UNIVERSITAT DE BARCELONA
Facultat de Matemàtiques i Informàtica

GRAU DE MATEMÀTIQUES

Treball final de grau

LA SIGNATURA DE UN CAMINO Y UNA APLICACIÓN PRÁCTICA

Autor: Rubén Jiménez Lumbreras

Director: Dr. José Manuel Corcuera Valverde
Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 8 de junio de 2024

Abstract

In this undergraduate thesis we will define the concept of the signature for piecewise class C^1 paths and we will study and show some of its most important properties, which indicate that there is a close relationship between the path and its signature, allowing us to use the latter to describe it. Furthermore, motivated by the aforementioned properties, we will analyse the role of the signature in the field of machine learning, one of its most recent and active applications, and we will study how it can be used to build attributes that are able to show and reveal the geometric properties of the considered data sequences. In addition, we will put this last part into practice by applying the signature together with machine learning techniques to the well-known problem of handwritten digit classification. Finally, we will propose two novel generalisations of the signature, motivated by fractional calculus, of which we will also study some of their properties and applications, comparing their results with those of the original signature in the handwritten digit recognition problem, where significant improvements in accuracy rates are observed.

Resumen

En este trabajo de final de grado se definirá el concepto de la signatura para caminos de clase C^1 a trozos y se estudiarán y demostrarán algunas de sus propiedades más importantes, que nos indican que existe una estrecha relación entre el camino y su signatura, cosa que nos permite utilizar esta última para describirlo. También, motivados por las propiedades comentadas previamente, se analizará el papel de la signatura en el campo del *machine learning*, una de sus aplicaciones más recientes y activas, estudiando como se puede utilizar para construir atributos que sean capaces de mostrar y exponer las propiedades geométricas de las secuencias de datos consideradas. Además, pondremos en práctica esta última parte, aplicando la signatura junto con técnicas de *machine learning* al problema clásico de clasificación de dígitos manuscritos. Finalmente, propondremos dos nuevas generalizaciones de la signatura, motivadas por el cálculo fraccionario, de las que estudiaremos también algunas de sus propiedades y aplicaciones, comparando sus resultados con los de la signatura original en el problema de reconocimiento de dígitos manuscritos, donde se observan mejoras significativas en las tasas de acierto.

2020 Mathematics Subject Classification. 60L10, 68Q32, 26A33

Agradecimientos

Me gustaría agradecerle profundamente al Dr. José Manuel Corcuera Valverde todo el apoyo, dedicación, tiempo y ayuda que me ha brindado a lo largo de este trabajo y que lo ha hecho posible.

También me gustaría agradecerle a mis compañeros de carrera y profesores su ayuda y los buenos momentos que he tenido el placer de compartir con ellos durante estos últimos años.

Finalmente, también quiero agradecerle a mi familia todo su apoyo, ayuda y consejos. Gracias a ellos he podido llegar hasta aquí y soy quién soy.

Índice

1. Introducción **1**

1.1. Estructura de la Memoria 1

2. Definición de la signatura **2**

3. Propiedades de la signatura **4**

3.1. Propiedades básicas 4

3.2. El producto *shuffle* 4

3.3. Identidad de Chen 6

4. Transformación logarítmica de la signatura **10**

4.1. Preliminares 10

4.2. Logaritmo de la signatura de un camino 13

5. Motivación de la signatura **16**

5.1. Interpretación geométrica 16

5.2. Relación de la signatura con la solución de EDOs controladas 20

6. Método de la signatura **24**

6.1. Construcción de caminos por interpolación lineal 24

6.2. Transformación *Lead-Lag* 25

6.3. Sumas parciales de secuencias de datos 29

6.4. Aplicación de la signatura en *machine learning* 31

7. Aplicación de la signatura al reconocimiento de dígitos manuscritos **32**

7.1. Base de datos MNIST 32

7.2. Implementación de la aplicación 32

7.3. Resultados de la aplicación 34

8. Generalizaciones fraccionarias de la signatura **35**

8.1. Signatura fraccionaria 35

8.2. Signatura fraccionaria discreta 40

8.3. Aplicación de la signatura fraccionaria discreta 43

9. Conclusiones **44**

1. Introducción

La signatura de un camino es una sucesión de integrales que se aplican iterativamente a las componentes del camino, que nos permite describirlo de manera precisa y resumir sus características, especialmente geométricas. Este fue un concepto que surgió en los años cincuenta y que fue estudiado originalmente por K. T. Chen entre otros autores, desarrollando la teoría y dando los primeros resultados notables que justificaban el interés por la signatura.

Posteriormente, la signatura formó parte de la teoría de caminos rugosos de Terry Lyons, clave en el área del cálculo estocástico y en el estudio de las ecuaciones diferenciales controladas por caminos rugosos. Gracias al desarrollo de esta teoría, la signatura se generalizó para aplicarse a ciertos caminos de variación finita y volvió a tomar cierta relevancia.

Más recientemente, también se han encontrado aplicaciones de la signatura en el campo del *machine learning*, donde sus propiedades para describir caminos son útiles para resumir las secuencias de datos con las que se trabaja en esta disciplina y para exponer sus propiedades, cosa que facilita el entrenamiento de los modelos que han de tomar decisiones a partir de los datos.

1.1. Estructura de la Memoria

Respecto a la estructura de la memoria, en las secciones 2, 3, 4 y 5 estudiaremos en detalle desde un punto de vista teórico este concepto que es la signatura de un camino, dando su definición, analizando sus propiedades y mostrando la estrecha relación que existe entre el camino y su signatura.

Posteriormente, en la sección 6, motivaremos y describiremos también su uso reciente en *machine learning*, donde nos permite construir atributos para los modelos que resalten la estructura geométrica de las secuencias de datos que se introducen. A modo de ejemplo de lo anterior, en la sección 7, incluiremos también una aplicación práctica de estos métodos para el reconocimiento de dígitos manuscritos.

Para acabar, en la sección 8, propondremos dos generalizaciones de la signatura, inspiradas en el cálculo fraccionario, de las que estudiaremos también algunas de sus propiedades y aplicaciones al *machine learning*, comparando sus resultados con los de la signatura original en el problema de reconocimiento de dígitos manuscritos.

2. Definición de la signatura

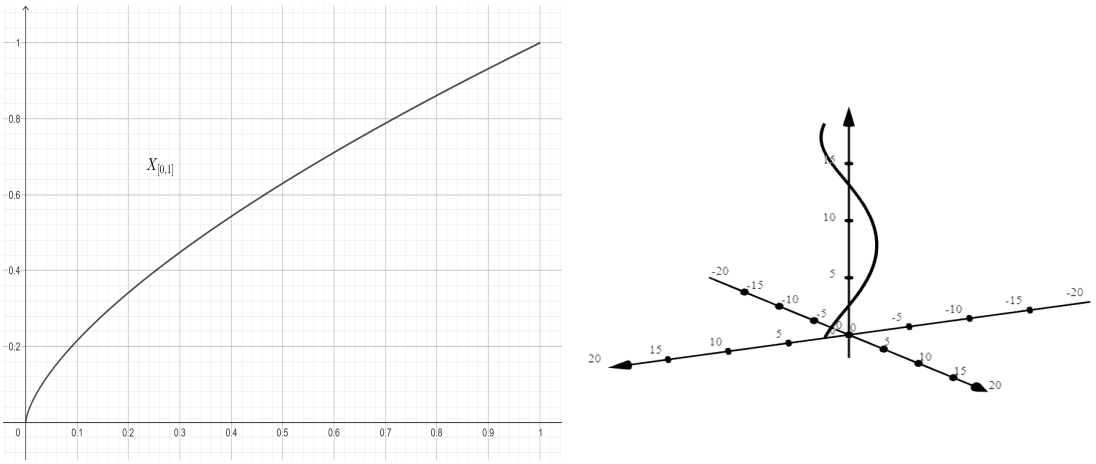
Para dar la definición formal de la signatura de un camino, antes de nada, necesitamos introducir diversos conceptos y nociones propias de la geometría diferencial, empezando por recordar la definición de un camino.

Definición 2.1 (Camino)**.** *Sea $[a,b] \subseteq \mathbb{R}$ un intervalo de $\mathbb{R}$ y sea también $d \geq 1 \in \mathbb{N}$, diremos que una aplicación $X : [a,b] \longrightarrow \mathbb{R}^d$ es un camino de $\mathbb{R}^d$ si es continua. Además, en este caso, llamaremos traza del camino X a su recorrido.*

Todo y eso, para poder reducirnos a la definición de signatura más elemental, nos restringiremos a considerar que todos los caminos con los que trabajamos son de clase C^1 a trozos. Es decir, en todo este proyecto, la noción de camino será la de una aplicación continua y de clase C^1 a trozos entre los espacios indicados. Aparte, por lo que respecta a la notación, sea $X : [a,b] \longrightarrow \mathbb{R}^d$ un camino cualquiera y $t \in [a,b]$ arbitrario, denotaremos la imagen de X en t como $X_t = \{X_t^1, \ldots, X_t^d\} \coloneqq (X^1(t), \ldots, X^d(t)) \in \mathbb{R}^d$, y, además, sea $A \subseteq [a,b]$ cualquiera, denotaremos también el conjunto imagen de A por X como $X_A \coloneqq X(A)$. En particular, con esta notación, $X_{[a,b]}$ es la traza del camino X.

Ejemplo 2.2. Podemos considerar el intervalo $[0,1]$ y la aplicación $X : [0,1] \longrightarrow \mathbb{R}^2$ dada por $X_t = \{t^3, t^2\} \quad \forall t \in [0,1]$, que es un camino de clase C^1 a trozos, porque sus componentes son funciones polinómicas y, por tanto, infinitamente diferenciables. En este caso, la traza de X es la indicada en la figura 1a.

Ejemplo 2.3. Sean $[0, 2\pi]$ y también $Y : [0, 2\pi] \longrightarrow \mathbb{R}^3$ la aplicación descrita de manera que $Y_t = \{2\cos t, 2\sin t, 3t\} \quad \forall t \in [0, 2\pi]$, como las funciones trigonométricas seno y coseno son infinitamente diferenciables, tenemos por definición que Y es en particular un camino de clase C^1 a trozos. De nuevo, mostramos en la figura 1b su traza, $Y_{[0,2\pi]}$.



(a) Traza del camino X (b) Traza del camino Y

Figura 1: Representación gráfica de las trazas

Aparte, pasamos también a introducir la noción de la integral de un camino contra otro (integral de Riemann-Stieltjes), necesaria para poder considerar la signatura de un camino.

Definición 2.4 (Integral de Riemann-Stieltjes)**.** *Sea $[a,b] \subseteq \mathbb{R}$ un intervalo de $\mathbb{R}$ y sean también X, $Y : [a,b] \longrightarrow \mathbb{R}$ caminos arbitrarios, definimos la integral de Y contra X, y*

la denotamos por $\int_a^b Y_t dX_t$, *como* $\int_a^b Y_t dX_t \coloneqq \int_a^b Y_t \cdot \dot{X}_t dt$, *donde esta segunda integral denota la integral de Riemann.*

Ejemplo 2.5. Sea $X : [0,1] \longrightarrow \mathbb{R}^2$ el camino considerado en el ejemplo anterior, entonces $X_t^1 = t^3$ y $X_t^2 = t^2 \quad \forall t \in [0,1]$. Podemos calcular la integral de cada una de las componentes de X contra la otra, aplicando la definición indicada: $\int_0^1 X_t^1 dX_t^2 \coloneqq \int_0^1 X_t^1 \dot{X}_t^2 dt = \int_0^1 t^3(2t)dt = 2\int_0^1 t^4 dt = \frac{2}{5}$ y, también, $\int_0^1 X_t^2 dX_t^1 \coloneqq \int_0^1 X_t^2 \dot{X}_t^1 dt = \int_0^1 t^2(3t^2)dt = 3\int_0^1 t^4 dt = \frac{3}{5}$.

Una vez recordados estos conceptos, podemos pasar a definir formalmente la signatura de un camino. Para ello, consideraremos la familia de todas las palabras formadas con un alfabeto de un cierto número de letras d, $\{1, \ldots, d\}$, que denotaremos por $W = \{(i_1, \ldots, i_k) | k \geq 1, \quad i_1, \ldots, i_k \in \{1, \ldots, d\}\}$. También, en este conjunto, tomaremos el orden lexicográfico, definido de manera que sean $I = (i_1, \ldots, i_k)$, $J = (j_1, \ldots, j_l) \in W$ cualesquiera, $I < J \iff k < l$ o $(k = l$ y $\{h | i_h \neq j_h\} \neq \emptyset$ y $i_{min\{h|i_h \neq j_h\}} < j_{min\{h|i_h \neq j_h\}})$.

Definición 2.6 (Signatura de un camino)**.** *Sea* $X : [a,b] \longrightarrow \mathbb{R}^d$ *un camino arbitrario y sea también* $I \in W$ *cualquiera,* $I = (i_1, \ldots, i_k)$*, podemos definir*

$$S(X)_{a,b}^I \coloneqq \int_{a<t_1 \ldots <t_k<b} dX_{t_1}^{i_1} dX_{t_2}^{i_2} \ldots dX_{t_k}^{i_k}$$

como el resultado de aplicar iterativamente las integrales de caminos con las componentes de X *indicadas por el multi-índice. Entonces, definimos la signatura de* X *como la sucesión* $S(X)_{a,b} = \{S(X)_{a,b}^I\}_{I \in W}$*, donde por convenio añadimos 1 como elemento inicial correspondiente a la palabra de 0 letras.*

Notemos, por tanto, que la signatura de un camino es una sucesión indexada a lo largo del conjunto de palabras W, donde consideramos los elementos de W ordenados siguiendo el orden lexicográfico ya descrito en niveles, según la longitud de las palabras. De hecho, es habitual tomar los elementos de la signatura formados por palabras de una cierta longitud, los denominados niveles de la signatura.

Definición 2.7. *Sean* $X : [a,b] \longrightarrow \mathbb{R}^d$ *un camino arbitrario,* $S(X)_{a,b} = \{S(X)_{a,b}^I\}_{I \in W}$ *su signatura y* $k \geq 1 \in \mathbb{N}$ *cualquiera, entonces definimos el nivel* k*-ésimo de la signatura de* X *como la colección de sus términos* $\{S(X)_{a,b}^I\}_{I \in W_k}$ *donde* $W_k = \{(i_1, \ldots, i_k) | i_1, \ldots i_k \in \{1, \ldots, d\}\}$*. Por convenio, definimos también el nivel* 0*-ésimo de la signatura como el formado únicamente por el elemento 1 inicial.*

Ejemplo 2.8. Sea $X : [0,1] \longrightarrow \mathbb{R}^2$ el camino dado por $X_t = \{X_t^1, X_t^2\} = \{t^3, t^2\} \quad \forall t \in [0,1]$, a modo de ejemplo calcularemos los dos primeros niveles de su signatura $S(X)_{0,1}$, dejando de lado el nivel 0-ésimo trivial. Para el primer nivel, $S(X)_{0,1}^1 = \int_{0<t<1} dX_t^1 = \int_0^1 \dot{X}_t^1 dt = \int_0^1 3t^2 dt = 1 - 0 = 1$ y $S(X)_{0,1}^2 = \int_{0<t<1} dX_t^2 = \int_0^1 \dot{X}_t^2 dt = \int_0^1 2t dt = 1 - 0 = 1$. Respecto al segundo nivel:

$$S(X)_{0,1}^{1,1} = \int_{0<t_1<t_2<1} dX_{t_1}^1 dX_{t_2}^1 = \int_0^1 (\int_0^{t_2} 3t_1^2 dt_1) 3t_2^2 dt_2 = \int_0^1 t_2^3 3t_2^2 dt_2 = \frac{1}{2}$$

$$S(X)_{0,1}^{1,2} = \int_{0<t_1<t_2<1} dX_{t_1}^1 dX_{t_2}^2 = \int_0^1 (\int_0^{t_2} 3t_1^2 dt_1) 2t_2 dt_2 = \int_0^1 t_2^3 2t_2 dt_2 = \frac{2}{5}$$

$$S(X)_{0,1}^{2,1} = \int_{0<t_1<t_2<1} dX_{t_1}^2 dX_{t_2}^1 = \int_0^1 (\int_0^{t_2} 2t_1 dt_1) 3t_2^2 dt_2 = \int_0^1 t_2^2 3t_2^2 dt_2 = \frac{3}{5}$$

$$S(X)_{0,1}^{2,2} = \int_{0<t_1<t_2<1} dX_{t_1}^2 dX_{t_2}^2 = \int_0^1 (\int_0^{t_2} 2t_1 dt_1) 2t_2 dt_2 = \int_0^1 t_2^2 2t_2 dt_2 = \frac{1}{2}$$

3. Propiedades de la signatura

3.1. Propiedades básicas

Una vez dada la definición de la signatura de un camino, pasamos a comentar brevemente sus propiedades, empezando por las más elementales.

Proposición 3.1 (Invariancia por translaciones)**.** *Sea $X : [a,b] \longrightarrow \mathbb{R}^d$ un camino cualquiera y sea también $c \in \mathbb{R}^d$ un vector arbitrario, consideramos el camino $X+c : [a,b] \longrightarrow \mathbb{R}^d$ dado por $(X+c)_t = X_t + c \quad \forall t \in [a,b]$ y entonces tenemos que se verifica que $S((X+c))_{a,b} = S(X)_{a,b}$, es decir, la signatura de X es invariante por translaciones.*

Demostración. Por definición de la signatura de un camino, para probar el resultado será suficiente concluir que la integral de un camino contra otro es invariante por translaciones del segundo camino. Para demostrarlo, sea $Y : [a,b] \longrightarrow \mathbb{R}$ un camino cualquiera y sean también $Z^1 : [a,b] \longrightarrow \mathbb{R}$ otro camino y $c \in \mathbb{R}$ arbitrarios, consideramos $Z^2 : [a,b] \longrightarrow \mathbb{R}$ tal que $Z^2_t = Z^1_t + c \quad \forall t \in [a,b]$ y, entonces, para empezar, es inmediato ver que $\dot{Z}^2_t = \dot{Z}^1_t \quad \forall t \in (a,b)$. En consecuencia, por definición de la integral de Riemann-Stieltjes, concluimos que $\int_a^b Y_t dZ^1_t = \int_a^b Y_t \dot{Z}^1_t dt = \int_a^b Y_t \dot{Z}^2_t dt = \int_a^b Y_t dZ^2_t$, que es exactamente lo que nos faltaba por ver, cosa que completa la prueba. □

También, otra propiedad inmediata de la signatura de un camino es el hecho de que es invariante por reparametrizaciones de clase C^1 del camino.

Definición 3.2. *Sean $[a,b],[c,d] \subseteq \mathbb{R}$ intervalos de $\mathbb{R}$, entonces definimos una reparametrización de $[c,d]$ en $[a,b]$ como una aplicación $\phi : [c,d] \longrightarrow [a,b]$ continua, no decreciente y exhaustiva. Además, decimos que ϕ es una reparametrización de clase C^1 si lo es como aplicación.*

Proposición 3.3 (Invariancia de la signatura por reparametrizaciones)**.** *Sea $X : [a,b] \longrightarrow \mathbb{R}^d$ un camino cualquiera y sea también $\phi : [c,d] \longrightarrow [a,b]$ una reparametrización de clase C^1 de $[c,d]$ en $[a,b]$, consideramos el camino $\overline{X} : [c,d] \longrightarrow \mathbb{R}^d$ dado por $\overline{X}_t := X_{\phi(t)}$ $\forall t \in [c,d]$ y entonces tenemos que $S(\overline{X})_{c,d} = S(X)_{a,b}$, es decir, las signaturas de los dos caminos coinciden.*

Demostración. Para probar este resultado, por definición de los términos de la signatura, será suficiente ver que la integral de caminos es invariante para estas reparametrizaciones. En efecto, sean $X, Y : [a,b] \longrightarrow \mathbb{R}$ caminos reales cualesquiera y sea $\phi : [c,d] \longrightarrow [a,b]$ una reparametrización de clase C^1 de $[c,d]$ en $[a,b]$, consideramos los caminos reparametrizados $\overline{X}, \overline{Y} : [c,d] \longrightarrow \mathbb{R}$ y, para empezar, sea $t \in [c,d]$ arbitrario, tenemos que $\dot{\overline{X}}_t = \dot{X}_{\phi(t)}\dot{\phi}(t)$. Entonces, a partir de aquí, podemos aplicar la definición de la integral de un camino contra otro y concluir que $\int_c^d \overline{Y}_t d\overline{X}_t = \int_c^d \overline{Y}_t \dot{\overline{X}}_t dt = \int_c^d Y_{\phi(t)} \dot{X}_{\phi(t)} \dot{\phi}_t dt =$ (por la fórmula del cambio de variable de las integrales de Riemann, teniendo presente las propiedades de ϕ) $= \int_a^b Y_u \dot{X}_u du = \int_a^b Y_u dX_u$, de manera que las dos integrales coinciden. □

3.2. El producto *shuffle*

Además, otra de las propiedades características de la signatura es el hecho de que cualquier producto de sus elementos se expresa como una suma de otros de sus elementos.

Para enunciar este resultado, antes de nada, tenemos que introducir diversas nociones, como la de permutación *shuffle* y la de producto *shuffle* de dos multi-índices.

Definición 3.4 (Permutación *shuffle*)**.** *Sean $k, m \geq 1 \in \mathbb{N}$ arbitrarios y sea también $\sigma \in S_{k+m}$ una permutación de $k+m$ elementos cualquiera, diremos que σ es un (k,m)-shuffle si verifica que $\sigma^{-1}(1) < \ldots < \sigma^{-1}(k)$ y $\sigma^{-1}(k+1) < \ldots < \sigma^{-1}(k+m)$. Además, podemos extender esta definición de manera natural al caso en que $k = 0$ o $m = 0$ no los dos nulos, de manera que σ es un (k,m)-shuffle si $\sigma = id \in S_{k+m}$. En ambos casos, denotamos por Shuffles$(k,m) \subseteq S_{k+m}$ al conjunto de todos los (k,m)-shuffles.*

Por tanto, informalmente, podemos observar que las permutaciones (k,m)-*shuffle* son aquellas que tienen como imagen ordenada en conjunto de valores $(1,\ldots,k)$ y $(k+1,\ldots,k+m)$ superpuestos e intercalados pero apareciendo en orden los elementos de $(1,\ldots,k)$ entre ellos y también los de $(k+1,\ldots,k+m)$.

Ejemplo 3.5. Por ejemplo, si tomamos $k = 3$ y $m = 2$, las permutaciones $\sigma \in S_5$ que son $(3,2)$-*shuffles* son, por definición, todas aquellas que verifican que $\sigma^{-1}(1) < \sigma^{-1}(2) < \sigma^{-1}(3)$ y $\sigma^{-1}(4) < \sigma^{-1}(5)$. En particular, si definimos $\sigma \in S_5$ como la permutación dada de manera que:

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 1 & 2 & 5 & 3 \end{pmatrix}$$

Podemos ver que σ es en efecto un $(3,2)$-*shuffle*, ya que $\sigma^{-1}(1) < \sigma^{-1}(2) < \sigma^{-1}(3)$ y $\sigma^{-1}(4) < \sigma^{-1}(5) \iff 2 < 3 < 5$ y $1 < 4$, cosa claramente cierta.

Definición 3.6 (Producto *shuffle*)**.** *Sean $I = (i_1,\ldots,i_k)$, $J = (j_1,\ldots,j_m) \in W$ multi-índices cualesquiera de longitudes $k \geq 1$ y $m \geq 1$ respectivamente, definimos el producto shuffle de I y J, y lo denotamos por $I \sqcup J$, como $I \sqcup J \coloneqq \{(a_{\sigma(1)},\ldots,a_{\sigma(k+m)}) |\ \ \sigma \in$ Shuffles$(k,m)\}$ de manera que $(a_1,\ldots,a_k,a_{k+1},\ldots,a_{k+m}) = (i_1,\ldots,i_k,j_1,\ldots,j_m)$. Por convenio, además, podemos extender esta definición al en caso en que una de las dos palabras se corresponda con la palabra degenerada de 0 letras de manera natural, utilizando la definición de Shuffles(k,m) en el caso en que $k = 0$ o $m = 0$.*

Ejemplo 3.7. Continuando con el ejemplo anterior, donde habíamos considerado la permutación $\sigma \in S_5$ ya definida con notación matricial y visto que σ era un $(3,2)$-*shuffle*, podemos ahora tomar las palabras pertenecientes a W $I = (i_1,i_2,i_3), J = (j_1,j_2)$ de 3 y 2 letras respectivamente. Como $\sigma \in$ *Shuffles*$(3,2)$, entonces recordando su definición concluimos que $(j_1,i_1,i_2,j_2,i_3) \in I \sqcup J$. Notemos que el último elemento de (j_1,i_1,i_2,j_2,i_3) es i_3 y, por tanto, es el último elemento de I. Esto es un hecho general, las palabras de $I \sqcup J$ tienen como último elemento el último de I o el de J, cosa que se obtiene como consecuencia directa de las desigualdades que dan lugar a la definición de las permutaciones *shuffle*.

A partir de aquí, podemos observar que la signatura de un camino tiene la propiedad comentada.

Teorema 3.8 (Identidad del producto *shuffle*)**.** *Sea $X : [a,b] \longrightarrow \mathbb{R}^d$ un camino arbitrario y sea $S(X)_{a,b} = \{S(X)^I_{a,b}\}_{I \in W}$ su signatura, consideramos multi-índices I, $J \in W$ cualesquiera y entonces podemos afirmar que se verifica la igualdad entre los términos de la signatura $S(X)^I_{a,b} S(X)^J_{a,b} = \sum_{K \in I \sqcup\sqcup J} S(X)^K_{a,b}$.*

Demostración. En efecto, continuando con las notaciones introducidas por el enunciado, probaremos que se cumple que $\forall I = (i_1, \dots, i_k), J = (j_1, \dots, j_m) \in W$, tenemos que $S(X)^I_{a,b} S(X)^J_{a,b} = \sum_{K \in I \sqcup\!\sqcup J} S(X)^K_{a,b}$ y para ello procederemos por inducción en $k + m$.

Caso inicial ($k + m = 1$): Para empezar, si $k + m = 1$, entonces como k y m se corresponden con las longitudes de los multi-índices, necesariamente tenemos que $k = 1$ y $m = 0$ o viceversa. Supondremos que estamos en el caso $k = 1$ y $m = 0$, el otro se prueba de manera análoga. Por tanto, sea $i \in \{1, \dots, d\}$ cualquiera, considerando $I = (i) \in W$ y J la palabra de 0 letras, tenemos que $I \sqcup\!\sqcup J = \{(i)\}$, ya que *Shuffles*$(1, 0) \subseteq S_1$ contiene solo por definición la permutación identidad. Por tanto, como $S(X)^I_{a,b} S(X)^J_{a,b} = S(X)^i_{a,b} \cdot 1 = S(X)^i_{a,b} = \sum_{K \in I \sqcup\!\sqcup J} S(X)^K_{a,b}$, queda probado el resultado en este caso inicial.

Caso sucesor: A partir de aquí, suponemos que el resultado es cierto para valores más pequeños que $k + m \geq 2$ y pasamos a probarlo para $k + m$. En efecto, sean $I = (i_1, \dots, i_k), J = (j_1, \dots, j_m) \in W$ arbitrarios, queremos concluir que $S(X)^I_{a,b} S(X)^J_{a,b} = \sum_{K \in I \sqcup\!\sqcup J} S(X)^K_{a,b}$. Para ello, podemos observar que: $S(X)^I_{a,b} S(X)^J_{a,b} =$ (fórmula de integración por partes) $\int_a^b S(X)^I_{a,t} dS(X)^J_{a,t} + \int_a^b S(X)^J_{a,t} dS(X)^I_{a,t} = \int_a^b S(X)^I_{a,t} (S(\dot{X})^J_{a,t}) dt + \int_a^b S(X)^J_{a,t} (S(\dot{X})^I_{a,t}) dt =$ (aplicando la definición de la signatura y el Teorema Fundamental del Cálculo) $= \int_a^b S(X)^I_{a,t} S(X)^{j_1,\dots,j_{m-1}}_{a,t} \dot{X}^{j_m}_t dt + \int_a^b S(X)^J_{a,t} S(X)^{i_1,\dots,i_{k-1}}_{a,t} \dot{X}^{i_k}_t dt = \int_a^b S(X)^I_{a,t} S(X)^{j_1,\dots,j_{m-1}}_{a,t} dX^{j_m}_t + \int_a^b S(X)^J_{a,t} S(X)^{i_1,\dots,i_{k-1}}_{a,t} dX^{i_k}_t =$ (aplicando la hipótesis de inducción) $= \sum_{K \in I \sqcup\!\sqcup (j_1,\dots,j_{m-1})} \int_a^b S(X)^K_{a,t} dX^{j_m}_t + \sum_{K \in (i_1,\dots,i_{k-1}) \sqcup\!\sqcup J} \int_a^b S(X)^K_{a,t} dX^{i_k}_t = \sum_{K \in I \sqcup\!\sqcup (j_1,\dots,j_{m-1})} S(X)^{K,j_m}_{a,b} + \sum_{K \in (i_1,\dots,i_{k-1}) \sqcup\!\sqcup J} S(X)^{K,i_k}_{a,b} = \sum_{K \in I \sqcup\!\sqcup J} S(X)^K_{a,b}$.

Notemos que esta última igualdad se debe al hecho de que todos los multi-índices de $I \sqcup\!\sqcup J$, por definición del producto *shuffle* y de las permutaciones (k, m)-*shuffle*, tienen como último elemento el último de I o el último de J, es decir, i_k o j_m. Por tanto, podemos separar las dos clases de multi-índices de la manera indicada por la igualdad anterior. Queda, en consecuencia, demostrado también el caso sucesor.

Por tanto, en conjunto, por el principio de inducción matemática, queda probado el resultado. □

Ejemplo 3.9. A modo de ejemplo sencillo de la propiedad anterior, recuperando aquello ya estudiado en el ejemplo 2.8, sea $X : [0, 1] \longrightarrow \mathbb{R}^2$ el camino ya considerado dado por $X_t = \{X^1_t, X^2_t\} = \{t^3, t^2\} \quad \forall t \in [0, 1]$, pasamos a estudiar el producto $S(X)^1_{0,1} S(X)^2_{0,1}$ usando la identidad anterior. En efecto, como en este caso $I = (1)$ y $J = (2)$, tenemos que por definición del producto *shuffle* $I \sqcup\!\sqcup J = \{(a_{\sigma(1)}, a_{\sigma(2)}) | \quad \sigma \in$ *Shuffles*$(1, 1)\}$ donde $(a_1, a_2) = (1, 2)$. Como los $(1, 1)$-*shuffles* son claramente las dos permutaciones de S_2, tenemos que $I \sqcup\!\sqcup J = \{(1, 2), (2, 1)\}$. En consecuencia, utilizando los valores de los términos de la signatura de X ya calculados, podemos comprobar que $S(X)^1_{0,1} S(X)^2_{0,1} = 1 \cdot 1 = 1 = \frac{2}{5} + \frac{3}{5} = S(X)^{1,2}_{0,1} + S(X)^{2,1}_{0,1} = \sum_{K \in I \sqcup\!\sqcup J} S(X)^K_{0,1}$.

3.3. Identidad de Chen

Otra de las propiedades más sorprendentes que verifica la signatura de un camino es la conocida como identidad de Chen.

Para introducirla, antes de nada, necesitamos considerar el $\mathbb{R}$-álgebra siguiente.

Definición 3.10. *Sean $e_1, \dots, e_d$ indeterminadas arbitrarias no conmutativas, consideramos entonces dos monomios $e_{i_1} \dots e_{i_k}, e_{j_1} \dots e_{j_m}$ cualesquiera y podemos definir su pro-*

ducto $\otimes$ *como* $e_{i_1} \dots e_{i_k} \otimes e_{j_1} \dots e_{j_m} \coloneqq e_{i_1} \dots e_{i_k} e_{j_1} \dots e_{j_m}$*, es decir, el producto dado por la yuxtaposición de indeterminadas.*

Definición 3.11 (ℝ-álgebra de las series formales)**.** *Sean* $e_1, \dots, e_d$ *indeterminadas arbitrarias no conmutativas, consideramos entonces el conjunto de las expresiones formales* $\sum_{k=0}^{\infty} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} \lambda_{i_1,\dots,i_k} e_{i_1} \dots e_{i_k}$ *con coeficientes reales. Este conjunto, con las operaciones naturales de suma y producto por escalar inducidas por las de* $\mathbb{R}$ *y el producto* $\otimes$ *extendido de manera que, sean* $x = \sum_{k=0}^{\infty} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} \lambda_{i_1,\dots,i_k} e_{i_1} \dots e_{i_k}$ *e* $y = \sum_{k=0}^{\infty} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} \mu_{i_1,\dots,i_k} e_{i_1} \dots e_{i_k}$ *series formales cualesquiera:*

$$x \otimes y = \sum_{k=0}^{\infty} \sum_{p=0}^{k} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} \lambda_{i_1,\dots,i_p} \mu_{i_{p+1},\dots,i_k} e_{i_1} \dots e_{i_k}$$

tiene una estructura de $\mathbb{R}$*-álgebra, que denotamos por* $\mathbb{R}\langle\langle e_1, \dots, e_d \rangle\rangle$*.*

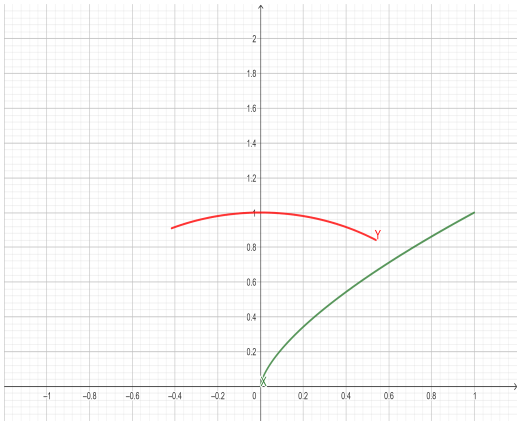
Ejemplo 3.12. En particular, sean $e_1, \dots, e_d$ indeterminadas arbitrarias no conmutativas, tomamos el ℝ-álgebra definida previamente, $\mathbb{R}\langle\langle e_1, \dots, e_d \rangle\rangle$, y observamos que sea $x = \sum_{k=0}^{\infty} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} \lambda_{i_1,\dots,i_k} e_{i_1} \dots e_{i_k} \in \mathbb{R}\langle\langle e_1, \dots, e_d \rangle\rangle$, si escogemos x de manera que $\exists N \in \mathbb{N}$ tal que $\forall k > N$ y $i_1, \dots, i_k \in \{1, \dots, d\}$ se verifica que $\lambda_{i_1,\dots,i_k} = 0$, entonces, $x = \sum_{k=0}^{N} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} \lambda_{i_1,\dots,i_k} e_{i_1} \dots e_{i_k}$ y es un polinomio no conmutativo en estas indeterminadas. Por tanto, tenemos que podemos pensar los polinomios no conmutativos en las indeterminadas $e_1, \dots, e_d$ como elementos de $\mathbb{R}\langle\langle e_1, \dots, e_d \rangle\rangle$. De hecho, si denotamos por $\mathbb{R}\langle e_1, \dots, e_d \rangle$ el conjunto de los polinomios no conmutativos en las indeterminadas $e_1, \dots, e_d$ con coeficientes reales, hemos visto que se puede pensar $\mathbb{R}\langle e_1, \dots, e_d \rangle \subseteq \mathbb{R}\langle\langle e_1, \dots, e_d \rangle\rangle$. Además, es fácil ver que $\mathbb{R}\langle e_1, \dots, e_d \rangle$ es una subálgebra de $\mathbb{R}\langle\langle e_1, \dots, e_d \rangle\rangle$, mostrando que operando polinomios no conmutativos obtenemos polinomios no conmutativos. No detallamos la prueba porque el estudio de estos polinomios se escapa de los objetivos del trabajo.

A partir de aquí, aplicado al estudio y descripción de la signatura de un camino, sea $X : [a, b] \longrightarrow \mathbb{R}^d$ un camino cualquiera, podemos utilizar las d indeterminadas para hacer referencia a las palabras construidas con un alfabeto de d letras, las cuales son elementos del conjunto W que indexa los términos de la signatura de X. A través de esta identificación, podemos pensar la signatura de X como un elemento del álgebra, de manera que $S(X)_{a,b} = \sum_{k=0}^{\infty} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} S(X)_{a,b}^{i_1,\dots,i_k} e_{i_1} \dots e_{i_k}$. En este contexto, podemos observar que se verifica la identidad de Chen que relaciona las signaturas de dos caminos y su camino concatenado.

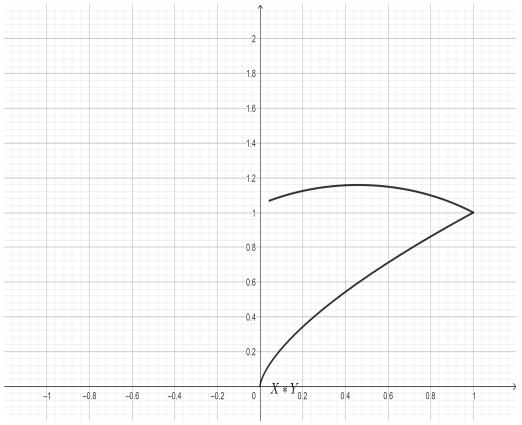
Definición 3.13 (Concatenación de caminos)**.** *Sean* $X : [a, b] \longrightarrow \mathbb{R}^d$ *y* $Y : [b, c] \longrightarrow \mathbb{R}^d$ *caminos cualesquiera, definimos la concatenación de* X *y* Y *como el camino* $X * Y : [a, c] \longrightarrow \mathbb{R}^d$ *dado por* $(X * Y)_t = X_t$ *si* $t \in [a, b]$ *y* $(X * Y)_t = X_b + (Y_t - Y_b)$ *si* $t \in [b, c]$*.*

Ejemplo 3.14. Si consideramos los caminos $X : [0, 1] \longrightarrow \mathbb{R}^2$ dado por $X_t = \{t^3, t^2\}$ $\forall t \in [0, 1]$ y $Y : [1, 2] \longrightarrow \mathbb{R}^2$ dado por $Y_t = \{\cos(t), \sin(t)\}$ $\quad \forall t \in [1, 2]$, entonces tenemos por definición que $X * Y : [0, 2] \longrightarrow \mathbb{R}^2$ viene descrito por $(X * Y)_t = \{t^3, t^2\}$ si $t \in [0, 1]$ y $(X * Y)_t = \{1 + \cos t - \cos 1, 1 + \sin t - \sin 1\}$ si $t \in [1, 2]$. Podemos representar en este caso la traza de $X * Y$ y compararla con las de X y Y, tal como se muestra en la figura 2.

Teorema 3.15 (Identidad de Chen)**.** *Sean* $X : [a, b] \longrightarrow \mathbb{R}^d$ *y* $Y : [b, c] \longrightarrow \mathbb{R}^d$ *caminos arbitrarios y* $X * Y : [a, c] \longrightarrow \mathbb{R}^d$ *su concatenación, entonces se verifica que* $S(X * Y)_{a,c} = S(X)_{a,b} \otimes S(Y)_{b,c}$*.*



(a) Trazas de X en verde y Y en rojo



(b) Traza del camino $X * Y$

Figura 2: Comparación de las trazas de los caminos y su concatenación

Demostración. Para demostrarlo, si denotamos por comodidad $Z = X * Y$, probaremos primero un resultado previo sobre Z. Afirmamos que, sea W el conjunto de todas las palabras construidas con el alfabeto de d letras $\{1, \dots, d\}$, se verifica que $\forall I = (i_1, \dots, i_k) \in W$ y $\forall h \in [a, c]$ tenemos $S(Z)^I_{a,c} = S(Z)^{i_1,\dots,i_k}_{a,h} + S(Z)^{i_1,\dots,i_{k-1}}_{a,h} S(Z)^{i_k}_{h,c} + \cdots + S(Z)^{i_1,\dots,i_r}_{a,h} S(Z)^{i_{r+1},\dots,i_k}_{h,c} + \cdots + S(Z)^{i_1,\dots,i_k}_{h,c}$. Pasamos, por tanto, a probar la afirmación, procediendo por inducción sobre k.

Caso inicial ($k = 1$): Por un lado, es evidente que, sea $i \in \{1, \dots, d\}$ cualquiera y $h \in [a, c]$ arbitraria, tenemos que $S(Z)^i_{a,c} = \int_a^c dZ^i_t = \int_a^c \dot{Z}^i_t dt = \int_a^h \dot{Z}^i_t dt + \int_h^c \dot{Z}^i_t dt = \int_a^h dZ^i_t + \int_h^c dZ^i_t = S(Z)^i_{a,h} + S(Z)^i_{h,c}$. Esta es la igualdad que teníamos que mostrar particularizada a $k = 1$, cosa que completa este caso de la inducción.

Caso sucesor: A partir de aquí, suponemos por hipótesis de inducción que el resultado se verifica para $k-1 \geq 1$ y pasamos a demostrar que entonces también es cierto para k. En efecto, sean $I = (i_1, \dots, i_k) \in W$ y $h \in [a, c]$ arbitrarios, podemos ver que por definición de los términos de la signatura de Z, que se obtienen iterativamente integrando los anteriores, tenemos: $S(Z)^I_{a,c} = \int_a^c S(Z)^{i_1,\dots,i_{k-1}}_{a,t} dZ^{i_k}_t = \int_a^h S(Z)^{i_1,\dots,i_{k-1}}_{a,t} dZ^{i_k}_t + \int_h^c S(Z)^{i_1,\dots,i_{k-1}}_{a,t} dZ^{i_k}_t =$ $S(Z)^{i_1,\dots,i_k}_{a,h} + \int_h^c (S(Z)^{i_1,\dots,i_{k-1}}_{a,h} + S(Z)^{i_1,\dots,i_{k-2}}_{a,h} S(Z)^{i_{k-1}}_{h,t} + \cdots + S(Z)^{i_1,\dots,i_r}_{a,h} S(Z)^{i_{r+1},\dots,i_{k-1}}_{h,t} + \cdots + S(Z)^{i_1,\dots,i_{k-1}}_{h,t}) dZ^{i_k}_t =$(por la linealidad de la integral y aplicando la definición) $=$ $S(Z)^{i_1,\dots,i_k}_{a,h} + S(Z)^{i_1,\dots,i_{k-1}}_{a,h} S(Z)^{i_k}_{h,c} + \cdots + S(Z)^{i_1,\dots,i_r}_{a,h} S(Z)^{i_{r+1},\dots,i_k}_{h,c} + \cdots + S(Z)^{i_1,\dots,i_k}_{h,c}$.

Esta es exactamente la igualdad que queríamos ver, cosa que completa el caso sucesor.

Por tanto, por el principio de inducción matemática, queda probada esta primera afirmación. A partir de aquí, podemos observar que como $\forall t \in [a, b], Z_t = X_t$ y $\forall t \in [b, c], Z_t = Y_t + v$ para algún $v \in \mathbb{R}^d$ constante, podemos deducir de la afirmación anterior que, en particular, $\forall I = (i_1, \dots, i_k) \in W$ tenemos que $S(Z)^I_{a,c} = S(X)^{i_1,\dots,i_k}_{a,b} + S(X)^{i_1,\dots,i_{k-1}}_{a,b} S(Y)^{i_k}_{b,c} + \cdots + S(X)^{i_1,\dots,i_r}_{a,b} S(Y)^{i_{r+1},\dots,i_k}_{b,c} + \cdots + S(Y)^{i_1,\dots,i_k}_{b,c}$.

Entonces, si pensamos las signaturas de X y Y como elementos de $\mathbb{R}\langle\langle e_1, \dots, e_d \rangle\rangle$, podemos ver que $S(X)_{a,b} \otimes S(Y)_{b,c} = \sum_{k=0}^{\infty} \sum_{i_1,\dots i_k \in \{1,\dots,d\}} c_{i_1,\dots,i_k} e_{i_1} \dots e_{i_k}$, donde por definición del producto $\otimes$, recorriendo todas las maneras de obtener $e_{i_1} \dots e_{i_k}$ por yuxtaposición, tenemos que $c_{i_1,\dots,i_k} = S(X)^{i_1,\dots,i_k}_{a,b} + S(X)^{i_1,\dots,i_{k-1}}_{a,b} S(Y)^{i_k}_{b,c} + \cdots + S(X)^{i_1,\dots,i_r}_{a,b} S(Y)^{i_{r+1},\dots,i_k}_{b,c} + \cdots + S(Y)^{i_1,\dots,i_k}_{b,c} =$ (por lo ya demostrado) $= S(Z)^{i_1,\dots,i_k}_{a,c}$. Esto prueba que $S(X * Y)_{a,c} =$

$S(Z)_{a,c} = S(X)_{a,b} \otimes S(Y)_{b,c}$, ya que las dos signaturas se expresan de la misma manera como elementos del álgebra. □

Por otro lado, continuando con la idea de la identidad anterior, podemos estudiar la relación de la signatura de un camino y la del camino recorrido en sentido contrario, pasando para empezar a recordar definición de este último concepto.

Definición 3.16 (Camino opuesto de un camino)**.** *Sea $X : [a,b] \longrightarrow \mathbb{R}^d$ un camino, definimos entonces el camino opuesto de X, recorrido en sentido contrario, como el camino $\overleftarrow{X} : [a,b] \longrightarrow \mathbb{R}^d$ dado por $\overleftarrow{X}_t \coloneqq X_{a+b-t} \quad \forall t \in [a,b]$.*

Entonces, se verifica, pensando sus signaturas como elementos del álgebra anterior, la siguiente igualdad.

Corolario 3.17. *Sea $X : [a,b] \longrightarrow \mathbb{R}^d$ un camino arbitrario, tomamos entonces su camino opuesto $\overleftarrow{X} : [a,b] \longrightarrow \mathbb{R}^d$ y podemos afirmar que se verifica $S(X)_{a,b} \otimes S(\overleftarrow{X})_{a,b} = 1$.*

Demostración. En efecto, continuando con las notaciones del enunciado anterior, podemos considerar entonces el camino $Z : [a, b + (b-a)] \longrightarrow \mathbb{R}^d$ definido de manera que $Z_t = X_t \quad \forall t \in [a,b]$ y $Z_{t+(b-a)} = \overleftarrow{X}_t \quad \forall t \in [a,b]$. A partir de aquí, aplicando la identidad de Chen y teniendo presente el resultado más general deducido durante su demostración, podemos afirmar que $S(Z)_{a,b+(b-a)} = S(X)_{a,b} \otimes S(Z)_{b,b+(b-a)} =$ (como el camino Z coincide en el intervalo $[b, b+(b-a)]$ con el camino $\overleftarrow{X}$ en el intervalo $[a,b]$, por definición de la signatura y propiedades de las integrales) $= S(X)_{a,b} \otimes S(\overleftarrow{X})_{a,b}$. En consecuencia, para probar el corolario, será suficiente ver que $S(Z)_{a,c} = 1$, donde $c \coloneqq b + (b-a)$. Para ello, por definición de la expresión de la signatura de un camino como elemento de $\mathbb{R}\langle\langle e_1, \ldots, e_d \rangle\rangle$, será suficiente deducir que $\forall I = (i_1, \ldots, i_k) \in W$ un multi-índice, $S(Z)^I_{a,c} = 0$ y, para hacerlo, probaremos una afirmación más general: $\forall I = (i_1, \ldots, i_k) \in W$ y $h \in [a,b]$, entonces $S(Z)^I_{a,h} = S(Z)^I_{a,c+a-h}$, por inducción en k. Notemos que si conseguimos demostrarla, en particular, para $h = a$, tendremos que $S(Z)^I_{a,c} = S(Z)^I_{a,a} = 0$, cosa que completa la prueba.

Caso inicial ($k = 1$): Para empezar, sea $i \in \{1, \ldots, d\}$ arbitrario y $h \in [a,b]$ también cualquiera, queremos ver que $S(Z)^i_{a,h} = Z^i_h - Z^i_a = X^i_h - Z^i_a = \overleftarrow{X}^i_{a+b-h} - Z^i_a = Z^i_{(a+b-h)+(b-a)} - Z^i_a = Z^i_{c+a-h} - Z^i_a = S(Z)^i_{a,c+a-h}$. Por tanto, esto completa el caso inicial.

Caso sucesor: Suponemos ahora que el resultado es cierto para multi-índices de longitud $k-1 \geq 1$ y pasamos a demostrarlo para los de k. Sea $I = (i_1, \ldots, i_k) \in W$ y $h \in [a,b]$, $S(Z)^I_{a,c+a-h} = \int_a^{c+a-h} S(Z)^{i_1,\ldots,i_{k-1}}_{a,t} dZ^{i_k}_t = \int_a^h S(Z)^{i_1,\ldots,i_{k-1}}_{a,t} dZ^{i_k}_t + \int_h^b S(Z)^{i_1,\ldots,i_{k-1}}_{a,t} dZ^{i_k}_t + \int_b^{c+a-h} S(Z)^{i_1,\ldots,i_{k-1}}_{a,c+a-(c+a-t)} dZ^{i_k}_t = S(Z)^I_{a,h} + \int_h^b S(Z)^{i_1,\ldots,i_{k-1}}_{a,t} dZ^{i_k}_t + \int_b^{c+a-h} S(Z)^{i_1,\ldots,i_{k-1}}_{a,c+a-t} dZ^{i_k}_t = S(Z)^I_{a,h}$.

Para justificar esta última igualdad, podemos observar que $\int_b^{c+a-h} S(Z)^{i_1,\ldots,i_{k-1}}_{a,c+a-t} dZ^{i_k}_t = \int_b^{c+a-h} S(Z)^{i_1,\ldots,i_{k-1}}_{a,c+a-t} \dot{Z}^{i_k}_t dt = -\int_b^{c+a-h} S(Z)^{i_1,\ldots,i_{k-1}}_{a,c+a-t} \dot{Z}^{i_k}_{c+a-t} dt =$(aplicando el cambio de variable $c+a-t = u$ en la integral) $= -\int_h^b S(Z)^{i_1,\ldots,i_{k-1}}_{a,u} \dot{Z}^{i_k}_u dt = -\int_h^b S(Z)^{i_1,\ldots,i_{k-1}}_{a,u} dZ^{i_k}_u$. Notemos que en esta prueba hemos utilizado que $\forall t \in (b, c+a-h)$ $\dot{Z}^{i_k}_t = (\overleftarrow{X}^{i_k}_{t-(b-a)}\dot{}) = (X^{i_k}_{2b-t}\dot{}) = -\dot{X}^{i_k}_{2b-t} = -\dot{X}^{i_k}_{c+a-t} = -\dot{Z}^{i_k}_{c+a-t}$. Esto deja resuelto también el caso sucesor.

Por el principio de inducción matemática, queda demostrado el resultado. □

4. Transformación logarítmica de la signatura

4.1. Preliminares

Una vez dadas las propiedades de la signatura, es necesario definir y comentar la transformación de la signatura a la signatura logarítmica de un camino. Para ello, nos interesa primero considerar, a modo de herramienta para la sección, la siguiente construcción. Sean $e_1, \dots, e_d$ indeterminadas no conmutativas arbitrarias y sean también $e'_1, \dots, e'_d$ indeterminadas no conmutativas cualesquiera tal que sí que conmutan con las anteriores, es decir, se tiene $e_i e'_j = e'_j e_i \, \forall i, j \in \{1, \dots, d\}$, entonces, podemos considerar el $\mathbb{R}$-álgebra, que denotaremos por A, de las series formales de coeficientes reales en las indeterminadas $e_1, \dots, e_d, e'_1, \dots, e'_d$, con las operaciones de suma y producto por escalar inducidas por las de $\mathbb{R}$ y el producto $\otimes$ definido de manera que extiende el de $\mathbb{R}\langle\langle e_1, \dots, e_d \rangle\rangle$: sean $x = \sum_{k=0}^{\infty} \sum_{l=0}^{k} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} \lambda^l_{i_1,\dots,i_k} e_{i_1} \dots e_{i_l} e'_{i_{l+1}} \dots e'_{i_k}$, $y = \sum_{k=0}^{\infty} \sum_{l=0}^{k} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} \mu^l_{i_1,\dots,i_k} e_{i_1} \dots e_{i_l} e'_{i_{l+1}} \dots e'_{i_k}$ cualesquiera, definimos que $x \otimes y =$

$$\sum_{k=0}^{\infty} \sum_{p=0}^{k} \sum_{l=0}^{p} \sum_{h=0}^{k-p} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} \lambda^l_{i_1,\dots,i_p} \mu^h_{i_{p+1},\dots,i_k} e_{i_1} \dots e_{i_l} e_{i_{p+1}} \dots e_{i_{p+h}} e'_{i_{l+1}} \dots e'_{i_p} e'_{i_{p+h+1}} \dots e'_{i_k}$$

Es cierto que esta es una construcción que se puede generalizar de diversas maneras y entender en múltiples contextos, pero como únicamente la necesitaremos para dar sentido a ciertas expresiones que aparecerán en la sección, nos restringiremos a introducir solo aquello que nos es necesario.

Dicho esto, sea $x \in \mathbb{R}\langle\langle e_1, \dots, e_d \rangle\rangle$ una serie formal cualquiera, de manera que, por tanto, $x = \sum_{k=0}^{\infty} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} \lambda_{i_1,\dots,i_k} e_{i_1} \dots e_{i_k}$ donde los coeficientes son reales cualesquiera, podemos pensar entonces en particular que $x \in A$. De hecho, como se deduce de las definiciones que las operaciones en A se obtienen extendiendo las de $\mathbb{R}\langle\langle e_1, \dots, e_d \rangle\rangle$, se puede observar que $\mathbb{R}\langle\langle e_1, \dots, e_d \rangle\rangle$ es una subálgebra de A. Además, podemos hacer explícitas las indeterminadas de las que depende x, denotando $x = x(e_1, \dots, e_d)$, y generalizar esta notación para incluir $x(s_1 e_1 + s'_1 e'_1, \dots, s_d e_d + s'_d e'_d) \coloneqq \sum_{k=0}^{\infty} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} \lambda_{i_1,\dots,i_k} ((s_{i_1} e_{i_1} + s'_{i_1} e'_{i_1}) \otimes \dots \otimes (s_{i_k} e_{i_k} + s'_{i_k} e'_{i_k})) \in A$ donde $\forall i \in \{1, \dots, d\}, s_i, s'_i \in \mathbb{R}$.

Una vez introducido A, observamos que, en esta álgebra, podemos definir las nociones de exponencial y logaritmo formales, dando alguna de sus propiedades.

Definición 4.1 (Exponencial de una serie)**.** *Sea $x \in A$ una serie formal con término independiente nulo cualquiera, entonces definimos su exponencial como la serie dada por $\exp(x) = \sum_{n \geq 0} \frac{1}{n!} x^{\otimes n}$, donde $x^{\otimes n}$ denota la serie obtenida multiplicando n veces x con el producto $\otimes$.*

Proposición 4.2. *Sean $x, y \in A$ series formales de A arbitrarias conmutativas, es decir $x \otimes y = y \otimes x$, tales que tienen término independiente nulo, se verifica entonces que $\exp(x + y) = \exp(x) \otimes \exp(y)$.*

Demostración. En efecto, sean $x, y \in A$ series formales de A arbitrarias conmutativas, podemos observar, aplicando la definición, que se verifica $\exp(x + y) = \sum_{n \geq 0} \frac{1}{n!} (x + y)^{\otimes n} =$ (por la fórmula del binomio de Newton, que podemos utilizar porque sabemos que x e y son conmutativas) $= \sum_{n \geq 0} \sum_{k=0}^{n} \binom{n}{k} \frac{x^{\otimes k} \otimes y^{\otimes (n-k)}}{n!} = \sum_{n \geq 0} \sum_{k=0}^{n} \frac{x^{\otimes k} \otimes y^{\otimes (n-k)}}{k!(n-k)!} =$ (observando que estamos sumando para cualquier pareja $(k, n-k)$ con componentes no

negativas, reindexando la sumación) $\sum_{r,s\geq 0} \frac{x^{\otimes r}\otimes y^{\otimes s}}{r!s!} =$ (expresando la suma anterior como un producto de series formales) $\sum_{r\geq 0} \frac{1}{r!}x^{\otimes r} \otimes \sum_{s\geq 0} \frac{1}{s!}y^{\otimes s} = \exp(x)\otimes\exp(y)$, cosa que completa la prueba. □

Definición 4.3 (Logaritmo de una serie)**.** *Sea $x \in A$ una serie formal cualquiera tal que se verifica que su término independiente, que podemos denotar por λ_0, es positivo, entonces definimos su logaritmo como la serie dada por* $\log(x) = \log(\lambda_0) + \sum_{n\geq 1} \frac{(-1)^{n-1}}{n}(\frac{x}{\lambda_0} - 1)^{\otimes n}$.

Observamos que claramente se sigue de la definición que el logaritmo de un elemento de $\mathbb{R}\langle\langle e_1, \ldots, e_d\rangle\rangle$ será también un elemento de $\mathbb{R}\langle\langle e_1, \ldots, e_d\rangle\rangle$. A partir de aquí, podemos observar que se verifica el resultado esperado.

Proposición 4.4. *Sea $x \in A$ una serie formal cualquiera tal que su término independiente es 1, entonces podemos afirmar que* $\exp(\log(x)) = x$.

Sea ahora $x \in A$ una serie formal cualquiera tal que su término independiente es nulo, entonces se verifica que $\log(\exp(x)) = x$.

Demostración. La demostración de este resultado se basa en el hecho de que las definiciones dadas de la exponencial y el logaritmo de series formales coinciden con los desarrollos en serie de Taylor de las mismas funciones exponencial y logaritmo reales de variable real en los dominios correspondientes.

Estas series de Taylor, en su dominio, son absolutamente convergentes y, por tanto, nos permiten llevar a cabo los mismos desarrollos y manipulaciones que podríamos hacer con las series formales. Esto, junto con el hecho de que para estas funciones reales exponencial y logaritmo, es un resultado elemental, por definición, que son una inversa de la otra, nos permite concluir que lo mismo es cierto para la exponencial y logaritmo formal.

Por falta de espacio en el trabajo y por no ser su objetivo principal el estudio en profundidad de las series formales, no daremos esta demostración con todo su detalle. □

Entonces, haciendo uso de todo lo observado anteriormente, podemos ver que se verifica la siguiente propiedad.

Proposición 4.5. *Sean $x, y \in A$ series formales cualesquiera conmutativas tal que sus términos independientes son 1, entonces se verifica que* $\log(x\otimes y) = \log(x) + \log(y)$.

Demostración. En efecto, sean $x, y \in A$ arbitrarias verificando las propiedades indicadas, entonces podemos escribir $x = 1 + x'$ y también $y = 1 + y'$ donde $x', y' \in A$ tienen término independiente nulo y claramente son conmutativas porque x, y lo son.

Podemos observar también que las series $\log(x)$, $\log(y)$ son conmutativas: $\log(x)\otimes\log(y) = \log(1+x')\otimes\log(1+y') = (\sum_{n\geq 1}\frac{(-1)^{n-1}}{n}(x')^{\otimes n})\otimes(\sum_{m\geq 1}\frac{(-1)^{m-1}}{m}(y')^{\otimes m}) = \sum_{n,m\geq 1}\frac{(-1)^{n-1}(-1)^{m-1}}{nm}(x')^{\otimes n}\otimes(y')^{\otimes m}$ =(utilizando la conmutatividad de las series x' y y' ya comentada) $\sum_{n,m\geq 1}\frac{(-1)^{n-1}(-1)^{m-1}}{nm}(y')^{\otimes m}\otimes(x')^{\otimes n} = (\sum_{m\geq 1}\frac{(-1)^{m-1}}{m}(y')^{\otimes m})\otimes(\sum_{n\geq 1}\frac{(-1)^{n-1}}{n}(x')^{\otimes n}) = \log(y)\otimes\log(x)$.

En consecuencia, podemos observar que $\log(x\otimes y) = \log(\exp(\log(x))\otimes\exp(\log(y))) =$ (como x e y son series conmutativas y ya hemos visto, por tanto, que $\log(x)$ y $\log(y)$ también lo son) $\log(\exp(\log(x)+\log(y))) = \log(x)+\log(y)$. □

A partir de aquí, una vez introducidas las nociones de exponencial y logaritmo para series formales, también a modo de preliminar y con el objetivo de estudiar la signatura logarítmica de un camino, nos centramos en estudiar el concepto de serie de Lie. Para ello, sean $e_1, \dots e_d$ indeterminadas no conmutativas arbitrarias, trabajaremos en el $\mathbb{R}$-álgebra $\mathbb{R}\langle\langle e_1, \dots, e_d\rangle\rangle$ ya habitual, donde definiremos las series de Lie como un cierto tipo de series formales en estas indeterminadas. En efecto, empezamos introduciendo la siguiente notación.

Definición 4.6 (Corchete de Lie)**.** *Sean x e y dos elementos cualesquiera del $\mathbb{R}$-álgebra $\mathbb{R}\langle\langle e_1, \dots, e_d\rangle\rangle$, entonces podemos definir su corchete de Lie como $[x, y] = x \otimes y - y \otimes x$. También, en el caso de tener solo un elemento, denotaremos $[x] = x$.*

Con esta notación podemos dar la definición de serie de Lie en este contexto.

Definición 4.7 (Serie de Lie)**.** *Sean $e_1, \dots, e_d$ indeterminadas no conmutativas y sea $x \in \mathbb{R}\langle\langle e_1, \dots, e_d\rangle\rangle$ un elemento cualquiera, decimos que x es una serie de Lie si se puede expresar como serie a partir de aplicar el corchete de Lie iterativamente, más precisamente, x es una serie de Lie si existen números reales $\lambda_{i_1,\dots,i_k}$ tal que se tiene:*

$$x = \sum_{k=1}^{\infty} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} \lambda_{i_1,\dots,i_k} [e_{i_1}, [e_{i_2}, \dots [e_{i_{k-1}}, e_{i_k}] \dots]]$$

A continuación, proponemos un ejemplo sencillo para clarificar estos conceptos, aunque ilustrativo de la dificultad que entraña probar a mano y a fuerza bruta que una cierta serie es de Lie.

Ejemplo 4.8. Como ejemplo de serie de Lie, sean e_1, e_2 indeterminadas no conmutativas, podemos tomar $x \in \mathbb{R}\langle\langle e_1, e_2\rangle\rangle$ de manera que $x = e_1 + e_2 + \frac{1}{2}e_1e_2 - \frac{1}{2}e_2e_1 + \frac{1}{12}e_1e_1e_2 - \frac{1}{6}e_1e_2e_1 + \frac{1}{12}e_1e_2e_2 + \frac{1}{12}e_2e_1e_1 - \frac{1}{6}e_2e_1e_2 + \frac{1}{12}e_2e_2e_1$ y, entonces, x es una serie de Lie. En efecto, observamos que $x = e_1 + e_2 + \frac{1}{2}e_1e_2 - \frac{1}{2}e_2e_1 + \frac{1}{12}e_1e_1e_2 - \frac{1}{6}e_1e_2e_1 + \frac{1}{12}e_2e_1e_1 + \frac{1}{12}e_2e_2e_1 - \frac{1}{6}e_2e_1e_2 + \frac{1}{12}e_1e_2e_2 = e_1 + e_2 + \frac{1}{2}(e_1e_2 - e_2e_1) + \frac{1}{12}(e_1e_1e_2 - 2e_1e_2e_1 + e_2e_1e_1 + e_2e_2e_1 - 2e_2e_1e_2 + e_1e_2e_2) = e_1 + e_2 + \frac{1}{2}[e_1, e_2] + \frac{1}{12}(e_1 \otimes (e_1e_2 - e_2e_1) - (e_1e_2 - e_2e_1) \otimes e_1 + e_2 \otimes (e_2e_1 - e_1e_2) - (e_2e_1 - e_1e_2) \otimes e_2) = e_1 + e_2 + \frac{1}{2}[e_1, e_2] + \frac{1}{12}(e_1 \otimes [e_1, e_2] - [e_1, e_2] \otimes e_1 + e_2 \otimes [e_2, e_1] - [e_2, e_1] \otimes e_2) = e_1 + e_2 + \frac{1}{2}[e_1, e_2] + \frac{1}{12}([e_1, [e_1, e_2]] + [e_2, [e_2, e_1]])$, cosa que prueba que x es una serie de Lie.

Finalmente, ya tenemos todo lo necesario para poder enunciar el siguiente resultado, central en el estudio de las series de Lie y que nos ofrece una caracterización sobre cuándo una serie es de Lie. Omitiremos su demostración, que se puede encontrar por ejemplo en la referencia [20], ya que es un teorema propio de la teoría de Lie, cuyo estudio detallado no es un objetivo de este trabajo y que llevaría diversas páginas demostrar con detalle.

Teorema 4.9 (Teorema de Friedrichs)**.** *Sean $e_1, \dots, e_d$ indeterminadas no conmutativas cualesquiera y sean también $e'_1, \dots, e'_d$ indeterminadas no conmutativas entre ellas tal que sí que conmutan con las anteriores. Consideramos entonces $x \in \mathbb{R}\langle\langle e_1, \dots, e_d\rangle\rangle$ y podemos afirmar que se verifica que x es una serie de Lie si, y solo si, $x(e_1 + e'_1, \dots, e_d + e'_d) = x(e_1, \dots, e_d) + x(e'_1, \dots, e'_d)$.*

Cabe destacar que la noción de corchete de Lie, serie de Lie y el teorema anterior se pueden generalizar y estudiar en contextos mucho más generales que el álgebra $\mathbb{R}\langle\langle e_1, \dots, e_d\rangle\rangle$, pero no profundizaremos en esta dirección, ya que no forma parte del tema de estudio de este trabajo.

4.2. Logaritmo de la signatura de un camino

Sea $X : [a,b] \longrightarrow \mathbb{R}^d$ un camino cualquiera, entonces ya hemos explicado que la signatura de X, $S(X)_{a,b}$, se puede pensar como un elemento del $\mathbb{R}$-álgebra de las series formales en las indeterminadas no conmutativas $e_1, \ldots, e_n$, $\mathbb{R}\langle\langle e_1, \ldots, e_d\rangle\rangle$, y, por tanto, también se puede considerar que $S(X)_{a,b} \in A$, con lo cual podemos aplicarle el logaritmo formal. En efecto, aplicándole esta operación a la signatura de X, obtenemos $\log S(X)_{a,b} \in A$ y, teniendo presente que por definición $S(X)_{a,b} \in \mathbb{R}\langle\langle e_1, \ldots, e_d\rangle\rangle$, es fácil darse cuenta de que $\log S(X)_{a,b} \in \mathbb{R}\langle\langle e_1, \ldots, e_d\rangle\rangle$. De hecho, para definir el logaritmo de la signatura de un camino, podríamos, por tanto, no haber introducido el álgebra A y continuar trabajando únicamente en $\mathbb{R}\langle\langle e_1, \ldots, e_d\rangle\rangle$, pero veremos que es más conveniente poder trabajar en A. De modo similar a la signatura, $\log S(X)_{a,b}$ también se puede utilizar para estudiar y describir las propiedades del camino.

Desarrollando los primeros términos de la signatura logarítmica de X, podemos observar que tenemos, utilizando la noción del corchete de Lie, $\log S(X)_{a,b} = \sum_{i=1}^{d} S(X)_{a,b}^{i} e_i + \sum_{1 \leq i < j \leq d} \frac{1}{2}(S(X)_{a,b}^{i,j} - S(X)_{a,b}^{j,i})[e_i, e_j] + \ldots$. Esto nos lleva a pensar que la signatura logarítmica de un camino es una serie de Lie y, en efecto, este es un resultado clásico que podemos enunciar y demostrar, usando los preliminares anteriores.

Teorema 4.10. *Sea $X : [a,b] \longrightarrow \mathbb{R}^d$ un camino cualquiera, entonces si pensamos su signatura $S(X)_{a,b}$ como un elemento de $\mathbb{R}\langle\langle e_1, \ldots, e_d\rangle\rangle$, se verifica que $\log S(X)_{a,b}$ es una serie de Lie.*

Demostración. En efecto, continuando con las notaciones introducidas por el enunciado, queremos ver que $\log S(X)_{a,b}$ es una serie de Lie. Para ello, podemos empezar simplificando las notaciones para facilitar la prueba, denotando por $S \coloneqq S(X)_{a,b}$, ya que no trabajaremos con otros caminos o intervalos en la demostración. Por el mismo motivo, sea $(i_1, \ldots, i_k) \in W$ un multi-índice, entonces utilizaremos la notación $S^{i_1,\ldots,i_k} \coloneqq S(X)_{a,b}^{i_1,\ldots,i_k}$. Sean $e'_1, \ldots, e'_d$ indeterminadas cualesquiera no conmutativas entre ellas, pero conmutativas con las anteriores $e_1, \ldots, e_d$, notemos que podemos utilizar todo lo ya estudiado en el apartado anterior de preliminares, incluyendo las notaciones donde se indican explícitamente las indeterminadas.

Dicho esto, podemos observar que se verifica:

$$S(e_1, \ldots, e_d) \otimes S(e'_1, \ldots, e'_d) =$$

$$(\sum_{k=0}^{\infty} \sum_{i_1,\ldots,i_k \in \{1,\ldots,d\}} S^{i_1,\ldots,i_k} e_{i_1} \ldots e_{i_k}) \otimes (\sum_{k=0}^{\infty} \sum_{i_1,\ldots,i_k \in \{1,\ldots,d\}} S^{i_1,\ldots,i_k} e'_{i_1} \ldots e'_{i_k}) =$$

$$\sum_{k=0}^{\infty} \sum_{p=0}^{k} \sum_{i_1,\ldots,i_k \in \{1,\ldots,d\}} S^{i_1,\ldots,i_p} S^{i_{p+1},\ldots,i_k} e_{i_1} \ldots e_{i_p} e'_{i_{p+1}} \ldots e'_{i_k} = \text{(por la identidad}$$

del producto *shuffle* ya estudiada en la sección de propiedades)

$$= \sum_{k=0}^{\infty} \sum_{p=0}^{k} \sum_{i_1,\ldots,i_k \in \{1,\ldots,d\}} \Big(\sum_{K \in (i_1,\ldots,i_p) \sqcup\!\sqcup (i_{p+1},\ldots,i_k)} S^K \Big) e_{i_1} \ldots e_{i_p} e'_{i_{p+1}} \ldots e'_{i_k}$$

También, por otro lado, podemos afirmar que:

$$S(e_1 + e'_1, \ldots, e_d + e'_d) = \sum_{k=0}^{\infty} \sum_{i_1,\ldots,i_k \in \{1,\ldots,d\}} S^{i_1,\ldots,i_k}((e_{i_1} + e'_{i_1}) \otimes \cdots \otimes (e_{i_k} + e'_{i_k}))$$

Desarrollando el producto de las sumas de las indeterminadas y utilizando sus propiedades de conmutatividad y no conmutatividad correspondientes, las cuales nos permiten colocar primero las variables sin prima seguidas de las con prima, pero nos obligan a mantener el orden dentro de cada uno de los tipos de variable, cosa que podemos describir usando permutaciones *shuffle* cuyas inversas son exactamente las que preservan estos órdenes, podemos observar que la expresión anterior es igual a:

$$\sum_{k=0}^{\infty} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} S^{i_1,\dots,i_k} \sum_{p=0}^{k} \sum_{\sigma \in \mathit{Shuffles(p,k-p)}} e_{i_{\sigma^{-1}(1)}} \dots e_{i_{\sigma^{-1}(p)}} e'_{i_{\sigma^{-1}(p+1)}} \dots e'_{i_{\sigma^{-1}(k)}}$$

Reordenando los sumatorios, podemos afirmar que lo anterior coincide con:

$$\sum_{k=0}^{\infty} \sum_{p=0}^{k} \sum_{\sigma \in \mathit{Shuffles(p,k-p)}} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} S^{i_1,\dots,i_k} e_{i_{\sigma^{-1}(1)}} \dots e_{i_{\sigma^{-1}(p)}} e'_{i_{\sigma^{-1}(p+1)}} \dots e'_{i_{\sigma^{-1}(k)}}$$

Dada $\sigma \in \mathit{Shuffles(p,k-p)}$, por ser una permutación, observamos que sumar sobre todos los multi-índices $(i_1, \dots, i_k)$ o sumar sobre todos los multi-índices permutados $(i_{\sigma(1)}, \dots, i_{\sigma(k)})$ nos da el mismo resultado, concluyendo que la expresión anterior es igual a:

$$\sum_{k=0}^{\infty} \sum_{p=0}^{k} \sum_{\sigma \in \mathit{Shuffles(p,k-p)}} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} S^{i_{\sigma(1)},\dots,i_{\sigma(k)}} e_{i_1} \dots e_{i_p} e'_{i_{p+1}} \dots e'_{i_k} =$$

$$\sum_{k=0}^{\infty} \sum_{p=0}^{k} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} \Big(\sum_{\sigma \in \mathit{Shuffles(p,k-p)}} S^{i_{\sigma(1)},\dots,i_{\sigma(k)}} \Big) e_{i_1} \dots e_{i_p} e'_{i_{p+1}} \dots e'_{i_k} =$$

$$\sum_{k=0}^{\infty} \sum_{p=0}^{k} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} \Big(\sum_{K \in (i_1,\dots,i_p) \sqcup\!\sqcup (i_{p+1},\dots,i_k)} S^{K} \Big) e_{i_1} \dots e_{i_p} e'_{i_{p+1}} \dots e'_{i_k}$$

Por tanto, poniendo en común los dos desarrollos anteriores, concluimos que se tiene la igualdad $S(e_1 + e'_1, \dots, e_d + e'_d) = S(e_1, \dots, e_d) \otimes S(e'_1, \dots, e'_d)$. Además, podemos ver fácilmente con ella que $S(e_1, \dots, e_d) \otimes S(e'_1, \dots, e'_d) = S(e_1 + e'_1, \dots, e_d + e'_d) = S(e'_1 + e_1, \dots, e'_d + e_d) = S(e'_1, \dots, e'_d) \otimes S(e_1, \dots, e_d)$, de manera que las series $S(e_1, \dots, e_d)$ y $S(e'_1, \dots, e'_d)$ conmutan, cosa que nos permite aplicar el logaritmo formal a los dos lados de la igualdad $S(e_1 + e'_1, \dots, e_d + e'_d) = S(e_1, \dots, e_d) \otimes S(e'_1, \dots, e'_d)$ y, por propiedades ya probadas, afirmar que $\log S(e_1 + e'_1, \dots, e_d + e'_d) = \log S(e_1, \dots, e_d) + \log S(e'_1, \dots, e'_d)$.

En consecuencia, como hemos visto que $\log S(e_1 + e'_1, \dots, e_d + e'_d) = \log S(e_1, \dots, e_d) + \log S(e'_1, \dots, e'_d)$, aplicando el teorema de Friedrichs, podemos afirmar directamente que $\log S(X)_{a,b}$ es una serie de Lie, que es lo que queríamos ver, cosa que completa la demostración. □

Para acabar esta sección, observamos que el resultado anterior generaliza, al menos en el $\mathbb{R}$-álgebra que es $\mathbb{R}\langle\langle e_1, \dots, e_d \rangle\rangle$, el teorema de Campbell-Baker-Hausdorff. Antes de enunciarlo, si e_i es una de las indeterminadas anteriores, no conmutativas, se verifica que como serie de $\mathbb{R}\langle\langle e_1, \dots, e_d \rangle\rangle$ tiene término independiente nulo, de manera que podemos considerar su exponencial $\exp(e_i) = \sum_{k=0}^{\infty} \frac{1}{k!} e_i^{\otimes k}$, un elemento de $\mathbb{R}\langle\langle e_1, \dots, e_d \rangle\rangle$. Entonces se tiene el siguiente resultado.

Teorema 4.11 (Teorema de Campbell-Baker-Hausdorff)**.** *Sean $e_1, \dots, e_d$ indeterminadas no conmutativas que dan lugar al $\mathbb{R}$-álgebra $\mathbb{R}\langle\langle e_1, \dots, e_d\rangle\rangle$, tomamos $i, j \in \{1, \dots, d\}$ cualesquiera y entonces se verifica que $\log(\exp(e_i) \otimes \exp(e_j))$ es una serie de Lie.*

Demostración. Continuando con las notaciones arbitrarias del enunciado, podemos considerar los caminos $X : [0,1] \longrightarrow \mathbb{R}^d$ dado por $X_t = \{0, \dots, \overset{i)}{t}, \dots, 0\}$ $\forall t \in [0,1]$ y $Y : [1,2] \longrightarrow \mathbb{R}^d$ de manera que $Y_t = \{0, \dots, \overset{j)}{t}, \dots, 0\}$ $\forall t \in [1,2]$.

Entonces, empezando por el camino X, como todas sus componentes salvo la i-ésima son nulas, recordando la definición de la signatura de un camino, es inmediato darse cuenta de que todos los términos de la signatura que involucren índices diferentes de i serán 0, de manera que $S(X)_{0,1} = \sum_{k=0}^{\infty} S(X)_{0,1}^{\overset{k)}{i,\dots,i}} e_i^{\otimes k} = \sum_{k=0}^{\infty} \frac{1}{k!} e_i^{\otimes k}$. Esta última igualdad se da porque $S(X)_{0,1}^{\overset{k)}{i,\dots,i}} = \frac{1}{k!}$, cosa que se puede probar fácilmente por inducción y que de hecho detallaremos más adelante en el trabajo, en la sección que se centra en la interpretación geométrica de la signatura. Análogamente, observamos que $S(Y)_{1,2} = \sum_{k=0}^{\infty} \frac{1}{k!} e_j^{\otimes k}$.

Por tanto, hemos conseguido probar que $S(X)_{0,1} = \exp(e_i)$ y $S(Y)_{1,2} = \exp(e_j)$ y, aplicando el teorema anterior, tenemos que $\log(\exp(e_i) \otimes \exp(e_j)) = \log(S(X)_{0,1} \otimes S(Y)_{1,2}) =$ (por la identidad de Chen) $= \log(S(X * Y)_{0,2})$ es una serie de Lie, que era lo que queríamos mostrar. □

Sin entrar en los detalles y justificaciones, a modo de motivación del teorema anterior, observamos que, como las indeterminadas e_i y e_j para $i, j \in \{1, \dots, d\}$ cualesquiera diferentes no son conmutativas, no es cierto que $\exp(e_i) \otimes \exp(e_j) = \exp(e_i + e_j)$, es decir, no es cierto que $\log(\exp(e_i) \otimes \exp(e_j)) = e_i + e_j$. Todo y eso, podríamos estar interesados en estudiar el elemento $\log(\exp(e_i) \otimes \exp(e_j))$ y el teorema anterior afirma que este se puede expresar como una serie de Lie.

5. Motivación de la signatura

5.1. Interpretación geométrica

A partir de aquí, pasamos a dar una intuición heurística de a qué corresponde la signatura de un camino y por qué la podemos utilizar para estudiarlos y conocer sus propiedades. En efecto, para empezar, sea $X : [a, b] \longrightarrow \mathbb{R}$ un camino unidimensional cualquiera, podemos estudiar cuáles son los primeros elementos de su signatura. Llevando a cabo los cálculos, podemos ver que $S(X)^1_{a,b} = \int_a^b dX^1_t = X_b - X_a$ i $S(X)^{1,1}_{a,b} = \int_a^b (\int_a^{t_2} dX^1_{t_1}) dX^1_{t_2} = \frac{(X_b - X_a)^2}{2!}$. De hecho, en general, se puede probar el siguiente resultado.

Proposición 5.1. *Sea $X : [a, b] \longrightarrow \mathbb{R}$ un camino unidimensional arbitrario, entonces se verifica que $\forall n \geq 1 \in \mathbb{N}, \quad S(X)^{n)}_{a,b}{}^{1,\dots,1} = \frac{(X_b - X_a)^n}{n!}$.*

Demostración. Para demostrar esta afirmación, continuando con las notaciones introducidas en el enunciado, procederemos por el método de inducción, observando que $S(X)^{n)}_{a,b}{}^{1,\dots,1} = \frac{(X_b - X_a)^n}{n!} \quad \forall n \geq 1$.

Caso inicial ($n = 1$): En efecto, es inmediato ver, tal y como lo hemos hecho previamente, que $S(X)^1_{a,b} = \int_a^b dX^1_t = \int_a^b \dot{X}^1_t dt = X_b - X_a = \frac{(X_b - X_a)^1}{1!}$, cosa que deja resuelto este caso.

Caso sucesor: Ahora, como hipótesis de inducción, suponemos que el resultado es cierto para $n \geq 1 \in \mathbb{N}$ cualquiera y pasamos a probar que también se cumple para $n + 1$. Por definición, tenemos que $S(X)^{n+1)}_{a,b}{}^{1,\dots,1} = \int_{a<t_1\dots<t_{n+1}<b} \quad dX^1_{t_1} dX^1_{t_2} \dots dX^1_{t_{n+1}} = \int_{a<t<b} S(X)^{n)}_{a,t}{}^{1,\dots,1} dX^1_t =$ (aplicando la hipótesis de inducción al camino en el intervalo $[a, t]$) $= \int_{a<t<b} \frac{(X_t - X_a)^n}{n!} \dot{X}_t dt =$ (por la regla de Barrow) $= (\frac{(X_t - X_a)^{n+1}}{(n+1)!}|^{t=b}_{t=a} = \frac{(X_b - X_a)^{n+1}}{(n+1)!}$. Esto es lo que queríamos ver, cosa que completa el caso sucesor.

Por el principio de inducción matemática, se deduce el resultado. □

Notemos que estas expresiones coinciden con el volumen n-dimensional de un símplice de lado $X_b - X_a$.

Una vez hecha esta observación para la signatura de los caminos unidimensionales, pasamos también a intentar dar una idea de la interpretación geométrica de los primeros niveles de la signatura para caminos en $\mathbb{R}^2$. Para ello, empezamos centrándonos en los caminos lineales y, como sabemos que la signatura de un camino es invariante para translaciones y reparametrizaciones de clase C^1, consideraremos caminos definidos en el intervalo $[0, 1]$ y con punto inicial el origen. Empezamos, por tanto, afirmando el siguiente resultado:

Proposición 5.2. *Sea $d \geq 1 \in \mathbb{N}$, sea $X : [0, 1] \longrightarrow \mathbb{R}^d$ un camino tal que $X_t = \{X^1_t, \dots, X^d_t\} = \{A_1 t, \dots, A_d t\} \quad \forall t \in [0, 1]$ para ciertas constantes fijadas $A_1, \dots, A_d \in \mathbb{R}$ y sea $I = (i_1, \dots, i_k) \in W$ un multi-índice cualquiera, entonces se verifica que $S(X)^I_{0,1} = \frac{\prod_{j=1}^k A_{i_j}}{k!}$.*

Demostración. En efecto, continuando con las notaciones del enunciado, antes de nada, podemos definir el camino unidimensional $Y : [0, 1] \longrightarrow \mathbb{R}$ dado por $Y_t = t \quad \forall t \in [0, 1]$. En-

tonces, podemos ver que por definición $S(X)^I_{0,1} = \int_{0<t_1\ldots<t_k<1} \quad dX^{i_1}_{t_1} dX^{i_2}_{t_2} \ldots dX^{i_k}_{t_k} =$ (por definición de la integral de Riemann-Stieltjes y la linealidad de la integral de Riemann y del operador derivada, extrayendo las constantes) $= (\prod_{j=1}^k A_{i_j}) \int_{0<t_1\ldots<t_k<1} \quad dt_1 dt_2 \ldots dt_k$ = (aplicando la definición del camino Y) $= (\prod_{j=1}^k A_{i_j}) \int_{0<t_1\ldots<t_k<1} \quad dY_{t_1} dY_{t_2} \ldots dY_{t_k} =$ $(\prod_{j=1}^k A_{i_j}) \int_{0<t_1\ldots<t_k<1} \quad dY^1_{t_1} dY^1_{t_2} \ldots dY^1_{t_k} =$ (por definición de la signatura de un camino) $= (\prod_{j=1}^k A_{i_j}) S(Y)^{1,\ldots,1}_{0,1} =$ (por el resultado ya probado para caminos unidimensionales) $= (\prod_{j=1}^k A_{i_j}) \frac{(1-0)^k}{k!} = \frac{\prod_{j=1}^k A_{i_j}}{k!}$, que es exactamente lo que queríamos demostrar. $\square$

En particular, podemos considerar un camino lineal en $\mathbb{R}^2$, es decir, $X : [0,1] \longrightarrow \mathbb{R}^2$ tal que $X_t = \{X^1_t, X^2_t\} = \{A_1 t, A_2 t\} \quad \forall t \in [0,1]$ para ciertas constantes fijadas $A_1, A_2 \in \mathbb{R}$ y el resultado anterior nos permite afirmar que $S(X)^{1,2}_{0,1} = S(X)^{2,1}_{0,1} = \frac{A_1 A_2}{2}$. Nuestro objetivo en lo que resta de la sección será intentar dar una interpretación geométrica del segundo nivel de la signatura de caminos lineales a trozos y, como podríamos proceder por inducción, será suficiente estudiar el caso de un camino a trozos dado por la concatenación de dos caminos lineales.

Sean, por tanto, $Y : [0,1] \longrightarrow \mathbb{R}^2$ dado de manera que $Y_t = \{A_1 t, A_2 t\} \quad \forall t \in [0,1]$ y $Z : [1,2] \longrightarrow \mathbb{R}^2$ tal que $Z_t = \{B_1 t, B_2 t\} \quad \forall t \in [1,2]$, entonces podemos considerar camino $X : [0,2] \longrightarrow \mathbb{R}^2$ definido a partir de la concatenación de los anteriores $X = Y * Z$.

En este caso, para estudiar los términos del segundo nivel de la signatura de X, observamos primero que los asociados a $(1,1)$ y $(2,2)$ se pueden pensar, por definición, como los elementos del segundo nivel de la signatura de los caminos unidimensionales X^1 y X^2 respectivamente. En consecuencia, tenemos que $S(X)^{1,1}_{0,2}$ se corresponde con el área del símplice 2-dimensional de lado $X^1_2 - X^1_0 = Y^1_1 + (Z^1_2 - Z^1_1) = A_1 + 2B_1 - B_1 = A_1 + B_1$. Análogamente, observamos que $S(X)^{2,2}_{0,2}$ es el área del símplice 2-dimensional de lado $A_2 + B_2$.

Por tanto, nos podemos centrar en estudiar los términos de la signatura dados por los multi-índices $(1,2)$ y $(2,1)$, haciendo uso primero de la expresión de la signatura como un elemento del álgebra $\mathbb{R}\langle\langle e_1, \ldots, e_d \rangle\rangle$ y de la identidad de Chen ya probada previamente. En efecto, podemos observar, antes de nada, que $S(X)_{0,2} =$ (por definición del camino X) $= S(Y * Z)_{0,2} =$ (por la identidad de Chen) $= S(Y)_{0,1} \otimes S(Z)_{1,2}$. Teniendo esto presente, sabemos que el elemento $S(X)^{1,2}_{0,2}$ es el coeficiente acompañado por $e_1 e_2$ en la serie formal del álgebra que es $S(X)_{0,2}$ y, por tanto, como las únicas maneras de obtener $e_1 e_2$ mediante el producto $\otimes$ son $1 \otimes e_1 e_2$, $e_1 \otimes e_2$ y $e_1 e_2 \otimes 1$, podemos afirmar que $S(X)^{1,2}_{0,2} = S(Y)^{1,2}_{0,1} + S(Y)^1_{0,1} S(Z)^2_{1,2} + S(Z)^{1,2}_{1,2} =$ (por ser Y y Z caminos lineales y ya haber hecho un estudio de los valores de su signatura) $= \frac{A_1 A_2}{2} + A_1 B_2 + \frac{B_1 B_2}{2}$, que es lo que queríamos ver.

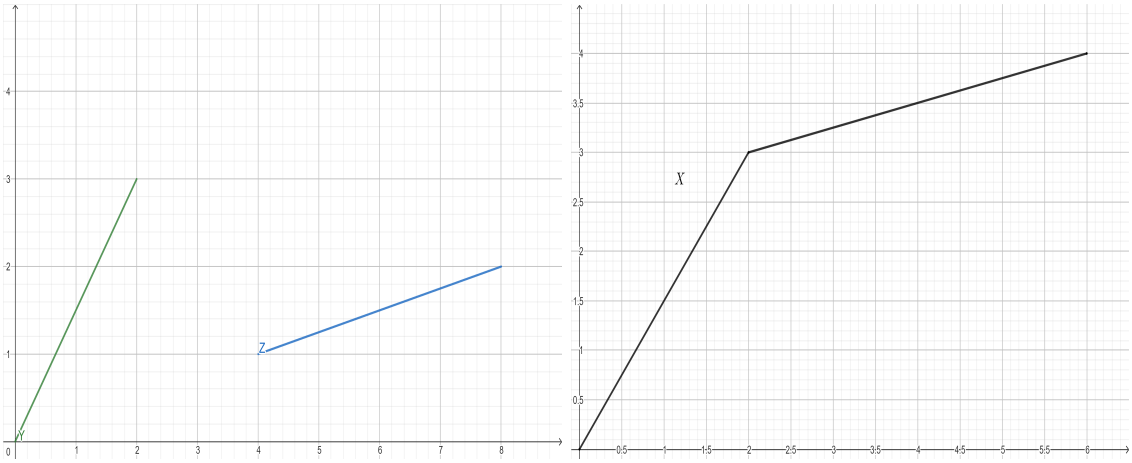
De manera completamente análoga, podemos ver también que $S(X)^{2,1}_{0,2} = \frac{A_1 A_2}{2} + A_2 B_1 + \frac{B_1 B_2}{2}$ y, en resumen, hemos deducido que:

$$S(X)^{1,2}_{0,2} = \frac{A_1 A_2}{2} + A_1 B_2 + \frac{B_1 B_2}{2}$$
$$S(X)^{2,1}_{0,2} = \frac{A_1 A_2}{2} + A_2 B_1 + \frac{B_1 B_2}{2}$$

Esto completa el cálculo de los elementos del segundo nivel de la signatura. Todo y eso, como nuestro objetivo era interpretar geométricamente el segundo nivel de la signatura, daremos a continuación dos ejemplos que ayudarán a dar sentido a los valores

obtenidos anteriormente en relación con las trazas de los caminos involucrados X, Y y Z, especialmente a los dados por los multi-índices $(1,2)$ y $(2,1)$.

Ejemplo 5.3. Para empezar, utilizando las mismas notaciones que en el desarrollo anterior, consideremos las constantes $A_1 = 2$, $A_2 = 3$, $B_1 = 4$, $B_2 = 1$ y los caminos correspondientes $Y : [0,1] \longrightarrow \mathbb{R}^2$ dado de manera que $Y_t = \{2t, 3t\} \quad \forall t \in [0,1]$ y $Z : [1,2] \longrightarrow \mathbb{R}^2$ tal que $Z_t = \{4t, t\} \quad \forall t \in [1,2]$. Con ellos, considerando su concatenación, podemos obtener el camino $X : [0,2] \longrightarrow \mathbb{R}^2$ definido a partir de los anteriores $X = Y * Z$. Para tener intuición gráfica sobre estos caminos y entenderlos mejor, podemos representar gráficamente sus trazas, mostradas en la figura 3.



(a) Trazas de Y en verde y Z en azul (b) Traza del camino $Y * Z$

Figura 3: Trazas de los caminos X, Y, Z

A partir de aquí, podemos calcular los términos del segundo nivel de la signatura de X, utilizando las deducciones hechas previamente, concluyendo que:

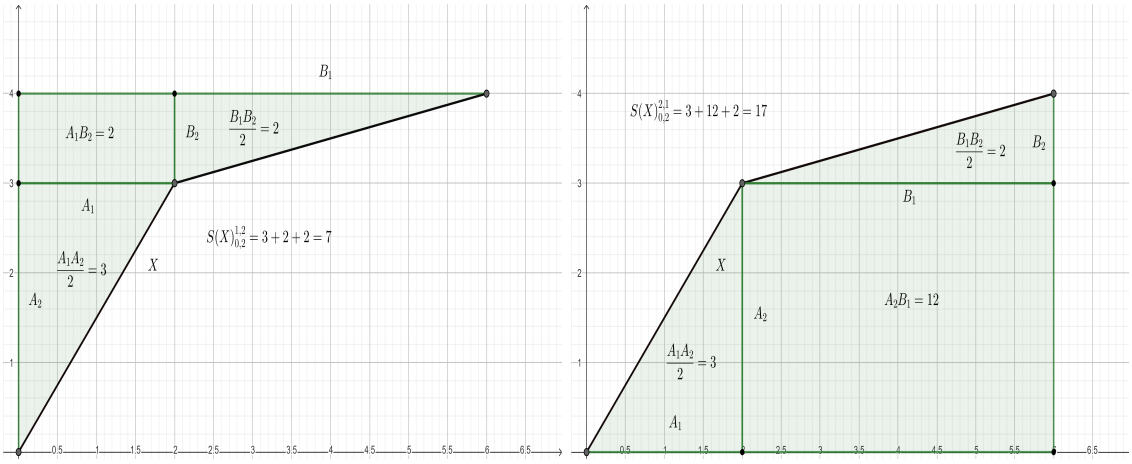
$$S(X)^{1,1}_{0,2} = \frac{(A_1 + B_1)^2}{2} = \frac{(2+4)^2}{2} = 18 \quad S(X)^{1,1}_{0,2} = \frac{(A_2 + B_2)^2}{2} = \frac{(3+1)^2}{2} = 8$$

$$S(X)^{1,2}_{0,2} = \frac{A_1 A_2}{2} + A_1 B_2 + \frac{B_1 B_2}{2} = \frac{2 \cdot 3}{2} + 2 \cdot 1 + \frac{4 \cdot 1}{2} = 7$$

$$S(X)^{2,1}_{0,2} = \frac{A_1 A_2}{2} + A_2 B_1 + \frac{B_1 B_2}{2} = \frac{2 \cdot 3}{2} + 3 \cdot 4 + \frac{4 \cdot 1}{2} = 17$$

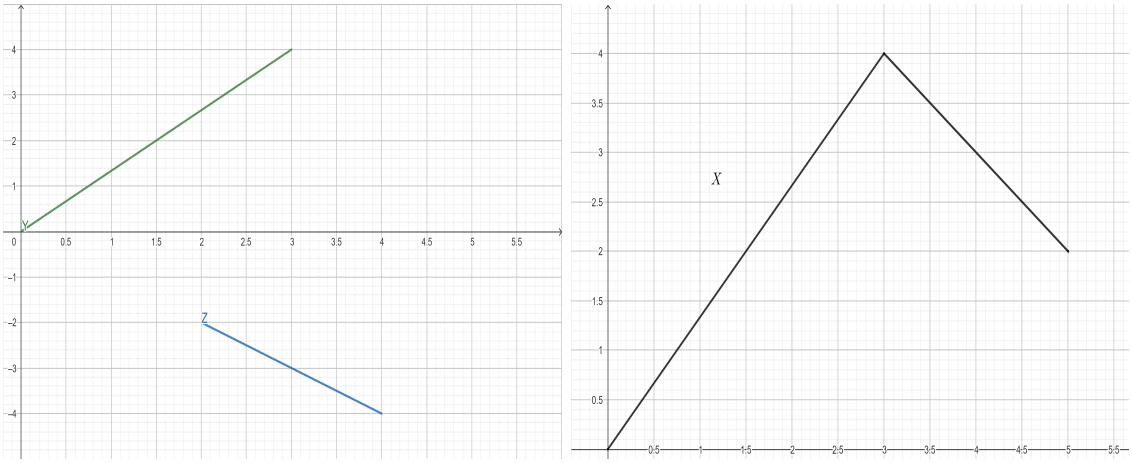
Ahora, podemos intentar relacionar los valores dados por los multi-índices $(1,2)$ y $(2,1)$ con la traza del camino X para obtener su interpretación geométrica, tal como se muestra en la figura 4.

Ejemplo 5.4. De nuevo, proponemos otro ejemplo para ayudar a clarificar la interpretación geométrica de la signatura de un camino. En este caso, consideramos las constantes, $A_1 = 3$, $A_2 = 4$, $B_1 = 2$, $B_2 = -2$ y los caminos correspondientes $Y : [0,1] \longrightarrow \mathbb{R}^2$ dado de manera que $Y_t = \{3t, 4t\} \quad \forall t \in [0,1]$ y $Z : [1,2] \longrightarrow \mathbb{R}^2$ tal que $Z_t = \{2t, -2t\} \quad \forall t \in [1,2]$. A partir de aquí, tomando como en el ejemplo anterior su concatenación, podemos obtener el camino $X : [0,2] \longrightarrow \mathbb{R}^2$ definido de manera que $X = Y * Z$. Siguiendo el mismo esquema que anteriormente, para dar intuición gráfica de estos caminos, empezamos representando gráficamente sus trazas, mostradas en la figura 5.



(a) Interpretación de $S(X)^{1,2}_{0,2}$ (b) Interpretación de $S(X)^{2,1}_{0,2}$

Figura 4: Interpretación geométrica de los términos $S(X)^{1,2}_{0,2}$ y $S(X)^{2,1}_{0,2}$



(a) Trazas de Y en verde y Z en azul (b) Traza del camino $Y * Z$

Figura 5: Trazas de los caminos X, Y, Z

De nuevo, utilizando los resultados anteriores, pasamos a calcular los elementos del segundo nivel de la signatura de X, concluyendo que:

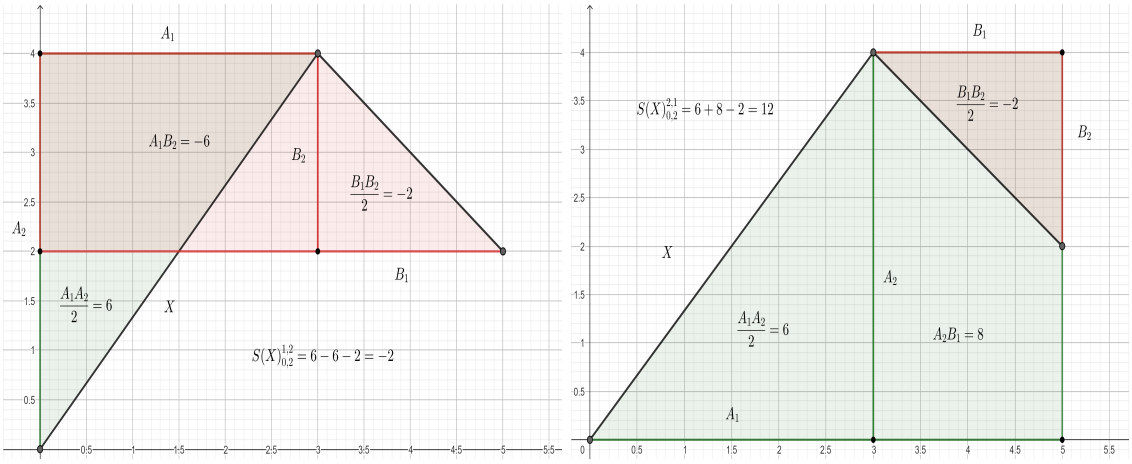
$$S(X)^{1,1}_{0,2} = \frac{(A_1+B_1)^2}{2} = \frac{(3+2)^2}{2} = 12{,}5 \quad S(X)^{1,1}_{0,2} = \frac{(A_2+B_2)^2}{2} = \frac{(4-2)^2}{2} = 2$$

$$S(X)^{1,2}_{0,2} = \frac{A_1A_2}{2} + A_1B_2 + \frac{B_1B_2}{2} = \frac{3\cdot 4}{2} + 3\cdot(-2) + \frac{2\cdot(-2)}{2} = -2$$

$$S(X)^{2,1}_{0,2} = \frac{A_1A_2}{2} + A_2B_1 + \frac{B_1B_2}{2} = \frac{3\cdot 4}{2} + 4\cdot 2 + \frac{2\cdot(-2)}{2} = 12$$

Una vez calculados estos valores, centrándonos en los dados por los multi-índices $(1,2)$ y $(2,1)$, podemos representarlos gráficamente junto con la traza del camino X para intentar darles una interpretación geométrica, tal como se muestra en la figura 6.

En conjunto, podemos observar que, en el primer ejemplo, donde el camino era creciente, sí que es fácil asociar estos términos de la signatura dados por los multi-índices



(a) Interpretación de $S(X)^{1,2}_{0,2}$ (b) Interpretación de $S(X)^{2,1}_{0,2}$

Figura 6: Interpretación geométrica de los términos $S(X)^{1,2}_{0,2}$ y $S(X)^{2,1}_{0,2}$

$(1,2)$ y $(2,1)$ con áreas sobre y bajo la traza del camino. En cambio, en el segundo, pasa a ser más difícil dar una idea intuitiva y geométrica de que representan estos términos, aunque se observa que están relacionados con una suma de las áreas con signo sobre y bajo de la traza respectivamente. Esto nos lleva a pensar, heurísticamente, que en general, especialmente para sus niveles superiores, no se puede dar una interpretación geométrica de la signatura de manera relativamente sencilla, aunque pueda estar relacionada con ciertos volúmenes y pueda, por tanto, ayudar a estudiar el camino.

5.2. Relación de la signatura con la solución de EDOs controladas

Una vez analizada la interpretación geométrica de los elementos de la signatura de un camino, pasamos ahora a dar también otra justificación de por qué la signatura es capaz de describir propiedades de un camino, esta vez desde el campo de las ecuaciones diferenciales controladas. Como nuestro objetivo es solo motivar la signatura, por comodidad, en esta sección supondremos que todos los caminos son al menos diferenciables con derivada continua.

En efecto, antes de nada, recordamos la definición de la solución de ciertas ecuaciones diferenciales controladas, en las que nos centraremos, dadas en un problema de Cauchy y teniendo presente que para $e, d \geq 1 \in \mathbb{N}$, $\mathbb{L}(\mathbb{R}^d, \mathbb{R}^e)$ denota el espacio de las aplicaciones lineales de $\mathbb{R}^d$ en $\mathbb{R}^e$. Notemos que podemos identificar este espacio, haciendo un abuso de notación habitual, con el de las matrices de e filas y d columnas, tomando las bases canónicas de $\mathbb{R}^d$ y $\mathbb{R}^e$ respectivamente.

Definición 5.5 (Solución de una ecuación diferencial controlada)**.** *Sean $X : [a,b] \longrightarrow \mathbb{R}^d$ y $Y : [a,b] \longrightarrow \mathbb{R}^e$ caminos arbitrarios, sea también $V : \mathbb{R}^e \longrightarrow \mathbb{L}(\mathbb{R}^d, \mathbb{R}^e)$ una función continua y sea $y \in \mathbb{R}^e$, decimos que Y es solución de la ecuación diferencial controlada $dY_t = V(Y_t)dX_t, \quad Y_a = y$ si se verifica que $\forall t \in [a,b], \quad Y_t = y + \int_a^t V(Y_s)dX_s$.*

A partir de aquí, podemos observar de manera trivial que, por tanto, en particular, estas ecuaciones diferenciales controladas son EDOs en el sentido más elemental. En efecto, pasamos a precisar esta última afirmación.

Proposición 5.6. *Sean $X : [a,b] \longrightarrow \mathbb{R}^d$ y $Y : [a,b] \longrightarrow \mathbb{R}^e$ caminos arbitrarios y sea también $V : \mathbb{R}^e \longrightarrow \mathbb{L}(\mathbb{R}^d, \mathbb{R}^e)$ una función continua, entonces si consideramos $f : [a,b] \times \mathbb{R}^e \longrightarrow \mathbb{R}^e$ la función dada por $f(t,x) = V(x)\dot{X}_t \quad \forall (t,x) \in [a,b] \times \mathbb{R}^e$, tenemos que Y es solución de la ecuación diferencial controlada $dY_t = V(Y_t)dX_t, \quad Y_a = y$ si, y solo si, Y es solución del problema de Cauchy $\dot{x} = f(t,x), \quad x(a) = y$, es decir, $\dot{Y}_t = f(t, Y_t) \quad \forall t \in [a,b]$ y también $Y_a = y$.*

Demostración. Para probar el resultado, demostraremos las dos implicaciones de la equivalencia.

$\Rightarrow$ Suponemos primero que Y es solución de la ecuación diferencial controlada y, por tanto, por definición, sea $t \in [a,b]$ cualquiera, tenemos que $Y_t = y + \int_a^t V(Y_s)dX_s = y + \int_a^y V(Y_s)\dot{X}_s ds$ y, en particular, podemos entonces ver que $Y_a = y$. Además, por el Teorema Fundamental del Cálculo, como X y Y son caminos y, por tanto, funciones continuas y V también lo es por hipótesis, concluimos que $\dot{Y}_t = V(Y_t)\dot{X}_t = f(t, Y_t)$, cosa que completa la implicación.

$\Leftarrow$ Recíprocamente, suponemos que Y es solución del problema de Cauchy indicado y, entonces, podemos observar que, sea $t \in [a,b]$, $Y_t - y = Y_t - Y_a =$ (por la regla de Barrow) $= \int_a^t \dot{Y}_s ds = \int_a^t f(s, Y_s)ds = \int_a^t V(Y_s)\dot{X}_s ds = \int_a^t V(Y_s)dX_s$. Esto completa la prueba. $\square$

Para relacionar la signatura de un camino con estas ecuaciones, nos centraremos en el caso en que $V : \mathbb{R}^e \longrightarrow \mathbb{L}(\mathbb{R}^d, \mathbb{R}^e)$ es una función lineal y, entonces, para empezar, tenemos el siguiente resultado.

Proposición 5.7. *Sean $X : [a,b] \longrightarrow \mathbb{R}^d$ un camino y $V : \mathbb{R}^e \longrightarrow \mathbb{L}(\mathbb{R}^d, \mathbb{R}^e)$ una función lineal cualesquiera, entonces si consideramos $f : [a,b] \times \mathbb{R}^e \longrightarrow \mathbb{R}^e$ la función dada por $f(t,y) = V(y)\dot{X}_t \quad \forall (t,y) \in [a,b] \times \mathbb{R}^e$, tenemos que existe $A : [a,b] \longrightarrow \mathbb{L}(\mathbb{R}^e, \mathbb{R}^e)$ una función continua tal que $f(t,y) = A(t)y \quad \forall (t,y) \in [a,b] \times \mathbb{R}^e$.*

Demostración. En efecto, continuando con las notaciones del enunciado, antes de nada, como V es lineal, tenemos que:

$$V(y) = \begin{pmatrix} \sum_{k=1}^e \lambda_{1,1}^k y_k & \cdots & \sum_{k=1}^e \lambda_{1,d}^k y_k \\ \vdots & \ddots & \vdots \\ \sum_{k=1}^e \lambda_{e,1}^k y_k & \cdots & \sum_{k=1}^e \lambda_{e,d}^k y_k \end{pmatrix} \quad \forall y = (y_1, \ldots, y_e) \in \mathbb{R}^e$$

para ciertas constantes $\lambda_{i,j}^k \in \mathbb{R} \quad \forall i \in \{1, \ldots, e\}$, $j \in \{1, \ldots, d\}$ y $k \in \{1, \ldots, e\}$.

Entonces, teniendo esta expresión presente, sea $(t,y) \in [a,b] \times \mathbb{R}^e$ arbitrario de manera que $y = (y_1, \ldots, y_e)$, podemos observar directamente que $f(t,y) = V(y)\dot{X}_t =$ (representando matricialmente la igualdad anterior) $=$

$$= \begin{pmatrix} \sum_{k=1}^e \lambda_{1,1}^k y_k & \cdots & \sum_{k=1}^e \lambda_{1,d}^k y_k \\ \vdots & \ddots & \vdots \\ \sum_{k=1}^e \lambda_{e,1}^k y_k & \cdots & \sum_{k=1}^e \lambda_{e,d}^k y_k \end{pmatrix} \begin{pmatrix} \dot{X}_t^1 \\ \vdots \\ \dot{X}_t^d \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^d \sum_{k=1}^e \lambda_{1,j}^k y_k \dot{X}_t^j \\ \vdots \\ \sum_{j=1}^d \sum_{k=1}^e \lambda_{e,j}^k y_k \dot{X}_t^j \end{pmatrix} =$$

$$= \begin{pmatrix} \sum_{k=1}^e \sum_{j=1}^d \lambda_{1,j}^k \dot{X}_t^j y_k \\ \vdots \\ \sum_{k=1}^e \sum_{j=1}^d \lambda_{e,j}^k \dot{X}_t^j y_k \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^d \lambda_{1,j}^1 \dot{X}_t^j & \cdots & \sum_{j=1}^d \lambda_{1,j}^e \dot{X}_t^j \\ \vdots & \ddots & \vdots \\ \sum_{j=1}^d \lambda_{e,j}^1 \dot{X}_t^j & \cdots & \sum_{j=1}^d \lambda_{e,j}^e \dot{X}_t^j \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_e \end{pmatrix}$$

Por tanto, tomando $A : [a,b] \longrightarrow \mathbb{L}(\mathbb{R}^e, \mathbb{R}^e)$ dada por:

$$A(t) = \begin{pmatrix} \sum_{j=1}^{d} \lambda_{1,j}^{1} \dot{X}_t^j & \cdots & \sum_{j=1}^{d} \lambda_{1,j}^{e} \dot{X}_t^j \\ \vdots & \ddots & \vdots \\ \sum_{j=1}^{d} \lambda_{e,j}^{1} \dot{X}_t^j & \cdots & \sum_{j=1}^{d} \lambda_{e,j}^{e} \dot{X}_t^j \end{pmatrix} \quad \forall t \in [a,b]$$

que claramente es continua porque estamos consideramos caminos con derivada continua, podemos ver que verifica que $f(t,y) = A(t)y \quad \forall (t,y) \in [a,b] \times \mathbb{R}^e$, cosa que completa la prueba. □

Por tanto, combinando los dos resultados anteriores, concluimos que la ecuación diferencial controlada dada por V se puede ver como una EDO lineal y, en consecuencia, podemos aplicar una cierta versión del teorema de Picard para garantizar que la solución $Y : [a,b] \longrightarrow \mathbb{R}^e$ de la ecuación existe y además se puede obtener como límite de la sucesión de los iterados de Picard correspondiente. Antes de explicitarla, recordamos el enunciado de estos resultados, cuyas demostraciones no daremos por falta de espacio y porque no son resultados propios de la teoría de la signatura. Se pueden encontrar en la referencia [22].

Teorema 5.8 (Teorema de Picard)**.** *Sea $I = [a,b] \subseteq \mathbb{R}$ un intervalo cerrado. Sean $\Omega = I \times \mathbb{R}^n$ y también $f : \Omega \longrightarrow \mathbb{R}^n$ una función continua y Lipschitz respecto de la segunda variable, es decir, $\exists K > 0 \in \mathbb{R}$ tal que $\forall (t,y_1), (t,y_2) \in \Omega$, $\|f(t,y_2) - f(t,y_1)\| \leq K\|y_2 - y_1\|$ (donde $\|\cdot\|$ denota aquí la norma usual de $\mathbb{R}^n$). Entonces, sea $(t_0,y_0) \in \Omega$, tenemos que el problema de Cauchy que determinan f y (t_0,y_0) admite una única solución definida en I y que se puede expresar como límite de los iterados de Picard.*

Corolario 5.9. *Sea $I = [a,b] \subseteq \mathbb{R}$ un intervalo cerrado y $\Omega = I \times \mathbb{R}^n$. Sea también $A : I \longrightarrow \mathbb{L}(\mathbb{R}^n, \mathbb{R}^n)$ continua, tomamos entonces $f : \Omega \longrightarrow \mathbb{R}^n$ dada por $f(t,y) = A(t)y \quad \forall (t,y) \in \Omega$ y se verifica que para $(t_0,x_0) \in \Omega$, el problema de Cauchy definido por f y (t_0,x_0) tiene solución única definida en I obtenida como límite de los iterados de Picard.*

Pasamos ahora a precisar cuál es la sucesión de los iterados de Picard en este contexto y cómo nos permite observar intuitivamente que la signatura de un camino se puede utilizar para describirlo.

Sean $X : [a,b] \longrightarrow \mathbb{R}^d$ un camino y $V : \mathbb{R}^e \longrightarrow \mathbb{L}(\mathbb{R}^d, \mathbb{R}^e)$ una función lineal cualesquiera. Sea también $y \in \mathbb{R}^e$ arbitrario, entonces podemos construir el camino solución de la ecuación diferencial controlada correspondiente $Y : [a,b] \longrightarrow \mathbb{R}^e$ como límite de la siguiente sucesión: tomamos $Y_t^0 = y \, \forall t \in [a,b]$ y, recursivamente, si suponemos que ya hemos definido Y^{n-1} para $n \geq 1$, tenemos que $Y_t^n = y + \int_a^t V(Y_s^{n-1}) dX_s \quad \forall t \in [a,b]$.

Finalmente, ya estamos preparados para enunciar y demostrar el resultado central de esta sección, que relaciona la signatura de un camino con todo lo anterior.

Teorema 5.10. *Sean $X : [a,b] \longrightarrow \mathbb{R}^d$ un camino y $V : \mathbb{R}^e \longrightarrow \mathbb{L}(\mathbb{R}^d, \mathbb{R}^e)$ una función lineal cualesquiera. Sea también $y \in \mathbb{R}^e$ arbitrario, entonces si $Y : [a,b] \longrightarrow \mathbb{R}^e$ es solución de la ecuación diferencial controlada $dY_s = V(Y_s)dX_s, \quad Y_a = y, \, \forall t \in [a,b]$ el valor Y_t^n se puede expresar como una combinación lineal de los términos hasta el nivel n-ésimo de la signatura $S(X)_{a,t}$ con coeficientes independientes de t. En particular, Y_t está completamente determinada por V, $S(X)_{a,t}$ e y.*

Demostración. Para probar este resultado, utilizando las notaciones anteriores, procedemos por inducción en $n \in \mathbb{N}$.

Caso inicial ($n = 0$): para empezar, si $n = 0$, entonces tenemos que sea $t \in [a, b]$ cualquiera, por definición $Y_t^0 = y = y \cdot 1$, que es lo que queríamos ver, ya que recordamos que el nivel 0-ésimo de la signatura está formado por el elemento 1.

Caso sucesor: ahora, suponemos que el resultado es cierto para $n - 1 \geq 0$ y pasamos a probarlo para n. En efecto, sea $t \in [a, b]$ arbitrario, podemos ver que por hipótesis de inducción $\forall s \in [a, b]$ tenemos que Y_s^{n-1} se puede expresar de manera que $Y_s^{n-1} = \sum_{k=0}^{n-1} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} \lambda_{i_1,\dots,i_k} S(X)_{a,s}^{i_1,\dots,i_k}$ para algunos $\lambda_{i_1,\dots,i_k} \in \mathbb{R}^e$. Entonces, directamente, concluimos que:

$$Y_t^n = y + \int_a^t V(Y_s^{n-1}) dX_s = y + \int_a^t V(Y_s^{n-1}) \dot{X}_s ds = \text{(por la linealidad de } V) =$$

$$= y + \sum_{k=0}^{n-1} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} \int_a^t V(\lambda_{i_1,\dots,i_k}) S(X)_{a,s}^{i_1,\dots,i_k} \dot{X}_s ds =$$

$$= y + \sum_{k=0}^{n-1} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} V(\lambda_{i_1,\dots,i_k}) \int_a^t S(X)_{a,s}^{i_1,\dots,i_k} dX_s =$$

$$y + \sum_{k=0}^{n-1} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} V(\lambda_{i_1,\dots,i_k}) \begin{pmatrix} \int_a^t S(X)_{a,s}^{i_1,\dots,i_k} dX_s^1 \\ \vdots \\ \int_a^t S(X)_{a,s}^{i_1,\dots,i_k} dX_s^d \end{pmatrix} =$$

$$= y + \sum_{k=0}^{n-1} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} V(\lambda_{i_1,\dots,i_k}) \begin{pmatrix} S(X)_{a,t}^{i_1,\dots,i_k,1} \\ \vdots \\ S(X)_{a,t}^{i_1,\dots,i_k,d} \end{pmatrix} =$$

(por ser $V(\lambda_{i_1,\dots,i_k}) \in \mathbb{L}(\mathbb{R}^d, \mathbb{R}^e)$) $= \sum_{k=0}^{n} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} \mu_{i_1,\dots,i_k} S(X)_{a,t}^{i_1,\dots,i_k}$ para algunos $\mu_{i_1,\dots,i_k} \in \mathbb{R}^e$ independientes de t y $\mu_0 = y$. Esto resuelve el caso sucesor.

En conjunto, por el principio de inducción matemática, queda probado el teorema. □

El hecho de que esto sea cierto independientemente de la elección de la función V nos indica intuitivamente que la signatura es capaz de dar una descripción muy precisa del camino y sus propiedades, no siendo necesario conocer el camino para obtener la solución de la ecuación en un cierto punto. De hecho, podríamos probar también un resultado muy relevante sobre la información que la signatura de un camino nos aporta, afirmando por ejemplo que para un cierto tipo de caminos la signatura caracteriza al camino [3]. Este resultado y su demostración queda fuera del alcance de este trabajo.

Finalmente, notemos que es habitual expresar la signatura de un camino y especialmente lo que hemos estudiado en esta sección en el lenguaje de los tensores y del álgebra tensorial, pero hemos decidido no hacerlo porque no ha sido necesario y porque no profundizaremos más en esta dirección.

6. Método de la signatura

Una vez estudiada la signatura de un camino desde un punto de vista formal, pasamos a justificar su interés en el campo de las matemáticas aplicadas, mostrando como se puede hacer uso de ella para construir los atributos en modelos de *machine learning*. En efecto, dado un conjunto de $n \geq 2$ valores, finito, que podemos denotar como $\{X_i\}_{i=1}^n$, para poder utilizar la signatura y extraer información de los datos, primero tendremos construir un camino con ellos del que podamos calcular su signatura, conociéndose esta transformación como un *embedding* de los datos originales. Hay diferentes maneras de proceder y según el *embedding* que escojamos, la signatura describirá mejor unas ciertas características de la información $\{X_i\}_{i=1}^n$. Por tanto, centraremos esta sección del trabajo en el estudio de los métodos más habituales para construir caminos a partir de una secuencia de datos como la indicada, $\{X_i\}_{i=1}^n$, donde supondremos en toda la sección que $n \geq 2$.

6.1. Construcción de caminos por interpolación lineal

Por un lado, dada $\{X_i\}_{i=1}^n$ una secuencia finita de datos, que para empezar consideramos unidimensionales, una de las opciones más sencillas para construir un camino a partir de ella se basa en el uso de la interpolación lineal a trozos. En este caso, para aumentar la dimensionalidad del conjunto de datos y así poder describir mejor sus propiedades geométricas, ya que ya hemos visto que para caminos unidimensionales la signatura solo depende de sus puntos inicial y final, es habitual añadir una secuencia de datos auxiliar de la forma $\{t_i\}_{i=1}^n = \{0, 1, \ldots, n-1\}$, que se puede interpretar como valores de tiempo.

A partir de aquí, utilizando los puntos de $\mathbb{R}^2$ de la secuencia $\{t_i, X_i\}_{i=1}^n$, podemos construir por interpolación lineal a trozos el camino $X : [0, n-1] \longrightarrow \mathbb{R}^2$ dado por $X_s = \{t_i + (s-i+1)(t_{i+1} - t_i), X_i + (s-i+1)(X_{i+1} - X_i)\}\ \forall s \in [i-1, i],\ i \in \{1, \ldots, n-1\}$. Notemos que, como la signatura de un camino es invariante por reparametrización, realmente podríamos haber utilizado otras maneras de parametrizar el camino anterior para obtener después el mismo resultado en el estudio de su signatura.

Una vez comentada esta construcción, otra posibilidad sencilla para llevar a cabo el *embedding* de una secuencia de datos, viene dada por la interpolación lineal a trozos a lo largo de los ejes. En este caso, dada la secuencia $\{X_i\}_{i=1}^n$, podemos empezar como en la construcción anterior, aumentando su dimensionalidad para extraer mejor sus propiedades y considerando, por tanto, $\{t_i, X_i\}_{i=1}^n$. Entonces, para llevar a cabo esta interpolación, añadimos puntos auxiliares a la secuencia, de manera que para $i \in \{1, \ldots, n-1\}$ cualquiera, entre los puntos (t_i, X_i) y (t_{i+1}, X_{i+1}) añadimos (t_{i+1}, X_i). Obtenemos entonces la nueva secuencia de datos $\{\overline{t}_i, \overline{X}_i\}_{i=1}^{2n-1}$ construida de esta manera y, a partir de aquí, tal y como ya lo hemos hecho en el método anterior, podemos interpolarla linealmente a trozos, construyendo el camino $\overline{X} : [0, 2n-2] \longrightarrow \mathbb{R}^2$ correspondiente.

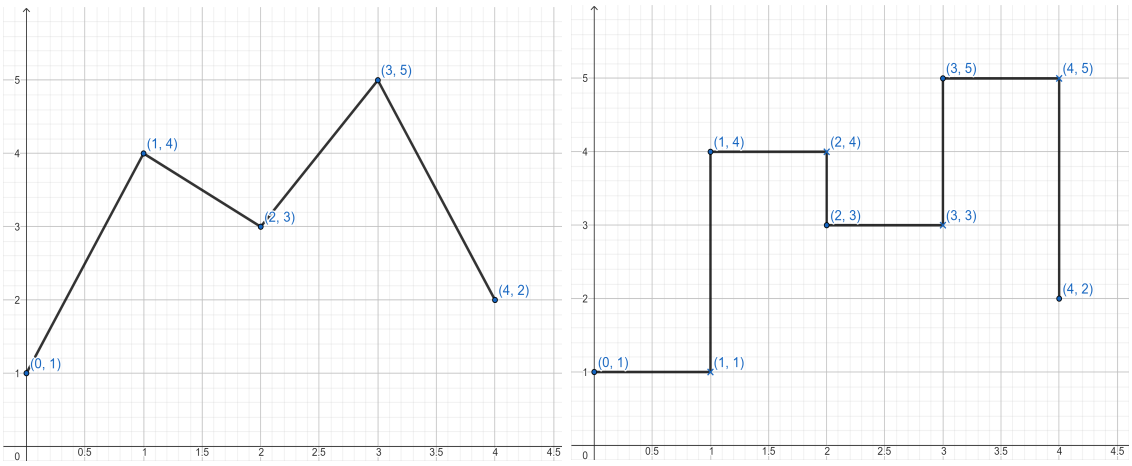
Para clarificar estas dos construcciones, podemos dar a continuación un ejemplo.

Ejemplo 6.1. En efecto, podemos considerar la secuencia de datos unidimensionales concreta, para $n = 5$, $\{X_i\}_{i=1}^5 = \{1, 4, 3, 5, 2\}$ y entonces, para aumentar su dimensionalidad por los motivos explicados, le añadimos los valores de tiempo $\{t_i\}_{i=1}^5 = \{0, 1, 2, 3, 4\}$. Por tanto, por un lado, podemos obtener la secuencia de puntos de $\mathbb{R}^2$ dada por las anteriores, $\{t_i, X_i\}_{i=1}^5$, y, a partir de ella, incluyendo los puntos auxiliares intermedios tal y como se ha explicado en el segundo método, construimos también $\{\overline{t}_i, \overline{X}_i\}_{i=1}^9$.

$$\{t_i, X_i\}_{i=1}^5 = \{(0,1), (1,4), (2,3), (3,5), (4,2)\}$$

$$\{\bar{t}_i, \overline{X}_i\}_{i=1}^{9} = \{(0,1),(1,1),(1,4),(2,4),(2,3),(3,3),(3,5),(4,5),(4,2)\}$$

Una vez hemos obtenido estas secuencias de puntos del plano, podemos aplicarles interpolación lineal a trozos para construir los caminos correspondientes $X : [0,4] \longrightarrow \mathbb{R}^2$ y $\overline{X} : [0,8] \longrightarrow \mathbb{R}^2$ ya explicados. Para ganar intuición de cómo son estos caminos, consideramos que lo mejor es mostrar mediante una representación gráfica cómo son sus trazas, que aparecen en la siguiente figura 7.



(a) Traza del camino X (b) Traza del camino $\overline{X}$

Figura 7: Representación gráfica de las trazas de los caminos lineales a trozos.

Finalmente, observamos que estos métodos que hemos introducido generalizan a conjuntos de datos de mayor dimensionalidad. En general, sea $\{X_i\}_{i=1}^{n} = \{X_i^1, \ldots, X_i^d\}_{i=1}^{n} \subseteq \mathbb{R}^d$ una secuencia de datos d-dimensionales, donde, por tanto, $X_i = (X_i^1, \ldots, X_i^d) \in \mathbb{R}^d \ \forall i \in \{1, \ldots, n\}$, siguiendo el mismo procedimiento de interpolación lineal a trozos que en el caso unidimensional, podemos considerar el camino $X : [0, n-1] \longrightarrow \mathbb{R}^d$ dado ahora por $X_t = \{X_t^1, \ldots, X_t^d\}$ de manera que, para $j \in \{1, \ldots, d\}$ cualquiera, $X_t^j = X_i^j + (t-i+1)(X_{i+1}^j - X_i^j) \ \forall t \in [i-1, i],\ i \in \{1, \ldots, n-1\}$. Además, también es posible utilizar la interpolación lineal a trozos a lo largo de los ejes para secuencias de datos multidimensionales. En efecto, continuando con la secuencia $\{X_i\}_{i=1}^{n} = \{X_i^1, \ldots, X_i^d\}_{i=1}^{n}$ anterior, para llevar a cabo este *embedding*, para todo $i \in \{1, \ldots, n-1\}$, añadimos entre $(X_i^1, \ldots, X_i^d)$ y $(X_{i+1}^1, \ldots, X_{i+1}^d)$ los $d-1$ puntos intermedios $(X_{i+1}^1, X_i^2, \ldots, X_i^{d-1}, X_i^d)$, $(X_{i+1}^1, X_{i+1}^2, \ldots, X_i^{d-1}, X_i^d)$ y así sucesivamente hasta $(X_{i+1}^1, X_{i+1}^2, \ldots, X_{i+1}^{d-1}, X_i^d)$ en este orden. Obtenemos así una nueva secuencia $\{\overline{X}_i\}_{i=1}^{n+(d-1)(n-1)}$ de datos a la que le podemos aplicar la interpolación lineal a trozos ya explicada para obtener el camino deseado.

Notemos que en este caso, para hacerlo con toda generalidad y porque en dimensiones mayores no es tan necesario para visibilizar la estructura geométrica del camino, no hemos añadido una secuencia auxiliar dada por el tiempo, aunque esta se podría incluir.

6.2. Transformación *Lead-Lag*

Una vez introducidos los métodos más sencillos para llevar a cabo el *embedding* de una secuencia de datos, basados en la interpolación lineal, pasamos a explicar la transformación *Lead-Lag*, otra técnica para construir *embeddings* habitual y que nos permite hacer visibles para la signatura del camino obtenido ciertas propiedades del conjunto de datos original.

Definición 6.2 (Secuencia *Lead*). *Sea $X = \{X_i\}_{i=1}^n \subseteq \mathbb{R}$ una secuencia de datos reales cualquiera, definimos entonces la secuencia Lead asociada como $X^{Lead} = \{X_i^{Lead}\}_{i=1}^{2n-1}$ donde, $\forall i \in \{1, \ldots, 2n-1\}$, se tiene que $X_i^{Lead} = X_{\frac{i+1}{2}}$ si i es impar y $X_i^{Lead} = X_{\frac{i}{2}+1}$ si i es par.*

Definición 6.3 (Secuencia *Lag*). *Sea $X = \{X_i\}_{i=1}^n \subseteq \mathbb{R}$ una secuencia de datos reales cualquiera, definimos entonces la secuencia Lag asociada como $X^{Lag} = \{X_i^{Lag}\}_{i=1}^{2n-1}$ donde, $\forall i \in \{1, \ldots, 2n-1\}$, se tiene que $X_i^{Lag} = X_{\frac{i+1}{2}}$ si i es impar y $X_i^{Lag} = X_{\frac{i}{2}}$ si i es par.*

Definición 6.4 (Transformación *Lead-Lag*). *Sea $X = \{X_i\}_{i=1}^n \subseteq \mathbb{R}$ una secuencia de datos reales cualquiera, podemos definir su transformación Lead-Lag como la secuencia de datos bidimensionales dada por $X^{Lead\text{-}Lag} = \{X_i^{Lead}, X_i^{Lag}\}_{i=1}^{2n-1}$.*

Por tanto, dada una secuencia de datos unidimensionales (posteriormente estudiaremos el caso multidimensional), una posibilidad para construir un camino asociado que nos permita estudiarla mediante la signatura pasa por obtener primero su transformación *Lead-Lag* y, entonces, aplicarle el método de interpolación lineal a trozos para obtener el camino deseado. Para clarificar estas nociones y procedimiento, damos un ejemplo.

Ejemplo 6.5. En efecto, continuando con la secuencia de datos unidimensionales introducida en el apartado anterior, $X = \{X_i\}_{i=1}^5 = \{1, 4, 3, 5, 2\}$, podemos empezar calculando sus secuencias *Lead* y *Lag* asociadas. Para ello, aplicamos las definiciones y de manera sencilla podemos obtener que:

$$X^{Lead} = \{X_i^{Lead}\}_{i=1}^9 = \{1, 4, 4, 3, 3, 5, 5, 2, 2\} \quad X^{Lag} = \{X_i^{Lag}\}_{i=1}^9 = \{1, 1, 4, 4, 3, 3, 5, 5, 2\}$$

Combinando estas dos nuevas secuencias, de nuevo aplicando la definición, podemos obtener entonces que la transformación *Lead-Lag* del conjunto de datos indexados original X es:

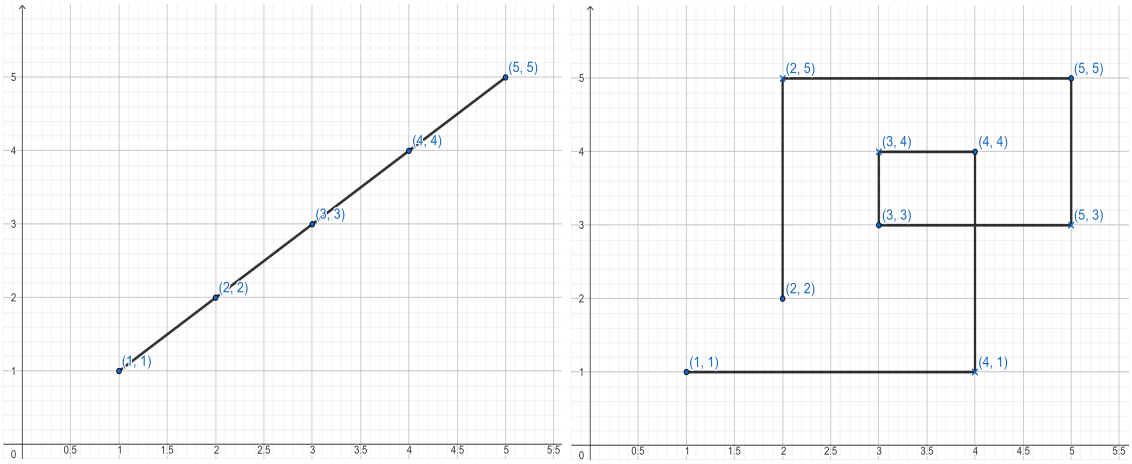
$$X^{Lead\text{-}Lag} = \{(1,1), (4,1), (4,4), (3,4), (3,3), (5,3), (5,5), (2,5), (2,2)\}$$

Finalmente, a esta última secuencia, le podemos aplicar el método de interpolación lineal a trozos, para obtener el camino correspondiente que se puede estudiar mediante su signatura. Para intuir mejor como es este camino, damos una representación gráfica de su traza, comparándola con la del camino obtenido por interpolación lineal a trozos directamente a partir de $\overline{X} = \{X_i, X_i\}_{i=1}^5 = \{(1,1), (4,4), (3,3), (5,5), (2,2)\}$, en la figura 8.

Por tanto, queda claro que aplicando primero la transformación *Lead-Lag*, obtenemos mucha más información sobre el comportamiento de la secuencia de datos, resaltando por ejemplo el orden en el que aparecen los valores, cosa no pasa si interpolamos directamente la secuencia de datos pensados en $\mathbb{R}^2$, duplicando sus valores. Heurísticamente, por tanto, este método nos da una buena alternativa, no necesariamente excluyente, a incluir la secuencia auxiliar dada por los valores de tiempo a la hora de construir el *embedding*.

Notemos que, tal y como se puede observar en la figura del ejemplo anterior, la interpolación lineal a trozos ya se produce a lo largo de los ejes. Esto es de hecho un fenómeno general, que podemos enunciar y demostrar de manera sencilla como proposición.

Proposición 6.6. *Sea $X = \{X_i\}_{i=1}^n \subseteq \mathbb{R}$ una secuencia de datos cualquiera y sea $X^{Lead\text{-}Lag} = \{X_i^{Lead}, X_i^{Lag}\}_{i=1}^{2n-1}$ su transformación Lead-Lag, entonces la traza del camino obtenido aplicando la interpolación lineal a trozos a la secuencia $X^{Lead\text{-}Lag}$ coincide con la traza del camino construido mediante la interpolación lineal a trozos a lo largo de los ejes.*



(a) Traza del camino interpolado de $\overline{X}$ (b) Traza del camino interpolado de $X^{Lead\text{-}Lag}$

Figura 8: Trazas de los caminos lineales a trozos correspondientes.

Demostración. En efecto, continuando con las notaciones arbitrarias del enunciado, dada la secuencia $X^{Lead\text{-}Lag} = \{X_i^{Lead}, X_i^{Lag}\}_{i=1}^{2n-1}$, recordemos que para obtener el camino dado por interpolación lineal a trozos a lo largo de los ejes, para todo $i \in \{1, \dots, 2n-2\}$ añadíamos a la secuencia, entre (X_i^{Lead}, X_i^{Lag}) y $(X_{i+1}^{Lead}, X_{i+1}^{Lag})$, el punto $(X_{i+1}^{Lead}, X_i^{Lag})$. Todo y eso, por definición de las secuencias *Lead* y *Lag*, por ser i y $i+1$ índices consecutivos, necesariamente $X_i^{Lead} = X_{i+1}^{Lead}$ o $X_i^{Lag} = X_{i+1}^{Lag}$, de manera que el punto intermedio añadido realmente coincide con uno de los dos originales, cosa que provoca que no se produzcan cambios en la traza del camino dado por interpolación lineal a trozos, ya que únicamente se han duplicado algunos puntos. □

Aparte, damos una de las propiedades más destacadas de la transformación *Lead-Lag*, que muestra cómo esta nos permite obtener información del conjunto de datos original combinándola con la signatura logarítmica del camino construido por interpolación. Antes, recordamos la noción de variación cuadrática de un conjunto de datos.

Definición 6.7. *Sea $X = \{X_i\}_{i=1}^n$ una secuencia de datos reales cualquiera, definimos la variación cuadrática de X como el valor $QV(X) \coloneqq \sum_{i=1}^{n-1}(X_{i+1} - X_i)^2 \in \mathbb{R}$.*

Este es un concepto propio de la teoría de procesos estocásticos y, además, se puede aplicar a conjuntos de datos finitos indexados para ayudar a describirlos, dándonos un valor relacionado también con su varianza. No entramos en detalles sobre las propiedades y aplicaciones de la variación cuadrática, pero, heurísticamente, se puede observar de su definición que estudia cómo varían los datos comparados con el anterior.

Proposición 6.8. *Sea $X = \{X_i\}_{i=1}^n \subseteq \mathbb{R}$ una secuencia de datos cualquiera y sea también $X^{Lead\text{-}Lag} = \{X_i^{Lead}, X_i^{Lag}\}_{i=1}^{2n-1}$ su transformación Lead-Lag, denotamos por $Y : [0, 2n-2] \longrightarrow \mathbb{R}^2$ el camino dado por interpolación lineal a trozos a partir de esta última secuencia y entonces se verifica que $\frac{QV(X)}{2}$ es uno de los coeficientes de la serie $\log S(Y)_{0,2n-2}$.*

Demostración. En efecto, continuando con las notaciones introducidas en el enunciado, para empezar, observando que Y es un camino bidimensional, afirmamos que el término de la signatura logarítmica dado por $\frac{1}{2}(S(Y)_{0,2n-2}^{1,2} - S(Y)_{0,2n-2}^{2,1})$, que ya habíamos visto previamente que realmente es uno de los primeros coeficientes de esta transformación de

la signatura, es el que verifica la propiedad. Por tanto, procedemos por inducción en n, recordando que en toda esta parte del trabajo estábamos suponiendo que $n \geq 2$, para probar que $\frac{QV(X)}{2} = \frac{1}{2}(S(Y)^{1,2}_{0,2n-2} - S(Y)^{2,1}_{0,2n-2})$, es decir, que $QV(X) = S(Y)^{1,2}_{0,2n-2} - S(Y)^{2,1}_{0,2n-2}$.

Caso inicial ($n = 2$): si tenemos que n es igual a 2, entonces notemos que, por definición, $X^{Lead\text{-}Lag} = \{X_i^{Lead}, X_i^{Lag}\}_{i=1}^3 = \{(X_1, X_1), (X_2, X_1), (X_2, X_2)\}$ y, por tanto, en vez de considerar el camino Y dado por interpolación lineal a trozos directamente, podemos construir las secuencias de datos ordenados $X^1 = \{(X_1, X_1), (X_2, X_1)\}$ y $X^2 = \{(X_2, X_1), (X_2, X_2)\}$ y tomar $Y^1 : [0,1] \longrightarrow \mathbb{R}^2$ y $Y^2 : [0,1] \longrightarrow \mathbb{R}^2$ los caminos obtenidos aplicando interpolación lineal a trozos respectivamente. Podemos describir fácilmente estos caminos, por tener como traza un segmento entre los dos puntos correspondientes, de manera que de hecho, $Y^1_t = \{X_1 + t(X_2 - X_1), X_1\}\ \forall t \in [0,1]$ y $Y^2_t = \{X_2, X_1 + t(X_2 - X_1)\}\ \forall t \in [0,1]$.

Entonces, aplicando a Y^2 la reparametrización habitual de $[1,2]$ a $[0,1]$ obtenida mediante una translación, restando 1 a todos los valores del intervalo $[1,2]$, tenemos también el nuevo camino $\overline{Y}^2 : [1,2] \longrightarrow \mathbb{R}^2$. Se verifica en esta situación, teniendo presente las definiciones y el procedimiento de interpolación lineal a trozos, que $Y = Y^1 * \overline{Y}^2$, de manera que aplicando la identidad de Chen podemos concluir que $S(Y)^{1,2}_{0,2} = S(Y^1)^{1,2}_{0,1} \otimes S(\overline{Y}^2)^{1,2}_{1,2} = S(Y^1)^{1,2}_{0,1} \otimes S(Y^2)^{1,2}_{0,1}$.

Por tanto, aplicando el desarrollo ya estudiado en la sección sobre la interpretación geométrica de la signatura, utilizando su invariancia por translaciones, podemos deducir directamente que $S(Y)^{1,2}_{0,2} = \frac{(X_2 - X_1)\cdot 0}{2} + (X_2 - X_1)(X_2 - X_1) + \frac{0\cdot(X_2 - X_1)}{2} = (X_2 - X_1)^2$ y también $S(Y)^{2,1}_{0,2} = \frac{(X_2 - X_1)\cdot 0}{2} + 0 \cdot 0 + \frac{0\cdot(X_2 - X_1)}{2} = 0$. Esto nos permite afirmar que $QV(X) = (X_2 - X_1)^2 = S(Y)^{1,2}_{0,2} - S(Y)^{2,1}_{0,2}$ y dejar así completo el caso inicial.

Caso sucesor: ahora, suponemos que el resultado es cierto para $n - 1 \geq 2$ y pasamos a probarlo para n, aplicando un razonamiento similar al del caso inicial. En efecto, sea $X = \{X_i\}_{i=1}^n$ la secuencia de datos estudiada, podemos considerar entonces las dos secuencias asociadas $X^1 = \{X_i\}_{i=1}^{n-1}$ y $X^2 = \{X_{i+n-2}\}_{i=1}^2 = \{X_{n-1}, X_n\}$, sus transformaciones *Lead-Lag* y los caminos que estas nos dan por interpolación lineal a trozos, que denotamos por $Y^1 : [0, 2n-4] \longrightarrow \mathbb{R}^2$ y $Y^2 : [0,2] \longrightarrow \mathbb{R}^2$ respectivamente. Además, considerando la reparametrización de clase C^1 de $[2n-4, 2n-2]$ a $[0,2]$ dada por una translación, restando $2n-4$ a los elementos del intervalo $[2n-4, 2n-2]$, podemos obtener el camino reparametrizado $\overline{Y}^2 : [2n-4, 2n-2] \longrightarrow \mathbb{R}^2$.

En conjunto, tomando la concatenación de estos dos caminos y teniendo presente la definición de la transformación *Lead-Lag*, podemos afirmar que si denotamos por $Y : [0, 2n-2] \longrightarrow \mathbb{R}^2$ al camino dado a partir de la transformación *Lead-Lag* de X, entonces se verifica que $Y = Y^1 * \overline{Y}^2$ y, por tanto, $S(Y)_{0,2n-2} = S(Y^1)_{0,2n-4} \otimes S(\overline{Y}^2)_{2n-4,2n-2} = S(Y^1)_{0,2n-4} \otimes S(Y^2)_{0,2}$.

Utilizando razonamientos ya detallados en la sección de la interpretación geométrica de la signatura, centrándonos en sus primeros términos, podemos ver que:

$$S(Y)^{1,2}_{0,2n-2} = S(Y^1)^{1,2}_{0,2n-4} + S(Y^1)^{1}_{0,2n-4} S(Y^2)^{2}_{0,2} + S(Y^2)^{1,2}_{0,2}$$

$$S(Y)^{2,1}_{0,2n-2} = S(Y^1)^{2,1}_{0,2n-4} + S(Y^1)^{2}_{0,2n-4} S(Y^2)^{1}_{0,2} + S(Y^2)^{2,1}_{0,2}$$

de manera que restando estas dos igualdades tenemos finalmente que se verifica la expresión $S(Y)^{1,2}_{0,2n-2} - S(Y)^{2,1}_{0,2n-2} = (S(Y^1)^{1,2}_{0,2n-4} - S(Y^1)^{2,1}_{0,2n-4}) + (S(Y^1)^{1}_{0,2n-4} S(Y^2)^{2}_{0,2} -$

$S(Y^1)^2_{0,2n-4}S(Y^2)^1_{0,2}) + (S(Y^2)^{1,2}_{0,2} - S(Y^2)^{2,1}_{0,2})$.

Aplicando la hipótesis de inducción, el cálculo de la signatura de un camino unidimensional ya estudiado y el hecho de que el camino Y^1 va de (X_1, X_1) a (X_{n-1}, X_{n-1}) y Y^2 va de (X_{n-1}, X_{n-1}) a (X_n, X_n), podemos concluir que $S(Y)^{1,2}_{0,2n-2} - S(Y)^{2,1}_{0,2n-2} = \sum_{i=1}^{n-2}(X_{i+1}-X_i)^2+(X_{n-1}-X_1)(X_n-X_{n-1})-(X_{n-1}-X_1)(X_n-X_{n-1})+(X_n-X_{n-1})^2 = \sum_{i=1}^{n-1}(X_{i+1}-X_i)^2 = QV(X)$, que es lo que nos faltaba por ver.

Por el principio de inducción matemática, queda probado el resultado. □

Finalmente, antes de concluir la sección, podemos indicar brevemente como se generaliza habitualmente la transformación *Lead-Lag* a secuencias de datos de mayor dimensionalidad. En efecto, sea $X = \{X_i\}_{i=1}^n = \{X_i^1, \ldots, X_i^d\}_{i=1}^n \subseteq \mathbb{R}^d$ una secuencia de datos d-dimensionales cualquiera, entonces para cada una de sus componentes $\{X_i^j\}_{i=1}^n$ con $j \in \{1, \ldots, d\}$, podemos considerar sus secuencias *Lead* y *Lag* asociadas, que denotamos por $\{X_i^{j,Lead}\}_{i=1}^{2n-1}$ y $\{X_i^{j,Lag}\}_{i=1}^{2n-1}$ respectivamente. Combinando en un solo conjunto de datos indexados todas las secuencias *Lead* obtenidas y algunas de las *Lag*, de aquellas componentes para las que queramos estudiar la variación de los datos, construimos la secuencia $\overline{X} = \{X_i^{1,Lead}, \ldots, X_i^{d,Lead}, X_i^{j_1,Lag}, \ldots, X_i^{j_k,Lag}\}_{i=1}^{2n-1}$ para algún $1 \leq k \leq d$ y $j_1 < \ldots < j_k \in \{1, \ldots, d\}$. A partir de aquí, podemos interpolar esta nueva secuencia siguiendo el método que consideremos pertinente para obtener el camino deseado. Notemos que, a diferencia de lo que pasaba en el caso unidimensional, en general no es cierto que aplicando interpolación lineal a trozos o la interpolación lineal a trozos a lo largo de los ejes obtengamos caminos con la misma traza, ya que en esta ocasión sí que es posible que se añadan puntos distintos de los anteriores.

6.3. Sumas parciales de secuencias de datos

A partir de aquí, pasamos a introducir otra de las posibilidades que se pueden utilizar para construir el *embedding* de una secuencia de datos, basada en la transformación *Lead-Lag* ya explicada. En efecto, centrándonos únicamente en el caso unidimensional, dada una secuencia de datos, podemos empezar definiendo su secuencia de sumas parciales correspondiente.

Definición 6.9 (Secuencia de sumas parciales)**.** *Sea $X = \{X_i\}_{i=1}^n \subseteq \mathbb{R}$ una secuencia de valores reales cualquiera, entonces definimos $CS(X)$ como su secuencia de sumas parciales, dada por $CS(X) = \{\tilde{X}_i\}_{i=0}^n$ donde para todo $i \in \{0, \ldots, n\}$, se verifica que $\tilde{X}_i = \sum_{j=1}^i X_j$. En particular, notemos que hemos empezado a indexar $CS(X)$ por $i = 0$ y $\tilde{X}_0 = 0$.*

En este caso, dado un conjunto de datos indexados, $X = \{X_i\}_{i=1}^n$, podemos considerar entonces su secuencia de sumas parciales $CS(X)$ y construir su camino asociado siguiendo alguno de los métodos explicados en las secciones anteriores. Entre ellos, una opción es aplicar la transformación *Lead-Lag* a esta nueva secuencia $CS(X)$, obteniendo posteriormente su camino dado por interpolación lineal a trozos. De esta manera, de nuevo, construimos un camino cuya signatura y signatura logarítmica tienen términos relevantes para el estudio del camino usando técnicas estadísticas, tal y como veremos a continuación. Otra posibilidad, una vez obtenida la secuencia $CS(X)$, podría ser también añadirle la secuencia auxiliar formada por los valores de tiempo para obtener después el camino correspondiente por interpolación. Para clarificar estas nociones, damos un ejemplo.

Ejemplo 6.10. En efecto, sea X la secuencia de datos reales que ya hemos utilizado en ejemplos anteriores, $X = \{X_i\}_{i=1}^5 = \{1, 4, 3, 5, 2\}$, para empezar, aplicando la definición, podemos calcular su secuencia de sumas parciales asociada, de manera que:

$$CS(X) = \{0, 1, 1+4, 1+4+3, 1+4+3+5, 1+4+3+5+2\} = \{0, 1, 5, 8, 13, 15\}$$

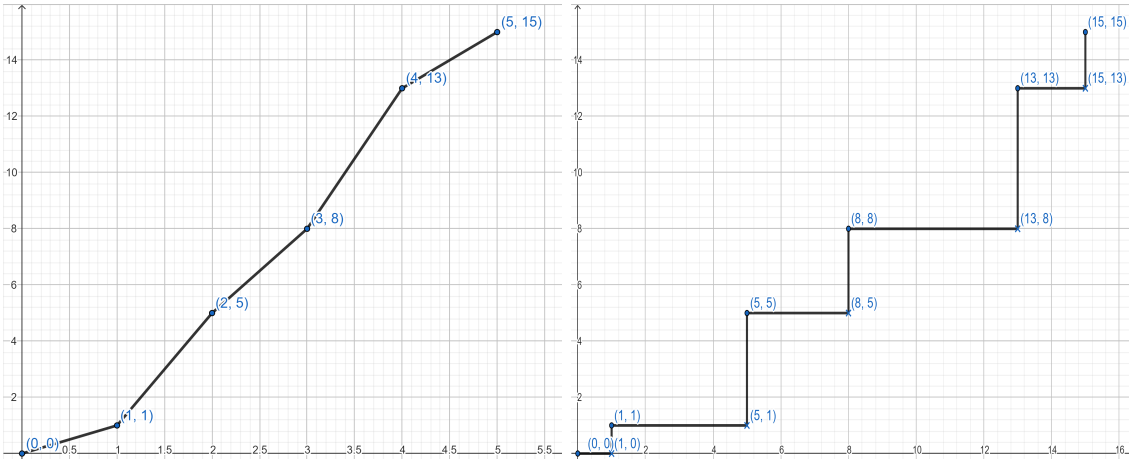
A partir de aquí, por un lado, podemos considerar también la secuencia auxiliar de valores de tiempo $\{t_i\}_{i=0}^5 = \{0, 1, 2, 3, 4, 5\}$ y, combinándola con la anterior, podemos obtener el conjunto de datos indexados $\{t_i, \tilde{X}_i\}_{i=0}^5$, denotando por $Y^1 : [0, 5] \longrightarrow \mathbb{R}^2$ su camino inducido por interpolación lineal a trozos (realmente, para estar en la misma situación que en secciones anteriores, calculamos el camino dado por interpolación a partir de $\{t_{i-1}, \tilde{X}_{i-1}\}_{i=1}^6$). Por otro lado, podemos también construir la transformación *Lead-Lag* de la secuencia $CS(X)$, que denotamos por $CS(X)^{Lead\text{-}Lag}$ (realmente de $\{\tilde{X}_{i-1}\}_{i=1}^6$ si queremos seguir estrictamente lo introducido en la sección anterior) y obtener su camino dado por interpolación lineal a trozos, $Y^2 : [0, 10] \longrightarrow \mathbb{R}^2$.

En resumen, tenemos secuencias de datos $\{t_i, \tilde{X}_i\}_{i=0}^5$ y $CS(X)^{Lead\text{-}Lag}$ respectivamente:

$$\{(0,0), (1,1), (2,5), (3,8), (4,13), (5,15)\}$$

$$\{(0,0), (1,0), (1,1), (5,1), (5,5), (8,5), (8,8), (13,8), (13,13), (15,13), (15,15)\}$$

y las trazas de los caminos Y^1 y Y^2 que inducen por interpolación lineal a trozos se muestran en la figura 9.



(a) Traza del camino Y^1 (b) Traza del camino Y^2

Figura 9: Trazas de los caminos lineales a trozos construidos

Una vez ejemplificadas las construcciones anteriores, dado un conjunto de datos indexados, centrándonos en la secuencia obtenida a partir de sus sumas parciales y de la transformación *Lead-Lag*, tal y como ya lo habíamos avanzado, podemos afirmar que se verifica la siguiente propiedad con respecto a los primeros momentos estadísticos de la secuencia de datos original.

Proposición 6.11. *Sea $X = \{X_i\}_{i=1}^n \subseteq \mathbb{R}$ una secuencia de valores reales cualquiera y sean entonces $CS(X)^{Lead\text{-}Lag}$ la transformación Lead-Lag de su secuencia de sumas parciales y Y su camino inducido por interpolación lineal a trozos, en este caso se pueden*

obtener la media y varianza del conjunto de datos X a partir de los términos de su signatura. Más precisamente, se tiene que, con las notaciones habituales:

$$\overline{X} = \frac{S(Y)^1_{0,2n}}{n} \quad Var(X) = \frac{1}{n}(S(Y)^{1,2}_{0,2n} - S(Y)^{2,1}_{0,2n}) - \frac{1}{n^2}(S(Y)^1_{0,2n})^2$$

Demostración. En efecto, continuando con las notaciones introducidas en el enunciado de la proposición, directamente, para empezar, podemos observar que el camino Y por definición de la secuencia de sumas parciales y de la transformación *Lead-Lag* va entre los puntos $(0,0)$ y $(\sum_{i=1}^n X_i, \sum_{i=1}^n X_i)$, cosa que nos permite afirmar, por lo ya estudiado sobre la signatura de los caminos unidimensionales, que $S(Y)^1_{0,2n} = \sum_{i=1}^n X_i - 0 = \sum_{i=1}^n X_i$, que es lo que queríamos ver. Por tanto, queda probado que $\overline{X} = \frac{\sum_{i=1}^n X_i}{n} = \frac{S(Y)^1_{0,2n}}{n}$.

A partir de aquí, centrándonos ahora en la otra igualdad, notemos que por definición Y es el camino obtenido interpolando la transformación *Lead-Lag* de la secuencia de datos $CS(X)$. Por tanto, podemos aplicar el resultado probado en la sección anterior sobre la variación cuadrática y concluir que $S(Y)^{1,2}_{0,2n} - S(Y)^{2,1}_{0,2n} = QV(CS(X)) = \sum_{i=0}^{n-1}(\tilde{X}_{i+1} - \tilde{X}_i)^2 =$ (por ser $\tilde{X}_{i+1} = X_{i+1} + \tilde{X}_i \ \forall i \in \{0, \dots, n-1\}$) $= \sum_{i=0}^{n-1}(X_{i+1})^2 = \sum_{i=1}^n (X_i)^2$. En consecuencia, poniendo en común todo lo anterior, tenemos que $\frac{1}{n}(S(Y)^{1,2}_{0,2n} - S(Y)^{2,1}_{0,2n}) - \frac{1}{n^2}(S(Y)^1_{0,2n})^2 = \frac{1}{n}\sum_{i=1}^n (X_i)^2 - \frac{1}{n^2}(\sum_{i=1}^n X_i)^2 =$ (por una identidad bien conocida de estadística) $= Var(X)$. Esto muestra la segunda afirmación y completa la prueba. □

6.4. Aplicación de la signatura en *machine learning*

Una vez introducidos diversos métodos para, dada una secuencia de datos, obtener un camino que la represente y que nos permita describir su comportamiento, pasamos a explicar brevemente el esquema que se sigue para utilizar la signatura en el campo del *machine learning*. Evidentemente, daremos solo una posibilidad sencilla de cómo hacerlo, pero, a partir de esta, hay infinidad de variantes más elaboradas en las que no entraremos.

Por tanto, para empezar, sea $X = \{X_i\}_{i=1}^n$ una secuencia de datos que queremos utilizar en un modelo de *machine learning*, podemos, tal y como se ha discutido previamente, llevar a cabo un *embedding* del conjunto de datos indexados y obtener Y el camino correspondiente. Cabe destacar que la técnica que se escoge para construir este camino, de entre todas las que hemos explicado y posiblemente otras, dependerá de cada situación concreta y puede ser que se tengan que hacer distintas pruebas para elegir la que nos da un mejor resultado, cosa que ejemplificaremos en la siguiente sección.

Entonces, una vez obtenido el camino Y, podemos calcular su signatura o su signatura logarítmica, según el caso y lo que queramos estudiar, truncándolas a un cierto nivel.

Finalmente, una vez calculados estos términos, ya podemos usarlos en el modelo de *machine learning* de nuestra elección, emparejándolos con el valor esperado y utilizándolos para entrenar o evaluar el modelo. De hecho, en la práctica, evidentemente, una sola secuencia de datos $X = \{X_i\}_{i=1}^n$ no es suficiente para entrenar uno de estos modelos de *machine learning*, de manera que realmente se trabaja con una familia de secuencias de datos $\{X^m\}_{m=1}^N$ donde $X^m = \{X_i^m\}_{i=1}^n \ \forall m \in \{1, \dots, N\}$ y para cada una de ellas se obtiene el camino correspondiente por interpolación Y^m y se calculan los primeros términos de su signatura o de su signatura logarítmica. Posteriormente, se emparejan con su valor esperado y se incluyen en el modelo.

7. Aplicación de la signatura al reconocimiento de dígitos manuscritos

En esta sección, describiremos una aplicación práctica de la signatura para ejemplificar su uso en el campo del *machine learning*, que ya hemos explicado previamente. Para ello, consideramos el problema de identificar dígitos escritos a mano, donde, dada una imagen con uno de estos dígitos manuscritos, el objetivo es decidir correctamente de qué dígito se trata, entre los posibles dígitos del 0 al 9. Este es un problema clásico y muy conocido en el campo del *machine learning* que se suele utilizar para probar y comparar los distintos algoritmos y métodos de clasificación disponibles. Además, es un problema idóneo para aplicar la signatura, ya que para reconocer un dígito es esencial poder describir adecuadamente su forma geométrica, que es en lo que destaca el método de la signatura.

7.1. Base de datos MNIST

Para llevar a cabo esta aplicación, antes de nada, tenemos que obtener los datos correspondientes a las imágenes con los dígitos manuscritos y, para ello, recurriremos a la base de datos MNIST. Esta es una base de datos de dominio público que procede del NIST (*National Institute of Standards and Technology*) y que cuenta con 60 000 imágenes de dígitos asociadas a su número correspondiente para entrenar el modelo de *machine learning* y con otras 10 000 imágenes para poner a prueba el método de clasificación utilizado y calcular su tasa de acierto. Cabe destacar que esta es una base de datos que está preparada para que sea fácil de usar sin tener que realizar ningún procesamiento previo, estando además equilibrada, cosa que evita que puedan surgir sesgos indeseados durante el aprendizaje.

Las imágenes de los dígitos manuscritos están formadas por 28×28 píxeles en blanco y negro que toman valores desde 0 (blanco) a 255 (negro) y vienen dadas por la secuencia de 784 datos de longitud que se obtiene recorriendo la imagen fila a fila de arriba a abajo. A continuación, mostramos como ejemplo algunas de estas imágenes.

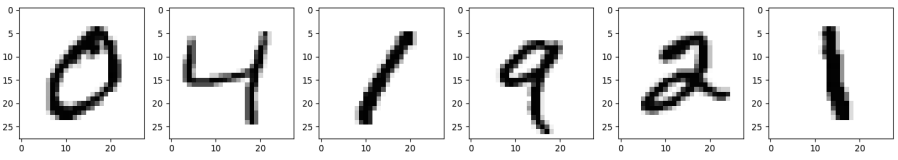


Figura 10: Muestra de las imágenes de dígitos manuscritos

7.2. Implementación de la aplicación

Para llevar a cabo este ejemplo de aplicación de la signatura, hemos seguido el proceso descrito al final de la sección anterior, implementándolo usando el lenguaje de programación *Python*. En primer lugar, una vez descargados los datos para entrenar el modelo y ponerlo después a prueba del paquete *Scikit-learn* [19], hemos empezado utilizando un programa de *Python* para calcular las signaturas de cada una de las secuencias de datos

correspondientes a las imágenes.

Más precisamente, para cada secuencia $X^m = \{X_i^m\}_{i=0}^{783}$ correspondiente a una imagen, tanto para los datos de entrenamiento como para los de comprobación posterior, primero, con el objetivo de obtener un buen *embedding* que sea capaz de representar la situación geométrica, le hemos añadido al valor de cada píxel su posición en la imagen. De esta manera, a partir de la secuencia anterior, hemos construido la nueva $Y^m = \{Y_i^m\}_{i=0}^{783}$, donde $\forall i \in \{0, \ldots, 783\}$ se tiene que $Y_i^m = (x, y, X_i^m)$ de forma que $i = 28x + y$ vienen dados por la división entera.

Una vez hecha esta transformación, a las secuencias de datos tridimensionales Y^m las hemos interpolado linealmente a trozos y hemos calculado los primeros niveles de la signatura de los caminos obtenidos, utilizando el paquete *iisignature* [21] para ello. Por tanto, al acabar, de esta manera, hemos conseguido tener una tabla con los valores de los primeros niveles de la signatura de cada una de las imágenes junto con el dígito real asociado que representan, esto tanto para los datos de entrenamiento como para los de comprobación.

A partir de aquí, para evitar que los valores estén en un rango desproporcionado y sea más fácil para posteriores modelos de *machine learning* aprender de los datos con el objetivo de llevar a cabo la clasificación, hemos estandarizado todas las columnas, restándole a cada elemento de una columna la media aritmética de la columna y dividiéndolo después entre la desviación típica de su misma columna. Notemos que, en este proceso, hemos decidido eliminar las columnas en que todos los datos fueran iguales, ya que no aportan nada a la clasificación posterior. Es natural que esto suceda con algunas columnas, como por ejemplo en aquellas que se corresponden con términos de la signatura que involucran solo a las dos primeras componentes del camino, que siempre son las mismas para todas las imágenes y, por tanto, el valor de estos términos de la signatura también. A continuación, para clarificar las ideas, mostramos como ha quedado la tabla definitiva que contiene por filas los valores estandarizados de la signatura junto con su dígito correspondiente.

	0	1	2	3	4	5	6	7	8	9	...	79	80	81	82	83	84	85	86	87	Pred
0	0.033285	-0.162497	-0.033285	0.162497	0.032216	-0.170656	0.018242	0.170666	-0.028742	0.032909	...	-0.027435	-0.024337	0.024726	0.121412	0.241865	-0.147420	-0.024726	0.147420	-0.117955	5
1	0.033285	-0.574795	-0.033285	0.574795	0.032216	-0.459495	0.018242	0.459502	-0.028742	0.032909	...	-0.027435	-0.024337	0.024726	0.569411	0.430022	-0.376593	-0.024726	0.376593	1.194145	0
2	0.033285	0.770889	-0.033285	-0.770889	0.032216	0.751843	0.018242	-0.751825	-0.028742	0.032909	...	-0.027435	-0.024337	0.024726	-0.958535	-0.870788	1.009237	-0.024726	-1.009237	0.151093	4
3	0.033285	1.037439	-0.033285	-1.037439	0.032216	0.965537	0.018242	-0.965517	-0.028742	0.032909	...	-0.027435	-0.024337	0.024726	-0.973295	-0.894985	0.824920	-0.024726	-0.824920	-0.382558	1
4	0.033285	0.335378	-0.033285	-0.335378	0.032216	0.318844	0.018242	-0.318830	-0.028742	0.032909	...	-0.027435	-0.024337	0.024726	-0.341176	-0.290436	0.354395	-0.024726	-0.354395	-0.264710	9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
59995	0.033285	0.036491	-0.033285	-0.036491	0.032216	0.043490	0.018242	-0.043479	-0.028742	0.032909	...	-0.027435	-0.024337	0.024726	-0.162659	-0.200636	0.307264	-0.024726	-0.307264	0.326477	8
59996	0.033285	-0.134780	-0.033285	0.134780	0.032216	-0.156545	0.018242	0.156555	-0.028742	0.032909	...	-0.027435	-0.024337	0.024726	0.105984	0.102412	-0.079119	-0.024726	0.079119	-0.014561	3
59997	0.033285	0.460338	-0.033285	-0.460338	0.032216	0.405409	0.018242	-0.405394	-0.028742	0.032909	...	-0.027435	-0.024337	0.024726	-0.519460	-0.499228	0.486115	-0.024726	-0.486115	-1.743507	5
59998	0.033285	0.657132	-0.033285	-0.657132	0.032216	0.707388	0.018242	-0.707371	-0.028742	0.032909	...	-0.027435	-0.024337	0.024726	-0.802312	-0.888163	0.941196	-0.024726	-0.941196	-0.239695	6
59999	0.033285	0.600311	-0.033285	-0.600311	0.032216	0.587475	0.018242	-0.587459	-0.028742	0.032909	...	-0.027435	-0.024337	0.024726	-0.755639	-0.777179	0.856635	-0.024726	-0.856635	-0.200227	8

60000 rows × 89 columns

Figura 11: Por filas valores de la signatura estandarizados de cada imagen con su dígito

Finalmente, hemos construido un modelo de *machine learning* con los datos anteriores ya procesados y lo hemos puesto a prueba para medir su tasa de acierto. Después de hacer diversas pruebas para elegir el modelo, nos hemos decantado por el algoritmo *HistGradientBoostingClassifier* del paquete para *Python* llamado *Scikit-learn*, de código abierto, que hemos notado que da buenos resultados y nos permite llevar a cabo este ejemplo.

Cabe recordar que nuestro objetivo es mostrar y dar un ejemplo sencillo de cómo se puede aplicar la signatura en el campo del *machine learning* para hacer predicciones, no entrar a estudiar en detalle con qué modelos y técnicas de este campo se puede combinar la signatura para obtener mejores resultados, cosa que depende de cada problema concreto y que queda fuera del alcance del trabajo.

Para acabar, hemos repetido el procedimiento anterior de manera completamente análoga, pero utilizando la signatura logarítmica en lugar de la signatura, para poder comparar posteriormente los resultados. También, de nuevo, hemos procedido de forma similar, pero esta vez, en lugar de hacer el *embedding* añadiendo las coordenadas de cada píxel, hemos partido de la secuencia unidimensional asociada a cada imagen y le hemos aplicado la transformación *Lead-lag* descrita en la sección anterior para después ya aplicarle la signatura o la signatura logarítmica.

En total, por tanto, hemos llevado a cabo 4 experimentos distintos y, ahora, pasamos a comparar los resultados obtenidos.

7.3. Resultados de la aplicación

A continuación, mostramos el cuadro 1 donde se comparan las tasas de acierto obtenidas sobre los datos de comprobación para cada uno de los 4 experimentos, correspondientes a usar uno u otro *embedding* y a utilizar la signatura o la signatura logarítmica.

Tasas de acierto		Método utilizado			
		Signatura		Signatura logarítmica	
		Lead-Lag	Posición píxel	*Lead-Lag*	Posición píxel
Nivel de la signatura	4	0,392	0,647	0,386	0,704
	5	0,406	0,838	0,410	0,850
	6	0,453	0,881	0,442	0,866
	7	0,503	0,906	0,493	0,873

Cuadro 1: Tasas de acierto de los modelos de *machine learning* construidos

Observando las tasas de acierto, llama la atención como el *embedding* utilizado es determinante para poder obtener buenos resultados, cosa que deja clara su importancia y recalca que es esencial estudiar primero el problema para poder elegir una buena transformación de los datos que sea capaz de describir y mostrar su estructura y propiedades. En todo caso, notemos también que los resultados continúan siendo muy mejorables, ya que hay muchos otros algoritmos que no se basan en la signatura y que han conseguido dar mejores tasas de acierto cuando se han aplicado a este problema. De hecho, a modo de ejemplo, el mismo algoritmo de *machine learning* utilizado en los experimentos anteriores, pero aplicado directamente a los datos originales, sin utilizar la signatura, da una mayor tasa de acierto, de 0,981. Todo y eso, recordemos que nuestro objetivo era únicamente dar un pequeño ejemplo de uso de la signatura y no encontrar su configuración óptima con la que fuera posible obtener el mejor resultado.

El código de todos los programas utilizados para llevar a cabo los experimentos de esta sección se puede encontrar en el anexo de este trabajo o en la carpeta de *Google Drive* asociada al enlace:

`https://drive.google.com/drive/folders/1F_HAwIAk1jClMwPksnf3YMgpNJbLvU4I?usp=drive_link`

8. Generalizaciones fraccionarias de la signatura

8.1. Signatura fraccionaria

Una vez desarrollada la aplicación práctica anterior, observamos que los resultados obtenidos no han sido excepcionalmente buenos, tal y como ya lo hemos comentado en la sección previa. Por ello, aunque no era nuestro objetivo encontrar la mejor configuración posible, sino dar un pequeño ejemplo, las observaciones anteriores nos motivan a proponer diversas generalizaciones de la signatura para intentar mejorar estos resultados.

Para ello, empezamos recordando la noción de la integral de Riemann-Liouville, propia del cálculo fraccionario.

Definición 8.1 (Integral de Riemann-Liouville)**.** *Sea $[a,b] \subseteq \mathbb{R}$ un intervalo, sea $f : [a,b] \longrightarrow \mathbb{R}$ una función continua a trozos (es decir, continua salvo en un número finito de puntos donde los límites laterales sí que existen aunque no coincidan) cualquiera y sea también $\alpha > 0 \in \mathbb{R}$ un parámetro fijado, definimos la integral de Riemann-Liouville de f en el intervalo $[a,b]$ de orden α como la función $I^{\alpha}_{a^+} f : [a,b] \longrightarrow \mathbb{R}$ dada por*

$$I^{\alpha}_{a^+} f(x) \coloneqq \frac{1}{\Gamma(\alpha)} \int_a^x (x-t)^{\alpha-1} f(t) dt \quad \forall x \in [a,b]$$

donde Γ denota la función Gamma habitual.

Entonces, teniendo presente lo anterior, proponemos sustituir en la definición de la signatura de un camino la integral original por esta de Riemann-Liouville, llamando a la nueva signatura obtenida signatura fraccionaria.

Definición 8.2 (Signatura fraccionaria de un camino)**.** *Sea $X : [a,b] \longrightarrow \mathbb{R}^d$ un camino arbitrario, sea $\alpha > 0 \in \mathbb{R}$ un parámetro y sea también $I \in W$ cualquiera, $I = (i_1, \ldots, i_k)$, podemos definir recursivamente las expresiones $S^{\alpha}(X)^I_{a,b}$ de la siguiente manera:*

Si $k = 1$, entonces:

$$S^{\alpha}(X)^I_{a,b} \coloneqq \frac{1}{\Gamma(\alpha)} \int_a^b (b-t)^{\alpha-1} dX_t^{i_1} = \frac{1}{\Gamma(\alpha)} \int_a^b (b-t)^{\alpha-1} \dot{X}_t^{i_1} dt$$

Si $k > 1$, denotamos $J = (i_1, \ldots, i_{k-1})$ y entonces tenemos que:

$$S^{\alpha}(X)^I_{a,b} \coloneqq \frac{1}{\Gamma(\alpha)} \int_a^b (b-t)^{\alpha-1} S^{\alpha}(X)^J_{a,t} dX_t^{i_k} = \frac{1}{\Gamma(\alpha)} \int_a^b (b-t)^{\alpha-1} S^{\alpha}(X)^J_{a,t} \dot{X}_t^{i_k} dt$$

Además, con las notaciones anteriores, definimos la signatura fraccionaria de X de orden α como la sucesión $S^{\alpha}(X)_{a,b} = \{S^{\alpha}(X)^I_{a,b}\}_{I \in W}$, donde por convenio añadimos 1 como elemento inicial correspondiente a la palabra de 0 letras.

Antes de nada, podemos observar que, como $\Gamma(1) = 1$ y también $x^{1-1} = 1 \, \forall x > 0 \in \mathbb{R}$, la signatura fraccionaria de orden 1 de un camino coincide con su signatura, ya sus términos pasan a ser la secuencia de integrales iteradas de Riemann-Stieltjes indicada por el multi-índice correspondiente. Por tanto, la signatura fraccionaria es una generalización de la signatura. Para conocer mejor como se comporta esta nueva signatura, empezamos calculándola explícitamente para caminos lineales.

Proposición 8.3 (Signatura fraccionaria de un camino lineal)**.** *Sea $[a,b]$ un intervalo real $(a<b)$, sea $\alpha>0$ un parámetro cualquiera y sea también $X:[a,b]\longrightarrow\mathbb{R}^d$ un camino lineal entre los puntos $(A_1,\ldots,A_d)$, $(B_1,\ldots,B_d)\in\mathbb{R}^d$, es decir, $X_t^i=\frac{1}{b-a}(A_i(b-t)+B_i(t-a))\,\forall t\in[a,b]$, entonces para $I\in W$ cualquiera, $I=(i_1,\ldots,i_k)$, se tiene que:*

$$S^\alpha(X)^I_{a,b}=\frac{\prod_{j=1}^k(B_{i_j}-A_{i_j})}{\Gamma(\alpha)\Gamma(1+(k-1)\alpha)}(b-a)^{(\alpha-1)k}\beta((k-1)\alpha+1,\alpha)$$

donde β denota la función beta.

Demostración. En efecto, continuando con las notaciones ya introducidas en el enunciado, antes de nada, observamos que es un resultado bien conocido el hecho de que $\beta(x,y)=\frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}\;\forall x,y>0\in\mathbb{R}$ i, por tanto, en particular, $\beta((k-1)\alpha+1,\alpha)=\frac{\Gamma((k-1)\alpha+1)\Gamma(\alpha)}{\Gamma(k\alpha+1)}$. En consecuencia, utilizando esta última igualdad, será suficiente probar que se tiene que:

$$S^\alpha(X)^I_{a,b}=\frac{\prod_{j=1}^k(B_{i_j}-A_{i_j})}{\Gamma(1+k\alpha)}(b-a)^{(\alpha-1)k}=\frac{\prod_{j=1}^k(B_{i_j}-A_{i_j})}{\Gamma(1+k\alpha)(b-a)^k}(b-a)^{k\alpha}$$

Notemos entonces que, aplicando la definición de los términos de la signatura fraccionaria, podemos afirmar que $S^\alpha(X)^I_{a,b}=\frac{1}{\Gamma(\alpha)}\int_a^b(b-t_k)^{\alpha-1}S^\alpha(X)^{i_1,\ldots,i_{k-1}}_{a,t_k}\dot{X}^{i_k}_{t_k}dt_k=\frac{1}{\Gamma(\alpha)}\int_a^b(b-t_k)^{\alpha-1}\frac{1}{\Gamma(\alpha)}\int_a^{t_k}(t_k-t_{k-1})^{\alpha-1}S^\alpha(X)^{i_1,\ldots,i_{k-2}}_{a,t_{k-1}}\dot{X}^{i_{k-1}}_{t_{k-1}}dt_{k-1}\dot{X}^{i_k}_{t_k}dt_k=\frac{1}{(\Gamma(\alpha))^2}\int_{a<t_{k-1}<t_k<b}(b-t_k)^{\alpha-1}(t_k-t_{k-1})^{\alpha-1}S^\alpha(X)^{i_1,\ldots,i_{k-2}}_{a,t_{k-1}}\dot{X}^{i_{k-1}}_{t_{k-1}}\dot{X}^{i_k}_{t_k}dt_{k-1}dt_k$ y así iterativamente concluimos que:

$$S^\alpha(X)^I_{a,b}=\frac{1}{\Gamma(\alpha)^k}\int_{a<t_1<\ldots<t_k<b}(b-t_k)^{\alpha-1}\ldots(t_2-t_1)^{\alpha-1}\dot{X}^{i_1}_{t_1}\ldots\dot{X}^{i_k}_{t_k}dt_1\ldots dt_k=$$

$$\frac{\prod_{j=1}^k(B_{i_j}-A_{i_j})}{\Gamma(\alpha)^k(b-a)^k}\int_{a<t_1<\ldots<t_k<b}(b-t_k)^{\alpha-1}(t_k-t_{k-1})^{\alpha-1}\ldots(t_2-t_1)^{\alpha-1}dt_1\ldots dt_k$$

donde hemos sustituido el valor de las derivadas del camino, que son constantes en todo punto y, por tanto, no dependen de las variables t_j correspondientes. En consecuencia, para probar el resultado, será suficiente ver por inducción en $k\geq1$ que:

$$\int_{a<t_1<\ldots<t_k<b}(b-t_k)^{\alpha-1}(t_k-t_{k-1})^{\alpha-1}\ldots(t_2-t_1)^{\alpha-1}dt_1\ldots dt_k=\frac{\Gamma(\alpha)^k}{\Gamma(1+k\alpha)}(b-a)^{k\alpha}$$

Caso inicial $(k=1)$: para empezar, es directo ver que $\int_{a<t_1<b}(b-t_1)^{\alpha-1}dt_1=\int_a^b(b-t_1)^{\alpha-1}dt_1=(-\frac{(b-t_1)^\alpha}{\alpha}|_a^b=\frac{(b-a)^\alpha}{\alpha}=\frac{\Gamma(\alpha)(b-a)^\alpha}{\Gamma(\alpha)\alpha}=\frac{\Gamma(\alpha)(b-a)^\alpha}{\Gamma(\alpha+1)}$, que es lo que teníamos que observar, cosa que completa el caso inicial.

Caso sucesor: suponemos que el resultado es cierto para $k-1\geq1$ y pasamos a probarlo para k. En efecto, $\int_{a<t_1<\ldots<t_k<b}(b-t_k)^{\alpha-1}(t_k-t_{k-1})^{\alpha-1}\ldots(t_2-t_1)^{\alpha-1}dt_1\ldots dt_k=\int_a^b(b-t_k)^{\alpha-1}(\int_{a<t_1<\ldots<t_{k-1}<t_k}(t_k-t_{k-1})^{\alpha-1}\ldots(t_2-t_1)^{\alpha-1}dt_1\ldots d_{t_{k-1}})dt_k=$ (aplicando la hipótesis de inducción) $=\int_a^b(b-t_k)^{\alpha-1}\frac{\Gamma(\alpha)^{k-1}}{\Gamma(1+(k-1)\alpha)}(t_k-a)^{(k-1)\alpha}dt_k=\frac{\Gamma(\alpha)^{k-1}}{\Gamma(1+(k-1)\alpha)}\int_a^b(b-a-(t_k-a))^{\alpha-1}(t_k-a)^{(k-1)\alpha}dt_k=\frac{\Gamma(\alpha)^{k-1}}{\Gamma(1+(k-1)\alpha)}\int_0^1(b-a-(b-a)u)^{\alpha-1}((b-a)u)^{(k-1)\alpha}(b-a)du=\frac{\Gamma(\alpha)^{k-1}}{\Gamma(1+(k-1)\alpha)}(b-a)^{k\alpha}\int_0^1(1-u)^{\alpha-1}u^{(k-1)\alpha}du=\frac{\Gamma(\alpha)^{k-1}}{\Gamma(1+(k-1)\alpha)}(b-a)^{k\alpha}\beta((k-1)\alpha+1,\alpha)=\frac{\Gamma(\alpha)^k}{\Gamma(1+k\alpha)}(b-a)^{k\alpha}$ donde hemos aplicado el cambio de variable $\frac{t_k-a}{b-a}=u$, cosa que completa el caso sucesor.

Por el principio de inducción matemática, queda probado el resultado. □

Además, podemos observar que la signatura fraccionaria, como la signatura, es invariante por translaciones del camino, pero no por reparametrizaciones de clase C^1, cosa que recogemos en la siguiente proposición.

Proposición 8.4. *Sea $X : [a,b] \longrightarrow \mathbb{R}^d$ un camino cualquiera, sea $\alpha > 0 \in \mathbb{R}$ un parámetro y sea también $c \in \mathbb{R}^d$ un vector arbitrario, consideramos el camino $X + c : [a,b] \longrightarrow \mathbb{R}^d$ dado por $(X+c)_t = X_t + c \quad \forall t \in [a,b]$ y entonces tenemos que se verifica que $S^\alpha((X+c))_{a,b} = S^\alpha(X)_{a,b}$, es decir, esta signatura de X es invariante por translaciones.*

En cambio, la signatura fraccionaria de un camino de orden $\alpha \neq 1$ no es invariante por reparametrizaciones de clase C^1.

Demostración. En efecto, continuando con las notaciones ya introducidas en el enunciado, observamos que la primera parte de este se puede demostrar de manera análoga al resultado correspondiente que verifica la signatura, basando la prueba en que $\forall t \in (a,b)$ e $i \in \{0,\dots,d\}$, $(\overset{\cdot}{X+c})^i_t = \dot{X}^i_t$ y en que en la definición de la signatura fraccionaria únicamente aparecen las derivadas de las componentes del camino pero no el camino directamente. Por tanto, la signatura fraccionaria de $X + c$ es la misma que la de X.

Respecto a la no invariancia por reparametrizaciones de la signatura fraccionaria para $\alpha \neq 1$, esto se puede deducir fácilmente del cálculo anterior para el caso de caminos lineales. Dando un contraejemplo sencillo, consideramos $X : [0,1] \longrightarrow \mathbb{R}$ el camino dado por $X_t = t \quad \forall t \in [0,1]$ y tomamos $Y : [0,2] \longrightarrow \mathbb{R}$ dado por $Y_t = \frac{t}{2} \quad \forall t \in [0,2]$, una reparametrización de X. Entonces, $S^\alpha(X)^1_{0,1} = \frac{1}{\Gamma(1+\alpha)} \neq \frac{1}{\Gamma(1+\alpha)} 2^{\alpha-1} = S^\alpha(Y)^1_{0,2}$ que claramente son diferentes, cosa que completa la prueba. □

A partir de aquí, una vez introducida la signatura fraccionaria de un camino, podemos observar que esta generaliza la propiedad que verificaba la signatura en el contexto de las ecuaciones diferenciales controladas lineales, siendo cierto ahora un resultado análogo para las ecuaciones diferenciales fraccionarias de Caputo. En efecto, empezamos recordando la definición de ecuación diferencial fraccionaria de Caputo y dando un resultado que generaliza el teorema de los iterados de Picard para ciertas ecuaciones lineales. Esta definición, teorema y demostración posterior se encuentran o se pueden deducir del contenido de las referencias [1], [15] y [22].

Definición 8.5 (Solución de un problema de valor inicial fraccionario de Caputo)**.** *Sean $a < h \leq b \in \mathbb{R}$, sea $f : [a,b] \times \mathbb{R}^n \longrightarrow \mathbb{R}^n$ una función continua, sean $0 < \alpha \leq 1$ un parámetro fijado y $u_0 \in \mathbb{R}^n$ y sea también $u : [a,h] \longrightarrow \mathbb{R}^n$ una función, decimos que u es una solución del problema de valor inicial fraccionario de Caputo ${}^cD^\alpha u = f(t,u)$, $u(a) = u_0$ si se verifica que $\forall t \in [a,h]$, $u(t) = u_0 + \frac{1}{\Gamma(\alpha)} \int_a^t (t-s)^{\alpha-1} f(s,u(s))ds$.*

Teorema 8.6 (Teorema de Picard para ecuaciones diferenciales lineales fraccionarias de Caputo)**.** *Sea $0 < \alpha \leq 1$ un parámetro fijado, $I = [a,b]$ un intervalo cerrado real y $\Omega = I \times \mathbb{R}^n$. Sea también $A : I \longrightarrow \mathbb{L}(\mathbb{R}^n, \mathbb{R}^n)$ continua. Tomamos entonces $f : \Omega \longrightarrow \mathbb{R}^n$ dada por $f(t,x) = A(t)x \quad \forall (t,x) \in \Omega$ y se verifica que para $(a,u_0) \in \Omega$, el problema de valor inicial fraccionario de Caputo determinado por f, (a,u_0) y α tiene solución única definida en I obtenida como límite uniforme de los iterados de Picard $\{u^n\}_n$ dados por $u^0(t) = u_0$ y $u^n(t) = u_0 + \frac{1}{\Gamma(\alpha)} \int_a^t (t-s)^{\alpha-1} f(s,u^{n-1}(s))ds \quad \forall t \in [a,b]$ y $n \geq 1$.*

Demostración. Con las notaciones introducidas por el enunciado, pasamos a probar la versión indicada del teorema de Picard para estas ecuaciones diferenciales. Para ello,

antes de nada, notemos que f es continua por serlo A por hipótesis y pasamos a ver que también es Lipschitz respecto de la segunda variable en Ω.

En efecto, sean $(t,x),(t,y)\in\Omega$ arbitrarios, denotando por $\|\cdot\|$ la norma euclidiana de $\mathbb{R}^n$, podemos ver que $\|f(t,x)-f(t,y)\| = \|A(t)x-A(t)y\| = \|A(t)(x-y)\| \leq$ (considerando la norma matricial inducida correspondiente) $\leq \|A(t)\|\,\|x-y\| \leq (\sup_{t\in I}\|A(t)\|)\|x-y\| =$ (por ser I un compacto y $\|A(t)\|$ una función continua por serlo A, sabemos que toda función continua definida en un compacto alcanza sus extremos y denotamos $L \coloneqq \max_{t\in I}\|A(t)\|) = L\|x-y\|$. Esto prueba que f tiene la propiedad para la constante $L\geq 0\in\mathbb{R}$.

A partir de aquí, consideramos $Z = C^0(I,\mathbb{R}^n)$ el espacio de las funciones continuas de I en $\mathbb{R}^n$ con la norma $\|\cdot\| : Z \longrightarrow \mathbb{R}$ dada por $\|\phi\| = \sup_{t\in I}\|\phi(t)\| \quad \forall \phi\in Z$ (en esta definición, llevamos a cabo un abuso de notación habitual y denotamos la norma en Z de la misma manera que la euclidiana de $\mathbb{R}^n$). Notemos que esta está bien definida, ya que I es un intervalo cerrado y acotado de $\mathbb{R}$ y, por tanto, es compacto, de manera que por resultados de topología las funciones continuas como $\|\phi\|$ definidas en un compacto son acotadas y alcanzan sus extremos. Además, con esta norma, es bien conocido que Z es un espacio de Banach.

Entonces, definimos el operador $T : Z \longrightarrow Z$ dado por $T(\phi)(t) = u_0 + \frac{1}{\Gamma(\alpha)}\int_a^t (t-s)^{\alpha-1}f(s,\phi(s))ds \quad \forall t\in I$ y $\phi\in Z$ y veamos que está bien definido. En efecto, sean $\phi\in Z$ y $t\in I$, observamos que $\frac{1}{\Gamma(\alpha)}\int_a^t(t-s)^{\alpha-1}f(s,\phi(s))ds\in\mathbb{R}^n$ y podemos evaluar f en $(s,\phi(s)) \quad \forall s\in[a,t]\subseteq I$, ya que $\phi(s)\in\mathbb{R}^n$ porque $\phi\in Z$ y en consecuencia $(s,\phi(s))\in\Omega$. Además, utilizando que f y ϕ son continuas, $0<\alpha\leq 1$ y que estamos trabajando en I un intervalo compacto, se puede observar directamente que la expresión anterior es continua en t, cosa que en conjunto prueba que $T(\phi)$ está bien definida en I y es continua, es decir, $T(\phi)\in Z$ y, por tanto, T está bien definido.

Ahora, sean $\phi_1,\phi_2\in Z$ arbitrarios, afirmamos que $\|T^n\phi_1(t)-T^n\phi_2(t)\| \leq \frac{L^n}{\Gamma(n\alpha+1)}(t-a)^{n\alpha}\|\phi_1-\phi_2\| \quad \forall t\in I$ y $n\geq 1\in\mathbb{N}$ y pasamos a probarlo por inducción.

Caso inicial ($n=1$): sea $t\in I$ cualquiera, directamente podemos ver que $\|T\phi_1(t) - T\phi_2(t)\| =$ (por definición de T) $= \|\frac{1}{\Gamma(\alpha)}\int_a^t(t-s)^{\alpha-1}(f(s,\phi_1(s))-f(s,\phi_2(s)))ds\| \leq \frac{1}{\Gamma(\alpha)}\int_a^t(t-s)^{\alpha-1}\|f(s,\phi_1(s))-f(s,\phi_2(s))\|ds \leq$ (por ser f de Lipschitz respecto de la segunda variable en su dominio) $\leq \frac{L}{\Gamma(\alpha)}\int_a^t(t-s)^{\alpha-1}\|\phi_1(s)-\phi_2(s)\|ds \leq$ (acotando por el supremo de la norma) $\leq \frac{L}{\Gamma(\alpha)}\|\phi_1-\phi_2\|(-\frac{(t-s)^\alpha}{\alpha}|_a^t = \frac{L}{\Gamma(\alpha+1)}(t-a)^\alpha\|\phi_1-\phi_2\|$, que es lo que teníamos que concluir.

Caso sucesor: suponemos ahora como hipótesis de inducción que el resultado es cierto para $n-1\geq 1$ y pasamos a probarlo para n. En efecto, tenemos que $\|T^n\phi_1(t)-T^n\phi_2(t)\| \leq$ (como en el caso inicial) $\leq \frac{1}{\Gamma(\alpha)}\int_a^t(t-s)^{\alpha-1}\|f(s,T^{n-1}\phi_1(s))-f(s,T^{n-1}\phi_2(s))\|ds \leq$ (por ser f Lipschitz respecto de la segunda variable) $\leq \frac{L}{\Gamma(\alpha)}\int_a^t(t-s)^{\alpha-1}\|T^{n-1}\phi_1(s)-T^{n-1}\phi_2(s)\|ds \leq$ (por hipótesis de inducción) $\leq \frac{L^n}{\Gamma((n-1)\alpha+1)\Gamma(\alpha)}\|\phi_1-\phi_2\|\int_a^t(t-a-(s-a))^{\alpha-1}(s-a)^{(n-1)\alpha}ds = \frac{L^n}{\Gamma((n-1)\alpha+1)\Gamma(\alpha)}\|\phi_1-\phi_2\|\int_0^1(t-a-(t-a)u)^{\alpha-1}((t-a)u)^{(n-1)\alpha}(t-a)du = \frac{L^n}{\Gamma((n-1)\alpha+1)\Gamma(\alpha)}\|\phi_1-\phi_2\|(t-a)^{n\alpha}\int_0^1(1-u)^{\alpha-1}u^{(n-1)\alpha}du = \frac{L^n}{\Gamma((n-1)\alpha+1)\Gamma(\alpha)}\|\phi_1-\phi_2\|(t-a)^{n\alpha}\beta((n-1)\alpha+1,\alpha) = \frac{L^n}{\Gamma(n\alpha+1)}(t-a)^{n\alpha}\|\phi_1-\phi_2\|$, que lo que queríamos, donde hemos hecho el cambio $\frac{s-a}{t-a}=u$ (hemos supuesto que $a<t$, en caso contrario es trivial).

Por el principio de inducción matemática, queda completa la prueba por inducción. En particular, como hemos visto que $\|T^n\phi_1(t)-T^n\phi_2(t)\| \leq \frac{L^n}{\Gamma(n\alpha+1)}(t-a)^{n\alpha}\|\phi_1-\phi_2\| \quad \forall t\in I$

y $n \geq 1 \in \mathbb{N}$, si denotamos por $l \coloneqq b - a \in \mathbb{R}$ la longitud de I, tenemos también que $\|T^n\phi_1(t) - T^n\phi_2(t)\| \leq \frac{(Ll^\alpha)^n}{\Gamma(n\alpha+1)}\|\phi_1 - \phi_2\| \quad \forall t \in I$ y $n \geq 1 \in \mathbb{N}$.

Entonces, sea $\{a_n\} \subseteq \mathbb{R}$ dada por $a_n = \frac{(Ll^\alpha)^n}{\Gamma(n\alpha+1)} \quad \forall n \geq 1 \in \mathbb{N}$, podemos ver que $\lim_n a_n = \lim_n \frac{(Ll^\alpha)^n}{\Gamma(n\alpha+1)} \leq \lim_n \frac{(Ll^\alpha)^n}{(n\alpha)^{n\alpha}e^{-n\alpha}\sqrt{2\pi n\alpha}} = \lim_n (L(\frac{le}{n\alpha})^\alpha)^n \frac{1}{\sqrt{2\pi n\alpha}} = 0$, donde hemos usado una desigualdad para la función Γ que se puede encontrar en la referencia [2].

Por tanto, sea $k \in (0,1)$ cualquiera fijada, por definición de límite, $\exists n_0 \in \mathbb{N}$ tal que en particular $\frac{(Ll^\alpha)^{n_0}}{\Gamma(n_0\alpha+1)} < k < 1$, que es lo que queríamos. En consecuencia, combinando lo anterior con la afirmación, tenemos que $\|T^{n_0}\phi_1(t) - T^{n_0}\phi_2(t)\| \leq k\|\phi_1 - \phi_2\|$ y, tomando supremos a los dos lados de la desigualdad, por ser válida para todo $t \in I$, recordando la definición de norma en Z, concluimos que $\|T^{n_0}\phi_1 - T^{n_0}\phi_2\| \leq k\|\phi_1 - \phi_2\|$. En resumen, hemos visto que T^{n_0} es contractiva y, por tanto, por un corolario del teorema del punto fijo de Banach, podemos afirmar que $\exists! \phi \in Z$ tal que $T(\phi) = \phi \iff \phi(t) = u_0 + \frac{1}{\Gamma(\alpha)}\int_a^t (t-s)^{\alpha-1} f(s,\phi(s))ds \quad \forall t \in I$ y que ϕ es límite uniforme de los iterados de Picard. Esto, juntamente con la definición de ser solución del problema de valor inicial fraccionario de Caputo, completa la prueba. □

Entonces, podemos observar que se verifica el siguiente resultado que generaliza el de la signatura en relación con las ecuaciones diferenciales.

Teorema 8.7. *Sean $0 < \alpha \leq 1$, $X : [a,b] \longrightarrow \mathbb{R}^d$ un camino de clase C^1 y $V : \mathbb{R}^e \longrightarrow \mathbb{L}(\mathbb{R}^d, \mathbb{R}^e)$ una función lineal cualesquiera. Sea también $y \in \mathbb{R}^e$ arbitrario, entonces si $Y : [a,b] \longrightarrow \mathbb{R}^e$ es solución del problema de valor inicial fraccionario de Caputo ${}^cD^\alpha Y = V(Y)\dot{X}_s, \quad Y_a = y$, se tiene que $\forall t \in [a,b]$ el valor del iterado de Picard Y_t^n se puede expresar como una combinación lineal de los términos hasta el nivel n-ésimo de la signatura fraccionaria $S^\alpha(X)_{a,t}$ con coeficientes independientes de t. En particular, Y_t está completamente determinada por V, $S^\alpha(X)_{a,t}$ e y.*

Demostración. En efecto, antes de nada, notemos que, si denotamos $f : [a,b] \times \mathbb{R}^e \longrightarrow \mathbb{R}^e$ la función dada por $f(t,y) = V(y)\dot{X}_t \quad \forall (t,y) \in [a,b] \times \mathbb{R}^e$, tal y como ya hemos demostrado en una sección anterior del trabajo, sabemos que existe $A : [a,b] \longrightarrow \mathbb{L}(\mathbb{R}^e, \mathbb{R}^e)$ una función continua tal que $f(t,y) = A(t)y \quad \forall (t,y) \in [a,b] \times \mathbb{R}^e$. Por tanto, el problema de valor inicial fraccionario de Caputo ${}^cD^\alpha Y = V(Y)\dot{X}_s, \quad Y_a = y$ se puede reescribir como ${}^cD^\alpha Y = f(s,Y), \quad Y_a = y$, cosa que, por el teorema anterior, justifica que este problema tiene solución única y realmente los iterados de Picard Y^n convergen a ella de manera uniforme.

Dicho esto, para probar el resultado, utilizando las notaciones anteriores, procedemos por inducción en $n \in \mathbb{N}$.

Caso inicial ($n = 0$): para empezar, si $n = 0$, entonces tenemos que sea $t \in [a,b]$ cualquiera, por definición $Y_t^0 = y = y \cdot 1$, que es lo que queríamos ver, ya que recordamos que el nivel 0-ésimo de la signatura fraccionaria está formado por el elemento 1.

Caso sucesor: ahora, suponemos que el resultado es cierto para $n - 1 \geq 0$ y pasamos a probarlo para n. En efecto, sea $t \in [a,b]$ arbitrario, podemos ver que por hipótesis de inducción $\forall s \in [a,b]$ tenemos Y_s^{n-1} se puede expresar de manera que $Y_s^{n-1} = \sum_{k=0}^{n-1}\sum_{i_1,\ldots,i_k \in \{1,\ldots,d\}} \lambda_{i_1,\ldots,i_k} S^\alpha(X)_{a,s}^{i_1,\ldots,i_k}$ para algunos $\lambda_{i_1,\ldots,i_k} \in \mathbb{R}^e$. Entonces, directamente, concluimos que:

$$Y_t^n = y + \frac{1}{\Gamma(\alpha)}\int_a^t (t-s)^{\alpha-1} V(Y_s^{n-1})\dot{X}_s ds = (\text{por la linealidad de } V) =$$

$$= y + \sum_{k=0}^{n-1} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} \frac{1}{\Gamma(\alpha)} \int_a^t (t-s)^{\alpha-1} V(\lambda_{i_1,\dots,i_k}) S^\alpha(X)_{a,s}^{i_1,\dots,i_k} \dot{X}_s ds =$$

$$= y + \sum_{k=0}^{n-1} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} V(\lambda_{i_1,\dots,i_k}) \frac{1}{\Gamma(\alpha)} \int_a^t (t-s)^{\alpha-1} S^\alpha(X)_{a,s}^{i_1,\dots,i_k} \dot{X}_s ds =$$

$$y + \sum_{k=0}^{n-1} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} V(\lambda_{i_1,\dots,i_k}) \begin{pmatrix} \frac{1}{\Gamma(\alpha)} \int_a^t (t-s)^{\alpha-1} S^\alpha(X)_{a,s}^{i_1,\dots,i_k} \dot{X}_s^1 ds \\ \vdots \\ \frac{1}{\Gamma(\alpha)} \int_a^t (t-s)^{\alpha-1} S^\alpha(X)_{a,s}^{i_1,\dots,i_k} \dot{X}_s^d ds \end{pmatrix} =$$

$$= y + \sum_{k=0}^{n-1} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} V(\lambda_{i_1,\dots,i_k}) \begin{pmatrix} S^\alpha(X)_{a,t}^{i_1,\dots,i_k,1} \\ \vdots \\ S^\alpha(X)_{a,t}^{i_1,\dots,i_k,d} \end{pmatrix} =$$

(por ser $V(\lambda_{i_1,\dots,i_k}) \in \mathbb{L}(\mathbb{R}^d, \mathbb{R}^e)) = \sum_{k=0}^n \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} \mu_{i_1,\dots,i_k} S^\alpha(X)_{a,t}^{i_1,\dots,i_k}$ para algunos $\mu_{i_1,\dots,i_k} \in \mathbb{R}^e$ independientes de t y $\mu_0 = y$. Esto resuelve el caso sucesor.

En conjunto, por el principio de inducción matemática, queda probado el teorema. □

8.2. Signatura fraccionaria discreta

A diferencia del resultado anterior, el autor de este trabajo no ha sido capaz de probar que la signatura fraccionaria de un camino verifique una propiedad análoga o que generalice la identidad de Chen, válida para la signatura. Esto dificulta trabajar con la signatura fraccionaria y aplicarla al *machine learning*. Como nuestro objetivo con la introducción de esta nueva signatura era mejorar los resultados obtenidos en esta aplicación práctica, pero hemos visto que no es viable utilizar directamente la signatura fraccionaria, buscamos una solución para esquivar este problema. Para ello, proponemos una nueva generalización de la signatura para caminos lineales a trozos, a medio camino entre la signatura y la signatura fraccionaria.

Definición 8.8 (Signatura fraccionaria discreta de un camino lineal a trozos)**.** *Sea* $X : [0, n-1] \longrightarrow \mathbb{R}^d$ *un camino lineal a trozos, es decir, existen puntos* $(A_i^1, \dots, A_i^d) \in \mathbb{R}^d \quad \forall i \in \{0, \dots, n-1\}$ *de manera que* $X_t^j = A_i^j(i+1-t) + A_{i+1}^j(t-i) \quad \forall i \in \{0, \dots, n-2\}$ *y* $t \in [i, i+1]$*. Sea* $\alpha > 0 \in \mathbb{R}$ *un parámetro, y sea también* $I \in W$ *cualquiera,* $I = (i_1, \dots, i_k)$*, podemos definir recursivamente las expresiones* ${}_d\tilde{S}^\alpha(X)_{a,b}^I = {}_d\tilde{S}^\alpha(X)_{a,b}^{i_1,\dots,i_k}$ *para* $0 \le a < b \le n-1 \in \mathbb{N}$ *y* $d \in \mathbb{R}$ *con* $d \ge b$ *de la siguiente manera:*

Si $b = a+1$*, entonces:*

$${}_d\tilde{S}^\alpha(X)_{a,b}^I \coloneqq \frac{\prod_{j=1}^k (A_b^{i_j} - A_a^{i_j})}{\Gamma(\alpha)\Gamma(1+(k-1)\alpha)} \left(\frac{(d-a)^\alpha}{b-a}\right)^k \beta_{\frac{b-a}{d-a}}((k-1)\alpha + 1, \alpha)$$

donde $\beta_z(x,y) = \int_0^z t^{x-1}(1-t)^{y-1} dt$ *para* $z \in (0,1]$ *y* $x, y > 0$ *denota la función beta incompleta y donde notemos que las diferencias* $b-a$ *de la expresión anterior son* 1*.*

Si $b > a+1$*, entonces tenemos que:*

$${}_d\tilde{S}^\alpha(X)_{a,b}^I \coloneqq {}_d\tilde{S}^\alpha(X)_{a,h}^I + {}_u\tilde{S}^\alpha(X)_{a,h}^{i_1,\dots,i_{k-1}} \cdot {}_d\tilde{S}^\alpha(X)_{h,b}^{i_k} + \dots + {}_u\tilde{S}^\alpha(X)_{a,h}^{i_1,\dots,i_r} \cdot {}_d\tilde{S}^\alpha(X)_{h,b}^{i_{r+1},\dots,i_k}$$

$$+ \dots + {}_d\tilde{S}^\alpha(X)_{h,b}^I$$

donde $u = \frac{b+h}{2}$ *y* h *es el menor natural mayor o igual a* $\frac{a+c}{2}$.

Además, con las notaciones anteriores, podemos definir la signatura fraccionaria discreta de X *de orden* α *como la sucesión* $\tilde{S}^{\alpha}(X)_{0,n-1} = \{_{n-1}\tilde{S}^{\alpha}(X)^{I}_{0,n-1}\}_{I\in W}$, *donde por convenio añadimos 1 como elemento inicial correspondiente a la palabra de 0 letras.*

Notemos que, en este caso, la signatura fraccionaria discreta verifica ya por definición un resultado análogo a la identidad de Chen, cosa que era nuestro objetivo y que nos permitirá aplicar esta signatura en el campo del *machine learning*. En todo lo que sigue, entenderemos por camino lineal a trozos una aplicación como la explicitada en la definición anterior.

Con esta definición, podemos probar fácilmente que para caminos lineales a trozos la signatura fraccionaria discreta es en efecto una generalización de la signatura.

Proposición 8.9. *Sea* $X : [0, n-1] \longrightarrow \mathbb{R}^d$ *un camino lineal a trozos arbitrario, entonces se verifica que* $\tilde{S}^1(X)_{0,n-1} = S(X)_{0,n-1}$, *es decir, para todo* $I \in W$, $I = (i_1, \ldots, i_k)$, *se tiene que* $_{n-1}\tilde{S}^1(X)^I_{0,n-1} = S(X)^I_{0,n-1}$. *Por tanto, para caminos lineales a trozos la signatura fraccionaria discreta es una generalización de la signatura.*

Demostración. En efecto, continuando con las notaciones arbitrarias ya introducidas en el enunciado, como $X : [0, n-1] \longrightarrow \mathbb{R}^d$ es un camino lineal a trozos, sabemos que existen puntos $(A^1_i, \ldots, A^d_i) \in \mathbb{R}^d \quad \forall i \in \{0, \ldots, n-1\}$ de manera que $X^j_t = A^j_i(i+1-t) + A^j_{i+1}(t-i) \quad \forall i \in \{0, \ldots, n-2\}$ y $t \in [i, i+1]$. Entonces, directamente por definición, podemos recordar que los términos de $\tilde{S}^1(X)_{0,n-1} = \{_{n-1}\tilde{S}^1(X)^I_{0,n-1}\}_{I\in W}$ vienen descritos por la recurrencia que tiene como caso base:

Si $b = a + 1$, entonces:

$$_d\tilde{S}^1(X)^I_{a,b} = \frac{\prod_{j=1}^k (A_b^{i_j} - A_a^{i_j})}{\Gamma(k)} (\frac{d-a}{b-a})^k \beta_{\frac{b-a}{d-a}}(k, 1) = \frac{\prod_{j=1}^k (A_b^{i_j} - A_a^{i_j})}{(k-1)!} (\frac{d-a}{b-a})^k \frac{(b-a)^k}{k(d-a)^k}$$

$$= \frac{\prod_{j=1}^k (A_b^{i_j} - A_a^{i_j})}{k!} = S(X)^I_{a,b}$$

Por tanto, notemos que en el caso base de la recurrencia, cuando $b = a + 1$, el valor del término correspondiente de la signatura fraccionaria discreta no depende de d, del parámetro que aparece abajo a la izquierda. En consecuencia, como para calcular un término de esta signatura al final siempre nos acabamos reduciendo a este caso base, podemos ignorar en todo momento el parámetro que aparece abajo a la izquierda de la notación, también cuando planteamos la fórmula para $b > a + 1$.

Observamos entonces que la fórmula para $b > a+1$ pasa a ser exactamente la identidad de Chen, en un caso particular, de la signatura original, cosa que junto con el hecho de que en el caso base los términos de las dos signaturas coinciden, nos permite afirmar que las dos signaturas tienen todos sus elementos iguales, $\tilde{S}^1(X)_{0,n-1} = S(X)_{0,n-1}$. □

Damos ahora un pequeño ejemplo donde comparamos los primeros niveles de las tres signaturas para un camino lineal a trozos.

Ejemplo 8.10. En este caso, sea $X : [0, 4] \longrightarrow \mathbb{R}^3$ el camino lineal a trozos dado por interpolación lineal a partir de los puntos $A^0 = (0, 0, 0)$, $A^1 = (2, 7, 4)$, $A^2 = (1, 5, 2)$, $A^3 =$

$(4,5,1)$, $A^4=(6,1,5)$, entonces se puede observar que calculando las signaturas indicadas hasta el nivel 2 y redondeando al primer decimal tenemos el siguiente resultado. De manera excepcional, separamos los términos de las signaturas por barras verticales y no por comas para evitar confusiones con la coma decimal.

$$S(X)_{0,4} = (1|6|1|5|18| - 16|18{,}5|22|0{,}5|9|11{,}5| - 4|12{,}5| \dots)$$

$$S^{0,9}(X)_{0,4} \approx (1|5{,}6| - 0{,}1|4{,}9|13{,}8| - 22{,}4|15{,}1|9{,}0| - 17{,}2| - 7{,}3|5{,}7| - 17{,}4|5{,}4| \dots)$$

$$\tilde{S}^{0,9}(X)_{0,4} \approx (1|5{,}6| - 0{,}1|4{,}9|17{,}3| - 15{,}8|19{,}0|16{,}9|5{,}0|7{,}4|10{,}4| - 4{,}2|14{,}1| \dots)$$

$$S^{1,1}(X)_{0,4} \approx (1|6{,}4|2{,}2|5{,}2|7{,}9| - 38{,}5|4{,}2|1{,}0| - 80{,}9| - 38{,}2| - 2{,}5| - 49{,}3| - 18{,}0| \dots)$$

$$\tilde{S}^{1,1}(X)_{0,4} \approx (1|6{,}4|2{,}2|5{,}2|18{,}6| - 16{,}2|17{,}7|27{,}3| - 4{,}2|9{,}7|12{,}9| - 4{,}3|10{,}8| \dots)$$

Notemos que, en el ejemplo anterior, el primer nivel de la signatura fraccionaria coincide con el de la fraccionaria discreta. Esto es un hecho general, que podemos incluir en la siguiente proposición, que nos justifica que hay una cierta similitud entre estas dos signaturas.

Proposición 8.11. *Sea $X:[0,n-1]\longrightarrow\mathbb{R}^d$ un camino lineal a trozos cualquiera y sea $\alpha>0$ un parámetro fijado, entonces se verifica que ${}_{n-1}\tilde{S}^{\alpha}(X)^j_{0,n-1}=S^{\alpha}(X)^j_{0,n-1}\quad\forall j\in\{1,\dots,d\}$, es decir, el primer nivel de las dos signaturas coincide.*

Además, sea $Y:[0,1]\longrightarrow\mathbb{R}^d$ un camino lineal cualquiera entre los puntos A_0, $A_1\in\mathbb{R}^d$, es decir, $Y^i_t=A^i_0(1-t)+A^i_1t\quad\forall t\in[0,1]$ e $i\in\{1,\dots,d\}$, entonces se tiene que $\tilde{S}^{\alpha}(Y)_{0,1}=S^{\alpha}(Y)_{0,1}$, las dos signaturas coinciden.

Demostración. En efecto, continuando con las notaciones ya introducidas en el enunciado, sea $j\in\{1,\dots,d\}$ arbitrario y usando que $X:[0,n-1]\longrightarrow\mathbb{R}^d$ es un camino lineal a trozos, es decir, existen puntos $(A^1_i,\dots,A^d_i)\in\mathbb{R}^d\quad\forall i\in\{0,\dots,n-1\}$ de manera que $X^j_t=A^j_i(i+1-t)+A^j_{i+1}(t-i)\quad\forall i\in\{0,\dots,n-2\}$ y $t\in[i,i+1]$, pasamos a ver que, por un lado: ${}_{n-1}\tilde{S}^{\alpha}(X)^j_{0,n-1}=$ (tomando h como el menor natural mayor o igual a $\frac{n-1}{2}$)) $={}_{n-1}\tilde{S}^{\alpha}(X)^j_{0,h}+{}_{n-1}\tilde{S}^{\alpha}(X)^j_{h,n-1}=$ (iterativamente) $=\sum_{i=0}^{n-2}{}_{n-1}\tilde{S}^{\alpha}(X)^j_{i,i+1}=\sum_{i=0}^{n-2}\frac{(A^j_{i+1}-A^j_i)}{\Gamma(\alpha)}(n-1-i)^{\alpha}\beta_{\frac{1}{n-1-i}}(1,\alpha)=\sum_{i=0}^{n-2}\frac{(A^j_{i+1}-A^j_i)}{\Gamma(\alpha)}(n-1-i)^{\alpha}\int_0^{\frac{1}{n-1-i}}(1-t)^{\alpha-1}dt=\sum_{i=0}^{n-2}\frac{(A^j_{i+1}-A^j_i)}{\Gamma(\alpha)}(n-1-i)^{\alpha}(-\frac{(1-t)^{\alpha}}{\alpha}|_0^{\frac{1}{n-1-i}}=\sum_{i=0}^{n-2}\frac{(A^j_{i+1}-A^j_i)}{\Gamma(\alpha+1)}(n-1-i)^{\alpha}(1-(1-\frac{1}{n-1-i})^{\alpha})=\frac{1}{\Gamma(\alpha+1)}\sum_{i=0}^{n-2}(A^j_{i+1}-A^j_i)((n-1-i)^{\alpha}-(n-2-i)^{\alpha})$.

Además, por el otro lado, también podemos ver que la signatura fraccionaria verifica que $S^{\alpha}(X)^j_{0,n-1}=\frac{1}{\Gamma(\alpha)}\int_0^{n-1}(n-1-t)^{\alpha-1}\dot{X}^j_t dt=\frac{1}{\Gamma(\alpha)}\sum_{i=0}^{n-2}\int_i^{i+1}(n-1-t)^{\alpha-1}(A^j_{i+1}-A^j_i)dt=\frac{1}{\Gamma(\alpha)}\sum_{i=0}^{n-2}(A^j_{i+1}-A^j_i)(-\frac{(n-1-t)^{\alpha}}{\alpha}|_i^{i+1}=\frac{1}{\Gamma(\alpha+1)}\sum_{i=0}^{n-2}(A^j_{i+1}-A^j_i)((n-1-i)^{\alpha}-(n-2-i)^{\alpha})$, que es exactamente lo que queríamos observar, cosa que completa la prueba de esta primera parte del enunciado.

A partir de aquí, respecto a la segunda afirmación, sea $I=(i_1,\dots,i_k)\in W$ un multiíndice arbitrario, será suficiente probar que ${}_1\tilde{S}^{\alpha}(Y)^I_{0,1}=S^{\alpha}(Y)^I_{0,1}$. Todo y eso, como en este caso $1=1+0$, para la signatura fraccionaria discreta, ${}_1\tilde{S}^{\alpha}(Y)^I_{0,1}$, nos encontramos en el caso base de la recurrencia, de manera que podemos afirmar directamente que:

$${}_1\tilde{S}^{\alpha}(Y)^I_{0,1}=\frac{\prod_{j=1}^k(A_1^{i_j}-A_0^{i_j})}{\Gamma(\alpha)\Gamma(1+(k-1)\alpha)}(\frac{(1-0)^{\alpha}}{1-0})^k\beta_{\frac{1-0}{1-0}}((k-1)\alpha+1,\alpha)=$$

$$\frac{\prod_{j=1}^{k}(A_1^{i_j} - A_0^{i_j})}{\Gamma(\alpha)\Gamma(1+(k-1)\alpha)}\beta((k-1)\alpha+1,\alpha) = S^{\alpha}(Y)_{0,1}^{I}$$

□

8.3. Aplicación de la signatura fraccionaria discreta

Una vez introducida esta nueva signatura fraccionaria discreta, podemos aplicarla, en el contexto del *machine learning*, al problema de reconocimiento de dígitos manuscritos, que ya hemos detallado en la sección anterior. En esta ocasión, dada $X^m = \{X_i^m\}_{i=0}^{783}$ una secuencia de valores correspondiente a una imagen de un dígito, tal y como ya lo hicimos en la implementación original, le añadiremos al valor de cada píxel su posición en la imagen, obteniendo la nueva secuencia $Y^m = \{Y_i^m\}_{i=0}^{783}$ donde $\forall i \in \{0,\ldots,783\}$ se tiene que $Y_i^m = (x, y, X_i^m)$ de manera que $i = 28x+y$ vienen dados por la división entera. Estas son las secuencias que interpolaremos linealmente a trozos en el intervalo $[0, 783]$ con sus puntos equidistantes y para las que calcularemos sus signaturas fraccionarias discretas.

A continuación, se muestran los resultados obtenidos. Por un lado, tomando la signatura fraccionaria discreta hasta el nivel 4, podemos calcular para diversos valores del parámetro α la tasa de acierto alcanzada, resumiendo todos los cálculos en la gráfica de la figura 12. Recordemos que, para $\alpha = 1$, recuperamos la tasa de acierto de la signatura.

Gracias a estas deducciones, vemos que, para este problema con la signatura fraccionaria discreta truncada en el cuarto nivel, el valor $\alpha = 1{,}15$ es el que da mejores resultados. Esto nos lleva a calcular, para este valor del parámetro, las tasas de acierto correspondientes a los niveles 4, 5, 6 y 7 de la signatura fraccionaria discreta, tal y como ya lo habíamos hecho para la signatura. Comparamos los resultados obtenidos en el cuadro 2.

Tasas		Signatura utilizada	
		Signatura	S. f. d. $\alpha = 1{,}15$
Nivel	4	0,647	0,873
	5	0,838	0,906
	6	0,881	0,924
	7	0,906	0,935

Cuadro 2: Comparación de tasas de acierto

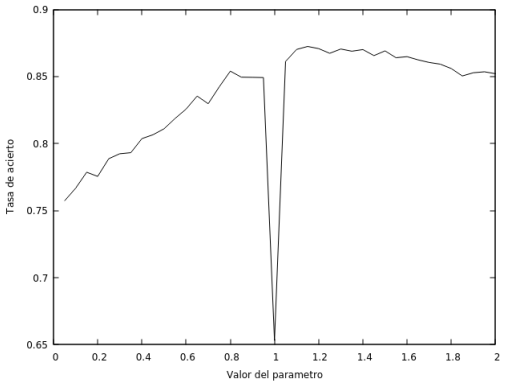


Figura 12: Parámetro - Tasa de acierto

Podemos ver, en primer lugar, que cuando las truncamos al cuarto nivel, los resultados obtenidos por la signatura fraccionaria discreta son significativamente mejores que los de la signatura original. Además, la signatura fraccionaria discreta también ha superado con un cierto margen a la original en niveles más altos. Esto nos lleva a pensar que quizás lo mismo sea cierto para otros problemas de *machine learning* y, por tanto, que la signatura fraccionaria discreta pueda llegar a ser una buena alternativa para la signatura.

El código que calcula estas signaturas fraccionarias discretas hasta el nivel 7 se puede encontrar en el anexo del trabajo o en la carpeta de *Google Drive* asociada al enlace:

`https://drive.google.com/drive/folders/1F_HAwIAk1jClMwPksnf3YMgpNJbLvU4I?usp=drive_link`

9. Conclusiones

En este trabajo hemos repasado las principales propiedades de la signatura de caminos, desde las más sencillas a otras más elaboradas, dando en cada caso las demostraciones correspondientes. También hemos mostrado cómo la signatura de un camino es capaz de describir de manera precisa sus propiedades y, por tanto, es una herramienta a la que podemos recurrir para el estudio de los caminos. Con este objetivo, hemos analizado la relación entre ciertos valores que aparecen en la signatura de un camino con algunas de sus áreas asociadas y, también, hemos mostrado cómo la signatura aparece de manera natural al considerar ciertas ecuaciones diferenciales controladas y sus soluciones.

Además, hemos analizado cómo la signatura tiene hoy en día una aplicación práctica en el campo del *machine learning*. Primero, nos hemos centrado, desde un punto de vista teórico y demostrando algunos de los resultados comentados, en estudiar las posibles transformaciones que se aplican a las secuencias de datos para obtener caminos que se puedan describir mediante su signatura. Después, hemos dado una aplicación concreta para mostrar en acción lo estudiado en esta segunda parte del trabajo, utilizando la signatura para tratar el problema de reconocimiento de dígitos manuscritos.

En relación con los resultados obtenidos durante la aplicación de la signatura al problema de *machine learning*, estos nos han permitido ejemplificar con éxito como esta y también su variante dada por la signatura logarítmica se pueden combinar con modelos de *machine learning* para obtener predicciones aceptablemente buenas. Además, nos han permitido resaltar la importancia de elegir un buen *embedding* para las secuencias de datos estudiadas, ya que hemos visto que según la transformación escogida, la precisión obtenida en los resultados ha variado significativamente. Por tanto, concluimos que, para aplicar la signatura a un problema del campo del *machine learning*, es importante estudiarlo con detalle previamente para decidir si la signatura será capaz de dar buenos resultados y con qué *embedding* se podrán alcanzar. Para ello, es necesario que estas transformaciones puedan exponer todo lo posible la estructura geométrica de las secuencias datos.

Aparte, también hemos introducido la signatura fraccionaria y la signatura fraccionaria discreta, observando que la primera generaliza el resultado para las ecuaciones diferenciales que verificaba la signatura original y que la segunda puede aplicarse para dar solución a problemas de *machine learning*. Centrándonos en los resultados obtenidos al aplicar la signatura fraccionaria discreta al problema de reconocimiento de dígitos manuscritos, estos han sido significativamente mejores que los dados por la signatura, muy especialmente cuando se toman signaturas truncadas a bajo nivel. Por tanto, concluimos que la signatura fraccionaria discreta es una buena alternativa que se puede considerar cuando se estudian problemas de *machine learning* donde es coherente aplicar la signatura original, en particular cuando estas se truncan a bajo nivel, posiblemente con el objetivo de reducir el número de atributos del modelo.

Todo y eso, aunque los resultados obtenidos en el mejor de los casos no son malos, hemos visto y existen en la literatura numerosos modelos alternativos de *machine learning* que, aplicados al mismo problema de reconocimiento de dígitos manuscritos, han alcanzado precisiones superiores [14]. Atribuimos este hecho a que este problema ha sido muy estudiado y a que nuestro objetivo original era simplemente ejemplificar cómo se pueden aplicar las signaturas al *machine learning* y, por tanto, no nos hemos centrado en buscar la configuración óptima que nos permitiese, utilizando la signatura o la signatura fraccionaria discreta, obtener los mejores resultados posibles.

Referencias

[1] Al-Abedeen, A. Z.; Arora, H. L.: A Global Existence and Uniqueness Theorem for Ordinary Differential Equations of Generalized Order. *Canad. Math. Bull.*, 21(3):267-271, 1978.

[2] Batir, N.: Inequalities for the gamma function. *Arch. Math. (Basel)*, 91(6):554-563, 2008.

[3] Chen, K.: Integration of Paths–A Faithful Representation of Paths by Noncommutative Formal Power Series, *Trans. Amer. Math. Soc.*, 89(2):395-407, 1958.

[4] Chen, K.: Integration of Paths, Geometric Invariants and a Generalized Baker-Hausdorff Formula, *Ann. of Math.*, 65(1):163-178, 1957.

[5] Chen, K.: Iterated Integrals and Exponential Homomorphisms, *Proc. Lond. Math. Soc.*, s3-4(1):502-512, 1954.

[6] Chevyrev, I.; Kormilitzin, A.: A Primer on the Signature Method in Machine Learning,
`arXiv:1603.03788 [stat.ML]`, marzo de 2016.

[7] Dagum, L.; Menon, R.: OpenMP: an industry standard API for shared-memory programming, *IEEE Comput. Sci. Eng.*, 5(1):46-55, 1998.

[8] do Carmo, M. P.: *Differential geometry of curves and surfaces*, Englewood Cliffs, New Jersey : Prentice-Hall, 1976.

[9] Ebrahimi-Fard, K.; Patras, F.; Tapia, N.; Zambotti, L.: Shifted Substitution in Non-Commutative Multivariate Power Series with a View Toward Free Probability, *SIGMA Symmetry Integrability Geom. Methods Appl.*, 19(38):17 pages, 2023.

[10] Galassi, M. et al: GNU Scientific Library Reference Manual (3.ª Ed.),
`http://www.gnu.org/software/gsl/`

[11] Gyurkó, L. G.; Lyons, T.; Kontkowski, M.; Field, J.: Extracting information from the signature of a financial data stream,
`arXiv:1307.7244v2 [q-fin.ST]`, julio de 2014.

[12] Hohenwarter, M.; Borcherds, M.; Ancsin, G.; Bencze, B.; Blossier, M.; Delobelle, A.; Denizet, C.; Éliás, J.; Fekete, Á; Gál, L.; Konečný, Z.; Kovács, Z.; Lizelfelner, S.; Parisse, B.; Sturr, G.: *GeoGebra 4.4*, `http://www.geogebra.org`, diciembre de 2013.

[13] Hohenwarter, M.: *GeoGebra: Ein Softwaresystem für dynamische Geometrie und Algebra der Ebene*. Tesis de Máster, Paris Lodron Universität, Salzburg, Austria, febrero de 2002. (en alemán).

[14] LeCun, Y.; Cortes, C.; Burges, C.J.C. : THE MNIST DATABASE of handwritten digits, Accedido a 11/3/2024 en `http://yann.lecun.com/exdb/mnist/`, 2010.

[15] Lyons, R.; Vatsala, A. S.; Chiquet, R. A.: Picard's Iterative Method for Caputo Fractional Differential Equations with Numerical Results, *Mathematics*, 5(4):65, 2017.

[16] Lyons, T.; McLeod, A. D.: Signature Methods in Machine Learning, `arXiv:2206.14674v4 [stat.ML]`, enero de 2023.

[17] Lyons, T.; Picard, J.; Lévy, T.: *Differential equations driven by rough paths: Ecole d'été de Probabilités de Saint-Flour XXXIV-2004*, Springer Berlin / Heidelberg, Berlin, 2007.

[18] Müger, M.: *Notes on the theorem of Baker-Campbell-Hausdorff-Dynkin*, Radboud Universiteit, 2019.

[19] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E.: Scikit-learn: Machine Learning in Python, *J. Mach. Learn. Res.*, 12:2825-2830, 2011.

[20] Ree, R.: Lie Elements and an Algebra Associated With Shuffles, *Ann. of Math.*, 68(2):210-220, 1958.

[21] Reizenstein, J.; Graham, B.: The iisignature library: efficient calculation of iterated-integral signatures and log signatures, `arXiv:1802.08252 [cs.DS]`, febrero de 2018.

[22] Sotomayor, J. M.: *Lições de equações diferenciais ordinárias*, Instituto de Matemática Pura e Aplicada, Río de Janeiro, 1979.

Anexo

Notemos que, tal y como ya lo hemos indicado en el trabajo, también se puede acceder al código de los programas que aparecen en este anexo en la carpeta de *Google Drive* asociada al enlace:

https://drive.google.com/drive/folders/1F_HAwIAk1jClMwPksnf3YMgpNJbLvU4I?usp=drive_link

Programa 1. Preparación de datos usando la transformación *Lead-Lag*

```
from iisignature import sig, prepare, logsig
import numpy as np
import pandas as pd
from sklearn.datasets import fetch_openml

# En este caso, empezamos importando las imágenes de los dígitos
    manuscritos con sus etiquetas correspondientes, que guardamos en la
    variable llamada mnist. Entonces, como para este programa no
    necesitamos las etiquetas asociadas a las imágenes, las descartamos y
    tomamos solo los datos de las imágenes que guardamos en data.

mnist = fetch_openml('mnist_784', parser = "pandas")
data = mnist.data

# Una vez importados estos datos, por comodidad, cambiamos el nombre de
    las columnas de las tablas que los contienen y los imprimimos por
    pantalla, para poder comprobar que no se ha producido ningún error y
    que lo mostrado se corresponde con los datos introducidos.

lista = []

for i in range(784):
    lista.append(str(i))

data.columns = lista

print(data)

# A partir de aquí, dividimos los datos para separar las secuencias
    correspondientes a los de entrenamiento y a los de comprobación,
    obteniendo un total de 2 nuevas variables donde los guardamos.

X_train = data[:60000]
X_test = data[60000:]

# Entonces, antes de nada, para construir el embedding con la transformaci
    ón Lead-Lag, doblamos el número de columnas de la tabla de datos
    repitiendo los valores, cosa que posteriormente nos permitirá desplazar
     cada fila sobre ella misma para obtener la transformación Lead-Lag.
    Notemos que a partir de este punto en el código, tenemos que decidir si
     hacemos los cálculos para los datos de entrenamiento o de comprobación
    , sustituyendo en la siguiente línea de código la variable X_train por
    X_test o al revés.

digitos = pd.DataFrame(np.repeat(X_test.values, 2, axis = 1))
print(digitos)

# Ahora, utilizando un bucle que recorre cada una de las filas de la nueva
     tabla de datos, para cada una de ellas, la consideramos dos veces,
```

```
    desplazando una de las copias y poniéndolas en común para obtener la
    transformación Lead-Lag de la secuencia. Posteriormente, según si
    tenemos la parte 1 o la parte 2 comentadas, le aplicaremos la signatura
     o signatura logarítmica y guardaremos el resultado en una lista.

lista = []
for val in range(len(digitos)):
    lag = digitos.iloc[val]
    lag.drop(lag.index[-1], inplace = True)
    lead = digitos.iloc[val]
    lead.drop(lead.index[0], inplace = True)
    aux = np.c_[lead, lag]
    depth = 5

#-----------------------------------------------
# Parte 1: Signatura logarítmica

    # s = prepare(2, depth)
    # sign = logsig(aux, s)

#-----------------------------------------------
# Parte 2: Signatura

    sign = sig(aux, depth, 0)

#-----------------------------------------------

    aux = sign.tolist()
    dicc = {str(i): [aux[i]] for i in range(len(aux))}
    aux = pd.DataFrame(dicc)

    lista.append(aux)
    print(val)

# Finalmente, ponemos en común todas las signaturas calculadas previamente
     para construir una nueva tabla de datos, la imprimimos por pantalla
    para comprobar que el resultado es el esperado y no se han producido
    errores y guardamos el archivo en el directorio en el que estemos
    trabajando para utilizarlo posteriormente, dándole el nombre que
    queramos.

digitos = pd.concat(lista)
print(digitos)
digitos.to_csv('mnist_X_test_refi_1_5_t.csv', index = False)
```

Programa 2. Preparación de datos adjuntando a cada píxel su posición

```
from iisignature import sig, prepare, logsig
import numpy as np
import pandas as pd
from sklearn.datasets import fetch_openml

#Para empezar, cargamos los datos correspondientes a las imágenes de los dígitos manuscritos, que contienen tanto los de entrenamiento como los de comprobación, sin etiquetas, y los guardamos en la variable data.

mnist = fetch_openml('mnist_784', parser = "pandas")
data = mnist.data

# Posteriormente, renombramos sus columnas para hacer más cómoda la manipulación de los datos e imprimimos por pantalla el resultado, mostrando un fragmento de la tabla de valores, de manera que cada una de sus filas se corresponde con una imagen.

lista = []

for i in range(784):
    lista.append(str(i))

data.columns = lista

print(data)

# A continuación, separamos las secuencias de datos que se corresponden con los de entrenamiento y los de comprobación. Utilizamos para ello un total de 2 nuevas variables, llamadas X_train y X_test respectivamente.

X_train = data[:60000]
X_test = data[60000:]

# A partir de aquí, preparamos la secuencia 2-dimensional formada por valores entre 0 y 27 incluidos que nos indicarán la posición de cada píxel en la imagen, cosa que tiene como objetivo exponer la estructura geométrica de los datos y que pueda ser aprovechada por la signatura.

lista = []
lista1 = []
for i in range(28):
    for j in range(28):
        lista.append(float(i))
        lista1.append(float(j))

serx = pd.Series(lista)
sery = pd.Series(lista1)
lista = []

# Ahora, recorremos todas las filas de la tabla con los datos y, para cada una de ellas, le adjuntamos a sus píxeles su posición para obtener una secuencia de datos tridimensionales a la que le calculamos entonces su signatura o signatura logarítmica. Notemos que para aplicar esta transformación a los datos de entrenamiento o a los de comprobación, tendremos que sustituir en la siguiente línea del código la variable X_train por X_test o al revés según convenga.
digitos = X_test

```

```
for val in range(len(digitos)):
    sec = digitos.iloc[val]
    aux = np.c_[serx, sery, sec]
    depth = 5

# En este punto, para decidir si utilizar la signatura o la signatura
    logarítmica, tenemos que dejar comentada una de las dos partes
    indicadas, aquella que no queramos utilizar.

# ----------------------------------
# Parte 1, correspondiente a la signatura logarítmica.

    s = prepare(3, depth)
    sign = logsig(aux, s)

# ----------------------------------
# Parte 2, correspondiente a la signatura.

  # sign = sig(aux, depth, 0)

# ----------------------------------

    aux = sign.tolist()
    dicc = {str(i): [aux[i]] for i in range(len(aux))}
    aux = pd.DataFrame(dicc)
    lista.append(aux)
    print(val)

# Finalmente, reunimos todas las signaturas calculadas en una tabla de
    valores por filas, la mostramos por pantalla para poder observar si el
    resultado es el deseado y la guardamos en el directorio en el que
    estemos trabajando para poder usarla posteriormente, con el nombre que
    queramos.

digitos = pd.concat(lista)
print(digitos)
digitos.to_csv('mnist_X_test_refi_4_5.csv', index = False)
```

Programa 3. Construcción de los modelos de *machine learning* y cálculo de su tasa de acierto

```
import numpy as np
import pandas as pd
from sklearn.ensemble import HistGradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.datasets import fetch_openml

# En este caso, para empezar, importamos los datos que queremos utilizar para construir el modelo de machine learning y ponerlo a prueba. Más precisamente, de los archivos creados con alguno de los programas anteriores, elegimos uno en sus dos versiones de entrenamiento y comprobación, que será para el que se realice la construcción del modelo. Además, importamos los datos originales de los que extraeremos las etiquetas con los dígitos reales.

X_train = pd.read_csv('mnist_X_train_refi_3_7.csv')
X_test = pd.read_csv('mnist_X_test_refi_3_7.csv')

mnist = fetch_openml('mnist_784', parser = "pandas")

# Una vez cargados estos datos, extraemos las etiquetas con los dígitos reales y las guardamos en las variables y_train e y_test. Después, imprimimos por pantalla algunos de los datos que estamos manipulando para ver que todo funciona adecuadamente.

y_train = mnist.target[:60000]
y_test = mnist.target[60000:]

print(X_train)
print(X_test)

# Ahora, como los datos originales de las signaturas contienen valores muy grandes y poco convenientes para aplicarles un modelo de machine learning, pasamos a estandarizar los valores columna a columna. Por tanto, para cada columna, calculamos su media aritmética y su desviación típica y realizamos la estandarización. En el caso de que la desviación típica sea 0, como todos los valores de la columna serán iguales, la eliminaremos porque no aportaría nada a la clasificación.

#Hacemos esto tanto para los datos de entrenamiento como para los de comprobación, observando que las medias y desviaciones típicas se calculan solo para los datos de entrenamiento y se aplican a los dos, ya que en la práctica solo tendríamos primero los datos de entrenamiento y nos tendríamos que guiar por estos para estandarizar futuros datos.

for col in range(len(X_train.columns)):
    val = X_train[str(col)].mean()
    val2 = X_train[str(col)].std()
    if val2 != 0.0:
        aux = X_train[str(col)].map(lambda p: (p-val)/val2)
        aux2 = X_test[str(col)].map(lambda p: (p-val)/val2)
        X_train[str(col)] = aux
        X_test[str(col)] = aux2
    else:
        X_train = X_train.drop(str(col), axis = 1)
        X_test = X_test.drop(str(col), axis = 1)

```

```
38
39 lista = [str(col) for col in range(len(X_train.columns))]
40 X_train.columns = lista
41 X_test.columns = lista
42
43 # Después de llevar a cabo la estandarización de la manera indicada,
       imprimimos por pantalla los nuevos datos transformados para comprobar
       que no se ha producido ningún problema y que ahora los datos se
       encuentran en un rango de valores más razonable.
44
45 print(X_train)
46 print(X_test)
47
48 # Finalmente, con los datos anteriores, entrenamos un modelo de machine
       learning y lo ponemos a prueba, calculando su tasa de acierto con los
       datos de comprobación.
49
50 model = HistGradientBoostingClassifier(max_iter = 300, random_state = 1,
       verbose = 1)
51 model.fit(X_train, y_train)
52
53 predic = model.predict(X_test)
54 print(accuracy_score(predic, y_test))
```

Programa 4. Construcción del modelo de *machine learning* que no usa la signatura y cálculo de su tasa de acierto

```
import numpy as np
import pandas as pd
from sklearn.ensemble import HistGradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.datasets import fetch_openml

# En este caso, para empezar, importamos los datos que queremos utilizar para construir el modelo de machine learning y ponerlo a prueba.

mnist = fetch_openml('mnist_784', parser = "pandas")

# Una vez importados estos datos, los dividimos entre datos de entrenamiento y de comprobación, separando también sus etiquetas correspondientes, generando cuatro nuevas variables tal y como se muestra. Después, imprimimos por pantalla algunos de los datos que estamos manipulando para ver que todo funciona adecuadamente.

X_train = mnist.data[:60000]
X_test = mnist.data[60000:]
y_train = mnist.target[:60000]
y_test = mnist.target[60000:]

print(X_train)
print(X_test)

# Para acabar, entrenamos el modelo de machine learning y lo ponemos a prueba, utilizando los datos anteriores y calculando posteriormente su tasa de acierto.

model = HistGradientBoostingClassifier(max_iter = 300, random_state = 1, verbose = 1)
model.fit(X_train, y_train)

predic = model.predict(X_test)
print(accuracy_score(predic, y_test))
```

Programa 5. Cálculo de la signatura fraccionaria discreta hasta el nivel 7, extrayendo los datos de un archivo y guardándolos en otro

```
/* Para que este programa funcione correctamente, entre otros, se necesita tener instalada la librería GNU Scientific Library, que se utiliza para evaluar ciertas funciones matemáticas. */

/* Este no es el único programa que hemos utilizado para estudiar y calcular la signatura fraccionaria discreta. Todo y eso, para que no quede un anexo excesivamente extenso y porque los otros programas se pueden obtener variando este de la manera adecuada, hemos decidido incluir solo este en el anexo del trabajo. */

/* El objetivo de este programa es, dado un archivo de tipo .csv que contiene por filas los valores de los 784 píxeles de cada imagen de un dígito manuscrito, considerar los caminos dados por interpolación lineal a trozos de la secuencia de valores de los píxeles junto con sus coordenadas en la imagen, tal y como se ha explicado en el trabajo, y calcular para cada uno de ellos su signatura fraccionaria discreta de parámetro alfa. Posteriormente, el programa también guarda en el formato .csv adecuado estas signaturas por filas en el archivo que se indique, para después poder aplicar los modelos de machine learning correspondientes. */

/* En el momento de compilar el programa, si se usa el compilador gcc tal y como lo ha hecho el alumno, se recomienda usar la opción -O2 para aumentar la velocidad de ejecución, ya que este es un programa que puede tardar en ejecutarse. */

/* Observamos que si se ejecuta este programa usando Valgrind para estudiar posibles problemas de memoria, puede que se obtenga que aún falta memoria por liberar. Esto no se debe a un error que le sea posible solucionar al alumno, sino que se produce por las técnicas de paralelización usadas en el programa, siendo un hecho bien conocido y esperable que no debería ser un problema. */

/* Notemos además que aunque este programa permite utilizar el valor de alfa = 1, para el que la signatura fraccionaria discreta coincide con la signatura original, no es recomendable utilizarlo, siendo preferible usar otras funciones y programas para el cálculo de la signatura original, como los que hemos usado en la parte práctica del trabajo. Esto se debe a que de manera natural e inevitable se introducen pequeños errores numéricos en el cálculo que para otros valores del parámetro son negligibles, pero para alfa = 1 provocan que existan dependencias que la signatura original no presenta. Estas pueden alterar los resultados de manera suficientemente significativa, especialmente si estandarizamos posteriormente los términos de las signaturas, como para afectar a las predicciones y tasas de acierto. */

#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<gsl/gsl_sf.h>
#include<string.h>
#include<omp.h>

void calcula_sig(int, int, int, int, double **, double ***, int **, double ***, double ****);

int main(void) {
```

```
23    char str[50000];
24    int i, a, d, k, h, j, l, comp = 3, m, n = 784, num_caminos = 10000, aux
      , auxp, lensig = 0, depth = 7;
25    int **mat;
26    double alfa = 1.15;
27    double **matsig, ***matvec, ***mat1, ****sig, ***auxvec;
28    FILE *arch;
29
30 /* Para empezar, reservamos memoria para la variable matsig, que
      utilizaremos para guardar las signaturas fraccionarias discretas de
      cada uno de los caminos que calculemos, para después imprimirlas en el
      archivo indicado. Tal y como se verá más adelante, procedemos de esta
      manera porque en este programa usaremos técnicas de paralelización para
       llevar a cabo el cálculo de las signaturas fraccionarias discretas,
      cosa que nos ha llevado a guardar en memoria las signaturas calculadas
      antes de imprimirlas para tener un mejor control de en que orden
      aparecen impresas en el archivo, que es esencial. */
31
32 /* Notemos que, después de cada reserva memoria dinámica, en el caso de
      que se produzca un error, liberamos toda la memoria que hubiéramos
      reservado previamente y cerramos todos los archivos con los que estuvié
      ramos trabajando. Esto será un procedimiento habitual a lo largo del
      programa y, por tanto, no lo resaltaremos en los comentarios que siguen
      . */
33
34    matsig = (double **) malloc(num_caminos*sizeof(double *));
35
36    if (matsig == NULL) {
37       printf ("Error en la reserva de memoria. Ponemos fin a la ejecucion\
      n");
38
39       return 1;
40    }
41
42    for (i = 1; i <= depth; i++) {
43       aux = 1;
44       for (j = 0; j < i; j++) {
45          aux *= comp;
46       }
47       lensig += aux;
48    }
49
50    for (i = 0; i < num_caminos; i++) {
51       matsig[i] = (double *) malloc(lensig*sizeof(double));
52
53       if (matsig[i] == NULL) {
54    printf ("Error en la reserva de memoria. Ponemos fin a la ejecucion\n")
      ;
55
56    for (j = 0; j < i; j++) {
57             free(matsig[j]);
58    }
59
60    free(matsig);
61
62    return 1;
63       }
64    }
65
66 /* A partir de aquí, abrimos el archivo indicado de donde se extraerán las
       secuencias de datos correspondientes a cada una de las imágenes de los
```

```
    dígitos manuscritos. Reservamos también memoria para la variable mat,
   que utilizaremos a continuación para guardar estas secuencias. */

   arch = fopen("datos_test.csv", "r");

   if (arch == NULL) {
      for (i = 0; i < num_caminos; i++) {
   free(matsig[i]);
      }

      free(matsig);
      printf ("Error al abrir un archivo. Ponemos fin a la ejecucion\n");

      return 1;
   }

   mat = (int **) malloc(num_caminos*sizeof(int *));

   if (mat == NULL) {

      for (i = 0; i < num_caminos; i++) {
         free(matsig[i]);
      }

      free(matsig);
      fclose(arch);

      printf("Error en la reserva de memoria. Ponemos fin a la ejecucion\n
   ");

      return 1;
   }

   for (i = 0; i < num_caminos; i++) {
      mat[i] = (int *) malloc(n*sizeof(int));

      if (mat[i] == NULL) {
   for (k = 0; k < i; k++) {
      free(mat[k]);
   }

   for (k = 0; k < num_caminos; k++) {
            free(matsig[k]);
        }

   free(matsig);
   free(mat);
   fclose(arch);

   printf ("Error en la reserva de memoria. Ponemos fin a la ejecucion\n")
   ;

   return 1;
      }
   }


   for (i = 0; i < num_caminos; i++) {

      if (fgets(str, sizeof(str), arch) == NULL) {
   printf ("Error en la lectura del archivo. Ponemos fin a la ejecucion\n"
```

```
    );


        for (i = 0; i < num_caminos; i++) {
           free(mat[i]);
           free(matsig[i]);
        }

        free(mat);
        free(matsig);
        fclose(arch);

   return 1;
      }

      for (j = 0; j < n; j++) {
         sscanf(str, "%d,", &mat[i][j]);
         if (mat[i][j]/10 == 0) {
            memmove(str, str + 2, strlen(str));
         } else{
            if(mat[i][j]/100 == 0) {
               memmove(str, str + 3, strlen(str));
            } else{
               memmove(str, str + 4, strlen(str));
            }
         }
      }
   }

   fclose(arch);

/* Una vez leídos los datos necesarios, reservamos memoria para la
    variable matvec, que se usará para guardar cada una de las secuencias
    anteriores pero aumentada con las coordenadas de cada uno de los pí
    xeles en la imagen correspondiente. Procedemos de esta manera porque
    reutilizaremos la variable mat posteriormente y necesitamos, por tanto,
     poderla liberar (notemos que, aunque en esta aplicación no es tan
    importante, porque los valores de los píxeles son enteros, matvec es de
     tipo double y mat no, cosa que podría ser relevante al aplicar este
    programa a otros problemas). */

   matvec = (double ***) malloc(num_caminos*sizeof(double **));

   if (matvec == NULL) {
      printf ("Error en la reserva de memoria. Ponemos fin a la ejecucion\
    n");

      for (i = 0; i < num_caminos; i++) {
   free(mat[i]);
   free(matsig[i]);
      }

      free(mat);
      free(matsig);

      return 1;
   }

   for (i = 0; i < num_caminos; i++) {
      matvec[i] = (double **) malloc(comp*sizeof(double *));

```

```
175         if (matvec[i] == NULL) {
176      printf ("Error en la reserva de memoria. Ponemos fin a la ejecucion\n")
         ;
177
178      for (k = 0; k < i; k++) {
179         for (a = 0; a < comp; a++) {
180            free(matvec[k][a]);
181         }
182
183         free(matvec[k]);
184      }
185
186      free(matvec);
187      for (k = 0; k < num_caminos; k++) {
188               free(mat[k]);
189         free(matsig[k]);
190            }
191
192            free(mat);
193            free(matsig);
194
195      return 1;
196
197         }
198
199         for (j = 0; j < comp; j++) {
200            matvec[i][j] = (double *) malloc(n*sizeof(double));
201
202      if (matvec[i][j] == NULL) {
203         printf ("Error en la reserva de memoria. Ponemos fin a la ejecucion\
       n");
204
205               for (k = 0; k < i; k++) {
206                  for (a = 0; a < comp; a++) {
207                     free(matvec[k][a]);
208                  }
209
210                  free(matvec[k]);
211               }
212
213         for (a = 0; a < j; a++) {
214            free(matvec[i][a]);
215         }
216
217         free(matvec[i]);
218               free(matvec);
219               for (k = 0; k < num_caminos; k++) {
220                  free(mat[k]);
221         free(matsig[k]);
222            }
223
224               free(mat);
225               free(matsig);
226
227               return 1;
228
229            }
230         }
231      }
232
233      for (i = 0; i < num_caminos; i++) {
```

```
234         for (j = 0; j < n; j++) {
235            aux = j/28;
236            matvec[i][0][j] = (double) aux;
237            aux = j%28;
238            matvec[i][1][j] = (double) aux;
239            matvec[i][2][j] = (double) mat[i][j];
240         }
241      }
242
243
244      for (i = 0; i < num_caminos; i++) {
245         free(mat[i]);
246      }
247
248      free(mat);
249
250   /* Una vez guardadas las secuencias aumentadas utilizando la variable
       matvec y liberada la memoria asociada a mat, empezamos a prepararnos
       para llevar a cabo los cálculos de las signaturas fraccionarias
       discretas. Para ello, utilizaremos las variables mat y mat1 para
       guardar información auxiliar que nos guiará a lo largo del algoritmo de
        cálculo, indicándonos que cálculos tenemos que realizar en cada
       momento. Para obtener las signaturas fraccionarias discretas, aunque
       podríamos programar el algoritmo dado por su identidad de Chen
       generalizada, esto resulta muy lento y no es viable. Por tanto, hemos
       optado por un algoritmo no recursivo, que calcule primero las
       signaturas fraccionarias discretas de los caminos lineales
       correspondientes y que después las utilice para calcular
       progresivamente otras signaturas intermedias hasta llegar a la deseada.
        Por tanto, necesitaremos los datos auxiliares que guardamos gracias a
       mat y mat1 para guiarnos y saber qué signaturas fraccionarias discretas
        tenemos que calcular en cada paso. */
251
252      m = floor(log(n)/log(2)) + 2;
253
254
255      mat = (int **) malloc(m*sizeof(int *));
256
257      if (mat == NULL) {
258         printf ("Error en la reserva de memoria. Ponemos fin a la ejecucion\
       n");
259
260         for (i = 0; i < num_caminos; i++) {
261            for (j = 0; j < comp; j++) {
262               free(matvec[i][j]);
263            }
264
265            free(matvec[i]);
266      free(matsig[i]);
267         }
268
269         free(matsig);
270         free(matvec);
271
272         return 1;
273      }
274
275      j = 2;
276      for (i = 0; i < m; i++) {
277         mat[i] = (int *) malloc (j*sizeof(int));
278
```

```
279       if (mat[i] == NULL) {
280    printf ("Error en la reserva de memoria. Ponemos fin a la ejecucion\n")
       ;
281
282    for (a = 0; a < i; a++) {
283       free(mat[a]);
284    }
285
286    for (a = 0; a < num_caminos; a++) {
287             for (k = 0; k < comp; k++) {
288                free(matvec[a][k]);
289             }
290
291             free(matvec[a]);
292       free(matsig[a]);
293          }
294
295          free(matsig);
296          free(matvec);
297    free(mat);
298
299    return 1;
300       }
301
302       if (i == 0) {
303          mat[i][0] = 0;
304          mat[i][1] = n-1;
305       } else {
306          for (k = 0; k < j; k++) {
307             if (k%2 == 0) {
308               mat[i][k] = mat[i-1][k/2];
309             } else {
310                mat[i][k] = (mat[i-1][k/2] + mat[i-1][k/2 + 1])/2 + (mat[i
       -1][k/2] + mat[i-1][k/2 + 1])%2;
311             }
312          }
313       }
314
315       j = 2*j -1;
316    }
317
318
319    j = 2;
320    mat1 = (double ***) malloc (m*sizeof(double **));
321
322    if (mat1 == NULL) {
323       printf ("Error en la reserva de memoria. Ponemos fin a la ejecucion\
       n");
324
325       for (i = 0; i < m; i++) {
326          free(mat[i]);
327       }
328
329       free(mat);
330
331       for (i = 0; i < num_caminos; i++) {
332          for (l = 0; l < comp; l++) {
333             free(matvec[i][l]);
334          }
335
336          free(matvec[i]);
```

```
337         free(matsig[i]);
338      }
339
340      free(matvec);
341      free(matsig);
342
343      return 1;
344   }
345
346   for (i = 0; i < m; i++) {
347      mat1[i] = (double **) malloc(j*sizeof(double *));
348
349      if (mat1[i] == NULL) {
350         printf ("Error en la reserva de memoria. Ponemos fin a la
    ejecucion\n");
351
352   l = 2;
353   for (a = 0; a < i; a++) {
354      for (k = 1; k < l; k++) {
355         free(mat1[a][k]);
356      }
357
358      l = 2*l-1;
359      free(mat1[a]);
360   }
361
362         for (i = 0; i < m; i++) {
363            free(mat[i]);
364         }
365
366         free(mat);
367         free(mat1);
368
369         for (i = 0; i < num_caminos; i++) {
370            for (l = 0; l < comp; l++) {
371               free(matvec[i][l]);
372            }
373
374            free(matvec[i]);
375            free(matsig[i]);
376         }
377
378         free(matvec);
379         free(matsig);
380
381         return 1;
382      }
383
384      for (k = 1; k < j; k++) {
385         mat1[i][k] = (double *) malloc((i+1)*sizeof(double));
386
387
388   if (mat1[i][k] == NULL) {
389            printf ("Error en la reserva de memoria. Ponemos fin a la
    ejecucion\n");
390
391      for (l = 1; l < k; l++) {
392         free(mat1[i][l]);
393      }
394
395      free(mat1[i]);
```

```
396
397             l = 2;
398             for (a = 0; a < i; a++) {
399                for (k = 1; k < l; k++) {
400                   free(mat1[a][k]);
401                }
402
403                l = 2*l-1;
404                free(mat1[a]);
405             }
406
407             for (i = 0; i < m; i++) {
408                free(mat[i]);
409             }
410
411             free(mat);
412             free(mat1);
413
414             for (i = 0; i < num_caminos; i++) {
415                for (l = 0; l < comp; l++) {
416                   free(matvec[i][l]);
417                }
418
419                free(matvec[i]);
420                free(matsig[i]);
421             }
422
423             free(matvec);
424             free(matsig);
425
426             return 1;
427          }
428
429    for (h = 0; h < i; h++) {
430             mat1[i][k][h] = mat1[i-1][k/2 + k%2][h];
431          }
432          if (i == 0) {
433             mat1[i][k][i] = n-1;
434          } else {
435             if (k%2 == 1) {
436               mat1[i][k][i] = (mat[i][k] + mat[i][k+1])/2.;
437               if (mat1[i][k][i] == (double) mat[i][k]) {
438                  mat1[i][k][i] = 0.;
439               }
440             } else {
441               mat1[i][k][i] = 0.;
442             }
443          }
444       }
445
446       j = 2*j -1;
447    }
448
449
450 /* Finalmente, para concluir los preparativos para el cálculo de las
     signaturas fraccionarias discretas, reservamos memoria dinámica para la
      variable auxvec, que de nuevo utilizaremos para precalcular y guardar
     valores que necesitaremos durante el algoritmo. En este caso,
     guardaremos los coeficientes por los que tenemos que multiplicar los
     diferentes términos de las signaturas fraccionarias discretas de los
     caminos lineales para calcularlas. De esta manera, solo se realizan los
```

```
    cálculos una vez y se reduce, por tanto, el tiempo total de ejecución.
     Notemos que, si en lugar de utilizar auxvec, que es un apuntador
    triple, usáramos 7 (profundidad de la signatura) apuntadores dobles
    aux1, ..., aux7, podríamos reducir ligeramente el tiempo de ejecución.
    Todo y esto, esta alternativa es más farragosa y aumentaría el código
    en unas 700 líneas. Aunque disponemos de un programa que sí que utiliza
     esta idea y realiza los mismos cálculos, hemos decido por los motivos
    indicados incluir esta versión en el anexo del trabajo. */
451
452 auxvec = (double ***) malloc(depth*sizeof(double **));
453
454 if (auxvec == NULL) {
455    printf ("Error en la reserva de memoria. Ponemos fin a la ejecucion\
    n");
456
457    j = 2;
458    for (i = 0; i < m; i++) {
459       free(mat[i]);
460       for (k = 1; k < j; k++) {
461          free(mat1[i][k]);
462       }
463
464       j = 2*j -1;
465       free(mat1[i]);
466    }
467
468    free(mat);
469    free(mat1);
470
471    for (i = 0; i < num_caminos; i++) {
472       for (j = 0; j < comp; j++) {
473          free(matvec[i][j]);
474       }
475
476       free(matvec[i]);
477       free(matsig[i]);
478    }
479
480    free(matvec);
481    free(matsig);
482
483    return 1;
484 }
485
486
487 for (i = 0; i < depth; i++) {
488    auxvec[i] = (double **) malloc((n-1)*sizeof(double *));
489
490    if (auxvec[i] == NULL) {
491       printf ("Error en la reserva de memoria. Ponemos fin a la
    ejecucion\n");
492
493       for (j = 0; j < i; j++) {
494          free(auxvec[j]);
495       }
496
497       j = 2;
498       for (i = 0; i < m; i++) {
499          free(mat[i]);
500          for (k = 1; k < j; k++) {
501             free(mat1[i][k]);
```

```
502             }
503
504             j = 2*j -1;
505             free(mat1[i]);
506          }
507
508          free(mat);
509          free(mat1);
510
511          for (i = 0; i < num_caminos; i++) {
512             for (j = 0; j < comp; j++) {
513                free(matvec[i][j]);
514             }
515
516             free(matvec[i]);
517       free(matsig[i]);
518          }
519
520    free(matsig);
521          free(matvec);
522          free(auxvec);
523
524          return 1;
525       }
526
527    }
528
529
530    for (l = 0; l < depth; l++) {
531       for (a = 0; a < n-1; a++) {
532
533          auxvec[l][a] = (double *) malloc ((2*n-1)*sizeof(double));
534
535          if (auxvec[l][a] == NULL) {
536
537             printf("Error en la reserva de memoria. Ponemos fin a la
    ejecucion\n");
538
539
540       for (i = 0; i < a; i++) {
541             free(auxvec[l][i]);
542          }
543
544          for (i = 0; i < l; i++) {
545             for (j = 0; j < n-1; j++) {
546                free(auxvec[i][j]);
547             }
548          }
549
550          for (i = 0; i < depth; i++) {
551             free(auxvec[i]);
552          }
553
554          free(auxvec);
555
556
557          j = 2;
558          for (i = 0; i < m; i++) {
559             free(mat[i]);
560             for (k = 1; k < j; k++) {
561                free(mat1[i][k]);
```

```
562                }
563
564                j = 2*j -1;
565                free(mat1[i]);
566             }
567
568             free(mat);
569             free(mat1);
570
571             for (i = 0; i < num_caminos; i++) {
572                for (j = 0; j < comp; j++) {
573                   free(matvec[i][j]);
574                }
575
576          free(matvec[i]);
577          free(matsig[i]);
578             }
579
580             free(matvec);
581       free(matsig);
582
583             return 1;
584          }
585
586          for (d = 2*a + 2; d < 2*n-1; d++) {
587             auxvec[l][a][d] = (pow(d/2.-a, alfa*(l+1))*gsl_sf_beta_inc(l*
    alfa + 1, alfa, 1./(d/2. - a))*gsl_sf_beta(l*alfa + 1, alfa))/(
    gsl_sf_gamma(alfa)*gsl_sf_gamma(l*alfa + 1));
588          }
589       }
590    }
591
592
593 /* A partir de aquí, una vez precalculados todos los valores y datos
    auxiliares que necesitábamos, iniciamos una sesión paralela haciendo
    uso de la librería OpenMp. En ella, utilizando la variable sig,
    reservaremos memoria dinámica para guardar todas las signaturas
    fraccionarias discretas que vayamos calculando a lo largo del algoritmo
    . Notemos que usamos las variables aux y auxp para controlar si se
    produce un error en la reserva de memoria y actuar entonces de la
    manera adecuada. */
594
595    aux = 0;
596
597 #pragma omp parallel num_threads(7) private(sig, i, h, j, k, a, d, l, auxp
    )
598 {
599
600    auxp = 0;
601
602    j = 2;
603
604    sig = (double ****) malloc(m*sizeof(double ***));
605
606    if (sig == NULL) {
607
608       #pragma omp critical
609       {
610          aux = 1;
611       }
612
```

```
613         auxp = 1;
614      }
615
616
617      for (i = 0; i < m && auxp == 0; i++) {
618         sig[i] = (double ***) malloc(j*sizeof(double *));
619
620         if (sig[i] == NULL) {
621
622            l = 2;
623            for (a = 0; a < i; a++) {
624               for (k = 1; k < l; k++) {
625                  for (h = 0; h <= a; h++) {
626                     free(sig[a][k][h]);
627                  }
628
629                  free(sig[a][k]);
630               }
631
632               l = 2*l-1;
633               free(sig[a]);
634            }
635
636            free(sig);
637
638            #pragma omp critical
639            {
640               aux = 1;
641            }
642
643            auxp = 1;
644         }
645
646         for (k = 1; k < j && auxp == 0; k++) {
647            sig[i][k] = (double **) malloc((i+1)*sizeof(double *));
648
649            if (sig[i][k] == NULL) {
650
651               for (l = 1; l < k; l++) {
652                  for (h = 0; h <= i; h++) {
653                     free(sig[i][l][h]);
654                  }
655
656                  free(sig[i][l]);
657               }
658
659               free(sig[i]);
660
661               l = 2;
662               for (a = 0; a < i; a++) {
663                  for (k = 1; k < l; k++) {
664                     for (h = 0; h <= a; h++) {
665                        free(sig[a][k][h]);
666                     }
667
668                     free(sig[a][k]);
669                  }
670
671                  l = 2*l-1;
672                  free(sig[a]);
673               }
```

```
674
675             free(sig);
676
677             #pragma omp critical
678             {
679                aux = 1;
680             }
681
682             auxp = 1;
683          }
684
685    for (h = 0; h <= i && auxp == 0; h++) {
686             sig[i][k][h] = (double *) malloc(lensig*sizeof(double));
687
688
689             if (sig[i][k][h] == NULL) {
690
691                for (l = 0; l < h; l++) {
692                   free(sig[i][k][l]);
693                }
694
695                free(sig[i][k]);
696
697                for (l = 1; l < k; l++) {
698                   for (h = 0; h <= i; h++) {
699                      free(sig[i][l][h]);
700                   }
701
702                   free(sig[i][l]);
703                }
704
705                free(sig[i]);
706
707                l = 2;
708                for (a = 0; a < i; a++) {
709                   for (k = 1; k < l; k++) {
710                      for (h = 0; h <= a; h++) {
711                         free(sig[a][k][h]);
712                      }
713
714                      free(sig[a][k]);
715                   }
716
717                   l = 2*l-1;
718                   free(sig[a]);
719                }
720
721                free(sig);
722
723                #pragma omp critical
724                {
725                   aux = 1;
726                }
727
728                auxp = 1;
729             }
730          }
731       }
732
733       j = 2*j - 1;
734    }
```

```
735
736
737     #pragma omp barrier
738
739     if (aux == 0) {
740        #pragma omp for schedule(static)
741
742        for (i = 0; i < num_caminos; i++) {
743           calcula_sig(n, m, comp, depth, matvec[i], auxvec, mat, mat1, sig)
    ;
744           for (j = 0; j < lensig; j++) {
745        matsig[i][j] = sig[0][1][0][j];
746           }
747           printf ("%d\n", i);
748        }
749     }
750
751     #pragma omp barrier
752
753     if (auxp == 0) {
754
755        j = 2;
756        for (i = 0; i < m; i++) {
757           for (k = 1; k < j; k++) {
758              for (h = 0; h <= i; h++) {
759                 free(sig[i][k][h]);
760              }
761              free(sig[i][k]);
762           }
763
764           j = 2*j -1;
765           free(sig[i]);
766        }
767
768        free(sig);
769     }
770
771  }
772
773
774  /* Una vez acabados los cálculos de las signaturas fraccionarias discretas
        , liberamos toda la memoria que teníamos reservada y escribimos en el
        archivo indicado los valores obtenidos, poniendo fin posteriormente a
        la ejecución. */
775
776
777     if (aux != 0) {
778        printf ("Error en la reserva de memoria. Ponemos fin a la ejecucion\
       n");
779
780        for (l = 0; l < depth; l++) {
781           for (i = 0; i < n-1; i++) {
782              free(auxvec[l][i]);
783           }
784
785           free(auxvec[l]);
786        }
787
788        free(auxvec);
789
790        j = 2;
```

```
791         for (i = 0; i < m; i++) {
792             free(mat[i]);
793             for (k = 1; k < j; k++) {
794                 free(mat1[i][k]);
795             }
796 
797             j = 2*j -1;
798             free(mat1[i]);
799         }
800 
801         free(mat);
802         free(mat1);
803 
804         for (i = 0; i < num_caminos; i++) {
805             free(matsig[i]);
806         }
807 
808         free(matsig);
809 
810         for (i = 0; i < num_caminos; i++) {
811             for (j = 0; j < comp; j++) {
812                 free(matvec[i][j]);
813             }
814 
815             free(matvec[i]);
816         }
817 
818         free(matvec);
819 
820         return 1;
821     }
822 
823 
824     for (l = 0; l < depth; l++) {
825         for (i = 0; i < n-1; i++) {
826             free(auxvec[l][i]);
827         }
828 
829         free(auxvec[l]);
830     }
831 
832     free(auxvec);
833 
834     j = 2;
835     for (i = 0; i < m; i++) {
836         free(mat[i]);
837         for (k = 1; k < j; k++) {
838             free(mat1[i][k]);
839         }
840 
841         j = 2*j -1;
842         free(mat1[i]);
843     }
844 
845     free(mat);
846     free(mat1);
847 
848 
849     arch = fopen("resultados_test.csv", "w");
850 
851     if (arch == NULL) {
```

```
852         printf ("Error al arbrir un archivo. Ponemos fin a la ejecucion\n");
853
854         for (i = 0; i < num_caminos; i++) {
855            free(matsig[i]);
856         }
857
858         free(matsig);
859
860         for (i = 0; i < num_caminos; i++) {
861            for (j = 0; j < comp; j++) {
862               free(matvec[i][j]);
863            }
864
865      free(matvec[i]);
866         }
867
868         free(matvec);
869
870         return 1;
871      }
872
873      for (i = 0; i < num_caminos; i++) {
874         for (j = 0; j < lensig - 1; j++) {
875            fprintf(arch, "%le,", matsig[i][j]);
876         }
877
878         fprintf(arch, "%le\n", matsig[i][lensig - 1]);
879
880         free(matsig[i]);
881      }
882
883      free(matsig);
884      fclose(arch);
885
886
887      for (i = 0; i < num_caminos; i++) {
888         for (j = 0; j < comp; j++) {
889            free(matvec[i][j]);
890         }
891
892         free(matvec[i]);
893      }
894
895      free(matvec);
896
897      return 0;
898
899   }
900
901
902   void calcula_sig (int n, int m, int comp, int depth, double **vec, double
         ***auxvec, int **mat, double ***mat1, double ****sig) {
903      int i, j, k, h, a, a1, d, ii, jj, kk, uu, vv, hh, ll, ind;
904
905   /* Utilizamos esta función para calcular las signaturas fraccionarias
         discretas de los caminos. Para ello, tal y como lo hemos indicado ya,
         no usaremos el algoritmo recursivo que nos sugeriría la definición
         basada en una generalización de la identidad de Chen por ser demasiado
         lento, sino que optamos por un algoritmo alternativo que calcula
         primero las signaturas fraccionarias discretas de los caminos lineales
         que forman el original y después las combina de la manera adecuada para
```

```
    obtener al final la signatura fraccionaria discreta deseada. */

906
907   j = 2;
908   for (i = 0; i < m-1; i++) {
909      j = 2*j -1;
910   }
911
912   for (i = m-1; i >= 0; i--) {
913      for (k = 1; k < j; k++) {
914   for (h = 0; h <= i; h++) {
915      if ((mat1[i][k][h] != 0.) && (mat[i][k]-mat[i][k-1] != 0)) {
916         if (mat[i][k-1] + 1 ==  mat[i][k]) {
917      d = (int)(2*mat1[i][k][h]);
918      a = mat[i][k-1];
919      a1 = mat[i][k];
920
921      ind = 0;
922                  for (ii = 0; ii < comp; ii++) {
923                     sig[i][k][h][ind] = (vec[ii][a1] - vec[ii][a])*auxvec
    [0][a][d];
924                     ind += 1;
925                  }
926                  for (ii = 0; ii < comp; ii++) {
927                     for (jj = 0; jj < comp; jj++) {
928                        sig[i][k][h][ind] = (vec[ii][a1] - vec[ii][a])*(
    vec[jj][a1] - vec[jj][a])*auxvec[1][a][d];
929                        ind += 1;
930         }
931                  }
932                  for (ii = 0; ii < comp; ii++) {
933                     for (jj = 0; jj < comp; jj++) {
934                        for (kk = 0; kk < comp; kk++) {
935                           sig[i][k][h][ind] = (vec[ii][a1] - vec[ii][a])
    *(vec[jj][a1] - vec[jj][a])*(vec[kk][a1] - vec[kk][a])*auxvec[2][a][d];
936                           ind += 1;
937                        }
938                     }
939                  }
940                  for (ii = 0; ii < comp; ii++) {
941                     for (jj = 0; jj < comp; jj++) {
942                        for (kk = 0; kk < comp; kk++) {
943                           for (uu = 0; uu < comp; uu++) {
944                              sig[i][k][h][ind] = (vec[ii][a1] - vec[ii][a
    ])*(vec[jj][a1] - vec[jj][a])*(vec[kk][a1] - vec[kk][a])*(vec[uu][a1] -
     vec[uu][a])*auxvec[3][a][d];
945                              ind += 1;
946                           }
947                        }
948                     }
949                  }
950
951      for (ii = 0; ii < comp; ii++) {
952                     for (jj = 0; jj < comp; jj++) {
953                        for (kk = 0; kk < comp; kk++) {
954                           for (uu = 0; uu < comp; uu++) {
955                              for (vv = 0; vv < comp; vv++) {
956                                 sig[i][k][h][ind] = (vec[ii][a1] - vec[ii
    ][a])*(vec[jj][a1] - vec[jj][a])*(vec[kk][a1] - vec[kk][a])*(vec[uu][a1
    ] - vec[uu][a])*(vec[vv][a1] - vec[vv][a])*auxvec[4][a][d];
957                                 ind += 1;
958                              }
```

```
                        }
                    }
                }
            }

            for (ii = 0; ii < comp; ii++) {
                for (jj = 0; jj < comp; jj++) {
                    for (kk = 0; kk < comp; kk++) {
                        for (uu = 0; uu < comp; uu++) {
                            for (vv = 0; vv < comp; vv++) {
                                for (hh = 0; hh < comp; hh++) {
                                    sig[i][k][h][ind] = (vec[ii][a1] - vec[ii][a])*(vec[jj][a1] - vec[jj][a])*(vec[kk][a1] - vec[kk][a])*(vec[uu][a1] - vec[uu][a])*(vec[vv][a1] - vec[vv][a])*(vec[hh][a1] - vec[hh][a])*auxvec[5][a][d];
                                    ind += 1;
                                }
                            }
                        }
                    }
                }
            }

            for (ii = 0; ii < comp; ii++) {
                for (jj = 0; jj < comp; jj++) {
                    for (kk = 0; kk < comp; kk++) {
                        for (uu = 0; uu < comp; uu++) {
                            for (vv = 0; vv < comp; vv++) {
                                for (hh = 0; hh < comp; hh++) {
                                    for (ll = 0; ll < comp; ll++) {
                                        sig[i][k][h][ind] = (vec[ii][a1] - vec[ii][a])*(vec[jj][a1] - vec[jj][a])*(vec[kk][a1] - vec[kk][a])*(vec[uu][a1] - vec[uu][a])*(vec[vv][a1] - vec[vv][a])*(vec[hh][a1] - vec[hh][a])*(vec[ll][a1] - vec[ll][a])*auxvec[6][a][d];
                                        ind += 1;
                                    }
                                }
                            }
                        }
                    }
                }
            }

        } else {
    a1 = 2*k;
    a = 2*k - 1;

    ind = 0;
            for (ii = 0; ii < comp; ii++) {
                sig[i][k][h][ind] = sig[i+1][a][h][ind] + sig[i+1][a1][h][ind];
                ind += 1;
            }
            for (ii = 0; ii < comp; ii++) {
                for (jj = 0; jj < comp; jj++) {
                    sig[i][k][h][ind] = sig[i+1][a][h][ind] + sig[i+1][a][i+1][ii]*sig[i+1][a1][h][jj] + sig[i+1][a1][h][ind];
                    ind += 1;
                }
            }
            for (ii = 0; ii < comp; ii++) {
```

```
                    for (jj = 0; jj < comp; jj++) {
                        for (kk = 0; kk < comp; kk++) {
                            sig[i][k][h][ind] = sig[i+1][a][h][ind] + sig[i
+1][a][i+1][comp*(ii+1) + jj]*sig[i+1][a1][h][kk] + sig[i+1][a][i+1][ii
]*sig[i+1][a1][h][comp*(jj+1) + kk] + sig[i+1][a1][h][ind];
                            ind += 1;
                        }
                    }
                }
                for (ii = 0; ii < comp; ii++) {
                    for (jj = 0; jj < comp; jj++) {
                        for (kk = 0; kk < comp; kk++) {
                            for (uu = 0; uu < comp; uu++) {
                                sig[i][k][h][ind] = sig[i+1][a][h][ind] +
sig[i+1][a][i+1][comp*(comp*(ii+1) + (jj+1)) + kk]*sig[i+1][a1][h][uu]
+ sig[i+1][a][i+1][comp*(ii+1) + jj]*sig[i+1][a1][h][comp*(kk+1) + uu]
+ sig[i+1][a][i+1][ii]*sig[i+1][a1][h][comp*(comp*(jj+1) + (kk+1)) + uu
] + sig[i+1][a1][h][ind];
                                ind += 1;
                            }
                        }
                    }
                }

    for (ii = 0; ii < comp; ii++) {
                    for (jj = 0; jj < comp; jj++) {
                        for (kk = 0; kk < comp; kk++) {
                            for (uu = 0; uu < comp; uu++) {
                                for (vv = 0; vv < comp; vv++) {
                                    sig[i][k][h][ind] = sig[i+1][a][h][ind] +
 sig[i+1][a][i+1][comp*(comp*(comp*(ii+1) + (jj+1)) + (kk + 1)) + uu]*
sig[i+1][a1][h][vv] + sig[i+1][a][i+1][comp*(comp*(ii+1) + (jj+1)) + kk
]*sig[i+1][a1][h][comp*(uu+1) + vv] + sig[i+1][a][i+1][comp*(ii+1) + jj
]*sig[i+1][a1][h][comp*(comp*(kk+1) + (uu+1)) + vv] + sig[i+1][a][i+1][
ii]*sig[i+1][a1][h][comp*(comp*(comp*(jj+1) + (kk+1)) + (uu+1)) + vv] +
 sig[i+1][a1][h][ind];
                                    ind += 1;
                                }
                            }
                        }
                    }
                }

    for (ii = 0; ii < comp; ii++) {
                    for (jj = 0; jj < comp; jj++) {
                        for (kk = 0; kk < comp; kk++) {
                            for (uu = 0; uu < comp; uu++) {
                                for (vv = 0; vv < comp; vv++) {
                                    for (hh = 0; hh < comp; hh++) {
                                        sig[i][k][h][ind] = sig[i+1][a][h][ind
] + sig[i+1][a][i+1][comp*(comp*(comp*(comp*(ii+1) + (jj+1)) + (kk + 1)
) + (uu+1)) + vv]*sig[i+1][a1][h][hh] + sig[i+1][a][i+1][comp*(comp*(
comp*(ii+1) + (jj+1)) + (kk+1)) + uu]*sig[i+1][a1][h][comp*(vv+1) + hh]
 + sig[i+1][a][i+1][comp*(comp*(ii+1) + (jj+1)) + kk]*sig[i+1][a1][h][
comp*(comp*(uu+1) + (vv+1)) + hh] + sig[i+1][a][i+1][comp*(ii+1) + jj]*
sig[i+1][a1][h][comp*(comp*(comp*(kk+1) + (uu+1)) + (vv+1)) + hh] + sig
[i+1][a][i+1][ii]*sig[i+1][a1][h][comp*(comp*(comp*(comp*(jj+1) + (kk
+1)) + (uu+1)) + (vv+1)) + hh] + sig[i+1][a1][h][ind];

                                        ind += 1;
                                    }
```

```
                                    }
                                }
                            }
                        }
                    }

/* Observemos que estamos utilizando expresiones largas para calcular los
    índices a los que accedemos usando los apuntadores y que estamos
    repitiendo muchos cálculos. Todo y eso, si se intentan precalcular
    estos índices para evitar estas repeticiones, los accesos a memoria
    resultan ser más costosos que los cálculos y el tiempo de ejecución es
    mayor. Por tanto, hemos optado por mantener aquí esta versión del có
    digo más farragosa, pero que da mejores resultados. */

       for (ii = 0; ii < comp; ii++) {
                        for (jj = 0; jj < comp; jj++) {
                            for (kk = 0; kk < comp; kk++) {
                                for (uu = 0; uu < comp; uu++) {
                                    for (vv = 0; vv < comp; vv++) {
                                        for (hh = 0; hh < comp; hh++) {
                                            for (ll = 0; ll < comp; ll++) {
                                                sig[i][k][h][ind] = sig[i+1][a][h][
    ind] + sig[i+1][a][i+1][comp*(comp*(comp*(comp*(comp*(ii+1) + (jj+1)) +
     (kk + 1)) + (uu+1)) + (vv+1)) + hh]*sig[i+1][a1][h][ll] + sig[i+1][a][
    i+1][comp*(comp*(comp*(comp*(ii+1) + (jj+1)) + (kk+1)) + (uu+1)) + vv]*
    sig[i+1][a1][h][comp*(hh+1) + ll] + sig[i+1][a][i+1][comp*(comp*(comp*(
    ii+1) + (jj+1)) + (kk+1)) + uu]*sig[i+1][a1][h][comp*(comp*(vv+1) + (hh
    +1)) + ll] + sig[i+1][a][i+1][comp*(comp*(ii+1) + (jj+1)) + kk]*sig[i
    +1][a1][h][comp*(comp*(comp*(uu+1) + (vv+1)) + (hh+1)) + ll] + sig[i
    +1][a][i+1][comp*(ii+1) + jj]*sig[i+1][a1][h][comp*(comp*(comp*(comp*(
    kk+1) + (uu+1)) + (vv+1)) + (hh+1)) + ll] + sig[i+1][a][i+1][ii]*sig[i
    +1][a1][h][comp*(comp*(comp*(comp*(comp*(jj+1) + (kk+1)) + (uu+1)) + (
    vv+1)) + (hh+1)) + ll] + sig[i+1][a1][h][ind];
                                                ind += 1;
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
        }
       }
   }
       }

       j = (j+1)/2;
   }

   return;

}
```