

**PROYECTO INTEGRADOR DE LA CARRERA DE  
INGENIERÍA NUCLEAR**

**APLICACIÓN DE TÉCNICAS DE REDUCCIÓN  
DIMENSIONAL EN EL MODELADO DE FLUJOS  
INESTABLES**

**Martín Madrid Wagner**  
**Estudiante**

**Dr. William Machaca Abregu**  
Director

**Dr. Eugenio Urdapilleta**  
Co-director

**Miembros del Jurado**  
Dr. Federico Teruel  
Dr. Luis Gregorio Moyano

Junio de 2024

Departamento de Mecánica Computacional - Centro Atómico Bariloche

Instituto Balseiro  
Universidad Nacional de Cuyo  
Comisión Nacional de Energía Atómica  
Argentina



*A mis viejos y a mis abuelas.*

*A Magdalena.*



# Índice de contenidos

<b>Índice de contenidos</b>	v
<b>Índice de figuras</b>	ix
<b>Índice de tablas</b>	xv
<b>Resumen</b>	xvii
<b>Abstract</b>	xix
<b>1. Introducción</b>	1
1.1. Motivación . . . . .	1
1.2. Reducción de la dimensionalidad . . . . .	2
1.3. Problema de estudio: flujo alrededor de un cilindro . . . . .	4
1.4. Objetivos . . . . .	6
<b>2. Simulación numérica directa con Xcompact3d</b>	9
2.1. Xcompact3d . . . . .	9
2.1.1. Descripción del <i>software</i> y métodos numéricos . . . . .	10
2.1.2. Validación DNS 3D . . . . .	11
2.1.3. Validación DNS 2D . . . . .	13
2.2. Base de datos: DNS del flujo alrededor de un cilindro . . . . .	15
<b>3. Reducción dimensional</b>	19
3.1. Notación y arreglo de datos . . . . .	19
3.1.1. Reducción espacial y temporal . . . . .	21
3.2. Descomposición por valores singulares (SVD) . . . . .	21
3.2.1. Interpretación de la SVD . . . . .	23
3.3. Análisis de Componentes Principales (PCA) . . . . .	24
3.3.1. Algoritmo PCA . . . . .	24
3.3.2. Interpretación geométrica . . . . .	27

3.3.3. Costo computacional . . . . .	27
3.4. Kernel PCA (KPCA) . . . . .	28
3.4.1. Algoritmo KPCA . . . . .	29
3.4.2. Reconstrucción . . . . .	32
3.4.3. Costo computacional . . . . .	32
3.5. Locally Linear Embedding (LLE) . . . . .	33
3.5.1. Algoritmo LLE . . . . .	33
3.5.2. Reconstrucción . . . . .	36
3.5.3. Costo computacional . . . . .	37
3.6. Isometric Mapping (Isomap) . . . . .	38
3.6.1. Algoritmo Isomap . . . . .	38
3.6.2. Reconstrucción . . . . .	40
3.6.3. Costo computacional . . . . .	40
3.7. Autoencoder (AE) . . . . .	41
3.7.1. Redes neuronales artificiales . . . . .	42
3.7.2. Redes neuronales profundas . . . . .	43
3.7.3. Entrenamiento de una ANN . . . . .	44
3.7.4. Autoencoder . . . . .	47
3.8. Conjuntos de entrenamiento y prueba . . . . .	48
<b>4. Resultados</b>	<b>51</b>
4.1. Metodología computacional . . . . .	51
4.1.1. Base de datos y preprocesamiento . . . . .	51
4.1.2. Métrica de rendimiento . . . . .	53
4.1.3. Herramientas computacionales . . . . .	53
4.2. PCA . . . . .	54
4.3. KPCA . . . . .	57
4.4. LLE . . . . .	59
4.5. Isomap . . . . .	63
4.6. Autoencoder . . . . .	65
4.7. Comparación de métodos (Re=100) . . . . .	66
<b>5. Conclusiones</b>	<b>71</b>
5.1. Resumen y conclusiones . . . . .	71
5.2. Recomendaciones . . . . .	73
<b>A. Autoencoder</b>	<b>75</b>
A.1. Arquitecturas y pérdida . . . . .	75

<b>B. Práctica Profesional Supervisada (PPS) y Actividades de Proyecto y Diseño (P&amp;D)</b>	<b>77</b>
B.1. Práctica Profesional Supervisada (PPS) . . . . .	77
B.2. Actividades de Proyecto y Diseño (P&D) . . . . .	77
<b>Bibliografía</b>	<b>79</b>
<b>Agradecimientos</b>	<b>83</b>



# Índice de figuras

1.1.	Patrones típicos en flujos. En el lado izquierdo, se muestra el fenómeno del desprendimiento de vórtices de von Kármán cuando las nubes pasan sobre la isla Rishiri, Japón. A la derecha se muestra la inestabilidad de Kelvin-Helmholtz, también en nubes. . . . .	2
1.2.	Reducción dimensional lineal. Los datos en 3D se proyectan a un subespacio de menor dimensión (2D), que captura la mayor cantidad de varianza posible en el conjunto original. Imagen extraída y editada de [6]. . . . .	3
1.3.	Ejemplo de un conjunto de datos en forma de <i>Swiss roll</i> . Si bien el conjunto pertenece a $\mathbb{R}^3$ , la estructura subyacente es de dimensión 2. Imagen extraída y editada de [6]. . . . .	3
1.4.	Proyección lineal (izquierda) vs. <i>unroll</i> no-lineal (derecha) de un <i>Swiss roll</i> . Imagen extraída y editada de [6]. . . . .	4
1.5.	Patrones presentes en el flujo alrededor de un cilindro distintos números de Reynolds. Imagen extraída y editada de [9]. . . . .	6
2.1.	Esquema espacial de los problemas simulados: flujo turbulento en canal de placas paralelas (Canal) y flujo alrededor de un cilindro (Cilindro). . . .	11
2.2.	Estadísticas de la velocidad en la dirección de la corriente del flujo turbulento en un canal periódico de $Re = 4200$ de las simulaciones DNS 3D con Xcompact3d (línea sólida) y de los datos de referencia de [20] ( $\times$ ) y [21] ( $+$ ). . . . .	13
2.3.	Estadísticas de la velocidad en la dirección de la corriente del flujo alrededor de un cilindro a $Re = 300$ de las simulaciones DNS 3D Xcompact3d (línea sólida) y de los datos de referencia de [22] ( $\diamond$ ). Se comparan resultados para tres mallas distintas. Las diferentes curvas corresponden a los perfiles a $1D$ , $2,5D$ , $5D$ , $7,5D$ y $10D$ diámetros detrás del cilindro. . . . .	14
2.4.	Comparación de las estadísticas de la velocidad del flujo alrededor de un cilindro para $Re = 50$ , $100$ y $200$ , obtenidos por DNS 2D ( $\times$ ) y 3D ( $+$ ) en Xcompact3d. Los resultados se grafican a distancias axiales $1D$ , $5D$ y $10D$ detrás del cilindro, donde $D$ es el diámetro del cilindro. . . . .	15

2.5. Instantáneas en el plano x-y de la componente horizontal de la velocidad $u_x$ (naranja), y el módulo de la vorticidad (azul) obtenidas por DNS 2D del flujo alrededor de un cilindro en Xcompact3d para distintos Reynolds.	17
3.1. Esquema del arreglo de datos a partir de las simulaciones DNS, dispuestos en una matriz $\mathbf{X} \in \mathbb{R}^{n \times m}$ , con $n = n_x \times n_y$ .	20
3.2. Esquema de la SVD completa y económica, donde $\mathbf{X}$ es la reconstrucción a partir de los $m$ valores singulares no nulos. $\hat{\mathbf{U}}^\perp$ es el complemento ortogonal de $\hat{\mathbf{U}}$ .	22
3.3. Interpretación de la SVD en el contexto de las simulaciones DNS. Las columnas $\mathbf{u}$ representan modos espaciales y las columnas $\mathbf{v}$ su evolución en el tiempo. En la imagen se representa la aproximación de rango 3. Notar que $\boldsymbol{\mu}\mathbf{1}_m = \sigma_1\mathbf{u}_1\mathbf{v}_1^\top$ , donde $\boldsymbol{\mu}$ es el flujo promedio y $\mathbf{1}_m \in \mathbb{R}^m$ un vector de unos.	24
3.4. Idea detrás del PCA en un conjunto de datos 2D. Los CPs $\mathbf{C}_1$ y $\mathbf{C}_2$ son las direcciones de mayor varianza. Se muestra un eje intermedio adicional a. A la derecha se representan las proyecciones de los datos en las tres direcciones. Imagen editada, extraída de la Ref. [6].	25
3.5. Interpretación geométrica del PCA en un conjunto de datos 2D. Los CPs son los ejes de la elipse que mejor se ajusta a los datos.	28
3.6. Resumen del algoritmo PCA clásico. En la práctica, no se calcula la matriz de covarianza, dado que los autovectores y autovalores se obtienen directamente de la SVD.	29
3.7. Representación conceptual del Kernel PCA. Los datos son mapeados a un espacio de mayor dimensión $\mathcal{F}$ donde se encuentran aproximadamente en un subespacio lineal. Cuando la función de <i>kernel</i> es el producto interno usual, se recupera el PCA lineal. Imagen extraída de la Ref. [24].	30
3.8. Ilustración del problema de la pre-imagen en el KPCA. La $\mathbf{X}$ representa un punto en el espacio latente, cuya reconstrucción está en $\mathcal{F}$ . La pre-imagen es el punto en el espacio original que se mapearía cerca del punto reconstruido. Imagen editada, extraída de la Ref. [6].	33
3.9. Resumen del algoritmo KPCA. Los primeros tres pasos corresponden a la construcción del espacio latente, y el cuarto a la reconstrucción de los datos a la dimensión original.	34

---

3.10. Pasos del algoritmo LLE. (a) Identificación de los $K$ -NN $\mathbf{x}_i$ en el espacio original (b) a partir de los cuales se optimiza la reconstrucción lineal de $\mathbf{x}_i$ . (c) Determinación de los $\mathbf{y}_i$ en el espacio latente usando los pesos calculados en (b). En la figura se representan 4 $K$ -NN en rojo. Figura tomada de la Ref. [7]. . . . .	34
3.11. Resumen del algoritmo LLE. Los primeros tres pasos corresponden a la construcción del espacio latente, y el cuarto a la reconstrucción de los datos a la dimensión original. . . . .	37
3.12. Distancia euclídea y geodésica entre dos puntos arbitrarios (redondeados) en un conjunto de datos 3D. (A) Las líneas punteada y sólida representan la distancia euclídea y geodésica (real), respectivamente. (B) El grafo de vecinos cercanos (líneas grises) permite aproximar la distancia geodésica real por el camino más corto por $K$ -NN (línea roja). (C) Espacio latente 2D obtenido por Isomap que preserva mejor la distancia geodésica. Imagen tomada de la Ref. [31]. . . . .	39
3.13. Resumen del algoritmo Isomap. Los primeros tres pasos corresponden a la construcción del espacio latente, y el cuarto a la reconstrucción de los datos a la dimensión original. . . . .	41
3.14. Ejemplo de una ANN completamente conectada con una capa oculta. La red tiene tres entradas ( $x$ ), dos salidas ( $y$ ) y una capa oculta con tres neuronas ( $h$ ). Imagen extraída de la Ref. [35]. . . . .	42
3.15. Cada neurona en una capa recibe entradas de las neuronas en la capa anterior, suma un sesgo y aplica una función de activación. Luego, envía la salida de la capa a las neuronas en la capa siguiente. Figura extraída y editada de [35]. . . . .	44
3.16. Arquitectura de un AE donde el <i>encoder</i> y el <i>decoder</i> comparten una estructura idéntica e inversa. Figura tomada de la Ref. [37]. . . . .	48
4.1. Dominio recortado ( $L'_x, L'_y$ ) de la simulación del flujo alrededor de un cilindro. Los puntos dentro del cilindro se excluyen de los datos. . . . .	52
4.2. Espectro de valores propios obtenidos mediante la SVD de la matriz $\mathbf{X}_{\text{train}}$ . (a) Valores singulares en escala normal y (b) en escala semi-log. (c) Energía acumulativa normalizada en función de los modos. Las líneas punteadas rojas indican el 99,9 % de la energía acumulativa, alcanzada en 31 modos. . . . .	55
4.3. Error relativo de reconstrucción en función de la cantidad de modos preservados en la reconstrucción de datos reducidos por PCA. Se grafican las curvas para cada $Re$ . . . . .	56
4.4. Flujo promedio alrededor de un cilindro de $Re = 50, 100, 150$ y $200$ . . . . .	56

4.5. Modos espaciales 1, 3, 5 y 7, para cada $Re$ . La magnitud de la velocidad en cada modo se normaliza entre 0 y 1 para facilitar la visualización. . . . .	56
4.6. Error relativo de reconstrucción en función de la cantidad de modos preservados de la reducción espacial mediante KPCA. Se grafican las curvas para cada $Re$ y distintos valores de $\gamma$ . . . . .	57
4.7. Error relativo de reconstrucción en función de la cantidad de modos preservados de la reducción temporal mediante KPCA. Se grafican las curvas para cada $Re$ y distintos valores de $\gamma$ . . . . .	58
4.8. Error relativo de reconstrucción en función de la cantidad de modos preservados de la reducción espacial y temporal mediante KPCA. Las curvas correspondientes a la reducción espacial y temporal se grafican para $\gamma = 0,1$ y $\gamma = 0,001$ , respectivamente. . . . .	59
4.9. Error relativo de reconstrucción en función de la cantidad de modos preservados de la reducción espacial mediante LLE. Se grafican las curvas para cada $Re$ y distintos números de $K. . . . .$	60
4.10. Error relativo de reconstrucción en función de la cantidad de modos preservados de la reducción temporal mediante LLE. Se grafican las curvas para cada $Re$ y distintos números de $K. . . . .$	61
4.11. Error relativo de reconstrucción en función de la cantidad de modos preservados, para cada $Re$ . Las curvas de la reducción espacial corresponden a $K = 25$ y de la reducción temporal a $K = 200$ . . . . .	62
4.12. Error relativo de reconstrucción en función de la cantidad de modos preservados de la reducción espacial mediante Isomap. Se grafican las curvas para cada $Re$ y distintos números de $K. . . . .$	63
4.13. Error relativo de reconstrucción en función de la cantidad de modos preservados de la reducción temporal mediante Isomap. Se grafican las curvas para cada $Re$ y distintos números de $K. . . . .$	64
4.14. Error relativo de reconstrucción en función de la cantidad de modos preservados, para cada $Re$ . Las curvas de la reducción espacial corresponden a $K = 50$ y de la reducción temporal a $K = 200$ . . . . .	65
4.15. Error relativo de reconstrucción en función de la cantidad de modos preservados, para la reducción espacial y temporal mediante <i>autoencoders</i> . Las curvas correspondientes a la reducción espacial se grafican para 2, 4 y 8 modos, y para la reducción temporal para 2, 4, 8 y 12 modos. . . . .	65





# Índice de tablas

2.1. Parámetros de entrada utilizados en las simulaciones de del flujo turbulento en un canal de placas paralelas (Canal) y alrededor de un cilindro (Cilindro). Los $n_i$ son el número de nodos en la dirección $i$ , $L_i$ es la longitud del dominio en la dirección $i$ , $\Delta t$ es el paso de tiempo, $Re$ es el número de Reynolds, $r$ es el radio del cilindro, $c$ es el centro del cilindro, y $x1, xn, y1, yn$ son las posiciones de las fronteras. . . . .	12
2.2. Expresiones de las cantidades adimensionalizadas utilizadas en la validación del flujo turbulento en un canal de placas paralelas. $u'_x, u'_y, u'_z$ son las fluctuaciones de la velocidad en las direcciones $x, y$ y $z$ , respectivamente.	13
3.1. Algoritmos de RD y su aplicación, dado los conjuntos de entrenamiento $\hat{\mathbf{X}}_{\text{train}}$ y prueba $\hat{\mathbf{X}}_{\text{test}}$ . . . . .	49
A.1. Estructuras del <i>autoencoder</i> para reducción espacial (AE-S) y temporal (AE-T). La S y T corresponden a reducción espacial y temporal respectivamente. El número al final del nombre corresponde al número de neuronas de la <i>bottleneck</i> (dimensión reducida). La estructura del <i>decoder</i> es la inversa del <i>encoder</i> . . . . .	75



# Resumen

La dinámica de fluidos computacional (CFD) se caracteriza por producir datos espacio-temporales de alta dimensión. Por lo tanto, identificar un conjunto óptimo de coordenadas para representar los datos en un subespacio latente de baja dimensión es un primer paso hacia el desarrollo de modelos de orden reducido. El abordaje tradicional es mediante el Análisis de Componentes Principales (PCA), que da una aproximación lineal óptima. Sin embargo, en general los flujos son complejos e inherentemente no-lineales, lo cual limita la capacidad de representación en baja dimensión del PCA. En consecuencia, recientemente se han comenzado a aplicar en el campo del CFD algoritmos de reducción de dimensionalidad (RD) no-lineales, originalmente desarrollados en el área del aprendizaje automático. En este contexto, en el presente trabajo se busca implementar y comparar diferentes métodos de RD, lineal y no-lineales, en un problema canónico de flujo inestable: el flujo alrededor de un cilindro inmerso en un flujo uniforme. Para ello, en primer lugar se genera una base de datos a partir de simulaciones numéricas directas (DNS) con el software Xcompact3d a diferentes números de Reynolds. Luego se implementan las distintas técnicas de RD. Las mismas incluyen el estándar lineal PCA, que actúa como referencia para comparar el rendimiento de los métodos no-lineales implementados: Kernel PCA (KPCA), Locally Linear Embedding (LLE), Isometric Mapping (Isomap) y *autoencoders* (AE). En este marco, se comparan cuantitativamente las calidades de reconstrucción de estos métodos. Las técnicas no-lineales presentan mejores resultados en la reducción espacial, pero no superan al PCA en la reducción temporal. Sin embargo, la disposición temporal de los datos permite extraer modos espaciales visualizables que resaltan las principales estructuras características del flujo, facilitando la comparación cualitativa de los métodos.

**Palabras clave:** REDUCCIÓN DIMENSIONAL, PCA, LLE, KPCA, ISOMAP, AUTO-ENCODER, XCOMPACT3D, DNS



# Abstract

Computational fluid dynamics (CFD) is characterized by producing high-dimensional spatiotemporal data. Therefore, identifying an optimal set of coordinates to represent the data in a low-dimensional latent subspace is a first step toward the development of reduced-order models. The traditional approach is Principal Component Analysis (PCA), which provides an optimal linear approximation. However, fluid flows are generally complex and inherently nonlinear, limiting the low-dimensional representation capability of PCA. Consequently, non-linear dimensionality reduction (DR) algorithms, originally developed in the field of machine learning, have recently begun to be applied in the field of CFD.

In this context, this work aims to implement and compare different linear and non-linear DR methods over a canonical unsteady flow problem: the flow around a cylinder immersed in a uniform stream. To this end, a database is generated from direct numerical simulations (DNS) using the Xcompact3d software at different Reynolds numbers. The DR techniques studied include the standard linear PCA, which serves as a reference for comparing the performance of the implemented nonlinear methods: Kernel PCA (KPCA), Locally Linear Embedding (LLE), Isometric Mapping (Isomap), and autoencoders (AE). The reconstruction quality of these methods is quantitatively compared. Nonlinear techniques show better results in spatial reduction, but do not outperform PCA in temporal reduction. However, the temporal arrangement of the data allows for the extraction of visualizable spatial modes that highlight the main characteristic structures of the flow, facilitating the qualitative comparison of the methods.

**Keywords:** DIMENSIONALITY REDUCTION, PCA, LLE, KPCA, ISOMAP, AUTOENCODER, XCOMPACT3D, DNS



# Capítulo 1

## Introducción

*“Trabajo, trabajo...”*

— Peón orco

### 1.1. Motivación

La dinámica de fluidos computacional (CFD) se ha convertido en una herramienta esencial para el estudio y análisis de la dinámica de fluidos. A pesar de los avances en las capacidades computacionales, la simulación numérica de flujos turbulentos sigue siendo un desafío debido a su alta dimensionalidad y no-linealidad. Incluso para flujos laminares, las simulaciones pueden producir millones de grados de libertad en el espacio y miles de pasos de tiempo, generando así una enorme cantidad de datos. Esta alta dimensionalidad suele ser producto de la resolución del CFD, determinada por la necesidad de una precisión específica en un dominio espacio-temporal discreto. Luego, la reducción de estas cantidades de datos de gran tamaño se vuelve cada vez más importante.

Sin embargo, algo notable es que en muchos casos, es posible identificar características físicas distintivas que se comparten a través de una variedad de flujos e incluso en un amplio rango de parámetros como el número de Reynolds y el número de Mach. Entre fenómenos comunes de este tipo se incluyen el desprendimiento de vórtices de von Kármán y la inestabilidad de Kelvin-Helmholtz (ver [Fig. 1.1](#)). Estas características se pueden identificar fácilmente a simple vista, en escalas muy distantes, e incluso bajo la presencia de perturbaciones. En consecuencia, surge la expectativa de que estas características pueden ser extraídas mediante algún procedimiento matemático y proporcionen un medio para describir el flujo completo [1]. Basándonos en estas consideraciones, la premisa de este trabajo es que muchos problemas de flujo complejos residen en una variedad oculta de baja dimensión, que una vez identificada, podría reducir drásticamente la alta

dimensionalidad de los datos [2, 3].

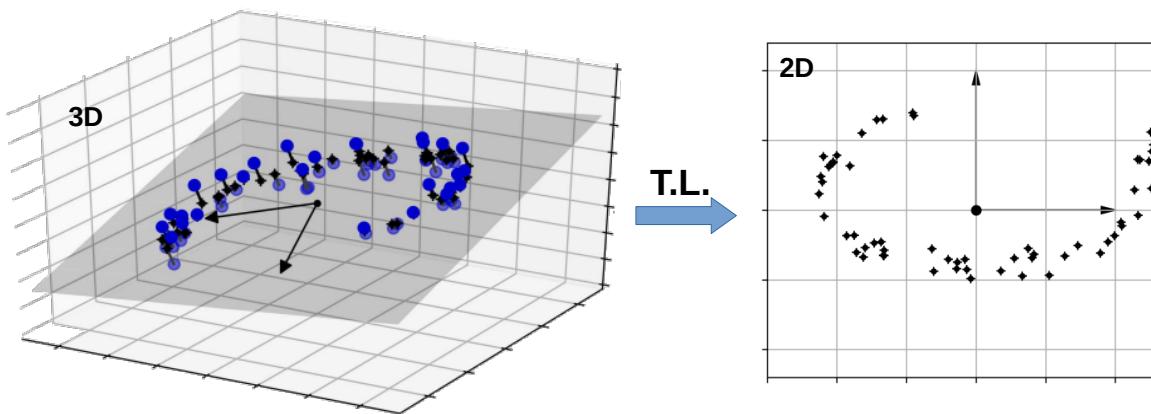


**Figura 1.1:** Patrones típicos en flujos. En el lado izquierdo, se muestra el fenómeno del desprendimiento de vórtices de von Kármán cuando las nubes pasan sobre la isla Rishiri, Japón. A la derecha se muestra la inestabilidad de Kelvin-Helmholtz, también en nubes.

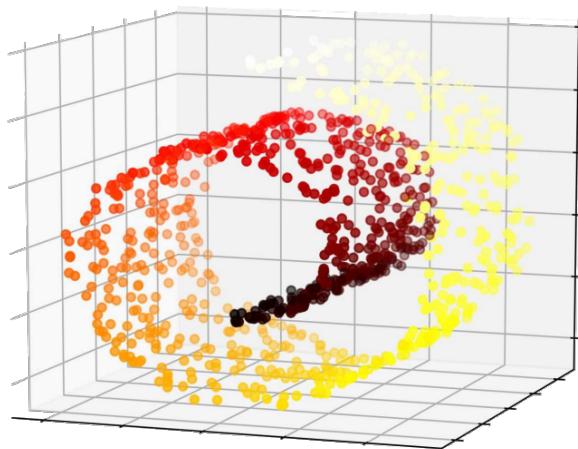
Los modelos de orden reducido (ROMs) tienen como objetivo descubrir esta baja dimensionalidad y capturar las características más importantes del flujo. El enfoque estándar para desarrollar ROMs en mecánica de fluidos es encontrar un nuevo sistema de coordenadas de baja dimensión utilizando el análisis de componentes principales (PCA), también conocido como descomposición ortogonal propia (POD), y luego derivar la dinámica del sistema mediante la proyección de Galerkin de las ecuaciones de Navier-Stokes en estos modos [4, 5]. El PCA es un método que mediante transformaciones lineales asigna datos de alta dimensión a un espacio de dimensión inferior. No obstante, el hecho de que sea lineal impone limitaciones al momento de capturar la estructura no-lineal del flujo. Por lo tanto, en los últimos años, se ha propuesto la aplicación de varios métodos no-lineales de reducción de dimensionalidad.

## 1.2. Reducción de la dimensionalidad

La reducción de dimensionalidad (RD) es un proceso que consiste en transformar datos de alta dimensión en un espacio de menor dimensión, conservando la mayor cantidad de información posible. Los dos abordajes principales son la RD lineal (proyección) y la RD no-lineal. La RD lineal es un método para asignar datos de alta dimensión a un espacio de dimensión inferior utilizando transformaciones lineales. Por ejemplo, los datos pueden ser proyectados a un subespacio de menor dimensión, capturando la mayor cantidad de varianza posible en los datos originales. Este enfoque puede ser ideal cuando los datos originales yacen aproximadamente en un subespacio lineal, como en el caso de la Fig. 1.2. Esto resume lo que hace el PCA, el método estándar en la RD.



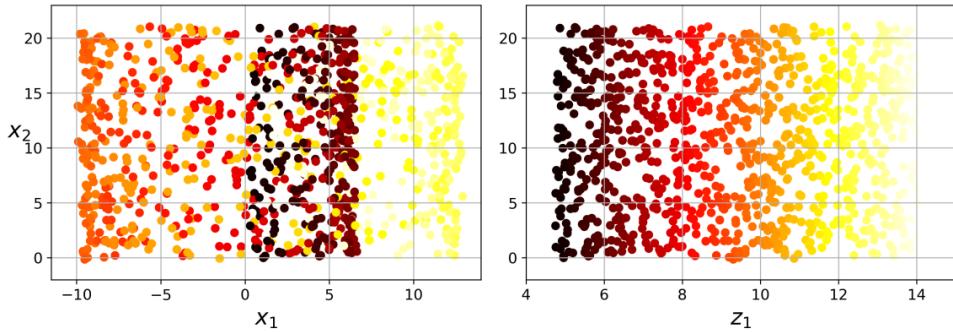
**Figura 1.2:** Reducción dimensional lineal. Los datos en 3D se proyectan a un subespacio de menor dimensión (2D), que captura la mayor cantidad de varianza posible en el conjunto original. Imagen extraída y editada de [6].



**Figura 1.3:** Ejemplo de un conjunto de datos en forma de *Swiss roll*. Si bien el conjunto pertenece a  $\mathbb{R}^3$ , la estructura subyacente es de dimensión 2. Imagen extraída y editada de [6].

Sin embargo, la proyección no es siempre el enfoque ideal para la reducción de dimensionalidad (RD). Por ejemplo, el subespacio reducido puede tener una geometría más compleja, como el famoso *Swiss roll*, representado en la Fig. 1.3. La proyección de este conjunto de datos sobre cualquier plano implicaría aplastarlo, perdiendo la estructura local inherente de los datos (Fig. 1.4, izquierda). En cambio, lo que se busca es desenrollar el *Swiss roll*, como ilustra la imagen en la derecha de la Fig. 1.4. Esto es lo que los algoritmos de RD no-lineal logran, en contraste con los lineales.

La RD no-lineal permite transformar datos de alta dimensionalidad a un subespacio de dimensión inferior, aplicando transformaciones no-lineales. Estos métodos se asocian comúnmente con técnicas de aprendizaje automático, en particular con el aprendizaje de variedades (del inglés, *Manifold Learning*). Los algoritmos estándar de RD no-lineal que se aplican en este trabajo son: *Kernel PCA* (KPCA), *Locally Linear Embedding*



**Figura 1.4:** Proyección lineal (izquierda) vs. *unroll* no-lineal (derecha) de un *Swiss roll*. Imagen extraída y editada de [6].

(LLE) e *Isometric Mapping* (Isomap); y del aprendizaje profundo (*Deep Learning*), los *autoencoders* (AE). El KPCA es un método basado en técnicas de *kernel* [7], que puede asignar datos a un espacio de características de alta dimensionalidad para descubrir mejor la estructura no-lineal de los datos. Por su parte, el LLE está basado en una aproximación lineal local, que puede preservar las relaciones locales de los datos en un espacio de baja dimensión. El Isomap se basa en construir un grafo en el conjunto de datos, que puede preservar la estructura topológica global de los datos. Los AE son una arquitectura dentro del campo de redes neuronales que aprende una representación de datos de baja dimensión a través de un proceso de codificación y decodificación.

### 1.3. Problema de estudio: flujo alrededor de un cilindro

En esta sección se presenta el caso de estudio del presente trabajo, al cual se le aplicarán las distintas técnicas de reducción dimensional: el flujo alrededor de un cilindro (ver Fig. 1.5). Este es un problema clásico en la mecánica de fluidos, cuyas ecuaciones que gobiernan el movimiento son las de Navier-Stokes para flujo incompresible [8]. Esta ecuación representa el equilibrio entre las fuerzas inerciales, de presión y las fuerzas viscosas, cuya expresión matemática es

$$\rho \left( \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} \right) = -\nabla p + \mu \Delta \mathbf{u}, \quad (1.1)$$

$$\nabla \cdot \mathbf{u} = 0, \quad (1.2)$$

donde  $\mathbf{x} = (x, y, z)$  son las coordenadas espaciales, siendo  $x$  la dirección de la corriente (ver Fig. 1.5).  $\mathbf{u} = (u_x, u_y, u_z)$  son los campos de velocidades en las direcciones  $x$ ,  $y$  y

$z$ , respectivamente,  $t$  el tiempo,  $\rho$  es la densidad,  $p$  es la presión y  $\mu$  es el coeficiente de viscosidad. La ecuación de continuidad  $\nabla \cdot \mathbf{u} = 0$  impone la condición de incompresibilidad. Por otro lado, la Ec. 1.1 está sujeta a una condición de contorno de velocidad nula (no deslizamiento) sobre toda la superficie del cilindro.

Para facilitar el análisis, las ecuaciones de Navier-Stokes se adimensionalizan introduciendo las variables adimensionales denotadas con sombrero

$$\hat{\mathbf{x}} = \frac{\mathbf{x}}{D}, \quad \hat{\mathbf{u}} = \frac{\mathbf{u}}{U}, \quad (1.3)$$

$$\hat{p} = \frac{p}{\rho U^2}, \quad \hat{t} = \frac{Ut}{D}, \quad (1.4)$$

donde  $D$  (diámetro del cilindro) es la longitud característica y  $U$  (el módulo de la velocidad del flujo entrante, uniforme) la velocidad característica. Adicionalmente, se define el número adimensional de Reynolds  $Re$ , parámetro que caracteriza el flujo, como

$$Re = \frac{UD}{\nu}, \quad (1.5)$$

donde  $\nu = \frac{\mu}{\delta}$  es la viscosidad cinemática. Entonces, las ecuaciones adimensionales quedan

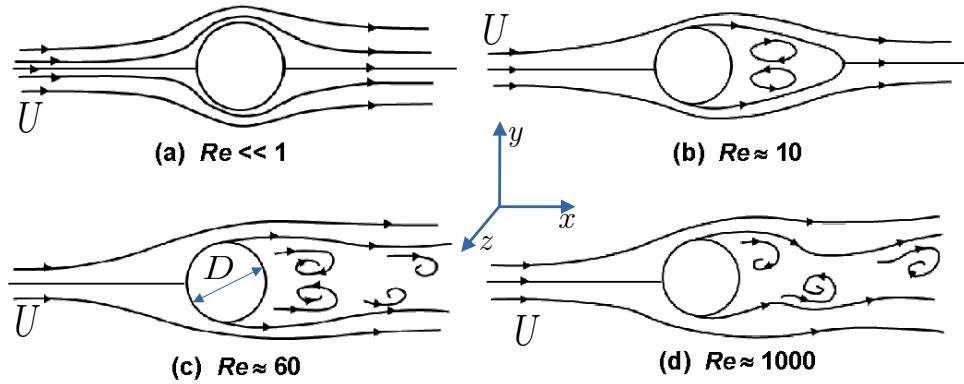
$$\frac{\partial \hat{\mathbf{u}}}{\partial \hat{t}} + (\hat{\mathbf{u}} \cdot \nabla) \hat{\mathbf{u}} = -\nabla \hat{p} + \frac{1}{Re} \nabla^2 \hat{\mathbf{u}}, \quad (1.6)$$

$$\nabla \cdot \hat{\mathbf{u}} = 0. \quad (1.7)$$

Notar que esta ecuación depende únicamente del número de Reynolds. Esto implica que flujos con el mismo número de Reynolds, aunque sean de geometrías o propiedades diferentes, pueden considerarse dinámicamente similares. El número de Reynolds aumenta no solo con la velocidad, sino también con la densidad del fluido, la inversa de la viscosidad y las dimensiones características. Es por tanto una medida adimensional de la velocidad de flujo.

Anteriormente, se indicó que existen fenómenos comunes en los fluidos, como el desprendimiento de vórtices de von Kármán de un frente de nubes atravesando una isla (ver Fig. 1.1). Este fenómeno es característico del flujo alrededor de un cilindro y depende fuertemente del  $Re$  [9]. La Fig. 1.5 muestra los distintos regímenes del flujo alrededor de un cilindro, y se esquematizan los patrones generados para diferentes valores de  $Re$ .

La Fig. 1.5 (a) muestra un flujo con un número de Reynolds muy pequeño. En este caso, las líneas de corriente son simétricas respecto del eje horizontal y se vuelven a pegar detrás del cilindro. A medida que el número de  $Re$  aumenta, la parte trasera del cilindro



**Figura 1.5:** Patrones presentes en el flujo alrededor de un cilindro distintos números de Reynolds. Imagen extraída y editada de [9].

se ensancha. Al acercarse a  $Re = 10$ , se forma una región cerrada de líneas de corriente en la estela del cilindro. El fluido en la mitad superior rota en sentido horario, mientras que el fluido en la mitad inferior rota en sentido antihorario, resultando en la formación de vórtices. La **Fig. 1.5 (b)** muestra este estado.

A medida que el número de  $Re$  aumenta, la longitud del vórtice aumenta. Cuando el  $Re$  supera el valor de 60, el flujo detrás del cilindro se vuelve inestable como se muestra en la **Fig. 1.5 (c)**, y los vórtices comienzan a oscilar en dirección vertical. El vórtice completamente formado es arrastrado por el flujo principal circundante, y posteriormente se genera otro vórtice (al otro lado) desfasado del anterior. Estos vórtices se expanden y eventualmente se disipan aguas abajo. De esta forma, mientras el flujo entrante al cilindro es uniforme, detrás del cilindro se da un fenómeno oscilatorio de desprendimiento de vórtices [10].

Como se muestra en la **Fig. 1.5 (d)**, cuando el número de Reynolds alcanza un valor de aproximadamente 1.000, los vórtices se mezclan entre sí, y el flujo detrás del cilindro (estela) se comporta de manera caótica, dando lugar a un flujo completamente turbulento. Sin embargo, la corriente libre distante de la región de la estela se mantiene en régimen laminar.

## 1.4. Objetivos

El objetivo principal de este trabajo es implementar y comparar diversas técnicas de reducción dimensional no-lineal en el marco de la dinámica de fluidos computacional, utilizando como referencia el enfoque lineal tradicional (PCA). El caso de estudio seleccionado es el flujo alrededor de un cilindro inmerso en un flujo uniforme. Para ello, se plantean los siguientes objetivos:

1. **Generar una base de datos** a la cual se aplicarán los algoritmos, mediante simulaciones numéricas directas (DNS) del flujo alrededor de un cilindro para diferentes números de Reynolds. Para ello se utilizará el *software* de código abierto Xcompact3d.
2. **Realizar el análisis por componentes principales (PCA)** sobre el conjunto de datos obtenido. Debido a su amplia utilización en la RD en el CFD, los resultados de este método proporcionan una referencia con la cual contrastar el rendimiento de los métodos no-lineales.
3. **Implementación de las técnicas de RD no-lineal** KPCA, LLE, Isomap y *autoencoders* sobre la base de datos. Realizar un análisis paramétrico para KPCA, LLE e Isomap.
4. **Comparar los resultados** de todos los métodos, tanto cuantitativa como cualitativamente. En particular, contrastar los resultados de los métodos no-lineales con el PCA.



# Capítulo 2

## Simulación numérica directa con Xcompact3d

*“LUCHO POR LA LUZ...”*

— Príncipe Arthas

El presente estudio consiste en aplicar técnicas de reducción dimensional a un problema fluidodinámico de interés: el flujo alrededor de un cilindro. Con el fin de obtener una base de datos para aplicar los distintos métodos de reducción dimensional, se optó por realizar simulaciones numéricas directas (DNS) con el *software* Xcompact3d. En este capítulo se describen las características principales del programa, el cual resuelve las ecuaciones de Navier-Stokes en un dominio tipo caja en una grilla cartesiana. Se valida el código simulando dos flujos canónicos bien documentados, comparando los resultados con datos de referencia. Por último, se llevan a cabo las simulaciones del flujo alrededor de un cilindro a distintos números de Reynolds, las cuales constituyen la base de datos para los distintos algoritmos de reducción dimensional.

### 2.1. Xcompact3d

Para el presente trabajo, se generó una base de datos mediante simulaciones numéricas directas (DNS) del flujo alrededor de un cilindro. Para ello, se empleó el *software* de código abierto Xcompact3d, basado en Fortran 90-95, dedicado a la resolución de las ecuaciones de Navier-Stokes (N-S) [11]. La herramienta numérica puede resolver las ecuaciones de gobierno del flujo desde dos enfoques: DNS y LES. La primera resuelve todas las escalas del flujo turbulento, capturando tanto las estructuras grandes como las pequeñas, mientras que la segunda se enfoca en las escalas turbulentas más grandes [12]. Las características que lo hacen una herramienta de gran valor para la comunidad de

CFD son su simplicidad, versatilidad, precisión, escalabilidad, portabilidad y eficiencia.

Originalmente fue diseñado en Francia a mediados de los años 90' para procesadores en serie y posteriormente fue adaptado a sistemas de alto rendimiento (HPC). Ahora se puede utilizar eficientemente en cientos de miles de núcleos de CPU para investigar problemas de turbulencia y transferencia de calor gracias a la biblioteca de código abierto 2DECOMP&FFT: un marco para la descomposición del dominio en lápices 2D basado en Fortran, para soportar la construcción de aplicaciones paralelas a gran escala en sistemas de memoria distribuida utilizando MPI [13]. Adicionalmente, cuenta con una interfaz de transformadas rápidas de Fourier, utilizada en la resolución de la ecuación de Poisson en la simulación de flujos incompresibles.

Actualmente, Xcompact3d está siendo utilizado por numerosos grupos de investigación en todo el mundo para estudiar corrientes de gravedad, turbulencia en el problema de capa límite, flujos de estela y chorro, parques eólicos y soluciones de control activo de flujo para mitigar la turbulencia. Este código ha dado lugar a más de un centenar de [publicaciones científicas](#), destacando su relevancia en el ámbito de la mecánica de fluidos [11, 14, 15].

### 2.1.1. Descripción del *software* y métodos numéricos

Los flujos turbulentos descritos en Xcompact3d están gobernados por las ecuaciones de Navier-Stokes, dadas para flujos de bajo número de Mach (LMN) en su notación indicial ( $i : x, y, z$ ) por

$$\frac{D\rho}{Dt} = -\rho \frac{\partial u_i}{\partial x_i}, \quad (1)$$

$$\frac{\partial p^{(0)}}{\partial x_j} = 0, \quad (2)$$

$$\rho \frac{DT}{Dt} = \frac{1}{Re \cdot Pr} \frac{\partial}{\partial x_j} \left( \kappa \frac{\partial T}{\partial x_j} \right), \quad (3)$$

$$\rho \frac{Du_i}{Dt} = -\frac{\partial p^{(1)}}{\partial x_i} + \frac{1}{Re} \frac{\partial \tau_{ij}}{\partial x_j} + \rho g_i, \quad (4)$$

$$p^{(0)} = \rho T, \quad (5)$$

donde  $u_i$  es la velocidad en la dirección  $i$ ,  $p$  la presión,  $\rho$  la densidad,  $T$  la temperatura,  $\tau_{ij}$  el tensor de esfuerzos viscosos,  $\kappa$  el coeficiente de difusión térmica,  $g_i$  la aceleración de la gravedad en la dirección  $i$  y  $\frac{D(\cdot)}{Dt}$  es la derivada material. El número de Reynolds ( $Re$ ) es un número adimensional definido como la relación entre fuerzas iniciales y viscosas, que determina el rango de escalas turbulentas a simular. Similarmente, el número de Prandtl

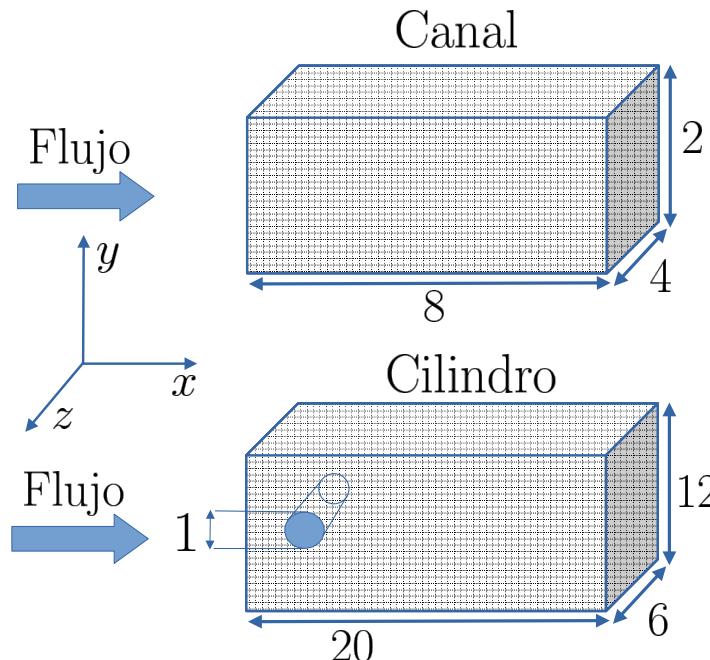
$(Pr)$  es un número adimensional definido como la relación entre la difusión de momento y la difusión térmica. Bajo la aproximación LMN, el campo de presión puede descomponerse en las presiones termodinámica ( $p^{(0)}$ ) y mecánica ( $p^{(1)}$ ). La primera es constante en el espacio, lo que permite filtrar las ondas de presión acústica de la solución, mientras que la presión mecánica se comporta como el campo de presión de un flujo incompresible. Esta formulación permite a Xcompact3d abordar flujos con densidad variable en el régimen  $M \lesssim 0,3$ , donde  $M$  es el número de Mach. En el caso de densidad constante, se reduce a las ecuaciones gobernantes para flujos incompresibles, las cuales son de interés en este trabajo. La expresión matemática correspondiente es la siguiente

$$\frac{\partial \hat{\mathbf{u}}}{\partial \hat{t}} + (\hat{\mathbf{u}} \cdot \nabla) \hat{\mathbf{u}} = -\nabla \hat{p} + \frac{1}{Re} \nabla^2 \hat{\mathbf{u}}, \quad (2.1)$$

$$\nabla \cdot \hat{\mathbf{u}} = 0. \quad (2.2)$$

Las ecuaciones se resuelven con esquemas compactos de diferencias finitas de sexto orden [16] en una malla cartesiana y discretización temporal explícita (Adams–Bashforth de 2do orden, Runge Kutta de 3er y 4to orden), con una estrategia convencional de paso fraccionado [17].

### 2.1.2. Validación DNS 3D



**Figura 2.1:** Esquema espacial de los problemas simulados: flujo turbulento en canal de placas paralelas (Canal) y flujo alrededor de un cilindro (Cilindro).

Para demostrar las capacidades de Xcompact3d, en esta sección se valida el código a partir de la solución de dos problemas canónicos: el flujo turbulento en canal de placas paralelas [12, 18] y el flujo alrededor de un cilindro [8, 19]. Ambos casos han sido estudiados ampliamente de manera teórica, experimental y numéricamente.

Se realizan simulaciones DNS de un flujo turbulento en canal de placas paralelas a  $Re = 4200$  (correspondiente a un  $Re_\tau = 180$ , definido más adelante) y del flujo alrededor de un cilindro a  $Re = 300$ <sup>1</sup> (ver Fig. 2.1). Los parámetros numéricos para estas simulaciones se hallan en la Tabla 2.1. El primer caso se simula con una sola grilla y el segundo con tres discretizaciones distintas: Gruesa, Intermedia y Fina. En ambos casos, se calculan los perfiles de velocidad media y de las fluctuaciones de la velocidad, ambos en la dirección de la corriente. Los mismos son contrastados con datos de referencia [20–22] en las Figs. 2.2 (canal) y 2.3 (cilindro). En el caso del canal, se emplean las variables adimensionalizadas de la velocidad ( $u^+$ ) y de la distancia a la pared ( $y^+$ ), cuya definición y cantidades asociadas se definen en la Tabla 2.2. Para el cilindro, las distintas curvas representan los perfiles en diferentes posiciones axiales aguas abajo del cilindro, desplazados entre sí para facilitar la visualización. Es decir, para cada curva, el valor de  $\bar{u}$  en los extremos laterales del gráfico ( $y = \pm 2$ ) es el mismo, igual a la velocidad del flujo externo  $U_0$ , y para las fluctuaciones de la velocidad 0.

Parámetros	Canal	Cilindro
Grilla (nx, ny, nz)	(128, 65, 64)	Gruesa (191, 64, 32) Intermedia (257, 128, 32) Fina (257, 160, 64)
Dominio (Lx, Ly, Lz)	(8, 2, 4)	(20, 12, 6)
Geometría	Caja rectangular	$r = 0,5$ $c = (5, 6)$
$\Delta t$	0,005	0,001
$Re$	4200	300
Condiciones de contorno	Sin deslizamiento (y1, yn) Periódicas (x, z)	Entrada-Salida (x1, xn), Periódicas (y, z)
Esquema temporal	RK3	AB3
Iteraciones	100,000	60,000

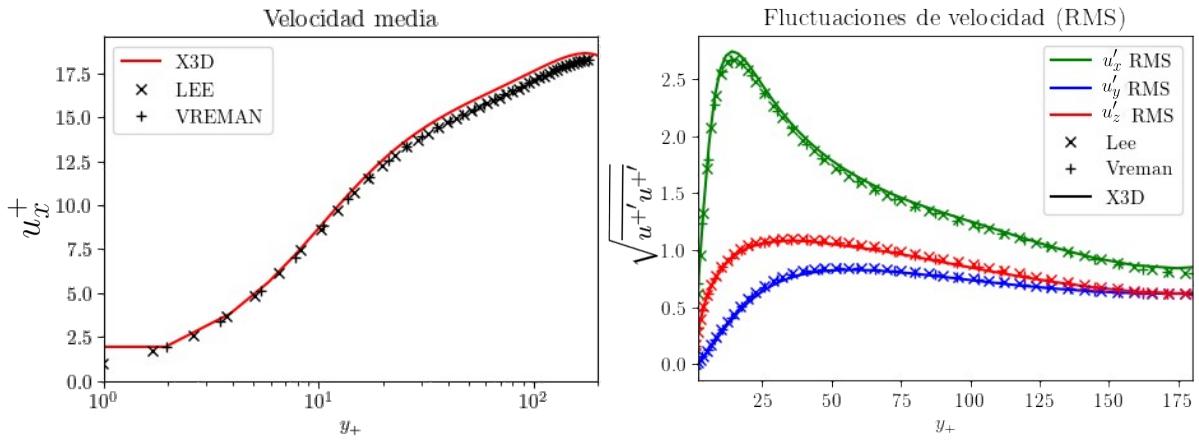
**Tabla 2.1:** Parámetros de entrada utilizados en las simulaciones de del flujo turbulento en un canal de placas paralelas (Canal) y alrededor de un cilindro (Cilindro). Los  $n_i$  son el número de nodos en la dirección  $i$ ,  $L_i$  es la longitud del dominio en la dirección  $i$ ,  $\Delta t$  es el paso de tiempo,  $Re$  es el número de Reynolds,  $r$  es el radio del cilindro,  $c$  es el centro del cilindro, y  $x1, xn, y1, yn$  son las posiciones de las fronteras.

La simulación del flujo en un canal turbulento logra una buena concordancia con la

<sup>1</sup>Basado en el diámetro  $D$  del cilindro y la velocidad de la corriente libre:  $Re = \frac{U_0 D}{\nu}$ .

Cantidad	Expresión
Tensión de corte en la pared	$\tau_w = \mu \frac{du}{dy} \Big _{y=0}$
Velocidad de fricción	$u_\tau = \sqrt{\frac{\tau_w}{\rho}}$
Reynolds de fricción	$Re_\tau = \frac{u_\tau H/2}{\nu}$
Distancia adimensional a la pared	$y^+ = \frac{yu_\tau}{\nu}$
Velocidad adimensional	$u_i^+ = \frac{u_i}{u_\tau}$
Fluctuación de la velocidad adimensional en RMS	$\sqrt{u_i'^+ u_i'^+}$

**Tabla 2.2:** Expresiones de las cantidades adimensionalizadas utilizadas en la validación del flujo turbulento en un canal de placas paralelas.  $u'_x$ ,  $u'_y$ ,  $u'_z$  son las fluctuaciones de la velocidad en las direcciones  $x$ ,  $y$  y  $z$ , respectivamente.

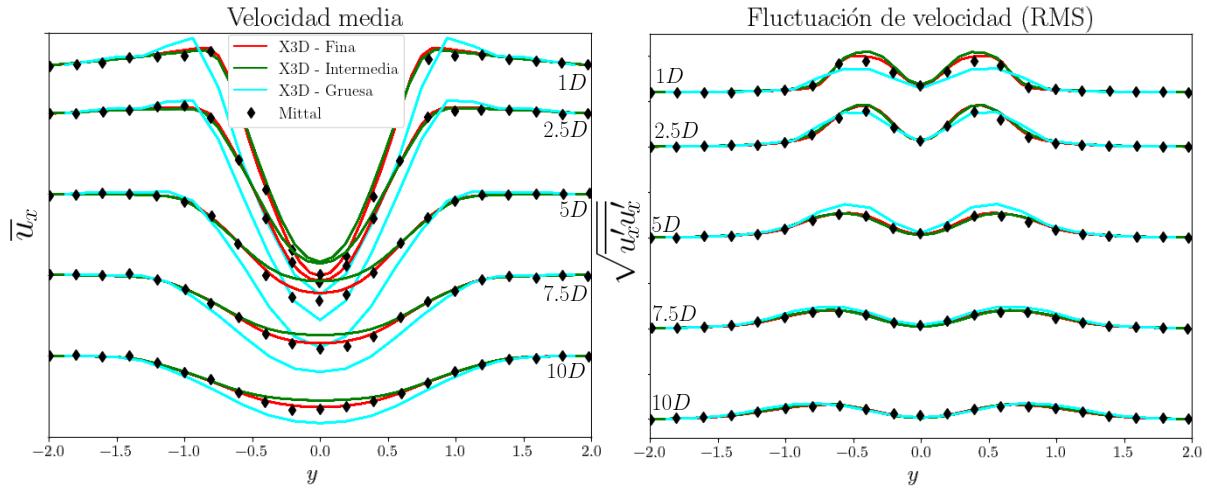


**Figura 2.2:** Estadísticas de la velocidad en la dirección de la corriente del flujo turbulento en un canal periódico de  $Re = 4200$  de las simulaciones DNS 3D con Xcompact3d (línea sólida) y de los datos de referencia de [20] ( $\times$ ) y [21] ( $+$ ).

referencia con una malla relativamente modesta, y se obtiene un  $Re_\tau = 174,5$ . Respecto al flujo alrededor del cilindro, la malla Gruesa da resultados desviados respecto de la referencia, particularmente alrededor de la coordenada vertical del cilindro. Por otro lado, con las mallas Intermedia y Fina se observa un mejor acuerdo. Específicamente, la malla Fina (257, 160, 64) presenta resultados muy similares a la referencia. Por lo tanto, se concluye que Xcompact3d es un código robusto y preciso para la simulación de flujos inestables, en particular para el problema pertinente a este trabajo.

### 2.1.3. Validación DNS 2D

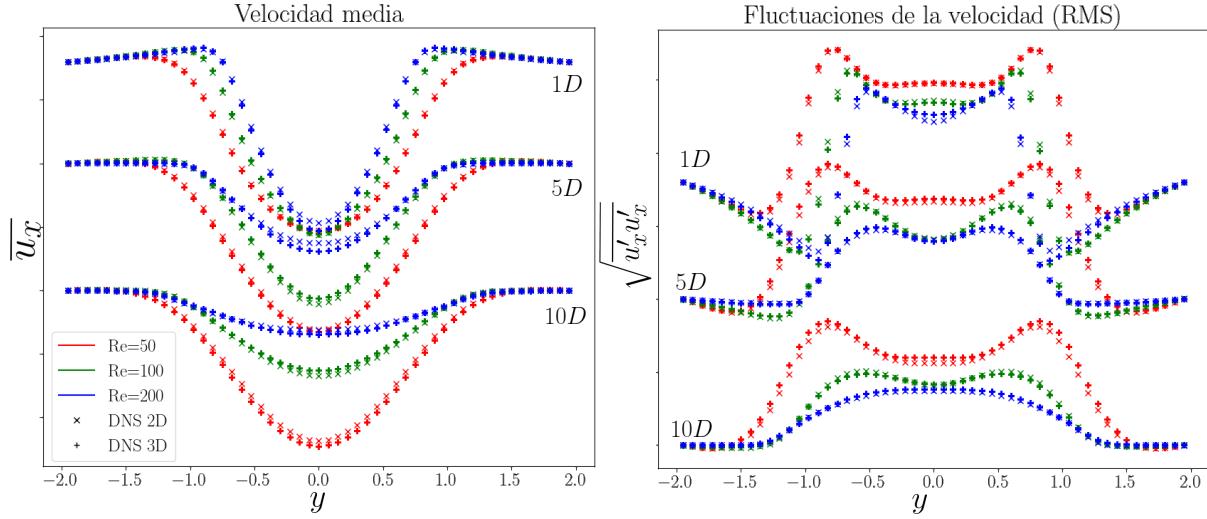
Hasta aquí, se validó el código Xcompact3d para las simulaciones DNS de las ecuaciones de N-S en 3D. Sin embargo, en este estudio se optó por realizar simulaciones 2D del flujo alrededor de un cilindro. Por un lado, se reduce el tamaño de los datos en un factor  $n_z$  (número de nodos en la dirección del ancho del canal), lo cual además de reducir el



**Figura 2.3:** Estadísticas de la velocidad en la dirección de la corriente del flujo alrededor de un cilindro a  $Re = 300$  de las simulaciones DNS 3D Xcompact3d (línea sólida) y de los datos de referencia de [22] ( $\diamond$ ). Se comparan resultados para tres mallas distintas. Las diferentes curvas corresponden a los perfiles a  $1D$ ,  $2.5D$ ,  $5D$ ,  $7.5D$  y  $10D$  diámetros detrás del cilindro.

costo computacional, favorece el rendimiento de los algoritmos de RD implementados, puesto que estos presentan limitaciones a mayor dimensionalidad (la cual ya de por sí es grande). Por otro lado, facilita el manejo de datos, ya que las instantáneas se disponen naturalmente en matrices 2D y la vectorización es simple.

Durante el desarrollo del trabajo, el código de Xcompact3d fue adaptado para simular el flujo alrededor de un cilindro en su enfoque bidimensional. El mismo se halla actualmente en una etapa de prueba, por lo que es de interés determinar la validez del código. De esta manera, con el fin de verificar el correcto funcionamiento del código 2D se contrastaron los resultados de simulaciones de flujo alrededor de un cilindro obtenidos mediante DNS 2D con aquellos obtenidos mediante DNS 3D (código previamente validado). Las simulaciones se llevaron a cabo en un rango de números de Reynolds de 50 a 200. En las simulaciones 3D, los parámetros son idénticos a los utilizados en la validación anterior, con la excepción del valor de  $Re$ , que se ajustó a los valores mencionados, y se ejecutaron 300.000 pasos de tiempo. En el caso 2D, los parámetros son los mismos que para el 3D, excepto por la malla (257, 160, 1) y el dominio (20, 12, 1). En la Fig. 2.4 se grafican los perfiles adimensionales de la velocidad media y de las fluctuaciones de la velocidad obtenidos a partir de las simulaciones DNS 2D y 3D. Se observa un buen acuerdo entre los resultados del código 2D y los obtenidos con el código 3D, lo que permitió verificar el correcto funcionamiento del código 2D.



**Figura 2.4:** Comparación de las estadísticas de la velocidad del flujo alrededor de un cilindro para  $Re = 50, 100$  y  $200$ , obtenidos por DNS 2D ( $\times$ ) y 3D (+) en Xcompact3d. Los resultados se grafican a distancias axiales  $1D$ ,  $5D$  y  $10D$  detrás del cilindro, donde  $D$  es el diámetro del cilindro.

## 2.2. Base de datos: DNS del flujo alrededor de un cilindro

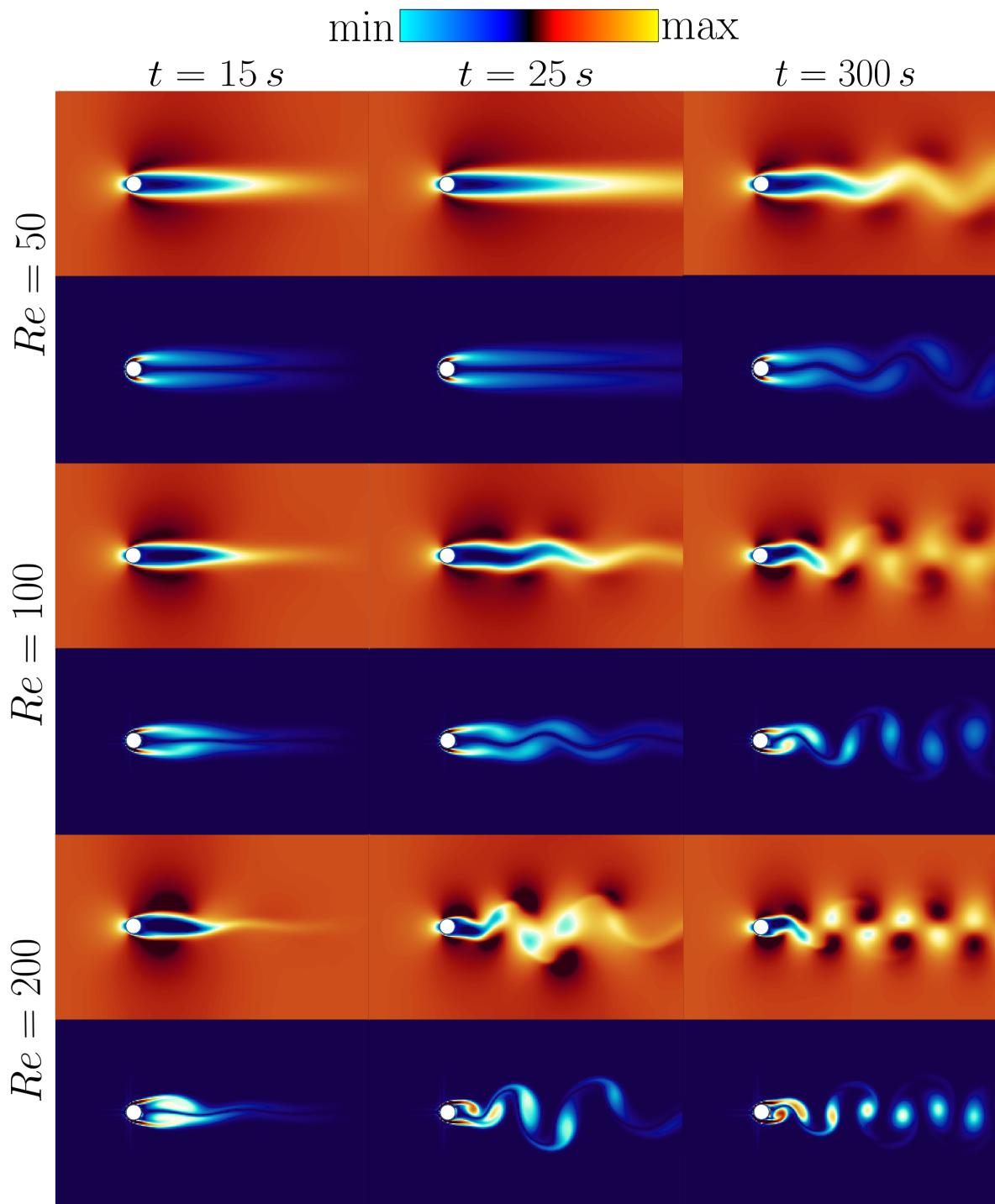
En esta sección, se describe el proceso de obtención de la base de datos utilizada en el presente trabajo, la cual consiste en simulaciones DNS 2D del flujo alrededor de un cilindro en un rango de números de Reynolds entre  $50$  y  $200$ . Estas simulaciones ofrecen una representación detallada y precisa del fenómeno en cuestión, abarcando un rango en  $Re$  que permite apreciar desde la suave oscilación del flujo a valores bajos, hasta el notable desprendimiento de vórtices en valores más altos. Los parámetros de entrada son los mismos de la Tab. 2.1 para malla fina, con la salvedad que se simularon  $500,000$  pasos de tiempo para asegurar el régimen estacionario en todo el rango de  $Re$ <sup>2</sup>.

En la Fig. 2.5 se visualizan las instantáneas de la componente de la velocidad en la dirección de desarrollo hidrodinámico  $u_x$ , y el módulo de la vorticidad  $\omega = \left| \frac{du_y}{dx} - \frac{du_x}{dy} \right|$  a distintos tiempos. Se observa claramente el desprendimiento de vórtices en la estela del cilindro, los cuales se desplazan en la dirección del flujo ( $x$ ). A medida que aumenta el número de Reynolds, el desprendimiento ocurre más temprano y los vórtices son más intensos. En las simulaciones realizadas, se alcanzó el régimen estadísticamente estacionario para  $t > 300$ , incluso para  $Re = 50$ .

De esta manera, la base de datos se conforma por  $2.000$  instantáneas tomadas cada  $100$  pasos de tiempo ( $\Delta t = 0,1$ ) a partir del paso de tiempo  $300.000$ . Es decir, se tienen

<sup>2</sup>Para Reynolds bajos, el tiempo de establecimiento del comportamiento oscilatorio estadísticamente estacionario es notablemente mayor respecto a valores más altos.

las instantáneas para  $300 < t < 500$ .



**Figura 2.5:** Instantáneas en el plano x-y de la componente horizontal de la velocidad  $u_x$  (naranja), y el módulo de la vorticidad (azul) obtenidas por DNS 2D del flujo alrededor de un cilindro en Xcompact3d para distintos Reynolds.



# Capítulo 3

## Reducción dimensional

*“Hay que purgar toda la ciudad...”*

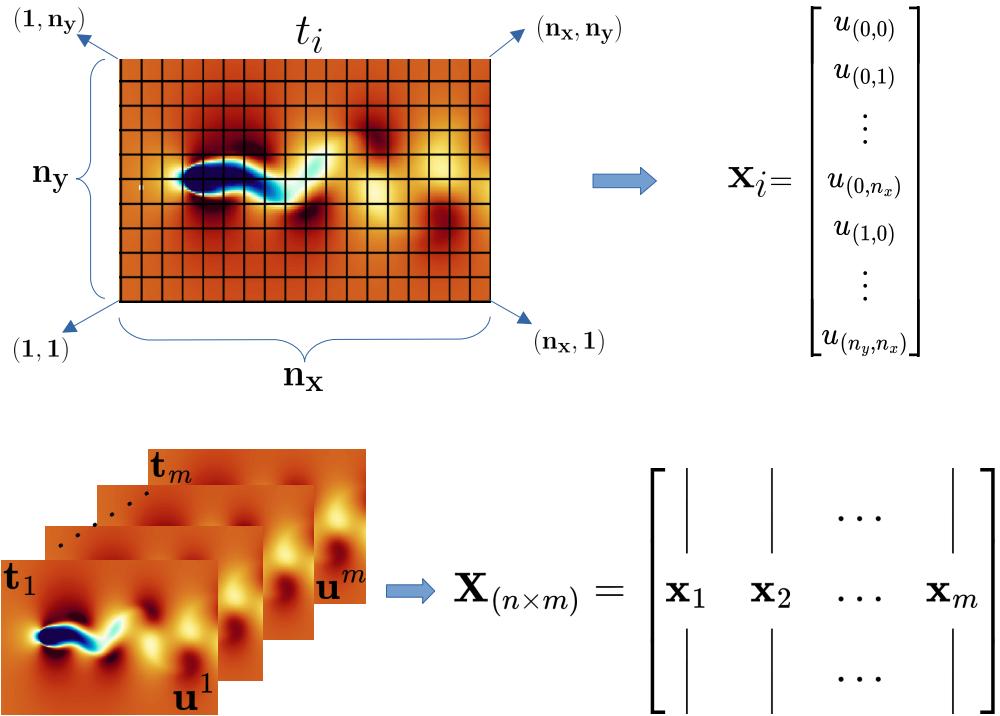
— Príncipe Arthas

En este capítulo se describen los métodos de reducción dimensional empleados. Se comienza por el Análisis de Componentes Principales (PCA) que proporciona una aproximación lineal óptima, basado en la Descomposición por Valores Singulares (SVD). Luego se abordan los métodos de reducción no-lineal: *Kernel PCA* (KPCA), *Locally Linear Embedding* (LLE) e *Isometric Mapping* (Isomap). Por último, se presenta una técnica de aprendizaje profundo para la reducción de dimensionalidad conocida como *autoencoders*. Antes de comenzar, se detalla la notación y arreglo de datos adoptada a lo largo del desarrollo del presente trabajo.

### 3.1. Notación y arreglo de datos

En muchos campos de la ciencia y la ingeniería, los sistemas complejos generan datos que naturalmente se disponen en matrices, o más general, en arreglos. Por ejemplo, en experimentos o simulaciones de series temporales, los datos pueden representarse en una matriz donde cada columna contiene mediciones de determinadas variables en un instante dado. Si las mediciones en cada instante de tiempo son multidimensionales, es posible reorganizar o transformar estos datos en un vector columna que forme parte de una matriz más amplia.

En el contexto de este trabajo, los datos son extraídos de simulaciones DNS en una grilla de alta resolución espacial ( $n_x, n_y$ ), donde  $n_x$  y  $n_y$  son los números de nodos en las direcciones horizontal y vertical respectivamente de la malla, en  $m$  pasos temporales. La metodología usual para el tratamiento de este tipo de datos consta de transformar la matriz de la instantánea  $\mathbf{x} \in \mathbb{R}^{n_x \times n_y}$  a un vector  $\mathbf{x} \in \mathbb{R}^n$ , donde  $n = n_x \times n_y$  es



**Figura 3.1:** Esquema del arreglo de datos a partir de las simulaciones DNS, dispuestos en una matriz  $\mathbf{X} \in \mathbb{R}^{n \times m}$ , con  $n = n_x \times n_y$ .

el número total de puntos en la malla. Luego, el conjunto de datos entero puede ser representado como una matriz de  $\mathbf{X} \in \mathbb{R}^{n \times m}$ , donde las columnas  $\mathbf{x}_i \in \mathbb{R}^n$  corresponden a las instantáneas vectorizadas del módulo de la velocidad  $u$  en el paso de tiempo  $i$ . De esta forma, la matriz de velocidad  $\mathbf{X}$  puede escribirse como

$$\mathbf{X}_{(n \times m)} = \begin{bmatrix} | & | & \dots & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_m \\ | & | & \dots & | \end{bmatrix} = \begin{bmatrix} | & \mathbf{\chi}_1 & | \\ | & \mathbf{\chi}_2 & | \\ \vdots & \vdots & \vdots \\ | & \mathbf{\chi}_n & | \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1m} \\ u_{21} & u_{22} & \dots & u_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ u_{n1} & u_{n2} & \dots & u_{nm} \end{bmatrix}, \quad (3.1)$$

donde,  $n$  (total de puntos espaciales de la malla) es la dimensión espacial, y  $m$  (número de instantáneas temporales) la dimensión temporal. También es posible pensar en términos de filas de  $\mathbf{X}$ , donde cada fila  $\mathbf{\chi} \in \mathbb{R}^m$  corresponde a la evolución temporal del módulo de la velocidad  $u$  en un punto de la malla. En la Fig. 3.1 se representa el esquema del arreglo de los datos a partir de las simulaciones de CFD.

### 3.1.1. Reducción espacial y temporal

En las próximas secciones se presentan 5 algoritmos de RD. El objetivo es construir un espacio latente de baja dimensionalidad  $\mathbf{Y}$  que capture las características más dominantes del flujo. Esto puede abordarse desde dos enfoques: espacial y temporal. La forma convencional de abordar la RD espacial de un flujo es tomar las columnas de  $\mathbf{X} \in \mathbb{R}^{n \times m}$ , correspondientes a instantáneas de velocidad  $\mathbf{x}_i \in \mathbb{R}^n$  como muestras individuales. Luego, la reducción de dimensionalidad dará lugar a  $m$  muestras de vectores  $\mathbf{y}_i \in \mathbb{R}^r$  en el espacio latente, donde  $r < n$ . Esta es la forma establecida para crear ROMs, modelando la dinámica del sistema en la representación de baja dimensionalidad, y luego mapeando de nuevo al espacio original. Sin embargo, esto no proporciona modos espaciales interpretables, ya que el espacio latente es inherentemente más pequeño que el dominio físico.

Alternativamente, se pueden tomar las filas de  $\mathbf{X}$  como muestras  $\mathbf{x}_i \in \mathbb{R}^m$ . Esto lleva a una reducción de dimensionalidad temporal, en la que se obtienen modos espaciales visualizables. Este enfoque es especialmente útil cuando el objetivo es entender las estructuras subyacentes del flujo, lo cual se encuentra fuera del alcance del trabajo. Además, permite una comparación directa entre diferentes métodos de RD y facilita la interpretación física de los modos espaciales obtenidos.

Cabe destacar que usualmente  $m \ll n$ , lo que resulta en la descomposición en autovectores de matrices  $(n \times n)$  en lugar de  $(m \times m)$  como se verá más adelante, resultando en requisitos computacionales significativamente mayores para el caso temporal respecto del espacial. Por lo tanto, en la práctica se suele preferir trabajar con la reducción espacial como primer paso hacia el modelado de ROMs. En este estudio, se consideran y comparan ambos enfoques. La reducción espacial puede conducir a representaciones de baja dimensionalidad potentes adecuadas para ROMs, mientras que modos físicamente interpretables y estructuras coherentes pueden encontrarse a través de la reducción temporal.

## 3.2. Descomposición por valores singulares (SVD)

El algoritmo fundamental detrás del análisis por componentes principales (PCA) es la Descomposición por Valores Singulares (SVD), considerado como uno de los algoritmos de factorización de matrices más importante en la era computacional [23]. La SVD es una generalización de la descomposición de autovectores y autovalores para matrices no cuadradas, y provee una forma sistemática de aproximar datos de alta dimensionalidad en términos de patrones dominantes. La SVD es numéricamente estable y proporciona una representación jerárquica de los datos en términos de un nuevo sistema de coordenadas definido por las correlaciones dominantes en ellos. Además, está garantizada de existir

para cualquier matriz, a diferencia de la descomposición en valores propios.

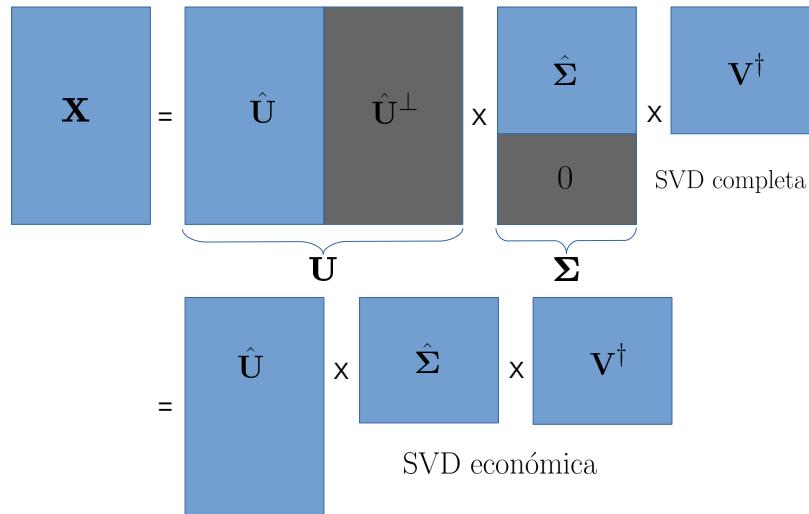
La SVD de una matriz  $\mathbf{X} \in \mathbb{C}^{n \times m}$  es una factorización de la forma

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^\dagger, \quad (3.2)$$

donde  $\mathbf{U} \in \mathbb{C}^{n \times n}$  y  $\mathbf{V} \in \mathbb{C}^{m \times m}$  son matrices unitarias<sup>1</sup>.  $\Sigma \in \mathbb{R}^{n \times m}$  es la matriz que contiene los valores singulares  $\sigma_i$  en su diagonal principal, ordenados de mayor a menor, y ceros fuera de la diagonal. Las columnas de  $\mathbf{U}$  y  $\mathbf{V}$  se denominan vectores singulares izquierdos y derechos, y forman bases ortonormales para los espacios de columnas y filas de  $\mathbf{X}$ , respectivamente.

Cuando  $n \gg m$ , la matriz  $\Sigma$  que es diagonal, tiene hasta  $m$  valores no nulos. Por lo tanto, es posible reconstruir exactamente  $\mathbf{X}$  usando la SVD económica (representado gráficamente en la Fig. 3.2.), dada por

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^\dagger = \begin{bmatrix} \hat{\mathbf{U}} & \hat{\mathbf{U}}^\perp \end{bmatrix} \begin{bmatrix} \hat{\Sigma} \\ \mathbf{0} \end{bmatrix} \mathbf{V}^\dagger = \hat{\mathbf{U}}\hat{\Sigma}\mathbf{V}^\dagger, \quad (3.3)$$



**Figura 3.2:** Esquema de la SVD completa y económica, donde  $\mathbf{X}$  es la reconstrucción a partir de los  $m$  valores singulares no nulos.  $\hat{\mathbf{U}}^\perp$  es el complemento ortogonal de  $\hat{\mathbf{U}}$ .

Adicionalmente, puede realizarse una aproximación de menor rango truncando la SVD en los  $r < m$  valores singulares más grandes

$$\mathbf{X}_r = \mathbf{U}_r\Sigma_r\mathbf{V}_r^\dagger, \quad (3.4)$$

donde  $\mathbf{U}_r \in \mathbb{R}^{n \times r}$ ,  $\Sigma_r \in \mathbb{R}^{r \times r}$  y  $\mathbf{V}_r \in \mathbb{R}^{m \times r}$  son las matrices truncadas de la SVD. La

---

<sup>1</sup>Una matriz cuadrada  $\mathbf{U}$  es unitaria si  $\mathbf{U}\mathbf{U}^\dagger = \mathbf{I}$ , donde  $\mathbf{U}^\dagger$  es la matriz transpuesta conjugada.

aproximación  $\mathbf{X}_r$  es la mejor aproximación de rango  $r$  de  $\mathbf{X}$  en términos de la norma de Frobenius (ver Ec. 4.1).

### 3.2.1. Interpretación de la SVD

La interpretación de la SVD se puede entender mejor en términos de descomposición en patrones espaciales y temporales. Las columnas de la matriz  $\mathbf{U}$  forman una base ortonormal para el espacio de columnas de  $\mathbf{X}$ , y las columnas de  $\mathbf{V}$  para el espacio de filas de  $\mathbf{X}$ . Por lo tanto, si las columnas de  $\mathbf{X}$  son mediciones espaciales en el tiempo, entonces  $\mathbf{U}$  codifica patrones espaciales, y similarmente,  $\mathbf{V}$  patrones temporales. Respecto a las simulaciones DNS, las columnas de  $\mathbf{U} \in \mathbb{R}^{n \times n}$  representan los modos espaciales del flujo y las columnas  $\mathbf{V} \in \mathbb{R}^{m \times m}$  las series temporales de cada modo. En la figura 3.3 se visualiza una columna  $\mathbf{u}$  y su serie temporal  $\mathbf{v}$  a partir de la simulación  $\mathbf{X}$  del flujo alrededor de un cilindro. Por lo tanto, la evolución completa del flujo  $\mathbf{X}$  se puede aproximar por sus  $r$  modos más significativos como

$$\mathbf{X}_r = \sum_{j=1}^r \sigma_j \mathbf{u}_j \mathbf{v}_j^\top. \quad (3.5)$$

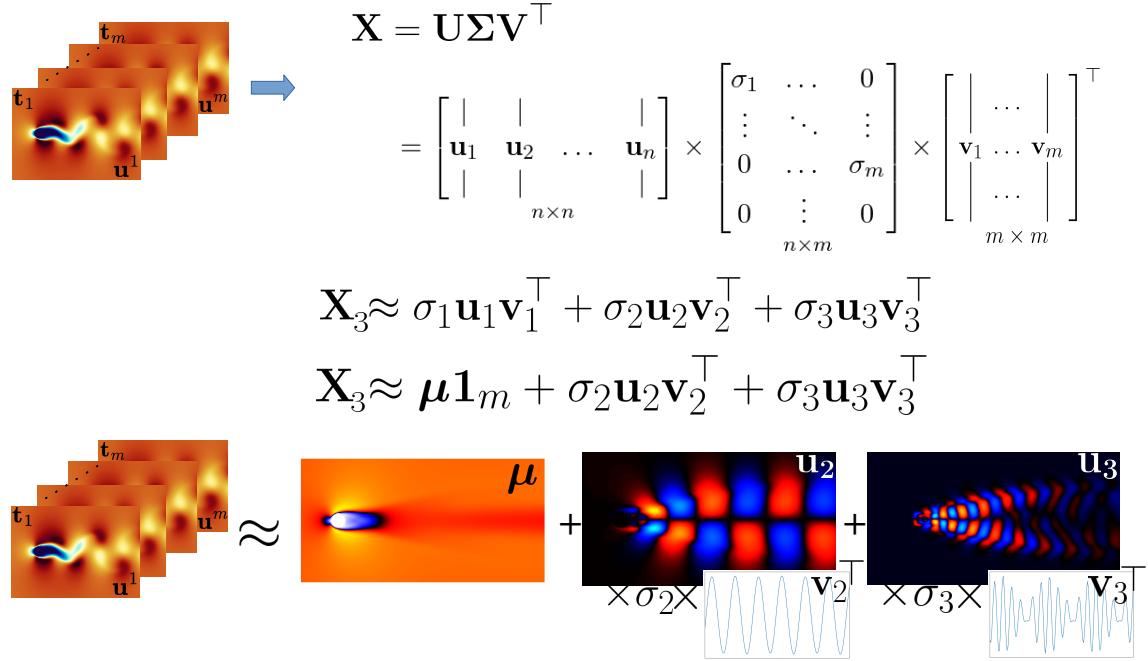
En la figura 3.3 se muestra un ejemplo en el que se aproxima  $\mathbf{X}$  con los primeros tres valores singulares. En este caso, el primer modo  $\sigma_1 \mathbf{u}_1 \mathbf{v}_1^\top$  corresponde simplemente al flujo promedio  $\boldsymbol{\mu}$ . En cierto sentido, es una generalización *data-driven* de la transformada de Fourier, ya que extrae patrones espaciales y temporales únicamente a partir de los datos, sin conocer la función que describe el comportamiento [23].

Adicionalmente, puede ser necesario determinar cuántos componentes son necesarios para capturar un cierto porcentaje de la información contenida en  $\mathbf{X}$ . Esto se puede evaluar a partir de la curva de energía acumulativa, definida como

$$\frac{\sum_{i=1}^r \sigma_i^2}{\sum_{i=1}^m \sigma_i^2}, \quad (3.6)$$

donde  $s_i$  son los valores singulares de  $\Sigma$ . La misma representa la proporción de la varianza total capturada por los primeros  $r$  modos singulares, y permite determinar cuántos de estos modos son necesarios para una reconstrucción adecuada de  $\mathbf{X}$ .

Todas estas consideraciones están profundamente vinculadas a la premisa del análisis por componentes principales. En la siguiente sección se refuerza la intuición de la SVD, donde se demuestra que  $\mathbf{V}$  y  $\mathbf{U}$  son precisamente los autovectores de las matrices de correlación temporal y espacial respectivamente, cuyas columnas contienen los componentes principales.



**Figura 3.3:** Interpretación de la SVD en el contexto de las simulaciones DNS. Las columnas  $\mathbf{u}$  representan modos espaciales y las columnas  $\mathbf{v}$  su evolución en el tiempo. En la imagen se representa la aproximación de rango 3. Notar que  $\mu\mathbf{1}_m = \sigma_1\mathbf{u}_1\mathbf{v}_1^\top$ , donde  $\mu$  es el flujo promedio y  $\mathbf{1}_m \in \mathbb{R}^m$  un vector de unos.

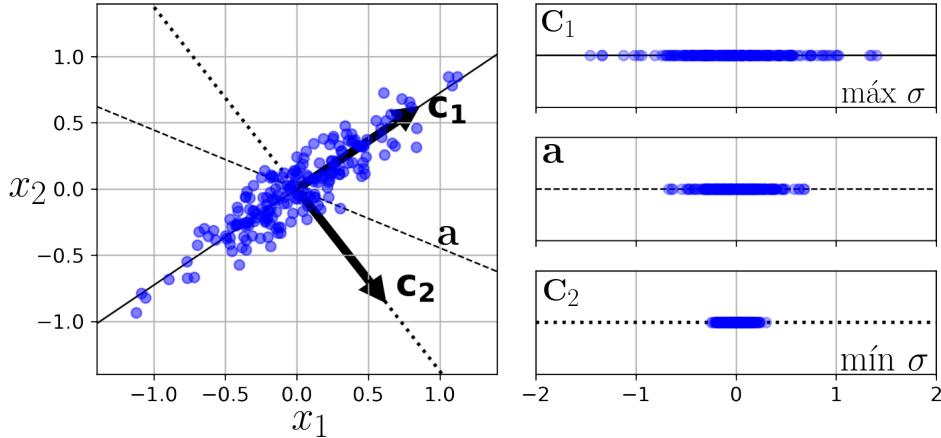
### 3.3. Análisis de Componentes Principales (PCA)

Uno de los usos centrales del SVD es el Análisis de Componentes Principales (PCA)<sup>2</sup>, considerado el estándar lineal para la RD [23]. Este método provee de un sistema coordenado jerárquico de pocas dimensiones, capaz de representar datos de alta dimensión altamente correlacionados. La geometría del sistema coordinado resultante está determinado por los componentes principales (CPs), descorrelacionados entre sí (ortogonales), pero que tienen máxima correlación con las muestras originales. Los CPs son ordenados de acuerdo a la varianza que capturan, de mayor a menor. La idea es proyectar los datos originales en el sistema descrito por los CPs, conservando la mayor cantidad de varianza posible y pérdida de información mínima. En la figura 3.4 se ilustra la aplicación del PCA en un conjunto de datos 2D.

#### 3.3.1. Algoritmo PCA

A continuación se detalla el algoritmo PCA, cuyo resumen se encuentra en el cuadro de la Fig. 3.6. En términos geométricos, el método consiste en el centrado, escaleo y rotación de los datos originales  $\mathbf{X}$ , de manera que los datos se hallen estadísticamente

<sup>2</sup>En la literatura también se lo puede encontrar como Descomposición Ortogonal Propia (POD).



**Figura 3.4:** Idea detrás del PCA en un conjunto de datos 2D. Los CPs  $\mathbf{C}_1$  y  $\mathbf{C}_2$  son las direcciones de mayor varianza. Se muestra un eje intermedio adicional  $\mathbf{a}$ . A la derecha se representan las proyecciones de los datos en las tres direcciones. Imagen editada, extraída de la Ref. [6].

descorrelacionados, para finalmente proyectarlos en las direcciones de mayor varianza. La notación empleada es distinta a la de la literatura, donde las columnas suelen ser las características y las filas las muestras. En este trabajo, las columnas representan las muestras (instantáneas) y las filas las características (velocidad del nodo  $i$ -ésimo de la grilla).

## 1. Estandarización

En primer lugar, se estandarizan los datos originales  $\mathbf{X} \in \mathbb{R}^{n \times m}$ , de manera que cada variable tenga media cero y varianza unitaria. La razón yace en que las variables pueden tener distintas escalas, lo que puede llevar a que aquellas que tengan mayor varianza dominen el análisis. La estandarización se realiza restando la media fila a fila y dividiendo por la desviación estándar temporal de cada variable

$$\hat{X}_{ij} = \frac{X_{ij} - \mu_i}{\sigma_i}, \quad (3.7)$$

donde la media (temporal en esta notación)  $\boldsymbol{\mu} \in \mathbb{R}^n$  se calcula como

$$\mu_i = \frac{1}{m} \sum_{j=1}^m X_{ij}, \quad (3.8)$$

y la desviación estándar  $\boldsymbol{\sigma} \in \mathbb{R}^n$  como

$$\sigma_i = \sqrt{\frac{1}{m} \sum_{j=1}^m (X_{ij} - \mu_j)^2}. \quad (3.9)$$

## 2. Matriz de covarianza

El paso siguiente es construir la matriz de covarianza de  $\hat{\mathbf{X}}$ . La matriz de covarianza  $\mathbf{C} \in \mathbb{R}^{m \times m}$  es una matriz simétrica y semidefinida positiva

$$\mathbf{C}_{\mathbf{T}} = \frac{1}{n-1} \hat{\mathbf{X}}^{\top} \hat{\mathbf{X}}. \quad (3.10)$$

En este contexto,  $\mathbf{C}_{\mathbf{T}}$  también se conoce como la matriz de correlación temporal, la cual toma el producto interno entre columnas de  $\hat{\mathbf{X}}$ . Alternativamente, se puede trabajar con la matriz de correlación espacial

$$\mathbf{C}_{\mathbf{S}} = \frac{1}{n-1} \hat{\mathbf{X}} \hat{\mathbf{X}}^{\top}. \quad (3.11)$$

A partir de la Ec. (3.2), ambas matrices se pueden expresar en términos de la SVD de la siguiente manera

$$\mathbf{C}_{\mathbf{T}} = \frac{1}{n-1} \mathbf{V} \boldsymbol{\Sigma}^2 \mathbf{V}^{\top}, \text{ } ^3 \quad (3.12)$$

$$\mathbf{C}_{\mathbf{S}} = \frac{1}{n-1} \mathbf{U} \boldsymbol{\Sigma}^2 \mathbf{U}^{\top}. \quad (3.13)$$

## 3. Componentes principales

Los componentes principales de  $\hat{\mathbf{X}}$  se determinan mediante los autovectores de la matriz de covarianza  $\mathbf{C}_T$  según

$$\mathbf{C}_{\mathbf{T}} \mathbf{V} = \mathbf{S} \mathbf{V}, \text{ donde} \quad (3.14)$$

$$\mathbf{C}_{\mathbf{T}} \mathbf{v}_i = \frac{1}{n-1} \sigma_i^2 \mathbf{v}_i, \quad (3.15)$$

Los autovectores  $\mathbf{v}_i \in \mathbb{R}^m$  son ortogonales entre sí, mientras que sus autovalores  $\frac{1}{n-1} \sigma_i^2$  representan la varianza en cada componente principal escalaada por  $\frac{1}{n-1}$ , y se encuentran ordenados de mayor a menor.

De la Ec. (3.12), se puede deducir que los autovectores  $\mathbf{V}$  son los vectores singulares derechos  $\mathbf{V}$  de la SVD y  $\sigma_i$  los valores singulares de  $\boldsymbol{\Sigma}$ , cumpliendo

$$\mathbf{C}_{\mathbf{T}} \mathbf{V} = \frac{1}{n-1} \mathbf{V} \boldsymbol{\Sigma}^2. \quad (3.16)$$

Análogamente, los autovectores de la matriz de correlación espacial corresponden a la

---

<sup>3</sup>Para matrices reales, la transpuesta conjugada es simplemente la transpuesta  $\mathbf{X}^{\dagger} = \mathbf{X}^{\top}$ .

matriz  $\mathbf{U}$  de la SVD, con mismos autovalores

$$\mathbf{C}_s \mathbf{U} = \frac{1}{n-1} \mathbf{U} \Sigma^2. \quad (3.17)$$

#### 4. Proyección al espacio latente y reconstrucción

Finalmente, para construir el espacio latente se proyectan los datos  $\hat{\mathbf{X}}$  en los primeros  $r$  CPs. De esta manera, el subespacio de dimensión temporal reducida  $\mathbf{Y} \in \mathbb{R}^{n \times r}$  se logra mediante

$$\mathbf{Y} = \hat{\mathbf{X}} \mathbf{V}_r = \mathbf{U}_r \Sigma_r, \quad (3.18)$$

donde  $\mathbf{U}_r$  y  $\mathbf{V}_r$  provienen de la aproximación de rango  $r$  de la SVD. Análogamente, se obtiene el subespacio de dimensión espacial reducida  $\mathbf{Y} \in \mathbb{R}^{r \times m}$  de

$$\mathbf{Y} = \mathbf{U}_r^\top \hat{\mathbf{X}} = \Sigma_r \mathbf{V}_r^\top. \quad (3.19)$$

Para reconstruir el espacio basta con rotar y proyectar los datos al espacio original a partir de la aproximación de  $\hat{\mathbf{X}}_r$  de la SVD dada por la Ec. (3.4), o alternativamente mediante

$$\mathbf{Y} = \hat{\mathbf{X}} \mathbf{V}_r \mathbf{V}_r^\top, \quad (3.20)$$

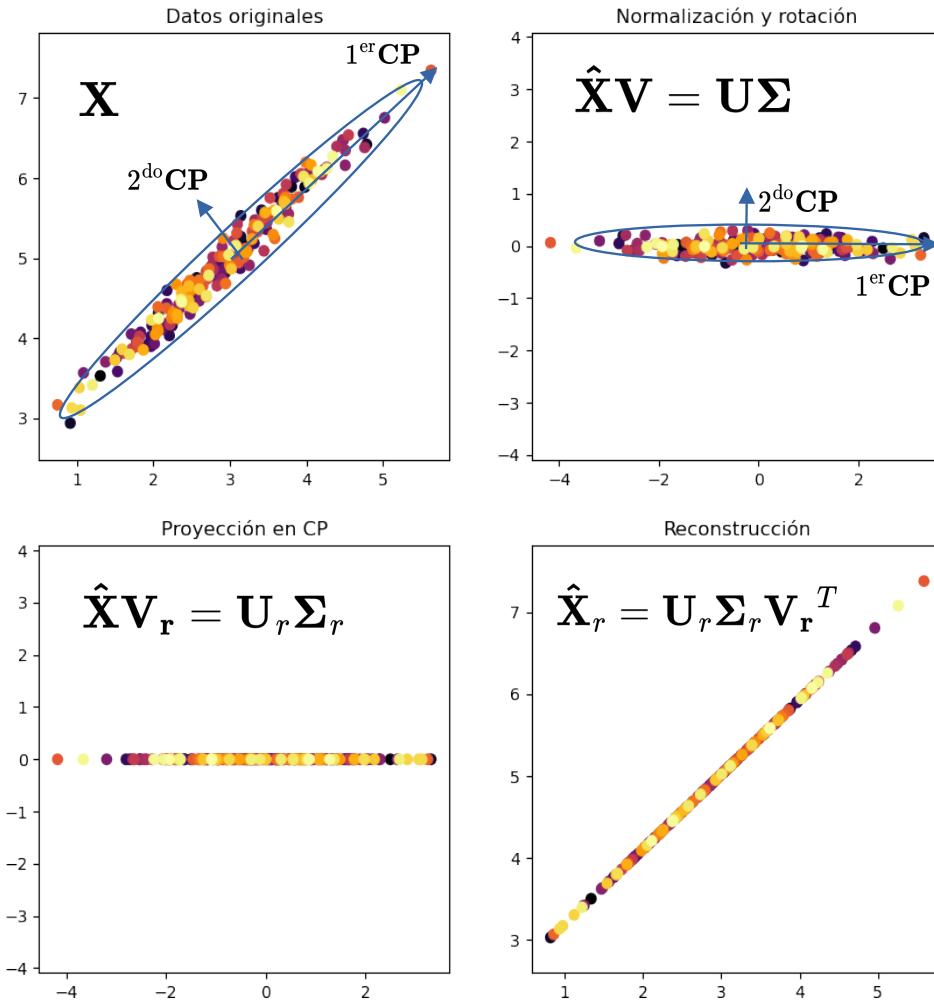
$$\mathbf{Y} = \mathbf{U}_r \mathbf{U}_r^\top \hat{\mathbf{X}}. \quad (3.21)$$

#### 3.3.2. Interpretación geométrica

El algoritmo del PCA se puede entender como el proceso que mejor ajusta un elipsoide  $r$ -dimensional al conjunto de datos luego de estandarizarlos. Los componentes principales son los ejes principales del elipsoide, congruentes a las direcciones que capturan la máxima varianza. La Figura 3.5 ilustra la interpretación geométrica del PCA en un conjunto de datos 2D. Cabe destacar que la operación  $\hat{\mathbf{X}} \mathbf{V}$  no tiene un propósito práctico, puesto que es una transformación dentro del mismo espacio y no reduce la dimensionalidad. No obstante, se grafica con el fin de facilitar la intuición geométrica. Normalmente, se proyecta directo al subespacio reducido mediante  $\hat{\mathbf{X}}_r \mathbf{V}_r$ .

#### 3.3.3. Costo computacional

El costo computacional del PCA hereda el costo de la SVD  $\mathcal{O}(n^2m)$ , con costo en memoria  $\mathcal{O}(n^2m)$ . Cabe remarcar la diferencia de optar por el camino directo de descomponer la matriz de covarianza en autovalores, sin recurrir a la SVD. En tal caso,



**Figura 3.5:** Interpretación geométrica del PCA en un conjunto de datos 2D. Los CPs son los ejes de la elipse que mejor se ajusta a los datos.

la complejidad de construir  $\mathbf{C}$  es  $\mathcal{O}(n \times m^2)$ , y la de calcular los autovectores  $\mathcal{O}(m^3)$ , resultando un costo general de  $\mathcal{O}(n \times m^2 + m^3)$ .

### 3.4. Kernel PCA (KPCA)

Una extensión no-lineal de PCA es el Kernel PCA (KPCA) [24]. Este método calcula los autovectores de la llamada matriz de *kernel*  $\mathcal{K}$ . Parte de la premisa de que los datos pueden estar en una variedad no-lineal. Por lo tanto, primero los datos se proyectan a un espacio de mayor dimensión  $D > n$ , de modo que se encuentren aproximadamente en un subespacio lineal, como ilustra la Figura 3.7. Luego, se aplica PCA a los datos en este espacio de mayor dimensión.

Sin embargo, lo poderoso del método radica en que el mapeo  $\Phi \in \mathbb{R}^{D \times m}$  nunca se calcula explícitamente gracias al “truco del kernel”. Un resumen del algoritmo KPCA se

### ALGORITMO PCA

1. Estandarizar los datos  $\mathbf{X}$  (media nula y varianza unitaria).
2. Construir la matriz de covarianza  $\mathbf{C}$ .
3. Calcular los autovectores  $\mathbf{V}$  y autovalores  $\mathbf{S}$  de  $\mathbf{C}$  (se obtienen de la SVD).
4. Proyectar los datos  $\hat{\mathbf{X}}$  en los  $r$  CPs, contenidos en las columnas de  $\mathbf{V}_r$  según Ec. (3.18)/(3.19).
5. Reconstruir rotando y proyectando los datos al espacio original según Ec. (3.20)/(3.21).

**Figura 3.6:** Resumen del algoritmo PCA clásico. En la práctica, no se calcula la matriz de covarianza, dado que los autovectores y autovalores se obtienen directamente de la SVD.

encuentra en la Fig. 3.9.

#### 3.4.1. Algoritmo KPCA

Se considera nuevamente el conjunto de datos  $\hat{\mathbf{X}} = [\mathbf{x}_1, \dots, \mathbf{x}_m] \in \mathbb{R}^{n \times m}$  y una transformación no-lineal arbitraria  $\Phi : \mathbb{R}^n \rightarrow \mathcal{F}$ , donde  $\mathcal{F}$  tiene alta dimensionalidad (incluso infinita). Asumamos por el momento que los datos mapeados  $[\Phi(\mathbf{x}_1), \dots, \Phi(\mathbf{x}_m)]$  se hallan centrados, i.e.  $\sum_{j=1}^m \Phi(\mathbf{x}_j) = 0$ . Luego, para aplicar PCA en el espacio  $\mathcal{F}$ , se debe encontrar los autovectores y autovalores de la matriz de covarianza

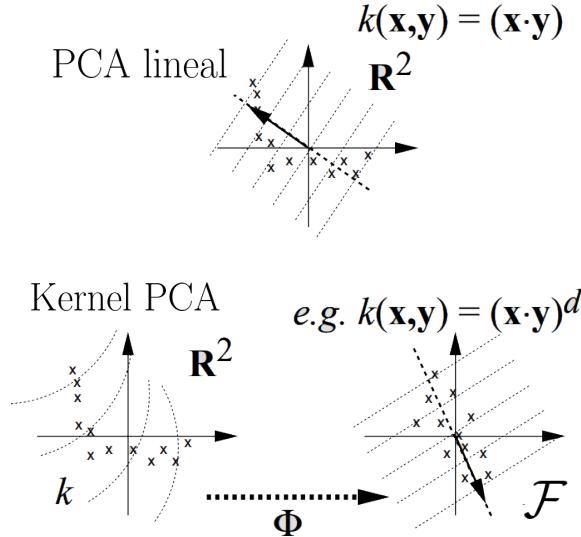
$$\mathbf{C}_{\mathcal{F}} = \Phi^T \Phi = \frac{1}{n-1} \sum_{j=1}^m \Phi(\mathbf{x}_j) \Phi(\mathbf{x}_j)^T. \quad (3.22)$$

Los mismos yacen en el espacio generado por  $\Phi(\mathbf{x}_1), \dots, \Phi(\mathbf{x}_n)$ , por lo tanto se puede considerar el sistema de ecuaciones

$$\lambda(\Phi(\mathbf{x}_j) \cdot \mathbf{V}) = (\Phi(\mathbf{x}_j) \cdot \mathbf{C}_{\mathcal{F}} \mathbf{V}) \text{ para todo } j = 1, \dots, m, \quad (3.23)$$

y existen coeficientes  $\alpha_i (i = 1, \dots, m)$  tal que

$$\mathbf{V} = \sum_{i=1}^m \alpha_i \Phi(\mathbf{x}_i). \quad (3.24)$$



**Figura 3.7:** Representación conceptual del Kernel PCA. Los datos son mapeados a un espacio de mayor dimensión  $\mathcal{F}$  donde se encuentran aproximadamente en un subespacio lineal. Cuando la función de *kernel* es el producto interno usual, se recupera el PCA lineal. Imagen extraída de la Ref. [24].

Combinando ambas ecuaciones se obtiene para todo  $j = 1, \dots, m$

$$\lambda \sum_{i=1}^m \alpha_i (\Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}_i)) = \frac{1}{m} \sum_{i=1}^m \alpha_i \left( \Phi(\mathbf{x}_j) \cdot \sum_{j=1}^m \Phi(\mathbf{x}_j) \right) (\Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}_i)). \quad (3.25)$$

Definimos la matriz de *kernel*  $\mathcal{K} \in \mathbb{R}^{m \times m}$

$$\mathcal{K}_{ij} = (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)) = \kappa(\mathbf{x}_i, \mathbf{x}_j), \quad (3.26)$$

luego, siendo  $\boldsymbol{\alpha}$  el vector de coeficientes, se obtiene la ecuación de autovalores y autovectores

$$\begin{aligned} m\lambda\mathcal{K}\boldsymbol{\alpha} &= \mathcal{K}^2\boldsymbol{\alpha} \\ \Rightarrow m\lambda\boldsymbol{\alpha} &= \mathcal{K}\boldsymbol{\alpha}. \end{aligned} \quad (3.27)$$

Por lo tanto,  $\boldsymbol{\alpha}$  es autovector de  $\mathcal{K}$ . Las soluciones  $\boldsymbol{\alpha}^4$  se normalizan, requiriendo que los vectores correspondientes en  $\mathcal{F}$  también estén normalizados, i.e.  $(\mathbf{V}^k \cdot \mathbf{V}^k) = 1$ , de manera que se tiene

---

<sup>4</sup>Soluciones de autovalores no nulos.

$$1 = \sum_{i,j=1}^m \alpha_i^k \alpha_j^k (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)) = (\boldsymbol{\alpha}^k \cdot K \boldsymbol{\alpha}^k) = \lambda_k (\boldsymbol{\alpha}^k \cdot \boldsymbol{\alpha}^k). \quad (3.28)$$

De esta manera, se proyectan los datos en los CPs de  $\mathcal{F}$  mediante la proyección de la imagen de un punto de prueba  $\Phi(\mathbf{x})$  en los autovectores  $\mathbf{V}^k$  de la siguiente forma

$$(\mathbf{V}^k \cdot \Phi(\mathbf{x})) = \sum_{i=1}^{\ell} \alpha_i^k (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x})). \quad (3.29)$$

Cabe remarcar, que tanto en la Ec.(3.26) como en la Ec. (3.29) no se requiere  $\Phi$  de forma explícita, sino que se trabaja directamente con el producto interno. Tomando en consideración que  $\Phi$  es arbitraria, existen funciones de *kernel*  $\kappa$  adecuadas que efectivamente determinan el producto interno en  $\mathcal{F}$  [25]. De esta manera, el truco del *kernel* nos permite trabajar en un espacio de alta dimensión sin necesidad de calcular explícitamente  $\Phi$ . Entre funciones de *kernel* populares se hallan las siguientes:

$$\begin{aligned} \kappa(\mathbf{x}, \mathbf{y}) &= (\mathbf{x}^\top \mathbf{y})^p \quad \text{polinómica,} \\ \kappa(\mathbf{x}, \mathbf{y}) &= \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|^2) \quad (\text{RBF}), \\ \kappa(\mathbf{x}, \mathbf{y}) &= \tanh(\kappa \mathbf{x}^\top \mathbf{y} + b) \quad \text{sigmoidal.} \end{aligned} \quad (3.30)$$

En este trabajo se emplea la *radial basis function* (RBF). El parámetro  $\gamma$  controla el ancho de la función RBF. Un valor alto de  $\gamma$  da como resultado una función RBF más estrecha, lo que significa que las instancias de datos cercanas (en  $\hat{\mathbf{X}}$ ) tendrán una mayor similitud en  $\mathcal{F}$  que las instancias de datos lejanas. Por el contrario, un valor bajo de  $\gamma$  da como resultado una función RBF más ancha, lo que significa que las instancias de datos cercanas y lejanas tendrán una similitud similar en  $\mathcal{F}$ .

### Centrado de los datos en $\mathcal{F}$

Al principio del desarrollo, se asumió que los datos en el espacio  $\mathcal{F}$  estaban centrados, lo cual es un requisito para el cálculo efectivo de componentes principales. En general esto no es así, por lo tanto se corrige de manera que

$$\hat{\Phi}(x^{(i)}) = \Phi(x^{(i)}) - \frac{1}{N} \sum_{k=1}^N \Phi(x^{(k)}). \quad (3.31)$$

De esta manera, se define la matriz de *kernel* centrada  $\tilde{\mathcal{K}}$

$$\tilde{\mathcal{K}} = \mathcal{K} - \mathbf{1}_m \mathcal{K} - \mathcal{K} \mathbf{1}_m + \mathbf{1}_m \mathcal{K} \mathbf{1}_m, \quad (3.32)$$

donde  $\mathbf{1}_m \in \mathbb{R}^{m \times m}$  es una matriz de unos.

### Proyección al espacio latente

La descomposición de la matriz de *kernel* centrada  $\tilde{\mathcal{K}} \in \mathbb{R}^{m \times m}$  y la proyección al espacio latente  $\mathbf{Y} \in \mathbb{R}^{r \times m}$  se obtiene de forma análoga a PCA. Primero, se descompone en autovectores y autovalores la matriz de *kernel* centrada

$$\tilde{\mathcal{K}} = \mathbf{W}\Lambda\mathbf{W}^\top. \quad (3.33)$$

Luego, se proyectan los datos en el espacio latente  $\mathbf{Y}$  mediante

$$\mathbf{Y} = \Lambda_r^{1/2}\mathbf{W}_r^\top, \quad (3.34)$$

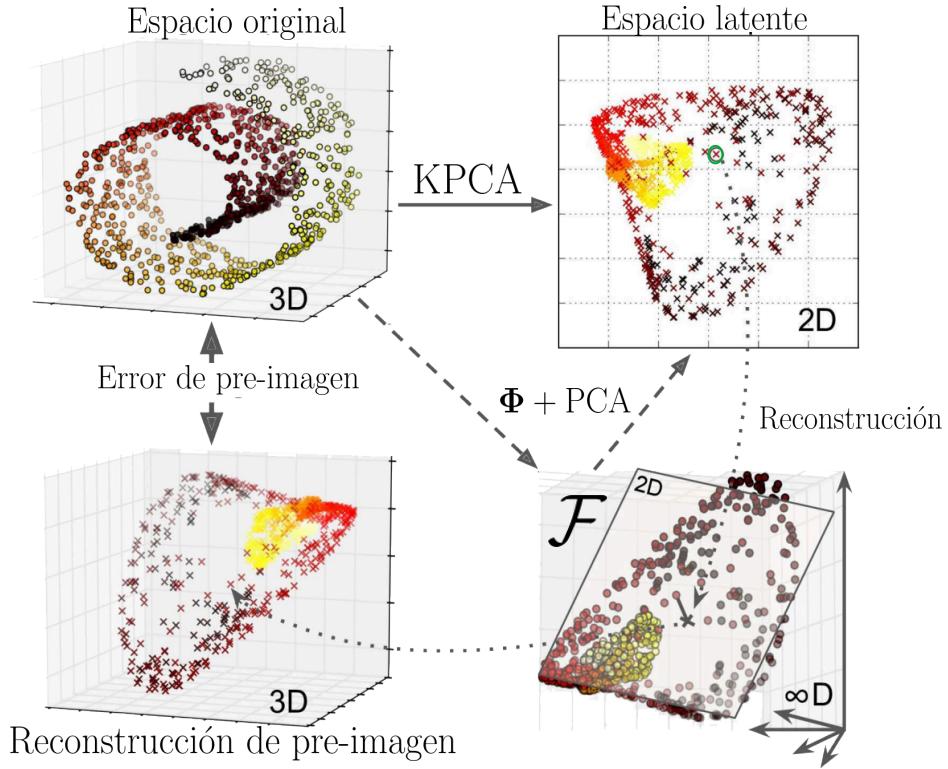
donde  $\mathbf{W}_r$  y  $\Lambda_r$  son las matrices de autovectores y autovalores truncados respectivamente. Por analogía al PCA, los autovalores  $\lambda_i$  de  $\Lambda$  representan la varianza de la matriz de *kernel*. Por lo tanto, se toma la raíz cuadrada.

### 3.4.2. Reconstrucción

En lo que respecta a la reconstrucción de los datos a partir del espacio latente, es un proceso no trivial dado que el mapeo inverso en general no existe, lo que se conoce como el problema de la pre-imagen [26]. Esto es, si quisieramos invertir el PCA a partir de un dato en el espacio reducido, su reconstrucción estaría en  $\mathcal{F}$  como ilustra en la Fig. 3.8 con una  $\times$ . Dado que  $\mathcal{F}$  puede ser infinito-dimensional, el mapeo inverso puede no existir, por lo tanto no se podría calcular el error de reconstrucción. Sin embargo, es posible encontrar un punto en el espacio original que se mapearía cerca del punto reconstruido. Este punto es la pre-imagen de reconstrucción, que una vez obtenido se puede determinar su distancia al punto original. La reconstrucción se puede abordar a partir del aprendizaje de pre-imagen [27], que se basa en entrenar un modelo supervisado de regresión, con datos proyectados como datos de entrenamiento y los originales como objetivos. De esta manera, aprende una función lineal en el espacio inducido. Más detalle sobre este enfoque en la Ref. [28].

### 3.4.3. Costo computacional

El costo computacional del KPCA corresponde a la construcción y resolución de la descomposición en autovectores de la matriz de *kernel*. Por consiguiente, el costo es de  $\mathcal{O}(n \times m^2 + m^3)$ .



**Figura 3.8:** Ilustración del problema de la pre-imagen en el KPCA. La  $X$  representa un punto en el espacio latente, cuya reconstrucción está en  $\mathcal{F}$ . La pre-imagen es el punto en el espacio original que se mapearía cerca del punto reconstruido. Imagen editada, extraída de la Ref. [6].

### 3.5. Locally Linear Embedding (LLE)

El *Locally Linear Embedding* (LLE) [29] es un método no-lineal de RD del aprendizaje de variedades, cuya idea principal se basa en la intuición geométrica de preservar las relaciones locales entre los datos. Al igual que el PCA, no involucra mínimos locales y, en contraste, es capaz de generar espacios latentes altamente no-lineales. En síntesis, el algoritmo consiste en primero cuantificar cuanto se relacionan linealmente los datos de  $\mathbf{X} \in \mathbb{R}^{n \times m}$  con sus vecinos más cercanos y luego buscar una representación de baja dimensión en  $\mathbf{Y} \in \mathbb{R}^{r \times m}$  que conserve estas relaciones.

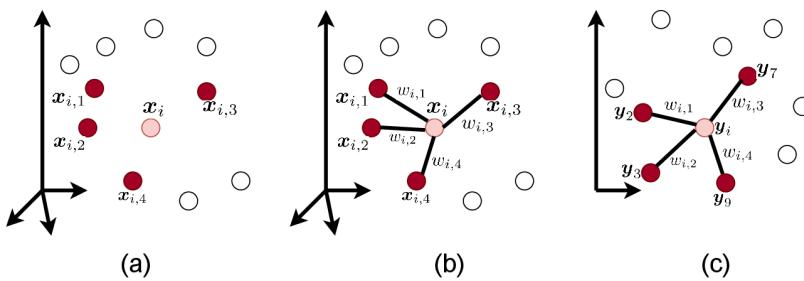
#### 3.5.1. Algoritmo LLE

A continuación se detalla el algoritmo del LLE [30], constituido por tres pasos para la construcción del espacio latente, ilustrados conceptualmente en la Fig. 3.10. Se parte de los datos preprocessados (estandarizados)  $\hat{\mathbf{X}} \in \mathbb{R}^{n \times m}$ , donde  $n$  es la dimensión espacial y  $m$  la dimensión temporal. En el cuadro de la Fig. 3.11 se encuentra un resumen del algoritmo.

### ALGORITMO KPCA

1. Elegir la función de *kernel*  $\kappa(\mathbf{x}_i, \mathbf{x}_j)$  y construir la matriz de *kernel* centrada  $\hat{\mathcal{K}}$  dada por la Ec. (3.32).
2. Calcular los autovectores y autovalores de  $\hat{\mathcal{K}}$ .
3. Proyectar los datos en el espacio latente  $\mathbf{Y} \in \mathbb{R}^{r \times m}$  como describe la Ec. (3.34).
4. Reconstruir los  $\mathbf{x}_i$  a partir del método de aprendizaje de pre-imagen.

**Figura 3.9:** Resumen del algoritmo KPCA. Los primeros tres pasos corresponden a la construcción del espacio latente, y el cuarto a la reconstrucción de los datos a la dimensión original.



**Figura 3.10:** Pasos del algoritmo LLE. (a) Identificación de los  $K$ -NN  $\mathbf{x}_i$  en el espacio original (b) a partir de los cuales se optimiza la reconstrucción lineal de  $\mathbf{x}_i$ . (c) Determinación de los  $\mathbf{y}_i$  en el espacio latente usando los pesos calculados en (b). En la figura se representan 4  $K$ -NN en rojo. Figura tomada de la Ref. [7].

#### 1. Identificación de los $K$ -NN

El LLE parte del supuesto de que la vecindad local de un punto  $\mathbf{x}_i \in \mathbb{R}^n$  puede aproximarse mediante el subespacio lineal generado por sus  $k$  vecinos más cercanos ( $K$ -NN). Por lo tanto, esta vecindad corresponde a las instantáneas de velocidad similares a  $\mathbf{x}_i$ . El primer paso del algoritmo consiste en encontrar los  $K$ -NN de cada punto  $\mathbf{x}_i$  en el espacio original. Esto se logra calculando las distancias entre  $\mathbf{x}_i$  y todos los demás puntos  $\mathbf{x}_j$ , asignando los  $K$  puntos más cercanos a  $\mathbf{x}_i$  como sus  $K$ -NN. La métrica del método puede ser elegida<sup>5</sup> y en este caso se utiliza la distancia euclídea.

El número de  $K$ -NN es el parámetro libre del método que se elige a priori. Tiene un impacto directo en el rendimiento, tanto computacionalmente como en calidad. Un valor de  $K$  demasiado pequeño puede resultar en una representación insuficiente de la estructura local de los datos. Por otro lado, un valor de  $K$  demasiado grande puede incorporar relaciones no locales, es decir, ponderar vecinos lejanos en el análisis, lo que

<sup>5</sup>Por ejemplo, podrían elegirse puntos dentro de una bola de radio fijo, o reglas más sofisticadas.

introduce ruido en la representación reducida.

## 2. Cálculo de coeficientes $w_{ij}$

Cada punto  $\mathbf{x}_i$  se puede aproximar como una combinación lineal de sus vecinos más cercanos con coeficientes  $w_{ij}$

$$\mathbf{x}_i \approx \sum_{j=1}^m w_{i,j} \mathbf{x}_j. \quad (3.35)$$

De esta forma, el segundo paso es calcular los coeficientes  $w_{ij}$  que minimizan el error entre  $\mathbf{x}_i$  y su aproximación resolviendo

$$\min_{w_{ij}} \sum_{j=1}^m \left\| \mathbf{x}_i - \sum_{i=1}^m w_{i,j} \mathbf{x}_j \right\|^2, \quad (3.36)$$

$$\text{sujeto a } \sum_{i=1}^m w_{ij} = 1. \quad (3.37)$$

Notar que si  $\mathbf{x}_j$  no es  $K$ -NN de  $\mathbf{x}_i$ , entonces  $w_{ij} = 0$ <sup>6</sup>. La solución a este problema de optimización para los pesos de  $\mathbf{x}_i$  tiene la forma

$$\tilde{\mathbf{w}}_i = \frac{\mathbf{G}_i^{-1} \mathbf{1}}{\mathbf{1}^T \mathbf{G}_i^{-1} \mathbf{1}} \in \mathbb{R}^k, \quad (3.38)$$

donde  $\mathbf{1} \in \mathbb{R}^k$  es un vector de unos. Aquí  $\mathbf{G}_i \in \mathbb{R}^{(K \times K)}$  es la matriz local de Gram en  $\mathbf{x}_i$ , definida como

$$\mathbf{G}_i = \mathbf{G}_{jk}^i = (\mathbf{x}_j - \mathbf{x}_i)^T (\mathbf{x}_k - \mathbf{x}_i), \quad (3.39)$$

donde  $\mathbf{x}_j$  y  $\mathbf{x}_k$  son  $K$ -NN de  $\mathbf{x}_i$ . Cabe destacar que  $\mathbf{w}_i \in \mathbb{R}^m$ , es decir,  $\tilde{\mathbf{w}}_i$  contiene los pesos no nulos de  $\mathbf{w}_i$  correspondientes únicamente a los  $K$ -NN de  $\mathbf{x}_i$ .

## 3. Construcción del espacio latente

Los pesos que minimizan el error en la Ec. (3.36) sujeto a la condición en la Ec. (3.37) presentan una simetría notable. Esto es, son invariantes ante rotaciones, escalamientos, y traslaciones de un punto y sus vecinos. Una consecuencia directa de esta simetría es que los pesos caracterizan propiedades geométricas intrínsecas de cada vecindad local, que no dependen del marco de referencia. Luego, se espera que la caracterización de la geometría local en el espacio original sea válida para recintos locales de la variedad. Esto se traduce

---

<sup>6</sup>Sin embargo, ello no implica que  $\mathbf{x}_i$  no sea  $K$ -NN de  $\mathbf{x}_j$ .

a que los pesos  $w_{ij}$  que reconstruyen  $\mathbf{x}_i$  en  $\mathbb{R}^n$ , deberían ser válidos para reconstruir  $\mathbf{y}_i$  en el espacio latente  $\mathbb{R}^r$ .

Basado en esta premisa, el espacio latente  $\mathbf{Y} \in \mathbb{R}^{r \times m}$ <sup>7</sup> se obtiene calculando los  $\mathbf{y}_i$  que minimizan el error de reconstrucción en el espacio latente dado por la ecuación

$$\min_{\mathbf{Y}} \sum_{i=1}^m \left\| \mathbf{y}_i - \sum_{j=1}^m w_{i,j} \mathbf{y}_j \right\|^2, \quad (3.40)$$

sujeto a las condiciones

$$\sum_{j=1}^m \mathbf{y}_j = \mathbf{0}, \quad (3.41)$$

$$\frac{1}{m} \sum_{j=1}^m \mathbf{y}_j \mathbf{y}_j^T = \mathbb{I}, \quad (3.42)$$

donde  $w_{ij}$  son los pesos calculados en el paso anterior.

La solución a este problema [7] de optimización son los  $r$  autovectores de la matriz

$$\mathcal{L} = (\mathbb{I} - \mathbf{W})^T (\mathbb{I} - \mathbf{W}) \in \mathbb{R}^{m \times m}, \quad (3.43)$$

correspondientes a los  $[2, r + 1]$  autovalores más pequeños<sup>8</sup>.  $\mathbb{I} \in \mathbb{R}^{m \times m}$  es la matriz identidad y  $\mathbf{W} \in \mathbb{R}^{m \times m}$  es la matriz de pesos  $\mathbf{w}_j$ . Se establecen los autovectores de  $\mathcal{L}$  como las filas de  $\mathbf{Y}$ .

### 3.5.2. Reconstrucción

En este trabajo se emplea el LLE inverso, presentado en la Ref. [7]. El procedimiento es similar al paso 2 del algoritmo, con la salvedad que se parte del espacio latente. Primero, se identifican los  $K$ -NN de cada  $\mathbf{y}_i$  en  $\mathbf{Y}$ . Luego, se calculan los pesos  $w_{ij}^*$  que mejor aproximan linealmente cada  $\mathbf{y}_i$  a partir de sus  $K$ -NN. Para ello, se minimiza el error de reconstrucción en el espacio latente dado por la ecuación

$$\min_{\mathbf{Y}} \sum_{i=1}^k \left\| \mathbf{y}_i - \sum_{j=1}^k w_{i,j}^* \mathbf{y}_j \right\|^2, \quad (3.44)$$

sujeto a

---

<sup>7</sup>La dimensión  $r$  de los  $\mathbf{y}_i$  en el espacio latente se elige a priori.

<sup>8</sup>El más pequeño es 0, por lo que se omite.

$$\sum_{i=1}^k w_{ij}^* = 1, \quad (3.45)$$

donde los  $\mathbf{y}_i$  son los hallados en el paso 2 del algoritmo. Luego, la reconstrucción de  $\mathbf{x}_i$  está dada por la combinación lineal de sus  $K$ -NN en  $\hat{\mathbf{X}}$  utilizando los pesos  $w_{ij}^*$

$$\tilde{\mathbf{x}}_i \approx \sum_{j=1}^k w_{i,j}^* \mathbf{x}_j. \quad (3.46)$$

### ALGORITMO LLE

1. Encontrar los  $K$ -NN de  $\mathbf{x}_i$ .
2. Calcular los pesos  $w_{ij}$  que mejor reconstruyen cada  $\mathbf{x}_i$  a partir de sus  $K$ -NN, minimizando el error en la Ec. (3.36) sujeto a las condiciones en la Ec. (3.37).
3. Calcular los vectores  $\mathbf{y}_i \in \mathbb{R}^r$  mejor reconstruidos por los pesos  $w_{ij}$ , minimizando el error en la Ec. (3.40) por sus vectores propios no nulos más pequeños.
4. Calcular los pesos  $w_{ij}^*$  que mejor reconstruyen cada  $\mathbf{y}_i$  a partir de sus  $K$ -NN en  $\mathbf{Y} \in \mathbb{R}^{r \times m}$ , minimizando el error en la Ec. (3.44) sujeto a las condiciones en la Ec. (3.45). Reconstruir  $\mathbf{x}_i$  con los pesos  $w_{ij}^*$  y sus  $K$ -NN en  $\hat{\mathbf{X}} \in \mathbb{R}^{n \times m}$ .

**Figura 3.11:** Resumen del algoritmo LLE. Los primeros tres pasos corresponden a la construcción del espacio latente, y el cuarto a la reconstrucción de los datos a la dimensión original.

### 3.5.3. Costo computacional

El cálculo del costo computacional del algoritmo LLE para la reducción de  $n$  a  $r$  dimensiones, de un conjunto de  $m$  muestras, comprende tres etapas.

1. **Búsqueda de vecinos más cercanos:** LLE utiliza el algoritmo *BallTree* para una búsqueda eficiente de vecinos. El costo es aproximadamente de orden  $\mathcal{O}[n \log(K)m \log(m)]$ , para  $K$  vecinos más cercanos.
2. **Construcción de la matriz de pesos:**  $\mathcal{O}[nmk^3]$ . La construcción de la matriz de pesos de LLE implica la solución de una ecuación lineal de  $k \times k$  para cada uno de los  $m$  vecindarios locales.
3. **Descomposición en autovectores y autovalores:** el espacio latente se codifica en los autovectores correspondientes a los  $r$  mayores autovalores de *kernel*  $\mathcal{L}$  del

LLE de tamaño  $m \times m$ . Para un *solver* denso, el costo es aproximadamente  $\mathcal{O}[rm^2]$ .

La complejidad total de LLE es  $\mathcal{O}[n \log(K)m \log(m)] + O[nmK^3] + O[rm^2]$ .

## 3.6. Isometric Mapping (Isomap)

El algoritmo del Isometric Mapping (Isomap) [31] parte de una premisa similar a la del LLE en cuanto a preservar similaridad local entre los datos en el espacio original y el espacio latente. Es una extensión no-lineal del Multi Dimensional Scaling (MDS) [32], el cual se basa en preservar las distancias euclídeas entre los puntos. La idea del Isomap, en vez de preservar la distancia euclídea, es preservar la distancia geodésica, que corresponde a la más corta entre dos puntos siguiendo la topología de la variedad. Esto se ejemplifica en la Fig. 3.12 (A) para un *Swiss roll*, donde la línea punteada representa la distancia euclídea y la sólida la geodésica.

### 3.6.1. Algoritmo Isomap

El algoritmo Isomap se compone de tres pasos para la RD. Se considera el conjunto de datos preprocesados (estandarizados)  $\hat{\mathbf{X}} \in \mathbb{R}^{n \times m}$ , donde  $n$  es la dimensión espacial y  $m$  la dimensión temporal. Un resumen del algoritmo se encuentra en el cuadro de la Fig 3.13.

#### 1. Construcción del grafo de vecinos más cercanos

En la práctica, en vez de contar explícitamente con el espacio al que pertenece el problema, se suele trabajar con un conjunto de muestras  $\hat{\mathbf{X}}$  pertenecientes a una variedad desconocida que se supone suave. Por consiguiente, se puede aproximar la distancia geodésica real por medio del camino más corto en un grafo de vecinos cercanos en  $\hat{\mathbf{X}}$ . De esta forma, el primer paso del algoritmo consiste en construir un grafo pesado de los  $K$ -NN de todos los datos, donde los nodos son los  $\mathbf{x}_i$  y las aristas (pesos del grafo) son las distancias euclidianas entre estos, calculadas como

$$d_{\hat{\mathbf{X}}}(i, j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2 \quad \text{si } \mathbf{x}_i \text{ y } \mathbf{x}_j \text{ son } K\text{-NN}. \quad (3.47)$$

#### 2. Cálculo de las distancias geodésicas

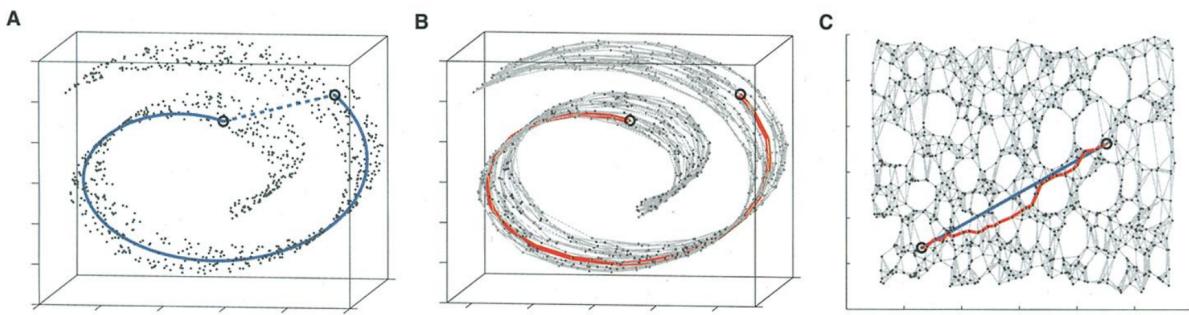
Una vez construido el grafo pesado de  $K$ -NN, se estiman las distancias geodésicas entre todos los pares de puntos. Dado dos puntos  $\mathbf{x}_i$  y  $\mathbf{x}_j$ , la distancia geodésica entre ellos es la suma de las distancias euclidianas  $d_{\hat{\mathbf{X}}}$  de las aristas que definen el camino más

corto entre ambos puntos (línea roja **Fig. 3.12 (B)**). De esta manera, se construye la matriz de distancias  $\mathbf{D} \in \mathbb{R}^{(m \times m)}$  donde

$$D_{ij} = \text{distancia geodésica entre } \mathbf{x}_i \text{ y } \mathbf{x}_j, \quad (3.48)$$

$$D_{ij} = \sum_{(p,q) \in P^*(i,j)} d_{\hat{\mathbf{x}}}(p,q), \quad (3.49)$$

donde  $P^*(i,j)$  representa el camino más corto en el grafo desde el nodo  $\mathbf{x}_i$  hasta el nodo  $\mathbf{x}_j$ . Para ello, el método más comúnmente utilizado es el algoritmo de Dijkstra, o alternativamente, el de Floyd-Warshall [33]. Posteriormente, se crea la matriz de distancias cuadradas:  $S_{ij} = D_{ij}^2$ .



**Figura 3.12:** Distancia euclídea y geodésica entre dos puntos arbitrarios (redondeados) en un conjunto de datos 3D. (A) Las líneas punteada y sólida representan la distancia euclídea y geodésica (real), respectivamente. (B) El grafo de vecinos cercanos (líneas grises) permite aproximar la distancia geodésica real por el camino más corto por K-NN (línea roja). (C) Espacio latente 2D obtenido por Isomap que preserva mejor la distancia geodésica. Imagen tomada de la Ref. [31].

### 3. Construcción del espacio latente

El tercer paso consiste en construir el espacio latente de dimensión reducida  $\mathbf{Y} \in \mathbb{R}^{r \times m}$  de manera similar al KPCA, donde  $\mathbf{K}$  toma el rol de la matriz de *kernel*. Primero, se define la matriz de disimilaridad doblemente centrada  $\mathbf{K} \in \mathbb{R}^{(m \times m)}$  (se resta la media de filas y columnas)

$$\mathbf{K} = -\frac{1}{2}\mathbf{H}\mathbf{S}\mathbf{H}, \quad (3.50)$$

donde  $\mathbf{H} = \mathbf{I} - \frac{1}{m}\mathbf{1}\mathbf{1}^T \in \mathbb{R}^{(m \times m)}$  es la matriz de centrado. La descomposición de autovectores y autovalores de  $\mathbf{K}$  puede escribirse como

$$\mathbf{K} = \mathbf{V}\Lambda\mathbf{V}^T, \quad (3.51)$$

donde  $\mathbf{V}$  y  $\Lambda$  contienen los vectores y valores propios de  $\mathbf{K}$ . De manera análoga al PCA,

el subespacio de baja dimensión  $\mathbf{Y} \in \mathbb{R}^{(r \times m)}$  puede escribirse entonces como

$$\mathbf{Y} = \Lambda_r^{1/2} \mathbf{V}_r^T, \quad (3.52)$$

donde el subíndice  $r$  se refiere a los primeros  $r$  valores y vectores propios más grandes correspondientes.

### 3.6.2. Reconstrucción

El método de reconstrucción en Isomap es el mismo que en LLE. Primero, se identifican los  $K$ -NN de cada  $\mathbf{y}_i$  en  $\mathbf{Y}$ . Luego, se calculan los pesos  $w_{ij}^*$  que mejor aproximan linealmente cada  $\mathbf{y}_i$  a partir de sus  $K$ -NN. Para ello, se minimiza el error de reconstrucción en el espacio latente dado por la siguiente fórmula

$$\min_{\mathbf{Y}} \sum_{i=1}^k \left\| \mathbf{y}_i - \sum_{j=1}^k w_{ij}^* \mathbf{y}_j \right\|^2, \quad (3.53)$$

$$\text{sujeto a } \sum_{i=1}^k w_{ij}^* = 1. \quad (3.54)$$

Luego, la reconstrucción de  $\mathbf{x}_i$  en el espacio original se realiza usando los mismos pesos  $w_{ij}^*$  aplicados a los vecinos  $K$ -NN en  $\hat{\mathbf{X}}$ .

Luego, la reconstrucción de  $\mathbf{x}_i$  se approxima por

$$\tilde{\mathbf{x}}_i \approx \sum_{j=1}^k w_{ij}^* \mathbf{x}_j. \quad (3.55)$$

### 3.6.3. Costo computacional

El cálculo del costo computacional del algoritmo Isomap para la reducción de  $n$  a  $r$  dimensiones, de un conjunto de  $m$  muestras, comprende tres etapas.

1. **Búsqueda de vecinos más cercanos:** al igual que el LLE, Isomap utiliza el algoritmo *BallTree* para una búsqueda eficiente de vecinos. El costo es aproximadamente  $\mathcal{O}[n \log(K)m \log(m)]$ , para  $K$  vecinos más cercanos.
2. **Búsqueda de caminos más cortos en el grafo:** los algoritmos más eficientes conocidos para esto son el algoritmo de Dijkstra, cuyo costo es aproximadamente  $\mathcal{O}[m^2(K + \log(m))]$ , o el algoritmo de Floyd-Warshall, cuyo costo es  $\mathcal{O}[m^3]$ . El

usuario puede seleccionar el algoritmo mediante el parámetro *path\_method* de Isomap. Si no se especifica, el código intenta elegir el mejor algoritmo para los datos de entrada.

3. **Descomposición en autovectores y autovalores:** el espacio latente se codifica en los autovectores correspondientes a los  $r$  mayores autovalores del *kernel* del Isomap de tamaño  $m \times m$ . Para un *solver* denso, el costo es aproximadamente  $\mathcal{O}[rm^2]$ .

La complejidad total de Isomap es  $\mathcal{O}[n \log(K)N \log(m)] + O[m^2(K + \log(m))] + O[rm^2]$ .

#### ALGORITMO ISOMAP

1. Construir el grafo pesado de  $K$ -NN de cada punto  $\mathbf{x}_i \in \mathbb{R}^n$ , cuyas aristas son las distancias euclídeas entre puntos.
2. Estimar la distancia geodésica entre todos los puntos calculando el camino más corto por  $K$ -NN en el grafo. Construir la matriz de distancias cuadradas  $S_{ij} = D_{ij}^2$ .
3. Construir el espacio latente  $\mathbf{Y} \in \mathbb{R}^{r \times m}$  aplicando MDS a la matriz de distancias cuadradas  $S_{ij}$ .
4. Calcular los pesos  $w_{ij}^*$  que mejor reconstruyen cada  $\mathbf{y}_i$  a partir de sus  $K$ -NN en  $\mathbf{Y} \in \mathbb{R}^r$ , minimizando el error en la Ec. (3.53) sujeto a las condiciones en la Ec. (3.54). Reconstruir  $\mathbf{x}_i$  con los pesos  $w_{ij}^*$  y sus  $K$ -NN en  $\hat{\mathbf{X}} \in \mathbb{R}^r$ .

**Figura 3.13:** Resumen del algoritmo Isomap. Los primeros tres pasos corresponden a la construcción del espacio latente, y el cuarto a la reconstrucción de los datos a la dimensión original.

## 3.7. Autoencoder (AE)

El último enfoque no-lineal que se presenta para la reducción de la dimensionalidad es el uso de redes neuronales, en particular, *autoencoders* (AE). Estos tienen una arquitectura específica, donde la salida está diseñada para ser igual que la entrada, y que tiene una capa *bottleneck* en el medio, con muchas menos neuronas. Esto obliga a la red a aprender una representación de baja dimensionalidad de los datos. A continuación, se presenta una introducción a las redes neuronales, para finalizar el capítulo con la descripción de un autoencoder.

### 3.7.1. Redes neuronales artificiales

Las redes neuronales artificiales (ANN) son sistemas inspirados en el cerebro humano, compuestos por neuronas interconectadas, nodos que transforman y transfieren información entre sí. Son capaces de aprender patrones complejos en los datos mediante el ajuste de los pesos de las conexiones entre neuronas, mediante un proceso iterativo de entrenamiento. Una ANN típica consiste en una capa de entrada, una o más capas ocultas y una capa de salida [34].

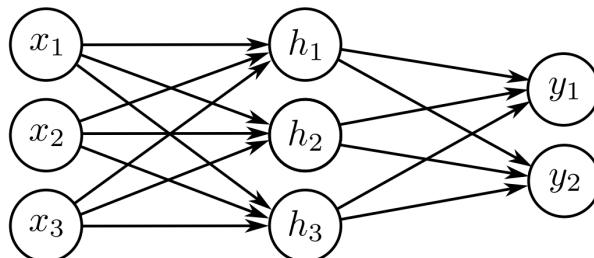
En el marco de este trabajo, se considera que cada neurona está conectada con todas las neuronas de la capa anterior. Este tipo de redes se conocen como completamente conectadas. En este tipo de arquitectura, las neuronas de una capa reciben como entrada las salidas o activaciones de todas las neuronas de la capa anterior. Para una capa dada, las activaciones de entrada se premultiplican por sus respectivos pesos de conexión ( $\Omega$ ), se suman y se añade un sesgo ( $\beta$ ). Este valor se utiliza como argumento de una función de activación ( $a$ ), cuya salida se propaga a las neuronas de la siguiente capa. La salida de cada neurona se calcula independientemente de las demás, y se puede expresar como:

$$h_i = \mathbf{a}(\beta_i + \omega_i \cdot \mathbf{h}_{in}), \quad \text{para } i = 1, 2, \dots, N, \quad (3.56)$$

donde  $h_i$  es la salida de la neurona  $i$ ,  $\mathbf{h}_{in}$  es el vector de entrada,  $\omega_i$  es el vector de pesos de conexión de la neurona  $i$ ,  $\beta_i$  es el sesgo de la neurona  $i$ , y  $\mathbf{a}$  es la función de activación. A su vez, la activación de una capa se puede escribir como

$$\mathbf{h} = \mathbf{a}(\beta + \Omega \mathbf{h}_{in}), \quad (3.57)$$

donde  $\mathbf{h}$  es el vector de salida,  $\Omega$  es la matriz de pesos,  $\beta$  es el vector de sesgo y  $\mathbf{a}$  es la función de activación. En la figura 3.14 se muestra un esquema de una ANN completamente conectada con una única capa oculta.



**Figura 3.14:** Ejemplo de una ANN completamente conectada con una capa oculta. La red tiene tres entradas ( $x$ ), dos salidas ( $y$ ) y una capa oculta con tres neuronas ( $h$ ). Imagen extraída de la Ref. [35].

La función de activación  $\mathbf{a}$  es la que permite al modelo describir relaciones no-lineales,

por lo que es inmediato elegir una no-lineal<sup>9</sup>. Existen varias funciones de activación, y entre las más comunes se encuentran la sigmoide, tangente hiperbólica y rectificada lineal (ReLU). Las expresiones de estas funciones son

$$\mathbf{a}(z) = \frac{1}{1+e^{-z}} \quad (\text{sigmoide}), \quad (3.58)$$

$$\mathbf{a}(z) = \tanh(z) \quad (\text{tangente hiperbólica}), \quad (3.59)$$

$$\mathbf{a}(z) = \max(0, z) \quad (\text{ReLU}). \quad (3.60)$$

La función de activación ReLU es la más utilizada en la actualidad y la que se emplea en este trabajo. Al argumento  $z$  se lo suele denominar pre-activación, y es la suma ponderada de las entradas de la neurona más el sesgo.

### 3.7.2. Redes neuronales profundas

Una red neuronal profunda (DNN) [35] es una ANN con varias capas entre la capa de entrada y la de salida. Las DNN funcionan como una serie de filtros sucesivos, cada uno refinando la representación de los datos de entrada. A medida que la información fluye a través de la red, se extraen niveles crecientes de abstracción, permitiendo a la red identificar patrones complejos. Al agregar y combinar suficientes capas, se puede aprender cualquier función, y luego “emularla”.

Un ejemplo de una DNN de tres capas ocultas se muestra en la Fig. 3.15, con la notación matricial asociada. Los pesos se almacenan en matrices  $\Omega_k$  que pre-multiplican las activaciones de la capa anterior para crear las pre-activaciones ( $\mathbf{z}$ ) en la capa siguiente. Por ejemplo, la matriz de pesos  $\Omega_1$  que calcula las pre-activaciones en  $\mathbf{h}_2$  a partir de las activaciones en  $\mathbf{h}_1$  tiene dimensión  $2 \times 4$ . Se aplica a las cuatro unidades<sup>10</sup> ocultas en la primera capa y crea las entradas para las dos unidades ocultas en la segunda capa. Los sesgos se almacenan en vectores  $\beta_k$  y tienen la dimensión de la capa a la que alimentan. Por ejemplo, el vector de sesgo  $\beta_2$  tiene longitud tres porque la capa  $\mathbf{h}_3$  contiene tres unidades ocultas.

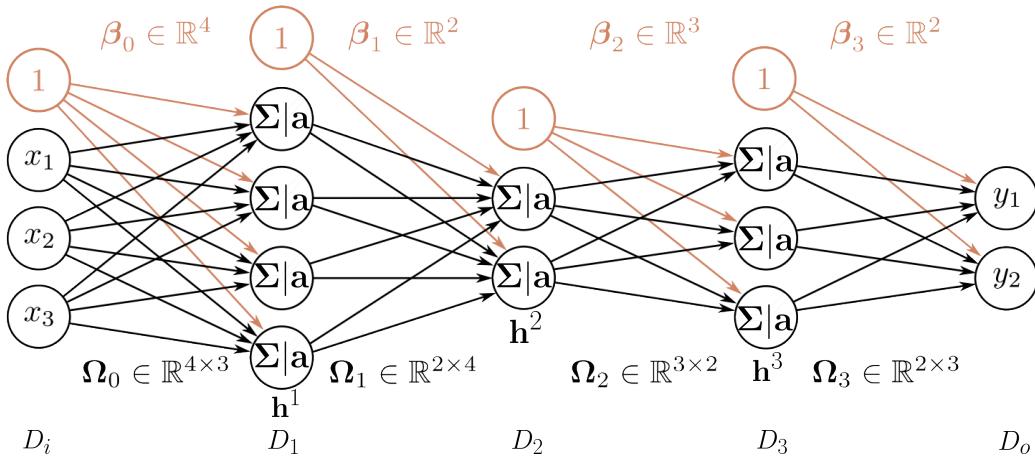
#### Formulación general

Se define el vector de unidades ocultas en la capa  $k$  como  $\mathbf{h}_k$ , el vector de sesgos que contribuyen a la capa oculta  $k + 1$  como  $\beta_k$ , y los pesos que se aplican a la capa  $k$  y

---

<sup>9</sup>De hecho, es fácil demostrar que si la función de activación es lineal, luego el mapeo general de la ANN también lo es.

<sup>10</sup>Se emplea el término unidad oculta para referirse a la neurona de una capa oculta de la red.



**Figura 3.15:** Cada neurona en una capa recibe entradas de las neuronas en la capa anterior, suma un sesgo y aplica una función de activación. Luego, envía la salida de la capa a las neuronas en la capa siguiente. Figura extraída y editada de [35].

contribuyen a la capa  $(k+1)$  como  $\Omega_k$ . Luego, la función de una DNN general  $\mathbf{y} = f[\mathbf{x}, \phi]$  con  $K$  capas se puede escribir como

$$\begin{aligned}\mathbf{h}_1 &= \mathbf{a} [\beta_0 + \Omega_0 \mathbf{x}], \\ \mathbf{h}_2 &= \mathbf{a} [\beta_1 + \Omega_1 \mathbf{h}_1], \\ \mathbf{h}_3 &= \mathbf{a} [\beta_2 + \Omega_2 \mathbf{h}_2], \\ &\vdots \\ \mathbf{h}_K &= \mathbf{a} [\beta_{K-1} + \Omega_{K-1} \mathbf{h}_{K-1}], \\ \mathbf{y} &= \beta_K + \Omega_K \mathbf{h}_K.\end{aligned}$$

Los parámetros  $\phi$  de este modelo comprenden todas estas matrices de pesos y vectores de sesgo  $\phi = \{\beta_k, \Omega_k\}_{k=0}^K$ . Podemos escribir de manera equivalente la red como una única función de la forma

$$\mathbf{y} = \beta_K + \Omega_K \mathbf{a} [\beta_{K-1} + \Omega_{K-1} \mathbf{a} [\dots \beta_2 + \Omega_2 \mathbf{a} [\beta_1 + \Omega_1 \mathbf{a} [\beta_0 + \Omega_0 \mathbf{x}]] \dots]]. \quad (3.61)$$

### 3.7.3. Entrenamiento de una ANN

Las redes neuronales artificiales aprenden los patrones complejos en los datos a través de un proceso de entrenamiento. En este proceso, los pesos y sesgos de cada capa se ajustan iterativamente para minimizar una *función de pérdida*, la cual cuantifica qué tan cerca está la salida de la red del valor objetivo deseado. El proceso consiste en elegir valores iniciales para los parámetros y luego iterar los siguientes dos pasos: (i) calcular

las derivadas (gradientes) de la función de pérdida con respecto a los parámetros, y (ii) ajustar los parámetros en función de los gradientes para disminuir la pérdida. Después de muchas iteraciones, esperamos alcanzar el mínimo global de la función de pérdida.

### Descenso del gradiente

Entrenar la red implica encontrar los parámetros  $\phi$  tal que el modelo  $f[\mathbf{x}_i, \phi]$  mapee la entrada  $\mathbf{x}_i$  al objetivo  $\hat{\mathbf{y}}_i$  con el mínimo error posible. En el caso del *autoencoder*, el valor objetivo es la misma entrada, se busca que la red comprima-descomprima la muestra con pérdidas mínimas. La función de pérdida más común es el error cuadrático medio (MSE), dado por la fórmula

$$\mathcal{L}(\phi) = \frac{1}{N} \sum_{i=1}^N \|f[\mathbf{x}_i, \phi] - \hat{\mathbf{y}}_i\|_2^2 = \frac{1}{N} \sum_{i=1}^N \|\mathbf{y}_i - \mathbf{x}_i\|_2^2 \quad (\text{en } \textit{autoencoders}), \quad (3.62)$$

donde  $\mathbf{y}_i$  es la salida de la red para la entrada  $\mathbf{x}_i$  y  $\hat{\mathbf{y}}_i$  es el objetivo.

Para minimizar la función de pérdida  $\mathcal{L}(\phi)$ , se utiliza el método de descenso del gradiente (GD), que ajusta iterativamente los parámetros  $\phi$  en la dirección opuesta al gradiente de la pérdida con respecto a estos parámetros. El gradiente  $\nabla_\phi \mathcal{L}(\phi)$  indica cómo cambiar los parámetros  $\phi$  para disminuir la pérdida más rápidamente. La regla de actualización para el descenso del gradiente estándar es

$$\phi \leftarrow \phi - \eta \nabla_\phi \mathcal{L}(\phi), \quad (3.63)$$

donde  $\eta$  es la tasa de aprendizaje, un hiperparámetro que determina el tamaño del paso de actualización.

En la práctica, calcular el gradiente sobre todo el conjunto de datos puede ser computacionalmente costoso, o incluso puede terminar en un mínimo local. En lugar de eso, se utiliza el *descenso del gradiente estocástico* (SGD) por *mini-batches* que trata de remediar este problema agregando ruido al gradiente en cada paso. El mecanismo por el que se agrega ese ruido es actualizando los parámetros a partir del gradiente de un subconjunto aleatorio de datos (*mini-batch*). De esta manera, en cada iteración del entrenamiento los parámetros se actualizan de la siguiente manera

$$\phi \leftarrow \phi - \eta \nabla_\phi \mathcal{L}(\phi; \mathbf{x}_{i:i+n}, \hat{\mathbf{y}}_{i:i+n}), \quad (3.64)$$

donde  $\mathcal{L}(\phi; \mathbf{x}_{i:i+n}, \hat{\mathbf{y}}_{i:i+n})$  es la pérdida para el *mini-batch*  $\mathbf{x}_{i:i+n}$  de tamaño  $n$ , y objetivos  $\hat{\mathbf{y}}_{i:i+n}$ . Los *mini-batches* generalmente se extraen del conjunto de datos sin reemplazo. El algoritmo los procesa uno por uno, actualizando los pesos de la red después de cada uno.

Una vez que se han procesado todos los *mini-batches*, cubriendo la totalidad del conjunto de datos, se dice que se ha completado una época de entrenamiento. En ese momento, se extraen nuevamente del conjunto completo de forma aleatoria y sin reemplazo, dando inicio a la siguiente época.

## Backpropagation

El algoritmo fundamental en el descenso del gradiente es la retropropagación (*backpropagation*), que calcula eficientemente los gradientes de la función de pérdida con respecto a los parámetros de la red en cada iteración. La idea principal es aplicar la regla de la cadena para propagar el error desde la salida de la red hacia las capas anteriores. Consta de dos pasos: un *forward pass* donde la entrada se propaga a través de la red para obtener la salida, y un *backward pass* donde el error se propaga hacia atrás desde la salida para calcular los gradientes.

En el *forward pass*, la entrada se propaga a través de la red para calcular la salida  $\mathbf{y}$ . Las activaciones de cada capa intermedia se almacenan para su uso en el *backward pass*, donde se calcula el gradiente de la pérdida con respecto a las activaciones y los pesos de cada capa utilizando la regla de la cadena. El gradiente se calcula mediante la siguiente ecuación

$$\delta^L = \nabla_{\mathbf{y}} \mathcal{L} \odot \mathbf{a}'(\mathbf{z}^L), \quad (3.65)$$

donde  $\delta^L$  es el gradiente de la pérdida en la capa de salida  $L$ ,  $\mathbf{a}'$  es la derivada de la función de activación, y  $\odot$  denota el producto de Hadamard (producto elemento a elemento). Para las capas anteriores, los gradientes se propagan de la siguiente manera

$$\delta^l = (\boldsymbol{\Omega}^{l+1})^T \delta^{l+1} \odot \mathbf{a}'(\mathbf{z}^l). \quad (3.66)$$

Finalmente, los gradientes con respecto a los pesos y los sesgos se calculan como

$$\nabla_{\phi} \mathcal{L} = \delta^l \mathbf{h}^{l-1}, \quad (3.67)$$

donde  $\mathbf{h}^{l-1}$  son las activaciones de la capa anterior.

## Optimizador de Adam

El descenso del gradiente estándar con una tasa de aprendizaje fija puede converger lentamente y ser inestable. El optimizador Adam (Adaptive Moment Estimation) [36], mejora la convergencia adaptando la tasa de aprendizaje para cada peso de forma individual. El optimizador Adam mantiene una estimación adaptativa de primer y segundo

momento de los gradientes y ajusta dinámicamente las tasas de aprendizaje para cada parámetro. Esto permite un entrenamiento más estable y rápido en comparación con el SGD estándar. El estimador de segundo momento  $\hat{v}_t$  es una estimación del cuadrado de los gradientes recientes promediados. Esto ayuda a suavizar las actualizaciones y evitar oscilaciones excesivas, escalando adaptativamente la tasa de aprendizaje. La regla de actualización de Adam se define como:

$$\phi_t \leftarrow \phi_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} \quad (3.68)$$

donde  $\phi_t$  son los parámetros en el paso  $t$ ,  $\eta$  es la tasa de aprendizaje,  $\hat{m}_t$  es el estimador de primer momento corregido de sesgo,  $\hat{v}_t$  es el estimador de segundo momento corregido de sesgo y  $\epsilon$  es un pequeño valor para evitar dividir por 0.

En este trabajo se utiliza el optimizador de Adamax [36], una variante del Adam que modifica la actualización de la segunda tasa de momento.

### 3.7.4. Autoencoder

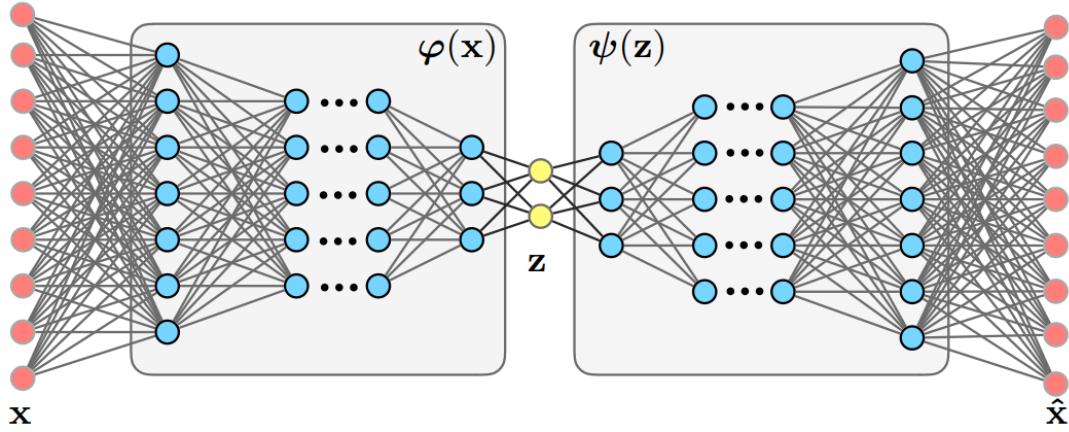
Un AE es una ANN entrenada para copiar su entrada a su salida. La arquitectura de un AE típicamente consiste en una capa de entrada, una o más capas ocultas que comprimen la entrada a una representación de menor dimensionalidad, y una o más capas ocultas que reconstruyen la entrada original a partir de esta representación comprimida. La capa de salida tiene la misma dimensión que la capa de entrada. La arquitectura de un AE se compone por dos partes:

- **Encoder:** representado como la función  $\varphi(\mathbf{x})$  que transforma la entrada  $\mathbf{x}$  a una representación de dimensión reducida, conocida como la *bottleneck*  $\mathbf{z}$ .
- **Decoder:** representado como la función  $\psi(\mathbf{z})$  que toma la representación latente  $\mathbf{z}$  y la mapea de vuelta a la dimensión original, reconstruyendo una aproximación  $\hat{\mathbf{x}}$  de la entrada original  $\mathbf{x}$ .

Si bien no es un requisito, es común que el *encoder* y el *decoder* tengan estructuras idénticas inversas, como el AE de la Fig. 3.16.

Volviendo a la RD del campo de velocidades del flujo alrededor de un cilindro, se considera la matriz (estandarizada) de la velocidad  $\hat{\mathbf{X}} \in \mathbb{R}^{(n \times m)}$ , donde  $n$  es la dimensión espacial y  $m$  la temporal. Si se desea reducir la dimensión espacial, entonces la capa de entrada de la red tendrá  $n$  neuronas. El espacio latente queda determinado por la *bottleneck* de  $r$  neuronas, descrito por la transformación

$$\mathbf{Y} \in \mathbb{R}^{r \times m} = \varphi(\hat{\mathbf{X}}), \quad (3.69)$$



**Figura 3.16:** Arquitectura de un AE donde el *encoder* y el *decoder* comparten una estructura idéntica e inversa. Figura tomada de la Ref. [37].

donde  $r$  es la dimensión reducida y  $\varphi$  la función que representa al *encoder*. En esta notación,  $\varphi(\hat{\mathbf{X}})$  representa la transformación de cada columna de  $\hat{\mathbf{X}}$  en un vector columna en  $\mathbb{R}^r$ , concatenados horizontalmente en una matriz  $\mathbf{Y}$ .

Si se desea reducir la dimensión temporal, los datos de entrada serán las filas  $\chi_i$  de  $\hat{\mathbf{X}}$ , y la capa de entrada de la red tendrá  $m$  neuronas. Para obtener modos espaciales interpretables, se construye  $\mathbf{Y} \in \mathbb{R}^{n \times r}$  concatenando ordenadamente las transformaciones  $\varphi(\hat{\chi}_i)$  para  $i = 1, \dots, n$ <sup>11</sup>. Es decir, es importante que las transformaciones  $\varphi(\hat{\chi}_i)$  mantengan el orden original, para luego reconstruir correctamente los modos espaciales en la malla cartesiana ( $n_x \times n_y$ ).

Una ventaja de los AEs sobre los métodos del aprendizaje de variedades es que la reconstrucción de los datos es sencilla. Pues, consiste simplemente en pasar los datos reducidos por el *decoder* como expresa la siguiente ecuación

$$\hat{\mathbf{X}} \approx \psi(\mathbf{Y}) = \psi(\varphi(\hat{\mathbf{X}})). \quad (3.70)$$

### 3.8. Conjuntos de entrenamiento y prueba

En el desarrollo teórico de los métodos se considera un único conjunto de datos con el fin de simplificar la notación y facilitar el seguimiento. Si bien el análisis en dicho marco permite investigar la calidad de representación del conjunto particular de datos en su espacio latente, en términos prácticos no es lo ideal. Esto se debe a que en problemas reales, los flujos (en general, cualquier sistema físico) son sensibles a las condiciones iniciales (configuración, perturbaciones, etc.), siendo la probabilidad de obtener dos flujos

<sup>11</sup>Notar que esta concatenación de columnas da lugar a una matriz en  $\mathbb{R}^{r \times n}$ , por lo que se debe transponer para efectivamente conseguir  $\mathbf{Y} \in \mathbb{R}^{n \times r}$ .

idénticos en distintas instancias prácticamente nula. Por lo tanto, es necesario evaluar la capacidad de generalización de los métodos.

Por ejemplo, en un sistema de ingeniería en el que se desea predecir o controlar un flujo a partir de mediciones en tiempo real, los datos presentarán cierta variabilidad. En tales circunstancias, no sería óptimo aplicar todos los pasos del algoritmo cada vez que se obtiene una nueva medición (en ciertos casos, puede ser muy costoso en tiempo de cómputo). En cambio, lo adecuado sería entrenar el modelo a priori con un conjunto de datos representativo, y luego lo único que restaría hacer es proyectar las nuevas mediciones al espacio latente para realizar las operaciones de interés.

De esta manera, para evaluar el rendimiento de los métodos, se dividen los datos en dos conjuntos, uno de entrenamiento  $\hat{\mathbf{X}}_{\text{train}}$  y otro de prueba  $\hat{\mathbf{X}}_{\text{test}}$ . El conjunto de entrenamiento se utiliza para ajustar los parámetros del modelo, mientras que el conjunto de prueba se emplea para evaluar el rendimiento del modelo y su capacidad de generalizar. En este contexto, la implementación de los métodos es exactamente la presentada, con la salvedad de que se debe prestar atención a qué conjunto de datos se utilizan en cada paso. Las coordenadas del espacio latente se obtienen a partir de los datos de entrenamiento, mientras que la transformación al espacio latente y la reconstrucción involucran ambos conjuntos. En la Tab. 3.1 se presenta un resumen de los métodos de RD y su aplicación, dado los conjuntos de entrenamiento y prueba.

Método	Coordenadas	Transformación	Reconstrucción
<b>PCA</b>	CPs de $\hat{\mathbf{X}}_{\text{train}}$	$\hat{\mathbf{X}}_{\text{test}} \mathbf{V}_{\text{train}}$	$\hat{\mathbf{X}}_{\text{test}} \mathbf{V}_{\text{train}} \mathbf{V}_{\text{train}}^{\top}$
<b>KPCA</b>	Autovalores y auto-vectores $\tilde{\mathcal{K}}_{\text{train}}$	$\mathbf{W}_{\text{train}}^{\top} \tilde{\mathcal{K}}_{\text{test-train}}$	Aprendizaje de pre-imagen de $\hat{\mathbf{X}}_{\text{test}}$ , proyectando en CPs de $\tilde{\mathcal{K}}_{\text{train}}$
<b>LLE</b>	Pesos $\mathbf{W}$ de $\hat{\mathbf{X}}_{\text{train}}$	Cálculo de $\mathbf{Y}_{\text{test}}$ con $\mathbf{W}_{\text{train}}$	Aproximación lineal a partir de $\mathbf{Y}_{\text{test}}$ y $\mathbf{W}_{\text{train}}^*$
<b>ISOMAP</b>	Autovalores y auto-vectores $\mathbf{K}$ de $\hat{\mathbf{X}}_{\text{train}}$	$\mathbf{V}_{\text{train}}^{\top} \mathbf{K}_{\text{train-test}}$	Idem LLE
<b>AE</b>	<i>Bottleneck</i> de la red entrenada con $\hat{\mathbf{X}}_{\text{train}}$	$\varphi(\hat{\mathbf{X}}_{\text{test}})$	$\psi(\varphi(\hat{\mathbf{X}}_{\text{test}}))$

**Tabla 3.1:** Algoritmos de RD y su aplicación, dado los conjuntos de entrenamiento  $\hat{\mathbf{X}}_{\text{train}}$  y prueba  $\hat{\mathbf{X}}_{\text{test}}$ .

Tanto para KPCA como para LLE, los elementos de la matriz de *kernel* entre los datos de prueba y entrenamiento son  $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ , donde  $\mathbf{x}_i \in \hat{\mathbf{X}}_{\text{test}}$  y  $\mathbf{x}_j \in \hat{\mathbf{X}}_{\text{train}}$ .



# Capítulo 4

## Resultados

*“¡Fui un estúpido confiando en la luz!”*

— Arthas (Caballero de la Muerte)

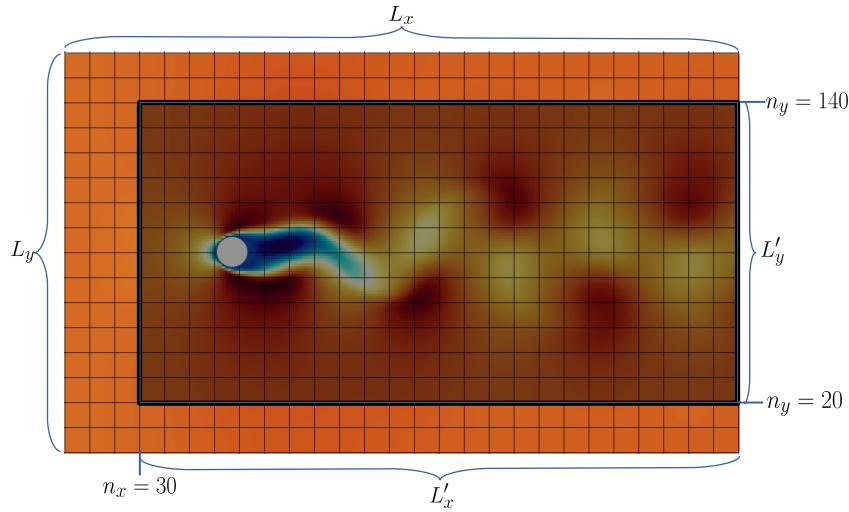
En este capítulo se presentan los resultados de la aplicación de algoritmos de RD a las simulaciones DNS 2D del flujo alrededor de un cilindro. Primero, se describe la base de datos, el preprocessamiento y la métrica de rendimiento. A continuación, se exponen los resultados obtenidos con PCA, el enfoque lineal estándar en la RD. Se realiza un análisis exhaustivo, ya que este método se utiliza como referencia para comparar con las técnicas no-lineales. Posteriormente, se presentan los resultados de técnicas de aprendizaje de variedades (KPCA, LLE e Isomap), realizando un análisis paramétrico de cada una, y de los *autoencoders*. Por último, se comparan los resultados de todos los algoritmos, realizando un análisis cuantitativo y cualitativo. Se discuten las ventajas y limitaciones de cada uno.

### 4.1. Metodología computacional

#### 4.1.1. Base de datos y preprocessamiento

La base de datos utilizada en este trabajo consiste en instantáneas 2D del módulo de la velocidad del flujo alrededor de un cilindro, obtenidas en una malla cartesiana en un dominio rectangular. La malla utilizada consta de  $n_x = 257$  nodos en la dirección horizontal y  $n_y = 160$  nodos en la dirección vertical, en un dominio de dimensiones  $L_x = 20$  y  $L_y = 12$ . A partir de simulaciones realizadas con el código Xcompact3d, se extraen las instantáneas correspondientes a las componentes horizontal  $u_x$  y vertical  $u_y$  de la velocidad, para un rango de tiempo adimensional  $300 \leq t \leq 500$ , con un paso  $\Delta t = 0,1$ . Luego, se toma el módulo de la velocidad  $u$  como  $u = \sqrt{u_x^2 + u_y^2}$ . Hasta aquí, contamos con

instantáneas de  $n_x \times n_y = 41,120$  puntos. Durante una etapa de prueba, se comprobó que este tamaño implica una demanda muy alta de recursos computacionales, especialmente en la RD temporal ( $>100$  GB de RAM y hasta 4 o 5 veces el tiempo de cómputo). Por esta razón, para cada instantánea se decidió recortar el dominio, eliminando las primeras 30 columnas (desde la izquierda, puntos en la dirección horizontal) y las primeras y últimas 20 filas (puntos en la dirección vertical). El dominio recortado se ilustra en la Fig. 4.1.



**Figura 4.1:** Dominio recortado ( $L'_x, L'_y$ ) de la simulación del flujo alrededor de un cilindro. Los puntos dentro del cilindro se excluyen de los datos.

El código de Xcompact3d utiliza el *Immersed Boundary Method* [38] para la resolución del flujo alrededor de un cilindro. Esto implica que los puntos dentro del cilindro (136 en total) son parte de la malla<sup>1</sup>, por lo tanto, se los excluye. De esta manera, la dimensión espacial de las instantáneas es de  $n = 227 \times 120 - 136 = 27,104$  puntos espaciales.

Se realizaron simulaciones para números de Reynolds 50, 100, 150 y 200, generando así cuatro conjuntos de datos separados. Cada uno de estos conjuntos se divide en dos subconjuntos de  $m = 1,000$  instantáneas cada uno. Un subconjunto se utiliza para ajustar el algoritmo ( $\mathbf{X}_{\text{train}}$ ), mientras que el otro se emplea para evaluar su rendimiento en datos nuevos ( $\mathbf{X}_{\text{test}}$ ). Por lo tanto, el tamaño de ambas matrices es  $(n \times m) = (27,104 \times 1,000)$ .

La división del conjunto se realiza tomando las primeras 1,000 instantáneas para el conjunto de entrenamiento y las 1,000 instantáneas restantes para el conjunto de prueba. Este método de división no aleatorio y ordenado es adecuado para datos de series temporales, que se caracterizan por la correlación entre observaciones cercanas en el tiempo (instantáneas consecutivas). Es importante aclarar esto, ya que en la práctica es común optar por una división aleatoria del conjunto de datos<sup>2</sup>. Sin embargo, al aplicar este tipo

<sup>1</sup>Si bien tienen valores de velocidad muy pequeños, no son nulos.

<sup>2</sup>Por ejemplo, seleccionar 1,000 instantáneas de manera aleatoria de un conjunto de 2,000 para el conjunto de entrenamiento y asignar el resto al conjunto de prueba.

de división aleatoria se obtuvieron resultados inferiores e inconsistentes, con variaciones significativas entre diferentes ejecuciones. Esto se debe a la naturaleza continua del flujo en el tiempo, donde la secuencia temporal de las instantáneas es crucial. En este caso, la división no aleatoria garantiza que las instantáneas consecutivas permanezcan juntas en cada conjunto.

Por último, se estandarizan los datos de la forma descrita en el apartado 3.3.1, restando la media temporal y dividiendo por la desviación estándar temporal de cada punto espacial. Las matrices estandarizadas se denotan como  $\hat{\mathbf{X}}_{\text{train}}$  y  $\hat{\mathbf{X}}_{\text{test}}$ . Esto se aplica a todos los métodos, excepto para el KPCA, lo cual se discutirá en su sección correspondiente.

### 4.1.2. Métrica de rendimiento

Para evaluar la calidad de la representación del problema en el espacio latente, se considera la reconstrucción al espacio original, denotada como  $\mathbf{X}_r$ . Esta matriz  $\mathbf{X}_r$  es obtenida a partir de la proyección de los datos de dimensión reducida nuevamente al espacio original. Por lo tanto, se define como métrica de rendimiento del algoritmo el error relativo de reconstrucción:

$$\varepsilon_r(\mathbf{X}, \mathbf{X}_r) = \frac{\|\mathbf{X} - \mathbf{X}_r\|_F}{\|\mathbf{X}\|_F}, \quad (4.1)$$

donde  $\mathbf{X} \in \mathbb{R}^{n \times m}$  es la matriz de datos original, y  $\mathbf{X}_r \in \mathbb{R}^{n \times m}$  es la matriz de datos reconstruida a partir del subespacio reducido. La norma de Frobenius  $\|\cdot\|_F$  se define para una matriz  $\mathbf{A} \in \mathbb{R}^{n \times m}$  mediante la ecuación

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^m |a_{ij}|^2}. \quad (4.2)$$

Esta métrica  $\varepsilon_r$  cuantifica la diferencia relativa entre la matriz original  $\mathbf{X}$  y su reconstrucción  $\mathbf{X}_r$ , proporcionando una medida de cuán bien el algoritmo logra preservar la información en la representación latente.

### 4.1.3. Herramientas computacionales

Para la implementación de los algoritmos de RD, se utilizó el lenguaje de programación Python. El PCA se realizó de dos formas: primero, implementando la SVD con `NumPy`, y segundo, utilizando directamente el objeto PCA de `Scikit-learn`, obteniendo resultados equivalentes. Los métodos de aprendizaje de variedades se implementaron utilizando el

módulo de *Manifold Learning* de *Scikit-learn* y para los *autoencoders* se utilizó la librería PyTorch.

Los autoencoders se entrenaron en una estación de trabajo con un procesador *Intel(R) Xeon(R) CPU E5-2670 v3 @2.30GHz* de 24 núcleos y 48 hilos, y 128 GB de RAM. El resto de los algoritmos se ejecutaron en una estación de trabajo con un procesador *Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz* de 8 núcleos y 16 hilos, y 128 GB de RAM.

El código desarrollado en este trabajo se encuentra disponible en el repositorio de GitHub: [https://github.com/martinmw13/ML\\_CFD](https://github.com/martinmw13/ML_CFD).

## 4.2. PCA

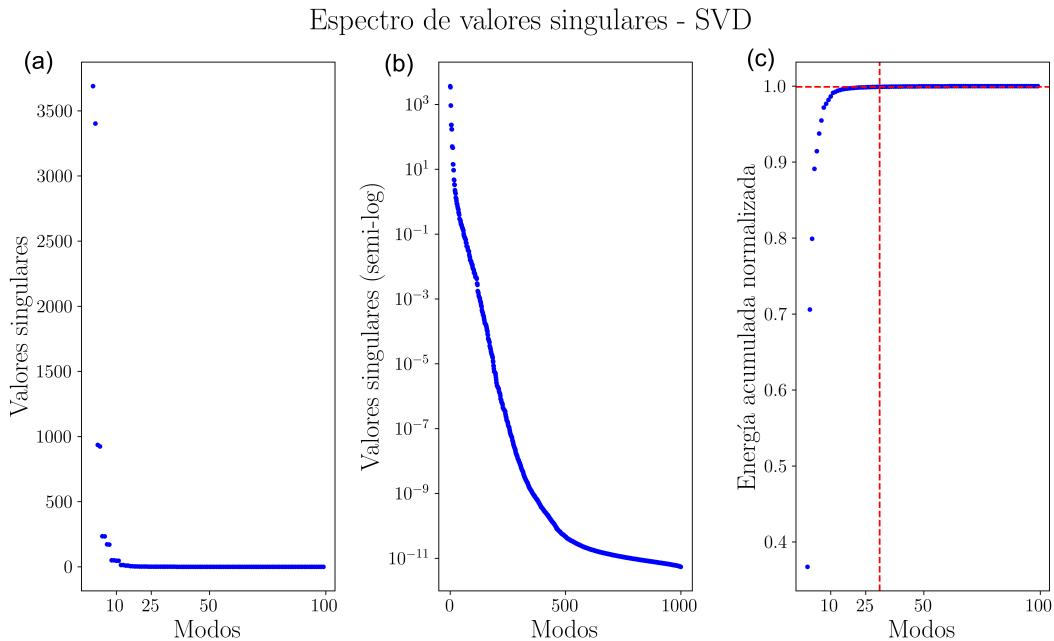
En primera instancia, para abordar la complejidad del flujo en términos de la varianza acumulada en sus modos principales, se realizó un análisis del espectro de valores propios. De esta manera, se realizó la SVD de la matriz  $\hat{\mathbf{X}}_{\text{train}}$  para el conjunto de datos de  $Re = 100$ . La implementación del algoritmo se llevó a cabo con el submódulo de álgebra lineal de la librería Numpy, consumiendo menos de un segundo de tiempo de cómputo.

Los resultados se visualizan en la Fig. 4.2, donde se hallan los valores singulares y la energía acumulativa normalizada en función de los modos. En escala normal (a), se observa que los valores singulares decaen rápidamente, alcanzando un *plateau* alrededor del modo 15. En escala semi-log (b), se observa en cambio un decaimiento pronunciado aproximadamente lineal (en escala semi-log) en los primeros 500 modos. Luego, siguen decayendo lentamente, en valores muy pequeños ( $< 10^{-10}$ ).

La energía acumulativa (definida en la Ec. (3.6)), es una medida de la cantidad de varianza capturada en función de los modos preservados en la reconstrucción. En tan solo 5 modos se alcanza más del 90 % de la energía, y en aproximadamente 30 el 99,9 %, indicado en la Fig. 4.2 (c) con líneas rojas punteadas. Esto significa que el comportamiento del flujo, comprendido por 1,000 instantáneas, puede representarse casi sin pérdida de información en tan solo unas decenas de modos.

Posteriormente, se implementó el método del análisis de componentes principales sobre el conjunto de datos  $\hat{\mathbf{X}}_{\text{train}}$  para los cuatro números de Reynolds. En la Fig. 4.3 se muestra el error relativo de reconstrucción de los datos de prueba  $\hat{\mathbf{X}}_{\text{test}}$  en función de la cantidad de modos preservados en la reconstrucción, para cada  $Re$ , en escala semi-log.

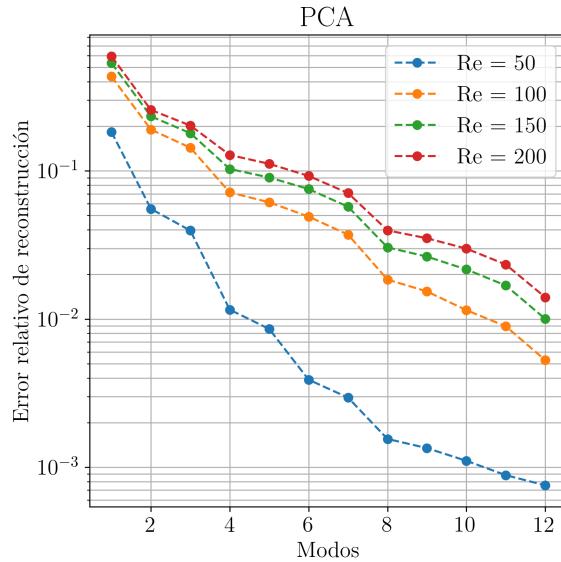
Se observa que el error relativo disminuye de manera estable (cercano a lineal en escala semi-log, i.e., exponencialmente) con la cantidad de modos, y crece con el  $Re$ . Las curvas de  $Re$  100, 150 y 200 presentan una forma casi idéntica, alcanzando valores cercanos al 0,01 en 12 modos. En contraste, la curva de  $Re$  50 se separa y cae más rápidamente, a un orden de magnitud menor al resto. Esto sugiere que la complejidad del flujo aumenta



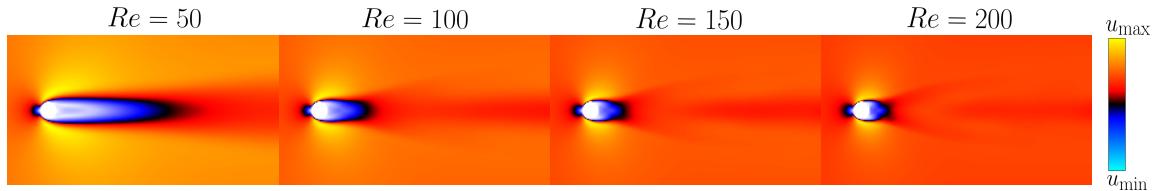
**Figura 4.2:** Espectro de valores propios obtenidos mediante la SVD de la matriz  $\mathbf{X}_{\text{train}}$ . (a) Valores singulares en escala normal y (b) en escala semi-log. (c) Energía acumulativa normalizada en función de los modos. Las líneas punteadas rojas indican el 99,9 % de la energía acumulativa, alcanzada en 31 modos.

con el  $Re$ , lo cual se refleja en la Fig. 2.5 con la presencia de vórtices más intensos a medida que  $Re$  aumenta. Además, a medida que el  $Re$  aumenta, el flujo se desplaza más rápido. Esto provoca que las instantáneas, tomadas a intervalos fijos de tiempo ( $\Delta t$ ), se distancien más entre sí en el espacio, implicando una mayor variabilidad respecto a menores  $Re$ . Por otro lado, a  $Re = 50$  el flujo es tan lento, que con un paso de tiempo de  $\Delta t = 0,1$  las instantáneas consecutivas son muy similares.

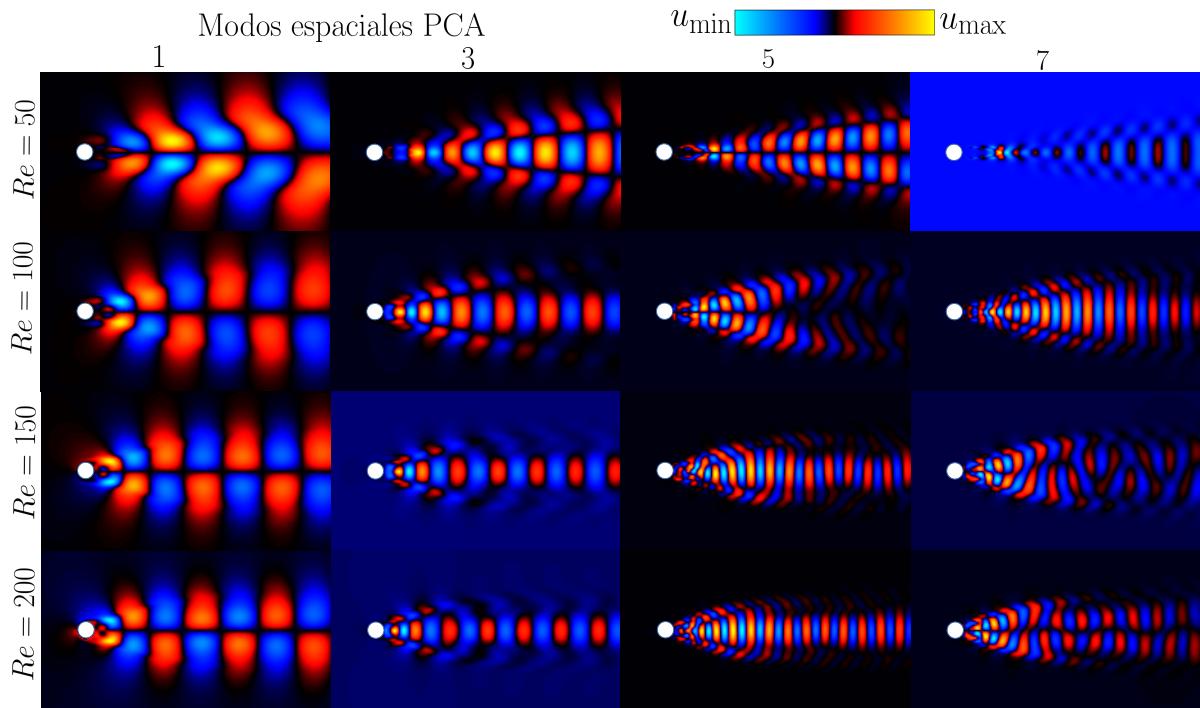
Una ventaja de la reducción temporal es que se pueden obtener modos espaciales interpretables. Es una característica única del PCA la obtención simultánea de modos espaciales y temporales, los cuales se corresponden, respectivamente, con las columnas de las matrices  $\mathbf{U}$  y  $\mathbf{V}$  resultantes de la SVD. En la Fig. 4.4 se muestra el flujo promedio y en la Fig. 4.5 cuatro modos espaciales representativos, para cada  $Re$ . Comúnmente, los modos vienen de a pares, con estructuras casi idénticas pero con signo cambiado. Por lo tanto, se visualizan modos con patrones distintivos (1, 3, 5 y 7). Si bien a cada modo le corresponde una escala de magnitud particular, con el fin de facilitar el análisis cualitativo, estas escalas se normalizan y se representa con una única barra de color.



**Figura 4.3:** Error relativo de reconstrucción en función de la cantidad de modos preservados en la reconstrucción de datos reducidos por PCA. Se grafican las curvas para cada  $Re$ .



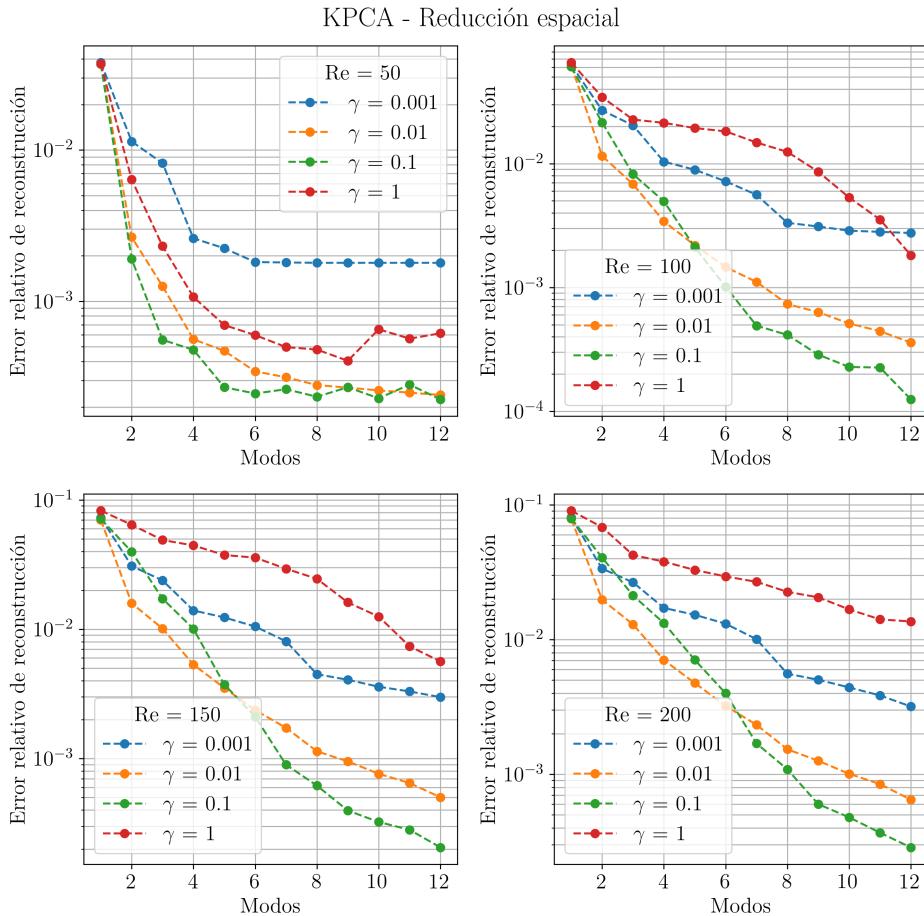
**Figura 4.4:** Flujo promedio alrededor de un cilindro de  $Re = 50, 100, 150$  y  $200$ .



**Figura 4.5:** Modos espaciales 1, 3, 5 y 7, para cada  $Re$ . La magnitud de la velocidad en cada modo se normaliza entre 0 y 1 para facilitar la visualización.

Se identifica la estructura subyacente del desprendimiento de vórtices, caracterizada por regiones alternadas de baja y alta velocidad, anti-simétricas respecto del plano horizontal medio. Los modos vienen en pares con signos opuestos, lo cual se corresponde con el comportamiento oscilatorio del flujo. En los modos de mayor orden, se distinguen regiones verticales alternadas más finas (ver zonas verticales azules y rojas en los modos 3, 5 y 7).

### 4.3. KPCA

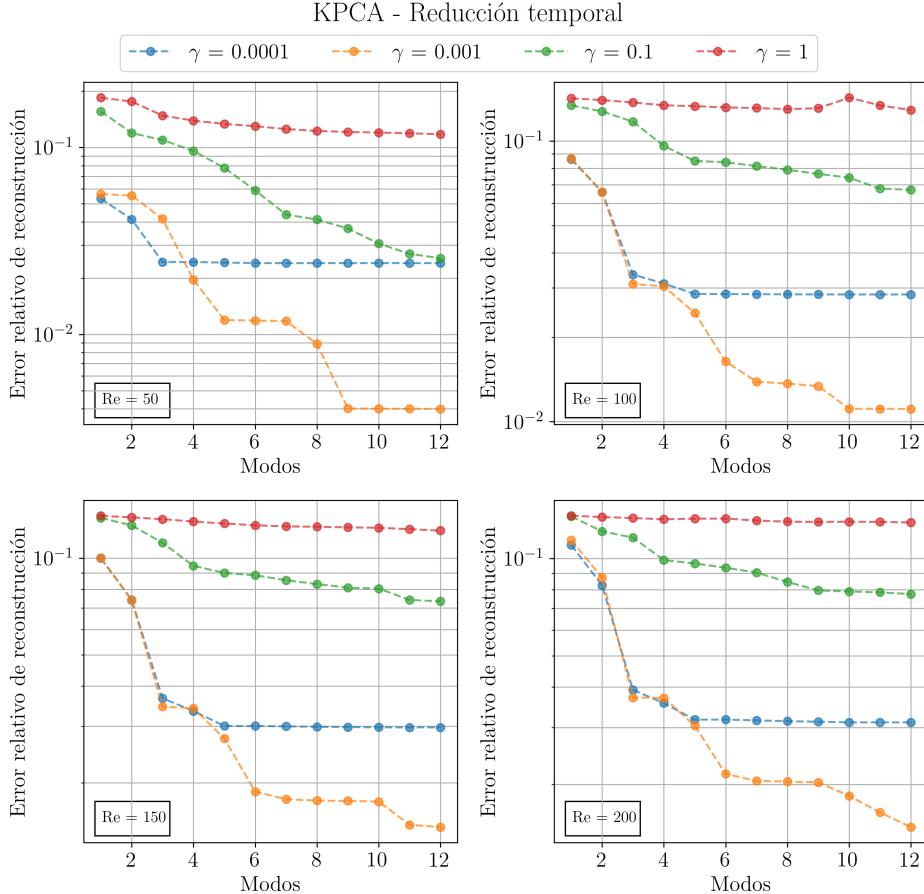


**Figura 4.6:** Error relativo de reconstrucción en función de la cantidad de modos preservados de la reducción espacial mediante KPCA. Se grafican las curvas para cada  $Re$  y distintos valores de  $\gamma$ .

El primer método no-lineal aplicado fue el KPCA. La implementación del algoritmo se llevó a cabo utilizando la librería `scikit-learn`. Cabe resaltar que, en contraste con el resto de los métodos, donde la estandarización es protocolar y necesaria para obtener buenos resultados, el algoritmo KPCA solamente presentó resultados aceptables cuando se lo aplicó al conjunto de datos original sin estandarizar ( $\mathbf{X}$  en la notación introducida en la Cap. 3). En cuanto al costo computacional, la reducción espacial requirió tiempos

menores a 1 segundo y ocupó menos de 3 GB de RAM. En contraste, la reducción temporal fue más demandante, requiriendo hasta 45 minutos de procesamiento y 20 GB de RAM.

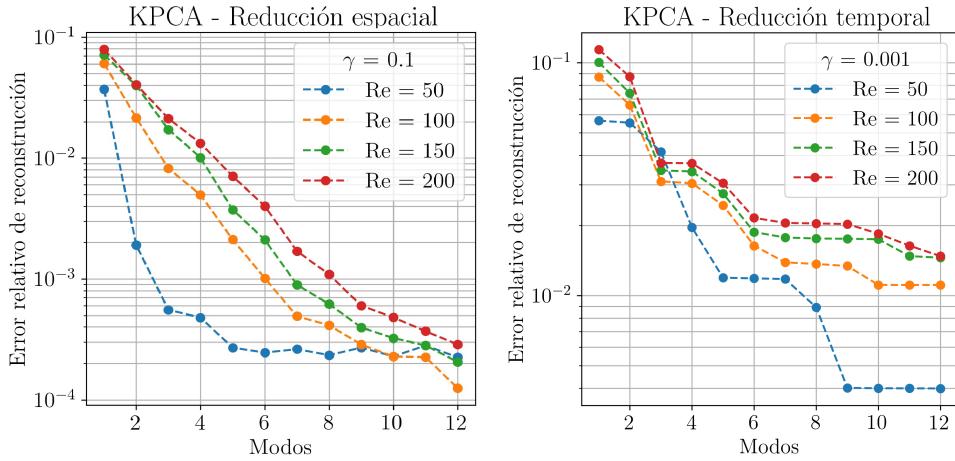
El algoritmo tiene dos hiperparámetros principales: la función de *kernel* y su parámetro característico. En este caso, se empleó la RBF (ver Ec. 3.30), cuyo parámetro libre es el  $\gamma$ , que representa una medida del ancho de la gaussiana. De esta manera, en primer lugar se realizó un análisis paramétrico en cada  $Re$ , variando el  $\gamma$ .



**Figura 4.7:** Error relativo de reconstrucción en función de la cantidad de modos preservados de la reducción temporal mediante KPCA. Se grafican las curvas para cada  $Re$  y distintos valores de  $\gamma$ .

En la Fig. 4.6 se muestra el error relativo de reconstrucción en función de la cantidad de modos preservados en la reconstrucción, para la **reducción espacial** mediante KPCA. Se grafican las curvas para distintos valores de  $\gamma$ , en cada  $Re$ . En todo el rango de  $Re$ , los valores de  $\gamma$  que dan los resultados más consistentes son 0,1 y 0,01. En los primeros 4 a 6 modos,  $\gamma = 0,01$  proporciona mejores resultados, pero para mayor número de modos,  $\gamma = 0,1$  da el menor error. Los otros dos extremos,  $\gamma = 0,001$  y  $\gamma = 1$ , son los menos efectivos en todo el rango de  $Re$ . Por lo tanto, se elige  $\gamma = 0,1$  como valor óptimo para análisis posterior.

En la Fig. 4.7 se muestra el error relativo de reconstrucción de los datos de prueba



**Figura 4.8:** Error relativo de reconstrucción en función de la cantidad de modos preservados de la reducción espacial y temporal mediante KPCA. Las curvas correspondientes a la reducción espacial y temporal se grafican para  $\gamma = 0,1$  y  $\gamma = 0,001$ , respectivamente.

$\mathbf{X}_{\text{test}}$  en función de la cantidad de modos preservados en la reconstrucción para la reducción temporal mediante KPCA. Se grafican las curvas para distintos valores de  $\gamma$ , en cada  $Re$ . En los primeros 5 modos, los resultados de  $\gamma = 0,001$  y  $\gamma = 0,0001$  son similares y los mejores. Luego,  $\gamma = 0,001$  se separa, obteniendo los mejores resultados, mientras que  $\gamma = 0,0001$  se mantiene constante. Por otro lado,  $\gamma = 1$  da los peores resultados, manteniéndose casi constante por encima de 0,1 en todos los modos y  $Re$ . Luego, se elige  $\gamma = 0,001$  como valor óptimo, el cual presenta un error de reconstrucción dos órdenes de magnitud menor al óptimo para la reducción espacial.

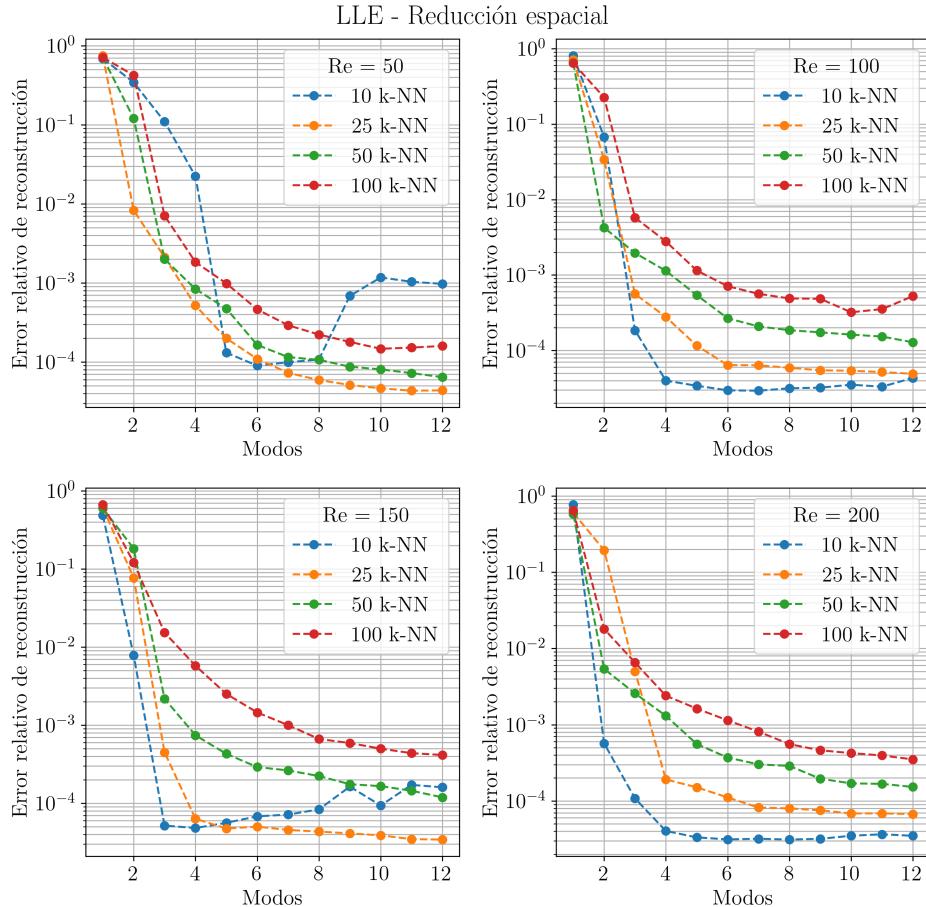
Para ver el comportamiento del error relativo de reconstrucción como función del  $Re$ , los mejores resultados ( $\gamma$ 's óptimos) se grafican en la Fig. 4.8, tanto para la reducción espacial como para la temporal mediante KPCA. Para la **reducción espacial** con  $\gamma = 0,1$ , se observa que el error relativo disminuye rápidamente en los primeros 3 modos para  $Re = 50$ , mientras que para el resto de los  $Re$  el error disminuye aproximadamente lineal en escala semi-log. En todos los casos, el error relativo alcanza valores cercanos al  $10^{-4}$  en 12 modos. El análisis de la **reducción temporal** es similar, con la diferencia de que el error relativo es significativamente mayor. Para  $Re = 50$  el error relativo es de  $4 \times 10^{-3}$  en 12 modos, mientras que para el resto de los  $Re$  el error relativo es aproximadamente de  $10^{-2}$ . Nuevamente, en todos los casos se aprecia que el error tiende a crecer con el  $Re$ .

## 4.4. LLE

El algoritmo LLE se implementó utilizando la librería `scikit-learn`. En cuanto al costo computacional, la reducción espacial requirió unos pocos segundos y ocupó menos de 3 GB de RAM. Por otro lado, la reducción temporal consumió hasta 50 minutos de

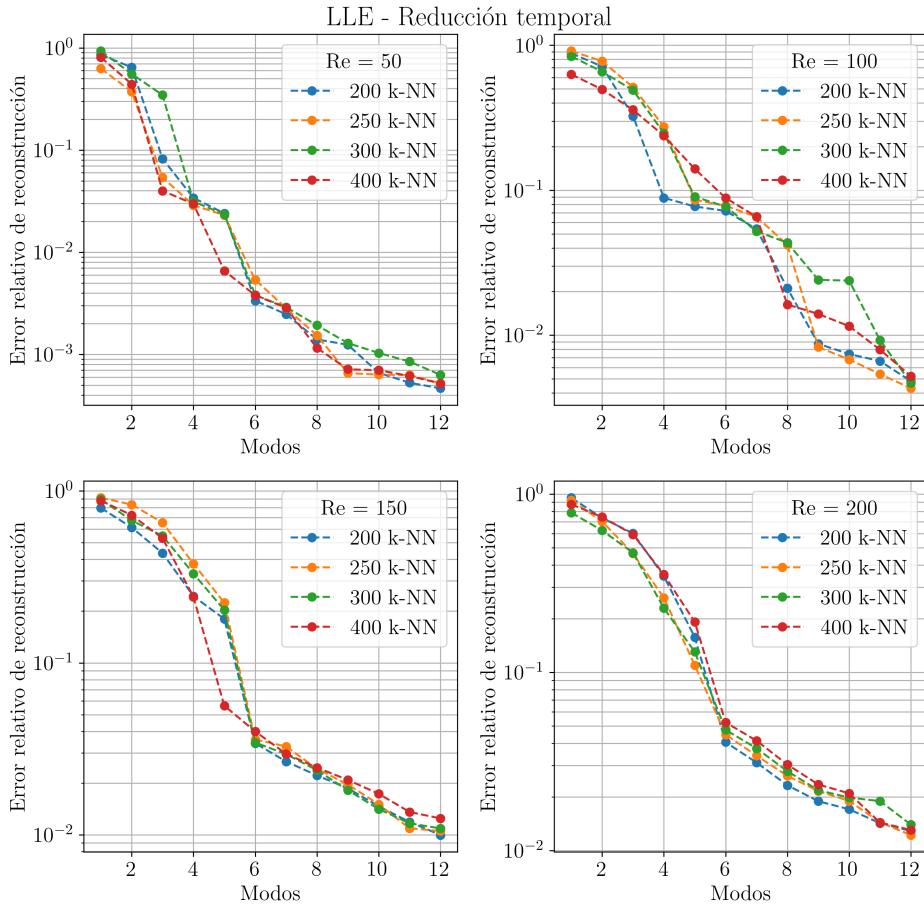
cómputo y 14 GB de RAM.

El algoritmo tiene un hiperparámetro libre, que es el número de vecinos más cercanos  $K$ . Como se discutió en el apartado 3.5.1, la elección del número de  $K$ -NN es un factor clave en el rendimiento del algoritmo. Un valor de  $K$  demasiado bajo puede resultar en una mala representación de la estructura local de los datos, mientras que un valor demasiado alto implica ponderar vecinos lejanos. En primer lugar, se realizó un análisis paramétrico en cada  $Re$ , variando el  $K$ .



**Figura 4.9:** Error relativo de reconstrucción en función de la cantidad de modos preservados de la reducción espacial mediante LLE. Se grafican las curvas para cada  $Re$  y distintos números de  $K$ -NN.

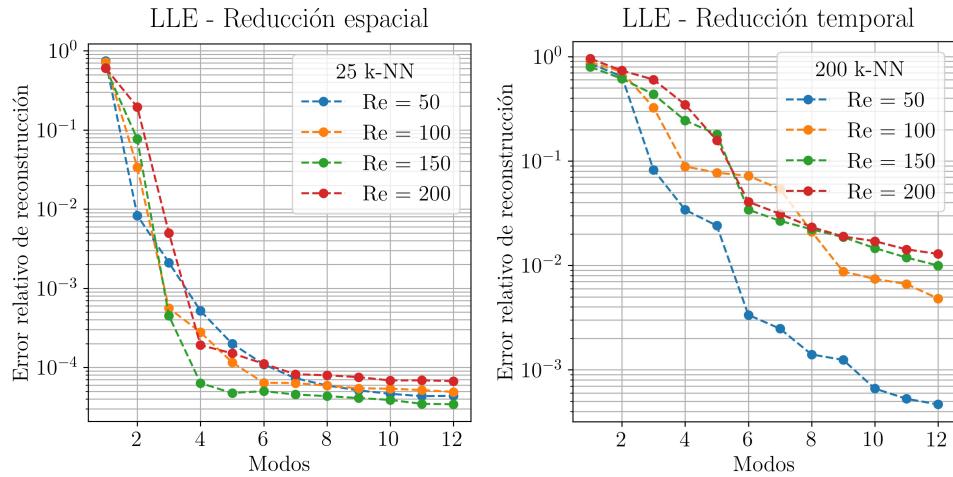
Para la **reducción espacial**, la Fig. 4.9 muestra el error relativo de reconstrucción en función de la cantidad de modos preservados. Se presentan curvas correspondientes a diferentes valores de  $K$ -NN para cada número de  $Re$ . Para 25, 50 y 100  $K$ -NN se observa un comportamiento similar en todo el rango de  $Re$ . El error relativo es mayor con un mayor número de  $K$ -NN a partir del cuarto modo. La curva para 10  $K$ -NN es menos consistente, presentando el menor error relativo en  $Re$  200 y 100, pero en  $Re$  50 y 150 el error aumenta significativamente en los últimos modos. Por lo tanto, se elige 25  $K$ -NN como valor óptimo para el análisis posterior.



**Figura 4.10:** Error relativo de reconstrucción en función de la cantidad de modos preservados de la reducción temporal mediante LLE. Se grafican las curvas para cada  $Re$  y distintos números de  $K$ -NN.

En la [Fig. 4.10](#) se muestra el mismo análisis para la **reducción temporal**. En este caso, todos los valores de  $K$ -NN dan resultados similares, con los valores extremos (200 y 400  $K$ -NN) rindiendo ligeramente mejor. Se elige  $K = 200$  como valor óptimo, dado que en general tiene mejor error relativo en los últimos modos y conlleva un menor costo computacional que usar  $K = 400$ .

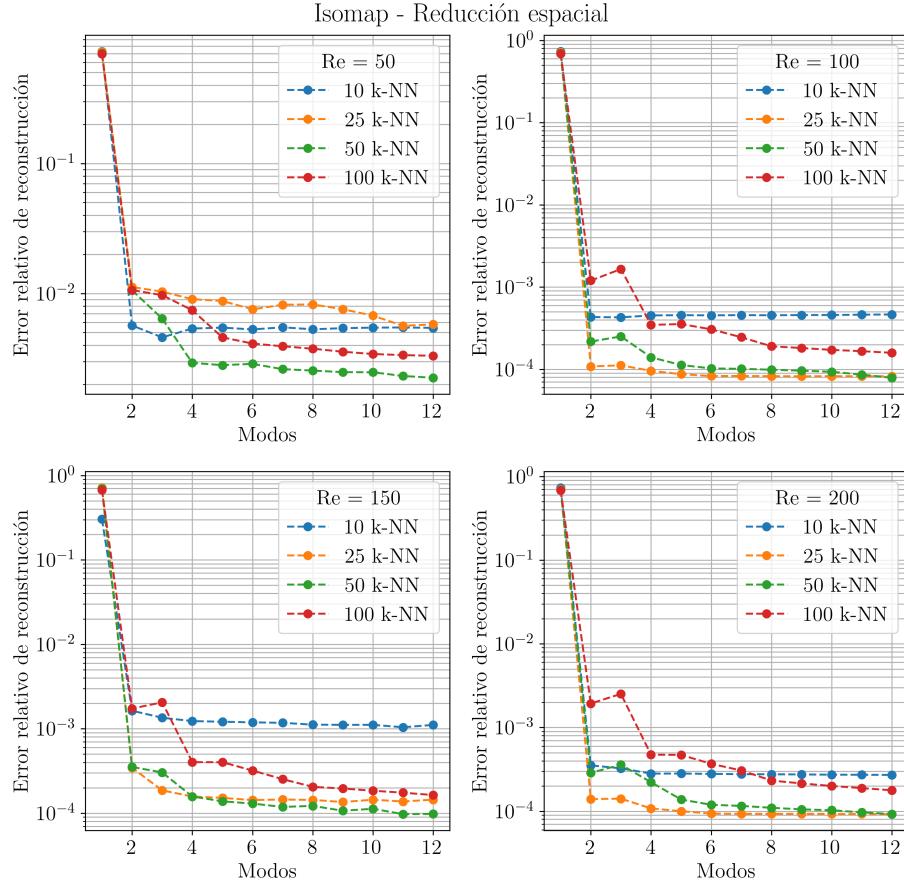
En la [Fig. 4.11](#) se muestra el error relativo de reconstrucción en función de la cantidad de modos preservados en la reconstrucción. Las curvas correspondientes a la reducción espacial usan  $K = 25$  vecinos más cercanos y para la reducción temporal  $K = 200$  vecinos. Para la reducción espacial, el error relativo de reconstrucción decrece rápidamente en los primeros 5 modos y luego se estabiliza, mostrando un comportamiento consistente en todo el rango de  $Re$ . En contraste, para la reducción temporal, el error relativo es mayor por dos órdenes de magnitud respecto del espacial, y el decrecimiento es gradual. El error tiende a crecer con el  $Re$ , salvo en los modos 6 y 7, donde la curva  $Re = 100$  cruza las curvas de  $Re = 150$  y  $200$ . Cabe notar la diferencia de  $K$ -NN necesarios en cada tipo de RD, cuya razón está principalmente en la diferencia de magnitud entre  $n$  y  $m$ .



**Figura 4.11:** Error relativo de reconstrucción en función de la cantidad de modos preservados, para cada  $Re$ . Las curvas de la reducción espacial corresponden a  $K = 25$  y de la reducción temporal a  $K = 200$ .

En la reducción espacial se tienen  $m$  muestras dispersas en un espacio de dimensión  $n$ , mientras que en la reducción temporal se tiene una densidad de muestras ( $n$ ) mucho mayor en un espacio de dimensión  $m$ . Por lo tanto, en el primer caso son suficiente menos K-NN para capturar la estructura local de los datos (si se tomaran, por ejemplo, 200 K-NN, se estaría considerando  $\frac{1}{4}$  del conjunto como K-NN, ponderando vecinos lejanos). En el segundo caso, al tener una densidad mayor de muestras en un espacio de menor dimensión, se necesitan más K-NN.

## 4.5. Isomap

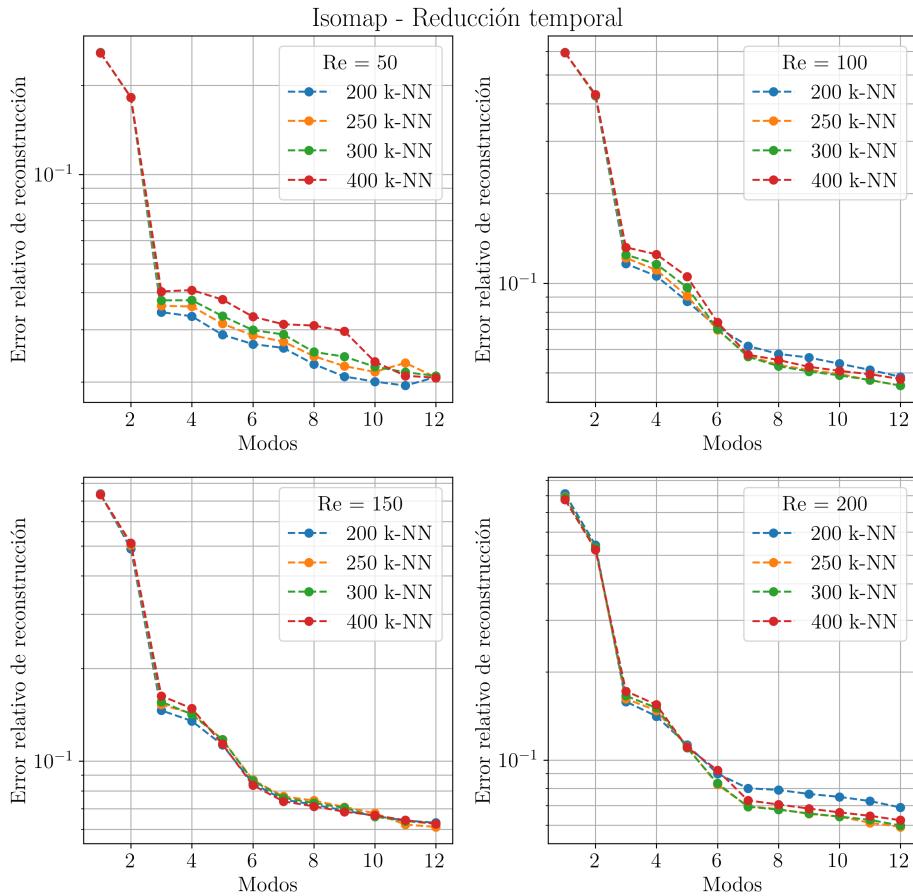


**Figura 4.12:** Error relativo de reconstrucción en función de la cantidad de modos preservados de la reducción espacial mediante Isomap. Se grafican las curvas para cada  $Re$  y distintos números de  $K$ -NN.

El algoritmo Isomap se implementó utilizando la librería `scikit-learn`. En cuanto al costo computacional, la reducción espacial requirió menos de 2 segundos y ocupó menos de 3 GB de RAM. Por otro lado, la reducción temporal consumió entre 1 y 3 horas de cómputo y 7.5 GB de RAM.

Al igual que el LLE, el hiperparámetro principal del algoritmo es el número de  $K$ -NN. Similarmente, se realiza un análisis paramétrico en cada  $Re$ , variando el  $K$ .

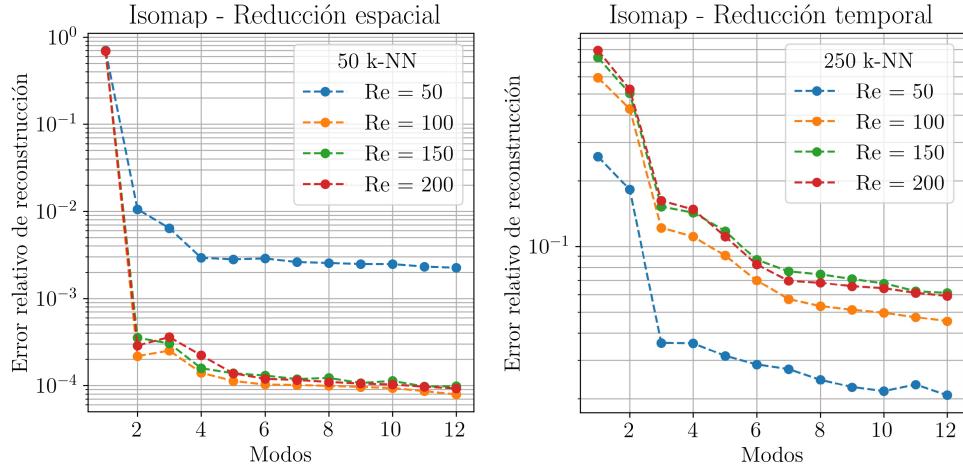
En la Fig. 4.12 se muestra para la la **reducción espacial** el error relativo de reconstrucción en función de la cantidad de modos preservados. Se presentan curvas correspondientes a diferentes valores de K-NN para cada número de  $Re$ . De manera similar a la reducción espacial con LLE, los dos valores extremos son los que peor rinden, y de los dos intermedios el que da mejores resultados es 50  $K$ -NN. En la Fig. 4.13 se muestran las curvas equivalentes de la **reducción temporal**. A diferencia del LLE, todos los valores de  $K$ -NN dan resultados similares y estables. En este caso, se elige  $K = 250$  como



**Figura 4.13:** Error relativo de reconstrucción en función de la cantidad de modos preservados de la reducción temporal mediante Isomap. Se grafican las curvas para cada  $Re$  y distintos números de  $K$ -NN.

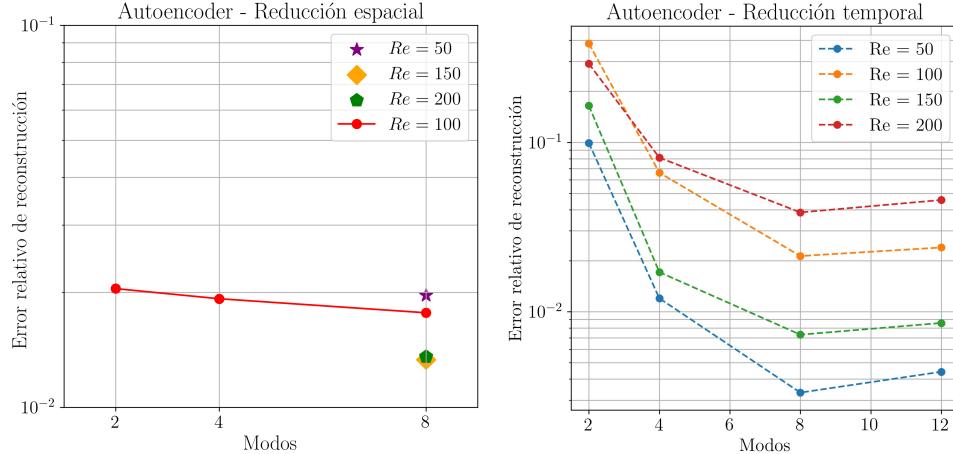
valor óptimo, aunque los resultados son casi idénticos para 300. Con el mismo criterio que en LLE, se elige el menor número de  $K$ -NN que mejor rinde, por cuestiones de costo computacional.

En la Fig. 4.14 se muestra el error relativo de reconstrucción en función de la cantidad de modos preservados, para cada  $Re$ . Las curvas correspondientes a la reducción espacial usan  $K = 25$  vecinos y para la reducción temporal  $K = 250$ . A diferencia de los métodos previos, Isomap exhibe en la reducción espacial una caída pronunciada del error de reconstrucción relativo en tan solo dos modos, permaneciendo constante a partir del cuarto. Se alcanzan valores tan pequeños como  $10^{-4}$  en 12 modos. No obstante, una diferencia con los resultados previos es que para  $Re = 50$  el error es mucho mayor que para el resto de  $Re$  ( $2 \times 10^{-3}$ ). En la reducción temporal, la caída se da en los primeros tres modos, a partir de los cuales decrece lentamente. Para  $Re = 50$ , el valor mínimo es de  $2 \times 10^{-2}$ , y para el resto alrededor de  $5 \times 10^{-2}$ .



**Figura 4.14:** Error relativo de reconstrucción en función de la cantidad de modos preservados, para cada  $Re$ . Las curvas de la reducción espacial corresponden a  $K = 50$  y de la reducción temporal a  $K = 200$ .

## 4.6. Autoencoder



**Figura 4.15:** Error relativo de reconstrucción en función de la cantidad de modos preservados, para la reducción espacial y temporal mediante *autoencoders*. Las curvas correspondientes a la reducción espacial se grafican para 2, 4 y 8 modos, y para la reducción temporal para 2, 4, 8 y 12 modos.

Los *autoencoders* se implementaron utilizando la librería PyTorch. En términos de costo computacional, a diferencia de los métodos anteriores, el AE demandó un tiempo de procesamiento considerablemente mayor en el caso espacial. Específicamente, la reducción espacial consumió hasta 12 GB de RAM y tardó aproximadamente 38 horas en completarse. Por otro lado, la reducción temporal requirió un menor uso de recursos, con 2 GB de RAM y un tiempo de ejecución de 4.5 horas.

Se emplearon redes totalmente conectadas. La función de activación ReLU se utilizó para imponer no-linealidad y se seleccionó el optimizador Adamax para actualizar los

parámetros. Se escogió una tasa de aprendizaje variable utilizando un *scheduler* de tasa de aprendizaje: tasa inicial fue de  $10^{-3}$ , y luego se redujo por un factor de 0.1 después de alcanzar  $\frac{1}{3}$  y  $\frac{2}{3}$  del número final de 9.000 épocas, lo que llevó a una tasa de aprendizaje final de  $10^{-5}$ . Estos criterios se adoptaron a partir de un trabajo de referencia en el que se obtuvieron buenos resultados con AEs [2]. La información sobre las estructuras de las redes entrenadas se halla en el Ap. A.1.

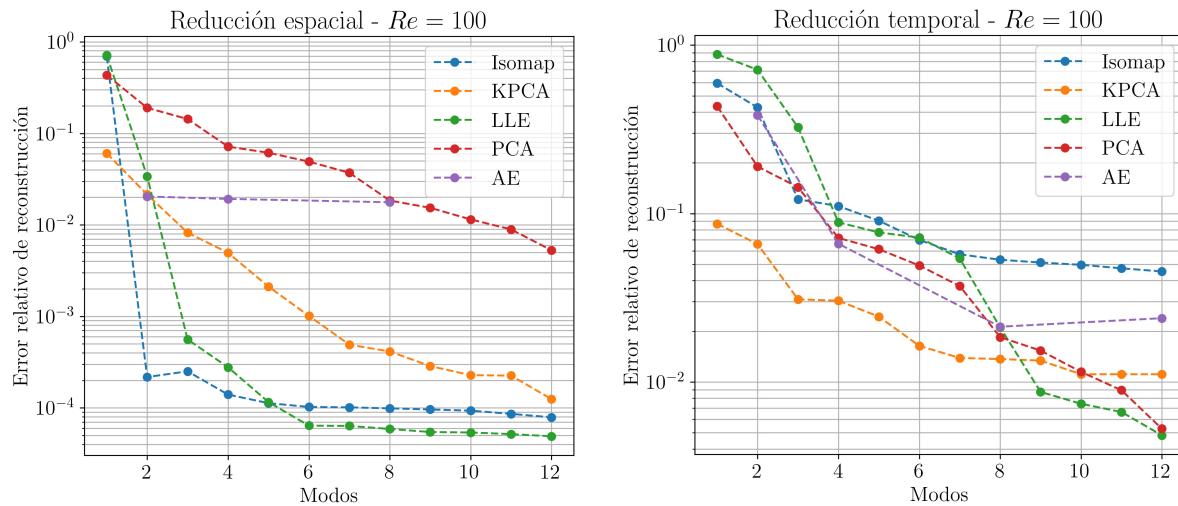
En la Fig. 4.15, se muestra el error relativo de reconstrucción en función de la cantidad de modos preservados (neuronas de la *bottleneck*), tanto para la reducción espacial como para la temporal. Por limitaciones de recursos computacionales, el análisis de la reducción espacial se restringió a 2, 4 y 6 modos en  $Re = 100$ , y únicamente se evaluó el error en todo el rango de  $Re$  para 8 modos. El error en la reducción espacial casi no varía, manteniéndose cerca de 0.01, y, contrariamente a los resultados previos, el error decrece con el  $Re$ . En cuanto a la reducción temporal, se consideraron 2, 4, 8 y 12 modos para todos los  $Re$ . El comportamiento es distinto, con el error disminuyendo en los primeros 8 modos, alcanzando un mínimo y aumentando levemente en 12 modos. En este caso, el error sí crece con el  $Re$ .

Una particularidad de los AE, es que los modos no están ordenados y vienen en un orden arbitrario; por ejemplo, el modo 1 no es necesariamente más significativo que el modo 4. Además, en el AE, cada configuración con diferentes tamaños de espacio latente resulta en modos diferentes. Los dos modos de un AE con un espacio latente de tamaño 2, podrían no ser los mismos que los dos primeros (o cualquier dos) modos de un AE con tamaños de espacio latente diferentes.

Por último, cabe destacar que el error relativo al haber pasado el conjunto de entrenamiento ( $\mathbf{X}_{\text{train}}$ ) por la red entrenada dio valores dos órdenes de magnitud menores ( $10^{-4}$ ) en reducción espacial, y de un orden menos en temporal ( $10^{-3}$ ). Esto sugiere que la red podría haberse sobre-ajustado. Sin embargo, esto podría ser beneficioso en el caso de que únicamente sea de interés la extracción de modos, y no la capacidad de reconstrucción general (e.g., para extrapolación). En el Ap. A.1 se encuentra la evolución de la función de pérdida durante las épocas de entrenamiento para cada red.

## 4.7. Comparación de métodos ( $Re=100$ )

En esta sección se compara el rendimiento de todos los métodos de RD estudiados, restringiéndose al flujo de  $Re = 100$ . Los resultados del AE corresponden a la arquitectura AE-S-8 (ver descripción en el Ap. A.1). En la Fig. 4.16 se presenta el error relativo de reconstrucción en función de la cantidad de modos preservados, para la reducción espacial y temporal. Cabe remarcar que el error relativo del PCA es exactamente el mismo ambos



**Figura 4.16:** Error relativo de reconstrucción en función de la cantidad de modos preservados, para cada método de RD. Para el caso AE, las curvas correspondientes a la reducción espacial se grafican para 2, 4 y 8 modos, y para la reducción temporal para 2, 4, 8 y 12 modos.

casos<sup>3</sup>, decreciendo aproximadamente lineal en escala semi-log, alcanzando un mínimo de  $0,5 \times 10^{-2}$  en 12 modos.

En el caso **espacial**, los métodos no-lineales presentan resultados superiores respecto al PCA. Las técnicas de aprendizaje de variedades (LLE, Isomap) presentan el mínimo error relativo, alcanzando valores de tres órdenes de magnitud menor al PCA en tan solo 3 modos. La curva del Isomap permanece prácticamente constante a partir del cuarto modo, mientras que LLE a partir del sexto. El KPCA provee la mejor calidad de reconstrucción usando un solo modo. Luego, decrece gradualmente y en 12 modos alcanza valores similares al LLE e Isomap. En 12 modos, estos tres métodos alcanzan un error relativo similar, cercano a  $10^{-4}$ , siendo LLE el mejor, seguido de Isomap y KPCA. En cuanto al AE, el rendimiento es distinto. El error relativo es prácticamente constante para 2, 4 y 8 modos, teniendo el peor rendimiento de los métodos no-lineales. Incluso, en 8 modos iguala al error del PCA.

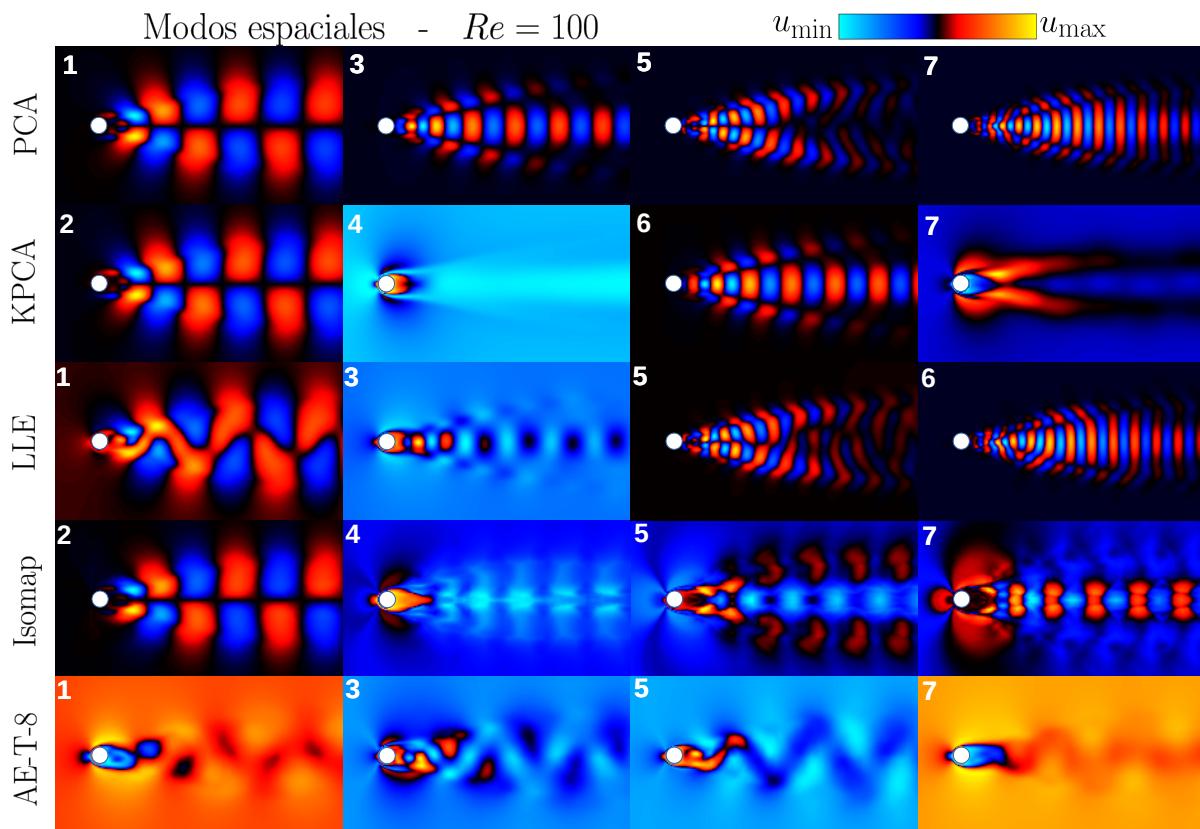
Los resultados de la **reducción temporal** difieren significativamente de la espacial. El KPCA muestra un error considerablemente menor que el resto en los primeros 8 modos, con decrecimiento despreciable a partir del 7. Al igual que en la reducción espacial, KPCA es el que mejor reconstruye el flujo en un único modo. Por otro lado, el LLE presenta el peor error en los primeros 3 modos y se mantiene por encima del PCA durante los primeros 8 modos. Sin embargo, a partir del 9 hasta el 12, LLE ofrece el mejor error, coincidiendo con el PCA en el modo 12 en un error de  $5 \times 10^{-3}$ . El Isomap es el que tiene el peor rendimiento a partir del sexto modo, llegando a un mínimo de un orden de

<sup>3</sup>La descomposición de la SVD es equivalente, con las matrices  $\mathbf{U}$  y  $\mathbf{V}$  intercambiadas de lugar. Por lo tanto, la reconstrucción es equivalente en ambos casos.

magnitud mayor al PCA. El AE, en 4 y 8 modos, coincide con el PCA, y termina siendo el segundo peor de todos los métodos en 12 modos ( $2 \times 10^{-2}$ ).

Una ventaja de la **reducción temporal** es la capacidad de extraer modos espaciales interpretables. La Fig. 4.17 exhibe cuatro modos representativos del flujo alrededor de un cilindro de  $Re = 100$ , para todos los métodos. Los modos suelen presentarse en pares con signos opuestos (o desfasados). Entonces, se omite la visualización de patrones idénticos desfasados. Por otro lado, el hecho de que el KPCA se aplicara al conjunto sin estandarizar, provocó que el primer modo resultara ser simplemente el flujo promedio, por lo que se omite. Respecto al AE, se muestran los modos de la red con 8 neuronas en la *bottleneck*(AE-S-8). Notar que los modos del AE no están ordenados jerárquicamente y que no necesariamente son los mismos que se extraen en otras estructuras (discutido en la Sec. 4.6). Por otro lado, si bien cada método tiene un rango de escala de color diferente, se normalizan a la misma escala para facilitar el enfoque cualitativo.

Se observan patrones similares en los distintos métodos. Por ejemplo, el primer modo de todos los métodos (excepto AE) muestra la estructura subyacente del desprendimiento de vórtices: regiones alternadas de alta y baja velocidad, antisimétricas a lo largo del plano horizontal medio. En algunos casos, los métodos extraen patrones casi idénticos, como los pares de modos (PCA-3, KPCA-6), (PCA-5, LLE-5) y (PCA-7, LLE-7). Estos muestran regiones verticales alternadas de velocidad más angostas que el primer modo. En cambio, los modos de AE no son simétricos inmediatamente detrás del cilindro. Aguas abajo, se observa un patrón de zig-zag de alta o baja velocidad según el modo. Por su parte, Isomap presenta las estructuras más complejas, con fronteras de alta y baja velocidad bien definidas. De manera interesante, KPCA presenta modos como KPCA-4 y KPCA-7 que no aportan información sobre el comportamiento oscilatorio.



**Figura 4.17:** Modos espaciales representativos de la reducción temporal del flujo alrededor de un cilindro de  $Re = 100$ . Se indica en blanco el orden del modo. La magnitud de la velocidad en cada caso se normaliza entre 0 y 1 para facilitar la visualización. Los modos del AE corresponden a la arquitectura AE-S-8 (descripción en el Ap. A.1) y no están ordenados por importancia.



# Capítulo 5

## Conclusiones

“(Suspiro) Ahh, ¡por fin!...”

— Arthas (Caballero de la Muerte)

### 5.1. Resumen y conclusiones

La dinámica de fluidos computacional (CFD) se caracteriza por producir datos espacio-temporales de alta dimensión. Por lo tanto, identificar un conjunto óptimo de coordenadas para representar los datos en un subespacio latente de baja dimensión es un primer paso hacia el desarrollo de modelos de orden reducido. El abordaje tradicional es mediante el Análisis de Componentes Principales (PCA), que da una aproximación lineal óptima. Sin embargo, en general los flujos son complejos e inherentemente no-lineales, que limitan la capacidad de representación en baja dimensión del PCA. En consecuencia, se han desarrollado varios algoritmos de reducción de dimensionalidad (RD) no-lineal, y recientemente se han comenzado a aplicar en el campo del CFD. En este contexto, en el presente trabajo se buscó implementar y comparar diferentes métodos de RD, lineal y no-lineales, en un problema canónico de flujo inestable: el flujo alrededor de un cilindro inmerso en un flujo uniforme. Las técnicas de RD estudiadas incluyeron el estándar lineal PCA, que actuó como referencia para comparar el rendimiento de los métodos no-lineales implementados: Kernel PCA (KPCA), Locally Linear Embedding (LLE), Isometric Mapping (Isomap) y *autoencoders* (AE).

Con el fin de obtener una base de datos para aplicar estos métodos, se realizaron simulaciones numéricas directas (DNS) con el software Xcompact3d del flujo alrededor del cilindro. Primero se validó la herramienta numérica simulando flujos canónicos y comparando con datos de referencia. Luego, se llevaron a cabo las simulaciones DNS 2D a diferentes números de Reynolds (50, 100, 150 y 200), constituyendo la base de datos

para el análisis.

Previo a la aplicación de los algoritmos de RD, se realizó un análisis espectral de la SVD de los datos, el cual brinda una medida de la complejidad del flujo. El mismo reveló que con solo 5 modos se captura más del 90 % de la varianza total, y con aproximadamente 30 modos se alcanza el 99.9 %. Esto sugiere que el comportamiento del flujo puede representarse con alta fidelidad utilizando un número reducido de modos.

A continuación, se aplicaron los métodos de RD al conjunto de datos, comenzando por el PCA, el cual establece valores de referencia. El mismo que el error de reconstrucción disminuye aproximadamente lineal en escala semi-logarítmica (i.e., exponencial) con la cantidad de modos, alcanzando valores cercanos a  $10^{-2}$  en 12 modos. Posteriormente, se realizó un análisis paramétrico en KPCA, LLE e Isomap para determinar los valores óptimos de los hiperparámetros correspondientes a cada método. En la reducción espacial, los métodos no-lineales lograron errores de reconstrucción mucho menores que PCA, especialmente LLE e Isomap que alcanzaron errores del orden de  $10^{-4}$  en pocos modos. Esto puede indicar que las relaciones no-lineales en los datos de flujo son críticas para capturar la estructura subyacente con mayor precisión. Por otro lado, el AE no logró superar al PCA luego de varios modos, lo que sugiere (por su éxito reciente en problemas similares [2, 34]) que el entrenamiento requiere de una configuración u optimización no explorada en este estudio. La reducción temporal de los métodos no-lineales resultó más compleja, obteniendo errores de hasta dos órdenes de magnitud mayor. Para lograr resultados razonables con LLE e Isomap, el número de vecinos más cercanos utilizado tuvo que ser significativamente mayor respecto a la RD espacial, lo cual se debe a que el número de muestras (evolución temporal en cada nodo) en este caso es mucho mayor ( $n$ ). El PCA no fue superado sino en los primeros modos por el KPCA, y por el LLE en los últimos. En general, tanto en la RD espacial como temporal se observó una tendencia del error relativo a crecer con el número de Reynolds, lo cual era de esperar dado que el flujo es más complejo a medida que este aumenta. No obstante, en la reducción espacial del Isomap el error de reconstrucción es mayor para  $Re = 50$ , lo cual contradice el comportamiento observado en los demás resultados. En cuanto al costo computacional, la reducción espacial en todos los métodos (salvo los AE) requirió de unos pocos segundos. En contraste, la reducción temporal requirió hasta algunas horas. Por otro lado, el entrenamiento de los AE tardaron días en para la reducción espacial y horas para la temporal, resultando el método más costoso en materia de tiempo.

Aunque el enfoque temporal condujo a resultados inferiores en comparación con la reducción espacial, permitió extraer modos espaciales visualizables que resaltan las principales estructuras características del flujo, facilitando la interpretación física. Todos los métodos capturaron con éxito la dinámica principal del flujo alrededor del cilindro: el

desprendimiento de vórtices. En general, los distintos métodos extrajeron patrones similares, con diferentes niveles de complejidad. Estos consisten en regiones alternadas de alta y baja velocidad, que son más delgadas a medida que aumenta el orden del modo. Distintamente, el AE exhibió un patrón en zig-zag en la estela del cilindro.

En conclusión, se presentaron y probaron métodos no-lineales para reducir el tamaño espacial y temporal de los datos de simulaciones CFD. Los métodos de RD no-lineales lograron superar ampliamente al PCA tradicional en la reducción espacial. Sin embargo, la reducción del tamaño temporal resultó ser más desafiante y exigente computacionalmente, donde no se consiguió una mejora significativa en la calidad de reconstrucción. Por otro lado, la disposición temporal de los datos condujo a modos espaciales interpretables, los cuales fueron comparados cualitativamente e interpretados físicamente.

## 5.2. Recomendaciones

Como se ha mencionado, en ciertos casos los métodos no-lineales son capaces de superar ampliamente el rendimiento del PCA, como se demostró en la RD espacial. Sin embargo, presentan claras desventajas y limitaciones. Por ejemplo, la necesidad de ajustar sus hiperparámetros (número de K-NN, parámetro del *kernel*), la dificultad en la reconstrucción de datos para ciertos métodos (e.g., problema de la pre-imagen), y su mayor costo computacional. Por otro lado, en este trabajo no dio el tiempo para experimentar lo suficiente con los AE, los cuales dieron resultados inferiores en general. Diseñar arquitecturas de redes neuronales requiere una planificación e investigación exhaustiva de múltiples opciones (e.g., tamaño y profundidad de la red, función de activación, optimizador). En este sentido, cada problema físico requiere su propia sintonización de hiperparámetros.

Por lo tanto, como trabajo a futuro se recomienda:

1. Explorar en profundidad el dominio de los AE, probando distintas e hiperparámetros. Se ha demostrado, que en casos similares al de este trabajo se consiguen resultados superiores al PCA [2, 39]. Por ejemplo, Cheng et al. [40] integraron Redes Generativas Adversarias (GANs) y *Autoencoders* Variacionales (VAEs) para el modelado de flujos no-lineales.
2. Explorar otras técnicas del aprendizaje de variedades como: *Laplacian Eigenmaps* [41], *Hessian LLE* [42], *Discriminant Kernel LLE* [43]. Durante el transcurso de este trabajo se experimentó con el UMAP sin éxito. Sin embargo, Wang et al. [44] obtuvieron resultados muy positivos en la implementación de los algoritmos t-SNE y UMAP.



# Apéndice A

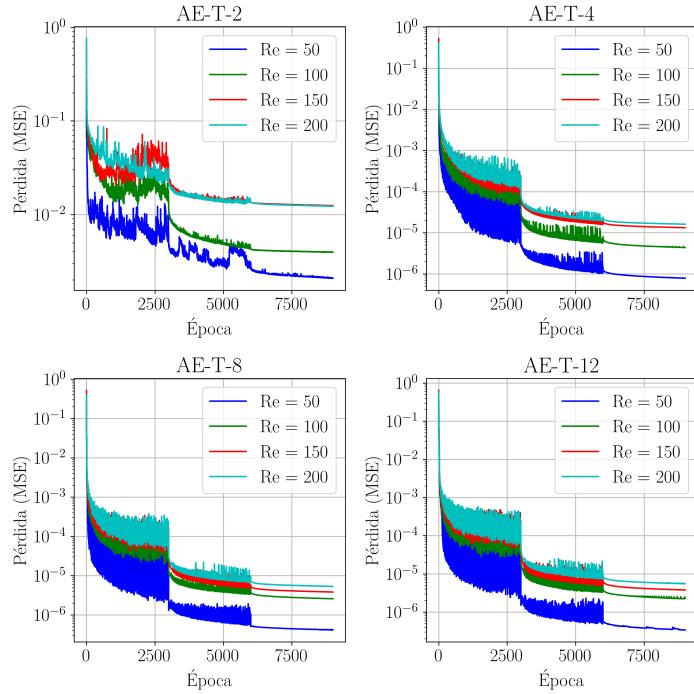
## Autoencoder

### A.1. Arquitecturas y pérdida

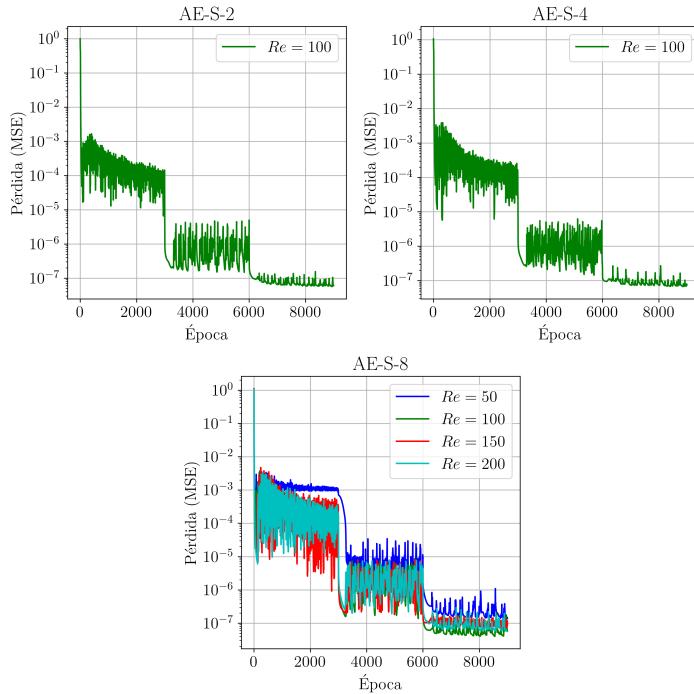
En la [Tabla A.1](#) se presentan las arquitecturas de los *autoencoders* empleados en este estudio. Las letras S y T en los casos indican reducción espacial y temporal, respectivamente. El número al final del nombre indica el número de neuronas en la capa de *bottleneck* (dimensión del espacio reducido). En todos los casos, las redes están completamente conectadas, utilizando la función de activación ReLU y la función de pérdida de error cuadrático medio (MSE). La estructura del *decoder* es inversa a la del *encoder*. La evolución de la función de pérdida en cada caso se muestran en las [Figs. A.1](#) y [A.2](#).

Caso	Estructura <i>encoder</i>	Capas	Tamaño del <i>batch</i>
AE-S-2	27104-8192-2048-512-128-32-8-2	8-1-8	128
AE-S-4	27104-8192-2048-512-128-32-8-4	8-1-8	128
AE-S-8	27104-8192-2048-512-128-32-8	7-1-7	128
AE-T-2	1000-512-256-64-16-8-2	7-1-7	256
AE-T-4	1000-512-256-64-16-4	6-1-6	256
AE-T-8	1000-512-256-64-32-8	6-1-6	256
AE-T-12	1000-512-256-64-32-12	6-1-6	256

**Tabla A.1:** Estructuras del *autoencoder* para reducción espacial (AE-S) y temporal (AE-T). La S y T corresponden a reducción espacial y temporal respectivamente. El número al final del nombre corresponde al número de neuronas de la *bottleneck* (dimensión reducida). La estructura del *decoder* es la inversa del *encoder*.



**Figura A.1:** Evolución de la función de pérdida en el entrenamiento de los AE para reducción temporal. Se muestran los gráficos para cada estructura (AE-T-2, AE-T-4, AE-T-8 y AE-T-12) y para cada  $Re$ .



**Figura A.2:** Evolución de la función de pérdida durante el entrenamiento de los AE para reducción espacial. Se presentan los gráficos correspondientes a cada estructura (AE-S-2, AE-S-4 y AE-S-8). Todas las curvas corresponden a  $Re = 100$ , salvo la de AE-S-8, que se incluyen la de cada  $Re$ .

## **Apéndice B**

# **Práctica Profesional Supervisada (PPS) y Actividades de Proyecto y Diseño (P&D)**

### **B.1. Práctica Profesional Supervisada (PPS)**

La PPS se realizó en el marco de la carrera de Ingeniería Nuclear del Instituto Balseiro en el Departamento de Mecánica Computacional del Centro Atómico Bariloche, en el formato de un proyecto integrador (PI). El presente PI se realizó bajo la dirección del Dr. William I. Machaca Abregú y la co-dirección del Dr. Eugenio Urdapilleta, durante el período de agosto del 2023 a junio del 2024, cumpliendo un mínimo de 200 hs.

### **B.2. Actividades de Proyecto y Diseño (P&D)**

Las actividades de P&D se realizaron en el marco del PI “Aplicación de técnicas de reducción dimensional en el modelado de flujos inestables”. El objetivo del proyecto fue estudiar la aplicación de técnicas de reducción dimensional en el modelado de flujos inestables. Dichas actividades incluyen, pero no se limitan, simulaciones numéricas de flujos canónicos, diseño e implementación de algoritmos de reducción dimensional, procesamiento y análisis de datos. Dichas tareas implicaron el estudio teórico e implementación de distintas tecnologías computacionales. En este contexto, se cumplieron con las 200 hs mínimas de actividades de P&D.



# Bibliografía

- [1] Taira, K., Brunton, S. L., Dawson, S. T. M., Rowley, C. W., Colonius, T., McKeon, B. J., *et al.* Modal analysis of fluid flows: An overview, 2017. [1](#)
- [2] Csala, H., Dawson, S., Arzani, A. Comparing different nonlinear dimensionality reduction techniques for data-driven unsteady fluid flow modeling. *Physics of Fluids*, **34** (11), 2022. [2](#), [66](#), [72](#), [73](#)
- [3] Wang, Z., Zhang, G., Xing, X., Xu, X., Sun, T. Comparison of dimensionality reduction techniques for multi-variable spatiotemporal flow fields. *Ocean Engineering*, **291**, 116421, 2024. [2](#)
- [4] Ahmed, S. E., Pawar, S., San, O., Rasheed, A., Iliescu, T., Noack, B. R. On closures for reduced order models—a spectrum of first-principle to machine-learned avenues. *Physics of Fluids*, **33** (9), 2021. [2](#)
- [5] Rowley, C. W., Colonius, T., Murray, R. M. Model reduction for compressible flows using pod and galerkin projection. *Physica D: Nonlinear Phenomena*, **189** (1), 115–129, 2004. URL <https://www.sciencedirect.com/science/article/pii/S0167278903003841>. [2](#)
- [6] Géron, A. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow. “O'Reilly Media, Inc.”, 2022. [ix](#), [ix](#), [ix](#), [x](#), [x](#), [3](#), [4](#), [25](#), [33](#)
- [7] Ghojogh, B., Crowley, M., Karray, F., Ghodsi, A. Elements of dimensionality reduction and manifold learning. Springer Nature, 2023. [xi](#), [4](#), [34](#), [36](#)
- [8] Kundu, P. K., Cohen, I. M., Dowling, D. R. Fluid mechanics. Academic press, 2015. [4](#), [12](#)
- [9] Sato, M., Kobayashi, T. A fundamental study of the flow past a circular cylinder using abaqus/cfd. En: 2012 SIMULIA Community Conference. 2012. [ix](#), [5](#), [6](#)
- [10] Hughes, W. F., Brighton, J. A. Fluid dynamics. McGraw-Hill, 1999. [6](#)

- [11] Bartholomew, P., Deskos, G., Frantz, R. A., Schuch, F. N., Lamballais, E., Laizet, S. Xcompact3d: An open-source framework for solving turbulence problems on a cartesian mesh. *SoftwareX*, **12**, 100550, 2020. [9](#), [10](#)
- [12] Pope, S. B. Turbulent flows. *Measurement Science and Technology*, **12** (11), 2020–2021, 2001. [9](#), [12](#)
- [13] Li, N., Laizet, S. 2decomp & fft-a highly scalable 2d decomposition library and fft interface. En: Cray user group 2010 conference, págs. 1–13. 2010. [10](#)
- [14] Machaca Abregu, W. I., Dari, E. A., Teruel, F. E. Conjugate heat transfer in spatial laminar-turbulent transitional channel flow. *International Communications in Heat and Mass Transfer*, **154**, 107430, 2024. URL <https://www.sciencedirect.com/science/article/pii/S0735193324001921>. [10](#)
- [15] Machaca Abregu, W. I., Dari, E. A., Teruel, F. E. Dns of heat transfer in a plane channel flow with spatial transition. *International Journal of Heat and Mass Transfer*, **209**, 124110, 2023. URL <https://www.sciencedirect.com/science/article/pii/S0017931023002636>. [10](#)
- [16] Lele, S. K. Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics*, **103** (1), 16–42, 1992. URL <https://www.sciencedirect.com/science/article/pii/002199919290324R>. [11](#)
- [17] Chorin, A. J. Numerical solution of the navier-stokes equations. *Mathematics of computation*, **22** (104), 745–762, 1968. [11](#)
- [18] Kader, B., Yaglom, A. Heat and mass transfer laws for fully turbulent wall flows. *International Journal of Heat and Mass Transfer*, **15** (12), 2329–2351, 1972. [12](#)
- [19] Schäfer, M., Turek, S., Durst, F., Krause, E., Rannacher, R. Benchmark computations of laminar flow around a cylinder. Springer, 1996. [12](#)
- [20] Lee, M., Moser, R. D. Direct numerical simulation of turbulent channel flow up to  $Re_\tau \approx 5200$ . *Journal of Fluid Mechanics*, **774**, 395–415, 2015. [ix](#), [12](#), [13](#)
- [21] Vreman, A., Kuerten, J. G. Comparison of direct numerical simulation databases of turbulent channel flow at  $re\tau = 180$ . *Physics of Fluids*, **26** (1), 2014. [ix](#), [13](#)
- [22] Mittal, R., Balachandar, S., *et al.* On the inclusion of three-dimensional effects in simulations of two-dimensional bluff-body wake flows. En: ASME fluids engineering division summer meeting, págs. 1–6. Citeseer, 1997. [ix](#), [12](#), [14](#)

- [23] Brunton, S. L., Kutz, J. N. Data-driven science and engineering: Machine learning, dynamical systems, and control. Cambridge University Press, 2022. [21](#), [23](#), [24](#)
- [24] Schölkopf, B., Smola, A., Müller, K.-R. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, **10** (5), 1299–1319, 1998. [x](#), [28](#), [30](#)
- [25] Aizerman, A. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control*, **25**, 821–837, 1964. [31](#)
- [26] Honeine, P., Richard, C. Preimage problem in kernel-based machine learning. *IEEE Signal Processing Magazine*, **28** (2), 77–88, 2011. [32](#)
- [27] Bakır, G. H., Weston, J., Schölkopf, B. Learning to find pre-images. *Advances in neural information processing systems*, **16**, 449–456, 2004. [32](#)
- [28] Murphy, K. P. Machine learning: a probabilistic perspective. MIT press, 2012. [32](#)
- [29] Roweis, S. T., Saul, L. K. Nonlinear dimensionality reduction by locally linear embedding. *science*, **290** (5500), 2323–2326, 2000. [33](#)
- [30] Saul, L. K., Roweis, S. T. An introduction to locally linear embedding. *unpublished. Available at: <http://www.cs.toronto.edu/~roweis/lle/publications.html>*, 2000. [33](#)
- [31] Tenenbaum, J. B., Silva, V. d., Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *science*, **290** (5500), 2319–2323, 2000. [xi](#), [38](#), [39](#)
- [32] Torgerson, W. S. Multidimensional scaling: I. theory and method. *Psychometrika*, **17** (4), 401–419, 1952. [38](#)
- [33] Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. Introduction to algorithms. MIT press, 2022. [39](#)
- [34] Plaut, E. From principal subspaces to principal components with linear autoencoders, 2018. [42](#), [72](#)
- [35] Prince, S. J. Understanding Deep Learning. MIT press, 2023. [xi](#), [xi](#), [42](#), [43](#), [44](#)
- [36] Kingma, D. P., Ba, J. Adam: A method for stochastic optimization, 2014. [46](#), [47](#)
- [37] Brunton, S. L., Noack, B. R., Koumoutsakos, P. Machine learning for fluid mechanics. *Annual review of fluid mechanics*, **52**, 477–508, 2020. [xi](#), [48](#)
- [38] Gautier, R., Laizet, S., Lamballais, E. A dns study of jet control with microjets using an immersed boundary method. *International Journal of Computational Fluid Dynamics*, **28** (6-10), 393–410, 2014. [52](#)

- [39] Agostini, L. Exploration and prediction of fluid dynamical systems using autoencoder technology. *Physics of Fluids*, **32** (6), 2020. [73](#)
- [40] Cheng, M., Fang, F., Pain, C., Navon, I. An advanced hybrid deep adversarial autoencoder for parameterized nonlinear fluid flow modelling. *Computer Methods in Applied Mechanics and Engineering*, **372**, 113375, 2020. [73](#)
- [41] Belkin, M., Niyogi, P. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation*, **15** (6), 1373–1396, 06 2003. URL <https://doi.org/10.1162/089976603321780317>. [73](#)
- [42] Donoho, D. L., Grimes, C. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, **100** (10), 5591–5596, 2003. [73](#)
- [43] Zhao, X., Zhang, S. Facial expression recognition using local binary patterns and discriminant kernel locally linear embedding. *EURASIP journal on Advances in signal processing*, **2012**, 1–9, 2012. [73](#)
- [44] Wang, Z., Zhang, G., Sun, T., Shi, C., Zhou, B. Data-driven methods for low-dimensional representation and state identification for the spatiotemporal structure of cavitation flow fields. *Physics of Fluids*, **35** (3), 2023. [73](#)

# Agradecimientos

A mamá y papá, mi causalidad, que me dieron la libertad y el soporte para seguir mi camino. A los que más le debo, este orgullo es cosecha de su de labor.

A la Piru, la barra brava n°1 en mi trayectoria académica. A Isabel, por su inexorable cariño. A mis abuelos, con quien me hubiera gustado compartirlo.

A toda mi familia, amigos y cada persona que me brindó cariño y ayudó en cualquier aspecto. En especial, a Pancho el desertor M., por su guía y amistad incondicional.

A mis compañeros y directores, por ayudarme en la tenaz tarea de recibirme. A JPM y a Raffe, por cada café con exquisita, por tirar del carro.

A Pedro, Lucho, Manu, Joaco, Berseker, Scofleta, Nacho, Cálvaro y Gimeno, quienes formaron el hilo del cual pendió mi cordura, los que hicieron que esto valga la pena.

A Magda, el amor de mi vida, quien me esperó y bancó incondicionalmente estos años. Con quien anhelo compartir el fruto de este esfuerzo.

