3

Detección de discontinuidades

3.1 Introducción

La detección de discontinuidades en señales digitales es una materia de estudio que tiene interés por sí misma. Su utilidad abarca campos como el procesado digital de imágenes, tratamiento de señales acústicas y dinámica de fluidos, entre otras. En métodos de compresión e interpolación, la localización de las discontinuidades permite evitar el llamado *fenómeno de Gibbs*, adaptando los algoritmos cuando nos encontramos ante una discontinuidad. Nosotros lo aplicaremos como paso previo en problemas de eliminación de ruido.

Para la detección de discontinuidades existen multitud de métodos y enfoques pero nos centraremos en estudiar los relacionados con las teorías expuestas en el presente trabajo. Trataremos en primer lugar los métodos de detección basados en interpolación adaptativa *ENO* y en magnitud de coeficientes *wavelet* para funciones sin ruido. Los analizaremos detalladamente e introduciremos modificaciones para que localicen únicamente discontinuidades reales. En segundo lugar nos centraremos en funciones con ruido, para las cuales los métodos de funciones suaves no sirven. Estudiaremos algunos métodos específicos para dichas funciones, propondremos nuevos esquemas y realizaremos comparativas a fin de decidir cuál proporciona mejores resultados.

Es importante señalar que la localización de discontinuidades es independiente del uso posterior de esta información. Así pues, es interesante disponer de localizadores fiables de discontinuidades que puedan ser aplicados a un amplio abanico de problemas.

3.2
Métodos de detección de discontinuidades para funciones sin ruido

3.2.1
Interpolación adaptativa: técnica *ENO*

La técnica *ENO* permite seleccionar para cada intervalo de trabajo el *stencil* de menor diferencia dividida. Con esto se evita utilizar *stencils* que crucen discontinuidades, mejorando la aproximación alrededor de éstas. Una evolución de lo anterior es la técnica *SR* donde se localiza la posición de esquinas dentro de una celda singular y se extienden los interpolantes adyacentes hasta el punto de discontinuidad.

Las técnicas *ENO* y *SR* están diseñadas como métodos de mejora de la aproximación en discontinuidades y no como localizadores

de éstas. Si se produce una situación donde es posible una mejora al suponer que existe una discontinuidad, se aplica entonces una modificación del algoritmo estándar. Por tanto, el localizar algunas falsas discontinuidades no plantea un error grave. En esta sección utilizaremos el algoritmo *SR* para hallar la posición de las discontinuidades, introduciendo las modificaciones necesarias para que cubran mayor parte de situaciones de detección que los métodos de partida.

Algoritmos

En el capítulo 1 se describió la técnica *ENO* y *SR*. Volvemos aquí a exponer estas ideas pero centrándonos en la detección. Los algoritmos *ENO* se utilizan aplicando varios niveles, aunque aquí sólo nos interesa uno. En lo que sigue, $(x_i, f_i)_{i=1}^N$ representará la función de partida, $f_i = f(x_i)$, y $\mathcal{S}_i^{ENO} = \{x_{s_i-1}, \ldots, x_{s_i+r-1}\}$ el *stencil ENO* para el intervalo $I_i = [x_{i-1}, x_i]$.

Valores puntuales

Al trabajar con valores puntuales, las únicas discontinuidades que podemos localizar con seguridad serán las esquinas. Los saltos pueden no ser detectados debido a que la función $G_i(x)$ puede no cambiar de signo en los extremos de un intervalo que contenga un salto. El algoritmo que se sigue es:

1.- Para cada i, calcular el *stencil ENO* asociado ($\mathcal{S}_i^{ENO}$) mediante una elección jerárquica, Algoritmo 1.1 (*ENO-VP-I*) o no jerárquica, Algoritmo 1.2 (*ENO-VP-II*).

2.- Detectar las posibles *celdas singulares*: si $\mathcal{S}_{i-1}^{ENO} \cap \mathcal{S}_{i+1}^{ENO} = \emptyset$, etiquetar la celda $I_i = [x_{i-1}, x_i]$ como *sospechosa* de contener una singularidad.

3.- Para cada celda sospechosa definir la función

$$G_i^{\overline{\mathcal{IP}}}(x) := q_{i+1}^{\overline{\mathcal{IP}}}(x; f^{k-1}, r) - q_{i-1}^{\overline{\mathcal{IP}}}(x; f^{k-1}, r), \tag{3.1}$$

donde $q^{\overline{\mathcal{IP}}}$ es el polinomio que interpola a $f(x)$ en $\mathcal{S}_i^{ENO}$.

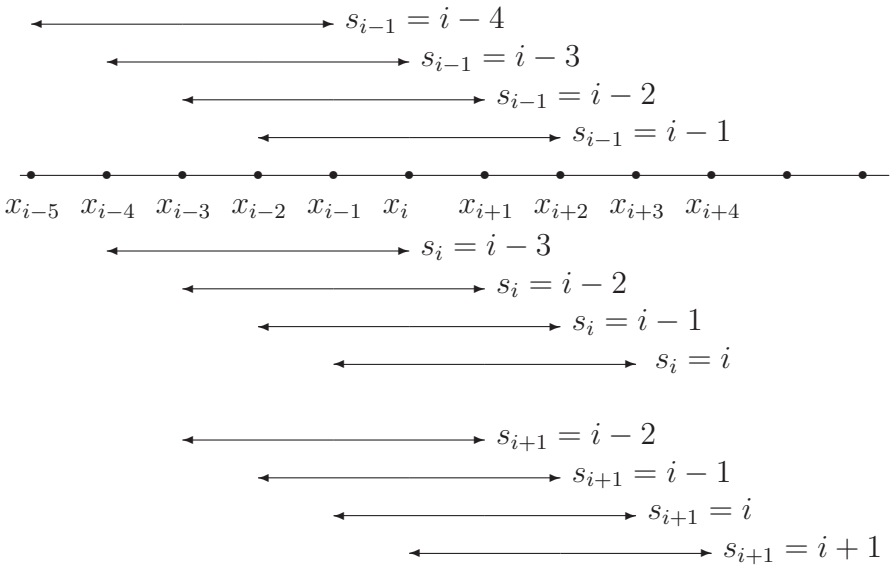

Figura 3.1: *Posibles elecciones para la técnica ENO de stencils de 5 nodos para los intervalos I_{i-1}, I_i y I_{i+1}.*

Si $G_i^{\overline{\mathcal{IP}}}(x_{i-1})G_i^{\overline{\mathcal{IP}}}(x_i) < 0 \Rightarrow$ existe una raíz θ_i de $G_i^{\overline{\mathcal{IP}}}(x)$, y etiquetamos la celda como *singular*, es decir, contiene una discontinuidad.

Mediante el algoritmo anterior podemos detectar las discontinuidades esquina que no se hallen en puntos del *grid*.

Supongamos, por ejemplo, que estamos trabajando con *stencils* de cinco nodos. Los diferentes *stencils* que podemos elegir para los intervalos I_{i-1}, I_i, I_{i+1}, los podemos ver en la Figura 3.1. Los casos donde la detección puede ser fallida son:

- Si en x_{i-1} hay una esquina y la elección *ENO* es tal que $s_{i-1} = i-4$ y $s_{i+1} = i$, se cumple que $\mathcal{S}_{i-1}^{ENO} \cap \mathcal{S}_{i+1}^{ENO} = x_{i-1}$ y por tanto la detección falla.

- Si en x_{i-1} hay una esquina, $G_i^{\overline{\mathcal{IP}}}(x_{i-1}) = 0$ (teóricamente) y puede fallar la condición $G_i^{\overline{\mathcal{IP}}}(x_{i-1})G_i^{\overline{\mathcal{IP}}}(x_i) < 0$.

El primer punto se soluciona sin más que incluir la posibilidad de que la intersección de *stencils* sea un punto. Para ello empleamos la siguiente notación: $der(s_i)$ es el índice del extremo derecho

del intervalo *ENO* para la celda I_i. Es decir, si r es el grado del polinomio $\overline{\mathcal{IP}}$, $der(s_i) = s_i + r - 1$ y $izq(s_i) = s_i - 1$. Con esta notación, la condición $\mathcal{S}_{i-1}^{ENO} \cap \mathcal{S}_{i+1}^{ENO} = \emptyset$ será sustituida por $der(s_{i-1}) \leq izq(s_{i+1})$.

Para el segundo punto, cuando cabe la posibilidad de esquina en x_{i-1} utilizamos la condición $G_{i-1}^{\overline{\mathcal{IP}}}(x_{i-2}) \cdot G_{i-1}^{\overline{\mathcal{IP}}}(x_i) < 0$.

Tras dichas modificaciones obtenemos el Algoritmo 3.1. Veamos que cubre todos los casos:

1.- La esquina cae en (x_{i-1}, x_i). Como $der(s_{i-1}) < izq(s_{i+1})$, entra en el primer **si**. Además $G_i^{\overline{\mathcal{IP}}}(x_{i-1}) \cdot G_i^{\overline{\mathcal{IP}}}(x_i) < 0$, $|G_i^{\overline{\mathcal{IP}}}(x_{i-1})| > 0$ y $|G_i^{\overline{\mathcal{IP}}}(x_i)| > 0$ ya que la esquina está en el interior del intervalo de trabajo. Luego se detecta correctamente.

2.- La esquina cae en x_{i-1}. Se cumple que $der(s_{i-1}) \leq izq(s_{i+1})$ con lo que entra en el primer **si**. Dado que $|G_i^{\overline{\mathcal{IP}}}(x_{i-1})| = 0$ no cumple el segundo **si**, aunque sí el tercero. La detección es correcta.

3.- La esquina cae en x_i. En el paso i no se detectará, ya que no cumple las condiciones, pero sí será detectada en el paso $i+1$.

Algoritmo 3.1 *LOCALIZA-ENO-SR-VP*: Localización de esquinas mediante *ENO-SR* y valores puntuales.

Entrada: $\{\bar{f}_i\}_{i=1}^N$, los datos iniciales; $\{s_i\}_{i=1}^N$, asignación ENO
Salida: Localización de las esquinas

para $i = 2, \ldots, N-1$ **hacer**
  si $der(s_{i-1}) \leq izq(s_{i+1})$ **entonces**
    si $G_i^{\overline{\mathcal{IP}}}(x_{i-1}) \cdot G_i^{\overline{\mathcal{IP}}}(x_i) < 0$ & $der(s_{i-1}) < izq(s_{i+1})$ & $\left|G_i^{\overline{\mathcal{IP}}}(x_{i-1})\right| > 0$ & $\left|G_i^{\overline{\mathcal{IP}}}(x_i)\right| > 0$ **entonces**
      Esquina en (x_{i-1}, x_i)
    si no si $G_{i-1}^{\overline{\mathcal{IP}}}(x_{i-2}) \cdot G_{i-1}^{\overline{\mathcal{IP}}}(x_i) < 0$ & $der(s_{i-1}) = izq(s_i)$ & $\left|G_{i-1}^{\overline{\mathcal{IP}}}(x_{i-1})\right| = 0$ & $\left|G_{i-1}^{\overline{\mathcal{IP}}}(x_{i-2})\right| > 0$ & $\left|G_{i-1}^{\overline{\mathcal{IP}}}(x_i)\right| > 0$ **entonces**
      Esquina en el punto x_{i-1}
    fin si
  fin si
fin para

NOTA 3.1. *En la práctica no podemos asegurar las igualdades a 0, ya que trabajamos con precisión finita. Al programarlo debemos sustituir el 0 por* ϵ*. En nuestro caso tomamos* $\epsilon = 10^{-10}$*.*

Teniendo en cuenta la Nota 3.1, advertimos que las discontinuidades muy próximas a x_{i-1} se pueden detectar como discontinuidades en x_{i-1}. Esto no es un problema, más bien una ventaja, ya que es más útil detectar como discontinuidad en un punto del *grid* una esquina muy próxima a dicho punto, que detectarla en el interior de un intervalo sin conocer su localización. La magnitud de dicho ϵ determinará un pequeño entorno de cada punto del *grid* donde se localizarán las esquinas como discontinuidades que caen justo en puntos del *grid*.

Las condiciones sobre $G_{i-1}^{\overline{\mathcal{IP}}}$ no se pueden sustituir por condiciones sobre $G_i^{\overline{\mathcal{IP}}}(x)$. En el siguiente ejemplo vemos porqué. Supongamos que trabajando con *stencils* de longitud 4 nos encontramos con la situación mostrada en la Figura 3.2: $s_{i-1} = i - 4$, $s_i = i$ y $s_{i+1} = i$. Con esto, puede detectarse erróneamente una esquina en el punto del *grid* ya que:

- $G_i^{\overline{\mathcal{IP}}}(x_i)$ y $G_i^{\overline{\mathcal{IP}}}(x_{i-2})$ pueden variar de signo, con lo cual $G_i^{\overline{\mathcal{IP}}}(x_i) \cdot G_i(x_{i-2}) < 0$.
- $der(s_{i-1}) = izq(s_{i+1})$.
- $|G_i^{\overline{\mathcal{IP}}}(x_{i-1})| < eps$.
- $|G_i^{\overline{\mathcal{IP}}}(x_i)| > eps$.
- $|G_i^{\overline{\mathcal{IP}}}(x_{i-2})| > eps$.

Al utilizar $G_{i-1}^{\overline{\mathcal{IP}}}(x)$ en vez de $G_i^{\overline{\mathcal{IP}}}(x)$ se asegura que los polinomios $q_{s_i}^{\overline{\mathcal{IP}}}$ y $q_{s_{i-2}}^{\overline{\mathcal{IP}}}$ se construyan con *stencils* que no crucen la discontinuidad. Si la discontinuidad es de salto, no se cumplirá que $|G_{i-1}^{\overline{\mathcal{IP}}}(x_{i-1})| < eps$ y por tanto no se entrará en el **si** de detección de esquinas en puntos del *grid*. Si la discontinuidad es una esquina en un punto del *grid* puede comprobarse que será detectada correctamente utilizando las condiciones sobre $G_{i-1}^{\overline{\mathcal{IP}}}$.

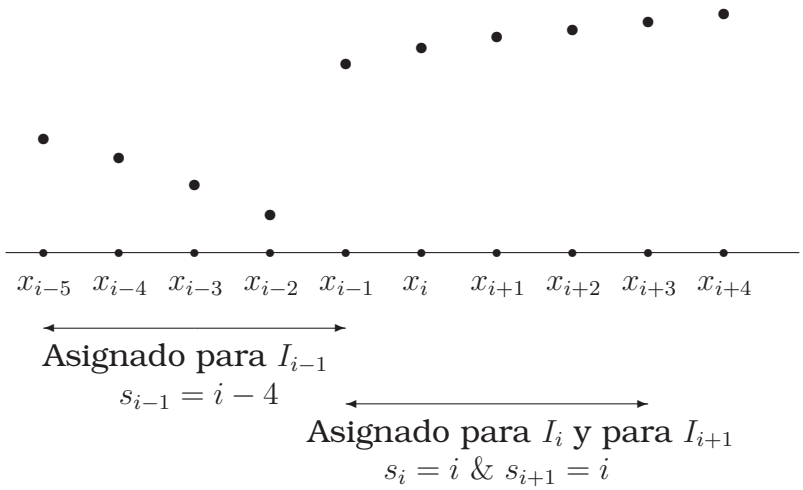

Figura 3.2: *Ejemplo de posible asignación de stencils para una función con un salto que da lugar a una detección de esquina en el punto x_{i-1} si utilizásemos condiciones sobre $G_i^{\overline{\mathcal{IP}}}(x)$ en lugar de hacerlo sobre $G_{i-1}^{\overline{\mathcal{IP}}}(x)$ en el Algoritmo 3.1, LOCALIZA-ENO-SR-VP.*

NOTA 3.2. *Las comprobaciones sobre la magnitud de G son necesarias, pues se pueden dar casos en que se cumplen el resto de condiciones, aunque realmente no estemos ante una esquina.*

Mediante el Algoritmo 3.1 se detectan correctamente discontinuidades esquinas, diferenciando si se trata de una esquina localizada en el interior de un intervalo o en un extremo. Si deseamos incluir la detección de saltos debemos hacer uso del concepto de medias en celda.

Medias en celda

Como vimos en el capítulo de preliminares, disponemos de una herramienta matemática que permite transformar saltos en esquinas. Al suponer que los valores de f (que denotaremos $\bar{f}$) son las medias en celda de una función F, se obtiene que en I_i hay una esquina para F si y sólo si en I_i hay un salto para $\bar{f}$.

Por tanto, la estrategia consiste en detectar esquinas en F me-

diante *ENO-SR* que equivaldrán a saltos en $\bar{f}$. Además recordemos que F nunca se forma explícitamente, sino que trabajamos sobre los valores de $\bar{f}$.

Para detectar las esquinas de F tenemos dos opciones:

1. Trabajar con los valores puntuales de F y el Algoritmo 3.1.

2. Trabajar con $\bar{f}$ y utilizar $G^{\overline{\mathcal{IP}}\prime}$.

Nosotros nos hemos decantado por la primera opción por simplicidad, aunque la segunda es igualmente válida.

Utilizando los algoritmos que aparecen en [16], [15], la detección de esquinas en F es como sigue:

1.- Calcular los *stencils ENO* según el Algoritmo 1.3 o el 1.4.

2.- Detectar las posibles *celdas singulares*: si $\mathcal{S}_{i-1}^{ENO} \cap \mathcal{S}_{i+1}^{ENO} = \emptyset$, etiquetar la celda como *sospechosa* de contener una singularidad.

3.- Para cada celda sospechosa definir la función

$$G_i^{\overline{\mathcal{IC}}}(x) := q_{i+1}^{\overline{\mathcal{IP}}}(x; F^{k-1}, r) - q_{i-1}^{\overline{\mathcal{IP}}}(x; F^{k-1}, r). \tag{3.2}$$

Si $G_i^{\overline{\mathcal{IC}}}(x_{i-1})G_i^{\overline{\mathcal{IC}}}(x_i) < 0 \Rightarrow$ existe una raíz θ_i de $G_i^{\overline{\mathcal{IC}}}(x)$, y etiquetamos la celda como *singular*.

El algoritmo presenta el siguiente problema: una discontinuidad de salto en $[x_{i-1}, x_i]$ se convierte en una esquina de F que cae justo en x_{i-1}. Con lo cual se tiene que $G_i^{\overline{\mathcal{IC}}}(x_{i-1}) \approx 0$ y la condición $G_i^{\overline{\mathcal{IC}}}(x_{i-1})G_i^{\overline{\mathcal{IC}}}(x_i) < 0$ puede no cumplirse. Por ello la sustituiremos por $G_i^{\overline{\mathcal{IC}}}(x_{i-2})G_i^{\overline{\mathcal{IC}}}(x_i) < 0$

Con el fin de localizar saltos hemos introducido una nueva condición al Algoritmo 3.1, obteniendo el Algoritmo 3.2. En caso de no cumplir las condiciones para esquina consideramos el intervalo como sospechoso para contener una esquina para F interpretando en este caso f como las medias en celda de F. Entonces si efectivamente contiene una esquina para F, concluimos que existe un salto para f. De esta manera conseguimos localizar en un mismo algoritmo esquinas interiores a un intervalo, esquinas en

puntos del *grid* y saltos, considerando los datos iniciales como valores puntuales o medias en celda según nos interese.

Algoritmo 3.2 *LOCALIZA-ENO-SR-VP-MC*: Localización de esquinas y saltos mediante *ENO-SR*, valores puntuales y medias en celda.

Entrada: $\{\bar{f}_i\}_{i=1}^N$, los datos iniciales; $\{s_i\}_{i=1}^N$, asignación *ENO*
Salida: Localización de saltos y esquinas

para $i = 2, \ldots, N-1$ **hacer**
  si $der(s_{i-1}) \leq izq(s_{i+1})$ **entonces**
    si $G_i^{\overline{\mathcal{IP}}}(x_{i-1}) \cdot G_i^{\overline{\mathcal{IP}}}(x_i) < 0$ & $der(s_{i-1}) < izq(s_{i+1})$ & $\left|G_i^{\overline{\mathcal{IP}}}(x_{i-1})\right| > 0$ & $\left|G_i^{\overline{\mathcal{IP}}}(x_i)\right| > 0$ **entonces**
      Esquina en (x_{i-1}, x_i)
    si no si $G_{i-1}^{\overline{\mathcal{IP}}}(x_{i-2}) \cdot G_{i-1}^{\overline{\mathcal{IP}}}(x_i) < 0$ & $der(s_{i-1}) = izq(s_i)$ & $\left|G_{i-1}^{\overline{\mathcal{IP}}}(x_{i-1})\right| = 0$ & $\left|G_{i-1}^{\overline{\mathcal{IP}}}(x_{i-2})\right| > 0$ & $\left|G_{i-1}^{\overline{\mathcal{IP}}}(x_i)\right| > 0$ **entonces**
      Esquina en el punto x_{i-1}
    si no si $G_i^{\overline{\mathcal{IC}}}(x_{i-2}) \cdot G_i^{\overline{\mathcal{IC}}}(x_i) < 0$ & $\left|G_i^{\overline{\mathcal{IC}}}(x_{i-2})\right| > 0$ & $\left|G_i^{\overline{\mathcal{IC}}}(x_i)\right| > 0$ & $\left|G_i^{\overline{\mathcal{IC}}}(x_{i-1})\right| = 0$ **entonces**
      Salto en $[x_{i-1}, x_i]$
    si no
      Celda $[x_{i-1}, x_i)$ sospechosa pero no discontinua
    fin si
  fin si
fin para

Experimentos numéricos

Aplicamos los distintos métodos a dos tipos de funciones, ambas con 65 nodos iniciales. La primera de ellas posee una discontinuidad esquina localizada exactamente en un punto del *grid* y otra discontinuidad de salto. La segunda contiene un salto y una esquina localizada en un punto interior a un intervalo, definida en (1.44). Obtenemos la Figura 3.3. En (a) y (b) utilizamos el esquema *LOCALIZA-ENO-SR-VP*, Algoritmo 3.1. La localización de esquinas

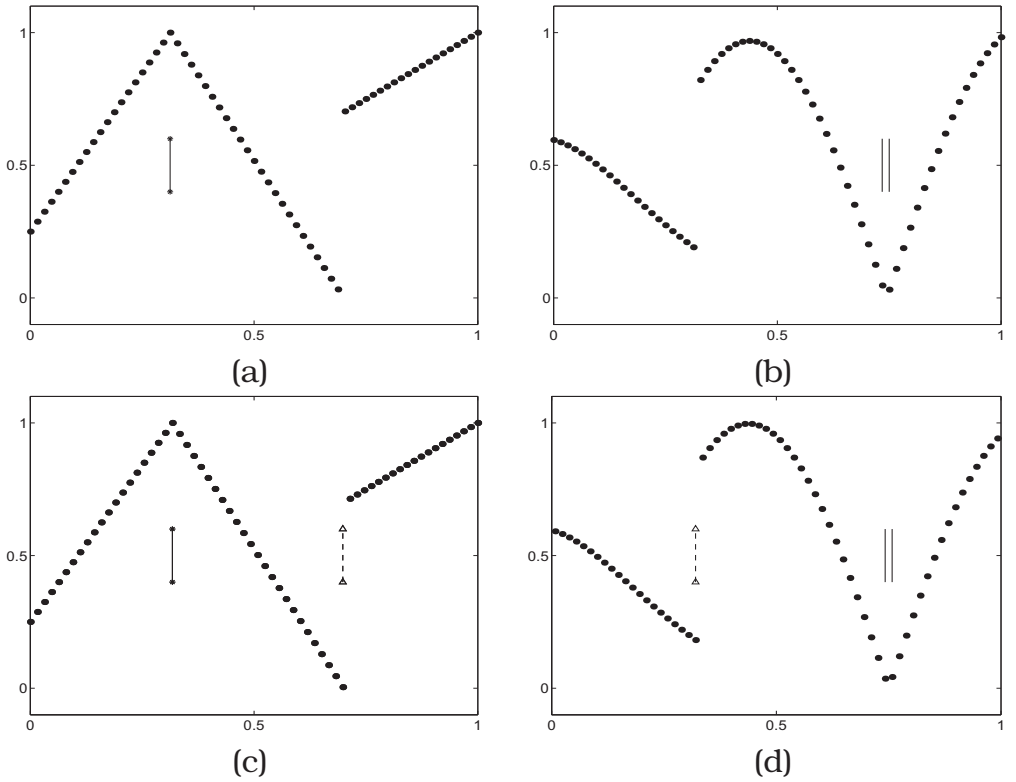

Figura 3.3: *Aplicación del Algoritmo 3.1 (LOCALIZA-ENO-SR-VP) en (a) y (b), y el Algoritmo 3.2 (LOCALIZA-ENO-SR-MC) en (c) y (d), a una función con una esquina en un punto del grid y un salto, y a otra con un salto y una esquina en el interior de un intervalo del grid, ambas con 65 nodos iniciales. Una doble línea continua significa que se ha detectado una esquina en un punto interior a un intervalo); una línea continua con * corresponde a una esquina en un punto del grid) y una línea discontinua con* $\triangle$ *será un salto.*

es correcta, discriminando si se trata de esquinas en puntos del *grid* o interiores a un intervalo. En las Figuras (c) y (d) aplicamos el esquema *LOCALIZA-ENO-SR-VP-MC*, Algoritmo 3.2 lo cual permite además localizar los saltos.

3.2.2 Magnitud de coeficientes *wavelet*

Los coeficientes de altas frecuencias que se obtienen al trabajar con la transformada *wavelet* contienen información acerca de la suavidad de la función. Así, aquellos coeficientes calculados a partir de una región suave son de pequeña magnitud, mientras que los calculados a partir de regiones que contienen discontinuidades son de magnitud elevada. Aprovechando este hecho podemos localizar la posición exacta de las discontinuidades comparando la magnitud de dichos coeficientes. Este método de localización ha sido desarrollado por Chan y Zhou en [26] para ser aplicado en la transformada *ENO-wavelet*.

En lo sucesivo supondremos que la función *wavelet* utilizada posee p momentos nulos y que la discretización aplicada es suficientemente fina como para que cada discontinuidad esté localizada al menos a una distancia de un *stencil* y un punto de otra. O dicho de manera más rigurosa:

Sea una función $f(x)$, llamamos D al conjunto de sus discontinuidades, es decir:

$$D = \{x_i : f(x) \text{ es discontinua en } x_i\} \tag{3.3}$$

DEFINICIÓN 3.1. *Dado un filtro wavelet con stencils de longitud* $l+1$*, decimos que la proyección de* f *en el espacio* V_j*, con paso espacial* $h = 2^{-j}$ *satisface la Propiedad de Separación de Discontinuidades (DSP), si* $(l+2)\cdot h < t$*, con* $t = min\{|x_i - x_j| : x_i, x_j \in D\}$ *y* D *definido en (3.3).*

Luego esto nos dice que en una proyección que satisface la *DSP* no debe haber dos *stencils* consecutivos que contengan dos discontinuidades.

Pasemos a estudiar cómo influyen las discontinuidades en la magnitud de los coeficientes de altas frecuencias para poder diseñar el método de localización. En [20], [69], se demuestra que en las regiones suaves se cumple $|\beta_{j,i}| = \left|f^{(p)}(x)\right| O(\Delta x^p)$, mientras que

en las que hay una discontinuidad en la función o en sus derivadas, $|\beta_{j,i}| = \left|f^{(p-1)}(x)\right| O(\Delta x^{p-1})$. Así, comparando las magnitudes de los $\beta_{j,i}$, podemos encontrar la posición exacta de las discontinuidades.

Veamos cómo lo hacemos. En las regiones suaves, desarrollando por Taylor, se cumple que $|\beta_{j,i}| = (1 + O(\Delta x))\,|\beta_{j,i-1}|$. Luego tomando un cierto $a > 1$, se cumplirá que $|\beta_{j,i}| \leq a\,|\beta_{j,i-1}|$. Si en $\beta_{j,i}$ hay una discontinuidad, pero no en $\beta_{j,i-1}$:

$$|\beta_{j,i}| = \left|f^{(p-1)}(x)\right| O(\Delta x^{p-1}) > a\left|f^{(p)}(x)\right| O(\Delta x^{p}) = |\beta_{j,i-1}|\,.$$

Posteriormente veremos qué a debemos elegir. También podemos razonar comparando cada $\beta_{j,i}$ con su posterior. Si $\beta_{j,i}$ contiene una discontinuidad, pero no $\beta_{j,i+1}$, se cumple: $|\beta_{j,i}| > a\,|\beta_{j,i+1}|$. Finalmente, si tanto $\beta_{j,i}$ como $\beta_{j,i+1}$ no contienen saltos: $|\beta_{j,i}| \leq a\,|\beta_{j,i+1}|$.

Con esta idea, veamos cómo hallamos la localización exacta del salto. Si el filtro tiene longitud $l+1$, con $l = 2k-1$, los $\beta_{j,i}$ están formados a partir de los siguientes *stencils*:

$$\begin{array}{lcl}
\beta_{j,i-1} & = & \{x_{2i-2}, x_{2i-1}, \ldots, x_{2i+l-3}, x_{2i+l-2}\},\\
\beta_{j,i} & = & \{x_{2i}, x_{2i+1}, \ldots, x_{2i+l-1}, x_{2i+l}\},\\
\ldots & = & \ldots\\
\beta_{j,i+k} & = & \{x_{2i+l+1}, x_{2i+l+2}, \ldots, x_{2i+2l}, x_{2i+2l+1}\}.
\end{array}$$

Supongamos que $|\beta_{j,i}| > a\,|\beta_{j,i-1}|$, entonces $\beta_{j,i}$ a sido calculado a partir de un *stencil* que cruza el salto. Como el *stencil* anterior no contiene el salto, sólo hay dos posibles posiciones donde podemos encontrar la discontinuidad: entre $\{x_{2i+l-2}, x_{2i+l-1}\}$ o entre $\{x_{2i+l-1}, x_{2i+l}\}$. Ahora bien, si el salto está entre $\{x_{2i+l-2}, x_{2i+l-1}\}$ habrá $(k-1)$ *stencils* afectados por la discontinuidad, en concreto $\beta_{j,m}$ para $i \leq m \leq (i+k-2)$. Sin embargo si el salto se encuentra entre $\{x_{2i+l-1}, x_{2i+l}\}$ entonces encontraremos k *stencils* afectados, ya que el $\beta_{j,i+k-1}$ también lo estará. Así pues, simplemente debemos comprobar si $\beta_{j,i+k-1}$ está o no afectado por el salto. Para ello debemos comparar $|\beta_{j,i+k-1}|$ y $|\beta_{j,i+k}|$. Con esto, podemos enunciar:

LEMA 3.1. *Sean filtros wavelet de longitud $l+1$, con $l = 2k-1$. Dado un índice i, si $|\beta_{j,i-1}| \leq a\,|\beta_{j,i-2}|$, pero $|\beta_{j,i}| > a\,|\beta_{j,i-1}|$, entonces:*

1. *Si $|\beta_{j,i+k-1}| > a\,|\beta_{j,i+k}|$, hay k stencils consecutivos que contienen la discontinuidad y se halla en (x_{2i+l-1}, x_{2i+l}).*
2. *Si $|\beta_{j,i+k-1}| \leq a\,|\beta_{j,i+k}|$, hay $(k-1)$ stencils consecutivos que contienen la discontinuidad y se halla en (x_{2i+l-2}, x_{2i+l-1}).*

Para finalizar este apartado veremos cómo hemos de tomar el valor de a para localizar correctamente la discontinuidad. Como se ve en [20], [69], si la función es Lipschitz $\gamma \leq p$ en x, es decir $|f(x+\delta) - f(x)| \leq \delta^\gamma$ para cualquier δ pequeño, el correspondiente coeficiente *wavelet* de alta frecuencia es de orden $O(\Delta x^\gamma)$. Como habíamos señalado antes, $|\beta_{j,i}| = \left|f^{(p)}(x)\right| O(\Delta x^p)$. Con lo cual, tomando desarrollos de Taylor, se cumple que en las regiones suaves $|\beta_{j,i}| = (1 + O(\Delta x))\,|\beta_{j,i-1}|$.

Sin embargo, si el $\beta_{j,i}$ ha sido calculado a partir de un *stencil* que cruza una discontinuidad, su orden es al menos uno menor que en las regiones suaves. En concreto, suponiendo que $f(x)$ tiene una discontinuidad en la derivada de orden m en un punto $x_0 \in (i\Delta x, (i+l)\Delta x)$ para algún entero i, utilizando desarrollos de Taylor, obtenemos en un entorno de x_0:

$$f(x) = g(x) + \begin{cases} f^{(m)}(x_o-)(x-x_0)^m + O(x-x_0)^{(m+1)} & x \leq x_0, \\ f^{(m)}(x_o+)(x-x_0)^m + O(x-x_o)^{(m+1)} & x > x_0, \end{cases}$$

con

$$g(x) = f(x_0) + f'(x_0)(x-x_0) + \frac{f''(x_0)}{2!}(x-x_0)^2 + \cdots + \frac{f^{(m-1)}(x_0)}{(m-1)!}(x-x_0)^{(m-1)}.$$

Utilizando que la función *wavelet* posee p momentos nulos:

$$\begin{aligned} |\beta_{j,i}| &= \left| \int f(x)\psi_{j,i}(x)dx \right| \\ &= \left| \int_{i\Delta x}^{x_0} (f^{(m)}(x_0-)(x-x_0)^m + O(x-x_0)^{(m+1)})\psi_{j,i}(x)dx + \right. \end{aligned}$$

$$\int_{x_0}^{(i+l)\Delta x} (f^{(m)}(x_0+)(x-x_0)^m + O(x-x_0)^{(m+1)})\psi_{j,i}(x)dx\Bigg| \\ = \left|[f^{(m)}(x_0)]\right| O(\Delta x^m). \tag{3.4}$$

Con lo cual la elección de a debe ser la siguiente:

$$(1+O(\Delta x)) \leq a \leq \min_x \left\{ \left|[f^{(m)}(x)]\right| O(\Delta x^{(m-p)}) \right\}. \tag{3.5}$$

Veamos que realmente cumple lo que queremos. Supongamos que $\beta_{j,i-1}$ no está afectado por una discontinuidad, entonces:

- Si $\beta_{j,i}$ no está afectado, $|\beta_{j,i}| = (1+O(\Delta x))\,|\beta_{j,i-1}|$ con lo cual, como $(1+O(\Delta x)) \leq a$ se cumple que $|\beta_{j,i}| \leq a\,|\beta_{j,i-1}|$.

- Si $\beta_{j,i}$ está afectado, como $a \leq \min_x \left\{ \left|[f^{(m)}(x)]\right| O(\Delta x^{(m-p)}) \right\}$, tenemos que:

$$\begin{aligned} |\beta_{j,i}| &= \left|[f^{(m)}(x_0)]\right| O(\Delta x^{(m)}) \\ &> \min_x \left\{ \left|[f^{(m)}]\right| O(\Delta x^{(m-p)}) \right\} \left|f^{(p)}(x)\right| O(\Delta x^p) \geq a\,|\beta_{j,i-1}|. \end{aligned}$$

De la misma manera podemos comprobar que se cumplen los resultados cuando $\beta_{j,i}$ está afectado por la discontinuidad pero $\beta_{j,i+1}$ no lo está.

NOTA 3.3. *Para prevenir la inestabilidad numérica causada por β's pequeños, en el algoritmo introduciremos la condición $|\beta_{j,i}| \geq \epsilon$, con ϵ un valor predefinido y mayor que cero.*

NOTA 3.4. *Una discontinuidad de la derivada m-ésima de intensidad menor que $O(\Delta x^{(p-m)})$ no será detectada por el método descrito anteriormente. Sin embargo, el error causado al no detectarse es relativamente pequeño.*

El Lema 3.1 puede fallar en el siguiente supuesto: $\beta_{j,i}$ ha sido calculado a partir de un *stencil* que cruza un salto, pero $\beta_{j,i-2} = 0$. En este caso no se cumple que $|\beta_{j,i-1}| \leq a\,|\beta_{j,i-2}|$ y la detección falla. El caso descrito, aunque poco frecuente, puede darse si estamos trabajando con rectas. Para evitarlo proponemos modificar el Lema 3.1 de la siguiente manera:

LEMA 3.2. *Sean filtros wavelet de longitud $l = 2k-1$. Dado un índice i, si $(|\beta_{j,i-1}| \leq a\,|\beta_{j,i-2}| \vee (|\beta_{j,i-2}| = 0 \wedge \left|\beta_{j,i-1)}\right| < \epsilon))$, pero $|\beta_{j,i}| > a\,|\beta_{j,i-1}|$, entonces (tomando ϵ como en la Nota 3.3):*

1. *Si $|\beta_{j,i+k-1}| > a\,|\beta_{j,i+k}|$, hay k stencils consecutivos que contienen el salto, con lo cual la discontinuidad está localizada entre $\{x_{2i+l-1}, x_{2i+l}\}$.*
2. *Si $(|\beta_{j,i+k-1}| \leq a\,|\beta_{j,i+k}| \vee (|\beta_{j,i+k}| = 0 \wedge |\beta_{j,i+k-1}| < \epsilon))$, hay $(k-1)$ stencils consecutivos que contienen el salto, con lo cual la discontinuidad está localizada entre $\{x_{2i+l-2}, x_{2i+l-1}\}$.*

Ejemplos

Nuestro objetivo es localizar saltos y esquinas. Por tanto podemos utilizar una base *wavelet* con $p = 2$ momentos nulos, como por ejemplo *DB4*. En (3.5) hemos establecido unas cotas para la constante a. La cota inferior se conoce directamente al disponer del espaciado de la discretización. Sin embargo, la cota superior no es calculable, ya que $\left|f^{(m)}(x)\right|$ es desconocido. En la práctica el valor de a abarca un intervalo amplio de valores y determinará la magnitud de las discontinuidades que deseemos considerar. Una elección aceptable de a es 2 y será la que utilicemos en nuestros experimentos.

Aplicamos el método del Lema 3.2 a dos funciones que poseen saltos y esquinas. La Figura 3.4 muestra el resultado. La localización de saltos y esquinas es correcta, aunque en este caso no se discrimina entre ellas.

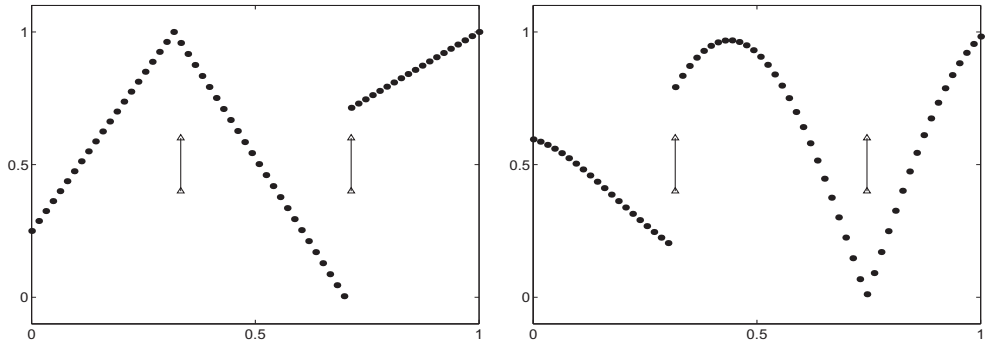

Figura 3.4: *Discontinuidades detectadas según el Lema 3.2. Parámetros de detección:* $a = 2$ *y* $\epsilon = 0{,}001$*. La localización de la discontinuidad se representa mediante una lnea vertical con triángulos en los extremos.*

3.3 Métodos de detección de discontinuidades para funciones con ruido

En esta sección trabajaremos con funciones del siguiente tipo:

$$\tilde{f}(x) = f(x) + n(x),$$

donde $f(x)$ es una función discontinua compuesta por trozos suaves y $n(x)$ representa el ruido *blanco gaussiano*. El ruido *blanco* es una señal aleatoria (proceso estocástico) que se caracteriza porque sus valores de señal en dos instantes de tiempo diferentes no guardan correlación estadística. Por *gaussiano* entendemos que sigue una distribución de Gauss[1].

Para expresar la magnitud del ruido utilizamos la medida conocida como SNR (*Signal-to-Noise-Ratio*), [38]:

$$SNR = \frac{\text{potencia media de la seal}}{\text{potencia media del ruido}}. \tag{3.6}$$

[1]En capítulos posteriores se justificará esta elección del tipo de ruido introducido.

Dado que SNR es una razón de dos potencias, se define a menudo en decibelios:

$$SNR(f,g) := 10\log_{10}\frac{\sum_{i=1}^{N} f_i^2}{\sum_{i=1}^{N}(f_i - g_i)^2}, \tag{3.7}$$

Así cuando decimos que hemos introducido ruido de magnitud $SNR = c$ entendemos que se ha añadido *ruido blanco gaussiano*[2] y la relación señal-ruido entre la función de partida y la que contiene ruido es c.

SNR es la razón entre lo que se quiere (señal) y lo que no se quiere (ruido). Una SNR alta indica que la señal está menos corrompida por ruido, mientras que una SNR baja indica que la señal está muy corrompida por el ruido.

3.3.1 Combinación *ENO* y mínimos cuadrados

Las diferencias divididas retienen la información necesaria para huir de las discontinuidades en funciones suaves. Sin embargo al trabajar con funciones con ruido dicha información no sirve y debemos aplicar métodos específicos para funciones con ruido. En los trabajos de Daphna Mizrachi [60] se propone aplicar la idea *ENO*, pero sustituyendo las diferencias divididas por mínimos cuadrados. Basándonos en estas ideas, las adaptaremos a nuestra nomenclatura y estudiaremos su aplicabilidad.

Evitando *stencils* que crucen discontinuidades

Sea $q_i^{\overline{\mathcal{AP}}_{nl,nr}}(x; f^{k-1}, r)$ el polinomio de grado r que calcula la reconstrucción por *aproximaciones* en x a partir de $\left[x_{i-nl}^{k-1}, \ldots, x_{i+nr-1}^{k-1}\right]$. Llamando $n := nl+nr$, los n posibles *stencils* que contienen al punto x_i son $\{x_{i-(n-k)}^{k-1}, \ldots, x_{i+k-1}^{k-1}\}$, $k = 1, \ldots, n$.

Para decidir cuál es el *stencil* que no cruza la discontinuidad se utilizan dos medidas. Veamos cada una de ellas.

[2] En MATLAB® podemos introducir *ruido blanco gaussiano* con las instrucciones *randn*, *mvnrnd*, *wgn* y *awgn*.

Calculamos los siguientes términos para cada i:

$$E_1(x_i,k) \equiv \left| \frac{d^r}{dx^r} q_i^{\overline{\mathcal{AP}}_{n-k,k}}(x_i; f^{k-1}, r) \right|. \tag{3.8}$$

Dado que el polinomio $q_i^{\overline{\mathcal{AP}}_{n-k,k}}(x_i; f^{k-1}, r) = \sum_{i=1}^{k+1} a_i x^{i-1}$ es de grado k dicha medida será $E_1(x_i,k) \equiv |r! a_{r+1}|$. Calculamos k_1^* como sigue:

$$E_1(x_i,k_1^*) = min\left\{E_1(x_i,1), E_1(x_i,2), \ldots, E_1(x_i,n)\right\}.$$

Definimos $i_1(i) := i - (n - k_1^*)$. Si para trabajar con x_i utilizamos el *stencil* $\left\{x_{i_1(i)}, x_{i_1(i)+2}, \ldots, x_{i_1(i)+n-1}\right\}$, nos evitamos cruzar discontinuidades.

Otra manera de seleccionar el *stencil* es fijándonos en los errores entre la aproximación y la función con ruido:

$$E_2(x_i,k_2) = \sum_{j=1}^{n} \left(q_i^{\overline{\mathcal{AP}}_{n-k,k}}(x_{i-(n-k)+j}; f^{k-1}, r) - \tilde{f}(x_{i-(n-k)+j}) \right)^2. \tag{3.9}$$

Como antes, calculamos k_2^*:

$$E_2(x_i,k_2^*) = min\left\{E_2(x_i,1), E_2(x_i,2), \ldots, E_2(x_i,n)\right\}. \tag{3.10}$$

Se define

$$i_2(i) := i - (n - k_2^*), \tag{3.11}$$

y el *stencil* asignado a x_i será $\left\{x_{i_2(i)}, x_{i_2(i)+1}, \ldots, x_{i_2(i)+n-1}\right\}$.

Detección de discontinuidades

Definimos x_d como un punto de discontinuidad si, dado n (tamaño del *stencil*), se cumple:

$$i_1(d) = d \quad \wedge \quad i_1(d-1) = d - n, \tag{3.12}$$

$$i_2(d) = d \quad \wedge \quad i_2(d-1) = d - n. \tag{3.13}$$

La Figura 3.5 muestra todas las posibles elecciones de *stencils* que contienen a x_{i-1}^{k-1} y a x_i^{k-1}, donde es fácil ver que si la discontinuidad se halla en $d = i$ se cumplen (3.12) y (3.13).

El porqué utilizamos simultáneamente E_1 y E_2 proviene de la observación de casos prácticos. Si nos basamos únicamente en E_1 o en E_2 se localizan numerosas falsas discontinuidades. Al usar ambos criterios se reducen dichos errores.

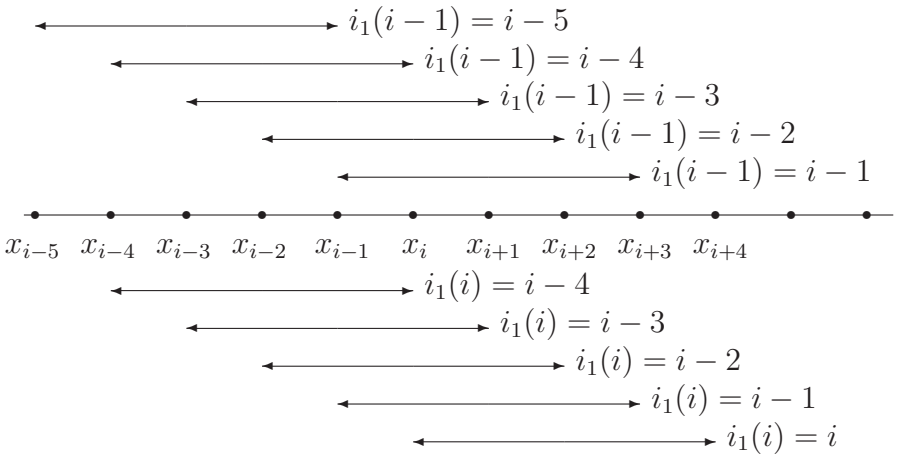

Figura 3.5: *Posibles stencils de 5 nodos que incluyen el punto x_i y los que incluyen el punto x_{i-1}.*

Ejemplos

En la Figura 3.6 aplicamos los métodos de detección a la función (1.44) con 256 nodos a la que hemos añadido ruido blanco gaussiano con un determinado SNR. En cada gráfico se han representado las discontinuidades detectadas mediante el criterio dado por E_1, por E_2 y por ambos simultáneamente. Cada línea vertical que aparece en el punto x_i significa que el intervalo $[x_{i-1}, x_i]$ contiene una discontinuidad. En ambas hemos utilizado $r = 1$ como grado del polinomio por aproximación y $n = 10$ como longitud de los *stencils*. Se observa claramente la necesidad de utilizar simultáneamente ambos criterios, ya que por separado se localizan un elevado número de falsas discontinuidades. En la figura izquierda la localización del salto es correcta, mientras que en la derecha a parte de ésta se localizan dos falsas discontinuidades. Otra conclusión de este ejemplo es que la discontinuidad esquina queda sin ser localizada, ya que el ruido enmascara la naturaleza de la singularidad.

La elección del grado del polinomio y del tamaño del *stencil* va a ser fundamental para que el método detecte las discontinuidades de forma satisfactoria. Para estudiar esta cuestión hemos creado

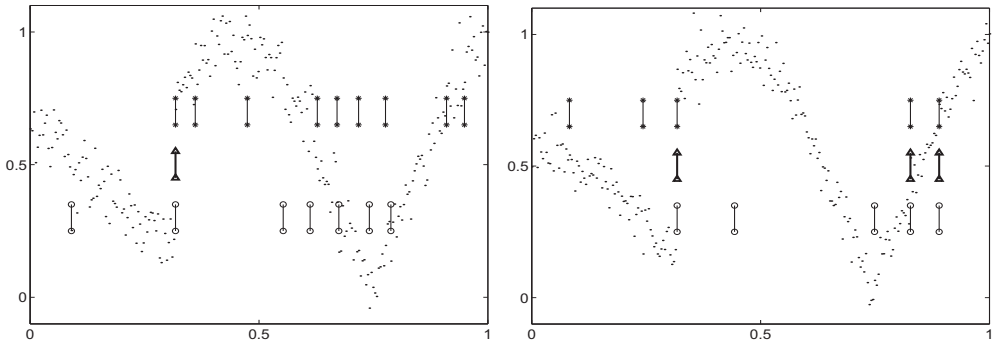

Figura 3.6: *Detección de discontinuidades con $E1$ y $E2$. Tomamos 256 nodos de la señal (1.44) con $SNR = 23$ (izquierda) y $SNR = 25$ (derecha). Para los polinomios utilizamos tamaño del stencil $n = 10$ y grado $r = 1$. Una línea vertical en x_i indica que el intervalo $[x_{i-1}, x_i]$ contiene una discontinuidad según el criterio: '*' con $E1$ (3.12); 'o' con $E2$ (3.13); '$\triangle$' simultáneamente con ambas medidas.*

la Tabla 3.1 donde aplicamos la detección con diferentes parámetros a la función (1.44) a la que añadimos diferentes niveles de ruido. Aunque los resultados corresponden a ejemplos concretos podemos extrapolar algunas conclusiones. Es preferible el uso de grado lineal, ya que ofrece mejores resultados que el resto. En cuanto al tamaño del *stencil* una buena elección es $n = 10$.

3.3.2 Magnitud de coeficientes *wavelet* para funciones con ruido

En este apartado aplicaremos el método descrito en la sección 3.2.2 a una función con ruido. Para una función sin ruido se localizan correctamente tanto discontinuidades de salto como esquinas. Sin embargo al añadir ruido, la detección falla, como puede observarse en la Figura 3.7, izquierda. Es sencillo comprobar que no existe una elección de a y ϵ que dé buenos resultados, con lo cual debemos estudiar otras alternativas.

En la Figura 3.7, derecha, representamos los coeficientes de altas frecuencias de la función (1.44) con 256 nodos a la que he-

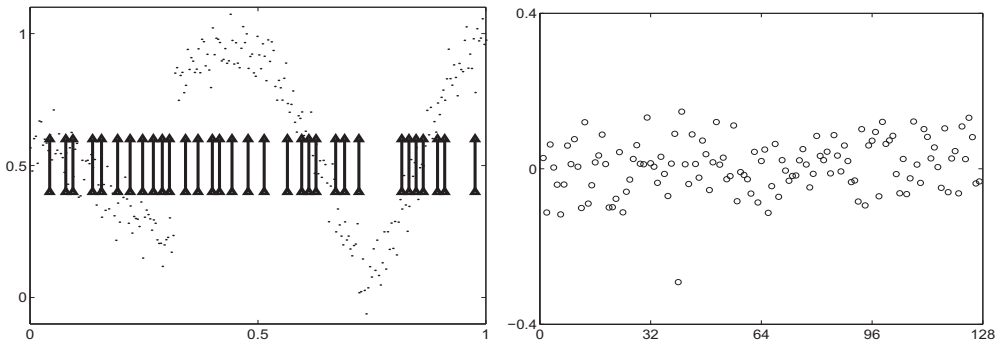

Figura 3.7: *Izquierda: detección de discontinuidades (líneas verticales) siguiendo el Lema 3.1 con $a = 2$, $\epsilon = 0{,}001$ en la función (1.44) con ruido de magnitud $SNR = 23$. Derecha: coeficientes de altas frecuencias asociados.*

mos añadido ruido con $SNR = 23$. Observamos que en el salto se produce un aumento detectable de la magnitud de los coeficientes $\beta_{j,i}$, aunque no para esquinas. Aunque el Lema 3.2 no sea viable en el caso de funciones con ruido, el anterior ejemplo pone de

		Ruido introducido							
		$SNR = 21$		$SNR = 23$		$SNR = 25$		$SNR = 27$	
k	n	D	FD	D	FD	D	FD	D	FD
1	5	C	1	C	3	C	4	C	1
1	10	C	0	C	0	C	0	C	0
1	15	C	0	C	0	C	1	C	1
2	5	C	1	C	1	I	3	C	2
2	10	I	0	C	0	C	0	I	0
2	15	I	0	I	0	I	0	I	0
3	5	I	2	I	2	C	2	I	0
3	10	I	1	I	0	I	0	I	0
3	15	I	0	I	0	I	0	I	0

Tabla 3.1: *Detección de la discontinuidad de salto de la función (1.44) con 256 nodos y ruido, mediante las medidas E_1 (3.12) y E_2 (3.13), variando el grado del polinomio por aproximación, r, y el tamaño del stencil, n. Las columnas rotuladas con D admiten dos indicadores: C (se ha detectado correctamente el salto), I (no se ha detectado el salto). Las columnas FD indican el número de falsas discontinuidades.*

manifiesto que podemos utilizar la información de los coeficientes de altas frecuencias para la detección de discontinuidades tipo salto.

La detección de β's de magnitud elevada constituye la base en *Wavelet Shrinkage Denoising* (p. ej. [63], [70], [35],) donde se utilizan cortes para los coeficientes de altas frecuencias. Para dichos cortes se define un umbral, λ, que decide si los coeficientes β constituyen información significativa de la señal. Así, los coeficientes mayores que λ se mantienen mientras que el resto se eliminan (*hard threshold*) o se reducen (*soft threshold*). Esto se describirá con mayor detalle en capítulos posteriores.

No existe una manera general y óptima para calcular λ en todos los casos, pero sí buenas aproximaciones como *VisuShrink* y *SureShrink*. *VisuShrink* cubre nuestras necesidades, ya que *SURE* mejora a *VISU* pero únicamente cuando trabajamos con más de un nivel de multirresolución.

VisuShrink, propuesto por Donoho y Johnstone [34] en 1994, ofrece el umbral óptimo en sentido asintótico cuando trabajamos con ruido blanco gaussiano. Minimiza la función $E\left\|Y_{Thres} - Y_{Orig}\right\|^2$, donde Y_{Thres} y Y_{Orig} son la aproximación y la función original respectivamente. Este valor de λ se conoce como *Universal Threshold* y viene dado por la expresión $\lambda^U = \sigma\sqrt{2log(N)}$, siendo N el tamaño de la señal original y σ^2 la varianza del ruido (suponemos que el ruido que hemos añadido sigue una $N(0, \sigma^2)$).

El dato que desconocemos es σ y es necesario estimarlo. A partir de la fórmula de una $N(0, \sigma^2)$ se puede deducir una aproximación razonablemente precisa mediante $\sigma \approx \frac{MAD}{0{,}6475}$, donde MAD es la mediana de las magnitudes de todos los coeficientes de altas frecuencias en la escala más fina.

A pesar de la optimalidad asintótica del método, en la práctica se pueden hallar mejores candidatos para λ. En la literatura referente a este método se utilizan otros cortes como: $\lambda = \sigma\sqrt{2log(N \cdot log(N))}$ o $\lambda = \sigma\frac{\sqrt{2log(N)}}{\sqrt{N}}$.

El umbral *VISU* nos proporciona aquellos coeficientes de altas frecuencias que han sido calculados a partir de *stencils* que cruzan discontinuidades. A partir de esta información, la localización del salto es trivial. Por ejemplo para *DB4*, observando el esquema representado en la Figura 1.13, se deduce que:

- Si $|\beta_{1,i}| \geq \lambda$ y $|\beta_{1,i+1}| \geq \lambda$, la discontinuidad se halla entre f_{2i+2} y f_{2i+3}.
- Si $|\beta_{1,i-1}| \leq \lambda$ y $|\beta_{1,i}| \geq \lambda$ y $|\beta_{1,i+1}| \leq \lambda$, la discontinuidad se halla entre f_{2i+1} y f_{2i+2}.

Ejemplos

La Figura 3.8 muestra la detección obtenida mediante el corte *VISU* y base *DB4* de los coeficientes de altas frecuencias. Hemos tomado la función (1.44) a la que hemos añadido ruido con $SNR = 23$. En la figura izquierda se muestra la localización de las discontinuidades. En este caso el salto queda correctamente localizado pero no la esquina. En la figura derecha hemos representado los coeficientes de altas frecuencias y el corte *VISU*. Observamos que hay un único $\beta_{j,i}$ que sobrepasa el umbral *VISU*. Dicho coeficiente proviene de un *stencil* que cruza el salto, de ahí su magnitud elevada y en consecuencia la correcta detección. Sin embargo la esquina no proporciona variaciones significativas en la magnitud de los coeficientes y por tanto es indetectable.

En base a esta función, en los experimentos numéricos hemos comprobado que el método detecta correctamente los saltos cuando el ruido añadido es de magnitud $SNR \geq 25$. Cuando $SNR = 23$ la detección es buena en la mayoría de casos. Pero si $SNR \leq 21$ la detección falla.

3.3.3 Reconstrucción por *aproximaciones* de *stencil* creciente

En esta sección presentamos un nuevo detector de discontinuidades para funciones con ruido. La idea se basa en comparar los errores que se obtienen a medida que se van añadiendo nodos en el *stencil* para la reconstrucción por *aproximaciones* mientras mantenemos un grado fijo. Cuando al introducir un nuevo nodo, el error entre la función original y la reconstrucción sea elevado, significará que hemos cruzado un salto.

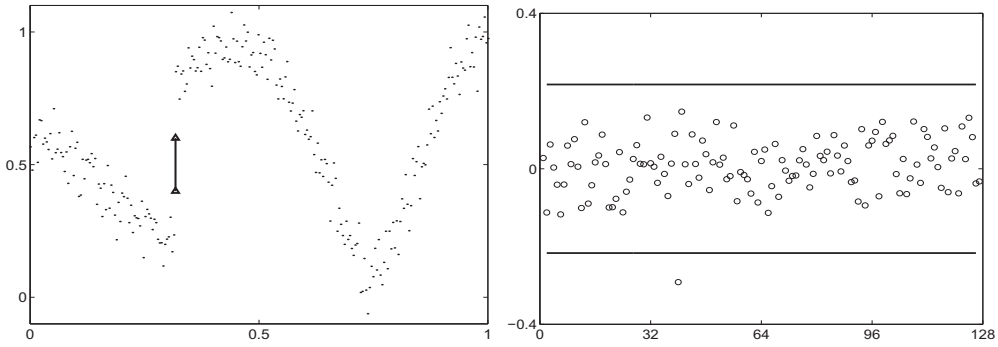

Figura 3.8: *Izquierda: detección de discontinuidades (línea vertical) según la magnitud de los coeficientes wavelet que sobrepasan el umbral VISU. Derecha: representamos los coeficientes de altas frecuencias asociados, con 'o', y el umbral VISU, con una línea horizontal. La función de partida es (1.44) con 256 nodos a la que se le ha añadido ruido con* $SNR = 23$.

Partiendo de los datos iniciales $\{(x_i, f_i)\}_{i=1}^{N}$, el esquema es el siguiente:

1. Tomamos un valor inicial, $i = 10$.
2. Tomamos los primeros i puntos y construimos el polinomio por aproximaciones con un grado fijo, por ejemplo $r = 3$.
3. Calculamos el error

$$e(i) = \sum_{j=1}^{i} \left(q_i^{\overline{AP}_{i-1,1}}(x_j; f, r) - f_j \right)^2 . \tag{3.14}$$

4. Añadimos un punto ($i = i + 1$) y volvemos al paso 2.

Cuando se detecte un aumento relevante del error cometido, ahí habrá una discontinuidad y volveremos a empezar de nuevo a la derecha de la discontinuidad detectada, iterando el proceso.

Veamos algún ejemplo sencillo. En la Figura 3.9 representamos los errores cometidos utilizando una función con un salto. Observamos que el salto produce un aumento detectable en el error. Evidentemente las esquinas no serán detectables, aunque es razonable al trabajar con funciones con ruido.

El siguiente paso es localizar salto. Para ello se propone lo siguiente:

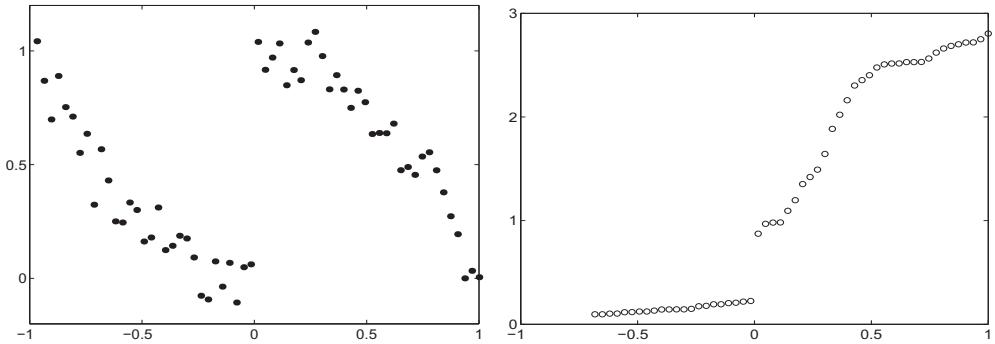

Figura 3.9: *Función con un salto y vector de errores e_i según (3.14).*

1. El vector d contiene la posición de las discontinuidades, de forma que si $d(k) = i$ entre $[x_{i-1}, x_i]$ hay un salto ($d(1) = 1$).

2. Calculamos las diferencias de los errores, $errdif(k) = |e(k) - e(k-1)|$ para $k = d(j-1), \ldots, i-1$ (en el primer paso $j = 2$).

3. Si $errdif(i) > factor \cdot max\{errdif(d(j-1)), \ldots, i-1\} \Rightarrow d(j) = i$, es decir hay una discontinuidad entre $[x_{i-1}, x_i] \Rightarrow j = j+1$, volver al paso 2.

NOTA 3.5. *El valor de $factor$ determinará la magnitud de los saltos que consideremos. Nosotros utilizamos $factor = 3$ y puede interpretarse que una celda será discontinua si el error que se produce al incluir su extremo derecho es el triple que la media de los errores acumulados entre la función original y la reconstrucción por aproximaciones para valores puntuales de grado fijo.*

Con este esquema y tomando unos 10 puntos como mínimo entre discontinuidad y discontinuidad, obtenemos los resultados de la Figura 3.10. Utilizamos a la derecha la función (1.44) y a la izquierda la función

$$f(x) = \begin{cases} -xsen\left(\frac{3}{2}\pi x^2\right) & \text{si} \quad -1 \leq x < -\frac{1}{3}, \\ |sen(2\pi x)| & \text{si} \quad -\frac{1}{3} \leq x < \frac{1}{3}, \\ 2x - 1 - \frac{1}{6}sen(3\pi x) & \text{si} \quad \frac{1}{3} \leq x < 1, \end{cases} \tag{3.15}$$

que contiene dos saltos y una esquina. La detección es correcta excepto en el caso de discontinuidades en la primera derivada. Del estudio de los resultados se deduce que el método reporta escasas falsas discontinuidades y funciona correctamente incluso con cantidades elevadas de ruido.

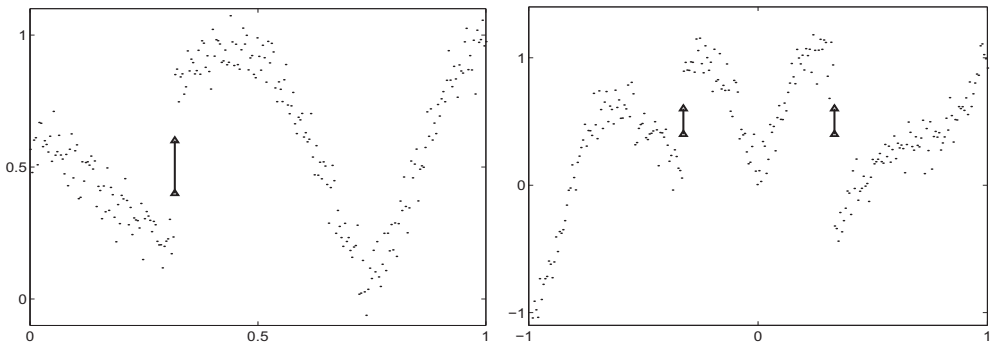

Figura 3.10: *Discontinuidades detectadas (lineas verticales) mediante el método reconstrucción por aproximaciones de stencil creciente. A la izquierda utilizamos la función (1.44) y a la derecha la función (3.15). Se utiliza grado* $r = 3$ *y* $SNR = 23$.

3.3.4 Comparativa entre los métodos de detección para funciones con ruido.

Para conocer qué localizadores proporcionan mejores resultados, hemos realizado[3] la Tabla 3.2. En ella aplicamos a la función (1.44) los diversos localizadores descritos en esta sección. Para cada método, marcamos con una C si se ha localizado correctamente el salto e incluimos el número de falsas discontinuidades. La cantidad de ruido introducido varia desde $SNR = 21$ hasta $SNR = 27$.

Según lo estudiado en este capítulo, podemos deducir:

E_1 y E_2. Es preferible utilizar grado 1 para el polinomio de la reconstrucción por *aproximaciones*. Es un método que localiza

[3]En http://www.uv.es/anims/noguera/ podemos descargar una GUI para realizar experimentos de detección tanto para funciones con ruido como sin él.

	Ruido introducido							
	$SNR = 21$		$SNR = 23$		$SNR = 25$		$SNR = 27$	
MÉTODO	D	FD	D	FD	D	FD	D	FD
E_1 & E_2	C	0	C	0	C	1	C	2
$MW - VISU$	I	0	C	0	C	0	C	0
$\overline{\mathcal{AP}} - SC$	C	0	C	0	C	0	C	0

Tabla 3.2: *Detección de la discontinuidad de salto de la función (1.44) con 256 nodos y ruido de diferentes magnitudes. Se aplican varios métodos: E_1 & E_2 (con grado 1 para el polinomio y tamaño del stencil 10), $MW - VISU$ magnitud de coeficientes wavelet que sobrepasan el umbral VISU, con la base DB4) y $\overline{\mathcal{AP}} - SC$ (reconstrucción por aproximaciones por valores puntuales de stencil creciente, con grado del polinomio 3). Las columnas rotuladas con D admiten dos indicadores: C (se ha detectado correctamente el salto), I (no se ha detectado el salto). Las columnas FD indican el número de falsas discontinuidades detectadas.*

correctamente las discontinuidades pero produce mayor número de falsas discontinuidades que el resto.

Corte *VISU* para coeficientes *wavelet*. Es un método fiable que produce escasas falsas discontinuidades, aunque con ruido elevado no localiza algunas discontinuidades verdaderas.

Reconstrucción por *aproximaciones* de *stencil* creciente. Ofrece mejores resultados que sus rivales incluso para valores de ruido elevados, aunque su coste computacional es mayor.

3.4 Conclusiones

En este capítulo hemos centrado nuestra atención en la localización de discontinuidades en funciones compuestas por trozos suaves. En primer lugar hemos revisado los algoritmos de localización para funciones sin ruido basados en la técnica *ENO-SR* y en la magnitud de coeficientes *wavelet*. Para los métodos basa-

dos en *ENO-SR*, [16], pueden darse casos en los que la detección sea fallida, tanto para valores puntuales como para medias en celda. Hemos introducido nuevas condiciones para que la detección sea correcta en cualquier situación en las que aparezcan saltos y esquinas, siempre y cuando estén lo suficientemente alejadas entre sí. El algoritmo definitivo basado en *ENO-SR* se ha construido combinando las ideas de valores puntuales y medias en celda. Se ha revisado la detección de discontinuidades basada en la magnitud de coeficientes *wavelet*, [26]. Hemos incluido condiciones para aumentar las situaciones de detección y hemos comprobado que los métodos funcionan correctamente.

En segundo lugar hemos estudiado la detección de discontinuidades para funciones con ruido. Se ha comprobado que los métodos para funciones sin ruido no sirven en este caso. Hemos estudiado los trabajos de Daphna Mizrachi [60], centrándonos en la detección de discontinuidades para funciones con ruido. Se ha comprobado que el método no produce resultados óptimos. A continuación se ha estudiado la detección de discontinuidades basadas en el umbral *VISU*, [34], de *Wavelet Shrinkage Denoising* según los coeficientes de altas frecuencias. Finalmente hemos diseñado un nuevo método de detección para funciones con ruido basado en la comparación de los errores que se obtienen entre la función original y su reconstrucción por aproximaciones utilizando un grado fijo pero aumentando la longitud del *stencil*. Al comparar experimentalmente los métodos se ha obtenido que este último es el que proporciona mejores resultados, ya que localiza eficientemente discontinuidades verdaderas y es aplicable a funciones con una elevada magnitud de ruido introducido.

En conclusión disponemos de métodos fiables para localizar discontinuidades tipo salto y esquina de funciones discontinuas compuestas por trozos suaves. Sin embargo, al introducir ruido únicamente podemos aspirar a localizar saltos, debiendo para ello utilizar métodos específicos para funciones con ruido.