Instituto Universitario de Investigación
en Ingeniería de Aragón
Universidad Zaragoza

Universidad de Zaragoza

Facultad de Ciencias:
Máster Universitario en Modelización e Investigación Matemática, Estadística y Computación

Descomposición Ortogonal Propia en registro mediante difeomorfismos

Rubén Muñoz Sierra

Dirigido por:
Dra. Mónica Hernández Giménez
Dra. Elvira Mayordomo Cámara

Ponente:
Dr. José Tomás Alcalá Nalvaiz

12 de septiembre de 2022

Resumen

Los métodos de registro mediante difeomorfismos se han posicionado como los métodos de referencia en el área del registro no-rígido de imágenes médicas. La metodología subyacente proporciona soluciones que mantienen la corrección biológica de los campos de deformación de las anatomías en términos de suavidad y conservación de la topología. El método *Large Deformation Diffeomorphic Metric Mapping* (LDDMM) fue pionero en el año 2005 sentando las bases en el cálculo de registro difeomorfo en el paradigma de grandes deformaciones. El principal problema de LDDMM es su gran carga computacional. El presente trabajo tiene como objetivo el estudio de la capacidad de los métodos denominados *Reduced Order Models* (ROM) para reducir la complejidad computacional de una de las variantes de LDDMM basada en la restricción del problema a campos vectoriales iniciales respetando la ecuación diferencial de Euler-Poincaré (EPDiff). El trabajo se centrará en el estudio del artículo Wen et al., *Data-driven Model Order Reduction For Diffeomorphic Image Registration*. En este, se presenta un ROM denominado *Proper Orthogonal Decomposition* (POD) para reducir la dimensionalidad de la ecuación de Euler-Poincaré. Así, en este trabajo, se estudiará la reproducibilidad el artículo, tanto a nivel teórico como a nivel experimental. Se estudiará la implementación del algoritmo mediante el registro de distintas imágenes, así como la viabilidad de este, mediante la comparación de distintas métricas como son el error final en el registro o el tiempo de computación.

Diffeomorphic registration methods have positioned themselves as the reference methods in the area of non-rigid registration of medical images. The underlying methodology provides solutions that maintain the biological correctness of the deformation fields of anatomies in terms of smoothness and topology preservation. The Large Deformation Diffeomorphic Metric Mapping (LDDMM) method was pioneered in 2005 laying the foundations of diffeomorphic registration computation in the large deformation paradigm. The main problem of LDDMM is its computational burden. The aim of this work is to study the ability of the so-called Reduced Order Models (ROM) methods to reduce the computational complexity of one of the variants of LDDMM based on the restriction of the problem to initial vector fields respecting the Euler-Poincaré differential equation (EPDiff). The paper will focus on the study of the article Wen et al., Data-driven Model Order Reduction For Diffeomorphic Image Registration. In this paper, a ROM called Proper Orthogonal Decomposition (POD) is presented to reduce the dimensionality of the image. POD is presented to reduce the dimensionality of the Euler-Poincaré equation. Thus, in this work, the reproducibility of the article will be studied, both theoretically and experimentally. The implementation of the algorithm will be studied by means of the registration of different images, as well as its viability, by comparing different metrics such as the final error in the registration or the computation time.

Índice general

Capítulo 1

Introducción

El registro de imágenes es una técnica de procesado digital de datos aplicado en imágenes. Se trata de una tarea fundamental en el procesamiento de estas, ya que se utiliza para alinear dos o más imágenes dadas, por ejemplo, en distintos momentos del tiempo, en diferentes orientaciones, o en diferentes puntos de vista. A lo largo de los años, se ha desarrollado una amplia gama de técnicas para diversos tipos de datos y problemas. Una de las áreas que más se beneficia de la investigación en este campo, es la medicina, donde se requiere del constante análisis de imágenes obtenidas mediante diferentes técnicas: radiografía, resonancia magnética (IRM), tomografía por emisión de positrones (PET)... Estas técnicas de registro de imágenes se pueden dividir en dos grandes ramas: el registro rígido, y el registro deformable. En el registro rígido, se trata de llevar a cabo una transformación geométrica (traslación, rotación, afín...) a una imagen para lograr el correcto alineamiento con otra imagen. El registro deformable se basa en que la deformación de la imagen ocurre a nivel local para alinearla con la imagen de referencia, y no a nivel global, como en el caso del registro rígido [1, 2].

El registro deformable de imágenes es una herramienta fundamental en el análisis de imágenes médicas dentro de la disciplina de la Anatomía Computacional [3]. Además, en muchas aplicaciones, se requiere que las transformaciones obtenidas a partir del registro sean difeomorfas. Es decir, diferenciables, biyectivas, y con inversa diferenciable. Los difeomorfismos aseguran ciertas propiedades interesantes a las imágenes transformadas, como son la conservación de la corrección biológica en las imágenes en términos de suavidad y la conservación de la topología. El mapeo métrico difeomorfo de grandes deformaciones o *Large Deformation Diffeomorphic Metric Mapping* (LDDMM) es una formulación variacional para el registro de imágenes difeomorfo ampliamente utilizada. Fue propuesto por primera vez por Beg et al. [4] en el año 2005 sentando las bases para el cálculo de registro difeomorfo en el espacio tangente de la variedad Riemanniana de difeomorfismos.

1.1. Mapeo Métrico Difeomorfo de Grandes Deformaciones

El objetivo del registro [4] es calcular una transformación $\varphi : \Omega \to \Omega$ donde $\Omega \subseteq \mathbb{R}^n$ representa el dominio de las imágenes. En el ámbito del análisis de imágenes médicas se puede trabajar con imágenes 2D, 3D, secuencias e incluso tensores, por lo que n puede tomar un amplio rango de valores enteros. Las imágenes se consideran funciones de cuadrado integrable $I : \Omega \to \mathbb{R}^n$ definidas en Ω. En el problema de registro, se denomina I_0 a la imagen fuente (*source*) o imagen fija (*fixed*) e I_1 a la imagen objetivo (*target*), o

imagen que se deforma (*moving*), respectivamente. La imagen I_0 transformada (*warped*) bajo tal transformación, corresponde a la imagen definida como $\varphi.I_0 = I_0 \circ \varphi^{-1} = I_0(\varphi^{-1})$.

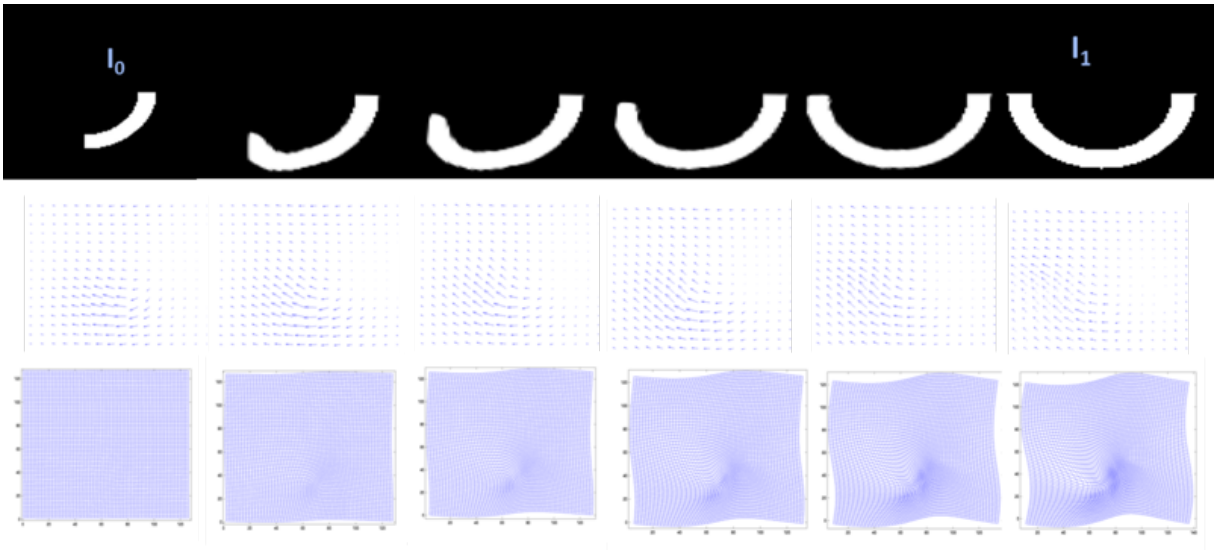


Figura 1.1: Ejemplo de un regristro de imágenes difeomorfo deformable [5].

En la figura (1.1) se ejemplifica un registro de imágenes difeomorfo deformable [5]. En esta aparece una imagen I_0 sobre la cual se realiza una transformación de forma que se asemeje a la imagen I_1. Las imágenes previas a I_1, son las transformaciones intermedias de la imagen I_0 de forma que en cada paso se asemeja cada vez más a la imagen objetivo. En la segunda y tercera fila se muestran el mapa de velocidades de cada una de las transformaciones y el desplazamiento de cada uno de los píxeles de la imagen para llegar a ser dicha imagen, respectivamente.

1.1.1. Paradigma de pequeñas deformaciones

En el paradigma de pequeñas deformaciones, la transformación φ es linearizada y generada a partir de un campo vectorial de desplazamientos $u : \Omega \to \mathbb{R}^n$ tal que $\varphi(x) = x + u(x)$ y $\varphi(x)^{-1} = x - u(x)$ para todos los puntos $x \in \Omega$. Así, la imagen fuente I_0 queda transformada como $I_0(\varphi^{-1}(x)) = I_0(x - u(x)), \forall x \in \Omega$. La efectividad de la transformación se mide mediante la siguiente función de coste:

$$E_{img}(I_0, I_1, \varphi) = \left|\left|I_0 \circ \varphi^{-1} - I_1\right|\right|^2_{L^2} \tag{1.1}$$

donde $||\cdot||_{L^2}$ es la norma L^2 estándar de las funciones de cuadrado integrable $||f||^2_{L^2} = \int_\Omega |f(x)|^2 \, dx$. La transformación óptima es la que minimiza esta función de coste. El campo vectorial óptimo u que genera tal transformación es, entre todas las soluciones posibles, elegido para ser el que presente un mayor control sobre su gradiente:

$$E_{reg}(u) = ||\nabla u||^2_{L^2} \, . \tag{1.2}$$

Así, el campo vectorial es calculado mediante la optimización de la siguiente función de coste:

$$\underset{u}{argmin}\ E_{reg}(u) + \frac{1}{\sigma^2}E_{img}(u) = \underset{u}{argmin}\ ||\nabla u||^2_{L^2} + \frac{1}{\sigma^2}\left|\left|I_0 \circ \varphi^{-1} - I_1\right|\right|^2_{L^2} \tag{1.3}$$

Una de las limitaciones de este método [4], es que no hay restricciones explícitas que aseguren que la transformación sea invertible, algo que es de considerable interés: que la transformación sea, no solo invertible sino que también conserve propiedades de la imagen transformada, como son la suavidad de las curvas, de las superficies, u otras características asociadadas a la anatomía. Así, las transformaciones difeomórficas, las cuales son transformaciones suaves e invertibles con una inversa suave son las óptimas para esta tarea. Restringir la transformación para que sea un difeomorfismo es una elección natural en el ámbito de la anatomía, de forma que las transformaciones se hagan de forma consistente y, además, que la suavidad de los elementos como curvas y superficies se conserven. Los métodos de registro difeomorfo más utilizados pertenecen al paradigma de grandes deformaciones.

1.1.2. Paradigma de grandes deformaciones

Un homeomorfismo en el dominio de la imagen Ω es una función biyectiva e invertible $\varphi : \Omega \to \Omega$ el cual, junto a su inversa φ^{-1}, es invertible. Este homeomorfismo actúa en el espacio Ω tal que $\mathrm{Hom}(\Omega)$. Los homeomorfismos forman un grupo para la ley de la composición $\psi \cdot \varphi \doteq \psi \circ \varphi$. Es más, para cualquier $\varphi \in \mathrm{Hom}(\Omega)$ y cualquier imagen $I : \Omega \to \mathbb{R}^d,\ \varphi.I \doteq I \circ \varphi^{-1}$ define la acción de $\mathrm{Hom}(\Omega)$ sobre el conjunto de todas las imágenes. Sea $\mathcal{G}$ el conjunto de difemorfismos $\mathrm{Diff}(\Omega)$ de cualquier $\varphi \in \mathrm{Hom}(\Omega)$ un subgrupo de $\mathrm{Hom}(\Omega)$, el cual es, junto a su inversa, continuo y diferenciable.

Dadas dos imágenes, el problema de registro en el paradigma de grandes deformaciones *Large Deformation Diffeomorphic Metric Mapping* (LDDMM) consiste en encontrar el elemento particular $\varphi \in \mathcal{G}$ que alinea estas imágenes $I_1 = \varphi.I_0 = I_0 \circ \varphi^{-1}$. Este elemento se calcula como el punto final del flujo asociado al campo vectorial suave y dependiente del tiempo. Sea $v : [0,1] \to V$ un campo vectorial de velocidades dependientes del tiempo, donde V es un espacio de Hilbert de campos vectoriales suaves, y de soporte compacto en Ω. Dejamos que tal campo vectorial de velocidades defina la evolución de una curva $\phi^v : [0,1] \to \mathcal{G}$ vía la ecuación de evolución:

$$\frac{d}{dt}\phi^v_t(x) = v_t\left(\phi^v_t(x)\right), \tag{1.4}$$

donde el superíndice v en ϕ^v es utilizado par denotar explícitamente la dependencia de ϕ con el campo vectorial de velocidades asociado v. El punto inicial de la curva ϕ^v en $t = 0$ es $\phi^v_0 = Id \in G$, donde Id es la transformación identidad $Id(x) = x,\ \forall x \in \Omega$. El punto final de la curva ϕ^v en el tiempo $t = 1$ es el difeomorfismo particular $\phi^v_1 = \varphi \in G$ que une los conjuntos de datos I_0 e I_1 tales que $I_1 = I_0 \circ \varphi^{-1}$. Así, se busca un campo vectorial v de velocidades dependientes del tiempo en el cual, al integrar la ecuación (1.4) de la forma

$$\varphi = \phi_1 = \phi_0 + \int_0^1 v_t(\phi_t)dt$$

se genere el difeomorfismo particular que empareja las dos imágenes dadas.

Sea $\phi_{t,s} : \Omega \to \Omega$ la composición $\phi_{t,s} = \phi_s \circ (\phi_t)^{-1}$. La interpretación de $\phi_{t,s}(x)$ es que es la posición a tiempo s de una partícula que está en la posición x y a tiempo t. Por lo tanto $\phi_1^v(x) = \phi_{0,1}^v(x)$ es la función que denota la posición en el instante $s = 1$ de una partícula que está en la posición x en el instante 0.

La existencia de las transformaciones generadas mediante la ecuación (1.4) depende de las restricciones que imponen la suavidad permitidas en V. Una forma de asegurar que existan soluciones en el espacio de los difemorfismos para esta ecuación [4], es construir el espacio V como la complección del espacio de los campos vectoriales suaves y compactamente definidos por el producto interno, definido a través del operador diferencial L comúnmente elegido para ser un operador diferencial de la forma $L = (-\alpha\Delta + \gamma)^\beta$

$$\langle f, g\rangle_V \doteq \langle Lf, Lg\rangle_{L^2} = \langle L^\dagger Lf, g\rangle_{L^2}\,, \tag{1.5}$$

donde $\langle\ ,\ \rangle$ es el producto L^2 para los campos vectoriales de cuadrado integrable en Ω y $L^\dagger$ es el adjunto de L. Con V definido de esta manera, el flujo de $v \in L^1([0,1], V)$ genera un subgrupo de difeomorfismos $\mathcal{G} \doteq \{\varphi|\ \varphi = \phi_1^v,\ v \in L^1([0,1], V)\}$ que son los puntos finales de los flujos asocialdos a los elementos $v \in L^1([0,1], V)$. Así, podemos obtener el espacio V mediante el operador compacto y autoadjunto $K : L^2(\Omega, \mathbb{R}^d) \to V$, de forma que está únicamente definido [4] por

$$\langle a, b\rangle_{L^2} = \langle Ka, b\rangle_V\,. \tag{1.6}$$

Así, finalmente, la estimación de la transformación óptima tiene la forma [4]:

$$\hat{v} = \underset{v:\in L^2([0,1],V)}{argmin}\ E(v) \tag{1.7}$$

donde

$$E(v) = \int_0^1 ||v_t||_V^2\, dt + \frac{1}{\sigma^2}\left|\left|I_0 \circ \phi_{1,0}^v - I_1\right|\right|_{L^2}^2\,. \tag{1.8}$$

Análogamente a la definición de la energía de la ecuación 1.3, el primer término de la suma es la energía de regularización que restringe la suavidad de la transformación mediante el control de sus derivadas segundas, y el segundo término de la suma es el que mide la similitud entre las imágenes. La solución numérica de este problema propuesta originalmente en [4] se implementa mediante un método de descenso del gradiente.

1.2. Disparo geodésico y ecuación de Euler-Poincaré

El campo vectorial de velocidad que resuelve el problema dado por la ecuación (1.8) define una trayectoria geodésica en la variedad de difeomorfismos. La longitud de esta trayectoria es una distancia entre las imágenes conectadas a través del difeomorfismo en el punto final del flujo geodésico [4]. Así, dada una velocidad inicial $v_0 \in V$, en $t = 0$, el camino geodésico $\phi \in \mathrm{Diff}(\Omega)$ bajo la métrica Riemanniana (1.6) que se encuentra en el

término de regularización de (1.8) está unívocamente determinado por las ecuaciones de Euler-Poincaré [4, 6, 7]

$$\frac{\partial v_t}{\partial t} = -K\left[(Dv_t)^T \cdot m_t + Dm_t \cdot v_t + m_t \cdot \text{div } v_t\right], \tag{1.9}$$

donde $L : V \to V^*$ es un operador diferencial simétrico y definido positivo de la forma $L = (-\alpha\Delta + \gamma)^\beta$, como se ha comentado anteriormente, que mapea al vector tangente $v_t \in V$ a su espacio dual como el vector momento $m_t \in V^*$. Así, $m_t = Lv_t \iff v_t = Km_t$, donde K es el operador inverso de L. Además, el operador D representa el operador de matriz Jacobiana, y $\cdot$ hace referencia al producto de dos matrices elemento a elemento, o producto de Hadamard.

La ecuación de Euler-Poincaré 1.9 se puede reescribir [6] como:

$$\frac{\partial v_t}{\partial t} = -\text{ad}^\dagger_{v_t} v_t = -K\text{ad}^*_{v_t} m_t, \tag{1.10}$$

donde el operador ad^* es el dual del corchete de Lie en campos vectoriales:

$$\text{ad}_{vt} w_t = -[v_t, w_t] = Dv_t \cdot w_t - Dw_t \cdot v_t. \tag{1.11}$$

1.3. LDDMM mediante disparo geodésico

El problema variacional original del método LDDMM se puede plantear en el espacio de velocidades iniciales sujetas a la ecuación EPDiff. Así, dada una velocidad inicial $v_0 \in V$, el nuevo problema variacional se plantea mediante la minimización de la energía

$$E(v_0) = ||v_0||^2_V + \frac{1}{\sigma^2}\,||I_0 \circ \phi_1 - I_1||^2_{L^2}\,. \tag{1.12}$$

Este problema se puede resolver mediante optimización basada en el descenso del gradiente. Existe una dependencia muy compleja entre ϕ_1 y v_0, que se refleja en la complejidad del cálculo del gradiente en v_0, $\nabla_{v_0} E(v_0)$.

No obstante, se ha propuesto recientemente calcular el gradiente de la energía en el instante de tiempo $t = 1$, y realizar su transporte paralelo hacia el instante de tiempo $t = 0$ para obtener $\nabla_{v_0} E(v_0)$ y actualizar el valor de v_0 mediante el descenso del gradiente [4, 6].

Así,

$$\nabla_{v_1} E(v_0) = K\left(\frac{1}{\sigma^2}(I_0 \circ \phi_1 - I_1) \cdot \nabla(I_0 \circ \phi_1)\right) \tag{1.13}$$

donde $\phi_1 \equiv \phi^v_{1,s}$, es decir, es el difeomorfismo en el instante temporal $t = 1$, y en la iteración (o instante) s. El transporte paralelo se obtiene mediante la integración de las ecuaciones de Jacobi adjuntas reducidas [6], que permiten obtener $\nabla_{v_0} E(v_0)$ mediante el transporte paralelo de $\nabla_{v_1} E(v_0)$. La ventaja de esta aproximación al problema es la simplicidad en la derivación de la expresión analítica del gradiente en v_1.

1.4. Ecuaciones de Jacobi

El cálculo del término de gradiente de nuestro modelo requiere los campos adjuntos de Jacobi [6], que sirven para integrar el término del gradiente desde $t = 1$ hacia atrás al punto inicial $t = 0$. Para ello, se utilizará una versión de las ecuaciones de Jacobi reducidas [6, 8], definidas como:

$$\frac{d}{dt}\begin{pmatrix} U_t \\ w_t \end{pmatrix} = \begin{pmatrix} -ad^{\dagger}_{v_t} & 0 \\ -I & -sym^{\dagger}_{v_t} \end{pmatrix}\begin{pmatrix} U_t \\ w_t \end{pmatrix} \tag{1.14}$$

donde $sym^{\dagger}_{v_t} w_t = -ad_{v_t} w_t + ad^{\dagger}_{w_t} v_t$ y las condiciones iniciales son:

$$w_{t=1} = 0$$

$$U_{t=1} = \nabla_{v_1} E(v_0)$$

Combinando las expresiones desarrolladas en los apartados anteriores, el algoritmo LDDMM que se utiliza como baseline en este TFM se divide en los siguientes pasos:

Algorithm 1 Algoritmo LDDMM con las ecuaciones de Jacobi reducidas

Entrada: $I_0,\ I_1,\ \sigma,\ \epsilon,\ v_0 \in V$

for $n \leftarrow 0$ **to** *convergencia* **do**

- Obtener v_t^{n+1} realizando el disparo geodésico mediante la ecuación de Euler-Poincaré (1.9) con condición inicial v_0^n.
- Calcular la transformación de difeomorfismos ϕ_1^{n+1} mediante la ecuación de estado (1.4) con condición inicial ϕ_0.
- Obtener el gradiente $\nabla_{v_1} E(v_t)^{n+1}$ en $t = 1$ mediante la expresión (1.13).
- Calcular el transporte paralelo hacia atrás de $\nabla_{v_1} E(v_t)^{n+1}$ integrando las ecuaciones de Jacobi adjuntas reducidas (1.14) con condiciones iniciales $w_{t=1} = 0$ y $U_{t=1} = \nabla_{v_1} E$. El gradiente en v_0, $\nabla_{v_0} E(v_t)^{n+1}$, se obtiene como $U_{t=0}$.
- Actualizar el valor de la velocidad inicial $v_0^{n+1} \leftarrow v_0^n - \epsilon \nabla_{v_0} E$

end

return $I_0 \circ \phi_1^n$

Capítulo 2

Modelos de Orden Reducido

Actualmente, los sistemas de gran dimensionalidad y sistemas complejos conllevan un desafío en el procesado de sus datos. Estos sistemas pueden involucrar grandes cantidades de datos y medidas como pueden ser imágenes médicas en 3D, vídeos generados a partir de sistemas físicos, medidas de la velocidad de fluidos de simulaciones o experimentos, etc. En muchos de estos sistemas, se observa que estos datos siguen ciertos patrones dominantes, los cuales son caracterizados por atractores de baja dimensión. Por ejemplo, aunque las simulaciones de fluidos de alta fidelidad con la realidad típicamente requieren de millones de grados de libertad, normalmente presentan estructuras coherentes en los flujos [9].

Los Modelos de Orden Reducido o *Reduced Order Models* (ROM) son un conjunto de técnicas de reducción de la complejidad computacional de modelos matemáticos en simulaciones numéricas. El objeto de estudio de este trabajo será el ROM de la Descomposición Ortogonal Propia o *Proper Orthogonal Decomposition* (POD), la cual se basa en la aplicación del algoritmo de Descomposición en Valores Singulares o SVD (*Singular Value Decomposition*) en la resolución de ecuaciones en derivadas parciales o *Partial Differential Equations* (PDEs). La Descomposición en Valores Singulares (SVD) [9, 10] proporciona una forma sistemática de determinar una aproximación de un sistema altamente dimensional a otro de bajas dimensiones en términos de sus patrones dominantes. Esta técnica es *data-driven*, o dirigida por datos, ya que los patrones encontrados son obtenidos en base a los datos, sin tener que añadir una experiencia o conocimiento previo sobre el ámbito de estudio. La SVD, además, es numéricamente estable y proporciona una representación jerárquica de los datos en términos de un nuevo sistema de coordenadas, definido por las correlaciones dominantes entre los datos.

2.1. Descomposición en Valores Singulares

Sea $A \in \mathbb{R}^{m \times n} \in$ una matriz $m \times n$ de rango k [10]. Sean entonces los números $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_k \geq \sigma_{k+1} \geq ... \geq \sigma_n > 0$ los valores singulares de A, la matriz unitaria $U = [u_1, ..., u_m]$ de dimensiones $m \times m$, y la matriz unitaria $V = [v_1, ..., v_n]$ de dimensiones $n \times n$, tales que $A = U\Sigma V^T$ y $\Sigma = U^T AV$, donde Σ es la matriz $m \times n$

$$\Sigma = \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} \sigma_1 & 0 & ... & 0 \\ 0 & \sigma_1 & ... & 0 \\ & ... & ... & \\ 0 & 0 & ... & \sigma_n \\ 0 & 0 & ... & 0 \\ 0 & ... & ... & 0 \\ 0 & 0 & ... & 0 \end{pmatrix} \in \mathbb{R}^{\mathrm{m}\times\mathrm{n}} \tag{2.1}$$

donde D es la matriz diagonal $k \times k$ con el i-ésimo elemento $d_{ii} = \sigma_i > 0$ para $1 \leq i \leq k$. Además, para $1 \leq i \leq k$, $u_i = \sigma_i^{-1} A v_i$ y $v_i = \sigma_i^{-1} A^T u_i$ son los autovectores de las matrices semi-definidas positivas AA^T y de $A^T A$, respectivamente, y ambos están asociados con los autovalores $\sigma_i > 0$. Los vectores u_i para $k+1 \leq i \leq m$ y v_i para $k+1 \leq i \leq n$ son los autovectores de AA^T y $A^T A$ respectivamente, todos asociados con el autovalor cero. Si A es real, entonces U y V pueden ser tomadas como matrices reales y, por lo tanto, ortogonales [10].

Viendo esta descomposición, la matriz A puede escribirse como combinación lineal de matrices tal que:

$$A = \sum_{i=1}^{n} \sigma_i u_i v_i^T \tag{2.2}$$

2.2. Descomposición Ortogonal Propia

La POD es la aplicación del algoritmo de SVD a la resolución de PDEs, y es una de las técnicas de reducción de dimensionalidad más importantes para estudiar sistemas complejos espacio-temporales, como pueden ser sistemas biológicos o físicos. La evolución de cierta cualidad o cantidad de estos sistemas, como puede ser la posición o la velocidad de una partícula, viene descrita por PDEs no lineales. El éxito de la POD está relacionado con la observación de que, aparentemente, en la mayoría de los sistemas complejos, los comportamientos significativos se encuentran en patrones de dimensión considerablemente más baja que el sistema completo. Así, la técnica de la POD se aprovecha de esta observación, para crear un sistema dinámico de dimensiones bajas a partir de estos patrones. Éste tiene que ser capaz de modelar el sistema completo con la mayor precisión posible. Su éxito radica en su capacidad de proporcionar decomposiciones de los datos espacio-temporales interpretables físicamente [9, 11, 12].

2.2.1. Formulación del problema

Sea el problema no lineal y altamente dimensional:

$$\begin{aligned} \frac{dw(t)}{dt} &= f\left(w(t), t\right) \\ y(t) &= g\left(w(t), t\right) \\ w_0 &= w(0) \end{aligned} \tag{2.3}$$

donde $w \in \mathbb{R}^{\mathrm{N}}$ es el vector de variables de estado e $y \in \mathbb{R}^{\mathrm{q}}$ es el vector de variables de salida. Además, se considerará la condición inicial fija $w_0 \in \mathbb{R}^{\mathrm{N}}$. El sistema de ecuaciones altamente dimensionado viene definido por $f(\cdot,\cdot) \in \mathbb{R}^{\mathrm{n}}$ junto a $dw(t)/dt$ [13].

Considérese la trayectoria de estados asociada en el intervalo de tiempo $[0, \mathcal{T}]$:

$$T_w = \{w(t)\}_{0 \leq t \leq \mathcal{T}}$$

La Descomposición Ortogonal Propia busca una proyección ortogonal $\Pi_{V,V}$ de rango fijo k que minimiza la integral del error de la proyección:

$$\int_0^{\mathcal{T}} ||w(t) - \Pi_{V,V} w(t)||_2^2 \, dt = \int_0^{\mathcal{T}} ||\epsilon_{V\perp}(t)||_2^2 \, dt = ||\epsilon_{V\perp}|| = \mathcal{J}(\Pi_{V,V}) \tag{2.4}$$

2.2.2. Solución al problema de minimización

Sea $\hat{K} \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ una matriz real, simétrica y semi-definida positiva como:

$$\hat{K} = \int_0^{\mathcal{T}} w(t) w(t)^T dt \tag{2.5}$$

Sean $\hat{\sigma}_1 \leq \hat{\sigma}_2 \leq ... \leq \hat{\sigma}_N \leq 0$ los autovalores ordenados de $\hat{K}$. Sean $\hat{\phi}_i \in \mathbb{R}^{\mathrm{N}}$, i $= 1, ..., \mathrm{N}$ los autovectores asociados, los cuales son también conocidos como modos POD. Así se tiene que:

$$\hat{K}\hat{\phi}_i = \hat{\sigma}_i \hat{\phi}_i, \;\; i = 1, ..., N \tag{2.6}$$

El subespacio $\hat{\mathcal{V}} = rango(\hat{V})$ de dimensión k que minimiza $\mathcal{J}(\Pi_{V,V})$ es el subespacio invariante de $\hat{K}$ asociado con los autovalores $\hat{\sigma}_1 \leq \hat{\sigma}_2 \leq ... \leq \hat{\sigma}_k$.

2.2.3. Discretización de la POD mediante el método de las instantáneas (*snapshots*)

Resolver este problema de autovalores $\hat{K}\hat{\phi}_i = \hat{\sigma}_i \hat{\phi}_i$ es en general computacionalmente muy costoso, debido a la gran dimensión de $\hat{K}$, que además, suele ser una matriz densa. Sin embargo, los datos de las variables de estado están disponibles bajo la forma de vectores de instantáneas o *snapshots*:

$$\{w(t_i)\}_{i=1}^{N_{snap}} \tag{2.7}$$

En este caso, la integral $\displaystyle\int_0^{\mathcal{T}} w(t) w(t)^T dt$ se puede reescribir mediante la regla de la cuadratura como:

$$K = \sum_{i=1}^{N_{snap}} \alpha_i w(t_i) w(t_i)^T \tag{2.8}$$

donde $\alpha_i, \;\; i = 1, ..., N_{snap}$ son los pesos de la cuadratura.

Sea $S \in \mathbb{R}^{\mathrm{N}\times\mathrm{N_{snap}}}$ la matriz de *snapshots* definida como:

$$S = [\sqrt{\alpha_1} w(t_1) \ ... \ \sqrt{\alpha_{N\,snap}} w(t_{N_{snap}})] \tag{2.9}$$

Resultando que

$$K = SS^T \tag{2.10}$$

donde K es la matriz completa de grandes dimensiones $(N \times N)$.

Además, hay que darse cuenta de que los valores no-nulos de la matriz $K = SS^T \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ son los mismos que los de la matriz $R = S^T S \in \mathbb{R}^{\mathrm{N_{snap}}\times\mathrm{N_{snap}}}$. Como normalmente $N_{snap} << N$, es más económico resolver así el problema de autovalores simétrico:

$$R\psi_i = \sigma_i \psi_i, \ 1, ..., N_{snap} \tag{2.11}$$

Si el rango de R es $ran(R) = r$, entonces, los primeros r modos de la POD ϕ_i vienen dados por:

$$\phi_i = \frac{1}{\sqrt{\sigma_i}} S\psi_i, \ i = 1, ..., r \tag{2.12}$$

Sean $\Phi = [\phi_1 \ ... \ \phi_r]$ y $\Psi = [\psi_1 \ ... \ \psi_r]$ con $\Psi^T\Psi = I_r \Rightarrow \Phi = S\Psi\Lambda^{-1/2}$ donde

$$\Lambda = \begin{bmatrix} \sigma_1 & & (0) \\ & \ddots & \\ (0) & & \sigma_r \end{bmatrix} \tag{2.13}$$

que es la matriz correspondiente a los autovalores no negativos.

2.3. Conexión entre SVD y POD

Dada una matriz $A \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}$ con $N \geq M$. Por el teorema de Eckart-Young [9, 13] la matriz $X \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}$ con rango $ran(X) = k < r = rango(A) \leq M$ que minimiza $||A - X||_2$ viene dada por:

$$\min_{X, \ rango(X)=k} ||A - X||_2 = \sigma_{k+1}(A), \text{ si } \sigma_k(A) > \sigma_{k+1}(A) \tag{2.14}$$

Por eso $X = \sum_{i=1}^{k} \sigma_i u_i v_i^T$ minimiza $||A - X||_2$, donde $A = U\Sigma V^T$.

La discretización de la POD mediante el método de las instantáneas requiere el calcular el espectro de la matriz $K = SS^T$:

$$\Phi^T K \Phi = \Phi^T S S^T \Phi = \Lambda$$

donde Λ viene dada por (2.13).

Conectándolo con la SVD de S:

$$S = \begin{bmatrix} U_r & U_{N-r} \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} V^T \tag{2.15}$$

donde $U_r \in \mathbb{R}^{\mathrm{N}\times\mathrm{r}}$ es la matriz que se identifica con $X \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}},\ \mathrm{N} \geq \mathrm{M} \geq \mathrm{r}$.

2.3.1. Elección del tamaño de la base de orden reducido

Se comienza aplicando a la matriz S la norma de Frobenius

$$||S||_F = \sqrt{\sum_{i=1}^{r} \sigma_i^2(S)} \tag{2.16}$$

Se considera el error medido con la norma de Frobenius inducida por la truncación de la base de la POD

$$\left|\left|(I_N - VV^T)S\right|\right|_F = \sqrt{\sum_{i=k+1}^{r} \sigma_i^2(S)} \tag{2.17}$$

Así, el cuadrado del error relativo da una indicación de la magnitud de la información "perdida":

$$\varepsilon_{POD}(k) = \frac{\sum_{i=1}^{k}\sigma_i^2(S)}{\sum_{i=1}^{r}\sigma_i^2(S)} \Rightarrow 1 - \varepsilon_{POD}(k) = \frac{\sum_{i=k+1}^{r}\sigma_i^2(S)}{\sum_{i=1}^{r}\sigma_i^2(S)} \tag{2.18}$$

donde $\varepsilon_{POD}(k)$ representa la energía relativa de las instantáneas capturadas por los primeros k modos de la base de la POD. Normalmente se elige k de forma que es el mínimo entero para el que se da

$$1 - \varepsilon_{POD}(k) \leq \epsilon \tag{2.19}$$

para una tolerancia dada $0 < \epsilon < 1$.

2.3.2. Ejemplo: Compresión de imágenes

A continuación, se va a visualizar la idea de la aproximación por la descomposición en valores singulares mediante un ejemplo recurrentemente utilizado en la literatura: la compresión de imágenes. Para ello, se va utilizar una imagen en escala de grises, interpretada como una matriz de números reales tal que $A \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}$, donde N y M son el número de píxeles en dirección vertical y horizontal respectivamente. Se descompone esta matriz mediante la técnica SVD para, posteriormente, reconstruirla a partir de distintas aproximaciones de la base de orden reducido. La imagen que se va a tratar de comprimir se muestra en la figura (2.1).

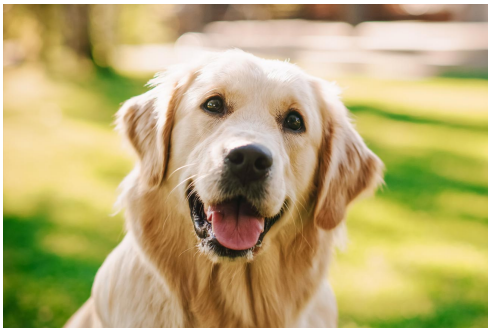


Figura 2.1: Imagen original que se va a transformar en escala de grises y se va a comprimir.

En la figura (2.2) se muestran dos gráficos: en el primero de ellos se representa el espectro de los valores singualares de la imagen original y, en el segundo, se muestra la energía acumulada por los autovalores de la imagen, la cual se corresponderá con la información que contiene la imagen comprimida. Se puede evidenciar el orden decreciente de los valores singulares, lo cual indica que la mayor cantidad de información está contenida en las direcciones asociadas a los valores singulares más grandes.

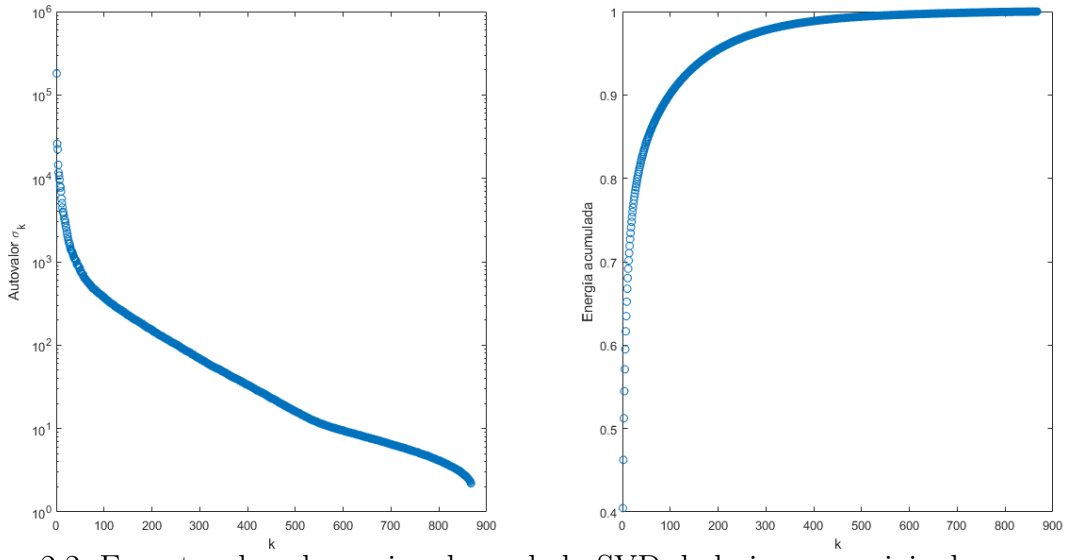


Figura 2.2: Espectro de valores singulares de la SVD de la imagen original y su espectro de energía acumulada.

Por último, se hace una representación de la imagen comprimida:

Figura 2.3: Diferentes compresiones de la imagen (2.1) en escala de grises.

En la figura (2.3) se comparan la imagen original y tres compresiones de esta. La primera con un rango de $r = 10$ valores singulares, con la cual se conserva un 66.77 % de la información de la imagen; la segunda con un rango de $r = 50$, con el cual se conserva un 84.32 % de la información y, finalmente, con un rango de $r = 100$ valores singulares, se alcanza un 90.13 % de la información de la imagen original.

2.3.3. Proyección de Galerkin

Es posible aproximar el estado de u de una PDE usando la proyección de Galerkin [9, 13]. Partimos del problema inicial dado por 2.3, y buscamos un sistema de orden reducido de la forma:

$$\begin{aligned} \frac{dq(t)}{dt} &= f_r\left(q(t), t\right) \\ y(t) &= g_r\left(q(t), t\right) \\ q_0 &= q(0) \end{aligned} \tag{2.20}$$

donde $q \in \mathbb{R}^{\mathrm{k}}$ es el vector de variables de estado de orden reducido e $y \in \mathbb{R}^{\mathrm{q}}$ es el vector de variables de salida. La descripciónn del sistema de ecuaciones de dimensión reducida viene completada por $f(\cdot, \cdot) \in \mathbb{R}^{\mathrm{k}}$, donde $k << N$.

Para ello, se va a utilizar la matriz reducida de *snapshots* U_r, de forma que:

$$U_r \frac{dq(t)}{dt} = f(U_r q(t), t) + r(t) \tag{2.21}$$

donde $r(t) \in \mathbb{R}^N$ es un residual que viene dado por el hecho de que $U_r q(t)$ no es la solución exacta del problema original. Si imponemos que este residual sea ortogonal al subespacio definido por U_r, tal que

$$U_r r(t) = 0$$

finalmente tendremos que:

$$U_r^T U_r \frac{dq(t)}{dt} = U_r^T f(U_r q(t), t) \tag{2.22}$$

Así, el problema de dimensiones reducidas tendrá la forma:

$$\begin{aligned} \frac{dq(t)}{dt} &= U_r^T f\left(U_r q(t), t\right) \\ y(t) &= g\left(U_r q(t), t\right) \\ q_0 &= U_r^T w(0) \end{aligned} \tag{2.23}$$

que es equivalente al problema inicial de grandes dimensiones reducido mediante la proyección de Galerkin.

Capítulo 3

POD - LDDMM

En esta sección se presenta el algoritmo desarrollado en este trabajo basado en el método de reducción de la POD para llevar a cabo el método LDDMM de una forma que sea menos costoso computacionalmente. El método se basa en la formulación de LDDMM bajo la ecuación de Euler-Poincaré, donde el gradiente de la energía en v_0 se calcula mediante transporte paralelo. Este método fue inicialmente propuesto en [12].

Dado un conjunto de velocidades de dimensionalidad completa $\{v_t\} \in \mathcal{V}^q$, donde $q = N \times M \times 2$ para una imagen en 2D discretizada de dimensiones $N \times M$, se quiere encontrar un subespacio aproximado $U^r = \text{span}\{u_1, ..., u_r\} \subset \mathcal{V}^q$ con $(r << q)$, donde $u_i \ \forall i = 1, ..., r$ es la base que mejor caracteriza los datos utilizados. Una proyección desde tal espacio de bajas dimensiones al espacio original viene dado por $v_t = U_r \alpha_t$, donde $U^{q \times r} = [u_1, ..., u_r]$ y α_t es un vector r-dimensional que representa los coeficientes temporales de esta transformación. Así, la proyección inversa viene dada por $\alpha_t = U_r^T v_t$.

Para encontrar la base U se realiza la descomposición en valores singulares del conjunto de velocidades $\{v_t\}$. Este quedará dividido en tres matrices, como se ha visto anteriormente: en la primera de ellas, U, cada columna corresponde a los modos espaciales del sistema; en la segunda matriz, Σ, cada elemento de la diagonal corresponde a la energía de cada uno de esos modos temporales y, finalmente, en la tercera matriz, V, contiene las componentes espaciales del sistema. Se elegirán las primeras r columnas de U, aprovechando el rápido decaimiento del espectro de autovalores, y que corresponderán con los modos más energéticos del sistema, los cuales gobernarán la dinámica de este.

Una vez se ha obtenido la dimensionalidad de la base del subespacio, se halla una estimación de la ecuación de Euler-Poincaré propuesta en [12] mediante la proyección de Galerkin. Así, partiendo de la discretización de la ecuación de Euler-Poincaré original en forma matricial (1.9), se obtiene:

$$\begin{aligned}
\frac{\partial v}{\partial t} &= -K \left[\text{diag}(Lv) D^T v + \text{diag}(v) D(Lv) + \text{diag}(Lv) div(v) \right] \\
&= -K \sum_{i=1}^{q} \left[\text{diag}(l_i) v_i D^T v + v_i D(Lv) + \text{diag}(l_i) v_i div(v) \right] \\
&= -K \sum_{i=1}^{q} \left[\text{diag}(l_i) D^T + DL + \text{diag}(l_i) div \right] v_i v
\end{aligned} \tag{3.1}$$

donde, para simplificar la notación, se ha eliminado el índice t que denota la dependencia temporal del campo de velocidades v. Aquí, v es un vector q-dimensional, y $\text{diag}(\cdot)$ convierte un vector en una matriz diagonal. Las matrices L, K, D y div representan la

discretización en forma matricial de los operadores L, su inversa K, el jacobiano y la divergencia, respectivamente. Por último, l_i es la i-ésima columna de la matriz L y v_i es el i-ésimo elemento del vector v.

De esta forma, se define el operador compuesto por

$$A_i^{q\times q} \equiv -K\left[\text{diag}(l_i)D^T + DL + \text{diag}(l_i)div\right],$$

pudiendo reescribirse la ecuación de Euler-Poincaré como:

$$\frac{\partial v}{\partial t} = \sum_{i=1}^{q} A_i v_i v. \tag{3.2}$$

A continuación, derivando la ecuación reducida mediante la proyección de Galerkin y sustituyendo $v = U^{q\times r}\alpha$ se obtiene:

$$\begin{aligned}
&\frac{\partial U\alpha}{\partial t} = \sum_{i=1}^{q} A_i (U\alpha)_i U\alpha \;\Rightarrow \\
&\quad\Rightarrow \frac{\partial \alpha}{\partial t} = U^T \sum_{i=1}^{q} A_i \left(\sum_{j=1}^{r} U_{ij}\alpha_j\right) = \sum_{i=1}^{q}\sum_{j=1}^{r} U^T A_i U U_{ij}\alpha_j \alpha \;\Rightarrow \\
&\quad\Rightarrow \frac{\partial \alpha}{\partial t} = \sum_{j=1}^{r} \tilde{A}_j \alpha_j \alpha,
\end{aligned} \tag{3.3}$$

donde se define el operador $\tilde{A_j}^{r\times r} = \sum_{i=1}^{q} U^T A_i U U_{ij}$ como el operador A_j reducido. Este operador se calculará una única vez para posteriormente integrar la ecuación reducida de Euler-Poincaré en el espacio de coeficientes α.

A continuación, debido a que la solución de la ecuación reducida de Euler-Poincaré viene dada en el espacio de coeficientes α, se presenta el modelo de orden reducido del disparo geodésico para el registro difeomórfico de imágenes, de forma que se realizarán los mismos pasos que en el algoritmo LDDMM pero proyectados en el subespacio reducido. Se calcula el descenso del gradiente en una velocidad inicial proyectada, representada por los coeficientes α_0 en el subespacio de dimensión reducida.

La función de energía del problema LDDMM se redefine como:

$$E(\alpha_0) = E_{reg}(\alpha_0) + \frac{1}{\sigma^2}\,||S\circ\psi_1 - T|\,|_2^2. \tag{3.4}$$

La expresión del regularizador se define en [12] como $E_{reg}(\alpha_0) = \langle L\alpha_0, \alpha_0\rangle$. No obstante, el operador L no es directamente aplicable a α por lo que suponemos que los autores están realizando un abuso de notación. Lo más lógico sería que el regularizador correspondiera con la expresión

$$\langle LU\alpha_0, U\alpha_0\rangle_{L^2}, \tag{3.5}$$

que es el que se ha utilizado en la implementación numérica del algoritmo en este trabajo.

A continuación, se calcula el término de gradiente usando un esquema *backward-forward*, como en el caso del algoritmo LDDMM de baseline. Para ello:

1. Se calcula el gradiente $\nabla_{\alpha_1} E(v_0)$ de la energía en el punto $t = 1$ integrando tanto el difeomorfismo ψ_t como el campo de velocidades proyectadas α_t adelante en el tiempo, tal que:

$$\nabla_{\alpha_1} E(v_0) = U^{-1} K \left(\frac{1}{\sigma^2} (I_0 \circ \phi_1 - I_1) \cdot \nabla (I_0 \circ \phi_1) \right) \tag{3.6}$$

   Se multiplica la inversa de la matriz U por el término del gradiente en el espacio V original, debido a que es la forma de transportar de un espacio a otro el término del gradiente de la energía. En el artículo original [12] se indica que el gradiente está calculado en el espacio de α pero la expresión dada se corresponde con un gradiente calculado en V. Se cree que se trata de una errata al faltar el término de proyección en el espacio de α.

2. Llevar el gradiente $\nabla_{\alpha_1} E$ del punto $t = 1$ atrás en el tiempo al punto $t = 0$ integrando las ecuaciones reducidas de Jacobi tales que:

$$\frac{d}{dt} \begin{pmatrix} \hat{e}_t \\ \hat{h}_t \end{pmatrix} = \begin{pmatrix} -ad^{\dagger}_{\alpha_t} & 0 \\ -I & -sym^{\dagger}_{\alpha_t} \end{pmatrix} \begin{pmatrix} \hat{e}_t \\ \hat{h}_t \end{pmatrix} \tag{3.7}$$

   donde $sym^{\dagger}_{\alpha_t} \hat{h}_t = -ad_{\alpha_t} \hat{h}_t + ad^{\dagger}_{\hat{h}_t} \alpha_t$ y las condiciones iniciales son:

$$\hat{h}_{t=1} = 0$$

$$\hat{e}_{t=1} = \nabla_{\alpha_1} E(v_0)$$

Así, finalmente el algoritmo consta de los siguientes pasos:

Algorithm 2 Algoritmo LDDMM-POD con las ecuaciones de Jacobi reducidas

Entrada: $I_0,\ I_1,\ \sigma,\ \epsilon,\ \alpha_0 = 0$

for $n \leftarrow 0$ **to** *convergencia* **do**

- Obtener α_t^{n+1} realizando el disparo geodésico mediante la ecuación de Euler-Poincaré (3.2) con condición inicial α_0^n.
- Realizar la proyeción $v_t = U\alpha_t$. Calcular la transformación de difeomorfismos ϕ_1^{n+1} mediante la ecuación de estado (1.4) con condición inicial ϕ_0.
- Obtener el gradiente $\nabla_{v_1} E(\alpha_t)^{n+1}$ en $t = 1$ mediante la expresión (3.6).
- Calcular el transporte paralelo hacia atrás de $\nabla_{\alpha_1} E(\alpha_t)^{n+1}$ integrando las ecuaciones de Jacobi adjuntas reducidas (3.7) con condiciones iniciales $\hat{h}_{t=1} = 0$ y $\hat{e}_{t=1} = \nabla_{\alpha_1} E$ y obteniendo el gradiente en $\nabla_{\alpha_0} E(\alpha_t)^{n+1}$ en $t = 0$.
- Actualizar el valor de la velocidad inicial $\alpha_0^{n+1} \leftarrow \alpha_0^n - \epsilon \nabla_{\alpha_0} E$

end

return $I_0 \circ \phi_1^n$

Para llevar a cabo la implementación de las ecuaciones del algoritmo 2 (al igual que en el algoritmo 1) se ha utilizado el método de integración propuesto en los ejemplos de [9], haciendo uso de la función *feval* de MATLAB que combina una función que implementa un método de Runge-Kutta de orden 3 con una función *rhs* que representa el *right hand side* de la ecuación diferencial. De esta forma, se evalúan los parámetros iniciales de la función, y se realiza la integración mediante Runge-Kutta. En cada paso de Runge-Kutta, mediante la función *rhs* se devuelve en un vector los resultados de la evaluación, y estos, a su vez, se evalúan de nuevo en el siguiente paso. Finalmente se obtiene una matriz con $n + 1$ columnas, donde n es el número de pasos de Runge-Kutta, y en cada columna almacena el vector resultado de la evaluación del Runge-Kutta en ese paso.

En el artículo [12] no había indicación de como llevar a cabo la implementación de las ecuaciones de Jacobi adjuntas en el espacio reducido. Después de un análisis del problema se llegó a la conclusión de que, mediante el método de integración utilizado explicado en el párrafo anterior, la forma correcta de implementar estas ecuaciones es la siguiente: como parámetros iniciales se utilizan los nombrados en la ecuación (3.7), que están en el espacio reducido. Tras esto, se proyectan al espacio original mediante la base U, y se lleva a cabo la integración con Runge-Kutta en el primer paso. Posteriormente, se vuelve a proyectar en el espacio reducido, y mediante la función *rhs*, se devuelven los resultados en el espacio reducido (resultados que volerán a ser utilizados como parámetros iniciales en el siguiente paso de la iteración).

Capítulo 4

Resultados

Para verificar la corrección de los algoritmos implementados en este trabajo, se procede a comparar sus resultados en un experimento tradicionalmente utilizado en la literatura de LDDMM [14]. Para ello se obtendrán diferentes métricas con las que se valorará la ejecución del algoritmo, como son el tiempo, el número de iteraciones, el error cuadrático medio o la dimensión de la base utilizada para la reducción de la dimensionalidad del sistema.

Se referirá con LDDMM al algoritmo de *baseline* (1), y con POD-LDDMM (2) al algoritmo que utiliza la reducción POD y se postula como una mejora eficiente del primer algoritmo.

4.1. Imágenes

Las imágenes utilizadas son las siguientes:

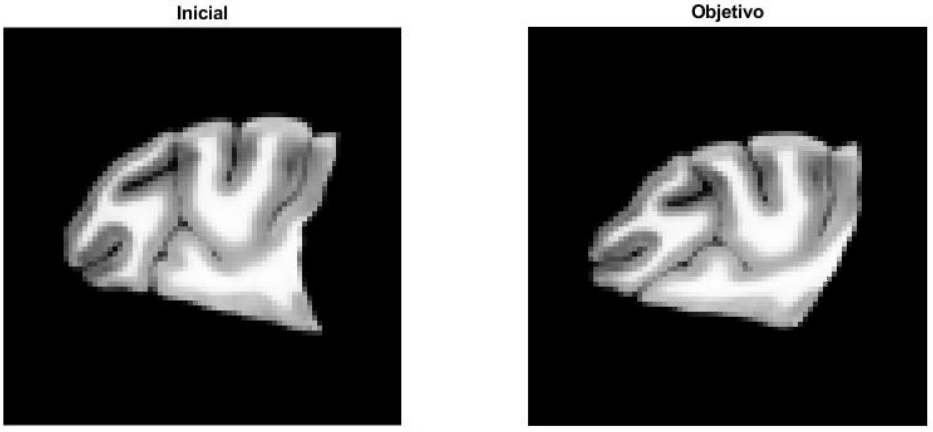


Figura 4.1: Imagen inicial e imagen objetivo con las cuales se van a realizar los experimentos.

Estas imágenes representan una adquisición de dos cortes histológicos de un cerebro de macaco y han sido cedidas amablemente por Mirza Faisal Beg. En la página web `https://www.cis.jhu.edu/software/lddmm-volume/validation.php` pueden encontrarse los experimentos de referencia realizados con el algoritmo LDDMM original. Las dimensiones de estas imágenes son de 80×80 píxeles. La imagen de la izquierda se considera la imagen fuente o deformable, y la de la derecha la imagen fija u objetivo. La primera imagen será la que sea transformada de forma difeomorfa hasta asemejarse lo máximo posible a la imagen fija.

4.2. Detalles de implementación y elección de los parámetros

Se ha implementado el algoritmo baseline de LDDMM (1) en dos dimensiones. Para ello se ha partido de los códigos del trabajo [15, 16], donde se propone extender el método baseline de LDDMM a un problema de registro condicionado con el método de optimización de Gauss-Newton-Krylov. Partiendo de la implementación del algoritmo baseline de LDDMM se ha procedido a la implementación del algoritmo de POD-LDDMM.

La implementación se ha llevado a cabo con el software MATLAB (R2021b), en un sistema operativo Windows 10, con una CPU Intel(R) Core(TM) i7.5500 CPU @ 2.40GHz y con una memoria RAM DDR3 de 12Gb.

4.3. Resultados con LDDMM de baseline

Los parámetros para llevar a cabo la implementación son: $\alpha = 0.01$, $\gamma = 1.0$, $s = 1.0$ y $\sigma = 1.0$. Los resultados obtenidos son los siguientes:

Figura 4.2: Imagen inicial, imagen objetivo e imagen deformada en la implementación del algoritmo LDDMM *baseline*.

A continuación se muestra la diferencia entre la imagen objetivo y la imagen inicial (equivalente a la imagen deformada al inicio del algoritmo y la imagen objetivo), y la imagen deformada final y la imagen objetivo:

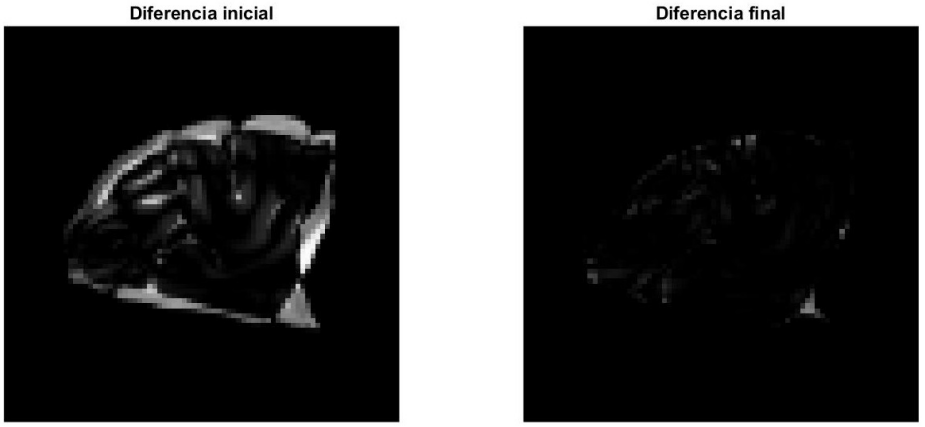
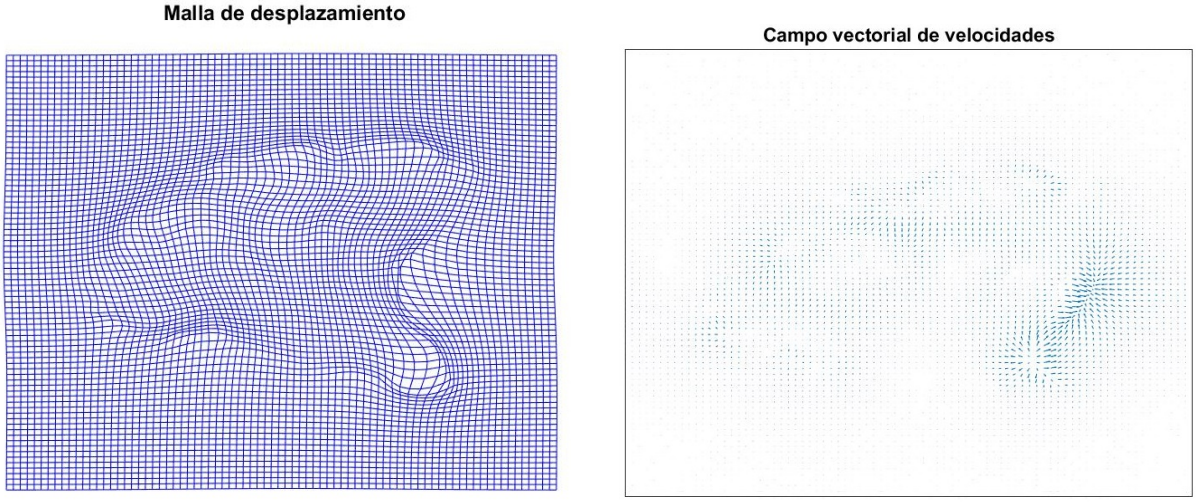


Figura 4.3: Diferencia entre las imágenes inicial y objetivo y deformada y objetivo tras la implementación del algoritmo LDDMM *baseline*.

Como se puede comprobar, se obtiene un buen registro de la imagen: en el estado inicial, la diferencia entre las imágenes es notable, mientras que en el estado final, la diferencia entre estas es mínima, y se reduce a los detalles de la imagen fija.

Tras la ejecución del programa se ha realizado una representación del movimiento de la imagen deformada, tanto del deplazamiento como el campo vectorial de velocidades de esta imagen obtenida mediante las ecuación de Euler-Poincaré 1.9:



(a) Malla de desplazamiento de la imagen deformada

(b) Campo vectorial de la velocidad de la imagen deformada

Figura 4.4: Imágenes que representan el movimiento de la imagen deformada.

Por último se representa un mapa de color del campo vectorial de velocidades. Esta representación de la velocidad será la que, posteriormente, se utilice como *snapshot* para la construcción de la base para la reducción del modelo. Esto será detallado en profundidad más adelante.

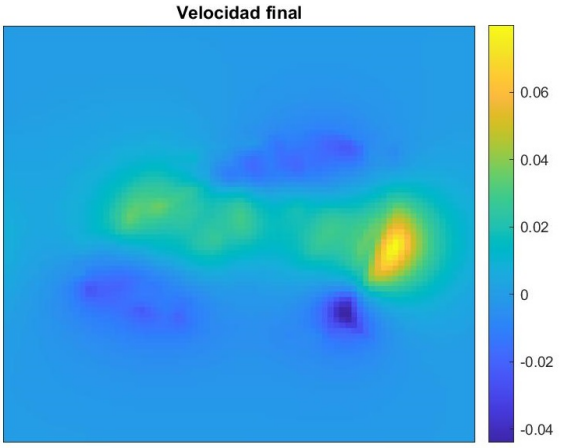


Figura 4.5: Mapa de color de la velocidad de la imagen deformada.

Finalmente, se representa el error cuadrático medio en cada iteración, calculado como la energía del imagen en la iteración k dividido entre la energía de la imagen original. El resultado para este algoritmo es el siguiente:

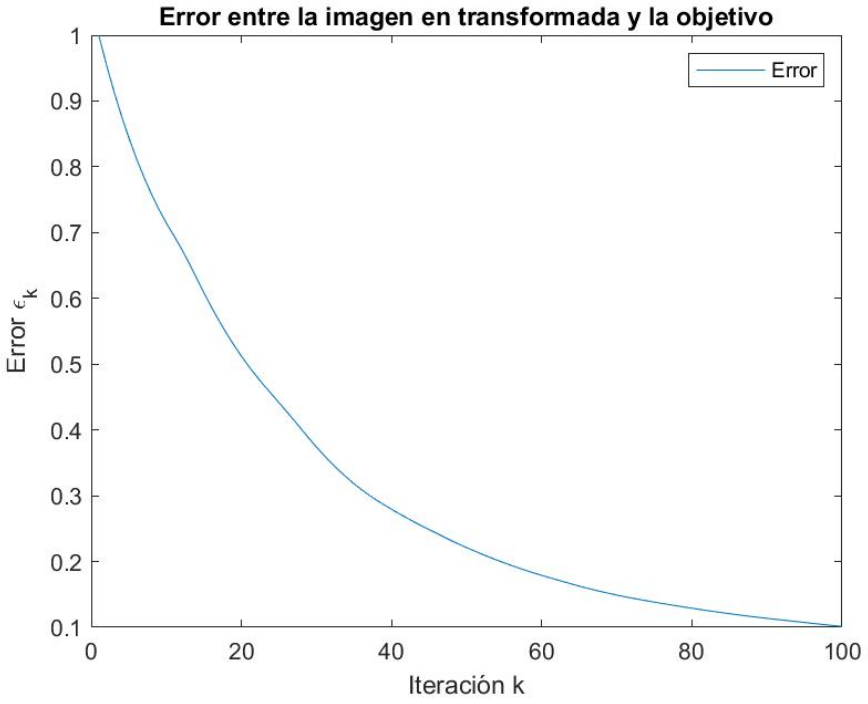


Figura 4.6: Error cuadrático medio de cada iteración en el algoritmo LDDMM.

Con este algoritmo, en 100 iteraciones, se ha alcanzado un error final del 10.13 % entre la imagen deformada y la imagen objetivo. Llevar a cabo estas 100 iteraciones y alcanzar ese error ha costado un total de 112.73 s.

4.4. Resultados con POD-LDDMM

A continuación, se van a mostrar los resultados obtenido en la implementación del algoritmo POD-LDDMM (2). Además, estos resultados se compararán con los obtenidos previamente mediante el algoritmo LDDMM.

Uno de los problemas que ha supuesto la realización de este trabajo es, en la implementación del algoritmo POD-LDDMM, la elección y construcción de la base U para la proyección al espacio reducido, ya que en el artículo sobre el que se ha basado el trabajo, se daban detalles ambiguos que podían interpretarse de formas diferentes [12]. Se contactó con los autores que nos confirmaron la falta de detalles de implementación cruciales para reproducir los experimentos llevados a cabo por los autores.

Así, se han implementado dos aproximaciones diferentes: en la primera, se calcula un registro personalizado mediante la ejecución de una serie de iteraciones del algoritmo LDDMM de baseline alcanzando una solución aceptable y, a partir de ella, se recogen los *snapshots*. La motivación de este método de construcción de la base es viene dada por la resolución de otros problemas mediante POD, para los cuales se llevaba a cabo el experimento original durante un cierto intervalo de tiempo, y a partir de ahí, se recogían las *snapshots*, para posteriormente volver a comenzar el experimento, esta vez con el POD [9, 11]. En la segunda, se recogen los *snapshots* tras solo una iteración del algoritmo LDDMM, poniendo así a prueba la capacidad de generalización del algoritmo. En ambos casos se realizan una serie de iteraciones del algoritmo de *baseline* (1), mediante el cual se recogen una serie de instantáneas o *snapshots* de la velocidad (como la imagen 4.5). Cada una de estas *snapshots* se reordena como una columna de la matriz X, a la cual se le aplicará SVD para la obtención de la base U.

Para ilustrar el método descrito en el párrafo anterior, se presenta un caso particular de construcción de la matriz X, su descomposición mediante SVD, y la obtención de la base U. Se han calculado 50 iteraciones del algoritmo de baseline. Aprovechando que para obtener el campo vectorial de velocidades v mediante la ecuación de Euler-Poincaré 1.9 se realiza una integración mediante un algoritmo Runge-Kutta de 30 pasos, se recogen los 31 pasos de la última iteración del algoritmo, obteniendo 31 *snapshots* de los campos vectoriales de las valocidades. Posteriormente, se reordenan, de forma que se escribe cada *snapshot* como una columna de la matriz X. Debido a que las imágenes tienen de tamaño $80 \times 80 \times 2$, obtenemos una matriz A de dimensiones 12800×31. Así, si se representa dicha matriz como un mapa de color, se obtiene la siguiente imagen:

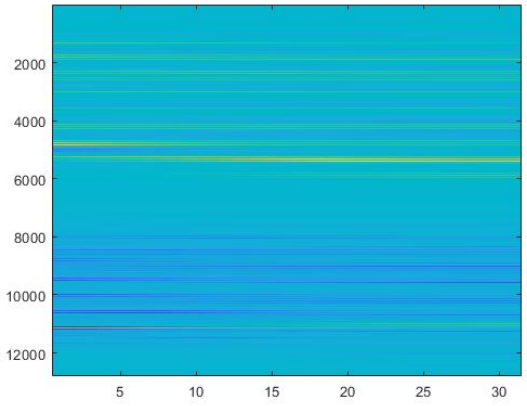
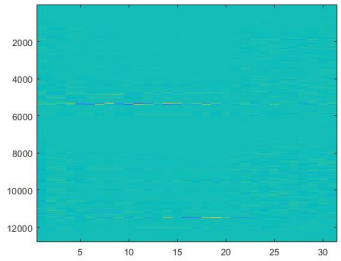
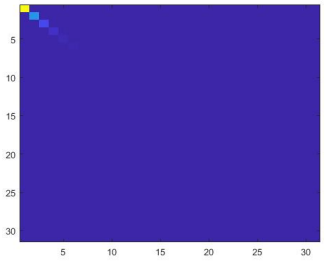


Figura 4.7: Representación de la matriz A que es una colección de *snapshots* del algoritmo LDDMM.

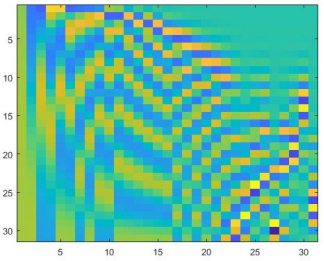
Tras aplicar SVD, hacemos una representación igual de las tres matrices resultantes, obteniendo:



(a) Matriz U de modos espaciales del sistema.



(b) Matriz Σ de energía de los modos espaciales.



(c) Matriz V de coeficientes temporales.

Figura 4.8: Representación en imágenes de la matriz de *snapshots* y su descomposición mediante SVD.

Al aplicarle SVD a la matriz A y descomponerla, se obtienen tres matrices: la matriz U, en la que se encuentra en cada columna un modo espacial del sistema (ordenados de más a menos energéticos), en la diagonal de Σ se hallan la energía correspondiente a cada modo (como se puede comprobar los más energéticos son los primeros), y en la matriz V, se encuentran los coeficientes temporales del sistema.

4.4.1. Experimento personalizado

El experimento personalizado será aquel en el que, previamente a ejecutar el algoritmo POD-LDDMM, se llevan a cabo un número determinado de iteraciones del algoritmo LDDMM. Lo que se consigue es la obtención de una solución tentativa, con la cual se construirá la matriz X para obtener la base U.

Para la elección del tamaño de la base r se ha utilizado el criterio expuesto en el apartado 2.3.1 mediante el cual se intenta maximizar la conservación de la energía: en la matriz de autovalores Σ cada autovalor corresponde con la energía asociada a un modo espacial del sistema. Así, se calculará el porcentaje de la energía de los modos elegidos, de forma que este sea como mínimo el 99 % del total (o, como se explica en el apartado, que el error sea $e < 0.01$.

A continuación, se presentan los experimentos llevados a cabo, en los que se ejecuta el algoritmo LDDMM 10, 20, 50 y 100 iteraciones antes de ejecutar el POD.

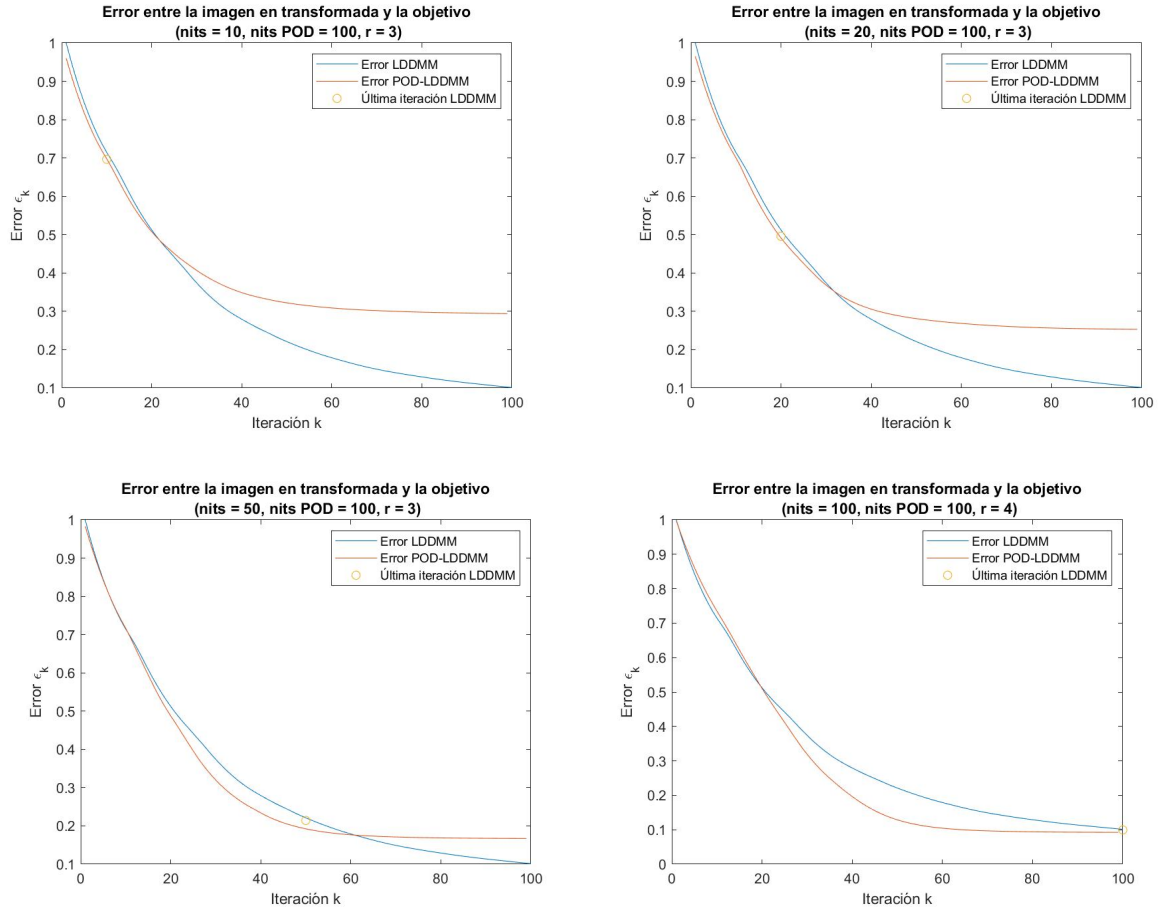


Figura 4.9: Error cuadrático medio de cada iteración en distintos experimentos personalizados.

En estas gráficas se representa el error del algoritmo LDDMM durante 100 iteraciones, comparado con el POD-LDDMM de también 100 iteraciones. El punto naranja indica el error que se obtiene tras la ejecución previa del LDDMM: es el error de la solución tentativa del algoritmo. Como se puede comprobar, a una mayor aproximación de esta solución tentativa del sistema, la convergencia del algoritmo es más rápida, pero alcanzando un error siempre menor o igual que el algoritmo original. Se puede alcanzar así un compromiso entre tiempo de computación y error alcanzado del algoritmo.

Iteraciones	$\mathbf{t}_{POD-LDDMM}$ **(s)**	**Error** (%)
10	101.34	29.39
20	101.47	25.31
50	102.80	16.72
100	108.25	9.24

Tabla 4.1: Resultados obtenidos mediante el POD-LDDMM en distintos experimentos personalizados.

4.4.2. Generalización de la base

A continuación, y como segunda propuesta de construcción de la base, se intenta comprobar la capacidad de generalización del algoritmo. Es decir, en el caso de que no haya una solución tentativa con la cual aproximarse a la solución final con POD, ver si el algoritmo es capaz de minimizar el error entre ambas imágenes. Para ello se realiza una iteración del algoritmo LDDMM, de forma que se tienen datos para construir una base, pero el error es próximo al 100 %, por lo que no se acerca a una solución tentativa. De esta forma, el resultado obtenido es el siguiente:

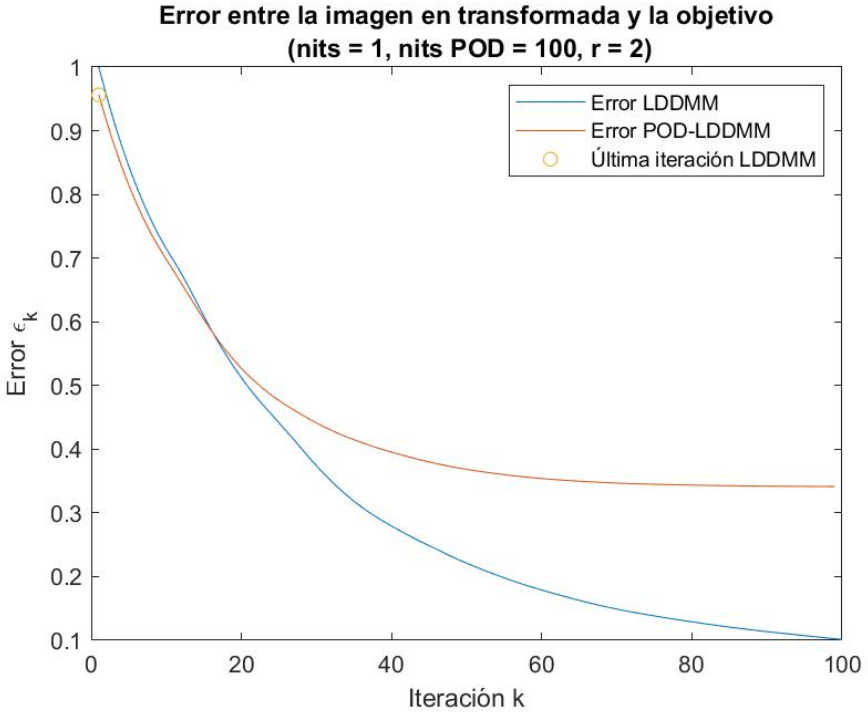


Figura 4.10: Error cuadrático medio del algoritmo POD-LDDMM tras una iteración del algoritmo LDDMM.

El error alcanzado es de 34.14 % en un tiempo de 102.53 s.

4.4.3. Dimensión r de la base

Tras realizar estos dos experimentos, se va a realizar la comprobación de la importancia del tamaño de la base en este problema. Se va a realizar el experimento personalizado de 50 iteraciones del LDDMM, y posteriormente se construirá la base U de diferentes tamaños.

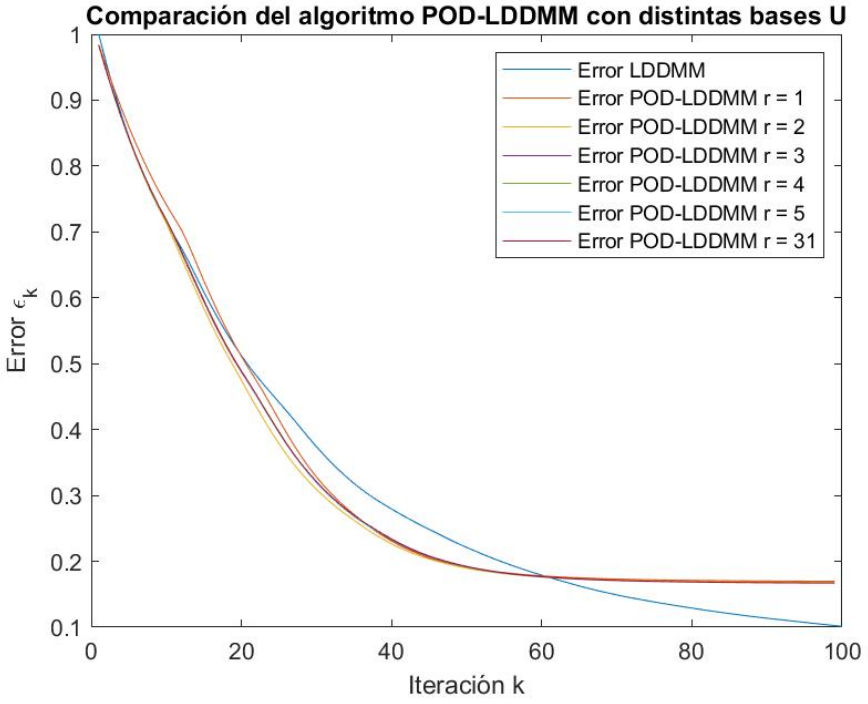


Figura 4.11: Error cuadrático medio en el algoritmo POD-LDDMM implementado con bases de distintos tamaños.

Los resultados se muestran en la siguiente tabla:

r	$\mathbf{t}_{POD-LDDMM}$ **(s)**	**Energía (%)**	**Error (%)**
1	98.62	98.56	16.97
2	102.64	99.95	16.82
3	102.80	100.00	16.72
4	108.80	100.00	16.71
5	110.11	100.00	16.71
31	143.33	100.00	16.70

Tabla 4.2: Resultados obtenidos mediante el POD-LDDMM con bases de distintos tamaños.

En esta tabla se recogen los siguientes datos, por columnas: la dimensión de la base U, el tiempo que tarda el algoritmo POD-LDDMM en computarse, el error de la energía de la base U (la suma de la energía de los modos elegidos dividido por la energía total del

sistema) y el error entre imágenes que alcanza el algoritmo.

Como se puede comprobar, el sistema está gobernado por el primer modo espacial, con el cual se alcanza un error del 16.97 % con un 98.56 % de la energía conservada. Aumentando el número de modos de la base U (añadiendo más detalles del sistema al POD) aumenta la energía conservada y disminuye el error que alcanza el algoritmo. Sin embargo, el margen de mejora es mínimo. Sí que se puede comprobar, que el tiempo de computación aumenta considerablemente. Con $r = 1$ se consigue que el algoritmo minimice su error en 98.62 s, lo que es un 68.81 % menos que con la base completa.

Por último, se van a mostrar las imágenes obtenidas y sus diferencias, para el mejor y el peor de los resultados del POD que se muestran en la tabla anterior 4.2.

En el caso en el que la reducción de la base viene dada por $r = 1$:

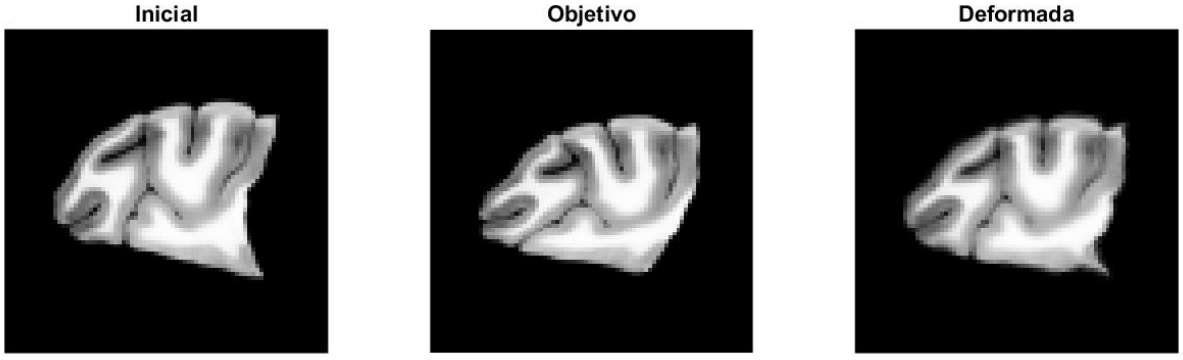


Figura 4.12: Imágenes inicial, objetivo, y deformada tras el registro.

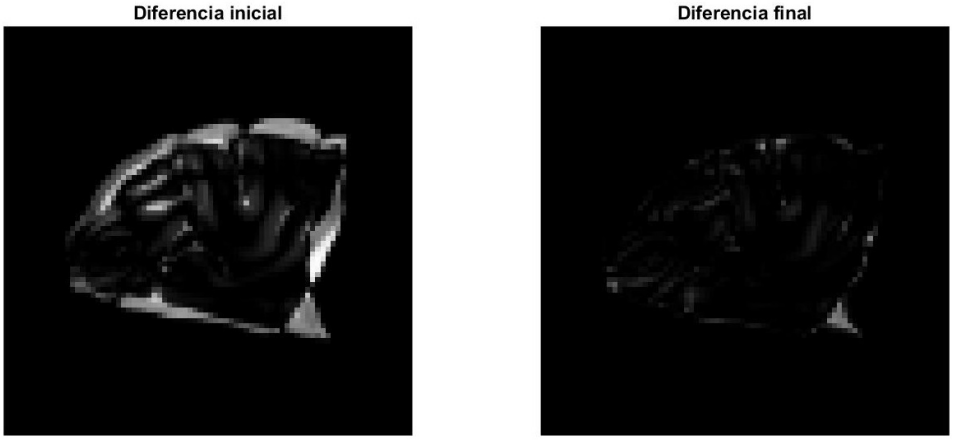
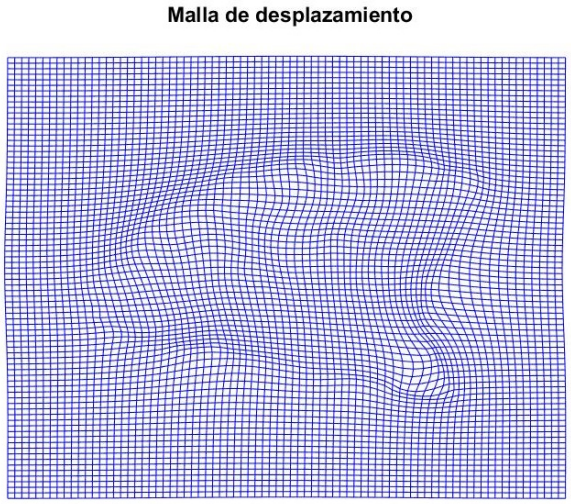


Figura 4.13: Diferencia entre las imágenes inicial y objetivo y deformada y objetivo tras la implementación del algoritmo.

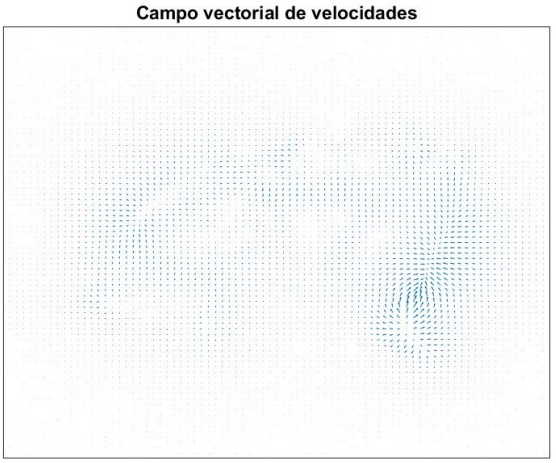
Como se puede comprobar, los resulados obtenidos también son buenos. De la misma forma que en la comparación hecha con los resultados del algoritmo LDDMM *baseline*

mostrado en la imagen 4.3, la diferencia entre las imágenes al inicio es notable, mientras que esta queda minimizada en el resultado final. Por la naturaleza del algoritmo POD, y como es visible en la imagen, donde más diferencia hay es entre los detalles de la imagen.

Finalmente se muestran tanto la malla de deformación como el campo vectorial de velocidadaes, tal y como en el primer ejemplo.



(a) Malla de desplazamiento de la imagen deformada



(b) Campo de velocidades de la imagen deformada

Figura 4.14: Imágenes que representan el movimiento de la imagen deformada.

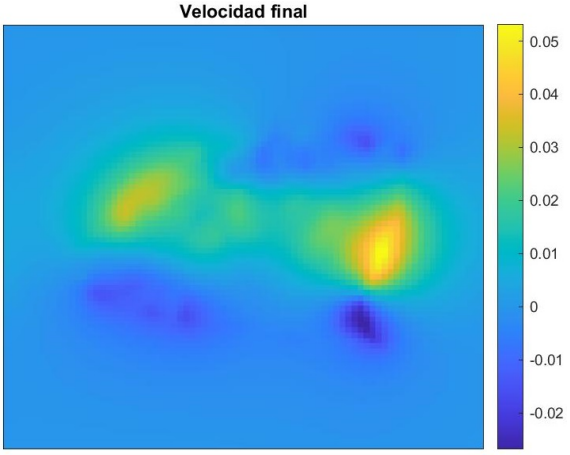


Figura 4.15: Mapa de color de la velocidad de la imagen deformada.

4.4.4. Imágenes reales

Como hasta ahora se ha visto que el método funciona en imágenes de prueba, se va a comprobar si este funciona en imágenes reales de cerebros humanos, que contienen más detalles y son más complejas que las utilizadas anteriormente. Los parámetros para llevar

a cabo la implementación son: $\alpha = 0.005$, $\gamma = 1.0$, $s = 1.0$ y $\sigma = 1.0$. A continuación se presentan las imágenes utilizadas y la imagen final deformada:

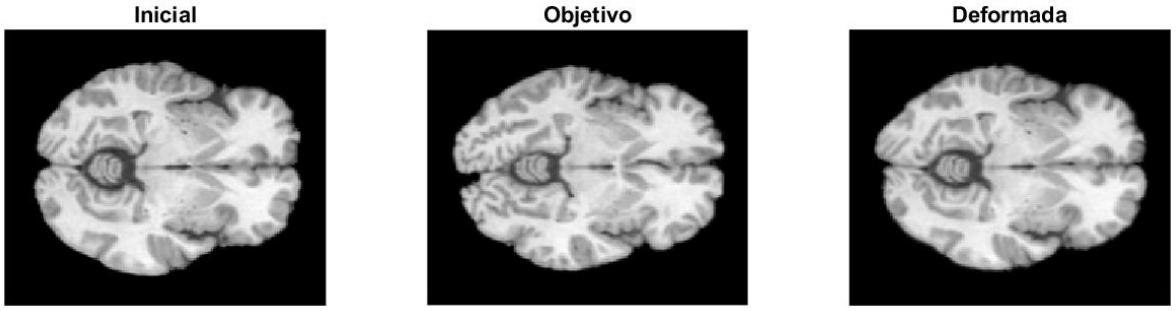


Figura 4.16: Imagen inicial, objetivo, y deformada tras el registro.

Las diferencias entre imágenes son las siguientes:

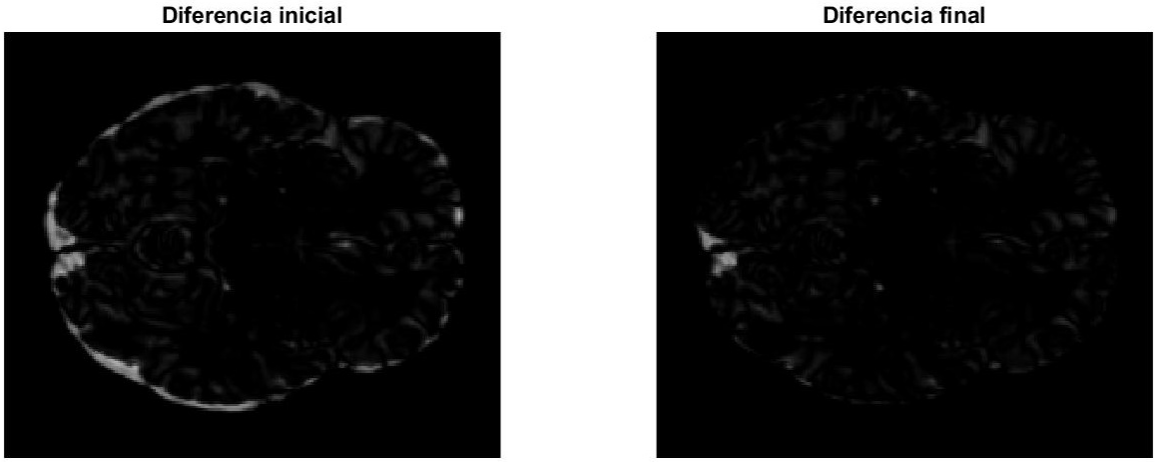
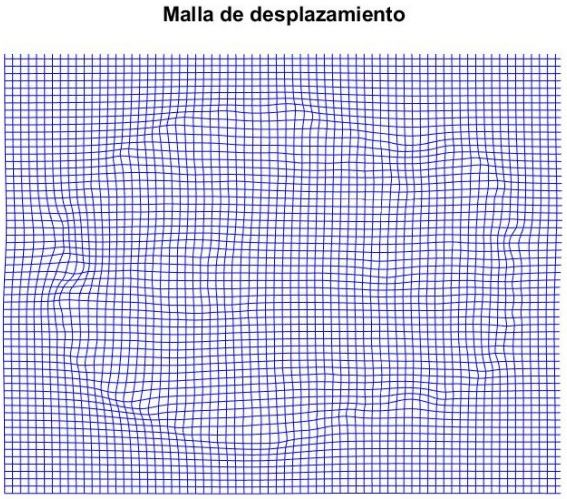


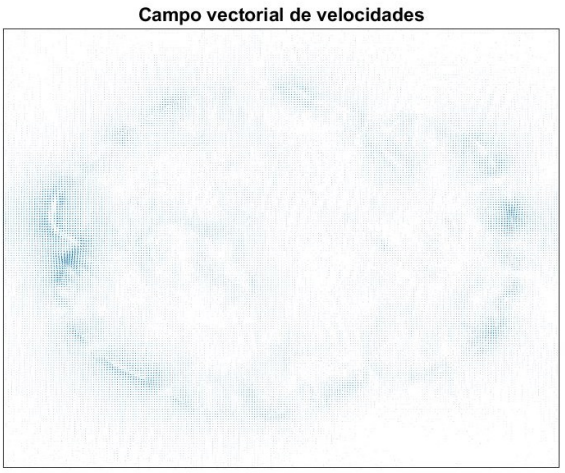
Figura 4.17: Imagen original que se va a transformar en escala de grises comprimir.

Como se puede comprobar, la diferencia entre las imágenes deformada final e imagen inicial es menor que entre las imágenes deformada inicial e imagen inicial, por lo que se concluye que el algoritmo funciona correctamente y minimiza la diferencia entre las imágenes. En este caso, el error alcanzado es de 43.35 % en 456.45 s. Tanto el error como el tiempo de computación es mayor que en los otros casos de ejemplo debido al mayor tamaño y complejidad de las imágenes utilizadas.

Finalmente, y al igual que en los ejemplos anteriores, se muestran la malla de desplazamiento, el campo vectorial de velocidades y el mapa de color de las velocidades:



(a) Malla de desplazamiento de la imagen deformada



(b) Campo de velocidades de la imagen deformada

Figura 4.18: Imágenes que representan el movimiento de la imagen deformada.

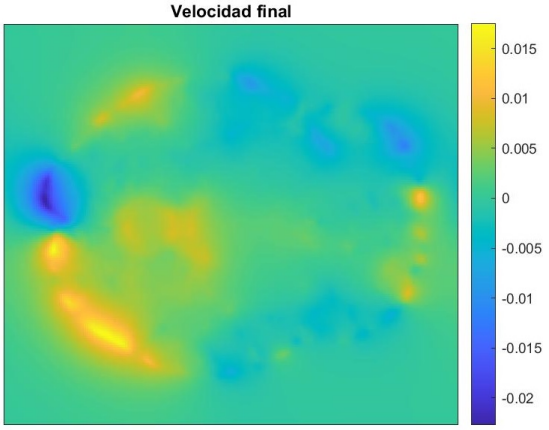


Figura 4.19: Mapa de color de la velocidad de la imagen deformada.

Capítulo 5

Conclusiones y trabajo futuro

En este trabajo se ha estudiado la parametrización del problema de registro difeomorfo de imágenes Mapeo Métrico Difeomorfo de Grandes Deformaciones (LDDMM) mediante un Modelo de Orden Reducido (ROM) como es la Descomposición Ortogonal Propia (POD). Para ello, se ha estudiado el método descrito en [12] y se ha implementado el algoritmo propuesto en 2D partiendo de los códigos de los métodos propuestos en [15, 16]. Se ha realizado un trabajo de corregir y detallar ciertos aspectos del artículo [12] tales como la creación de la matriz de *snapshots* y la proyección de las ecuaciones de Jacobi en el espacio reducido. Se ha mostrado el correcto funcionamiento de la implementación tanto para imágenes de cerebros de macacos como resonancias magnéticas de cerebros humanos.

Los registros obtenidos han mostrado aumentar substancialmente la similitud entre las imágenes, y la reducción de la dimensionalidad aportada por POD ha conseguido una reducción del tiempo de computación del algoritmo. Sin embargo, se han observado diferentes debilidades comentadas a continuación: La primera de ellas, es la alta dependencia del algoritmo de la entrada utilizada para la construcción de la matriz de *snapshots*. Esta ha sido una de las grandes dificultades en la implementación, debido al conjunto de opciones posibles y la ambigüedad mostrada en el artículo al respecto. Se ha llegado a la conclusión de que un experimento personalizado es la mejor opción para obtener un registro preciso. En segundo lugar, la elección del tamaño de la base influye considerablemente en el tiempo de computación. Así, se hace necesario encontrar un compromiso entre estas variaciones de forma que compense la implementación del POD-LDDMM frente al LDDMM de *baseline* en cuanto a tiempo de computación y error final en el registro.

Con todo, como futuras líneas de desarrollo se proponen varios temas:

1. La construcción de una matriz de *snapshots* generalizada a partir de una gran cantidad de datos de entrenamiento, para posteriormente poder ser aplicada a imágenes de test, y así no depender de la realización de experimentos personalizados para la obtención de buenos resultados.

2. La utilización de otros métodos de reducción de la dimensionalidad, como pueden ser los *autoencoders*, para la resolución del problema LDDMM.

3. La implementación de POD en otros algoritmos LDDMM y la comparación de las prestaciones de POD respecto a la parametrización de banda limitada propuesta en [15].

Bibliografía

[1] Lisa Gottesfeld Brown. "A survey of image registration techniques". En: *ACM computing surveys (CSUR)* 24.4 (1992), págs. 325-376.

[2] Barbara Zitova y Jan Flusser. "Image registration methods: a survey". En: *Image and vision computing* 21.11 (2003), págs. 977-1000.

[3] Michael I Miller. "Computational anatomy: shape, growth, and atrophy comparison via diffeomorphisms". En: *NeuroImage* 23 (2004), S19-S33.

[4] Mirza Faisal Beg y col. "Computing Large Deformation Metric Mappings via Geodesic Flows of Diffeomorphisms." En: *International Journal of Computer Vision* 61.2 (2005), págs. 139-157. URL: https://doi.org/10.1023/B:VISI.0000043755.93987.aa.

[5] Mim.cis. *Faisal Beg LDDMM Algorithnm.png*. This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit https://creativecommons.org/licenses/by-sa/4.0/. 2016. URL: https://commons.wikimedia.org/wiki/File:Faisal_Beg_LDDMM_Algorithnm.png#/media/File:Faisal_Beg_LDDMM_Algorithnm.png.

[6] Miaomiao Zhang y P. Thomas Fletcher. "Finite-Dimensional Lie Algebras for Fast Diffeomorphic Image Registration". En: (2015). Ed. por Sebastien Ourselin y col., págs. 249-260.

[7] Vladimir. Arnold. "Sur la géométrie différentielle des groupes de Lie de dimension infinie et ses applications à l'hydrodynamique des fluides parfaits." En: (1966), págs. 319-361. URL: http://www.numdam.org/articles/10.5802/aif.233/.

[8] Francesco. Bullo. "Inviariant affine connections and controlability on the Lie groups." En: (1995).

[9] Steven L. Brunton y J. Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019. DOI: 10.1017/9781108380690.

[10] Ben Noble y James W Daniel. *Applied linear algebra*. Englewood Cliffs, NJ : Prentice-Hall, 2008.

[11] Michele Girfoglio, Annalisa Quaini y Gianluigi Rozza. "A POD-Galerkin reduced order model for the Navier-Stokes equations in stream function-vorticity formulation". En: *arXiv preprint arXiv:2201.00756* (2022).

[12] Jian Wang y col. "Data-driven Model Order Reduction For Diffeomorphic Image Registration". En: mar. de 2019.

[13] Charbel Farhat y Spencer Anderson. *Model Reduction*. Department of Aeronautics and Astronautics, University of Standfor. Spring de 2017.

[14] Joo Hyun Song y col. "Evaluating image registration using NIREP". En: *International Workshop on Biomedical Image Registration*. Springer. 2010, págs. 140-150.

[15] Monica Hernandez. “A Comparative Study of Different Variants of Newton–Krylov PDE-Constrained Stokes-LDDMM Parameterized in the Space of Band-Limited Vector Fields”. En: *SIAM Journal on Imaging Sciences* 12.2 (2019), págs. 1038-1070.

[16] Monica Hernandez. “Efficient momentum conservation constrained PDE-LDDMM with Gauss–Newton–Krylov optimization, Semi-Lagrangian Runge–Kutta solvers, and the band-limited parameterization”. En: *Journal of Computational Science* 55 (2021), pág. 101470.