# 5

# Aplicaciones a la eliminación de ruido

## 5.1 Introducción

En este capítulo estudiaremos la capacidad de las reconstrucciones y transformadas vistas en capítulos anteriores para la eliminación de ruido en señales digitales.

Una de las causas del deterioro de las señales digitales es el ruido. La definición y tipo de ruido depende del contexto en el que trabajemos. Podemos definir el ruido como aquel elemento independiente de la señal que interviene en la comunicación, procesamiento o medida de dicha señal, [27]. En el contexto de transmisión de señales por cable, podemos hablar de ruido térmico (debido al mo-

vimiento aleatorio de los electrones en un cable), inducido (debido a fuentes externas como motores y electrodomésticos), cruces (debido a el efecto de un cable sobre otro al actuar uno como antena emisora y otro como receptora) e impulsos (debido a picos producidos por lineas de potencia, iluminación...). En el contexto del sonido podemos encontrar el ruido aditivo (debido a la adición de otra señal procedente de fuentes que coexisten en el entorno), reverberación (producida por la propagación multitrayecto que se da en los entornos acústicos cerrados o semicerrados) y eco (producido generalmente por el acoplamiento entre micrófonos y altavoces). Aunque estos dos últimos no se tratan exactamente de ruido sino de distorsión (la señal cambia su forma de onda). En el contexto de imágenes el ruido se introduce en la captación de la imagen (los sensores utilizados para digitalizar pueden dañarse, o afectarles las condiciones climáticas, fuentes electromagnéticas, ruido eléctrico...), o puede introducirse en el proceso de transmisión.

Nosotros nos centraremos únicamente en ruido aditivo. Al introducir matemáticamente el ruido también podemos hacerlo de varias maneras, mediante una distribución uniforme, gaussiana, gamma, exponencial,... Además con correlación en el tiempo o no. Cada distribución modelará mejor un tipo de caso real[1] que contamina las señales digitales, aunque el modelo que cubre un mayor número de casos es el ruido *blanco gaussiano*. Será pues éste el que introduciremos en el presente trabajo.

Al igual que en la sección 3.3 consideraremos funciones del tipo $\tilde{f}(x) = f(x)+n(x)$, con $f(x)$ una función discontinua compuesta por trozos suaves y $n(x)$ el ruido *blanco gaussiano*.

Como medida objetiva para medir el ruido en una señal, a parte de *SNR*, (3.7), utilizaremos también el *Error Cuadrático Medio*:

$$ECM(f,\tilde{f}) = \frac{1}{N^2}\sum_{i=1}^{N}\left(f_i - \tilde{f}_i\right)^2. \tag{5.1}$$

[1]Por ejemplo si el CCD de la cámara tiene polvo, el ruido que se introduce es de tipo *sal y pimienta* o de *impulso*, que correspondería a una función de densidad $P(z)$ tal que $P(z) = P_a$ para $z = a$, $P(z) = P_b$ para $z = b$ y $P(z) = 0$ en otro caso.

# 5.2 Trabajando en una escala

En esta sección veremos varios métodos de reducción de ruido pero trabajando únicamente en una escala.

## 5.2.1 Aproximación por Mínimos Cuadrados (*AMC*) entre discontinuidades

Si la función no presenta discontinuidades, los mínimos cuadrados polinómicos son una excelente solución para el problema planteado. Sin embargo, si la función contiene discontinuidades no podemos aplicar mínimos cuadrados directamente, ya que la aproximación obtenida es un polinomio continuo y por tanto difiere en exceso de la función a reconstruir.

Motivados por los trabajos de Daphna Mizrachi [60], una sencilla solución para abordar el problema de las discontinuidades consiste en:

1. Localizar las discontinuidades.
2. Aproximar por mínimos cuadrados cada trozo entre las discontinuidades.

El problema se basa entonces en la localización correcta de las discontinuidades, punto que fue tratado el capítulo 3. Para funciones con ruido disponemos de varios métodos que localizan discontinuidades de salto pero no esquinas.

### Localización de discontinuidades

En [60] se utiliza el método de localización basado en la combinación de las medidas $E_1$ (3.8) y $E_2$ (3.9). Como se comprobó en el capítulo 3 dicho método era poco fiable y diseñamos mejores

alternativas de localización. Nos hemos decantado por la *reconstrucción por aproximaciones de stencil creciente* ($\overline{\mathcal{AP}}-SC$, pag. 159) ya que, según los experimentos numéricos, produce escasas falsas discontinuidades y detecta correctamente saltos incluso con funciones en las que el ruido introducido es elevado.

## Grado de la *AMC* de cada trozo entre discontinuidades

Si suponemos localizadas correctamente las discontinuidades debemos establecer un algoritmo que permita obtener el grado óptimo para la *AMC* en cada trozo suave.

El parámetro de correlación

$$r(\epsilon) = \frac{\sum_{i=2}^{n}(\epsilon_i \epsilon_{i-1})}{\sum_{i=2}^{n} \epsilon_i^2}, \tag{5.2}$$

varía en el intervalo $[-1, 1]$. Los valores cercanos a 0 indican no correlación, mientras que los valores cercanos a 1 y -1 nos revelan una estructura correlada. Como ya indicamos, nosotros trabajamos con ruido no correlado, y por tanto si $\epsilon$ representa el ruido, $r(\epsilon) = 0$. Este hecho puede ayudarnos a nuestro propósito, ya que si hemos realizado una buena eliminación de ruido a una señal suave, la diferencia entre la señal con ruido y la que hemos obtenido tras eliminarlo, producirá una serie de datos no correlados, pues constituirá el error (en el caso ideal) que hemos introducido. Dicho en otros términos, si $f$ representa la señal inicialmente suave, $\tilde{f}$, a la que añadimos ruido, y $\hat{f}$ la obtenida tras su eliminación, debería ocurrir que $r(f - \hat{f}) \approx 0$.

Sea $P(x, \tilde{f})$ la función polinómica a trozos que aproxima $\tilde{f}$ en los puntos $\{x_i\}$. Entre dos puntos en los que $\tilde{f}$ es discontinua, $x_{d_i} < x < x_{d_{i+1}}$, $P$ viene dada mediante la *AMC*, de grado $m$:

$$P(x; \tilde{f}) \equiv q^{\overline{\mathcal{AP}}}(x; \tilde{f}, m) \text{ para } x_{d_i} < x < x_{d_{i+1}}.$$

Observemos las siguientes cantidades para $d_i < s \leq d_{i+1} - 1$:

- Los elementos de ruido: $\epsilon(x_s) \equiv \tilde{f}(x_s) - f(x_s)$,

- El error de aproximación: $E_m(x_s) \equiv q^{\overline{\mathcal{AP}}}(x_s; \tilde{f}, m) - f(x_s)$,
- Los residuales: $e_m(x_s) = q^{\overline{\mathcal{AP}}}(x; \tilde{f}, m) - \tilde{f}(x_s)$.

Las dos primeras cantidades no son calculables ya que desconocemos $f(x)$. Sin embargo podemos poner la última en función de las anteriores:

$$\begin{aligned} e_m(x_s) &= q^{\overline{\mathcal{AP}}}(x_s; \tilde{f}, m) - \tilde{f}(x_s) = q^{\overline{\mathcal{AP}}}(x; \tilde{f}, m) - f(x_s) + f(x_s) - \tilde{f}(x_s) \\ &= E_m(x_s) - \epsilon(x_s). \end{aligned} \tag{5.3}$$

Así, los residuales dependen del error de aproximación (que a su vez depende de $r$) y de los elementos de ruido (que siguen una distribución con estructura no correlada). Podemos extraer las siguientes conclusiones:

- Para $m$ pequeño:

$$q^{\overline{\mathcal{AP}}}(x; \tilde{f}, m) \approx q^{\overline{\mathcal{AP}}}(x; f, m) \Longrightarrow E_m(x) \approx q^{\overline{\mathcal{AP}}}(x; f, m) - f(x).$$

- Al incrementar $m$ (para $m$ suficientemente pequeño) $E_m$ es mucho menor que el ruido y por tanto $r(e_m) \approx r(\epsilon) \approx 0$.
- Para valores altos de $m$, $q^{\overline{\mathcal{AP}}}(x; \tilde{f}, m)$ se acerca a $\tilde{f}$.

De lo anterior deducimos que lo que nos interesa buscar es el menor $m$ tal que $r(e_m) \approx 0$. El estadístico más comúnmente utilizado para medir el parámetro de correlación es:

$$d = \frac{\sum_{i=2}^{n} (\epsilon_i - \epsilon_{i-1})^2}{\sum_{i=2}^{n} \epsilon_i^2}. \tag{5.4}$$

Existe la siguiente relación entre $d$ y $r$: $d = 2(1 - r)$. Por tanto, que $r(\epsilon) = 0$ equivale a $d(\epsilon) \approx 2$. Nos interesa minimizar la cantidad $|d(e_m) - 2|$ y utilizar un test de autocorrelación. El estadístico $d$ se utiliza para verificar la hipótesis nula $H_0$ (residuales no correlados), frente la hipótesis alternativa $H_1$ (correlados). Un conocido test de autocorrelación es el de Durbin-Watson[2], que funciona de la siguiente manera: calculamos el estadístico $d$, si

[2]En MATLAB® podemos utilizar el programa *dwtest.m* para dicho test.

1. $d < d_L$ rechazar $H_0$,

2. $d > d_U$ no rechazar $H_0$,

3. $d_L < d < d_U$ el test es inconcluyente.

Los valores $(d_L, d_U)$, según el número de muestras, fueron tabulados por Durbin y Watson (ver [37]), y posteriormente ampliados en [64] por Saving y White.

El Algoritmo 5.1 resume lo expuesto para hallar el grado óptimo, $m_0$, según cada trozo entre discontinuidades.

**Algoritmo 5.1** Grado óptimo para $AMC$ entre discontinuidades

**Entrada:** $\{f_i\}_{d_i \leq i < d_{i+1}}$, los datos entre discontinuidades, $m_{max}$ el máximo grado admitido
**Salida:** $m_0$, el grado óptimo

```
1: m = 1
2: entra = 0
3: cálculo de e_1 y d(e_1)
4: mientras m ≤ m_max hacer
5:     cálculo de e_m y d(e_m)
6:     si d(e_m) > d_U entonces
7:         si |d(e_{m-1}) − 2| < |d(e_m) − 2| entonces
8:             m_0 = m − 1;
9:             m = m_max;
10:            entra = 1;
11:        si no
12:            m = m + 1;
13:        fin si
14:    si no
15:        m = m + 1;
16:    fin si
17: fin mientras
18: si entra == 0 entonces
19:    m_0 = m_max;
20: fin si
```

### Experimentos numéricos

Hemos realizado varios experimentos con funciones que contienen saltos y esquinas. Se parte de 256 puntos iniciales y se utiliza grado 3 para realizar el método de detección de discontinuidades $\overline{\mathcal{AP}} - SC$. En cada gráfico aparece la función sin ruido, con ruido y la aproximación, indicándose los grados calculados para cada trozo suave y el *ECM* y *SNR* entre la función original con ruido y la reconstrucción[3]. El grado máximo admitido, $m_{max}$, es 10.

En la Figura 5.1 (a) partimos de la función (1.44) a la que añadimos ruido blanco gaussiano con *SNR*=24. La función contiene un salto que es detectado correctamente. El primer trozo se aproxima con un polinomio de grado 1, mientras que el segundo trozo se aproxima con un polinomio de grado 10, ya que la esquina no se localiza y por tanto se necesita un grado elevado para el polinomio por *aproximaciones*. En (b) la función de partida contiene dos saltos y una esquina, (3.15), con la misma magnitud de ruido. Los grados hallados son 5, 8 y 3. El grado 8 corresponde a la zona central que incluye una discontinuidad esquina que no ha sido detectada.

En general el método proporciona resultados aceptables siempre que se localicen correctamente los saltos. Si esto no ocurre, se obtienen soluciones que distan mucho de la función original, lo cual motiva a que sigamos analizando otros métodos de eliminación de ruido.

## 5.2.2
## Reconstrucción $\overline{\mathcal{AP}}$ para cada punto

En este apartado estudiaremos la aplicación de mínimos cuadrados 'punto a punto', es decir, se tomará un determinado *stencil* que contenga el punto y el valor de la función se sustituirá por el valor del polinomio $q^{\overline{\mathcal{AP}}}$ en dicho punto.

[3]También podríamos haber calculado el error entre la función original sin ruido y la reconstrucción. Pero dado que la función original sin ruido nos es, en principio, desconocida, optamos por calcular (mientras no indiquemos lo contrario) los errores entre la original con ruido y la reconstrucción.

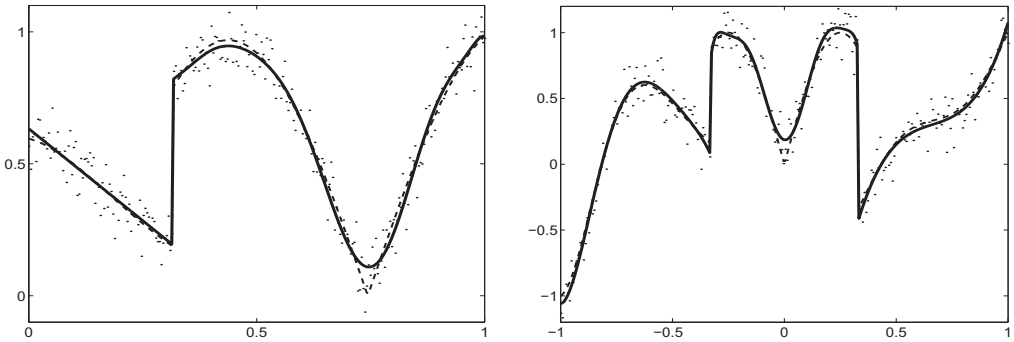
De forma más rigurosa, dada $\{(x_i, f_i)\}_{i=1}^{N}$ la función inicial, para cada $x_i$ se calcula $q_i^{\overline{\mathcal{AP}}_{nl,nr}}(x; f, r)$ como el polinomio de grado $r$ obtenido por mínimos cuadrados calculados a partir de los $m = nl + nr \geq r + 2$ puntos $\{x_{i-nl}, \ldots, x_{i+nr-1}\}$. Con esto sustituimos cada $f_i$ por $q_i^{\overline{\mathcal{AP}}_{nl,nr}}(x_i; f, r)$.

Si centramos el intervalo de trabajo, es decir $nl = nr$ obtenemos la Figura 5.2. En ella observamos:

- El aumentar el grado del polinomio no mejora la eliminación de ruido, ya que la aproximación de los datos con ruido aumenta.

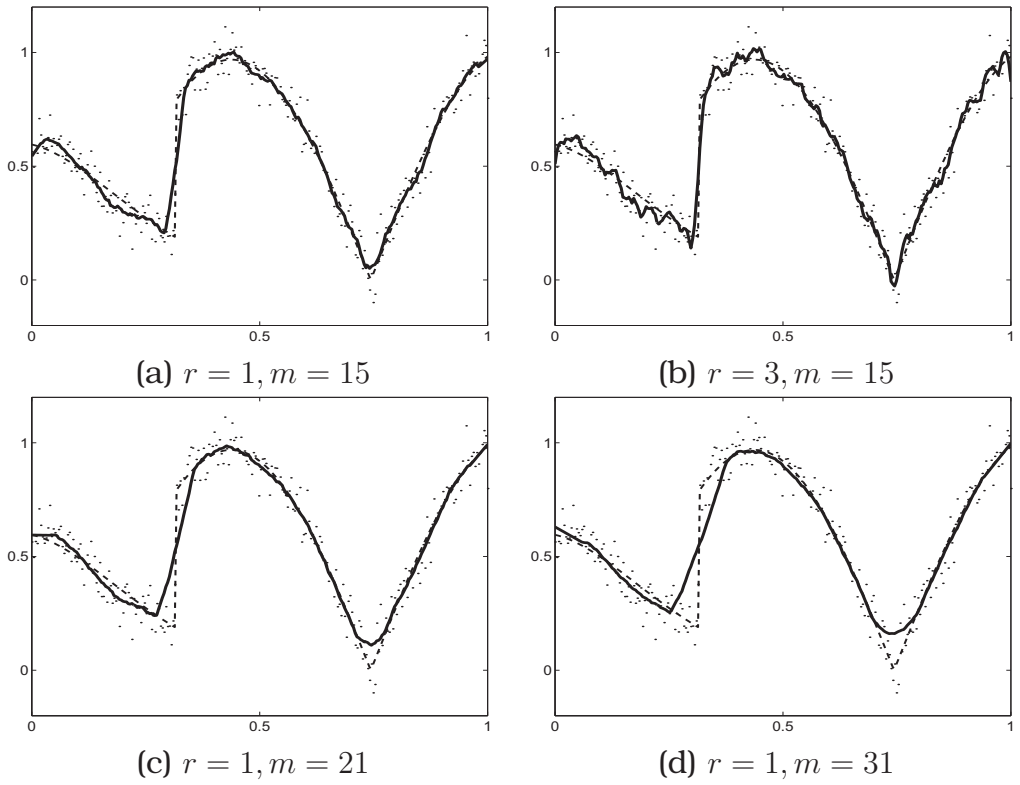
- Al aumentar la longitud del *stencil* los resultados mejoran, aunque los saltos quedan más suavizados y aumenta el *fenómeno de Gibbs*.

Para resolver el problema planteado en el segundo punto debemos evitar la utilización de *stencils* que crucen discontinuidades.



(a) Grados: [1,10]. (b) Grados: [5, 8, 3].

**Figura 5.1:** *Aplicamos AMC entre discontinuidades, determinadas por el método $\overline{\mathcal{AP}} - SC$ a las funciones (1.44) en (a) y (3.15) en (b), ambas con 256 nodos y $SNR = 24$. El grado de cada trozo, determinado por el test de Durbin-Watson, se indica bajo la figura. La línea discontinua representa la función sin ruido, los puntos son la función con ruido añadido y la lnea continua es la reconstrucción obtenida. Errores: (a) $ECM$= 1.79e-5, $SNR$= 19.53; (b) $ECM$= 5.20e-5, $SNR$=14.36.*



(a) $r = 1, m = 15$ (b) $r = 3, m = 15$

(c) $r = 1, m = 21$ (d) $r = 1, m = 31$

**Figura 5.2:** *$\overline{\mathcal{AP}}$ para cada punto con stencil centrado (en las fronteras sí se adapta) aplicado a la función (1.44) con 256 puntos y $SNR = 24$. $r$ es el grado del polinomio de la reconstrucción, $m$ es el tamaño del stencil. La línea discontinua representa la función sin ruido, los puntos son la función con ruido añadido y la línea continua es la reconstrucción obtenida. Errores: (a) $ECM$=1.81e-5, $SNR$=19.43; (b) $ECM$=1.42e-5, $SNR$=20.51; (c) $ECM$=2.41e-5, $SNR$=18.13; (d) $ECM$=3.26e-5, $SNR$=16.77.*

## Evitando cruzar discontinuidades

Aplicaremos la idea *ENO* eligiendo *stencils* que no crucen discontinuidades. Dado que trabajamos con ruido, el elegir el *stencil* de menor diferencia dividida no asegura que no se cruce un salto. Podemos recurrir a otras medidas, como $E_1$ y $E_2$, introducidas en la sección 3.3.1, que consiguen evitar utilizar *stencils* que cruzan discontinuidades para funciones que contienen ruido.

En la Figura 5.3 representamos las soluciones que obtenemos al elegir el *stencil* que minimiza el valor de $E_1$ o $E_2$. Al comparar con métodos centrados, se observa que la disminución del error es menor, aunque se consigue no suavizar el salto en la mayoría de casos. La disminución de ruido es más escasa ya que para un cierto intervalo de puntos se utiliza el mismo *stencil* (el que minimiza $E_1$ o $E_2$) y no uno diferente cada vez como en el caso centrado.

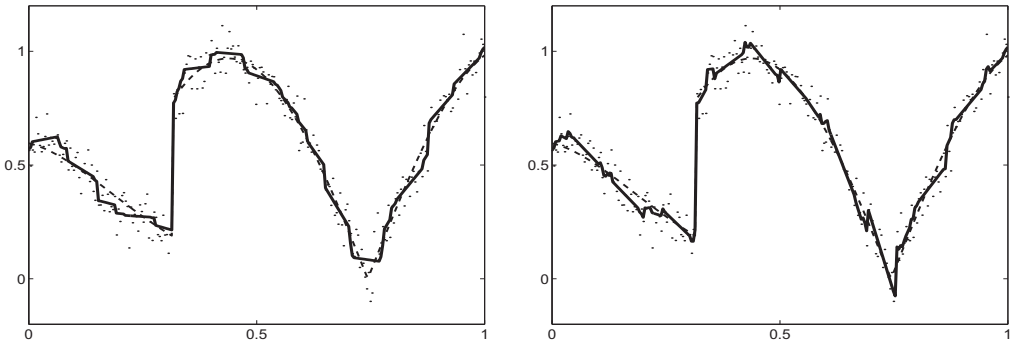



**Figura 5.3:** *Tomando 256 puntos de la función (1.44), con $SNR = 24$, sustituimos cada punto la reconstrucción $\overline{AP}$ de grado 1 y longitud de stencil 31, eligiendo el stencil según la medida $E_1$ en (a) y según $E_2$ en (b). La línea discontinua representa la función sin ruido, los puntos son la función con ruido añadido y la línea continua es la reconstrucción obtenida. Errores: (a) $ECM$=1.53e-5, $SNR$=20.26; (b) $ECM$=1.20e-5, $SNR$=21.29.*

## *Stencil* centrado excepto en las discontinuidades

Hemos visto que al tomar *stencils* centrados, la eliminación de ruido es mayor excepto en las zonas de discontinuidad. Si evitamos cruzar los saltos, mejoramos en la discontinuidad pero empeoramos la eliminación de ruido en las zonas suaves. Para aprovechar los beneficios de cada caso planteamos el siguiente esquema: localizamos las discontinuidades y utilizamos un *stencil* centrado, excepto en los puntos donde se crucen saltos. En dichos puntos se toman *stencils* que no crucen el salto. El Algoritmo 5.2 muestra este esquema. Una vez más la localización de discontinuidades juega un papel importante, aunque si hubiese algún error

no sería tan grave como en el método de aproximar por *AMC* entre discontinuidades. En este caso si no se localiza un salto, simplemente quedará suavizado.

---

**Algoritmo 5.2** Recostrucción $\overline{\mathcal{AP}}$ con *stencil* centrado excepto en las discontinuidades.

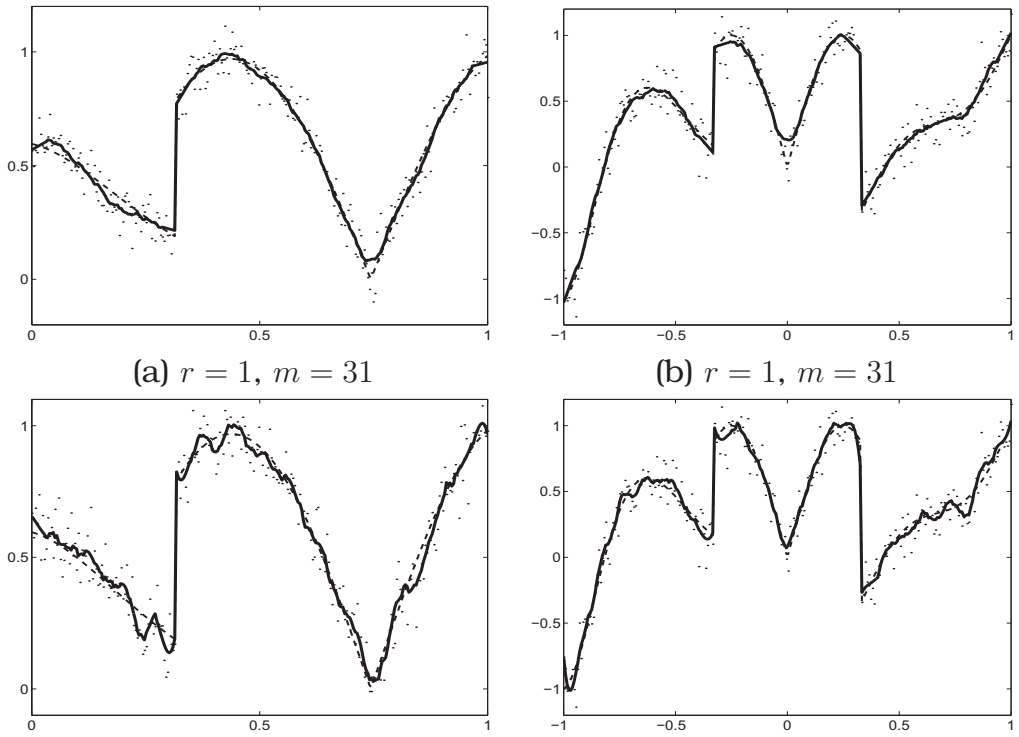
---

**Entrada:** Datos iniciales $\{f_i\}_{i=1}^N$, localización de discontinuidades $\{d_i\}_{i=1}^p$, $m$ tamaño del *stencil*, $r$ grado del polinomio $\overline{\mathcal{AP}}$.
**Salida:** Aproximación obtenida $\{\tilde{f}_i\}_{i=1}^N$.

**para** $j = 2, \ldots, p$ **hacer**
  $k = 0$;
  **para** $i = d_{j-1}, \ldots, d_{j-1} + \frac{m-1}{2} - 1$ **hacer**
    $\tilde{f}_i = q^{\overline{\mathcal{AP}}^r}$ con *stencil* $\{x_{i-k}, \ldots, x_{i+m-1-k}\}$
    $k = k + 1$
  **fin para**
  **para** $i = d_{j-1} + \frac{m-1}{2}, \ldots, d_j - \frac{m-1}{2}$ **hacer**
    $\tilde{f}_i = q^{\overline{\mathcal{AP}}^r}$ con *stencil* $\{x_{i-\frac{m-1}{2}}, \ldots, x_{i+\frac{m-1}{2}}\}$
  **fin para**
  $k = 0$
  **para** $i = d_j - \frac{m-1}{2}, \ldots, d_j - 1$ **hacer**
    $k = k + 1$
    $\tilde{f}_i = q^{\overline{\mathcal{AP}}^r}$ con *stencil* $\{x_{i-\frac{m-1}{2}-k}, \ldots, x_{i+\frac{m-1}{2}-k}\}$
  **fin para**
**fin para**

---

En la Figura 5.4 hemos aplicado este método para la funciones con uno y dos saltos. El método de localización de discontinuidades será $\overline{\mathcal{AP}} - SC$ con grado 3. Apreciamos la correcta aproximación en los saltos en todos los casos, ya que han sido localizados correctamente. Las discontinuidades esquina quedan suavizadas ya que el método de detección no las localiza. Si aumentamos el grado del polinomio de la reconstrucción $\overline{\mathcal{AP}}$, la eliminación de ruido empeora, aunque la aproximación en las esquinas mejora.



(a) $r = 1$, $m = 31$ (b) $r = 1$, $m = 31$

(c)$r = 3$, $m = 31$ (d) $r = 3$, $m = 31$

**Figura 5.4:** *Tomando 256 puntos de (1.44), con $SNR = 24$, se sustituye cada punto por el valor de $\overline{\mathcal{AP}}$ de grado $r$ y longitud de stencil $m$, con stencil centrado excepto en las discontinuidades, localizadas mediante $\overline{\mathcal{AP}} - SC$ (Algoritmo 5.2). La línea discontinua representa la función sin ruido, los puntos son la función con ruido añadido y la línea continua es la reconstrucción obtenida. Errores: (a) $ECM$=1.55e-5; $SNR$=20.10; (b) $ECM$=5.07e-5, $SNR$=14.22 ; (c) $ECM$=1.75e-5, $SNR$=19.60; (d) $ECM$=3.92e-5, $SNR$=15.46.*

## 5.2.3

## Iteración por sustitución de valores medios

En este apartado presentamos un sencillo algoritmo de reducción de ruido que no necesita calcular reconstrucciones. El algoritmo se basa en sustituir cada punto por la media de él mismo y

su anterior, excepto si se detecta una posible discontinuidad, en cuyo caso se sustituye por la media entre él mismo y el siguiente. Posteriormente este paso se itera.

La condición de discontinuidad es la siguiente:

$$factor \cdot \frac{1}{i-2} \cdot \sum_{j=2}^{i-1} \left| \bar{f}_j - f_j \right| < \left| \bar{f}_i - f_i \right|, \tag{5.5}$$

donde $\bar{f}_i$ representa la media $|f_i - f_{i-1}|/2$.

El Algoritmo 5.3 muestra este proceso. Tomando un valor adecuado de *factor* se consigue que la media de los valores se calcule a partir de datos de un solo lado de la discontinuidad. Experimentalmente hemos comprobado que valores de *factor* iguales a 5 o 6 dan buenos resultados. La magnitud de *factor* determinará el tamaño de los saltos que respetaremos. En el algoritmo sobrescribimos $f_i$ en cada iteración con lo que se consigue un aumento de la velocidad de eliminación de ruido. Dado que la mayoría de valores se sustituyen por la media de él mismo con su anterior, se produce un desplazamiento de la función hacia la derecha. Para compensarlo creamos el Algoritmo 5.4, donde la media se realiza con el siguiente valor. Así pues, tras aplicar el algoritmo que realiza la media a la izquierda, se aplica el que la realiza a la derecha, compensando de esta manera el desplazamiento. Posteriormente se itera el proceso, obteniendo el Algoritmo 5.5.

Veamos algunos ejemplos. En la Figura 5.5 aplicamos 8 iteraciones a varias funciones en las que se han introducido diferentes magnitudes de ruido. Con tan sólo 8 iteraciones se consigue una eliminación eficiente de ruido. El aumentarlas no mejora los resultados ya que las esquinas se siguen suavizando con cada nueva iteración.

## 5.3 Trabajando con varias escalas

Los esquemas de multirresolución son utilizados satisfactoriamente para la compresión de señales. Al bajar cada nivel se elimi-

**Algoritmo 5.3** Media de cada valor con su anterior excepto en posibles discontinuidades.

**Entrada:** $\{f_i\}_{i=1}^n$, la función de partida; $factor$, determina la magnitud de los saltos respetados.
**Salida:** $\{f_i\}_{i=1}^n$

1: $err_2 = |f_2 - f_1|/2$
2: **para** $i = 3, 4, \ldots, n-1$ **hacer**
3: $aux = f_i$
4: $f_i = (f_{i-1} + f_i)/2$
5: $err_i = |aux - f_{i-1}|/2$
6: **si** $factor * \frac{1}{i-2} \sum_{j=2}^{i-1} err_j < err_i$ **entonces**
7: $f_i = (aux + f_{i+1})/2$
8: $err = |f_i - f_{i+1}|/2$
9: **fin si**
10: $f_1 = (f_1 + f_2)/2$
11: $f_2 = (f_2 + f_3)/2$
12: $f_n = (f_{n-1} + f_n)/2$
13: **fin para**

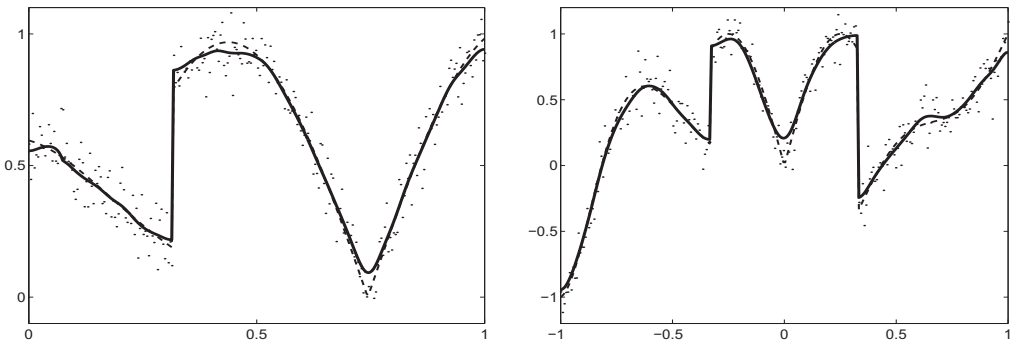



**Figura 5.5:** *8 iteraciones del Algoritmo 5.5. Partimos con 256 puntos de las funciones (1.44) en (a), y (3.15) en (b), con ruido de magnitud $SNR = 24$. La línea discontinua representa la función sin ruido, los puntos son la función con ruido añadido y la línea continua es la reconstrucción obtenida. Errores: (a) $ECM$=1.96e-5, $SNR$=29.53; (b) $ECM$=4.88e-5, $SNR$=22.80.*

**Algoritmo 5.4** Media de cada valor con su siguiente excepto en posibles discontinuidades.

**Entrada:** $\{f_i\}_{i=1}^n$, la función de partida; $factor$, determina la magnitud de los saltos respetados
**Salida:** $\{f_i\}_{i=1}^n$

$err_n = |f_n - f_{n-1}|/2$;
**para** $i = n-1, n-2, \ldots, 2$ **hacer**
  $aux = f_i$
  $f_i = (f_i + f_{i+1})/2$
  $err_i = |f_{i+1} - aux|/2$
  **si** $factor \cdot \frac{1}{n-i} \sum_{j=i+1}^{n} err_j < err_i$ **entonces**
    $f_i = (aux + f_{i-1})/2$
    $err_i = |f_i - f_{i-1}|/2$
  **fin si**
  $f_1 = (f_1 + f_2)/2$
  $f_n = (f_{n-1} + f_n)/2$
**fin para**

na información prescindible para recuperar la señal del nivel superior bajo una cierta tolerancia. Esta idea guarda gran relación con el problema aquí tratado, ya que dicha información prescindible puede interpretarse como ruido en este caso. Es por ello que en esta sección trabajaremos con varias escalas.

Comenzaremos estudiando las posibilidades que ofrecen las descomposiciones multiescala vistas en los capítulos 1 y 2. Posteriormente revisaremos las eliminación de ruido utilizando *wavelets* y finalmente veremos si estas ideas se pueden adaptar a las descomposiciones multiescala.

## 5.3.1

## Eliminación de ruido mediante descomposiciones multiescala *à la Harten*

Debemos plantearnos cuáles son las mejores opciones que conducen a nuestro objetivo: la eliminación de ruido en funciones

**Algoritmo 5.5** Eliminación de ruido mediante proceso iterativo por sustitución de medias.

**Entrada:** $\{f_i\}_{i=1}^n$, la función de partida; $factor$, determina la magnitud de los saltos respetados; $iter$, número de iteraciones
**Salida:** $\{f_i\}_{i=1}^n$, la función de salida
1: **para** $i = 1, 3, \ldots, iter$ **hacer**
2: $\{f_i\}_{i=1}^n \leftarrow$ Algoritmo 5.3 con datos iniciales $\{f_i\}_{i=1}^n$, $factor$
3: $\{f_i\}_{i=1}^n \leftarrow$ Algoritmo 5.4 con datos iniciales $\{f_i\}_{i=1}^n$, $factor$
4: **fin para**

compuestas por trozos suaves. Para la decimación hemos estudiado dos posibilidades: los valores puntuales y las medias en celda. Para la predicción tres: reconstrucciones por *interpolación*, por *interpolación aproximación* y por *aproximaciones*.

En cuanto a la decimación, si utilizamos valores puntuales al bajar de niveles no se produce ninguna eliminación de ruido. Con esto, tras bajar $n$ niveles nos encontramos con una función con menos nodos pero con la misma magnitud de ruido que la inicial. A continuación, si queremos eliminar ruido, podemos utilizar $\overline{\mathcal{AP}}$. Aquí nos encontramos con dos problemas:

1. *ENO* basado en la magnitud de diferencias divididas no es adecuado, ya que los datos contienen ruido.

2. La reconstrucción basada en

$$\begin{cases} \text{Para } k = 1, \ldots, L & \\ f_{2i}^k = f_i^{k-1} & i = 0, \ldots, J_{k-1}, \\ f_{2i-1}^k = \overline{\mathcal{AP}}(x_{2i-1}^k; f^{k-1}) & i = 1, \ldots, J_{k-1}. \end{cases} \tag{5.6}$$

   no contribuye a la eliminación de ruido, ya que los nodos pares no se modifican.

Aunque estos problemas podrían solucionarse utilizando la medida $E_2$ y el algoritmo que aproxima también en los pares (2.42), los saltos quedarán también suavizados, ya que *SR* no mejora la aproximación en ellos. Así pues, el uso de medias en celda proporcionará mejores resultados en cuanto a eliminación de ruido que valores puntuales. Es por ello que descartamos trabajar con

valores puntuales y nos centramos en medias en celda, donde la eliminación de ruido es doble: al decimar y al predecir

Como puede comprobarse en la Figura 5.6, si bajamos el suficiente número de niveles no es necesario modificar la asignación *ENO*, ya que el ruido disminuye y las diferencias divididas pueden mantenerse. Aunque este 'suficiente' número de niveles va a depender del ruido inicial introducido, que a priori nos es desconocido. Como alternativa podemos sustituir las diferencias divididas por $E_2$ que también evita discontinuidades y puede ser utilizado en presencia de ruido. Para reconstruir $\bar{f}^k_{2i-1}$ debemos utilizar el *stencil* $\left\{x^{k-1}_{i_2(i)}, x^{k-1}_{i_2(i)+1}, \ldots, x^{k-1}_{i_2(i)+m-1}\right\}$, donde $m = nl + nr + 1$, y $i_2(i)$ se halla según (3.9), (3.10) y (3.11).

Llegados a este punto existen varias combinaciones de algoritmos que podemos aplicar. Intentaremos razonar qué opciones son las mejores:

**Reconstrucción interpolatoria, $\overline{\mathcal{IC}}$:** utilizamos diferencias divididas para la asignación *ENO*, ya que para $E_2$ los *stencils* no son de suficiente longitud como para realizar mínimos cuadrados.

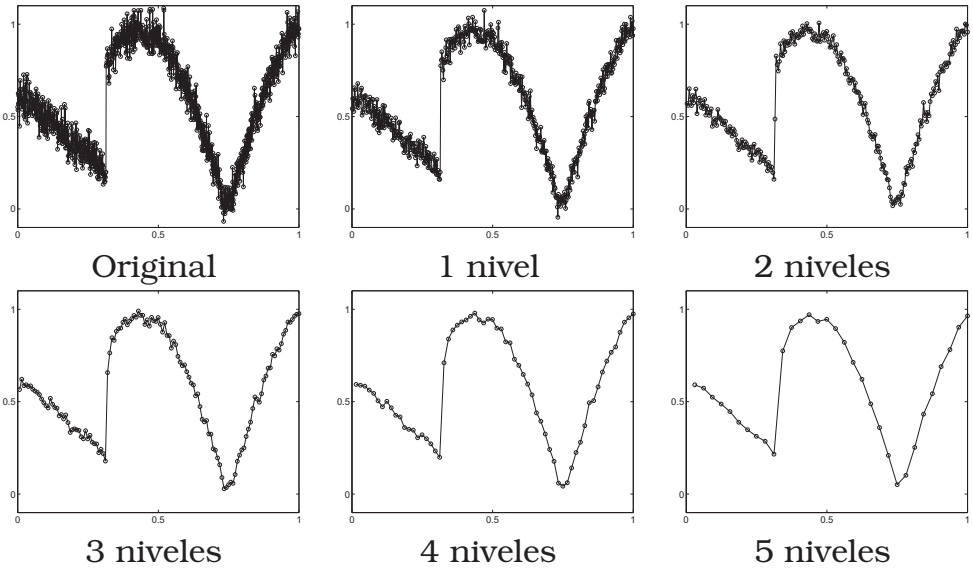



**Figura 5.6:** *Decimación por medias en celda.*

Por tanto no introducimos ninguna modificación al algoritmo, pero debemos utilizar un número de niveles suficiente para que la reducción de ruido permita la utilización correcta del esquema (unos 4 niveles es suficiente, aunque depende de la magnitud del ruido).

**Reconstrucción *interpolación-aproximación*, $\overline{\mathcal{IAC}}$:** en este caso sí podemos utilizar la medida $E_2$. Aunque esto no es una gran ventaja, ya que si bajamos pocos niveles, por un lado la reducción del ruido es escasa y por otro la condición para que se cumpla la consistencia en presencia de ruido no favorece su eliminación. Así pues debemos bajar suficientes niveles y podemos utilizar $E_2$ para asegurar que se evita cruzar discontinuidades.

**Reconstrucción por *aproximaciones*, $\overline{\mathcal{AC}}$:** podemos hacer uso de la medida $E_2$, aunque tenemos los mismos problemas que el caso anterior. Sin embargo podemos utilizar la modificación (2.51) que daba lugar a una reconstrucción no consistente. Otra opción es (2.52) que sí es consistente, pero también limita la reducción de ruido. Esto puede utilizarse en casos de ruido elevado en el que no podemos bajar suficientes niveles, aunque al ser no consistente la posición de los saltos al reconstruir no queda garantizada.

En definitiva, si no pretendemos utilizar un número suficiente (al menos 4) de niveles, es mejor decantarnos por los métodos vistos para una escala. Si a pesar de ello queremos bajar algún nivel podemos utilizar $\overline{\mathcal{AC}}$ con la modificación de no consistencia (2.51) y aplicando *ENO* mediante la medida $E_2$.

## Ejemplos

Trabajando con la función (1.70), tomamos 1024 puntos iniciales para que nos permita la utilización de suficientes niveles.

En un primer experimento introducimos ruido de magnitud $SNR = 25$ y bajamos 5 niveles por medias en celda, obteniendo la Figura 5.7. Las reconstrucciones utilizadas son $\overline{\mathcal{IC}}$ (a), $\overline{\mathcal{IAC}}$ (b), $\overline{\mathcal{AC}}$ (c) y $\overline{\mathcal{AC}}$ con la modificación (2.51) (d), todas ellas haciendo uso de la técnica *SR*. En los tres últimos sustituimos las diferencias divididas por la medida $E_2$ para decidir los *stencils ENO*. Los resul-

tados son satisfactorios, fruto de haber bajado 5 niveles. Al utilizar medias en celda, la función se ha suavizado lo suficiente como para que las diferencias divididas puedan ser utilizadas en la técnica *ENO*, como vemos en (a). Además los requisitos de consistencia no introducen variaciones significativas dado que las reconstrucciones obtenidas son suaves entre discontinuidades. Notar que en (d) el salto ha sido desplazado debido a la utilización de la reconstrucción no consistente.

En un segundo experimento nos planteamos qué ocurre si aumentamos el ruido a magnitud $SNR = 20$ y bajamos únicamente tres niveles. Se obtiene la Figura 5.8. Observamos que en (a) se aprecia el *fenómeno de Gibbs* al utilizarse *stencils* que cruzan el salto debido a que trabajamos con diferencias divididas. En (b) y (c) este problema queda solucionado al hacer uso de $E_2$. Sin embargo la eliminación de ruido es escasa, incluso con el uso de la reconstrucción $\overline{\mathcal{AC}}$ debido a que con 3 niveles no se elimina suficiente ruido. Además el problema se agrava debido a la condición de consistencia, lo cual provoca que la reconstrucción se quiebre más. Al utilizar la reconstrucción $\overline{\mathcal{AC}}$ aproximando también en los nodos pares (2.51) obtenemos (d), donde se elimina más ruido, aunque aún se producen algunos picos indeseables fruto de la localización de falsas discontinuidades por parte de $E_2$.

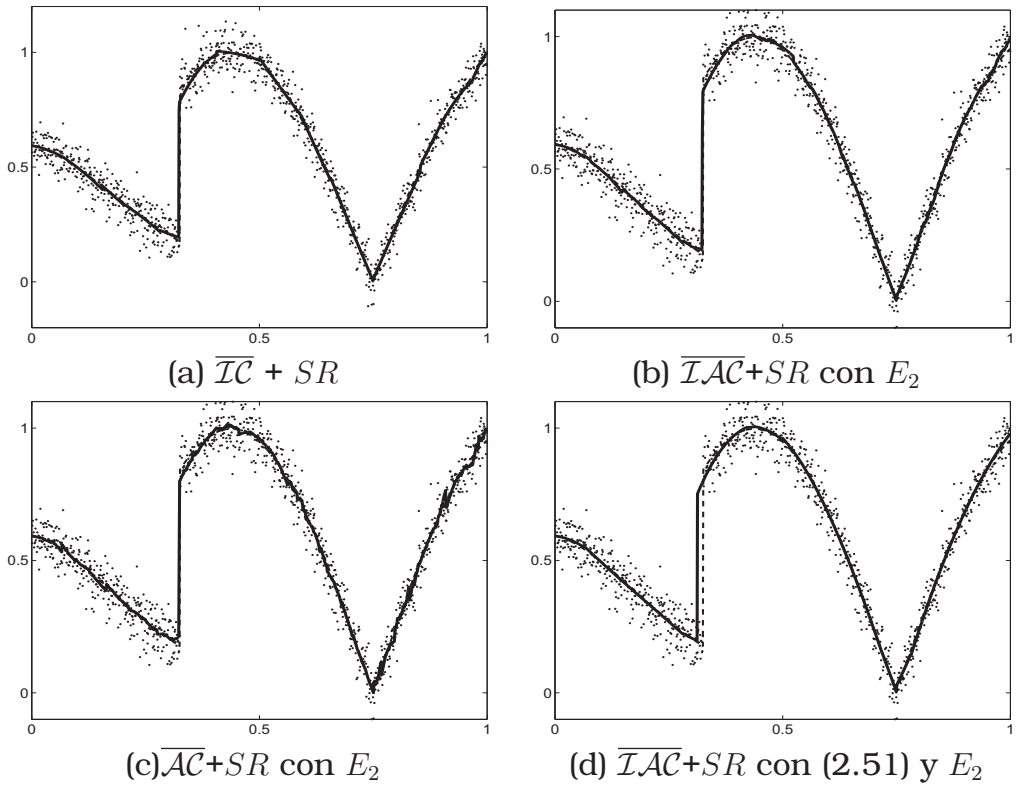
En la mayoría de casos las diferencias divididas pueden utilizarse sin problemas siempre que bajemos 4 o 5 niveles. Aún así $E_2$ da lugar a resultados más fiables que las diferencias divididas. En la Figura 5.9 trabajamos con la reconstrucción $\overline{\mathcal{IAC}}$, con 5 niveles y $SNR = 20$. A la derecha utilizamos diferencias divididas y a la izquierda $E_2$, obteniendo una mejor aproximación en el salto en este último caso.

En vista de los resultados podemos concluir que este método elimina eficientemente ruido siempre que podamos bajar suficientes niveles. El número de niveles adecuado dependerá de la magnitud del ruido inicial. Además es preferible el uso de $E_2$ frente a diferencias divididas. Dicho cambio sólo puede utilizarse en las reconstrucciones $\overline{\mathcal{IAC}}$ y $\overline{\mathcal{AC}}$. Con la modificación de no consistencia para $\overline{\mathcal{AC}}$ no se obtienen ventajas claras, ya que si utilizamos suficientes niveles no se asegura una reconstrucción correcta de los saltos (y con $\overline{\mathcal{AC}}$ ya se obtienen buenos resultados) y además

con pocos niveles no se elimina eficientemente ruido, siendo preferible otros métodos como por ejemplo los vistos para una escala.

## 5.3.2 *Multiscale Shrinkage*

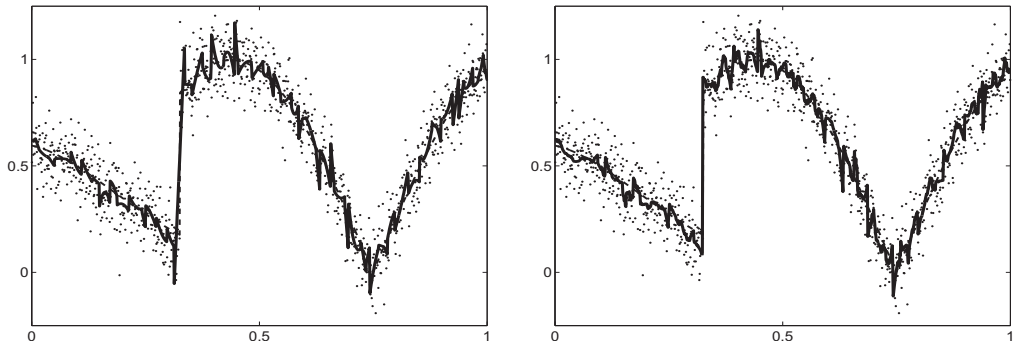
En los capítulos previos hemos utilizado las descomposiciones multiescala para eliminar ruido en señales discontinuas. Para me-



(a) $\overline{\mathcal{IC}}$ + $SR$ (b) $\overline{\mathcal{IAC}}$+$SR$ con $E_2$

(c)$\overline{\mathcal{AC}}$+$SR$ con $E_2$ (d) $\overline{\mathcal{IAC}}$+$SR$ con (2.51) y $E_2$

**Figura 5.7:** *Decimamos 5 niveles mediante medias en celda y subimos otros 5 según cada reconstrucción, con $r = 3$ y $nl = nr = 3$ (excepto en (a)), para la función (1.70) con 1024 puntos y $SNR = 25$. La línea discontinua representa la función sin ruido, los puntos son la función con ruido añadido y la línea continua es la reconstrucción obtenida. Errores: (a) $ECM$=3.71e-6, $SNR$=20.33; (b) $ECM$=3.57e-6, $SNR$=20.50; (c) $ECM$=3.49e-6, $SNR$=20.59; (d) 7,24e-6, $SNR$=17.50.*

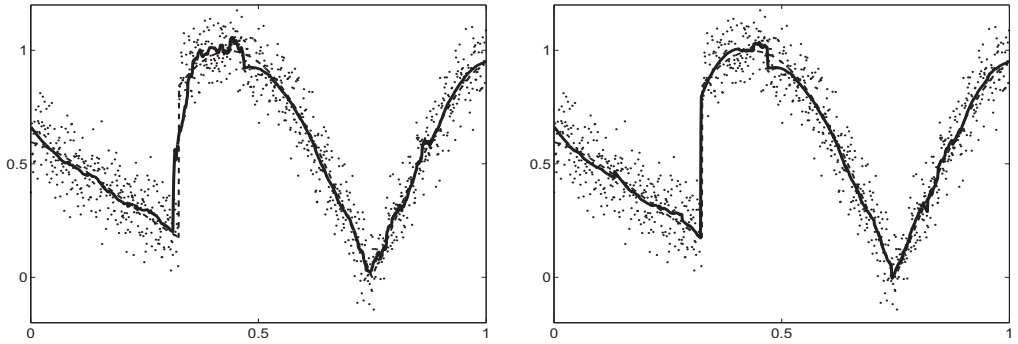


(a) $\overline{\mathcal{IC}}$ + $SR$ (b) $\overline{\mathcal{IAC}}$+$SR$ con $E_2$

(c) $\overline{\mathcal{AC}}$+$SR$ con $E_2$ (d) $\overline{\mathcal{IAC}}$+$SR$ con (2.51) y $E_2$

**Figura 5.8:** *Decimamos 3 niveles mediante medias en celda y subimos 3 veces según cada reconstrucción, con* $r = 3$ *y* $nl = nr = 3$ *(excepto en (a)), para la función (1.70) con 1024 puntos y* $SNR = 20$*. La línea discontinua representa la función sin ruido, los puntos son la función con ruido añadido y la línea continua es la reconstrucción obtenida. Errores: (a)* $ECM$*=9.42e-6,* $SNR$*=16.31; (b)* $ECM$*=9.04e-6,* $SNR$*=16.50; (c)* $ECM$*=1.02e-5,* $SNR$*=15.99; (d)*$SNR$*=1.12e-5,* $SNR$*=15.53.*

jorar la aproximación en las proximidades de las discontinuidades hemos aplicado las técnicas *ENO* y *SR*, evitando crear coeficientes de escala de magnitud elevada y poder así eliminarlos sin graves consecuencias. Trabajaremos un enfoque diferente, empleado en *wavelets*, basado en cortar los coeficientes de altas frecuencias que superen un cierto umbral. Dicho umbral deberá eliminar los coeficientes que provienen de ruido y mantener aquellos que surgen alrededor de las discontinuidades. Esto se conoce como *Wavelet Shrinkage Denoising*. En lo que sigue analizaremos estos



(a) $\overline{\mathcal{IAC}}$ y diferencias divididas (b) $\overline{\mathcal{IAC}}$+$SR$ y $E_2$

**Figura 5.9:** *Decimamos 5 niveles mediante medias en celda y subimos 5 veces según cada reconstrucción, con* $r = 3$ *y* $nl = nr = 3$*, para la función (1.70) con 1024 puntos y* $SNR = 20$*. La línea discontinua representa la función sin ruido, los puntos son la función con ruido añadido y la línea continua es la reconstrucción obtenida. Errores: (a)* $ECM$*=1.08e-5,* $SNR$*=15.70; (b)* $ECM$*= 9.98e-6,* $SNR$*=16.06.*

métodos y estudiaremos la viabilidad de adaptarlos a la multirresolución *à la Harten*.

## *Wavelet Shrinkage Denoising*

Una de las técnicas comúnmente utilizadas para la eliminación de ruido en señales se basa en la utilización de las transformadas *wavelet*, [63], [70], [35], [23], [41]. Es por ello que no podemos obviarlas si de eliminar ruido se trata. Ciertos investigadores han afirmado que el método *wavelet shrinkage*[4] ofrece "todo cuanto podemos desear de una técnica, desde generalidad hasta optimalidad" [36], aunque esta afirmación es sólo aceptable bajo la teoría asintótica, ya que en experimentos extraídos del mundo real debemos ser más cautos.

[4] *Wavelet shrinkage denoising* no debe ser confundido con suavizado (*smoothing*), aunque algunos autores utilicen *smoothing* como sinónimo de *denoising*. Mientras que *smoothig* elimina las altas frecuencias y mantiene las bajas, *denoising* intenta eliminar sólo el ruido que contamina la señal original, preservando sus frecuencias. Por ejemplo, al eliminar ruido en música, queremos mantener las altas (*treble*) y bajas (*bass*) frecuencias.

En 1984, Donoho y Johnstone establecieron el algoritmo *Wavelet Shrinkade Denoising* como una nueva herramienta de eliminación de ruido. Básicamente consiste en tres pasos: aplicación de una trasformada *wavelet* directa, una eliminación no lineal de ruido y finalmente la aplicación de la trasformada *wavelet* inversa correspondiente. Aunque las transformadas *wavelet* son métodos lineales, el segundo paso indica que el método es no lineal.

En términos de la transformada *wavelet* rápida, según las notaciones (1.77) y (1.77), si $W(\cdot)$ y $W^{-1}(\cdot)$ denotan los operadores *wavelet* directo e inverso y $T(\cdot, \lambda)$ es el operador de umbralizado, el proceso en el nivel $j$ es el siguiente, (Figura 5.10):

1. Aplicamos la Transformada *Wavelet* Directa a los coeficientes de bajas frecuencias $W(\{\alpha_{j,i}\}_i) = (\{\alpha_{j-1,i}\}_i, \{\beta_{j-1,i}\}_i)$.

2. Aplicamos el Operador de Umbralizado (*shrink*) a los coeficientes de altas frecuencias $T(\{\beta_{j-1,i}\}_i, \lambda) = \{\hat{\beta}_{j-1,i}\}_i$.

3. Aplicamos la Transformada *Wavelet* Inversa con los coeficientes de altas frecuencias obtenidos: $W^{-1}(\{\alpha_{j-1,i}\}_i\}, \{\hat{\beta}_{j-1,i}\}_i) = \{\hat{\alpha}_{j,i}\}_i$.

Los coeficientes $\{\hat{\alpha}_{j,i}\}_i$ son una aproximación a $\{\alpha_{j,i}\}_i$ con un menor contenido en ruido.

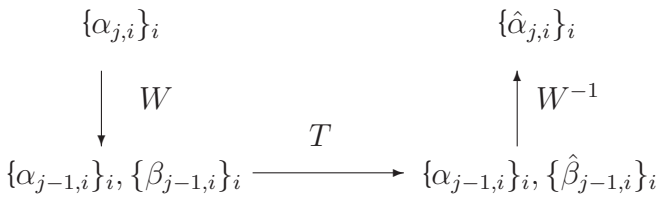



**Figura 5.10:** *Un paso de Wavelet Shrinkage Denoising.*

El operador de umbralizado $T(\cdot, \lambda)$ se define de tres maneras:

**Lineal**

$$T(U, \lambda) = 0 \quad \forall\ \lambda. \tag{5.7}$$

**Hard-threshold**

$$T(U, \lambda) = \begin{cases} U & \text{si } |U| > \lambda, \\ 0 & \text{en otro caso.} \end{cases} \tag{5.8}$$

**Soft-threshold**

$$T(U,\lambda) = sgn(U)max(0,|U| - \lambda). \tag{5.9}$$

*Hard-threshold* en apariencia es más intuitivo, sin embargo la continuidad de *soft-threshold* tiene varias ventajas, ya que produce algoritmos mejor tratables matemáticamente. Además, los coeficiente de ruido puro en ocasiones pueden pasar el *hard-threshold* y aparecer como saltos en los resultados. Por estos motivos se utiliza mayoritariamente *soft-threshold* en *wavelet shrinkage*.

La elección de $\lambda$ juega un papel determinante en el diseño del método. Existen numerosos trabajos para definir un $\lambda$ apropiado, como *TAWS*, [71], *Firm Shrink*, [40], *Oracle Shrink*, [70], *nonnegative Garrote*, [39]. También podemos encontrar diversas transformaciones, como *Ridgelets*, [24], [61], *Contourlets*, [57], [33], *Curvelets*, [67]. El analizar y comparar los diferentes métodos existentes sería una tarea excesivamente extensa y constituiría otro trabajo independiente por sí mismo. Lo que pretendemos es conocer las técnicas básicas de *Wavelets Shrinkage* y estudiar su posible aplicación a la multirresolución *à la Harten*. Es por ello que en este apartado nos centraremos en los tres métodos básicos y más utilizados de determinación de $\lambda$: *VisuShrink*, *SureShrink* y *BayeShrink*.

***VisuShrink***

Este método se describió en la sección 3.3.2. Recordemos que el umbral utilizado es $\lambda_U = \sigma\sqrt{2log(N)}$, siendo $N$ el tamaño de la señal original y $\sigma^2$ la varianza del ruido, Suponemos que el ruido que hemos añadido es *blanco*, *gaussiano* y sigue una $N(0,\sigma^2)$. El valor de $\sigma$ se estima mediante $\sigma \approx \frac{MAD}{0{,}6475}$ con $MAD$ la mediana de las magnitudes de todos los coeficientes de altas frecuencias en la escala más fina.

El $\lambda_U$ es óptimo en sentido asintótico [56], aunque si lo aplicamos a imágenes reales observamos un efecto de emborronado en los bordes lo cual nos indica que ciertos coeficientes que no provienen de ruido son eliminados (o suavizados). Cuantos más niveles se utilicen, más evidente es este efecto. En la práctica, dado que no trabajamos con tamaños de la señal que tienden a $\infty$, podemos buscar mejores candidatos de $\lambda$.

***SureShrink***

*SURE*, *Stein's Unbiased Risk Estimator*, [51], fue desarrollado por Donoho y Johnstone en 1995 y se basa en calcular un umbral diferente para cada nivel. Si $\beta = \{\beta_i\}_{i=1,\dots,n}$ son los coeficientes de altas frecuencias del nivel de resolución en el que nos hallamos, el umbral *SURE* se calcula a partir del resultado de Stein [51], :

$$SURE(t;\beta) = n - 2 \cdot \#\{i : |\beta_i| < t\} + \sum_{i=1}^{n} min(|\beta_i|, t)^2. \tag{5.10}$$

El estimador imparcial de riesgo de Stein, ofrece una estimación del riesgo para un valor particular del umbral $\lambda$. El umbral para *SURE*, $\lambda_S$, es el valor de $t$, menor que el corte universal, $\lambda_U$, que minimiza la anterior función:

$$\lambda_S = \text{argmin}_{0<t<\lambda_U} SURE(t;\beta). \tag{5.11}$$

Este problema de optimización es computacionalmente sencillo. Sin pérdida de generalidad, podemos ordenar $\{|\beta_i|\}_{i=1,\dots,n}$ en orden creciente. En los intervalos donde $t$ se halla entre dos valores de $|\beta_i|$, $SURE(t)$ es estrictamente creciente. Con lo cual, el mínimo valor de $\lambda_S$ es uno de los $|\beta_i|$. Como máximo hay $n$ valores y $\lambda_S$ se puede obtener mediante $O(n\ \log(n))$ operaciones.

NOTA 5.1. *Algebraicamente, para la deducción de SURE se necesita que la función umbralizadora sea Lipschitz. Por tanto SURE sólo puede utilizarse bajo soft-threshold, [62].*

***BayeShrink***

Mediante este método se realiza una estimación del valor óptimo que minimiza el Riesgo Bayesiano suponiendo una Distribución Gaussiana Generalizada, [70]. En este caso se utiliza un *threshold* diferente para cada nivel y está diseñado para ser utilizado con *soft-threshold*. Denotamos $X$,$Y$ y $V$ a las transformadas *wavelet* de la señal sin ruido, con ruido y de las componentes de ruido respectivamente. Podemos expresar que $Y = X + V$. Dado de

$X$ y $V$ son mutuamente independientes, las varianzas $\sigma_x$, $\sigma_y$ y $\sigma_v$ de $X$, $Y$, $V$ cumplen: $\sigma_y^2 = \sigma_x^2 + \sigma_v^2$, donde $v$ sigue una $N(0, \sigma_v^2)$.

En primer lugar debemos estimar la varianza del ruido mediante el estimador robusto $\hat{\sigma}_v = \frac{MAD}{0{,}6745}$. En segundo lugar estimamos la varianza de la señal con ruido: $\hat{\sigma}_y^2 = (\frac{1}{M}\sum_{i=1}^{M} \beta_{j,i}^2)^2$, donde $\{\beta_{j,i}\}_{i=1,\ldots,M}$ son los coeficientes de altas frecuencias del nivel $j$.

El umbral que se utiliza es el siguiente, [70]: $\frac{\hat{\sigma}_v^2}{\hat{\sigma}_x}$, donde $\hat{\sigma}_x = \sqrt{max\left(\hat{\sigma}_y^2 - \hat{\sigma}_v^2, 0\right)}$.

Notar que si $\hat{\sigma}_y^2 \leq \hat{\sigma}_v^2$ entonces $\lambda_B \to \infty$ y tendríamos problemas al calcularlo. En la práctica si esto ocurre, elegimos $\lambda^B = \max_i\{|\beta_{j,i}|\}$, con lo que todos los coeficientes se igualan a 0.

Resumiendo, el corte utilizado para *BayeShrink* es como sigue:

$$\lambda_B = \begin{cases} \frac{\hat{\sigma}_v^2}{\hat{\sigma}_x}, & \text{si } \sigma_y^2 > \sigma_v^2, \\ \max_i\{|\beta_{j,i}|\}, & \text{en otro caso.} \end{cases}$$

## Ejemplos

Aplicaremos *VisuShrink*, *SureShrink* y *BayeShrink* a ejemplos concretos mediante la base *wavelet DB4*, que es adecuada para la eliminación de ruido, [53]. Como vimos en el capítulo 1 para aplicar la transformada *wavelet* y mantener la ortogonalidad de las matrices de coeficientes, podemos considerar que la función es periódica, estando formada por copias de sí misma. Así pues, las fronteras se considerarían como discontinuidades y al cortar los coeficientes de altas frecuencias podrían producirse malas aproximaciones en ellas. Dado que nos interesa comparar el *ECM* de estas soluciones con las obtenidas mediante multirresolución, es preferible eliminar dicho error para que no afecte al cómputo total del *ECM*. Una posible solución consiste en extender a izquierda y derecha los valores. Por ejemplo, para *DB4*, si los datos iniciales son $\{f_i\}_{i=1,\ldots,n}$, pasamos a utilizar $\{f_1, f_1, f_1, f_2, \ldots, f_{n-1}, f_n, f_n, f_n\}$. Esto lo realizamos en cada nivel, evitando la mala aproximación en las fronteras. Si la función no contuviese ruido, la solución más elegante sería utilizar en las fronteras la técnica *ENO-wavelet* vista en el capítulo anterior, pero las extrapolaciones de datos con ruido también pueden afectar al error.

En la Figura 5.11 hemos aplicado 5 niveles de *DB4* a una función con 1024 puntos iniciales a la que hemos añadimos ruido de magnitud $SNR = 25$. En (a) eliminamos todos los coeficientes de altas frecuencias. Hay una notable reducción de ruido pero el salto queda suavizado. En (b), (c) y (d) aplicamos *VisuShrink*, *SureShrink* y *BayeShrink* respectivamente, utilizando *soft-threshold*. En este caso la aproximación de los saltos mejora. Aunque *BayeShrink* consigue mejores resultados desde el punto de vista de la *ECM* y *SNR*, visualmente es mucho mejor *VISU* o *SURE*. Se debe a que estamos comparando la señal con ruido con la reconstrucción y no la señal sin ruido y la reconstrucción. En cualquier caso, aunque *ECM* y *SNR* son medidas objetivas, es bien conocido que para comprobar el resultado de la eliminación de ruido la impresión visual es también un dato a considerar. En este caso los mejores resultados se obtienen con *SureShrink*.

En la Figura 5.12 mostramos los coeficientes de altas frecuencias y los umbrales según cada método en un nivel concreto.

## 5.3.3

## Multirresolución *à la Harten* con *Shrinkage*

En esta sección estudiaremos la viabilidad de adaptar las técnicas utilizadas en *Wavelet Shrinkade Denoising* para multirresolución *à la Harten*. Según las notaciones del capítulo 1, deberemos cortar los coeficientes de escala $d_i^k$ de modo similar a los coeficientes de altas frecuencias, $\beta_{j,i}$.

Si los datos de entrada en el nivel $k$ vienen representados por $v^j$, un paso del proceso se reduce a:

1. Mediante decimación (1.3) bajamos de nivel, es decir, $v^{k-1} = D_k^{k-1} v^k$ y $d^k = G_k Q_k v^k$, siendo los operadores $D, G, Q$ los definidos en la sección 1.1.

2. Aplicamos el operador de umbralizado a los coeficientes de escala: $T(d^k, \lambda) = \hat{d}^k$

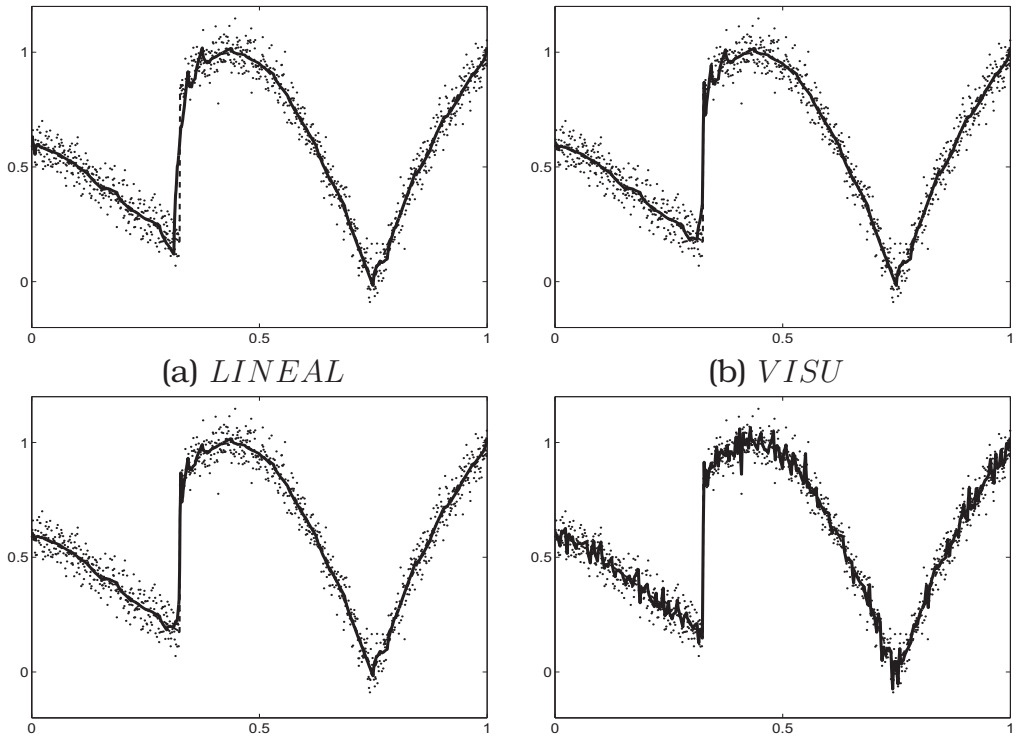



**Figura 5.11:** *Tomando 1024 puntos de la función (1.70) con $SNR = 25$, aplicamos 5 niveles de la transformada DB4, donde los coeficientes de altas frecuencias se modifican mediante soft-threshold según cada umbral. La línea discontinua representa la función sin ruido, los puntos son la función con ruido añadido y la línea continua es la reconstrucción obtenida. Errores: (a) $ECM$=4.37e-6, $SNR$=19.62; (b) $ECM$= 3.32e-6, $SNR$=20.81; (c) $ECM$=3.17e-6, $SNR$=21.02; (d) $ECM$= 2.41e-6, $SNR$=22.21.*

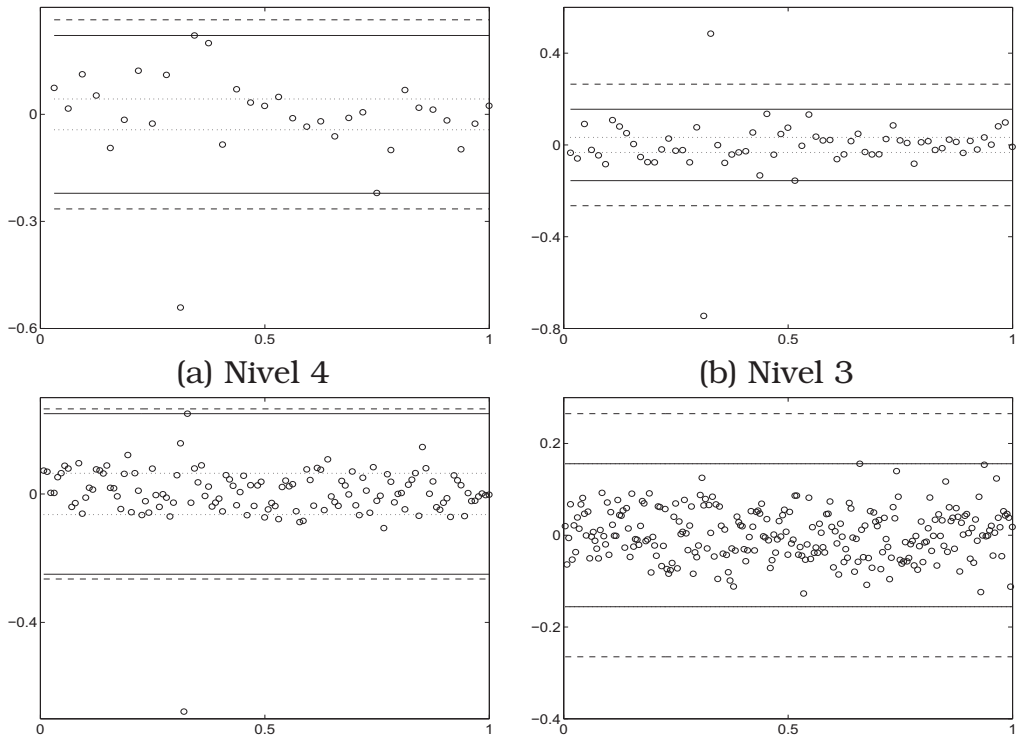
3. Finalmente aplicamos predicción con los nuevos coeficientes de escala: $P_{k-1}^{k} v^{k-1} + E_k \hat{d}^k = \hat{v}^k$.

El problema se reduce a determinar el $\lambda$ adecuado en el operador $T(\cdot, \lambda)$. En este caso los $\lambda$'s utilizados en *wavelets* no son aplicables directamente, ya que los coeficientes *wavelet* de altas frecuencias y los coeficientes de escala son conceptos distintos.

Antes de buscar matemáticamente un umbral, lo hacemos experimentalmente. En base a los resultados de eliminación de ruido que obtengamos decidiremos si es provechoso el cálculo teórico de

dicho umbral. Al igual que en apartados anteriores trabajamos con medias en celda ya que al bajar niveles se elimina ruido. Para buscar el umbral óptimo, partiendo de la función (1.70) con 1024 puntos y $SNR = 25$, tomamos valores de $\lambda$ de 0 a 0.5 en intervalos de 0.01 y representamos el *ECM* y el *SNR* entre la reconstrucción obtenida y la función sin ruido. Obtenemos la Figura 5.13, donde el umbral óptimo para este ejemplo concreto es $\lambda = 0{,}05$.

En la Figura 5.14 (a), mostramos el resultado de aplicar 5 niveles con reconstrucción interpolatoria y medias en celda utilizando *soft-threshold* con umbral $0{,}05$. Observamos que no se produce *fenómeno de Gibbs* y por tanto los coeficientes de escala que provienen de *stencils* que cruzan el salto, han sido eliminados. Hay




(a) Nivel 4 (b) Nivel 3

(c) Nivel 2 (d) Nivel 1

**Figura 5.12:** *Tomamos 1024 valores de la función (1.70) con $SNR = 25$. Representamos los coeficientes de altas frecuencias ('o') y los umbrales $\lambda_U$ (línea continua), $\lambda_S$ (línea discontinua) y $\lambda_B$ (línea punteada), según varios niveles. En el nivel 1 $\lambda_U$ y $\lambda_B$ coinciden.*

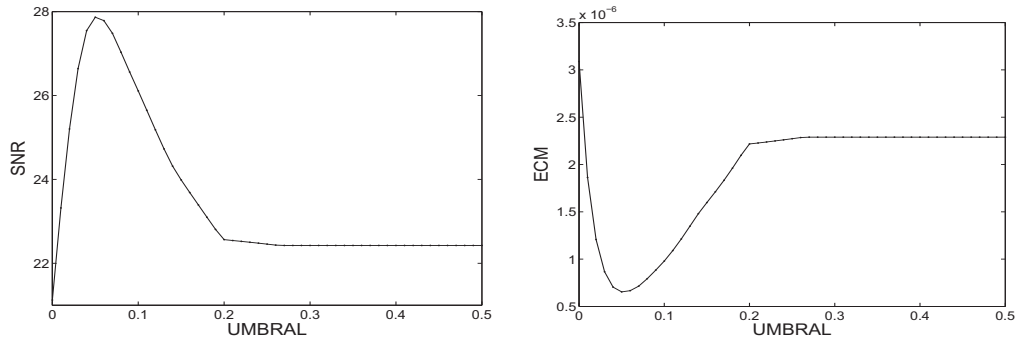



**Figura 5.13:** *Aplicamos 5 niveles de multirresolución por medias en ceda con reconstrucción interpolatoria y soft-threshold a la función (1.70) con 1024 puntos y $SNR = 25$. Izquierda: umbral vs SNR entre la reconstrucción obtenida y la función sin ruido. Derecha: umbral vs ECM.*

eliminación de ruido, pero menor que con otros métodos vistos en apartados anteriores. Para mejorar los resultados podemos buscar un umbral para cada nivel, como se hace en *SURE*. Nuevamente, procediendo de forma experimental, hemos obtenido los umbrales $[0{,}01, 0{,}02, 0{,}04, 0{,}06, 0{,}1]$, siendo $0{,}1$ el umbral para el nivel más grosero. Con dichos umbrales obtenemos la Figura 5.14 (b) donde la eliminación de ruido ha mejorado.

En la Figura 5.14 (c) y (d) hemos utilizado la reconstrucción $\overline{\mathcal{IAC}}$. En (c) el umbral es el mismo para todos los niveles, mientras que en (d) varía para cada nivel. Dichos umbrales se calculan experimentalmente y en este ejemplo concreto coinciden con el caso de reconstrucción interpolatoria.

En la Figura 5.15 hemos utilizado la reconstrucción $\overline{\mathcal{AC}}$ con un umbral para todos los niveles (a) y con umbrales adaptados (b). En este caso la eliminación de ruido es respecto de las otras reconstrucciones.

Los mejores resultados se obtienen para $\overline{\mathcal{IC}}$ y $\overline{\mathcal{IAC}}$, aunque no mejoran sustancialmente a la aplicación directa de las reconstrucciones eliminando todos coeficientes de escala (umbral lineal) y aplicando las técnicas *ENO-SR*. Una posible ventaja podría ser si se observase una mejora al aplicar menos niveles, ya que con los métodos anteriores necesitamos bajar un suficiente número de niveles. Sin embargo, en la Figura 5.15 (c) y (d) hemos repetido algu-

nos experimentos de este apartado con 3 niveles, obteniendo una escasa eliminación de ruido. Por tanto desestimamos la búsqueda teórica de umbrales óptimos para los coeficientes de escala ya que la eliminación de ruido es similar a otros métodos ya estudiados, aunque con los experimentos expuestos hemos comprobado la viabilidad de la adaptación de las ideas de *Wavelets Shinkage Denoising* a descomposiciones multiescala.

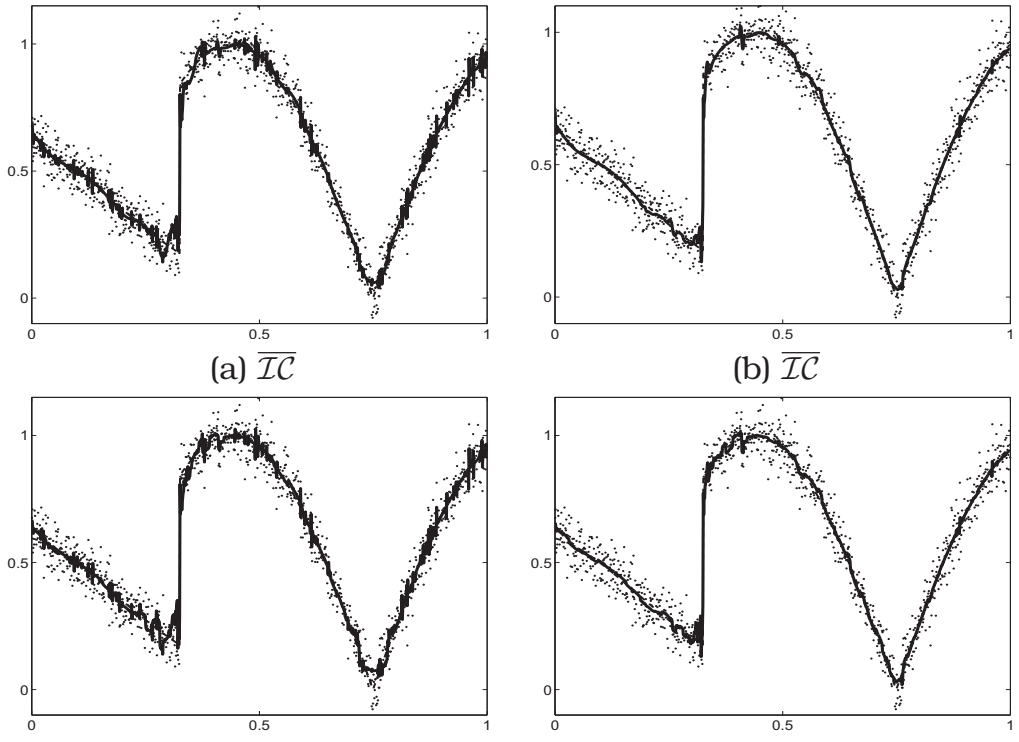
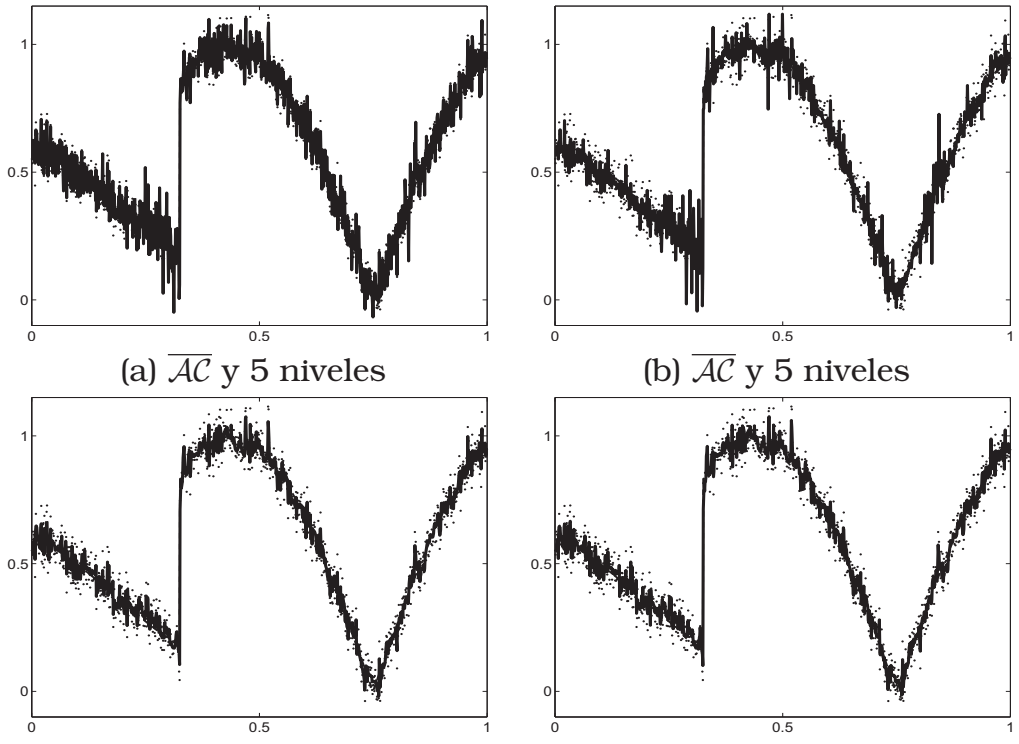



**Figura 5.14:** *Partimos de (1.70) con 1024 puntos y* $SNR = 25$*. Aplicamos cinco niveles de multirresolución por medias en ceda con las reconstrucciones indicadas y soft-threshold. En (a) y (c) usamos el umbral* $\lambda = 0{,}05$ *para todos los niveles. En (b) y (d) utilizamos un umbral diferente para cada nivel* $\lambda_0 = 0{,}01$*,* $\lambda_1 = 0{,}02$*,* $\lambda_2 = 0{,}04$*,* $\lambda_3 = 0{,}06$*,* $\lambda_4 = 0{,}1$*. La línea discontinua representa la función sin ruido, los puntos son la función con ruido añadido y la línea continua es la reconstrucción obtenida. Errores: (a)* $ECM$*=2.79e-6,* $SNR$*=21.55; (b)* $ECM$*=3.01e-6,* $SNR$*= 21.22; (c)* $ECM$*=2.87e-6,* $SNR$*=21.43; (d)* $ECM$*=3.02e-6,* $SNR$*=21.24.*

## 5.3.4 *ENO-wavelets*

Para aplicar *ENO-DB2p-TI* a una función con ruido, debemos introducir las siguientes modificaciones:

- Aplicar un detector de discontinuidades válido para funciones



(a) $\overline{\mathcal{AC}}$ y 5 niveles (b) $\overline{\mathcal{AC}}$ y 5 niveles

(c) $\overline{\mathcal{IC}}$ y 3 niveles (d) $\overline{\mathcal{IAC}}$ y 3 niveles

**Figura 5.15:** *Tomando 1024 puntos de (1.70) con $SNR = 25$, decimamos 5 niveles por medias en ceda y subimos con las reconstrucciones indicadas con soft-threshold. En (a) usamos $\lambda = 0{,}01$ para todos los niveles. En (b) utilizamos un umbral diferente para cada nivel: $\lambda_0 = 0{,}01$, $\lambda_1 = 0{,}01$, $\lambda_2 = 0{,}03$, $\lambda_3 = 0{,}04$, $\lambda_4 = 0{,}07$; al igual que en (c) y (d). La línea discontinua representa la función sin ruido, los puntos son la función con ruido añadido y la línea continua es la reconstrucción obtenida. Errores: (a) $ECM$=1.20e-6, $SNR$=25.28 ; (b) $ECM$=3.26e-6, $SNR$=20.92; (c) $ECM$=1.09e-6, $SNR$= 25.68; (d) $ECM$=1.08e-6, $SNR$=25.70.*

con ruido, como por ejemplo $\overline{\mathcal{AP}} - SC$, $E_1\&E_2$ o *VISU*.

- Realizar las extrapolaciones mediante polinomios de mínimos cuadrados, como $q^{\overline{\mathcal{AP}}}$ o de interpolación aproximación, $q^{\overline{\mathcal{IAP}}}$.

*ENO-DB2p-TI* con las anteriores modificaciones lo denotaremos *ENO-DB2p-TI-N*. Siempre que la detección de discontinuidades sea correcta, no se van a crear coeficientes de altas frecuencias de magnitud elevada. Así pues, no es necesario ningún procesado posterior de los coeficientes de altas frecuencias, como *soft* o *hard threshold*, y la eliminación total de los coeficientes de altas frecuencias dará como resultado la eliminación del ruido.

Veamos algún ejemplo. En la Figura 5.16 hemos aplicado el método *ENO-DB2p-TI-N* con 5 niveles a funciones con uno, (1.44), y dos, (3.15), saltos. Hemos utilizado dos detectores de discontinuidades: $\overline{\mathcal{AP}} - SC$ y *VISU*, vistos en la sección 3.3. Para $p = 2$ la extrapolación se ha realizado con 4 nodos y grado 1, y para $p = 3$ con 5 nodos y grado 2. Si observamos las Figuras (a), (b), (c) y (d), los mejores resultados se obtienen para *DB4* con detector *VISU*. En ese caso no se produce fenómeno de Gibbs y la reconstrucción es muy próxima a lo buscado. Sin embargo, este resultado no es extrapolable en general. Observemos la Figura (e), donde la discontinuidad derecha se ha prolongado en exceso debido a las extrapolaciones de grado 1, perdiendo la curvatura de la señal original. La solución no consiste en aumentar el grado de la extrapolación, como en (f), ya que a medida que aumentamos el grado las extrapolaciones también distan más de lo esperado. Si aplicamos pocos niveles las soluciones son buenas, pero la eliminación de ruido es escasa. Los pequeños errores de extrapolación a partir de datos con ruido se van acumulando al subir de niveles lo cual provoca pérdida de aproximación.

El método necesita localizar con exactitud la posición de los saltos y los detectores que manejamos no son excesivamente fiables. Además, para *DB6*, si analizamos los coeficientes de altas frecuencias podemos encontrarnos que hay un sólo coeficiente de magnitud elevada, aunque teóricamente las únicas opciones serían 2 o 3 coeficientes consecutivos afectados, dificultando de este modo la localización exacta del salto.

Por estos motivos, concluimos que la aplicación de este método, tal como se ha diseñado, es poco fiable para la eliminación de ruido, debiéndose mejorar el modo de realizar las extrapolaciones y las detecciones para un uso práctico.

## 5.4 Comparativa

En esta sección aplicaremos a una misma función los métodos en los que hemos obtenido mejores resultados en cuanto a eliminación de ruido según cada apartado[5]. Esto no significará que el método sea el mejor y debamos desechar el resto, ya que hay situaciones en que debemos utilizar uno en concreto. Por ejemplo si la función de partida tiene 128 puntos, los métodos por multirresolución son inviables, ya que no podremos bajar suficientes niveles. En cualquier caso para realizar esta comparativa partiremos de una función con 1024 puntos a la que añadimos ruido de magnitud $SNR = 25$. Los métodos que compararemos son:

**M1** *AMC* entre discontinuidades, con localizador de discontinuidades $\overline{\mathcal{AP}} - SC$.

**M2** $\overline{\mathcal{AP}}$ punto a punto con *stencil* centrado excepto en las discontinuidades (localizadas con $\overline{\mathcal{AP}} - SC$).

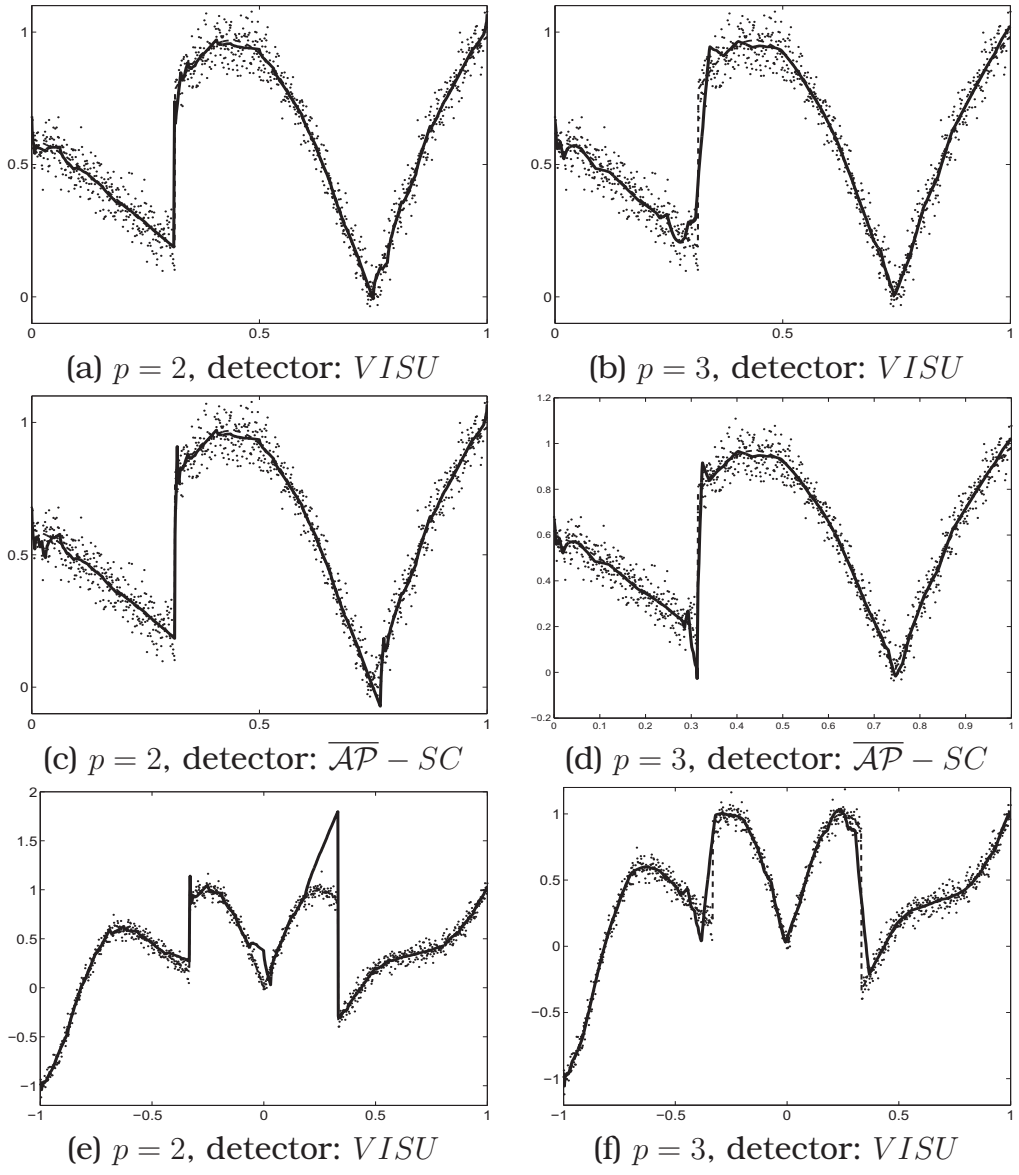
**M3** 10 iteraciones por sustitución de valores medios a izquierda y derecha.

**M4** 5 niveles de $\overline{\mathcal{IAC}}$ con *SR*, evitando cruzar discontinuidades con $E_2$.

**M5** 5 niveles de *SureShrink* con base *DB4*.

**M6** 5 niveles de $\overline{\mathcal{IAC}}$, con *soft-threshold* y umbral adaptado a cada nivel (calculado de forma experimental).

[5]Desde http://www.uv.es/anims/noguera/ podemos descargar una GUI en MATLAB® que permite realizar experimentos de eliminación de ruido según los datos introducidos por el usuario.



(a) $p = 2$, detector: $VISU$ (b) $p = 3$, detector: $VISU$

(c) $p = 2$, detector: $\overline{\mathcal{AP}} - SC$ (d) $p = 3$, detector: $\overline{\mathcal{AP}} - SC$

(e) $p = 2$, detector: $VISU$ (f) $p = 3$, detector: $VISU$

**Figura 5.16:** *Aplicamos 5 niveles de ENO-DBp-TI-N con los detectores y $p$ que se indican, a las funciones (1.44) ((a), (b), (c), (b)) y (3.15) ((e), (f)) con 1024 puntos y $SNR = 25$. Se realiza una aproximación lineal. La función original sin ruido se representa mediante línea discontinua, con ruido mediante puntos y la reconstrucción mediante línea continua. Errores: (a) $ECM$=3.45e-6, $SNR$=20.60; (b) $ECM$=3.87e-6, $SNR$=20.10; (c) $ECM$=3.09e-6, $SNR$=21.09; (d) $ECM$=3.62e-6, $SNR$=20.36; (e) $ECM$ =2.44e-5, $SNR$=12.49; (f) $ECM$=1.09e-5, $SNR$=15.02.*

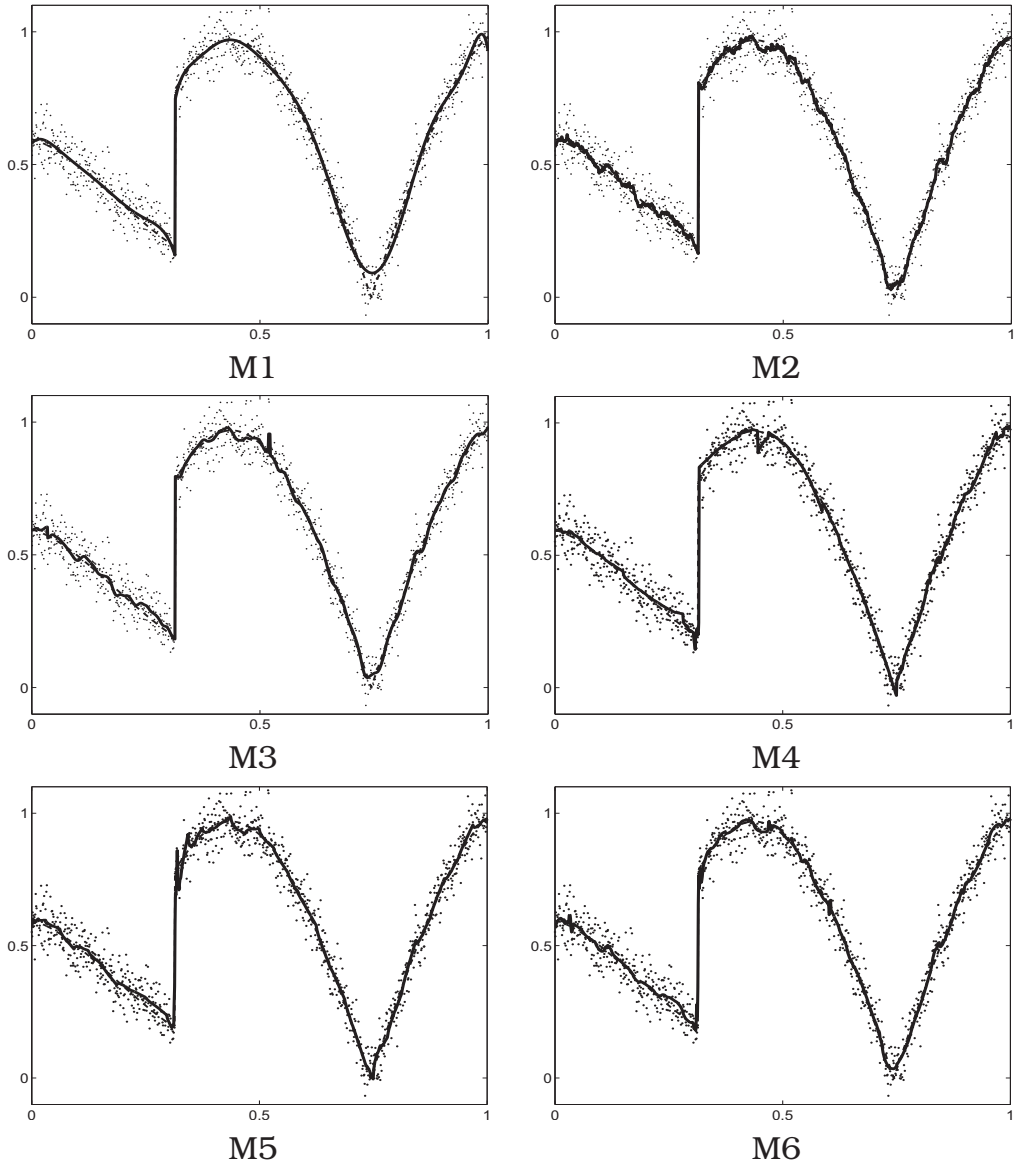



**Figura 5.17:** *Aplicación de diferentes métodos de eliminación de ruido especificados en la sección 5.4 aplicados a la función (1.70) con 1024 puntos y ruido de magnitud $SNR = 25$. La línea discontinua representa la función sin ruido, los puntos son la función con ruido añadido y la línea continua es la reconstrucción obtenida. La Tabla 5.1 muestra el $ECM$ y $SNR$ de cada método.*

| | Entre $\tilde{f}$ y $\hat{f}$ | | Entre $f$ y $\hat{f}$ | |
|---|---|---|---|---|
| | $ECM$ | $SNR$ | $ECM$ | $SNR$ |
| M1 | 3.2499e-6 | 20.1955 | 3.4076e-7 | 30.6755 |
| M2 | 2.8282e-6 | 21.5024 | 2.7289e-7 | 31.6401 |
| M3 | 2.7100e-6 | 21.6829 | 2.1780e-7 | 32.5345 |
| M4 | 3.6203e-6 | 20.4396 | 8.2422e-7 | 26.8396 |
| M5 | 3.0380e-6 | 21.1859 | 3.3788e-7 | 30.7124 |
| M6 | 2.8344e-6 | 21.4930 | 2.8918e-7 | 31.3884 |

**Tabla 5.1:** *$SNR$ y $EMC$ entre la función original sin ruido ($f$) y la reconstrucción ($\hat{f}$); y entre la función original con ruido ($\tilde{f}$) y la reconstrucción, para los experimentos de la Figura 5.17.*

La Figura 5.17 muestra las reconstrucciones obtenidas. Los errores $ECM$ y $SNR$ son de orden similar en todos los casos, Tabla 5.1, aunque hay diferencias notables para decidir la aplicación de uno u otro método:

***AMC* entre discontinuidades**. Lo fundamental del método es la localización de los saltos. Si esto no se cumple la solución obtenida es inaceptable. Por tanto evitaremos su aplicación en señales donde el método de detección pueda fallar, por ejemplo si el ruido es elevado, si tenemos gran número de discontinuidades y no están lo suficientemente separadas,...

$\overline{\mathcal{AP}}$ **punto a punto**, con *stencil* centrado excepto en las discontinuidades (localizadas con $\overline{\mathcal{AP}} - SC$). El método también está sujeto a una localización previa de saltos. Sin embargo un fallo no va a provocar una pérdida total de las características de la función, sino un suavizado del salto. Para evitarlo se podría elegir el *stencil* según $E_1$ o $E_2$, pero como vimos se eliminaba poco ruido. El método es aplicable con escasos nodos iniciales, siendo la única restricción que los saltos estén suficientemente separados, como mínimo que entre dos discontinuidades haya un número de nodos igual al tamaño del *stencil* usado (preferiblemente mayor que 20 ya que para menos nodos no se elimina ruido eficientemente).

**Iteración por sustitución de de valores medios** a izquierda y derecha. El método también contiene un localizador de posibles discontinuidades, pero si falla sólo provoca un suavizado del salto. La mayor ventaja de este método reside en su bajo coste computacional (es de orden lineal en cada iteración) y que las discontinuidades sólo necesitan estar separadas por 1 punto.

$\overline{\mathcal{IAC}}$ **con** $SR$, evitando cruzar discontinuidades con $E_2$. No se necesita el uso de un localizador de discontinuidades. Es un método que proporciona buenos resultados, aunque tiene dos limitaciones. La primera es que las discontinuidades deben estar suficientemente separadas para aplicar *ENO* con $E_2$ en todos los niveles y la segunda es que partamos de un número suficiente de datos como para poder aplicar un mínimo de 4 niveles de multirresolución.

***SureShrink* con base *DB4*.** Es un método fiable, y su única restricción es que dispongamos de un tamaño de dados que permita utilizar suficientes niveles (mínimo 4) y que se cumpla la *DSP*.

$\overline{\mathcal{IAC}}$ **con *soft-threshold*** y umbral adaptado a cada nivel. Es similar a *SURE* con sus mismas ventajas e inconvenientes. No hemos calculado de forma teórica un umbral, sino de forma experimental con lo cual los resultados que mostramos corresponden a la mejor solución posible para el ejemplo expuesto. Para conseguir un método aplicable en la práctica se debería calcular el umbral de forma teórica ya que de forma experimental el coste computacional se dispara.

***ENO-wavelets*** con detector *VISU* para cada nivel. Es un método poco fiable en presencia de ruido y por ello no ha sido incluido en la comparativa. Se consiguen unos errores de aproximación similares a los anteriores pero el *fenómeno de Gibbs* aparece con frecuencia. Si se solucionasen estos problemas sería un buen método para eliminar ruido y comprimir de forma simultánea.

Con estas ideas y de forma simplificada, en la Figura 5.18 mostramos un diagrama que ayuda a decidir entre uno u otro método.

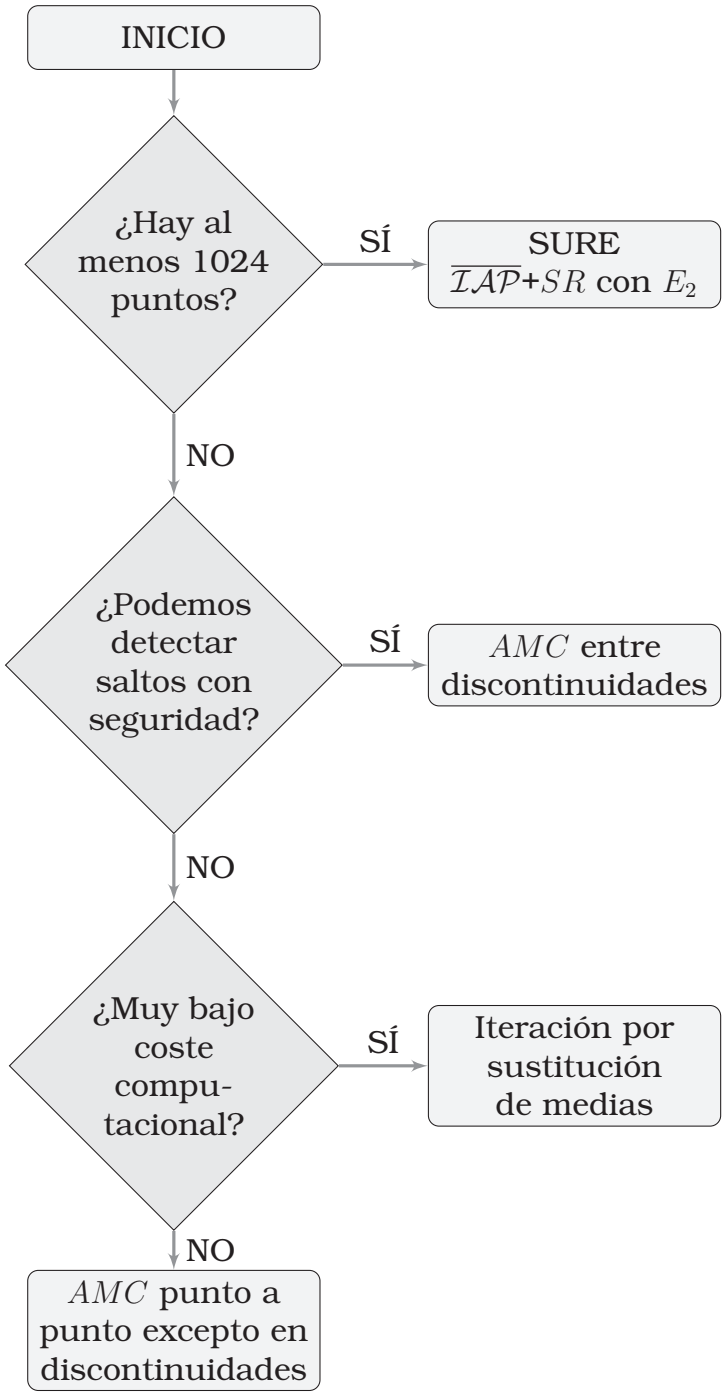



**Figura 5.18:** *Indicaciones para la elección de un método de eliminación de ruido.*

# 5.5 Conclusiones

En este capítulo hemos aplicado las herramientas matemáticas propuestas en capítulos anteriores a la eliminación de ruido. Hemos dividido este estudio en dos grandes bloques. El primero de ellos se ha basado en el estudio en una escala. Cuando la función de partida tiene pocos nodos ésta será nuestra única opción. Se han estudiado tres maneras de proceder. La primera se basa en una idea sencilla expuesta en [60], donde se sustituye cada trozo que proviene de una función suave por su aproximación por mínimos cuadrados. Los resultados son satisfactorios siempre que la localización de saltos no falle, en cuyo caso el método no es aplicable ya que los resultados distan en exceso de la función original. En segundo lugar se ha estudiado la idea de sustituir cada punto por su aproximación por mínimos cuadrados bajo un determinado *stencil*. Tras el estudio realizado se concluye que con estas ideas la mejor opción es utilizar *stencils* centrados excepto en las discontinuidades, que nuevamente deberán ser previamente detectadas. Para finalizar este bloque se propone un algoritmo iterativo extremadamente simple, pero que consigue una eliminación de ruido aceptable respetando las discontinuidades. Este algoritmo será una buena opción cuando nos interese un bajo coste computacional.

En un segundo bloque hacemos uso de las descomposiciones multiescala. La multirresolución *à la Harten* ha sido comúnmente aplicada a la compresión de señales pero en escasas ocasiones a la reducción de ruido. Hemos estudiado este tema adaptando las técnicas *ENO-SR* para mejorar la eficacia de los algoritmos en presencia de ruido. Trabajando con medias en celda y disponiendo de datos como para aplicar suficientes niveles, la reducción de ruido es satisfactoria, siendo la mejor opción la reconstrucción por *interpolación aproximación*, ya que permite una selección del *stencil* mediante la medida $E_2$ (en presencia de ruido más fiable que diferencias divididas) y proporciona una mayor reducción de ruido que la reconstrucción por *aproximación*.

Dado que la técnica más utilizada para la eliminación de ruido

es *Wavelet Shrinkage Denoising*, se han estudiado sus métodos básicos para compararlos con los anteriores y se ha comprobado la viabilidad de aplicación de estas ideas a las descomposiciones multiescala que manejamos en este trabajo.

Siguiendo la transformada *wavelet* se han introducido cambios en el esquema *ENO-wavelet* para poder adaptarlos a la eliminación de ruido. Su principal ventaja es que no es necesario un postproceso en los coeficientes de altas frecuencias y su punto débil reside en que los pequeños errores en detección de discontinuidades o en las extrapolaciones de datos con ruido se propagan y aumentan, disminuyendo la fiabilidad del método.

En definitiva, los esquemas de multirresolución *à la Harten*, bajo simples adaptaciones, son aplicables a la eliminación de ruido de funciones discontinuas compuestas por trozos suaves contaminadas por ruido *blanco gaussiano*, obteniendo resultados tan buenos como los que se utilizan normalmente es este campo, basados en *wavelets*.