

% Predicción de los valores en la frontera derecha.

f(n-1:-2:n-or+3)=v(n-1:-2:n-or+3)+b((n+1)/2:-1:(n+1)/2-or+1)*mas;
```

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


% Este programa calcula las máscaras del esquema de subdivisión
% basado en Lagrange de orden n 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 conteniendo las máscaras
% numer vector conteniendo el numerador de las máscaras
% deno vector conteniendo el denominador de las máscaras


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 de la función
% para aplicar la reconstrucción PPH de orden n. Estos valores modificados
% se obtienen a partir de los valores iniciales de la función
% fm=f_modify(f)
% Variables de entrada:
% f vector con los valores discretos de la función f
% h espaciado uniforme entre los puntos del mallado
% Variables de salida:
% fm vector con los valores modificados de la función


n=length(f);

   % Bucle para las sucesivas modificaciones.

for i=1:n/2-1


% Diferencia dividida a la izquierda.

   vl=f(n/2-i:n/2+i);
   dl=diff(vl,2*i);

% Diferencia dividida a la derecha.

   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 calcula los coeficientes necesarios
% para computar los valores modificados de la función en la
% reconstrucción PPH de orden n
% c=comb(n)
% Variables de entrada:
% n longitud del vector de 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 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
```

4.2. Zoom de datos bidimensionales

4.2.1. *Diagrama de Flujo*

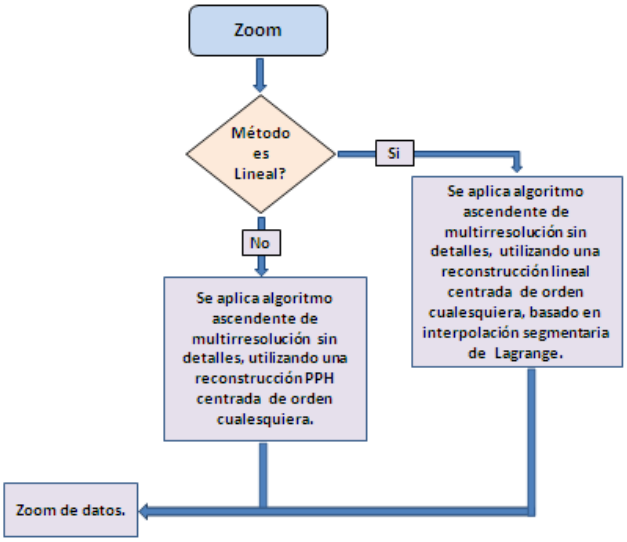

Figura 4.4: Diagrama de Flujo de Zoom de datos bidimensionales.

4.2.2. *Código*

```
function c =zoom_mapa(mapa,l,met,or)


% Función que realiza el proceso de ascenso en la pirámide de
% resolución (zoom) basándose en interpolación de Lagrange
% o en la reconstrucción PPH.
% c =zoom_mapa(mapa,l,met,or)
% Variables de entrada:
% mapa matriz original al que aplicamos zoom.
% l niveles de zoom.
% met método utilizado 'lin' ó 'PPH'.
% or orden de zoom.
% Variables de salida:
% c zoom de mapa original.


% Dimensión de los datos.

[n,m]=size(mapa);

% Inicialización de variables.

c=mapa;

if met=='lin'

% Bucle para los niveles de zoom.

    for k=1:l

        nf=2*n-1;
        nc=2*m-1;
        b=zeros(nf,nc);
        b(1:2:nf,1:2:nc)=c(1:n,1:m);


% Algoritmo de predicción para las columnas.

        for j=1:nc
            f=linealzoom(b(1:nf,j)',nf,or);
            b(2:2:nf-1,j)=f';
        end
        clear f;
```

```
% Algoritmo de predicción para las filas.
        for i=1:nf
            f=linealzoom(b(i,1:nc),nc,or);
            b(i,2:2:nc-1)=f;
        end
        clear f;

        n=nf;
        m=nc;
        c=b;
    end

elseif met=='pph'
% Bucle para los niveles de zoom.
    for k=1:l

        nf=2*n-1; nc=2*m-1;
        b=zeros(nf,nc);
        b(1:2:nf,1:2:nc)=c(1:n,1:m);
% Algoritmo de predicción para las columnas.
        for j=1:2:nc
            f=pphezoom(b(1:nf,j)',nf,or);
            b(2:2:nf-1,j)=f';
        end
        clear f;
% Algoritmo de predicción para las filas.
        for i=1:nf
            f=pphezoom(b(i,1:nc),nc,or);
            b(i,2:2:nc-1)=f;
        end
        clear f;

        n=nf;
        m=nc;
        c=b;
    end
end
```

```
function f=linealzoom(v,n,or)


% Programa que sube un nivel de multirresolución utilizando una
% reconstrucción lineal centrada de or puntos basada en
% interpolación segmentaria de Lagrange.
% f=linealzoom(v,n,or)
% Variables de entrada:
% v vector de dimensión n con valores significativos de la escala
% inferior.
% n dimensión escala superior.
% or orden de la reconstrucción.
% Variables de salida:
% f predicción entre las escalas.
 % Valores significativos de la escala inferior.

f1=v(1:2:n);

% Longitud del vector f1.

nk1=(n+1)/2;

% Cálculo de las máscaras necesarias.

masc=maskslr(or/2,or/2);
mas=zeros(or,or/2-1);


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

% Predicción de los valores en la frontera izquierda.

f(1:or/2-1)=f1(1:or)*mas;

% Predicción de los valores intermedios de f.

for i=or/2:nk1-or/2
    f(i)=masc*f1(i-or/2+1:i+or/2)';
end

% Predicción de los valores en la frontera derecha.

f(nk1-1:-1:nk1-or/2+1)=f1(nk1:-1:nk1-or+1)*mas;
```

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


% Este programa calcula las máscaras del esquema de subdivisión
% basado en Lagrange de orden n 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 conteniendo las máscaras
% numer vector conteniendo el numerador de las máscaras
% deno vector conteniendo el denominador de las máscaras


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 f=pphezoom(v,n,or)


% Programa que sube un nivel de multirresolución utilizando
% una reconstrucción pph centrada de or puntos.
% f=pphezoom(v,n,or)
% Variables de entrada:
% v vector de dimensión n con los valores significativos
% de la escala inferior.
% n dimensión escala superior.
% or orden de la reconstrucción.
% Variables de salida:
% f predicción entre las escalas.

% Valores significativos de la escala inferior.

f1=v(1:2:n);

% Longitud del vector f1.

nk1=(n+1)/2;

% Cálculo de las máscaras necesarias.

masc=maskslr(or/2,or/2);
mas=zeros(or,or/2-1);

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

% Predicción de los valores en la frontera izquierda.

f(1:or/2-1)=f1(1:or)*mas;

% Predicción de los valores intermedios.

for i=or/2:nk1-or/2
    f1_modify=f_modify(f1(i-or/2+1:i+or/2));
    f(i)=masc*f1_modify';
end

% Predicción de los valores en la frontera derecha.

f(nk1-1:-1:nk1-or/2+1)=f1(nk1:-1:nk1-or+1)*mas;
```

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


% Este programa calcula las máscaras del esquema de subdivisión
% basado en Lagrange de orden n 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 conteniendo las máscaras
% numer vector conteniendo el numerador de las máscaras
% deno vector conteniendo el denominador de las máscaras


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 de la función
% para aplicar la reconstrucción PPH de orden n. Estos valores modificados
% se obtienen a partir de los valores iniciales de la función
% fm=f_modify(f)
% Variables de entrada:
% f vector con los valores discretos de la función f
% h espaciado uniforme entre los puntos del mallado
% Variables de salida:
% fm vector con los valores modificados de la función

n=length(f);

% Bucle para las sucesivas modificaciones.
for i=1:n/2-1

% Diferencia dividida a la izquierda.
   vl=f(n/2-i:n/2+i);
   dl=diff(vl,2*i);
% Diferencia dividida a la derecha.
   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 calcula los coeficientes necesarios
% para computar los valores modificados de la función en la
% reconstrucción PPH de orden n
% c=comb(n)
% Variables de entrada:
% n longitud del vector de 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 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
```

Capítulo 5

Interfaz gráfica

Aprovechando la versatilidad de Matlab para crear entornos gráficos, se ha empleado el entorno de desarrollo GUIDE *(Graphical User Interface Development Environment)* para poder probar los distintos algoritmos implementados y visualizar de una manera gráfica los parámetros de éstos.

5.1. Ejecución interfaz gráfica

Para ejecutar la interfaz gráfica, tenemos que introducir en el intérprete de comandos de matlab

$$>> UPCT$$

Antes de ejecutar dicha instrucción tenemos que situarnos en el directorio donde está contenida la GUI.

5.2. Documentación de la interfaz gráfica

Ejecutada la interfaz gráfica se nos abre una ventana como la que se ilustra en la Figura 5.1

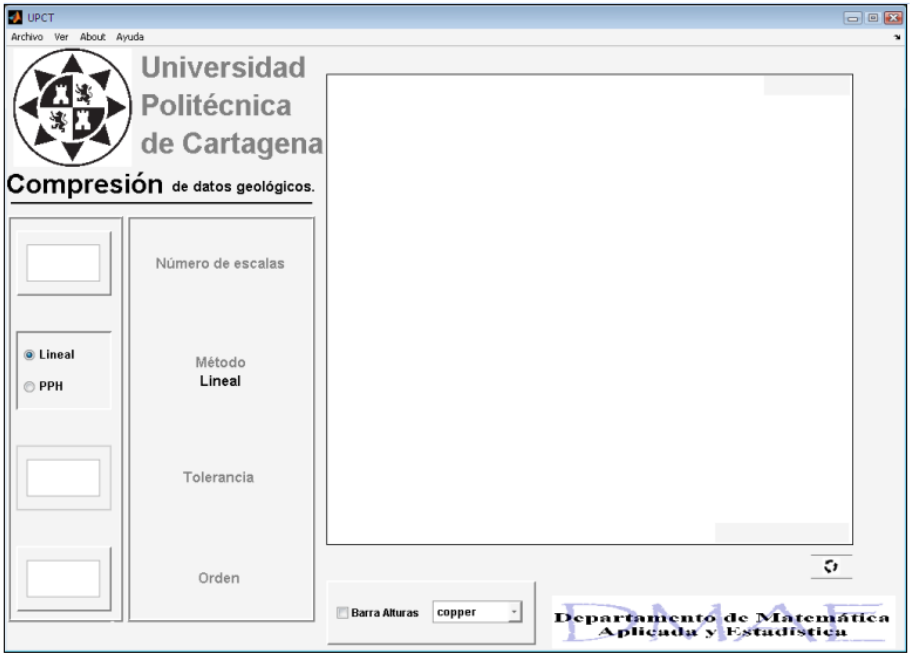

Figura 5.1: Interfaz gráfica principal de usuario.

La aplicación cuenta con un menú (Figura 5.2) que consiste en una lista de opciones que puede desplegarse para mostrar las distintas funciones y acceder así a las diferentes herramientas.

Figura 5.2: Menú de usuario.

A continuación se detallan minucisamente todos los submenús contenidos en el menú de la interfaz gráfica:

- **Archivo**

  - **Abrir**
    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 en el que se especifican los diferentes subdirectorios y ficheros, y abrir un archivo.

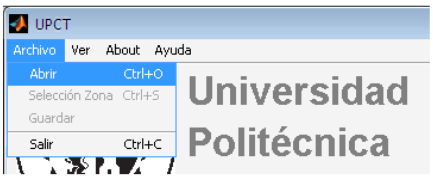

Figura 5.3: Submenú Abrir.

La opción **Abrir** nos lleva a una nueva ventana como la que se ilustra en la Figura 5.4, esta ventana es en la que se va a mostrar el contenido de los archivos MAT [1]

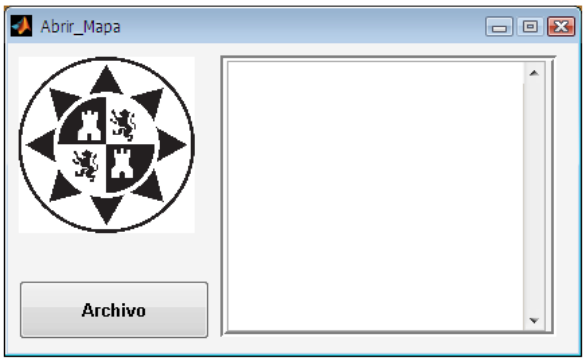

Figura 5.4: Interfaz gráfica Abrir Mapa.

[1] En nuestro caso, los archivos MAT van a contener mapa/s, un mapa está representado por una matriz de datos.

Si se pulsa el botón **Archivo** de la interfaz gráfica Abrir Mapa nos va a llevar a un nuevo cuadro diálogo (Ver Figura 5.5), en el que se pueden seleccionar todos los archivos de extensión .mat situados en los discos locales.

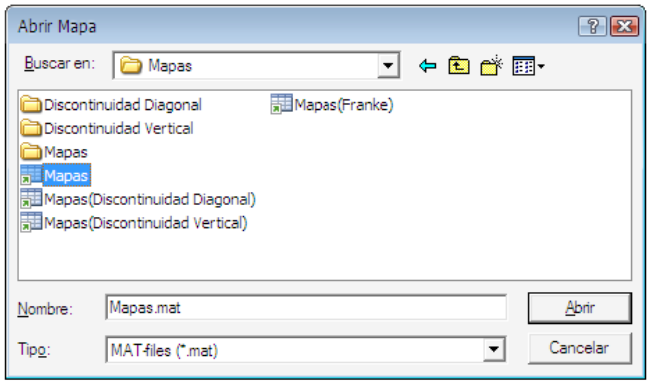

Figura 5.5: Cuadro diálgo de apertura de archivos MAT.

Una vez abierto el archivo MAT, la interfaz gráfica Abrir Mapa, nos queda como se ilustra en la Figura 5.6.

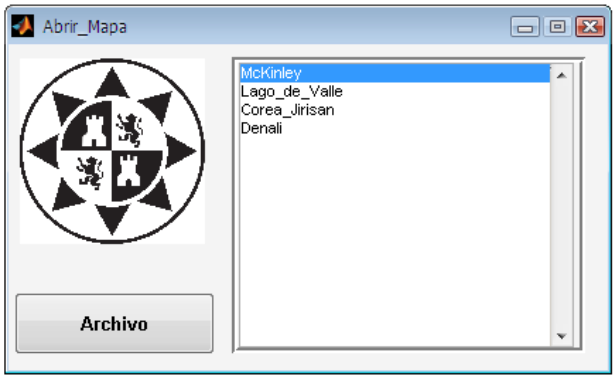

Figura 5.6: Interfaz con contenido de archivos MAT.

Finalmente, seleccionando el nombre, el mapa es abierto en la interfaz principal de usuario como se puede ver en la Figura 5.7.

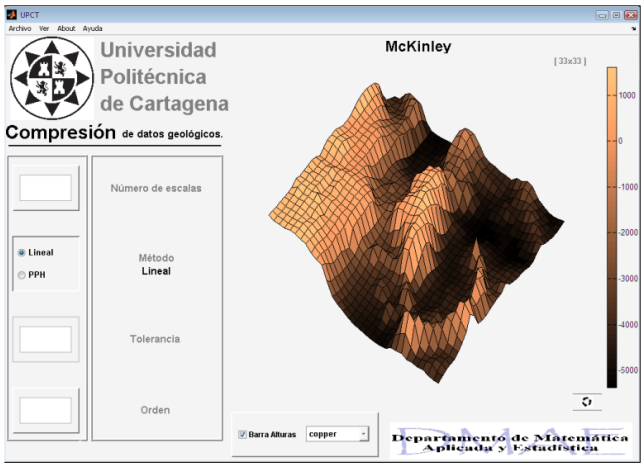

Figura 5.7: Mapa cargado en interfaz principal de usuario.

- **Selección Zona**
  Permite acotar una zona seleccionada del mapa de datos, inicialmente esta opción se encuentra desactivada, su estado permanece pasivo mientras no se haya abierto algún mapa con anterioridad.

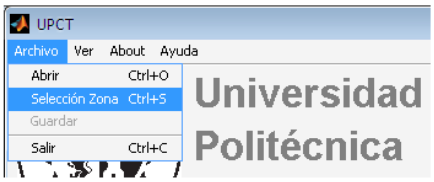

Figura 5.8: Submenú Seleccion Zona.

Al pulsar sobre el submenú **Selección Zona** se nos abrirá una nueva aplicación (Figura 5.9)

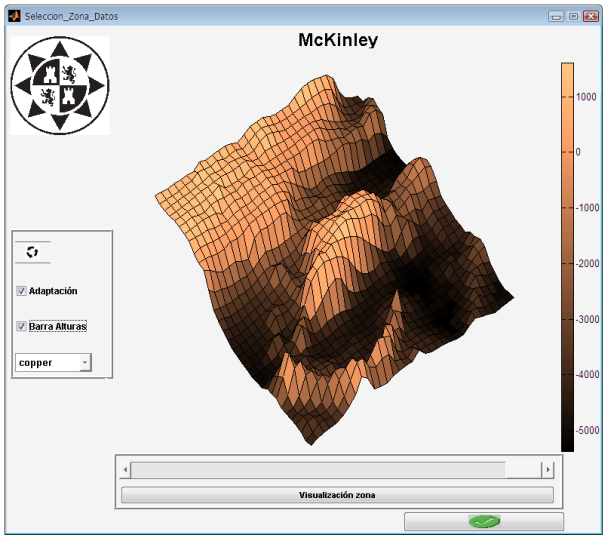

Figura 5.9: Interfaz gráfica Selección Zona Datos.

Para la selección de una zona, se debe pulsar sobre un punto en el mapa, céntrico a la zona que deseamos acotar Figura 5.10.

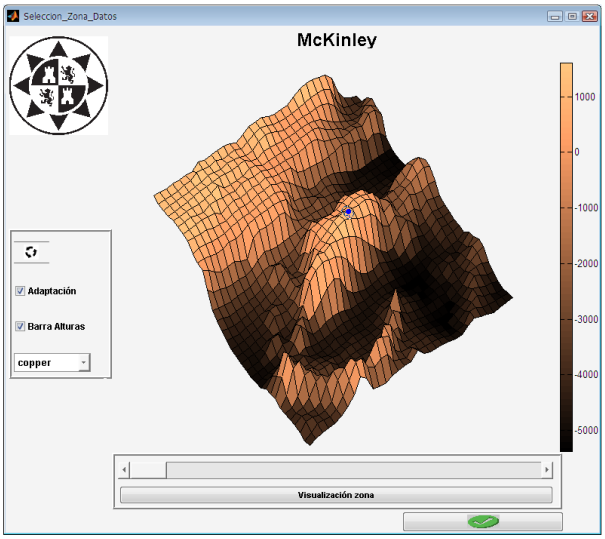

Figura 5.10: Selección de punto.

La posición de los puntos seleccionados son mostrados mediante un indicador de coordenadas como nos ilustra la Figura 5.11.

Figura 5.11: Indicador de coordenadas.

Seleccionado el punto, la zona a acotar se gradua con la barra deslizadora obteniendo un aumento o disminución de la zona seleccionada, esta zona a acotar puede ser visualizada pulsando el botón **visualización de datos**.

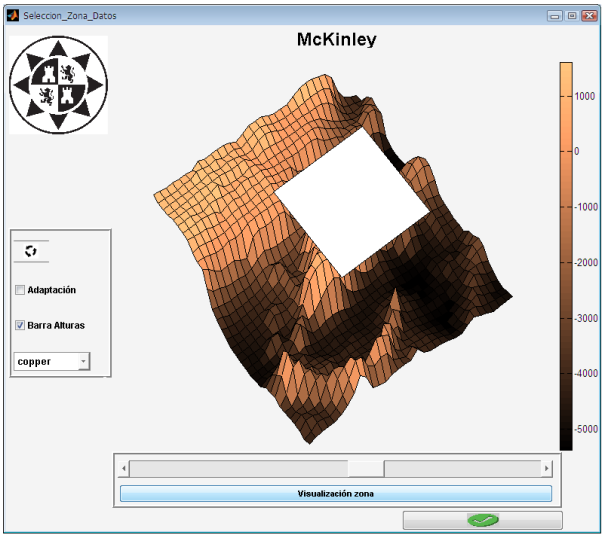

Figura 5.12: Visualización de zona a acotar.

En el caso en el que la zona seleccionada no sea la esperada, se debe seleccionar otro punto del mapa y realizar los pasos ante-

riormente detallados. Si por el contrario la zona seleccionada es la correcta se pulsa el botón **aplicar**, devolviendo a la interfaz principal de usuario la zona de mapa seleccionada (Figura 5.13).

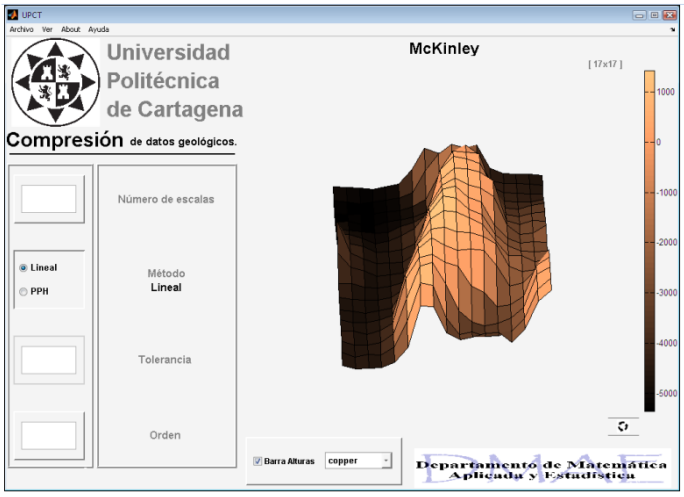

Figura 5.13: Zona acotada del mapa en interfaz principal.

Tener en cuenta, que la interfaz de Selección de Zona de datos dispone de un panel de configuración (Ver Figura 5.14), el cual permite realizar acciones sobre los mapas como cambiar el color, rotar, colocar una barra para visualización de alturas o adaptar la zona seleccionada a una dimensión de $2^k - 1$ puntos, dimensión imprescindible para la compresión de mapas de datos.

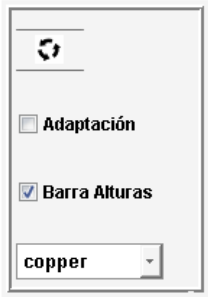

Figura 5.14: Panel configuración.

- **Guardar**

  Inicialmente esta opción del menú Archivo se encuentra desactivada, su estado se activa, sólo si se ha acotado una zona del mapa original.

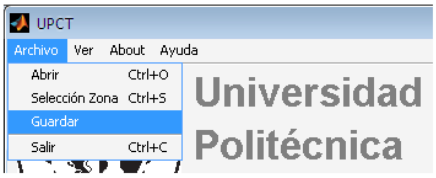

Figura 5.15: Submenú guardar.

La opción **guardar** almacena el mapa acotado en un archivo MAT y de manera transparente al usuario, guarda en el directorio seleccionado las simulaciones provocadas por el mapa acotado en archivos de extensión .rtf.

Tener en cuenta que una vez guardado, todas las simulaciones posteriores se guardarán automáticamente en el mismo archivo de simulación.

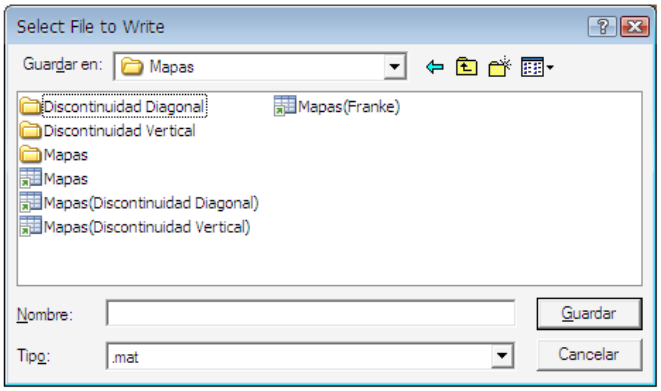

Figura 5.16: Cuadro diálogo guardar.

Los nombres de los archivos MAT podrán ser elegidos aleatoriamente por el usuario, mientras que los nombres de los mapas y archivos de simulación son creados por la aplicación.

Sintaxis que se toma:

⋆ Nombre de los mapas almacenados.
nombre_mapa|*dimensión*|*X_posición*[2]|*Y_posición*

⋆ Nombre archivos de simulación.
nombre_mapa|*dimensión*|*X_posición*|*Y_posición_Compresión.rtf*
nombre_mapa|*dimensión*|*X_posición*|*Y_posición_Zoom.rtf*

- **Salir**
  Cierra la aplicación por completo. Si esta opción es seleccionada te pedirá que confirmes tu elección para evitar salidas accidentales. También te advertirá en el caso en el que un mapa acotado no haya sido almacenado con anterioridad.

▪ **Ver**

- **Compresión**

▷ **Tolerancia**

Figura 5.17: Cuadro diálogo guardar.

Cambia la aplicación para que pueda realizar compresión del mapa de datos, mediante la introducción de una tolerancia

[2](X_posición, Y_posición) posición del centro del mapa.

de truncamiento de valores como se ilustra en la Figura 5.18.

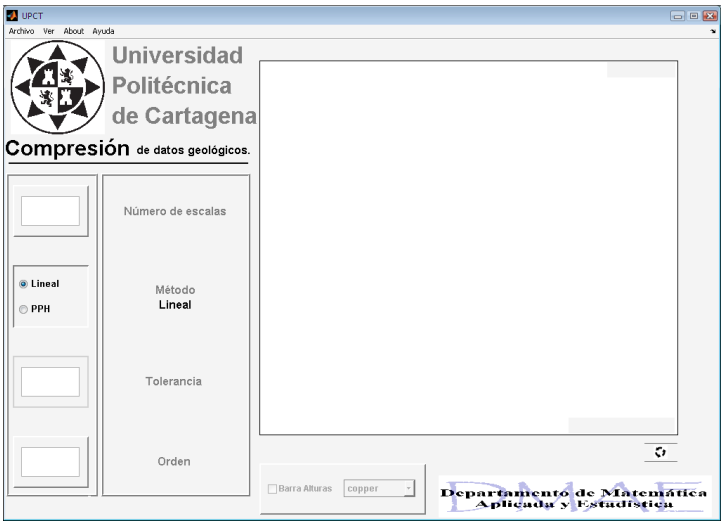

Figura 5.18: Interfaz principal de usuario.

▷ Porcentaje de Compresión

Figura 5.19: Cuadro diálogo guardar.

Cambia la aplicación para que pueda realizar compresión del mapa de datos, mediante un porcentaje de compresión como se ilustra en la Figura 5.20.

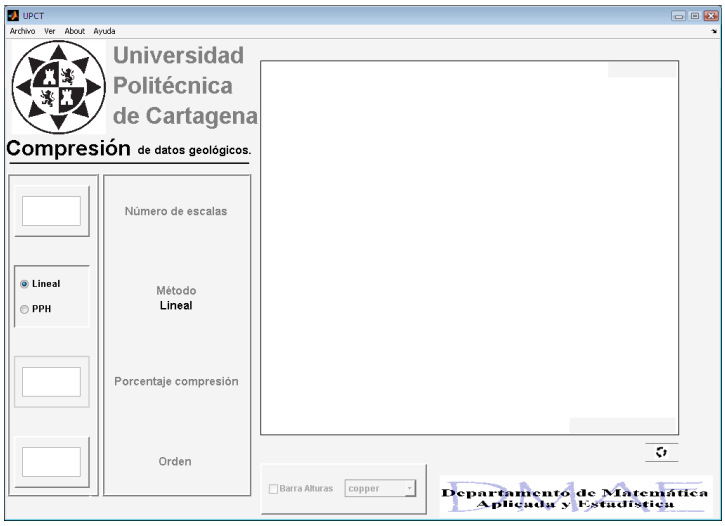

Figura 5.20: Interfaz principal de usuario.

- **Zoom**

Figura 5.21: Submenú zoom.

Cambia la aplicación para que pueda realizar zoom del mapa de datos como se ilustra en la Figura 5.22.

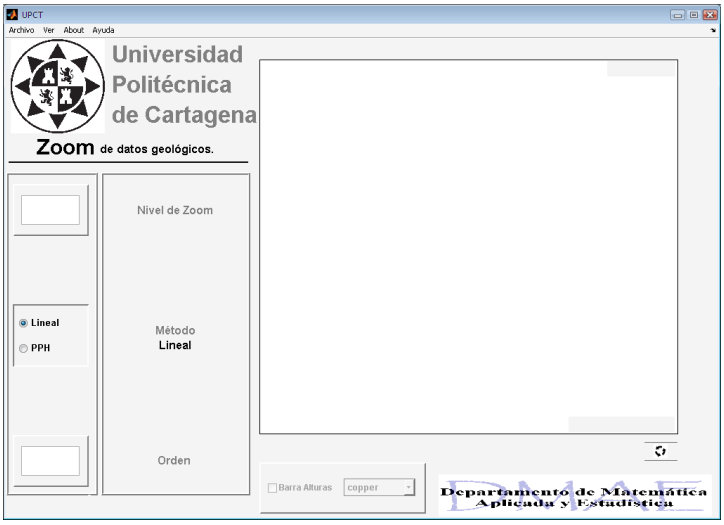

Figura 5.22: Interfaz principal de usuario.

- **Resultado**

Figura 5.23: Submenú resultado.

Permite la comparación visual del mapa original, antes de ninguna simulación, con el obtenido una vez realizada la compresión o el zoom del mapa. También muestra los valores obtenidos según el procedimiento escogido para el tratamiento de los datos.(Ver Figura 5.24).

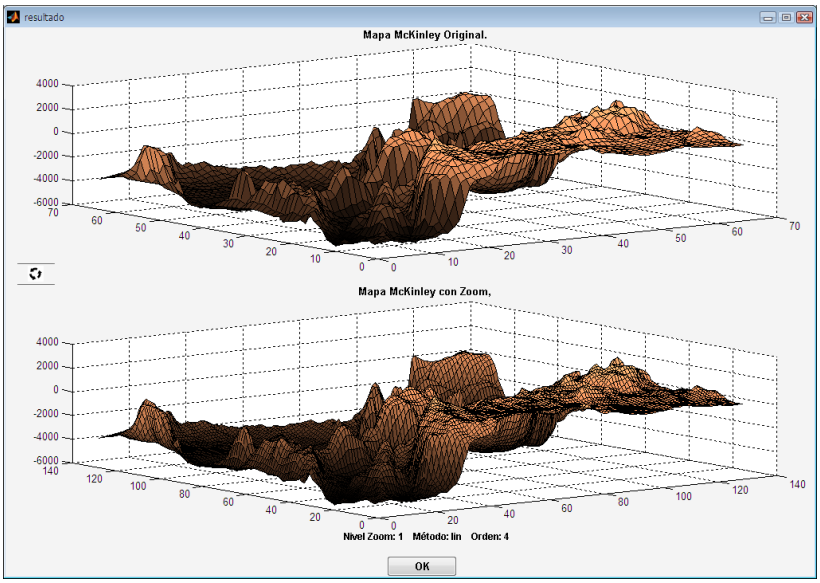

Figura 5.24: Interfaz gráfica Resultado.

- **About**
  Nos aparece una ventana emergente mostrando la información más relevante de la aplicación, como el autor de la misma.

- **Ayuda**
  Muestra ayuda sobre los distintos menús de la interfaz gráfica.

La interfaz principal dispone de una panel de configuración que nos permite realizar sobre los mapas tanto la modificación del color, como la visualización de una barra de alturas.(Ver Figura 5.25).

Figura 5.25: Panel Configuración.

También cuenta con un botón rotar que nos permite girar los mapas para su visualización en distintas posiciones como se ilustra en la Figura 5.26.

Figura 5.26: Botón rotar.

Cuando se realizan las distintas simulaciones de Zoom o Compresión aparece un panel mostrando los valores obtenidos, en el cual se podrán introducir los comentarios oportunos sobre los resultados. (Ver Figura 5.25).

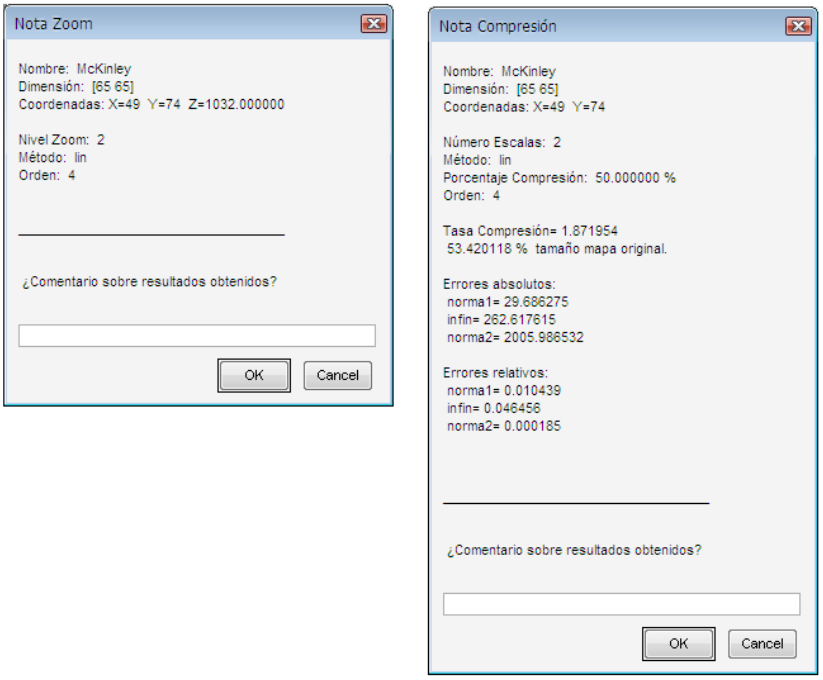

Figura 5.27: Panel de valores obtenidos.

Tener en cuenta que las simulaciones realizadas sobre los mapas son almacenadas de manera transparente al usuario en archivos de extensión .rtf. La sintaxis de los archivos es:

nombre_mapa_Compresión.rtf
nombre_mapa_Zoom.rtf

Los archivos siguen esta sintaxis en el caso en el que el mapa no haya sido acotado con anterioridad, en caso contrario la sintaxis que toma es la siguiente:

nombre_mapa|*dimensión*|*X_posición*|*Y_posición_Compresión.rtf*
nombre_mapa|*dimensión*|*X_posición*|*Y_posición_Zoom.rtf*

Capítulo 6

Experimentos numéricos

En esta sección vamos a desarrollar algunos experimentos con mapas de diferentes tipos. En la Figuras 6.1, 6.2 vemos los mapas elegidos. El primero de ellos, el mapa Mckinley, es un mapa correspondiente a una zona muy montañosa, donde encontramos muchas zonas abruptas. El segundo mapa, mapa Franke, corresponde en realidad a la gráfica de una función suave que no presenta discontinuidades. En el tercero, mapa Ondulado, encontramos una discontinuidad orientada en la dirección diagonal, y representa una zona marina. En el cuarto, mapa Sobreelevación, representamos una región con una discontinuidad orientada verticalmente.

En primer desarrollamos experimentos correspondientes a la compresión de estos mapas. Los resultados los hemos recogido en las tablas que aparecen a continuación. En general nuestras observaciones son las siguientes. Podemos ver como en en los mapas correspondientes a zonas abruptas, como puede ser el mapa Mckinley, los métodos funcionan mejor si se elige un orden de aproximación bajo. Esto puede observarse claramente en las Tablas 6.1, 6.2. También puede observarse que el método no lineal PPH obtiene una ligera ganancia (notar que para diferenciar PPH del lineal tenemos que hablar de orden mayor o igual a 4). Por el contrario en los mapas relativos a zonas suaves, como el mapa Franke, vemos que sí obtenemos mejores resultados al aumentar el orden de aproximación. Además en estos casos, el usar la no linealidad, no mejora los resultados. Estos comentarios vienen motivados por la observación de las Tablas 6.3, 6.4. Dichas observaciones se repiten con los demás experimentos. Hemos incluido las tablas correspondientes, Tablas 6.5, 6.6, 6.7, 6.8 para su análisis.

En general, la elección debería ser dependiente del mapa utilizado. Es decir, si el mapa es suave entonces una buena elección es un método lineal

y orden alto. Si por el contrario el mapa es muy abrupto, entonces la mejor elección según nuestros experimentos sería un método no lineal con orden bajo, o bien utilizar simplemente el esquema de 2 puntos en casos extremos. Si el caso es como puede fácilmente ocurrir en la práctica que tenemos mapas mixtos y que nos queremos decidir por un método, entonces nuestra elección sería tomar el método PPH de orden 4. Así obtendremos resultados satisfactorios tanto en mapas suaves como en mapas con singularidades notables.

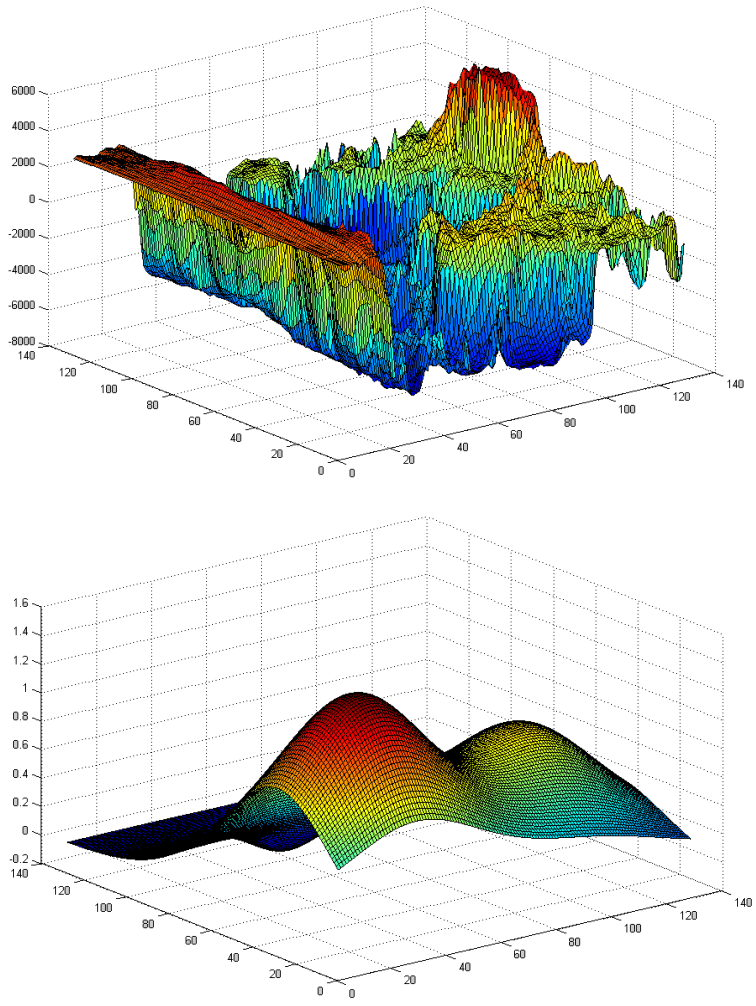

Figura 6.1: Mapa McKinley, Mapa Franke.

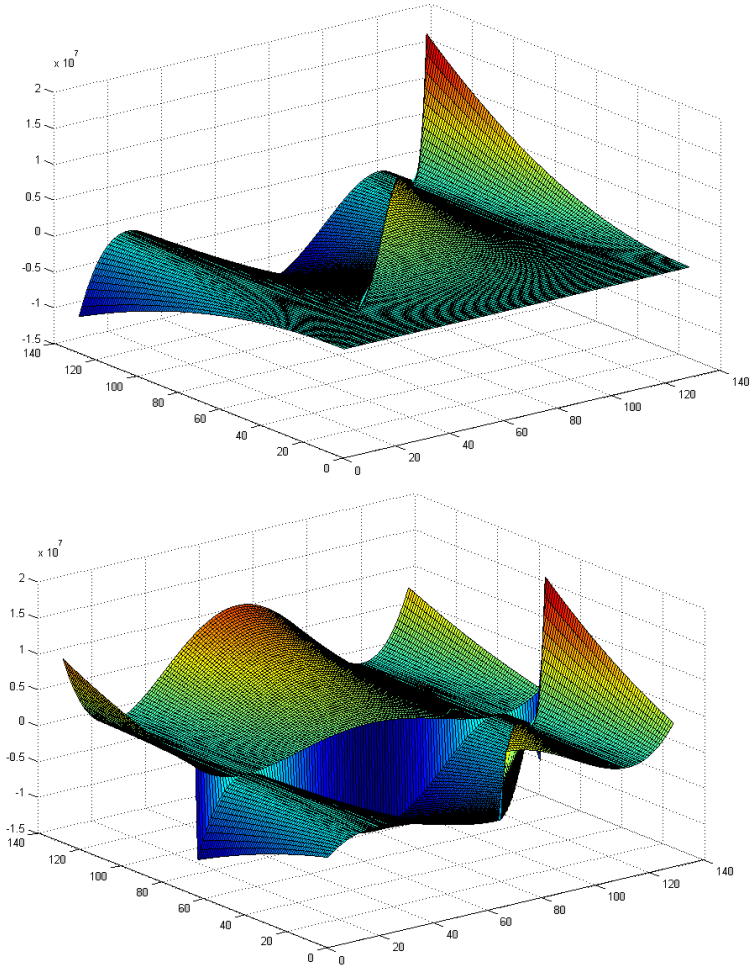

Figura 6.2: Mapa Ondulado, Mapa Sobreelevación.

Por completitud vamos a incluir también los resultados gráficos de las reconstrucciones de los mapas después del proceso de descompresión que pueden verse en las Figuras 6.3, 6.5, 6.7, 6.9, así como algunos zooms de zonas acotadas concretas de dichos mapas que pueden observarse en las Figuras 6.4, 6.6, 6.8, 6.10.

Método	**% Compresión**	**Orden**	***Error relativo 1***	***Error infinito***	***Error relativo 2***
Lineal	20	2	0,000786	0,004351	0,000002
		4	0,000922	0,005017	0,000002
		6	0,001129	0,006152	0,000005
		8	0,001473	0,12760	0,000005
	40	2	0,004317	0,014358	0,000033
		4	0,004886	0,017782	0,000040
		6	0,005924	0,019172	0,000056
		8	0,007197	0,041065	0,000088
	60	2	0,014988	0,038298	0,000306
		4	0,016102	0,057250	0,000356
		6	0,018284	0,068282	0,000452
		8	0,024282	0,152765	0,000885
	80	2	0,042410	0,111236	0,002250
		4	0,046632	0,109977	0,002738
		6	0,055703	0,171095	0,003978
		8	0,074679	0,304169	0,007537

Tabla 6.1: **Mapa McKinley**, Número de escalas 4, Dimensión 129x129.

Método	**% Compresión**	**Orden**	***Error relativo 1***	***Error infinito***	***Error relativo 2***
PPH	20	2	0,000786	0,004351	0,000002
		4	0,000885	0,004598	0,000002
		6	0,000969	0,005344	0,000002
		8	0,001296	0,009248	0,000004
	40	2	0,004317	0,014358	0,000033
		4	0,004511	0,015944	0,000035
		6	0,005193	0,018933	0,000045
		8	0,006590	0,036784	0,000074
	60	2	0,014988	0,038298	0,000306
		4	0,015292	0,037674	0,000318
		6	0,016531	0,067831	0,000379
		8	0,020511	0,115892	0,000630
	80	2	0,042410	0,111236	0,002250
		4	0,045401	0,107471	0,002551
		6	0,051678	0,170947	0,003506
		8	0,066360	0,280820	0,006043

Tabla 6.2: **Mapa McKinley**, Número de escalas 4, Dimensión 129x129.

Método	**% Compresión**	**Orden**	***Error relativo 1***	***Error infinito***	***Error relativo 2***
Lineal	90	2	0,000801	0,001168	0,000001
		4	0,000013	0,000027	0
		6	0,000001	0,000002	0
		8	0	0	0
	94	2	0,001293	0,002002	0,000002
		4	0,000032	0,000055	0
		6	0,000002	0,000004	0
		8	0	0,000002	0
	98	2	0,003566	0,005812	0,000012
		4	0,000254	0,000418	0
		6	0,000054	0,000089	0
		8	0,000035	0,000146	0
	100	2	0,032881	0,084607	0,001604
		4	0,011771	0,037045	0,000181
		6	0,013788	0,032596	0,000239
		8	0,019986	0,083903	0,000841

Tabla 6.3: **Mapa Franke**, Número de escalas 4, Dimensión 129x129.

Método	**% Compresión**	**Orden**	***Error relativo 1***	***Error infinito***	***Error relativo 2***
PPH	90	2	0,000801	0,001168	0,000001
		4	0,000036	0,000083	0
		6	0,000007	0,000016	0
		8	0,000003	0,000014	0
	94	2	0,001293	0,002002	0,000002
		4	0,000085	0,000155	0
		6	0,000020	0,000049	0
		8	0,000003	0,000014	0
	98	2	0,003566	0,005812	0,000012
		4	0,000580	0,001190	0
		6	0,000214	0,000397	0
		8	0,000198	0,000736	0
	100	2	0,032881	0,084607	0,001604
		4	0,016961	0,046472	0,000391
		6	0,018754	0,032603	0,000371
		8	0,023123	0,084970	0,000892

Tabla 6.4: **Mapa Franke**, Número de escalas 4, Dimensión 129x129.

Método	**% Compresión**	**Orden**	***Error relativo 1***	***Error infinito***	***Error relativo 2***
Lineal	90	2	0,002980	0,001822	0,000006
		4	0,000119	0,000241	0
		6	0,000804	0,003314	0,000001
		8	0,004101	0,014280	0,000022
	94	2	0,007335	0,004597	0,000036
		4	0,006824	0,009571	0,000046
		6	0,010969	0,017489	0,000119
		8	0,029710	0,047165	0,000782
	98	2	0,088480	0,122541	0,007511
		4	0,060118	0,193559	0,007258
		6	0,087867	0,212482	0,011429
		8	0,237013	0,314855	0,062447
	100	2	0,253207	0,902944	0,112045
		4	0,154156	0,885544	0,080504
		6	0,205083	0,863847	0,097298
		8	0,500879	1,457953	0,400767

Tabla 6.5: **Mapa Ondulado**, Número de escalas 4, Dimensión 129x129, Discontinuidad diagonal.

Método	**% Compresión**	**Orden**	***Error relativo 1***	***Error infinito***	***Error relativo 2***
PPH	90	2	0,002980	0,001822	0,000006
		4	0,000058	0,000064	0
		6	0,000011	0,000023	0
		8	0,000016	0,000026	0
	94	2	0,007335	0,004597	0,000036
		4	0,000617	0,000498	0
		6	0,000471	0,001345	0
		8	0,001616	0,007230	0,000003
	98	2	0,088480	0,122541	0,007511
		4	0,050734	0,147120	0,004514
		6	0,066635	0,176442	0,006805
		8	0,187081	0,299405	0,043774
	100	2	0,253207	0,902944	0,112045
		4	0,158421	0,885939	0,080526
		6	0,196254	0,863809	0,095785
		8	0,481570	1,456393	0,389570

Tabla 6.6: **Mapa Ondulado**, Número de escalas 4, Dimensión 129x129, Discontinuidad diagonal.

Método	**% Compresión**	**Orden**	***Error relativo 1***	***Error infinito***	***Error relativo 2***
Lineal	90	2	0,002526	0,002694	0,000006
		4	0,000028	0,000034	0
		6	0,000023	0,000084	0
		8	0,001227	0,003046	0,000003
	94	2	0,005206	0,004524	0,000024
		4	0,000773	0,002368	0,000001
		6	0,007480	0,013872	0,000084
		8	0,014327	0,049542	0,000382
	98	2	0,039255	0,055849	0,001505
		4	0,045872	0,140047	0,004235
		6	0,069607	0,197870	0,008601
		8	0,144332	0,310157	0,029855
	100	2	0,272597	0,791525	0,203312
		4	0,206486	0,809098	0,189107
		6	0,250517	0,811224	0,214131
		8	0,441456	1,085955	0,404876

Tabla 6.7: **Mapa Sobreelevación**, Número de escalas 4, Dimensión 129x129, Discontinuidad vertical.

Método	**% Compresión**	**Orden**	***Error relativo 1***	***Error infinito***	***Error relativo 2***
PPH	90	2	0,002526	0,002694	0,000006
		4	0,000035	0,000060	0
		6	0,000008	0,000018	0
		8	0,000009	0,000023	0
	94	2	0,005206	0,004524	0,000024
		4	0,000121	0,000193	0
		6	0,000026	0,000066	0
		8	0,000044	0,000148	02
	98	2	0,039255	0,055849	0,001505
		4	0,016546	0,015887	0,000285
		6	0,015842	0,054205	0,000477
		8	0,081572	0,228527	0,0,012114
	100	2	0,272597	0,791525	0,203312
		4	0,195052	0,791525	0,188927
		6	0,228882	0,791525	0,213261
		8	0,426303	1,059509	0,399380

Tabla 6.8: **Mapa Sobreelevación**, Número de escalas 4, Dimensión 129x129, Discontinuidad vertical.

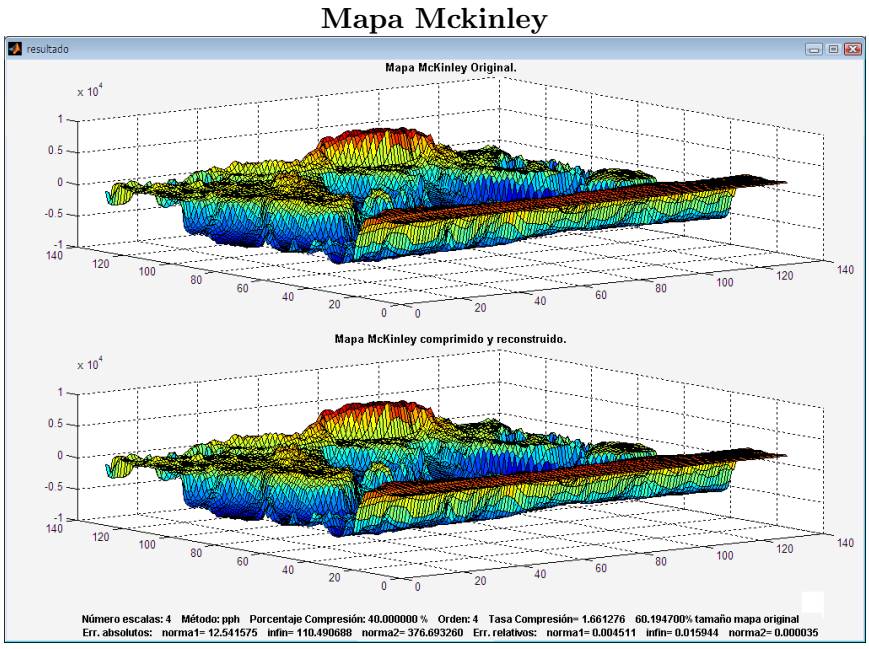

Figura 6.3: Método PPH, Número escalas 4, Orden 4, 40 % de compresión.

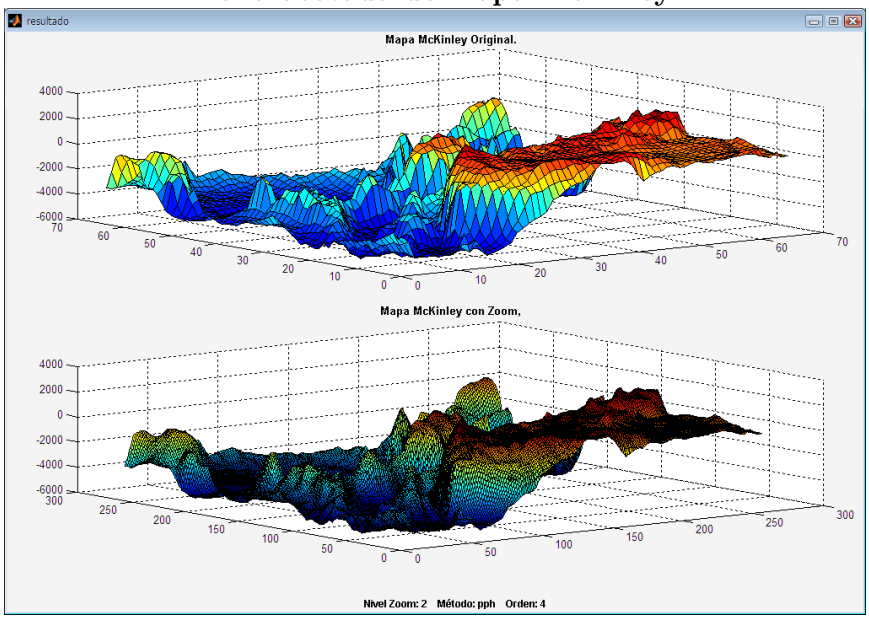

Figura 6.4: Método PPH, Nivel de Zoom 2, Orden 4.

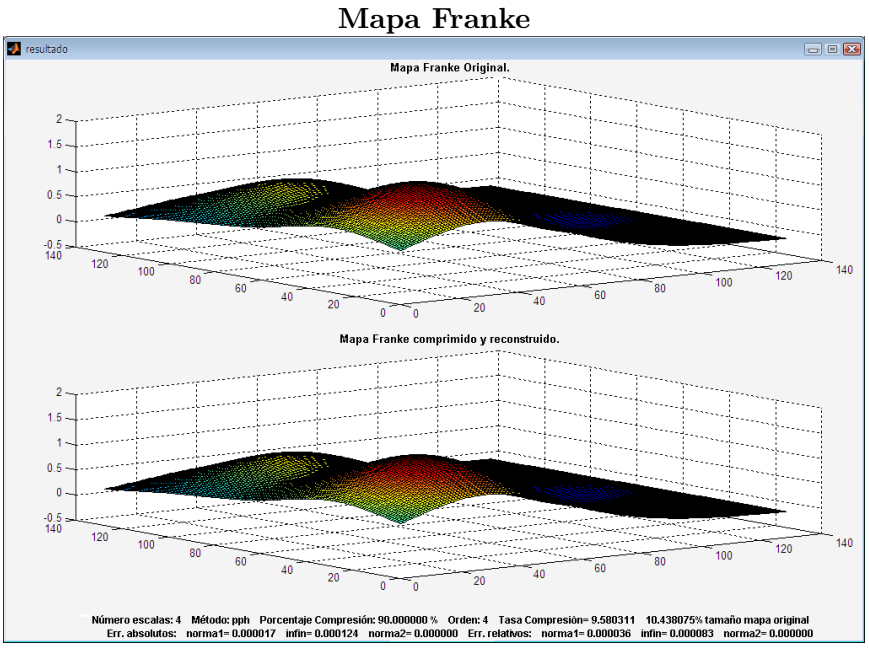

Figura 6.5: Método PPH, Número escalas 4, Orden 4, 90 % de compresión.

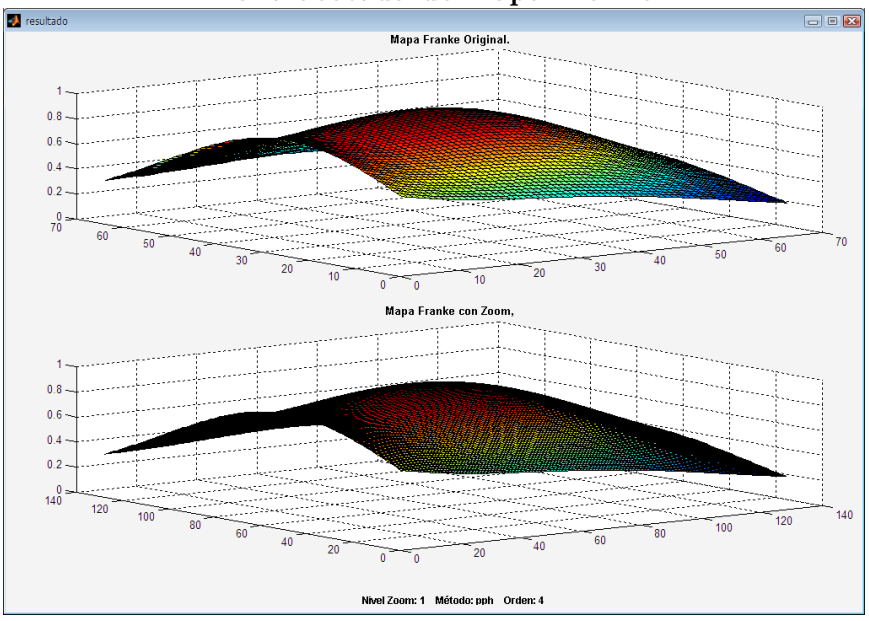

Figura 6.6: Método PPH, Nivel de Zoom 1, Orden 4.

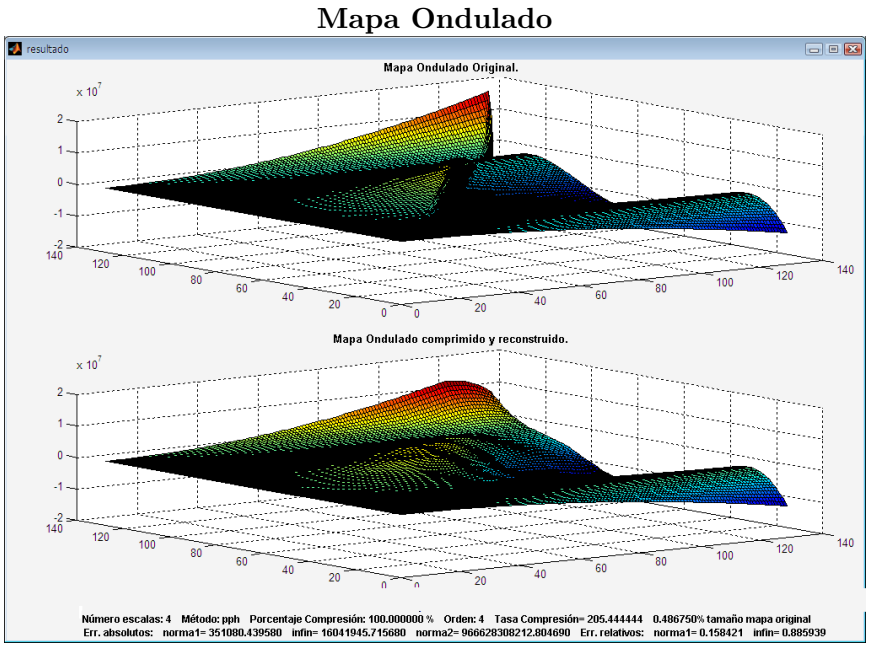

Figura 6.7: Método PPH, Número escalas 4, Orden 4, 100 % de compresión.

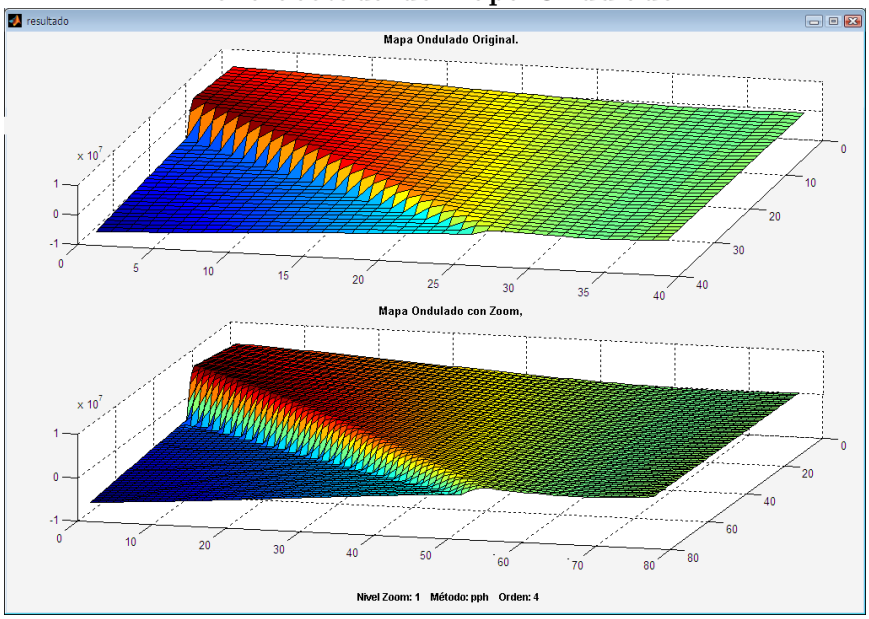

Figura 6.8: Método PPH, Nivel de Zoom 1, Orden 4.

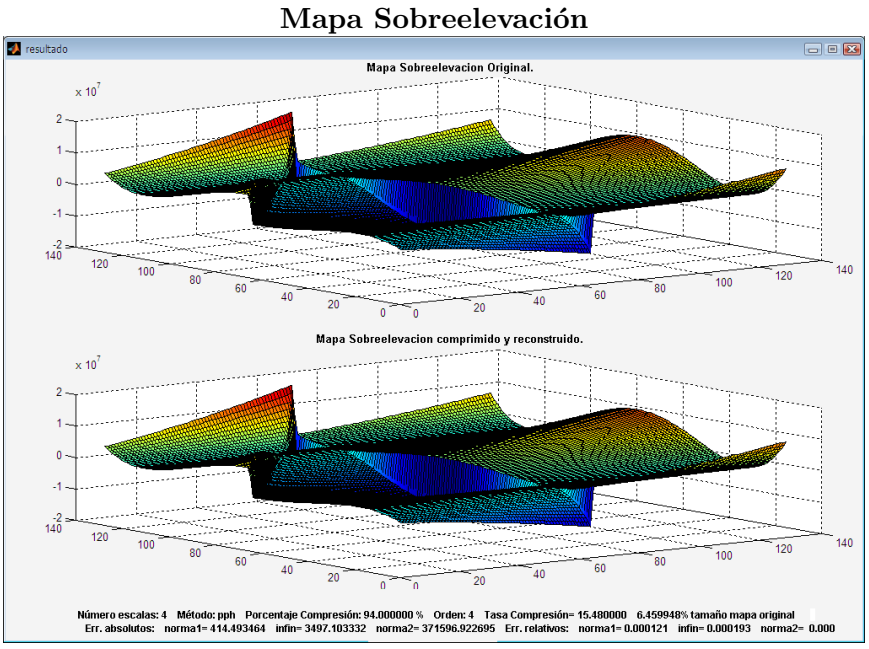

Figura 6.9: Método PPH, Número escalas 4, Orden 4, 94 % de compresión.

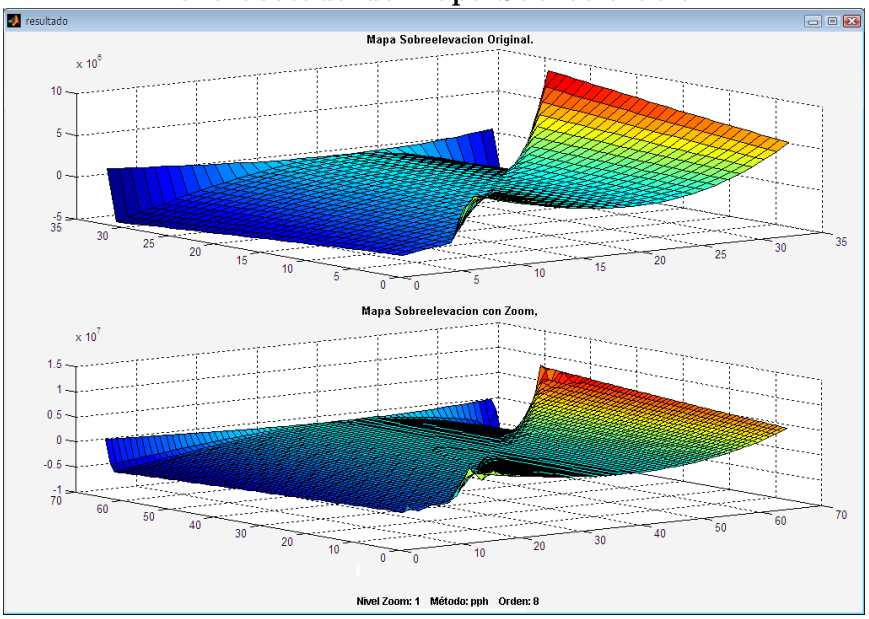

Figura 6.10: Método PPH, Nivel de Zoom 1, Orden 8.

Capítulo 7

Conclusiones

En este proyecto hemos desarrollado una interfaz gráfica cómoda para el usuario para llevar a cabo zoom y compresión de datos geológicos. La programación de todos los algoritmos ha sido llevada a cabo en el lenguaje de programación Matlab 7.0. Se han considerado dos tipos de algoritmos con distintos ordenes de aproximación. Por un lado los algoritmos lineales de multirresolución basados en interpolación segmentaria de Lagrange centrada con $2s$ puntos, y por otro lado una modificación no lineal que trata de adaptarse a las posibles singularidades presentes en los datos manteniendo siempre el mayor orden de aproximación factible. En las comparaciones numéricas hemos observado como una buena alternativa sería aplicar los métodos no lineales de cuarto orden. En general podemos decir que en zonas de suavidad los ordenes superiores se comportan mejor, pero que esto cambia radicalmente cuando la región presenta zonas más abruptas, en cuyo caso los ordenes bajos son preferibles. Por todo esto, pensamos que una buena alternativa es tomar orden 4 para obtener buenos resultados en ambos casos. La comparativa que podemos hacer entre el algoritmo lineal y el no lineal, es la ya conocida para otras aplicaciones, es decir, que en zonas de suavidad ambos algoritmos son similares, y que en zonas que presentan discontinuidades el algoritmo no lineal supera los resultados de su homólogo lineal ya que logra adaptarse a dichas singularidades. Por tanto, nuestra elección entre los algoritmos considerados sería aplicar el algoritmo no lineal PPH de orden 4 tanto para llevar a cabo el zoom como la compresión de los datos geológicos.

Bibliografía

[1] S.Amat, *A review on the piecewise polynomial harmonic interpolation* Appl. Num. Math., **58** (8), 1168-1185, (2008).

[2] 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).

[3] 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).

[4] S.Amat, K.Dadourian and J.Liandrat, *Analysis of a class of nonlinear subdivision schemes and associated multi-resolution transforms*, submitted (2008).

[5] S.Amat, K.Dadourian, J.Liandrat, J.C. Trillo *On a class of nonlinear interpolatory subdivision schemes*, submitted (2009).

[6] S.Amat, R.Donat, J.Liandrat and J.C.Trillo, *Analysis of a new nonlinear subdivision scheme. Applications in image processing*, Found. Comput. Math., **6** (2), 193–225, (2006).

[7] 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. , doi:10.1016/j.jmaa.2009.04.038, (2009).

[8] S.Amat and J.Liandrat. *On the stability of PPH nonlinear multiresolution*, Appl. Comp. Harm. Anal., **18** (2), 198-206, (2005).

[9] F. Aràndiga and R. Donat, *Nonlinear Multi-scale Decomposition: The Approach of A.Harten*, Numer. Algorith., **23**, 175-216, (2000).

[10] A. Cohen, N. Dyn and B. Matei, *Quasilinear subdivision schemes with applications to ENO interpolation*, Appl. Comp. Harm. Anal. **15**, 89-116, (2003).

[11] N.Dyn, *Subdivision schemes in computer aided geometric design*, Oxford University Press, **20**(4), 36-104, (1992).

[12] 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).

[13] A.Harten, *Multi-resolution Representation of Data II: General Framework*, SIAM J. Numer. Anal. **33** (3), 1205-1256, (1996).

[14] A.Harten, *ENO schemes with subcell resolution*, J. Comput. Phys., **83**, 148-184, (1989).

[15] 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).

[16] F.Kuijt, *Convexity Preserving Interpolation - Stationary Nonlinear Subdivision and Splines*. PhD Thesis, University of Twenty, Faculty of Mathematical Sciences, (1998).

[17] A. Marquina, *Local piecewise hyperbolic reconstruction of numerical fluxes for nonlinear scalar conservation laws*, SIAM J. Sci. Comput., **15** (4), 892-915, (1994).

[18] M.Rabbani and P.W.Jones, Digital Image Compression Techniques. Tutorial Text, Society of Photo-Optical Instrumentation Engineers (SPIE), TT07, 1991.

[19] S.Serna and A.Marquina, *Power ENO methods: a fifth order accurate Weighted Power ENO method*, J. Comput. Phys., **194**, 632-658, (2004).

[20] J.C. Trillo, *Nonlinear multiresolution and applications in image processing*, PhD in the University of Valencia, Spain, (2007).