UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DEPARTAMENT DE MATEMÀTICA APLICADA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Matemática Aplicada

Medidas de heterogeneidad para flujos eléctricos

Trabajo Fin de Máster

Máster Universitario en Investigación Matemática

AUTOR/A: Ramírez Muñoz, José Miguel

Tutor/a: Peris Manguillot, Alfredo

Cotutor/a: Millet Roig, José

Cotutor/a: Castells Ramón, Francisco Sales

CURSO ACADÉMICO: 2023/2024

VNIVERSITAT DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Trabajo Fin de Máster - Curs 2023/2024

Medidas de heterogeneidad para flujos eléctricos

Autor: **Jose Miguel Ramírez Muñoz**

Tutor: Alfred Peris Manguillot

Tutor: José Millet Roig

Tutor: Francisco Castells Ramon

Facultat de Ciències Matemàtiques

dma Departament de Matemàtica Aplicada

investmat

Gracias al grupo de investigación EP Analaytics Lab
por la ayuda y la oportunidad de poder trabajar con ellos.

Índice

Resumen

El objetivo de este trabajo consiste en definir una nueva forma de medir la variabilidad de un campo vectorial con el fin de estudiarla para el caso del catéter HDgrid en la detección de distintos tejidos.
Para ello, primero se introduccirá el catéter y como mide las direcciones del campo eléctrico del corazón. Después, un breve resumen sobre el artículo original.
En la segunda sección, se darán las definiciones de esta métrica y sobre que condiciones se puede calcular. Más tarde, veremos el caso concreto del HDgrid y plantearemos el problema de optimización en condiciones concretas para encontrar un patron que nos dé el valor máximo, lo cual no era conocido para otras métricas de heterogeneidad previamente estudiadas.

1. Introducción

Las enfermedades del corazón son una de las causas de muerte más importantes en el mundo, por ello es necesario investigar en técnicas de detección y procedimientos para sanarlas. Una de las patologías más comunes son las arritmias, latidos irregulares del corazón que provocan un mal funcionamiento de este. Para sanar esta afección existen diferentes técnicas según sea el origen del problema, que determinará el especialista pertinente, una de ellas es el cateterismo, un procedimiento invasivo que consiste en introducir un catéter hasta el corazón, determinar cual es la zona que produce una irregularidad en la propagación del campo eléctrico y aixlarla mediante una ablación en el tegido circundante. El cateter que dió pie al desarrollo de este trabajo fué el *HDgrid*, un catéter multielectrodo de alta intensidad que permite extraer la dirección y velocidad del campo eléctrico del corazón mediante el mapeo omnipolar durante un pulso. Este mejoraba el anterior dispositivo dado que es independiente de la dirección en la que se coloque sobre las paredes del corazón. El *HDgrid* consiste en una matriz de 4×4 electrodos, estos formas pares ortogonales de bipolares que forman un conjunto triangular conocidos como *cliques*. La dirección entonces queda determinada por el bucle omnipolar que forman los cliques. En el artículo [3] se encuentra una explicación más detallada de como se obtienen estos datos. En el apartado **A** de la figura 1 podemos encontrar un ejemplo del cateter y la omnipolar.

La métrica propuesta en este trabajo ha sido desarrollada en base a la introducida en el artículo [5], que parte de la idea de intentar medir de alguna forma como de desorganizado esta una región de un campo vectorial, tomando como inspiración el rotacional o la divergencia del análisis vectorial, que también eran usadas para la detección de problemas cardíacos, con la idea de poder determinar que zonas del corazón son las que provocan un mal transporte de la onda, con ello se obtiene un marcador que permita al profesional que realice la ablación poder determinar si se trata de una zona problemática. Nuestra métrica consiste en la mejora de la introducida en [5], la cual permite obtener el óptimo (máximo) teórico de heterogeneidad para cualquier catéter dispuesto en HDgrid, algo que no se había podido realizar hasta ahora.

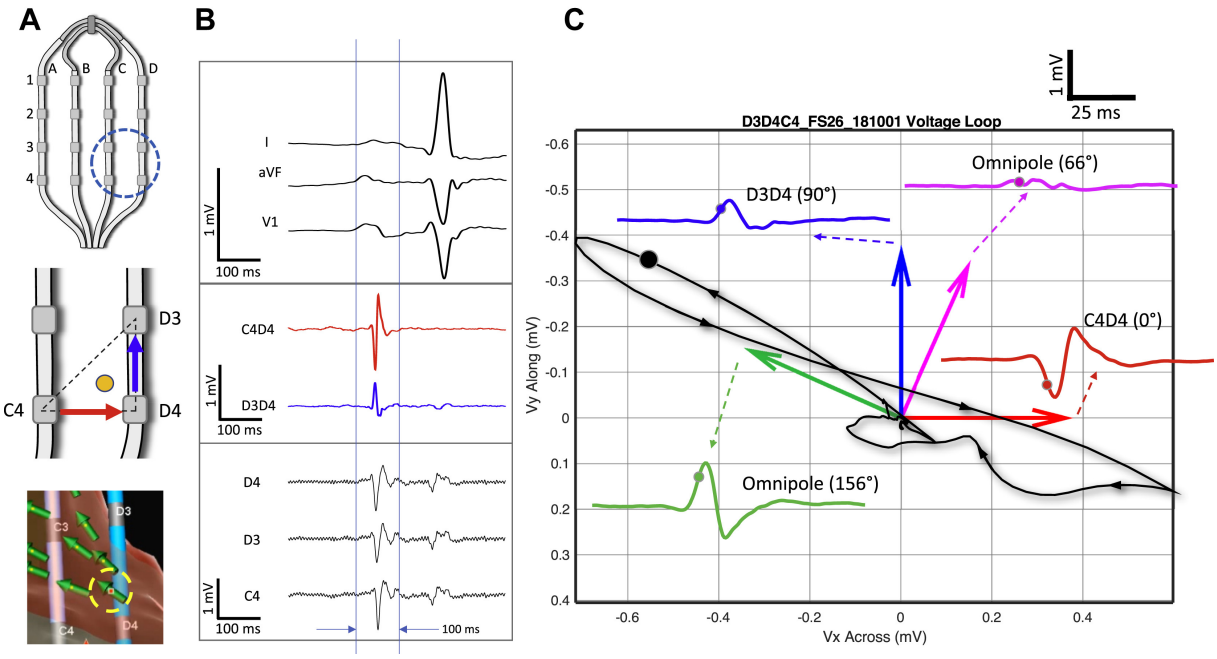


Figura 1: Imagen original del articulo [3]. En el apartado **A** encontramos el cateter y las bipolares ortoganeles.

2. Métrica de heterogeneidad para campo continuo

Originalmente, para el cálculo de la heterogeneidad se suponía una distancia infinitesimal h entre los distintos nodos. Después, si $X = (x_{ij})_{p\times q}$ es la malla de nodos y $Y = (y_{ij})_{p\times q}$ son los valores obtenidos en cada uno de los nodos, entonces se calculaban la media de las diferencias en valor absoluto divididas por la distancia entre los nodos supuestos igualmente espaciados, por ejemplo para la dirección u_0 con ángulo $\theta_0 = 0$

$$\left(\frac{\Delta\Gamma}{\Delta u_0}\right)_{ij} = \|\frac{|y_{i(j+1)} - y_{ij}| + |y_{i(j-1)} - y_{ij}|}{2h}\|$$

Así se procede para todas las direcciones y, para los casos en que no haya un elemento "anterior" o "posterior" como son los casos de la primera y última columna para la dirección u_0, pues se toma el "posterior" o "anterior" respectivamente.
Si quitamos los valores absolutos de la definición anterior entonces obtenemos

$$\left(\frac{\Delta\Gamma}{\Delta u_0}\right)_{ij} = \|\frac{y_{i(j+1)} - 2y_{ij} + y_{i(j-1)}}{2h}\|$$

que se trata de la expresión de la derivada direccional en diferencias centradas.
Si suponemos que los valores Y son las imagenes de una cierta función F en los distintos

nodos de X, es decir, $y_{ij} = F(x_{ij})$ y tomamos el límite cuando la distancia entre los nodos tiende a 0, entonces

$$\begin{aligned} D_{u_0}F(x_{ij}) &= \lim_{h\to 0^+}\left(\frac{\Delta\Gamma}{\Delta u_0}\right)_{ij} \\ &= \lim_{h\to 0^+}\|\frac{y_{i(j+1)} - 2y_{ij} + y_{i(j-1)}}{2h}\| \end{aligned}$$

Supongamos ahora que conocemos la función F sobre un dominio D. Tomemos un valor $x \in D$, el objetivo consiste en observar la variación cuadrática media sobre todas las direcciones posibles para este punto x, como existen una cantidad infinita de direcciones, tomemos el conjunto de direcciones unitarias

$$\mathbb{B} := \{v \in \mathbb{R}^n : \|v\| = 1\}$$

que también puede ser expresado en forma polar, por ejemplo, si nos encontramos en $\mathbb{R}^2$ entonces

$$\mathbb{B} = \{(\cos\theta, sin\theta);\ \ \theta \in [0, 2\pi]\}$$

Por último, esta variación cuadrática media, $\Psi(x)$, queda definida como

$$\Psi(x) = \sqrt{\frac{1}{m(\mathbb{B})}\int_{\mathbb{B}} \|D_u F(x)\|^2\, du}$$

para un punto $x \in D$ y con $m(\mathbb{B})$ la medida de Lebesgue del conjunto de direcciones.

Una de las preguntas que podemos hacernos es que condiciones se necesitan en la función F para que la integral sea convergente, si tomamos la función F diferenciable en el punto x, entonces la función es diferenciable, con la derivada continua en cualquier dirección y, por tanto, es integrable Lebesgue.

Definición 2.1 (Heterogeneidad de un campo vectorial) *Sea $D \subseteq \mathbb{R}^n$ un conjunto abierto, sea también $F : D \longrightarrow \mathbb{R}^m$ diferenciable en $x \in D$. Se define la heterogeneidad de un campo vectorial en un punto $x \in D$ como*

$$\Psi(x) = \sqrt{\frac{1}{m(\mathbb{B})}\int_{\mathbb{B}} \|D_u F(x)\|^2\, du} \tag{1}$$

donde $\mathbb{B}$ es el conjunto de direcciones unitarias, $m(\mathbb{B})$ la medida de Lebesgue y $D_u F(x)$ es la derivada de F en la dirección de $u \in \mathbb{B}$ evaluada en $x \in D$.

Supongamos que tenemos un campo vectorial definido en la clausura del conjunto compacto $D \subset \mathbb{R}^n$. Hasta ahora hemos dado una definición para la heterogeneidad en un punto, para calcular la heterogeneidad en una región del campo vectorial utilizaremos, de la misma forma que para un punto, la media cuadrática. Dado que se trata de la clausura de un conjunto compacto, entonces $m(D) < \infty$, por lo que podemos dar la siguiente definición

Definición 2.2 (Heterogeneidad en un conjunto) *Sea $D \subset \mathbb{R}^n$ la clausura del interior de un compacto, sea $F : D \longrightarrow \mathbb{R}^m$ un campo vectorial tal que $F \in \mathcal{C}^1(\mathring{D})$. Entonces se define la heterogeneidad en D como*

$$\mathcal{V}(D) := \sqrt{\frac{1}{m(D)} \int_D \Psi(x)^2 \, dx} \tag{2}$$

donde $m(D)$ es la medida de Lebesgue del conjunto.

Si suponemos que el campo vectorial se encuentra en polares para el caso en $\mathbb{R}^2$, esto quiere decir que

$$F(x, y) = (\cos\theta(x, y), \sin\theta(x, y))$$

y dado $u \in \mathbb{B}$ una dirección unitaria, consideramos

$$D_u F(a, b) = \lim_{t \to 0^+} \frac{F((a, b) + t \cdot u) - F(a, b)}{t}$$

Entonces, si $u = (\cos\alpha, \sin\alpha)$, bajo las condiciones de diferenciabilidad

$$D_u F = \begin{bmatrix} \langle \nabla F_1, u \rangle \\ \langle \nabla F_2, u \rangle \end{bmatrix} = dF \cdot \begin{bmatrix} \cos\alpha \\ \sin\alpha \end{bmatrix}$$

donde se tiene que

$$dF = \begin{bmatrix} -\sin\theta & -\sin\theta \\ \cos\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \frac{\partial\theta}{\partial x} & 0 \\ 0 & \frac{\partial\theta}{\partial y} \end{bmatrix}$$

Por lo tanto

$$\|D_u F\|^2 = \left(\cos\alpha \frac{\partial\theta}{\partial x} + \sin\alpha \frac{\partial\theta}{\partial y} \right)^2$$

Con lo que al sustituir en la expresión de la heterogeneidad en un punto se tiene

$$\Psi(a, b) = \left(\frac{1}{2\pi} \int_{[0,2\pi]} \left(\cos\alpha \frac{\partial\theta}{\partial x} + \sin\alpha \frac{\partial\theta}{\partial y} \right)^2 d\alpha \right)^{1/2}$$

Y en la heterogeneidad en un conjunto se tiene entonces

$$\mathcal{V}(D) = \left(\frac{1}{2\pi m(D)} \int_D \left[\int_{[0,2\pi]} \left(\cos\alpha \frac{\partial\theta}{\partial x} + \sin\alpha \frac{\partial\theta}{\partial y} \right)^2 d\alpha \right] dx \, dy \right)^{1/2}$$

Por último, si nos fijamos en que

$$\int_{[0,2\pi]} \left(\cos\alpha\frac{\partial\theta}{\partial x} + \sin\alpha\frac{\partial\theta}{\partial y}\right)^2 d\alpha =$$

$$= \int_{[0,2\pi]} \left(\cos^2\alpha\left(\frac{\partial\theta}{\partial x}\right)^2 + \sin^2\alpha\left(\frac{\partial\theta}{\partial y}\right)^2 + 2\cos\alpha\sin\alpha\frac{\partial\theta}{\partial x}\frac{\partial\theta}{\partial y}\right) d\alpha$$

$$= \pi\left[\left(\frac{\partial\theta}{\partial x}\right)^2 + \left(\frac{\partial\theta}{\partial y}\right)^2\right]$$

Entonces

$$\mathcal{V}(D) = \left(\frac{1}{2m(D)}\right)^{1/2}\left(\int_D \left(\left(\frac{\partial\theta}{\partial x}\right)^2 + \left(\frac{\partial\theta}{\partial y}\right)^2\right) dx\,dy\right)^{1/2}$$

2.1. Unidades de la métrica

Hagamos un breve análisis de las unidades para el caso de $\mathbb{R}^2$, supongamos que nuestro campo vectorial F se trata de un campo eléctrico. Este campo tiene como unidades V/m, su derivada direccional tiene unidades de V/m^2, con lo que el cuadrado de esta presenta unidades de $(V/m^2)^2$. Al hacer la integral sobre las direcciones estamos multiplicando por las unidades de la dirección que son anuladas por la medida del conjunto $\mathbb{B}$, puesto que ambas contienen las mismas unidades. Por lo tanto, la función $\Psi(x)$ tiene unidades de V/m^2.

Si estudiamos ahora las unidades de la función $\mathcal{V}$ bajo las mismas condiciones que antes, podemos observar que al integrar sobre un area añadimos sus unidades, que son m^2, claro que al dividir por la medida de esta area, estas se anulan, obteniendo así las mismas unidades que Ψ, puesto que el cuadrado también se anula con la raíz.

En la tabla 1 se encuentra un resumen de lo estudiado en los párrafos anteriores.

F	$\longleftrightarrow$	V/m	Ψ	$\longleftrightarrow$	V/m^2
D_uF	$\longleftrightarrow$	V/m^2	$\mathcal{V}$	$\longleftrightarrow$	V/m^2

Tabla 1: Resumen de las unidades.

3. Discretización de la métrica de heterogeneidad

A la hora de discretizar la métrica, dependemos en cierta medida de como estan dispuestos los nodos en el catéter. Para el caso del HDgrid, cada nodo esta separado por

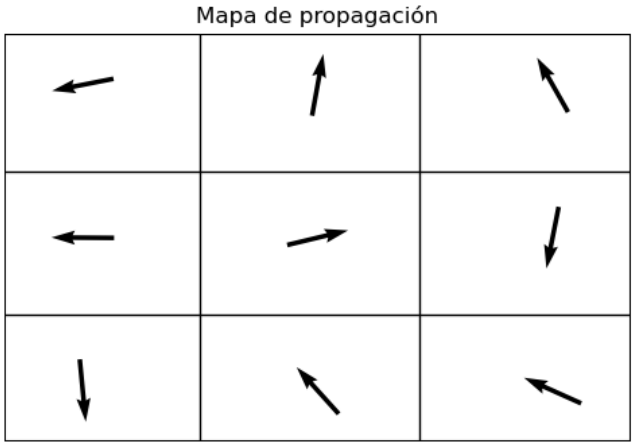


Figura 2: Mapa de propagación aleatorio.

una distancia de 4 mm si se tratan de los verticales u horizontales, o de $4\sqrt{2}$ mm si son los diagonales. Por lo tanto, tenemos dos distancias respectivas, por un lado $h = 4$ mm y por otro $h^* = h\sqrt{2} = 4\sqrt{2}$ mm. Con esto en mente, supongamos que tenemos la malla

$$X = \begin{pmatrix} x_{00} & x_{01} & x_{02} \\ x_{10} & x_{11} & x_{12} \\ x_{20} & x_{21} & x_{22} \end{pmatrix}.$$

De esta malla de nodos sabemos los valores $y_{ij} = F(x_{ij})$, por lo tanto, obtenemos

$$Y = \begin{pmatrix} y_{00} & y_{01} & y_{02} \\ y_{10} & y_{11} & y_{12} \\ y_{20} & y_{21} & y_{22} \end{pmatrix}.$$

Esta matriz Y corresponde al mapa de propagación para los nodos X, en la figura 2 puede observarse como sería para un caso aleatorio.
Dado que la derivada direccional se puede expresar como

$$D_u F(x) = \lim_{h \to 0^+} \frac{F(x + hu) - F(x)}{h\|u\|}$$

donde $u \in \mathbb{B}$, entonces tomando un $h > 0$ lo suficientemente pequeño se tiene que

$$D_u F(x) \approx \frac{F(x + hu) - F(x)}{h}$$

Con lo que quedaría que

$$\Psi(x) \approx \sqrt{\frac{1}{m(\mathbb{B})} \int_{\mathbb{B}} \frac{\|F(x + hu) - F(x)\|^2}{h^2}\, du}$$

Para el HDgrid, las direcciones a tomar son aquellas que sus ángulos vienen dados por $\theta_k = k \cdot \frac{\pi}{4}$ donde $k \in \{0, 1, \dots 7\}$ y tienen módulo unitario, es decir, $u_k = (\cos\theta_k, \sin\theta_k)$. Cada nodo interior puede tomar las 8 direcciones a la vez, por ejemplo para el nodo x_{00} las direcciones que se consideran son las correspondientes a los ángulos θ_0, θ_6 y θ_7, mientras que para el nodo x_{11} si que se pueden tomar todas. Entonces, dado que se trata de una media cuadrática, la discretización queda como

$$\Psi(x_{ij}) \approx \Psi_{ij} = \sqrt{\frac{1}{2\pi} \sum_{u \in N_{ij}} \frac{\|F(x_{ij} + h(u) \cdot u) - F(x_{ij})\|^2}{h(u)^2}} \tag{3}$$

donde N_{ij} es el conjunto de direcciones posibles de cada nodo y la función $h(u)$ es tal que

$$h(u) := \begin{cases} h\sqrt{2} & \text{si } u \text{ es diagonal} \\ h & \text{otro caso} \end{cases}$$

con u diagonal si $u = u_k$ con $k \in \{1, 3, 5, 7\}$, y $x_{ij} + h \cdot u$ es algún nodo x_{st} según sea la dirección u.

Una vez ya tenemos calculados los diferentes Ψ_{ij}, el cálculo de la heterogeneidad cuadrática media en un conjunto queda como

$$\mathcal{V} = \sqrt{\frac{1}{9} \sum_{i,j=0}^{2} \Psi_{ij}^2}$$

Por lo tanto, para concretar la modelización, los distintos valores Ψ_{ij} son, para $i = 0$

$$\Psi_{00}^2 = \frac{1}{2\pi} \left[\|y_{01} - y_{00}\|^2 + \|y_{10} - y_{00}\|^2 + \frac{\|y_{11} - y_{00}\|^2}{2} \right]$$

$$\Psi_{01}^2 = \frac{1}{2\pi} \left[\|y_{00} - y_{01}\|^2 + \|y_{02} - y_{01}\|^2 + \|y_{11} - y_{01}\|^2 + \frac{\|y_{10} - y_{01}\|^2}{2} + \frac{\|y_{12} - y_{01}\|^2}{2} \right]$$

$$\Psi_{02}^2 = \frac{1}{2\pi} \left[\|y_{01} - y_{02}\|^2 + \|y_{12} - y_{02}\|^2 + \frac{\|y_{11} - y_{02}\|^2}{2} \right]$$

para $i = 1$

$$\Psi^2_{10} = \frac{1}{2\pi}\left[\|y_{00} - y_{10}\|^2 + \|y_{11} - y_{10}\|^2 + \|y_{20} - y_{10}\|^2 + \frac{\|y_{01} - y_{10}\|^2}{2} + \frac{\|y_{21} - y_{10}\|^2}{2}\right]$$

$$\Psi^2_{11} = \frac{1}{2\pi}\left[\|y_{01} - y_{11}\|^2 + \|y_{10} - y_{11}\|^2 + \|y_{21} - y_{11}\|^2 + \|y_{12} - y_{11}\|^2 + \frac{\|y_{00} - y_{11}\|^2}{2} + \frac{\|y_{20} - y_{11}\|^2}{2} + \frac{\|y_{22} - y_{11}\|^2}{2} + \frac{\|y_{02} - y_{11}\|^2}{2}\right]$$

$$\Psi^2_{12} = \frac{1}{2\pi}\left[\|y_{02} - y_{12}\|^2 + \|y_{11} - y_{12}\|^2 + \|y_{22} - y_{12}\|^2 + \frac{\|y_{01} - y_{12}\|^2}{2} + \frac{\|y_{21} - y_{12}\|^2}{2}\right]$$

y, por último, para $i = 2$

$$\Psi^2_{20} = \frac{1}{2\pi}\left[\|y_{10} - y_{20}\|^2 + \|y_{21} - y_{20}\|^2 + \frac{\|y_{11} - y_{20}\|^2}{2}\right]$$

$$\Psi^2_{21} = \frac{1}{2\pi}\left[\|y_{11} - y_{21}\|^2 + \|y_{20} - y_{21}\|^2 + \|y_{22} - y_{21}\|^2 + \frac{\|y_{10} - y_{21}\|^2}{2} + \frac{\|y_{12} - y_{21}\|^2}{2}\right]$$

$$\Psi^2_{22} = \frac{1}{2\pi}\left[\|y_{12} - y_{22}\|^2 + \|y_{21} - y_{22}\|^2 + \frac{\|y_{11} - y_{22}\|^2}{2}\right]$$

3.1. Discretización en 3 dimensiones

Para el caso concreto de 3 dimensiones, tenemos en total 26 posibles direcciones. Sea $X = (x_{ijk})_{3\times3\times3}$ un cubo de nodos, y sea $Y = (y_{ijk})_{3\times3\times}$ sus vectores asociados. En este caso, cada dirección viene determinada por 2 ángulos, el azimutal $\theta \in [0, 2\pi]$,

y la colatitud $\varphi \in [0, \pi]$. Entonces, los ángulos pueden ser $\theta_i = \{0, \pi/4, \ldots, 7\pi/4\}$ y $\varphi_k = \{0, \pi/4, \pi/2, 3\pi/4, \pi\}$, donde se toman todas combinaciones posibles entre los dos ángulos menos para φ_0 y φ_4, que van independientes puesto que se tratan de las direcciones $u_{00} = (0, 0, 1)$ y $u_{04} = (0, 0, -1)$. Cada dirección $u_{ik} = (u^1_{ik}, u^2_{ik}, u^3_{ik})$ viene determinada por las ecuaciones

$$u^1_{ik} = \cos(\theta_i)\sin(\varphi_k)$$

$$u^2_{ik} = \sin(\theta_i)\sin(\varphi_k)$$

$$u^1_{ik} = \cos(\varphi_k)$$

De la misma forma que para el caso de 2 dimensiones, se suponen los nodos igualmente espaciados, con distancias $h > 0$, excepto las diagonales que se diferencian en dos caso: aquellas en las que el ángulo azimutal tenga como subíndice i un número impar y $k = 2$, o si i es par y k es impar, que tendrán como valor $h^* = h\sqrt{2}$; mientras que cuando i sea impar y k impar también, tendremos $h^{**} = h\sqrt{3}$. En resumen, queda algo análogo a (3), con el detalle de que ahora nuestros nodos tienen 3 coordenadas

$$\Psi(x_{ijk}) \approx \Psi_{ijk} = \sqrt{\frac{1}{4\pi} \sum_{u \in N_{ijk}} \frac{\|F(x_{ijk} + h(u) \cdot u) - F(x_{ijk})\|^2}{h(u)^2}}$$

donde N_{ijk} representa lo mismo que en (3), y la función $h(u)$ ahora es tal que

$$h(u_{ik}) := \begin{cases} h\sqrt{2} & \text{si } (k = 2 \wedge 2 \nmid i) \vee (2 \nmid k \wedge 2|i) \\ h\sqrt{3} & \text{si } 2 \nmid k, i \\ h & \text{otro caso} \end{cases}$$

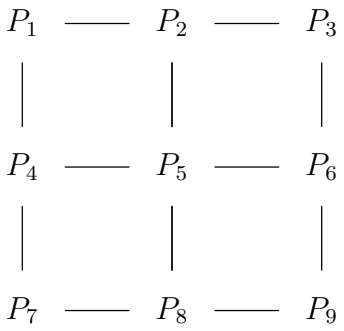
3.2. Máximo teórico

Supongamos m puntos distintos $\{P_1, P_2, \ldots, P_m\} \subset \mathbb{R}^n$ y consideremos sus respectivos vectores unitarios $\{v_1, v_2, \ldots, v_m\}$. Nuestro objetivo es maximizar la función

$$f(v_1, v_2, \ldots, v_m) = \sum_{\substack{(i,j) \in A \\ i \neq j}} \frac{\|v_i - v_j\|^2}{\|P_i - P_j\|^2} \tag{4}$$

donde $A \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\}$ un conjunto determinado.
Generalmente f se puede subdividir en una suma de funciones cuyos máximos comunes se

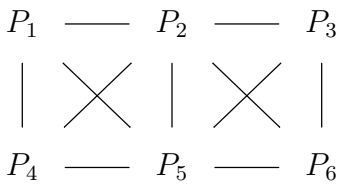
alcanzan en cierta m-tupla $v = (v_1, v_2, \ldots, v_m)$. Fundamentalmente nos interesa el caso en $\mathbb{R}^2$, cuando los puntos $\{P_1, P_2, \ldots, P_m\}$ estan dispuestos como en el HDgrid, por ejemplo en la disposición



, tal que en cada cuadrado que lo forma, dado $(i, j) \in A$, o bien $\|P_i - P_j\|^2 = l^2$, siendo l la longitud del lado del cuadrado; o bien $\|P_i - P_j\|^2 = 2l^2$ cuando los puntos son diagonalmente opuestos en el cuadrado.
En ese caso, dividimos f como

$$f(v_1, v_2, \ldots, v_m) = \frac{1}{2l^2} \sum_{k=1}^{r \cdot s} \left(\sum_{(i,j) \in A_k} \|v_i - v_j\|^2 \right) + \frac{1}{2l^2} \sum_{(i,j) \in B} \|v_i - v_j\|^2$$

donde A_k esta formado por los pares de vértices adyacentes del k-ésimo cuadrado de la disposición, el cual tiene $r \cdot s$ cuadrados y B está formado por los pares de vértices que forman algún lado exterior de la disposición, por ejemplo para el siguiente caso



se tendría que los conjuntos de índices se toman como

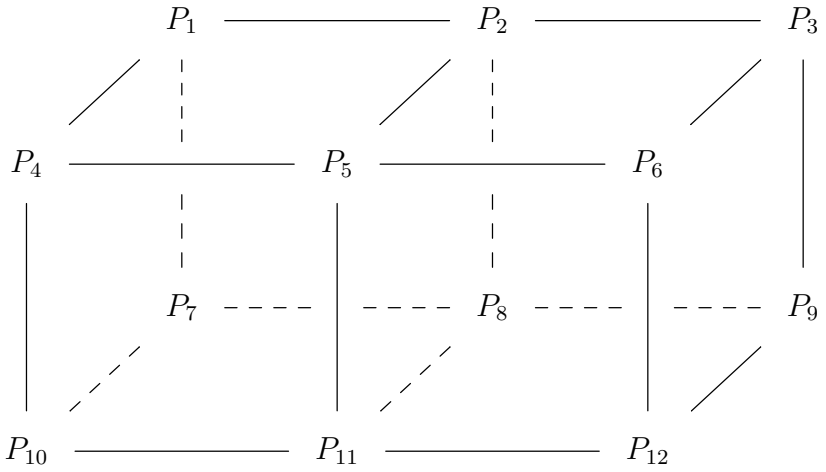
$$A_1 = \{(1,2), (1,4), (1,5), (2,4), (2,5), (4,5)\}$$

$$A_2 = \{(2,3), (2,5), (2,6), (3,5), (3,6), (5,6)\}$$

$$B = \{(1,2), (2,3), (3,6), (1,4), (4,5), (5,6)\}$$

Para el caso en $\mathbb{R}^3$, si lo planteamos análogamente, podemos suponer una disposición de puntos que forman un prisma rectangular que es unión de cubos, por ejemplo, de la

siguiente forma



donde evaluamos suma de cuadrados $\|v_i - v_j\|^2 / \|P_i - P - j\|^2$ con cada par (i, j) correspondiente a pares de vértices del mismo cubo.
Por tanto, en cada cubo tenemos 8 aristas con $\|P_i - P_j\|^2 = l^2$ siendo l la longitud del lado, 12 diagonales en las caras del cubo con $\|P_i - P_j\|^2 = 2l^2$, y 4 diagonales principales del cubo con $\|P_i - P_j\|^2 = 3l^2$. Aquí la partición que planteamos de la función f a maximizar es de la forma

$$f(v_1, v_2, \ldots, v_m) = \frac{1}{4l^2} \sum_{k=1}^{r \cdot s \cdot t} \left(\sum_{(i,j) \in A_k} \|v_i - v_j\|^2 \right) + \frac{1}{4l^2} \sum_{k=1}^{N} \left(\sum_{(i,j) \in B_k} \|v_i - v_j\|^2 \right)$$
$$+ \frac{1}{4l^2} \sum_{(i,j) \in C} \|v_i - v_j\|^2 + \frac{1}{12l^2} \sum_{(i,j) \in D} \|v_i - v_j\|^2$$

donde A_k está formado por las parejas de índices asociados a los 8 vértices del k-ésimo cubo del grid, que tiene $r \cdot s \cdot t$ cubos, B_k está formado por las parejas asociadas a los 4 vértices de la k-ésima cara exterior, C está formado por aquellas parejas de índices que forman un lado de una de las arístas del prisma del mallado, y D está formado por las parejas de índices de diagonales principales de un cuadrado cualquiera que forme la malla. A modo de ejemplo principal, suponiendo el caso anterior, se tiene que algunos de los conjuntos son

$$A_1 = \{(1,2), (1,4), (1,5), (1,7), (1,8), (1,10), (1,11),$$
$$(2,4), (2,5), (2,7), (2,8), (2,10), (4,5), (4,7), (4,10), (4,11),$$
$$(5,7), (5,8), (5,10), (5,11), (7,8), (7,10), (7,11),$$

$$(8,10),(8,11),(10,11)\}$$

$$B_1 = \{(1,2),(1,4),(1,5),(2,4),(2,5),(4,5)\}$$

$$C = \{(1,2),(1,4),(1,7),(2,3),(3,6),(3,9),(4,5),(5,6),(6,12),$$

$$(7,8),(7,10),(8,9),(9,12),(10,11),(11,12)\}$$

$$D = (D_1 \cup ((1,1) + D_1))$$

donde $(1,1) + D_1$ es la traslación del conjunto por el vector $(1,1)$, con

$$D_1 = \{(1,11),(2,10),(4,8),(5,9)\}$$

Para ambos casos podemos obtener el valor máximo, la clave esta en maximizar los sumandos del tipo

$$g(v_1, v_2, \ldots, v_k) = \sum_{\substack{(i,j)\in\{1,2,\ldots,k\}^2 \\ i<j}} \|v_i - v_j\|^2 \tag{5}$$

sujeto a que $\|v_i\| = 1, i \in \{1,2,\ldots,k\}$, ya que el resto de sumandos de la partición van a ser más simples de maximizar. Para ello se necesita el siguiente lema

Lema 3.1 *La función g del problema (5) alcanza su valor máximo de k^2 en $\mathbb{R}^2$ en cualquier k-tupla $\{v_1, v_2, \ldots, v_k\}$ de vectores unitarios que cumpla que*

$$\sum_{i=1}^{k} v_i = 0$$

Demostración. En primer lugar observamos que, dados $\|v_i\| = \|v_j\| = 1$, se cumple que

$$\|v_i - v_j\|^2 = 2 - 2\langle v_i, v_j\rangle = 2(1 - \cos(\widehat{v_i v_j}))$$

donde $\alpha_{ij} := \widehat{v_i v_j}$ es el ángulo formado entre los vectores v_i y v_j. Por tanto

$$\begin{aligned} g(v_1, v_2, \ldots, v_k) &= 2 \sum_{\substack{(i,j)\in\{1,2,\ldots,k\}^2 \\ i<j}} (1 - \cos(\alpha_{ij})) \\ &= 2\left(\frac{k(k-1)}{2} - \sum_{\substack{(i,j)\in\{1,2,\ldots,k\}^2 \\ i<j}} \cos(\alpha_{ij})\right) \\ &= k(k-1) - \sum_{\substack{(i,j)\in\{1,2,\ldots,k\}^2 \\ i\neq j}} \langle v_i, v_j\rangle \end{aligned} \tag{6}$$

teniendo en cuenta que el sumando se puede escribir como

$$\sum_{\substack{(i,j)\in\{1,2,\ldots,k\}^2 \\ i\neq j}} \langle v_i, v_j\rangle = \langle \sum_{i=1}^{k} v_i, \sum_{j=1}^{k} v_j\rangle - \sum_{i=1}^{k}\langle v_i, v_i\rangle$$

entonces si sutituimos en (6) y teniendo en cuenta que $v = \sum_{i=1}^{k} v_i$, se tiene

$$\begin{aligned} g(v_1, v_2, \ldots, v_k) &= k(k-1) - \langle v, v\rangle + k \\ &= k^2 - \langle v, v\rangle \end{aligned}$$

luego g alcanza su valor máximo k^2 cuando $v = 0$. ■

Aplicando el lema al caso del HDgrid para el caso bidimensional obtenemos el siguiente teorema

Teorema 3.2 *Sean* $\{P_{ij}/i = 0, 1, \ldots, r; j = 0, 1, \ldots, s\}$ *puntos distribuidos en el mallado del HDgrid. El valor máximo de la heterogeneidad cuadrática media*

$$\mathcal{V} = \left(\frac{1}{(r+1)(s+1)} \sum_{i=0}^{r}\sum_{j=0}^{s} \Psi_{ij}^2\right)^{1/2}$$

es

$$\mathcal{V}_{\max} = \frac{2}{l\sqrt{\pi}} \left(\frac{r}{r+1} + \frac{s}{s+1}\right)^{1/2}$$

donde $l = \|P_{00} - P_{01}\|$ *la longitud de sus lados.*
Además, dicho máximo se alcanza en la configuración de vectores unitarios

$$v_{ij} = (-1)^{i+j} u$$

para $i = 0, 1, \ldots, r$, $j = 0, 1, \ldots, s$, *con* $\|u\| = 1$.

Demostración. En primer lugar observamos que nuestro problema es equivalente a maximizar

$$f = (r+1)(s+1)\pi\mathcal{V}^2$$

en el cual la división por 2 la hacemos para eliminar la duplicidad de los sumandos puesto que el cociente

$$\frac{\|v_i - v_j\|^2}{\|P_i - P_j\|^2}$$

aparece tanto en Ψ_{ij} como en Ψ_{ji}.
Aplicando ahora la partición, tenemos

$$f = \frac{1}{2l^2} \sum_{k=1}^{r \cdot s} \left(\sum_{(i,j) \in A_k} \|v_i - v_j\|^2 \right) + \frac{1}{2l^2} \sum_{(i,j) \in B} \|v_i - v_j\|^2 \tag{7}$$

donde A_k son los pares de vértices del k-ésimo cuadrado del mallado, y B son los pares de vértices que forman algún lado en la frontera del mallado. Según el lema previo, llamando

$$g_k = \sum_{(i,j) \in A_k} \|v_i - v_j\|^2$$

cada una de las funciones g_k alcanza su valor máximo 16 en la configuración

$$v_{ij} = (-1)^{i+j} u$$

para $i = 0, 1, \ldots, r$, $j = 0, 1, \ldots, s$ y con $\|u\| = 1$. Como hay $r \cdot s$ funciones g_k, el primer sumatior en (7) vale como máximo $16r \cdot s$. Respecto al segundo sumando, hay $2(r + s)$ lados de los cuadrados situados en la frontera del mallado, y el máximo de cada término $\|v_i - v_j\|^2$ es 4, alcanzándose también en la configuración mencionada. Por lo tanto el máximo del segundo sumatorio en (7) es de $8(r + s)$, luego en total se tiene que

$$f_{\max} = \frac{4}{l^2}(r + s + 2rs)$$

que al despejar resulta en

$$\mathcal{V}_{\max} = \left(\frac{4}{l^2 \pi} \left[\frac{r + s + 2rs}{(r + 1)(s + 1)} \right] \right)^{1/2}$$

■

Si sustituimos los valores para el caso concreto del *HDgrid* de tamañano 3×3, obtenemos

$$\mathcal{V}_{\max} = \frac{4}{3\sqrt{2\pi}} \sqrt{6} \approx 1{,}3029$$

En el caso de que se quisiera tener un valor normalizado dentro del intervalo $[0, 1]$, simplemente se tomaría el cociente sobre este valor máximo, es decir

$$VFH = \frac{\mathcal{V}}{\mathcal{V}_{\max}} \in [0, 1]$$

4. Simulaciones numéricas

En esta sección se presentaran las simulaciones realizadas sobre la métrica. Para empezar, se quería estudiar como evolucionaba la métrica según la heterogeneidad de los mapas vectoriales, para ello se tomaron 100 mapas de 3×3 donde cada elemento se escoge de una distribución uniforme en el intervalo $[-\alpha, \alpha]$, donde $\alpha \in \{0, 5, \ldots, 180\}$ son los grados. Sobre estos mapas se calcula su heterogeneidad media y después se representan sobre un diagrama de cajas. Además de lo anterior, tambiés se hacen 1000 simulaciones más para cada uno de los α y luego se calcula la media de todas ellas. En la figura 3 encontramos la gráfica de estas simulaciones, donde la recta superior es el valor máximo calculado en el apartado anterior. Como podemos observar, a medida que aumenta el valor de α también lo hace la heterogeneidad media, hasta $\alpha \geq 100$ que es entonces cuando las medias de las heterogeneidades de los mapas aleatorios empiezan a tener un parecido mayor, y a partir de $\alpha = 140$ las medias se estabilizan. Como podemos observar, los resultados son similares que aquellos que presentaron en el trabajo original.

Además, también se realizaron simulaciones para el caso 3-dimensional, en este tomamos los ángulos azimutal y la colatitud $\theta_i, \varphi_k \in \{1, 5, \ldots, 180\}$ en grados, para cada una de las simulaciones, en este caso en concreto solo se calcularon las medias de las 100 simulaciones para cada uno de los pares de ángulos. En la figura 4 podemos encontrar su representación gráfica.

Además, podemos comprobar numéricamente que el valor máximo que alcanza la métrica es $\mathcal{V}_{\max}$.

5. Conclusión

Cuando se propuso la realización de este trabajo, en las reuniones se fijaron 3 objetivos importantes: la formulación continua del trabajo, el cálculo de un máximo teórico y la formulación del problema en n-dimensiones. Sobre el primero de los objetivos, existe una formulación continua, consistente y generalizada para casos en varias dimensiones, con lo que el tercero de los objetivos se puede considerar superado. Sobre el segundo, podemos comprobar que existe un máximo, que se ha comprobado tanto de forma teórica como empírica, y que esto nos permite definir un marcador entre 0 y 1 que sirva como apoyo para futuras intervenciones.

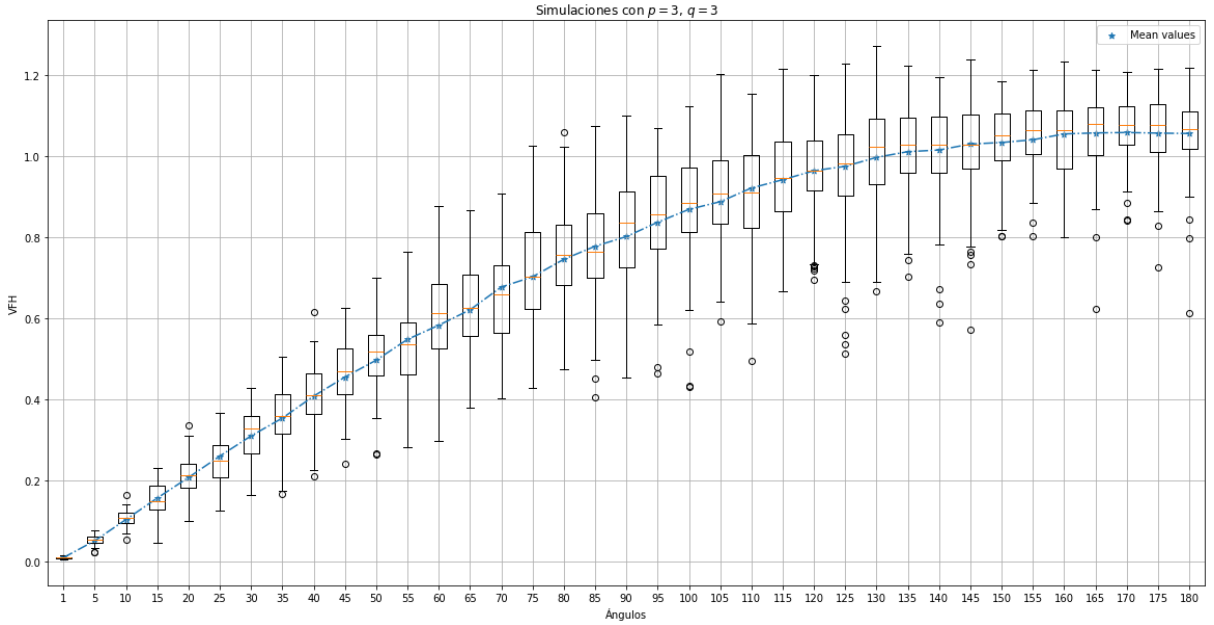


Figura 3: Simulaciones sobre mapas vectoriales aleatorios.

Además, se presentará la definición y el cálculo de su máximo teórico a revistas del ámbito matemático, el artículo en cuestión es [6].

Sobre posibles trabajos futuros, existe la posibilidad de aplicar el trabajo para la interpolación de señales y la reconstrucción de los campos vectoriales correspondientes al campo eléctrico del corazón, con la finalidad de poder crear simulaciones por ordenador y estudiar con más detalle las intervenciones futuras para los pacientes.

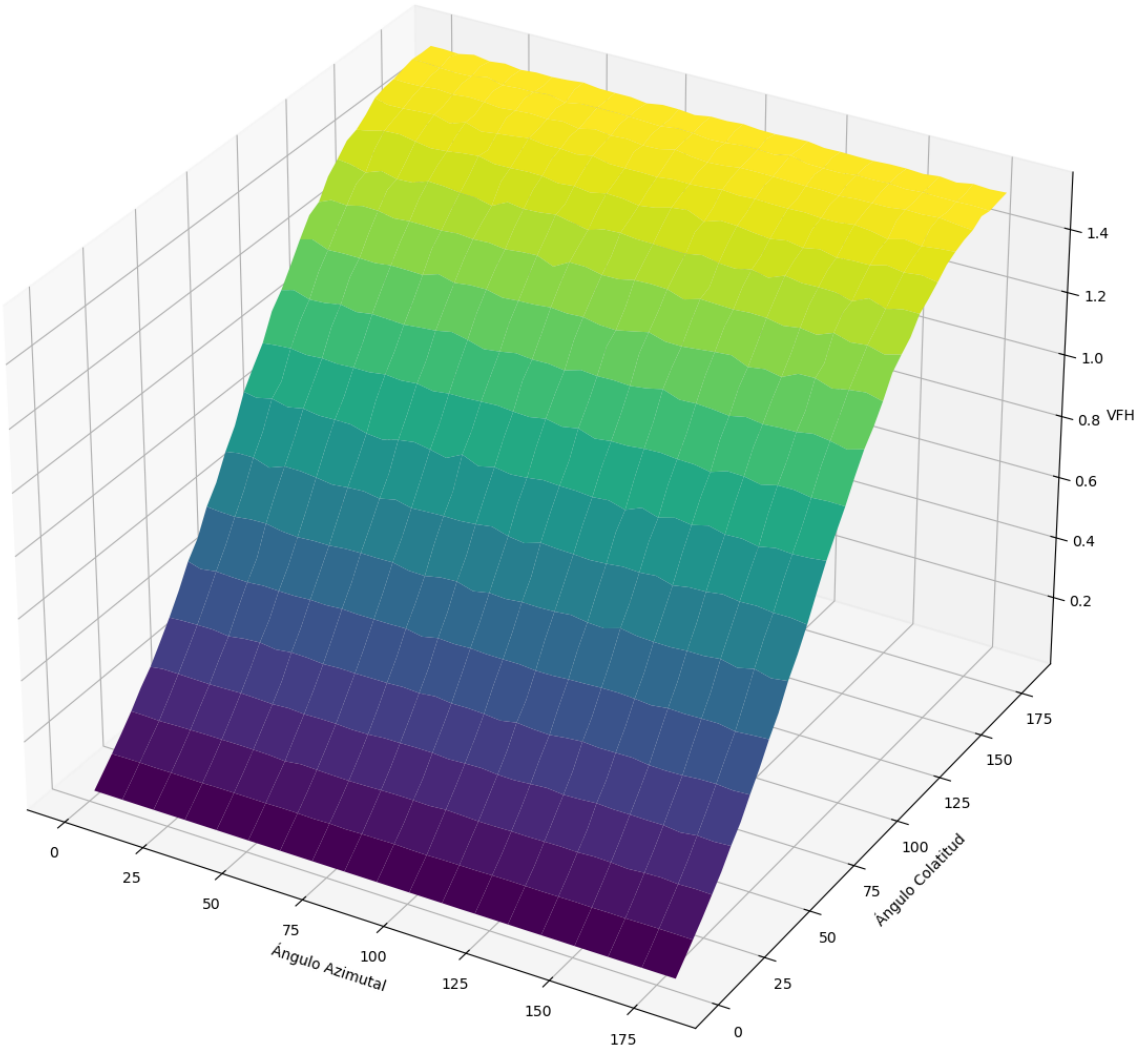


Figura 4: Simulaciones sobre mapas vectoriales aleatorios para el caso 3-dimensional.

Referencias

[1] Apostol, T.M.: *Calculus. Vol. II: Multi-variable Calculus and Linear Algebra, with Applications to Differential Equations and Probability.* Xerox College Publ. 1969.

[2] Aragón, F.J. and Goberna, M.A. and López, M.A. and Rodríguez, M.M.L.: *Nonlinear Optimization.* Springer Undergraduate Texts in Mathematics and Technology. 2019.

[3] D. Curtis Deno and Abhishek Bhaskaran and Dennis J. Morgan and Fikri Goksu and Katherine Batman and Gregory K. Olson and Karl Magtibay and Sachin Nayyar and Andreu Porta-Sánchez and Michael A. Laflamme and Stéphane Massé and Prashant Aukhojee and Krishnakumar Nair and Kumaraswamy Nanthakumar: *High-resolution, live, directional mapping* in Heart Rhythm, vol. 17, pp. 1621-1628, 2020.

[4] Galbis, A. and Maestre, M: *Vector Analysis Versus Vector Calculus.* Springer. 2012.

[5] Pancorbo, L. and Ruipérez-Campillo, S. and Tormos, A. and Guill, A. and Cervigón, R. and Alberola, A. and Chorro, F. J. and Millet, J. and Castells, F.: *Vector Field Heterogeneity for the Assessment of Locally Disorganised Cardiac Electrical Propagation Wavefronts From High-Density Multielectrodes,* in IEEE Open Journal of Engineering in Medicine and Biology, vol. 5, pp. 32-44, 2024.

[6] Peris, A. and Castells, F. and Millet, J. and Ramirez-Muñoz, J.M.: *Maximal vector field heterogenity discrete metric.*

A. Código en python

Esta sección consiste en explicar el funcionamiento del código elaborado en *python* para realizar los cálculos de la heterogeneidad media. Se han usado para ello el paquete de `numpy`. Por lo explicado con anterioridad, para el caso 2-dimensional tenemos el array

$$Y = \begin{array}{|c|c|c|c|c|c|} \hline y_{00} & y_{01} & y_{02} & \cdots & y_{0(q-1)} & y_{0q} \\ \hline y_{10} & y_{11} & y_{12} & \cdots & y_{1(q-1)} & y_{1q} \\ \hline y_{20} & y_{21} & y_{22} & \cdots & y_{2(q-1)} & y_{2q} \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \hline y_{(p-1)0} & y_{(p-1)1} & y_{(p-1)2} & \cdots & y_{(p-1)(q-1)} & y_{(p-1)q} \\ \hline y_{p0} & y_{p1} & y_{p2} & \cdots & y_{p(q-1)} & y_{pq} \\ \hline \end{array}$$

donde cada $y_{ij} = (y_{ij}^1, y_{ij}^2)$. Entonces si quisieramos calcular la discretización de la dirección $u_0 = (1, 0)$ para el caso de y_{ij} tendríamos que

$$D_{u_0} y_{ij} = \frac{\|y_{ij} - y_{i(j+1)}\|^2}{h^2}$$

Entonces si tomamos ahora el array

$$Y_{u_0} = \begin{array}{|c|c|c|c|c|c|} \hline y_{01} & y_{02} & y_{03} & \cdots & y_{0q} & y_{0q} \\ \hline y_{11} & y_{12} & y_{13} & \cdots & y_{1q} & y_{1q} \\ \hline y_{21} & y_{22} & y_{23} & \cdots & y_{2q} & y_{2q} \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \hline y_{(p-1)1} & y_{(p-1)2} & y_{(p-1)3} & \cdots & y_{(p-1)q} & y_{(p-1)q} \\ \hline y_{p1} & y_{p2} & y_{p3} & \cdots & y_{pq} & y_{pq} \\ \hline \end{array}$$

y se calcula la diferencia $Y - Y_{u_0}$ entonces nos queda

$$Y - Y_{u_0} = \begin{array}{|c|c|c|c|c|} \hline y_{00} - y_{01} & y_{01} - y_{02} & \cdots & y_{0(q-1)} - y_{0q} & 0 \\ \hline y_{10} - y_{11} & y_{11} - y_{12} & \cdots & y_{1(q-1)} - y_{1q} & 0 \\ \hline y_{20} - y_{21} & y_{21} - y_{22} & \cdots & y_{2(q-1)} - y_{2q} & 0 \\ \hline \vdots & \vdots & \ddots & \vdots & \vdots \\ \hline y_{(p-1)0} - y_{(p-1)1} & y_{(p-1)1} - y_{(p-1)2} & \cdots & y_{(p-1)(q-1)} - y_{(p-1)q} & 0 \\ \hline y_{p0} - y_{p1} & y_{p1} - y_{p2} & \cdots & y_{p(q-1)} - y_{pq} & 0 \\ \hline \end{array}$$

Este cálculo de arrays se encuentra para cada una de las direcciones en la función `diferenceCalculator()`.

Por la definición de norma y teniendo en cuenta que $(Y-Y_{u_0})_k$ se tratan de la diferencia de las k-ésimas coordenadas con $k \in \{1,2\}$, entonces

$$\|Y - Y_{u_0}\|^2 = (Y - Y_{u_0})_1^2 + (Y - Y_{u_0})_2^2$$

Una vez calculada la norma para cada una de las direcciones, entonces se suman los resultados para cada una de ellas, ponderadas por 0,5 aquellas direcciones que sean diagonales, después se calcula la raíz cuadrada de la suma total. Este cálculo de la matriz con las heterogeneidades en cada punto se encuentra en la función `psiCalculator`. Después, la heterogeneidad se calcula mediante la función `VFH`.

El siguiente código se trata del script para calcular la heterogeneidad de manera discreta para el caso en 2 dimensiones.

```
'''
Codigo para el calculo de la metrica de VFH de manera discreta.
'''
import numpy as np

def diferenceCalculator(X, p, q):

    # Diferencia horizontal
    Xhr = np.copy(X)
    Xhl = np.copy(X)

    Xhr[:, :q-1] = X[:, 1:]
    Xhl[:, 1:] = X[:, :q-1]

    r =  (X-Xhr) # matriz de diferencias derecha
    l =  (X-Xhl) # matriz de diferencias izquierda

    # Diferencia vertical

    Xvu = np.copy(X)
    Xvd = np.copy(X)

    Xvu[1:, :] = X[:p-1, :]
    Xvd[:p-1, :] = X[1:, :]
```

```
    u =  (X-Xvu) # matriz de diferencias arriba
    d = (X-Xvd) # matriz de diferencias abajo


    # Diferencia diagonal 1
    Xd1u = np.copy(X)
    Xd1d = np.copy(X)

    Xd1u[1:, :q-1] = X[:p-1, 1:]
    Xd1d[:p-1, 1:] = X[1:, :q-1]

    d1_u = (X-Xd1u) # matriz de diferencias diagonal 1 arriba
    d1_d = (X-Xd1d) # matriz de diferencias diagonal 1 abajo

    # Diferencia diagonal 2
    Xd2r = np.copy(X)
    Xd2l = np.copy(X)

    Xd2r[0:p-1, 0:q-1] = X[1:, 1:]
    Xd2l[1:,1:] = X[0:p-1,0:q-1]

    d2_r = (X-Xd2r)  # matriz de diferencias diagonal 2 derecha
    d2_l = (X-Xd2l) # matriz de diferencias diagonal 2 izquierda

    return r, l, u, d, d1_u, d1_d, d2_r, d2_l

def norm(X, Y):
    a = np.multiply(X, X)
    b = np.multiply(Y, Y)
    return np.sqrt(a+b)


def PsiCalculator(X, Y, p, q):
    Xr, Xl, Xu, Xd, Xd1_u, Xd1_d,\
        Xd2_r, Xd2_l = diferenceCalculator(X, p ,q)
```

```
        Yr, Yl, Yu, Yd, Yd1_u, Yd1_d,\
            Yd2_r, Yd2_l = diferenceCalculator(Y, p ,q)

        r = norm(Xr, Yr)**2
        l = norm(Xl, Yl)**2

        u = norm(Xu, Yu)**2
        d = norm(Xd, Yd)**2

        d1_u = norm(Xd1_u, Yd1_u)**2
        d1_d = norm(Xd1_d, Yd1_d)**2

        d2_r = norm(Xd2_r, Yd2_r)**2
        d2_l = norm(Xd2_l, Yd2_l)**2

        Psi = r + l + u + d \
            + 0.5*( d1_u + d1_d + d2_r + d2_l)


        return np.sqrt(Psi/(2*np.pi))


    def VFH(X, Y, p, q):
        Psi = PsiCalculator(X, Y, p, q)
        Psi2 =  np.multiply(Psi, Psi)
        VFH = np.sum(Psi2)/(p*q)

        return np.sqrt(VFH)
```

El siguiente script es la versión en 3 dimensiones del anterior. En este caso solo es posible para cubos de $3 \times 3 \times 3$.

```
import numpy as np

from numpy import pi


def diferenceCalculator(X, nx, ny, nz):
    '''
    Esta funcion calcula los valores en diferencias.

    n := es el tamanyo de los tres ejes en array [z, y, x]

    X := es la matriz con los datos, de dimension nx*ny*nz*

    Las arrays se construyen por capas hacia abajo, es decir,
    si tenemos una array 3-dimensional M, entonces M[0,:,:]
    es la primera capa
    '''

    # Diferencia en el eje x (left - right)
    Xxr = np.copy(X)
    Xxl = np.copy(X)

    Xxr[:, :, :ny-1] = X[:, :, 1:]
    Xxl[:, :, 1:] = X[:, :, :ny-1]

    xr =  (X-Xxr) # matriz de diferencias right
    xl =  (X-Xxl) # matriz de diferencias left


    # Diferencia en el eje y (front - back)
    Xyf = np.copy(X)
    Xyb = np.copy(X)

    Xyf[:, :nx-1, :] = X[:, 1:, :]
```

```
Xyb[:, 1:, :] = X[:, :nx-1, :]


yf =  (X-Xyf) # matriz de diferencias front
yb = (X-Xyb) # matriz de diferencias back


# Diferencia en el eje z (up - down)
Xzu = np.copy(X)
Xzd = np.copy(X)

Xzu[1:, :, :] = X[:nz-1, :, :]
Xzd[:nz-1, :, :] = X[1:, :, :]

zu =  (X-Xzu) # matriz de diferencias up
zd = (X-Xzd) # matriz de diferencias down


# Diferencia diagonal 1 sobre el plano
Xd1u = np.copy(X)
Xd1d = np.copy(X)

Xd1u[:, 1:, :ny-1] = X[:, :nx-1, 1:]
Xd1d[:, :nx-1, 1:] = X[:, 1:, :ny-1]

d1_u = (X-Xd1u) # matriz de diferencias diagonal sobre un plano 1
d1_d = (X-Xd1d) # matriz de diferencias diagonal sobre un plano 2

# Diferencia diagonal 2
Xd2r = np.copy(X)
Xd2l = np.copy(X)

Xd2r[:, 0:nx-1, 0:ny-1] = X[:, 1:, 1:]
Xd2l[:,1:,1:] = X[:, 0:nx-1,0:ny-1]

d2_r = (X-Xd2r)  # matriz de diferencias diagonal 2 derecha
```

```
d2_l = (X-Xd2l) # matriz de diferencias diagonal 2 izquierda


# Diferencia diagonal front-back up-down
Xdfb_1 = np.copy(X)
Xdfb_2 = np.copy(X)

Xdfb_1[:nz-1 , 1:, :] = X[1:, :nx-1, :]
Xdfb_2[1:, :nx-1, :] = X[:nz-1 , 1:, :]

dfb_1 = (X-Xdfb_1) # matriz de diferencias front-back
dfb_2 = (X-Xdfb_2) # matriz de diferencias back-front

# Diferencia diagonal front-back down-up
Xdfb_3 = np.copy(X)
Xdfb_4 = np.copy(X)

Xdfb_3[:nz-1, :nx-1, :] = X[1:, 1:, :]
Xdfb_4[1:, 1:, :] = X[:nz-1 , :nx-1, :]

dfb_3 = (X-Xdfb_3) # matriz de diferencias front-back down
dfb_4 = (X-Xdfb_4) # matriz de diferencias back-front up


# Diferencia diagonal left-right
Xdlr_1 = np.copy(X)
Xdlr_2 = np.copy(X)

Xdlr_1[:nz-1 ,: , :ny-1] = X[1:, :, 1:]
Xdlr_2[1:, :, 1:] = X[:nz-1 ,: , :ny-1]

dlr_1 = (X-Xdlr_1)  # matriz de diferencias diagonal left right
dlr_2 = (X-Xdlr_2) # matriz de diferencias diagonal right left


# Diferencia diagonal left-right
```

```
Xdlr_3 = np.copy(X)
Xdlr_4 = np.copy(X)

Xdlr_3[:nz-1 ,: , 1:] = X[1:, :, :ny-1]
Xdlr_4[1:, :, :ny-1] = X[:nz-1 ,: , 1:]

dlr_3 = (X-Xdlr_3)  # matriz de diferencias diagonal left right
dlr_4 = (X-Xdlr_4) # matriz de diferencias diagonal right left


# Diferencia diagonal left/front-right/back
Xdcornerfrontleft_1 = np.copy(X)
Xdcornerfrontleft_2 = np.copy(X)

Xdcornerfrontleft_1[:nz-1 , :nx-1 , :ny-1] = X[1:, 1:, 1:]
Xdcornerfrontleft_2[1:, 1:, 1:] = X[:nz-1 , :nx-1 , :ny-1]


# matriz de diferencias diagonal left/front right/back
dcornerfrontleft_1 = (X-Xdcornerfrontleft_1)

# matriz de diferencias diagonal left/front right/back
dcornerfrontleft_2 = (X-Xdcornerfrontleft_2)


# Diferencia diagonal left/front-right/back
Xdcornerfrontleft_3 = np.copy(X)
Xdcornerfrontleft_4 = np.copy(X)

Xdcornerfrontleft_3[1: , :nx-1 , :ny-1] = X[:nz-1, 1:, 1:]
Xdcornerfrontleft_4[:nz-1, 1:, 1:] = X[1: , :nx-1 , :ny-1]


# matriz de diferencias diagonal left/front right/back
dcornerfrontleft_3 = (X-Xdcornerfrontleft_3)
```

```
# matriz de diferencias diagonal left/front right/back
dcornerfrontleft_4 = (X-Xdcornerfrontleft_4)


# Diferencia diagonal left/back-right/front
Xdcornerbackleft_1 = np.copy(X)
Xdcornerbackleft_2 = np.copy(X)

Xdcornerbackleft_1[:nz-1 , 1: , :ny-1] = X[1:, :nx-1, 1:]
Xdcornerbackleft_2[1:, :nx-1, 1:] = X[:nz-1 , 1: , :ny-1]


# matriz de diferencias diagonal left/front right/back
dcornebackleft_1 = (X-Xdcornerbackleft_1)

# matriz de diferencias diagonal left/front right/back
dcornerbackleft_2 = (X-Xdcornerbackleft_2)


# Diferencia diagonal left/front-right/back
Xdcornerbackleft_3 = np.copy(X)
Xdcornerbackleft_4 = np.copy(X)

Xdcornerbackleft_3[:nz-1 , :nx-1 , 1:] = X[1:, 1:, :ny-1]
Xdcornerbackleft_4[1:, 1:, :ny-1] = X[:nz-1 , :nx-1 , 1:]


# matriz de diferencias diagonal left/front right/back
dcornerbackleft_3 = (X-Xdcornerbackleft_3)

# matriz de diferencias diagonal left/front right/back
dcornerbackleft_4 = (X-Xdcornerbackleft_4)


return np.array([xl, xr, yf, yb, zu, zd, d1_u, d1_d, d2_l, d2_r,  dfb_1, dfb_
                dfb_3, dfb_4, dlr_1, dlr_2, dlr_3, dlr_4,\
```

```
                    dcornerfrontleft_1, dcornerfrontleft_2, \
                    dcornerfrontleft_3, dcornerfrontleft_4,\
                    dcornebackleft_1, dcornerbackleft_2,\
                    dcornerbackleft_3, dcornerbackleft_4])


def norm(X, Y, Z):
    a = np.multiply(X, X)
    b = np.multiply(Y, Y)
    c = np.multiply(Z, Z)
    return np.sqrt(a+b+c)


def PsiCalculator(M):

    # Calculo de las matrices de diferencias para cada una de las variables
    # de los nodos
    Xdiference = diferenceCalculator(M[0], 3, 3, 3)
    Ydiference = diferenceCalculator(M[1], 3, 3, 3)
    Zdiference = diferenceCalculator(M[2], 3, 3, 3)

    # Inicializacion del vector con las normas
    s = np.zeros((26, 3,3,3))
    for k in range(26):
        # cada uno de los elementos de s es la norma
        # de las variaciones en los nodos
        s[k] = norm(Xdiference[k, :, :, :], Ydiference[k, :, :, :], Zdiference[k,


    s[6:18] *= 0.5  # ajuste para las diagonales con distancia sqrt(2)
    s[18:] *= 1/3 # ajuste para las diagonales con distancia sqrt(3)

    # Matriz con los valores psi para cada nodo
    psi = np.sum(s, axis=0)
```

```
        return np.sqrt(psi/(2*np.pi))


    def VFH3(M):
        Psi = PsiCalculator(M)
        Psi2 = np.multiply(Psi, Psi)
        VFH = np.sum(Psi2)/27

        return np.sqrt(VFH)
```

Los dos siguientes códigos se tratan de las simulaciones tomando mapas de vectores aleatorios

```
import numpy as np
import numpy.random as rd
import matplotlib.pyplot as plt

from VFH import VFH

rd.seed(2000)

p = 3
q = 3

angulos = np.arange(0, 181, 5)
angulos[0] = 1

i = 0
res = np.zeros((100, 37))
vector = np.zeros((1000, 37))


for k in range(37):
    i = 0
    j = 0

    ang = angulos[k]
    while i < 100:

        random_angles = rd.uniform(-ang, ang, size=(p,q))
        random_angles = (np.pi/180)*random_angles


        X = np.cos(random_angles)
        Y = np.sin(random_angles)

        res[i, k] = VFH(X, Y, p, q)
```

```
        i += 1

    while j < 1000:
        random_angles = rd.uniform(-ang, ang, size=(p,q))
        random_angles = (np.pi/180)*random_angles

        X = np.cos(random_angles)
        Y = np.sin(random_angles)

        vector[j, k] = VFH(X, Y, p, q)
        j += 1

    medias = np.mean(vector, axis=(0))



arange = np.arange(1,38)

# Boxplot de las simulaciones

plt.figure(figsize=(20,10))
plt.boxplot(res)
plt.xticks(arange, labels = angulos)

plt.plot(arange, medias, linestyle = '-.')
plt.scatter(arange, medias, label = 'Mean values', marker = '*')
plt.grid()


plt.title('Simulaciones con $p=3$, $q=3$')
plt.ylabel('VFH')
plt.xlabel('Ángulos')

plt.legend()
plt.show()
```

```
import numpy as np
import sympy as sp
import numpy.random as rd
import numpy.linalg as alg

from numpy import pi

import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm

from VFH3 import VFH3

angulos1 = np.arange(0, 181, 5)
angulos1[0] = 1

angulos2 = np.arange(0, 181, 5)
angulos2[0] = 1


res = np.zeros((100, 37, 37))
vector = np.zeros((1000, 37, 37))

for k in range(37):
    for r in range(37):
        i = 0
        j = 0

        ang1 = angulos1[k]
        ang2 = angulos2[k]

        while i < 100:

        theta = rd.uniform(-ang1, ang1, size=(3,3,3))
        theta = (np.pi/180)*theta
```

```
            delta = rd.uniform(0, ang2, size=(3,3,3))
            delta = (np.pi/180)*delta

            X = np.sin(delta)*np.cos(theta)
            Y = np.sin(delta)*np.sin(theta)
            Z = np.cos(delta)

            M = np.array([X, Y, Z])

            res[i, k, r] = VFH3(M, 3, 3, 3)

            i += 1



medias = np.mean(res, axis=0)
nX, nY = np.meshgrid(angulos1, angulos2)


plt.rcParams['figure.dpi']=100
fig=plt.figure(1, figsize=[15,15] )
ax=fig.add_subplot(projection='3d')
ax.plot_surface(nX,nY,medias,rstride=2,cstride=2,cmap=cm.viridis,linewidth=0)

ax.set_xlabel('Ángulo Azimutal')
ax.set_ylabel('Ángulo Colatitud')
ax.set_zlabel('VFH')


plt.show()
```