Universidad Zaragoza
1542

Trabajo Fin de Máster

Modelado físicamente riguroso de tejidos biológicos blandos a partir de datos

Physics-aware data-driven modeling of biological soft tissues

Autor

Nicolás Escribano

Directores

David González Ibáñez

Beatriz Moya García

Máster en Ingeniería Biomédica

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2021

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe remitirse a seceina@unizar.es dentro del plazo de depósito)

D./Dª. Nicolás Escribano ,

en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de Estudios de la titulación de

Máster Universitario en Ingeniería Biomédica (Título del Trabajo)

Modelado físicamente riguroso de tejidos biológicos blandos a partir de datos.
(Physics-aware data-driven modeling of biological soft tissues)

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 15 de noviembre de 2021

Fdo: Nicolás Escribano

AGRADECIMIENTOS

A mi familia, por estar siempre ahí.

A mis compañeros, por enseñarme lo que no se enseña en las aulas.

Y sobre todo a David y Bea, sin su ayuda y apoyo este trabajo no habría sido posible.

[...]

Y a todos los que me dejo en el tintero, que seguro, no son pocos.

Modelado físicamente riguroso de tejidos biológicos blandos a partir de datos

RESUMEN

El presente Trabajo de Fin de Máster tiene como objetivo desarrollar una herramienta basada en inteligencia artificial, más concretamente en las llamadas redes SPNN (*Structure-Preserving Neural Networks*), que sea capaz de aprender leyes de comportamiento de tejidos biológicos blandos para reproducir la evolución temporal de diferentes variables seleccionadas que definen su comportamiento en base al formalismo GENERIC (*General Equation for the Non-equilibrium Reversible-Irreversible Coupling*), que asegura el cumplimiento de las leyes básicas de la termodinámica.

Primero se reproducirán dos modelos computacionales de validación, un modelo simplificado de la córnea humana y un modelo de la capa adventicia de la arteria aorta ilíaca. En cada uno de estos modelos se harán simulaciones con el programa Abaqus para obtener datos de la evolución temporal de las variables seleccionadas y con esto generar una base de datos para cada modelo. Estas bases de datos servirán para entrenar a la red neuronal en los diferentes modelos y poder recrear su evolución temporal. Por último, finalizaremos con la imitación de un caso experimental, emulando los datos reales y entrenando a la red con variables observables.

La principal dificultad del proyecto radica en la adaptación de las redes SPNN a cada caso específico debido a la alta no linealidad y complejidad de los comportamientos que queremos aprender, adaptando los hiperparámetros de la red para conseguir reproducciones fieles de la evolución temporal de las variables de estado de cada problema.

Por último, se analizarán y discutirán los resultados obtenidos, especialmente la capacidad de la red para adaptarse a diferentes modelos de tejidos biológicos blandos, así como su eficacia frente a redes neuronales tradicionales de aprendizaje profundo.

Índice

Capítulo 1

Introducción

1.1. Objetivo y alcance.

El principal objetivo del presente Trabajo del Fin de Máster es aprender modelos de comportamiento subyacente en datos de tejidos biológicos blandos mediante el uso de redes neuronales para entrenar modelos físicamente rigurosos en base al formalismo GENERIC (*General Equation for the Nonequilibrium Reversible-Irreversible Coupling*) [1] [2].

Se desarrollará un algoritmo de redes neuronales de tipo SPNN (*Structure-Preserving Neural Networks*) [3] para aprender dichos modelos. Estas redes neuronales han sido ya aplicadas en diferentes trabajos [4] y aseguran el cumplimiento de las leyes de la termodinámica en base al formalismo GENERIC, también aplicado en diferentes trabajos para la resolución de problemas de físicas complejas y materiales biológicos blandos [5] [6].

Estas redes serán aplicadas primero sobre datos sintéticos obtenidos mediante la simulación computacional avanzada del comportamiento de tejidos biológicos con el software comercial de elementos finitos Abaqus. Se realizarán varias simulaciones con diferentes condiciones de cargas y así tener una base de datos con las que entrenar nuestras redes. En segundo lugar, a partir de datos sintéticos, se emulará la extracción de datos experimentales, trabajando solo con variables observables que presenten ruido común de medidas reales para comprobar la robustez del método y el aprendizaje de nuevas muestras de paciente especifico.

El aprendizaje de dichas leyes de comportamiento se llevará a cabo con redes neuronales que aseguren la consistencia física del modelo mediante la imposición de las leyes de la termodinámica. Dicha premisa promete dotar de mayor generalidad y precisión al modelo obtenido, siendo capaz de aprender el comportamiento de nuevos casos con alta dispersión de propiedades de manera eficiente. Además, mediante la imposición de este conocimiento en los entrenamientos, no serán necesarios tantos datos para aprender sus características, y el entrenamiento sera más eficiente.

La principal aportación del trabajo, por tanto, consiste en la aplicación y análisis de técnicas de *Deep Learning* con restricciones físicas, y sus beneficios frente a modelos tradicionales de aprendizaje automático, en el aprendizaje de leyes de comportamiento de tejidos biológicos con impacto no sólo en el campo de la bioingeniería, sino también en el aprendizaje de físicas complejas.

1.2. Tema y contexto.

El aprendizaje automático (*Machine learning*) ha supuesto una revolución en los últimos años en el ámbito científico. Sin embargo, la aplicación de estas técnicas ha seguido una evolución temporal mayor [7]. El auge de las redes neuronales ha hecho que crezca su aplicación en herramientas computacionales. Las redes neuronales nos dan un gran poder para aprender información a partir de datos. Con el aprendizaje profundo (Deep learning) se consigue, por medio de redes neuronales con más capas, aprender conceptos cada vez más abstractos y problemas más complejos [8].

Las redes neuronales en el ámbito del comportamiento de materiales han sido aplicadas para aprender modelos que matemáticamente tenían una complejidad muy elevada [9] [10] [11] [12]. La mayoría de estas redes usan las herramientas de aprendizaje profundo para aprender los diferentes comportamientos de los materiales sin ningún tipo de sesgo. Ahora se ha dado un paso mas allá y se ha dotado a las redes de sesgos inductivos en forma de restricciones físicas con las cuales se consigue que las redes neuronales aprendan realmente los comportamientos que se hallan en los datos con los que aprende. Los sesgos inductivos de un algoritmo son las suposiciones que se utilizan para predecir la salida dadas las entradas que aún no ha encontrado [13].

Estos avances en la forma de caracterizar el comportamiento de diferentes materiales también han llegado al campo de la bioingeniería y biomecánica [14] [15] [16]. Muchos de estos trabajos, además de incorporar el aprendizaje profundo, también se hacen eco de la implementación de restricciones en las redes neuronales pudiendo así llegar

a resultados óptimos, interpretables y generalizables, en entrenamientos de redes menos profundas en problemas complejos [17] [18]. Estás imposiciones físicas sobre la estructura de las redes neuronales hacen capaz al algoritmo de aprender las verdaderas leyes que gobiernan los materiales mediante un entrenamiento guiado y así tener una reproducción eficaz de su comportamiento.

En [3] se implementa el formalismo GENERIC [1] en una red neuronal de aprendizaje profundo. En este trabajo las restricciones de las cuales se ayuda la red son el cumplimiento de las leyes fundamentales de la termodinámica en la evolución temporal de diferentes sistemas físicos. Estas redes, llamadas SPNN (Structure-Preserving Neural Networks), con probada eficacia en diferentes problemas dinámicos [4], son las que aplicaremos en este proyecto para realizar el aprendizaje de tejidos biológicos blandos.

1.3. Contenido del proyecto.

La memoria está dividida en los siguientes capítulos, que explican el trabajo desarrollado en este proyecto Final de Máster.

- Capítulo 2: El formalismo GENERIC y las redes SPNN.
  En este capítulo se explicará el formalismo GENERIC en el que basamos las redes neuronales para que se consigan satisfacer las leyes de la termodinámica, así como su adaptación al aprendizaje profundo (redes SPNN) y se realizará una validación de esta estructura en problemas simples.

- Capítulo 3: Aplicación de la red SPNN a modelos computacionales.
  En el capítulo 3 se explicará cómo se ha llevado a cabo la implementación de las redes SPNN para dos diferentes modelos computacionales, el primero un modelo de la córnea humana y el segundo un modelo de la capa adventicia de la arteria aorta ilíaca. En ambos casos se explicarán los modelos computacionales, la extracción de datos y los resultados obtenidos por las redes.

- Capítulo 4: Aplicación de la red SPNN a un caso pseudo-experimental.
  En el capítulo 4 se explicará cómo ha sido la implementación de las redes SPNN para la reproducción de un caso pseudo-experimental en el modelo de la capa adventicia de la arteria aorta ilíaca y se analizarán los resultados obtenidos para este caso.

– Capítulo 5: Conclusiones y líneas futuras.
  En el capítulo final se analizarán los principales resultados obtenidos para las aplicaciones a diferentes modelos, se comentarán las principales conclusiones y se escribirán líneas de trabajo futuras.

Capítulo 2

El formalismo GENERIC y las redes SPNN.

Las redes neuronales están abriendo un mundo de posibilidades en muchos ámbitos de la ingeniería y la ciencia de datos.

En este capítulo se explicará el formalismo que usaremos dentro de las redes SPNN (Structure-Preserving Neural Networks) para que las redes neuronales, en vez de ser una "caja negra", se basen en leyes físicas básicas y con esto ayudar al aprendizaje para que, con menos poder de cálculo, se pueda llegar a resultados óptimos.

2.1. El formalismo GENERIC.

El formalismo GENERIC (General Equation for Non Equilibrium Reversible-Irreversible Coupling) propuesto por Miroslav Grmela y Hans Christian Öttinger en 1997, describe la evolución dinámica de un conjunto de variables de estado respecto a la evolución de energía y entropía del sistema [1] [2].

Este formalismo establece una ecuación genérica para la dinámica de un sistema en condiciones reversibles e irreversibles. Con esta descripción, en casos tanto conservativos como disipativos, podemos construir modelos que aseguren una estructura termodinámica sólida (garantizan las leyes de conservación/disipación de la energía y la entropía).

La ecuación de GENERIC se define de la siguiente manera:

$$\dot{\mathbf{z}}_t = \mathbf{L}(\mathbf{z}_t)\nabla E(\mathbf{z}_t) + \mathbf{M}(\mathbf{z}_t)\nabla S(\mathbf{z}_t) \tag{2.1}$$

Donde $\dot{\mathbf{z}}_t$ son las variables de estado, las variables que más influyen en la evolución dinámica del problema en una descripción a nivel de mesoescala. Además, la selección de estas variables permite una descripción completa de la energía y entropía del sistema. Si no se cumple esta condición, la descripción carece de la estructura GENERIC. $\mathbf{L}$ es la matriz de Poisson y tiene que ser antisimétrica y E representa la energía del sistema en función de las variables de estado. $\mathbf{M}$ es la matriz de fricción y tiene que ser semidefinida positiva y por último S representa la entropía del sistema en función de las variables de estado en un tiempo t.

$\mathbf{L}$ junto con el gradiente de energía (∇E) representan la parte reversible del sistema (parte conservativa o Hamiltoniana) y $\mathbf{M}$ junto con el gradiente de entropía (∇S) reflejan los fenómenos disipativos.

Otro requisito del formalismo GENERIC son las llamadas condiciones de degeneración.

$$\mathbf{L}(\mathbf{z}_t)\nabla E(\mathbf{z}_t) = 0, \tag{2.2}$$

$$\mathbf{M}(\mathbf{z}_t)\nabla S(\mathbf{z}_t) = 0. \tag{2.3}$$

Esto significa que el potencial energético no contribuye a la producción de entropía y que la función de entropía no contribuye a la conservación de la energía. Estas condiciones de degeneración más los requisitos ya mencionados de las matrices $\mathbf{L}(\mathbf{z})$ y $\mathbf{M}(\mathbf{z})$ nos garantizan:

$$\dot{E}(\mathbf{z}) = \nabla E(\mathbf{z}) \cdot \dot{\mathbf{z}} = \nabla E(\mathbf{z}) \cdot \mathbf{L}(\mathbf{z})\nabla E(\mathbf{z}) + \nabla E(\mathbf{z}) \cdot \mathbf{M}(\mathbf{z}_t)\nabla S(\mathbf{z}_t) = 0 \tag{2.4}$$

$$\dot{S}(\mathbf{z}) = \nabla S(\mathbf{z}) \cdot \dot{\mathbf{z}} = \nabla S(\mathbf{z}) \cdot \mathbf{L}(\mathbf{z})\nabla E(\mathbf{z}) + \nabla S(\mathbf{z}) \cdot \mathbf{M}(\mathbf{z}_t)\nabla S(\mathbf{z}_t) \geq 0 \tag{2.5}$$

La ecuación 2.4 expresa la conservación de la energía en un sistema aislado, primera ley de la termodinámica, y la ecuación 2.5 establece que la cantidad de entropía en el universo aumenta, segunda ley de la termodinámica. GENERIC nos establece una base

capaz de encontrar las leyes constitutivas de sistemas complejos sin ninguna suposición más que las restricciones de las leyes fundamentales de la termodinámica.

2.2. Aplicación del formalismo GENERIC en las redes SPNN.

El contexto físico GENERIC es de gran utilidad aplicado a la resolución de problemas discretos [6]. Aunque se conoce la forma de GENERIC para muchos sistemas, en este trabajo se quiere aprender a partir de datos para que se ajuste a cada caso específico que planteamos mediante una aproximación discreta para este formalismo.

2.2.1. Algoritmo de integración.

Para poder resolver numéricamente la ecuación de GENERIC hay que formular la versión discretizada de la ecuación 2.1.

$$\frac{\mathbf{z_{n+1}} - \mathbf{z_n}}{\Delta t} = \mathbf{L} \cdot \frac{DE}{D\mathbf{z}} + \mathbf{M} \cdot \frac{DS}{D\mathbf{z}} \tag{2.6}$$

La derivada temporal de la ecuación original se discretiza siguiendo un esquema de Euler hacia adelante en incrementos de tiempo Δt siendo $\mathbf{z}_{n+1}$ el vector de variables de estado en el siguiente instante temporal. $\mathbf{L}$ y $\mathbf{M}$ son las versiones discretizadas de las matrices de Poisson y fricción nombradas anteriormente y $\frac{DE}{D\mathbf{z}}$ y $\frac{DS}{D\mathbf{z}}$ representan los gradientes discretizados que se pueden aproximar por:

$$\frac{DE}{D\mathbf{z}} \approx \mathbf{A}\mathbf{z}, \tag{2.7}$$

$$\frac{DS}{D\mathbf{z}} \approx \mathbf{B}\mathbf{z}. \tag{2.8}$$

Donde $\mathbf{A}$ y $\mathbf{B}$ representan las matrices discretizadas de los operadores gradientes.

Con esto las condiciones de degeneración dicretizadas quedarían como:

$$\mathbf{L} \cdot \mathbf{B}\mathbf{z}_n = 0, \tag{2.9}$$

$$\mathbf{M} \cdot \mathbf{A}\mathbf{z}_n = 0. \tag{2.10}$$

Finalmente, manipulando algebraicamente la ecuación 2.6 con la ecuación 2.1 incluyendo las condiciones de degeneración de las ecuaciones 2.9 y 2.10, el esquema de integración propuesto para predecir la dinámica de un sistema físico es el siguiente:

$$\mathbf{z}_{n+1} = \mathbf{z}_n + \Delta t(\mathbf{L} \cdot \mathbf{B}\mathbf{z}_n + \mathbf{M} \cdot \mathbf{A}\mathbf{z}_n) \tag{2.11}$$

Con el formalismo discretizado se puede realizar su implantación computacional para poder ser aplicado en algoritmos de aprendizaje automático.

2.2.2. Redes SPNN.

Las redes neuronales tienen una gran versatilidad en el entendimiento de sistemas dinámicos y su aplicación va más allá del campo de la computación, pudiéndose ajustar a diferentes problemas físicos por su capacidad de adaptación a casos complejos no lineales.

En este trabajo se utiliza un tipo de redes neuronales de aprendizaje profundo que se basan en múltiples capas de neuronas totalmente conectadas entre sí con una capa de entrada y una capa de salida (Fig. 2.1). Como factor diferenciador, estas redes incluyen sesgos inductivos para guiar el aprendizaje hacia una solución físicamente consistente, específicamente aquella que preserve la estructura GENERIC [3] [4].

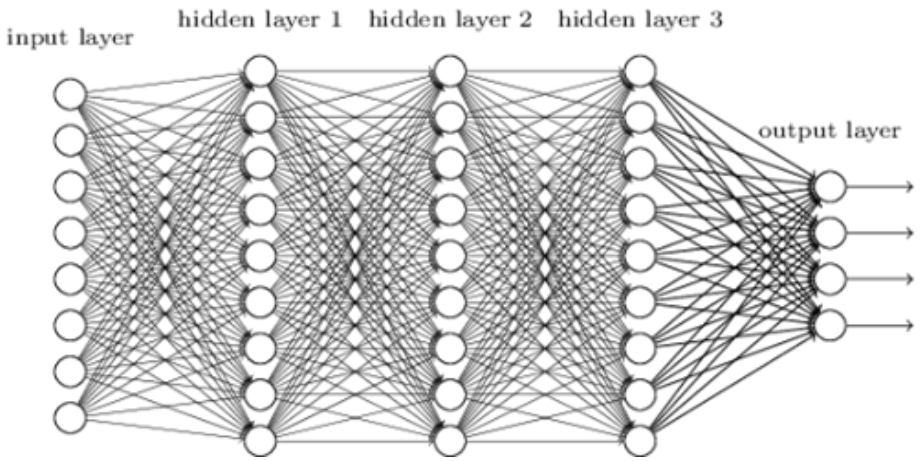

Figura 2.1: Representación de una red de neuronas completamente conectadas (Fuente: Internet).

Aunque la eficacia de las redes neuronales esta probada en multitud de trabajos [9] [10] [11] [12] se ha demostrado que introducir sesgos inductivos [17] [18] mejora notablemente los entrenamientos necesitando menos datos para que las redes sean

entrenadas, además de dotar de interpretabilidad y generalidad a los algoritmos. En nuestro caso, el sesgo que se impondrá es el del formalismo GENERIC por la estructura del integrador que aplicamos en el aprendizaje y las condiciones de degeneración explicadas en la sección anterior.

La entrada de la red neuronal será el vector de variables de estado en un instante de tiempo dado ($\mathbf{z}_n$) y las salidas serán las matrices de GENERIC concatenadas ($\mathbf{L}_n$, $\mathbf{M}_n$) junto con los gradiente de energía y entropía (DE y DS). Después usando la expresión de GENERIC discretizada (Ecuación 2.11), se obtendrá el vector de variables de estado en el siguiente instante de tiempo ($\mathbf{z}_{n+1}$).

La función de pérdida para las redes SPNN está compuesta de tres términos diferentes:

- Error de reconstrucción: La principal función de pérdida es la equivalencia entre la salida de la red y los datos reales. Se calcula como la suma de los errores cuadráticos entre la predicción y la realidad para cada instante de tiempo.

$$\mathcal{L}^{Recons} = \frac{1}{N_{entrenamiento}} \sum_{i=1}^{N_{entrenamiento}} (\mathbf{z}_{i+1}^{Real} - \mathbf{z}_{i+1}^{SPNN})^2 \tag{2.12}$$

- El cumplimiento de las condiciones de degeneración: La función de pérdida también tendrá en cuenta las condiciones de degeneración (Ecuación 2.10 y 2.9) para asegurar el cumplimiento de las leyes de la termodinámica. Esta es implementada como la suma de los de los vectores de degeneración al cuadrado en cada instante de tiempo.

$$\mathcal{L}_n^{Degen} = \frac{1}{N_{entrenamiento}} \sum_{i=1}^{N_{entrenamiento}} (\mathbf{L}_i \cdot D\mathbf{S}_i)^2 + (\mathbf{M}_i \cdot D\mathbf{E}_i)^2 \tag{2.13}$$

- Regularización L2: para asegurarnos de que la red no sobreentrene, se añade una regularización L2 extra a la función de pérdida.

$$\mathcal{L}^{Reg} = \sum_{l}^{L} \sum_{i}^{n^{[l]}} \sum_{j}^{n^{[l+1]}} (w_{i,j}^{[l],SPNN])^2} \tag{2.14}$$

Estas funciones de pérdida se ponderan con dos hiperparámetros adicionales λ_d y λ_r, que representan la influencia relativa en la función de pérdida total frente a la restricción por degeneración.

$$\mathcal{L}^{SPNN} = \frac{1}{N_{entrenamiento}} \sum_{n=0}^{N_{entrenamiento}} (\lambda_d^{SPNN} \mathcal{L}_n^{Recons} + \mathcal{L}_n^{Degen}) + \lambda_r^{SPNN} \mathcal{L}^{Reg} \tag{2.15}$$

La base de datos, que consiste en vectores de las variables de estado evaluadas en cada instante de tiempo, se dividirá en dos. El 80 % de estas instantáneas será para el entrenamiento y el 20 % restante para la test. ($N_{total} = N_{entrenamiento} + N_{test}$).

En estas redes se utiliza el algoritmo habitual de retropropagación (*Backpropagation*) [19] para calcular el gradiente de la función de pérdida para cada parámetro de la red, que se actualiza con la técnica del descenso del gradiente [20].

Por último la prueba de validación consiste en la integración temporal completa del vector de variables de estado inicial $\mathbf{z}_t$ en $t = 0$ a lo largo de todo el intervalo de tiempo de simulación dado solo el vector de variables de estado en el primer instante de tiempo. El esquema del proceso se puede ver en la figura 2.2. Se evaluará su eficacia con el error cuadrático medio (MSE) entre el vector de estado que nos da la red y la referencia para toda la evolución temporal total.

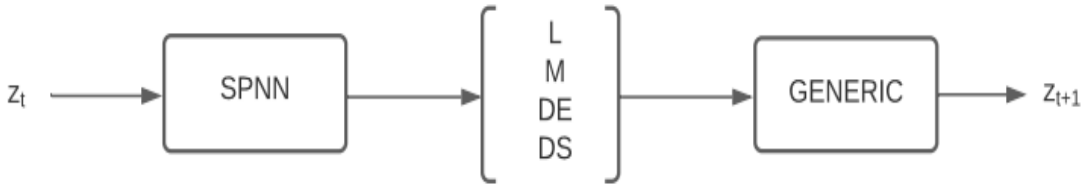

Figura 2.2: Esquema de la red SPNN.

2.3. Ejemplos funcionales de redes SPNN.

A continuación se presentan ejemplos de las redes SPNN para su aprendizaje y prueba ya implementados en la literatura [3].

2.3.1. Péndulo simple.

El primer ejemplo de validación es el péndulo simple (Fig. 2.3), que en nuestro caso será puramente Hamiltoniano debido a la ausencia de disipación. Por lo tanto, el término disipativo ($\mathbf{M}$) es nulo.

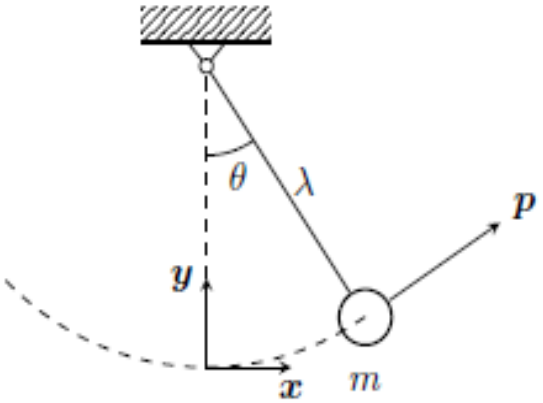

Figura 2.3: Esquema de un pendulo simple (Fuente: [3]).

La ecuación en derivadas parciales (EDP) de un péndulo simple es conocida y es de la siguiente forma:

$$\frac{d^2\theta}{dt^2} + \frac{g}{\lambda}\sin\theta = 0 \tag{2.16}$$

Donde θ es el ángulo que se forma con la vertical, m es la masa del péndulo, λ es la longitud del cable del péndulo y, por último, g es la aceleración de la gravedad. Este es un problema ya descrito en la bibliografía [6].

La variables que describen la energía del sistema para la descripción de GENERIC son el ángulo θ y la velocidad angular, $\omega = \dot{\theta} = \frac{d\theta}{dt}$, por lo que el vector de variables de estado es el siguiente:

$$\mathbf{z} = (\theta, \omega) \in (\mathbb{R} \times \mathbb{R}) \tag{2.17}$$

Como ya hemos mencionado, en este caso el péndulo es puramente Hamiltoniano por lo que la matriz de fricción **M** sera nula y, por lo tanto, las matrices del sistema quedan como:

$$\mathbf{L} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \mathbf{M} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.18}$$

En este caso, el gradiente de energía es el único elemento de la optimización en el aprendizaje del algoritmo.

Una vez definido el problema y con la implementación de las redes ya descritas vamos a crear los datos necesarios para el entrenamiento.

Los datos para el entrenamiento se generan con el programa Matlab resolviendo la EDP (2.16), en la que se han definido la masa como $m = 5kg$ y la longitud del cable como $\lambda = 1m$. El tiempo de la simulación es de 5 segundos con incrementos de tiempo de 0.05 segundos por lo que tendremos 100 incrementos de tiempo.

Por lo tanto, el tamaño de la entrada para la red será el número de variables de estado ($dim_{in} = 2$), el tamaño de salida de la red viene dada por la expresión $dim_{out} = dim_{in} \cdot (dim_{in}^2) = 8$, el número de capas ocultas será de $N_{Ocultas} = 3$ con 8 neuronas cada capa, la función de activación para las capas ocultas será del tipo ReLU y lineal en la capa de salida. La red se inicializa según el método de Kaiming [21] con una distribución normal y el optimizador usado es Adam [22] con una tasa de aprendizaje (*Learning rate*) $L_r = 10^-5$ y el peso (*Weight decay*) $\lambda_r = 10^-5$.

Los resultados para esta primera simulación los podemos ver en la figura 2.4.

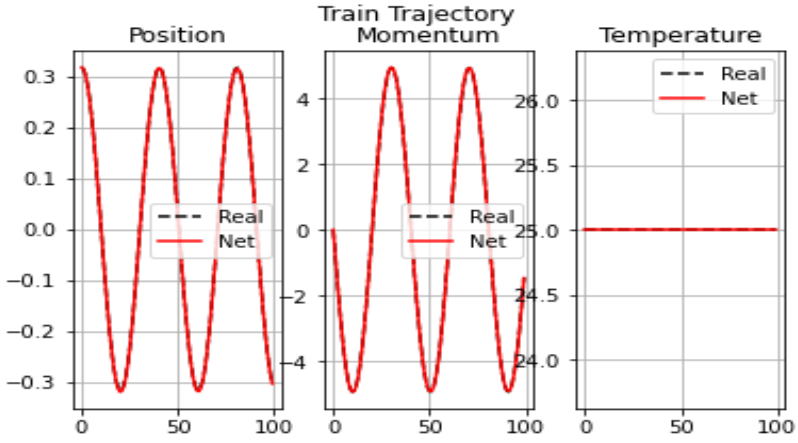

Figura 2.4: Resultados de la validación de la red para el péndulo simple.

En la tabla 2.1 podemos ver el error medio cuadrático de la simulación en el entrenamiento de la red neuronal, donde se aprecia, al igual que en las imágenes, que los errores son muy bajos en ambas variables de estado. Además la temperatura no varía por lo que se verifica se aprende la estructura real del sistema y se preservan los principios termodinámicos.

Variable	Error medio cuadratico (MSE)
Posición	1.99E-06
Momento	6.36E-04
Temperatura	0.00E+00

Tabla 2.1: Error medio cuadrático de las diferentes variables en el ejemplo de péndulo simple.

En este primer ejemplo de validación se puede ver que una arquitectura muy sencilla de red neuronal llega a adaptar perfectamente la simulación de un sistema dinámico no lineal.

2.3.2. Flujo de Couette.

El siguiente ejemplo es un flujo de Couette de un fluido Oldroyd-B, que se trata de un modelo constitutivo para fluidos viscoelásticos y consiste en mancuernas elásticas lineales, que representan cadenas de polímeros inmersas en un disolvente.

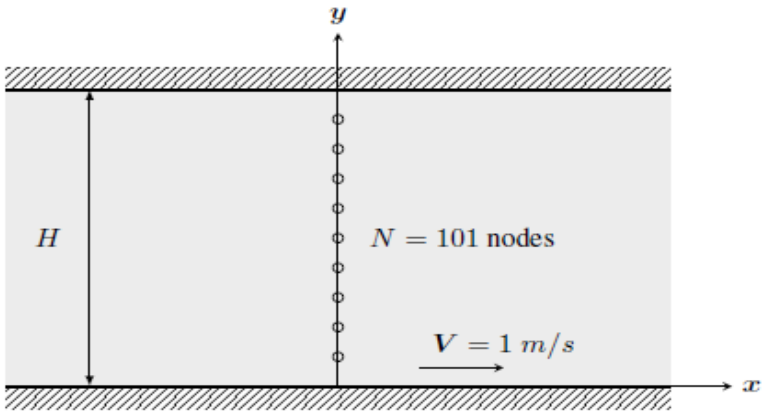

Figura 2.5: Flujo de Coutte en un fluido Oldroy-B (Fuente: [3]).

Las variables de estado seleccionadas para este problema son la posición del fluido en cada nodo de la malla q, su velocidad v en la dirección x, la energía interna e y la componente tangencial del tensor de conformación τ:

$$\mathbf{z} = (\mathbf{q}, v, e, \tau) \in (\mathbb{R}^2 \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}) \tag{2.19}$$

Y las matrices de Poisson y de fricción asociadas, $\mathbf{L}$ y $\mathbf{M}$ respectivamente descritas en la literatura [6], son:

$$\mathbf{L} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \mathbf{M} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.20}$$

Los datos de entrenamiento para el modelo fueron generados en MatLab. El fluido está discretizado en vertical con $N = 100$ elementos (101 nodos) en una altura $H = 1m$ (Fig. 2.5). Se ha considerado un total de 10.000 mancuernas en cada nodo del modelo. La velocidad de fluencia se fija en $V = 1m/s$, el tiempo de simulación es $T = 1s$ y los incrementos de tiempo $\Delta t = 0{,}0067s$ con lo que tendremos 150 instantes de tiempo. Para la base de datos se ha cogido las variables de estado de los 100 primeros nodos,

dividiendo así el 80 % de estos para el entrenamiento y el 20 % restante para la prueba de validación.

El tamaño de la entrada para la red, por lo tanto, será el número de variables de estado ($dim_{in} = 5$), el tamaño de salida de la red viene dada por la expresión $dim_{out} = dim_{in} \cdot (dim_{in}^2) = 50$, el número de capas ocultas será de $N_{Ocultas} = 10$ con 50 neuronas en cada capa, la función de activación para las capas ocultas será del tipo ReLU y lineal en la capa de salida. La red se inicializa según el método de Kaiming con una distribución normal y el optimizador usado es el Adam con una tasa de aprendizaje (Learning rate) $L_r = 10^-4$ y el peso (Weight decay) $\lambda_r = 10^-5$.

Los resultados conseguidos para esta red se pueden ver en las figuras 2.6 y 2.7 , que son resultados para las 4 variables de estado tanto en el entrenamiento como en la reproducción de la validación.

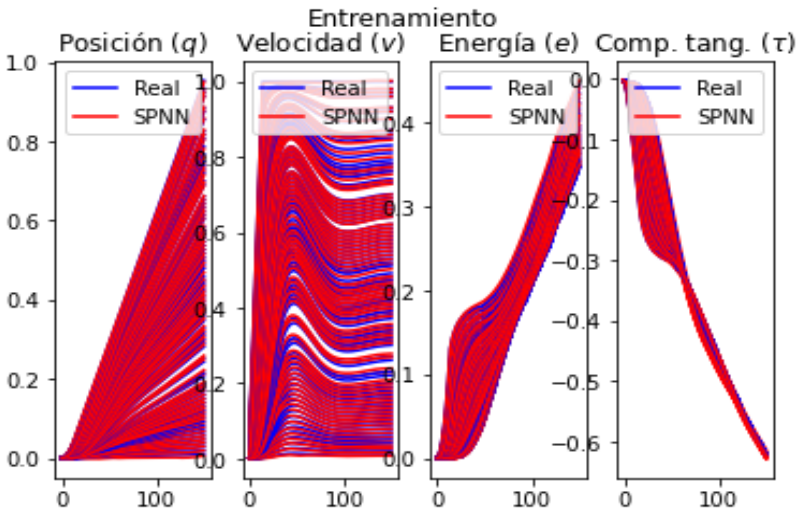

Figura 2.6: Resultados del entrenamiento para las variables de estado del flujo de Coutte.

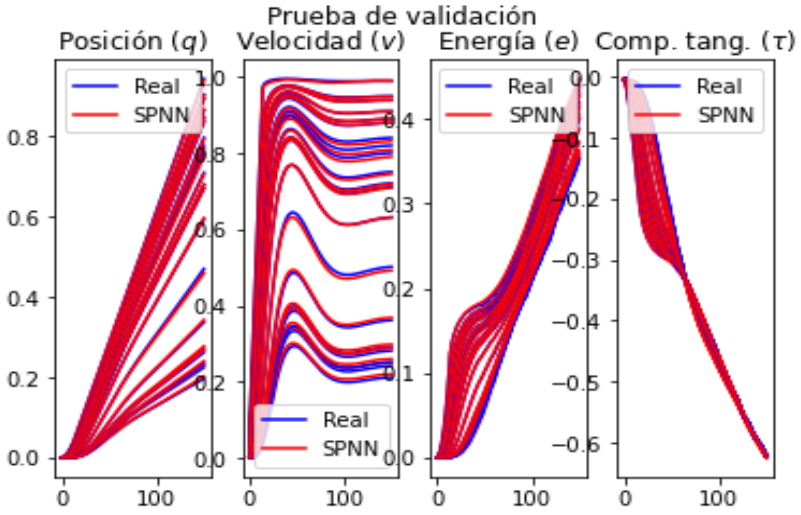

Figura 2.7: Resultados de la prueba de validación para las variables de estado del flujo de Coutte.

En la tabla 2.2 podemos ver el error medio cuadrático de las variables de estado del problema, se puede apreciar que los resultados de las predicciones son precisos tanto en las trayectorias de entrenamiento como en las de la prueba de validación.

Variable	MSE del entrenamiento	MSE de la validación
Posición	5.50E-06	5.67E-06
Velocidad	3.10E-05	3.28E-05
Energía	6.63E-06	7.43E-06
Componente tangencial	1.68E-05	1.92E-05

Tabla 2.2: Error medio cuadrático de las diferentes variables en el ejemplo de flujo de Coutte.

Con este segundo ejemplo se comprueba la eficacia de GENERIC y de las redes SPNN en la reproducción de la evolución temporal de sistemas complejos y su posibilidad de aplicación a diferentes modelos con físicas complejas.

Capítulo 3

Aplicación de la red SPNN a modelos computacionales.

En este capítulo implementaremos las redes SPNN en dos modelos computacionales diferentes, primero un modelo de la córnea humana y segundo a un modelo de la capa adventicia de la arteria aorta ilíaca. Para ambos modelos describiremos su simulación computacional, la extracción de los datos para la red y terminaremos con la implementación de la red y los resultados obtenidos.

3.1. Aplicación de la red SPNN en un modelo ocular hiperelástico.

3.1.1. Descripción del modelo.

El primer caso biológico en el que implementaremos las redes SPNN será un modelo simplificado de la córnea.

La córnea es un tejido biológico blando estructurado por capas (Fig. 3.1). En nuestro modelo se simulará un corte transversal de la córnea y en una geometría aproximada sin curvatura.

Para este problema se hará un modelo 2D de $12mm$ de largo y $0{,}55\mu m$ de alto simulando una córnea sana. En este modelo simularemos un ensayo de tracción uniaxial. Para ello modelaremos un cuarto de la probeta con condiciones de simetría en el eje x y en el eje y (Fig. 3.2).

El tipo de material que se ha escogido para este modelo es de tipo Yeoh como modelo de material hiperelástico para modelar el comportamiento de las distintas capas de la

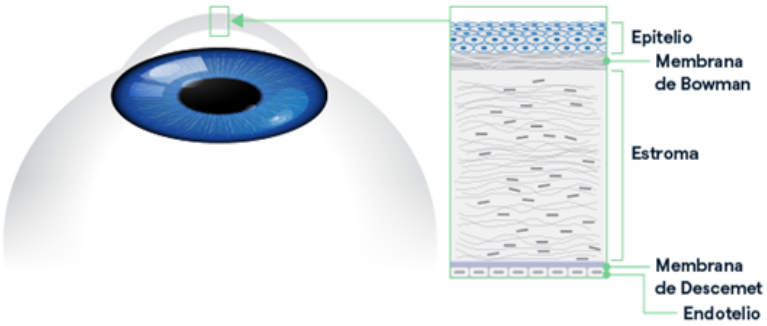

Figura 3.1: Capas de la córnea (Fuente: Internet).

Figura 3.2: Modelo simplificado de la córnea.

córnea. Los parámetros se han tomado de [23] y se pueden ver en la tabla 3.1.

La forma del potencial de energía de deformación Yeoh [24] es:

$$U = C_{10}(\bar{I}_1 - 3) + C_{20}(\bar{I}_1 - 3)^2 + C_{30}(\bar{I}_1 - 3)^3 + \frac{1}{D_1}(J^{el} - 1)^2 + \frac{1}{D_2}(J^{el} - 1)^4 + \frac{1}{D_3}(J^{el} - 1)^6, \tag{3.1}$$

donde U es la energía de deformación por unidad de volumen de referencia; C_{i0} y D_i son los parámetros del material que dependen de la temperatura, $\bar{I}_1$ es el primer invariante de deformación definido como:

$$\bar{I}_1 = \bar{\lambda}_1^2 + \bar{\lambda}_2^2 + \bar{\lambda}_3^2, \tag{3.2}$$

donde las deformaciones desviatorias $\bar{\lambda}_i = J^{\frac{-1}{3}}\lambda_i$, J es la relación de volumen total, J^{el} es la relación de volumen elástica en expansión térmica y λ_i son las deformaciones iniciales.

C_{10} $[kPa]$	C_{20} $[kPa]$	C_{30} $[kPa]$	D_k $[MPa^{-1}]$
35.5	3.2	1.9	10^{-5}

Tabla 3.1: Parámetros del material tipo YEOH para el modelo 2D de la córnea.

Figura 3.3: Mallado del modelo de la córnea.

En este modelo (Fig. 3.3) aplicaremos control de fuerzas. Las fuerzas aplicadas estarán entorno a la presión ocular, 1 kPa. Se realizarán 5 simulaciones con cargas entre 0.5 kPa y 2.5 kPa. Esta carga será aplicada en un intervalo temporal de 1 segundo, el cual se discretizará en instantes de tiempo de 0.01 segundos ($\Delta t = 0{,}01s$), por lo que tendremos 101 instantes de tiempo para todo el intervalo.

Una vez realizadas las simulaciones se escoge un nodo para extraer los resultados necesarios para el entrenamiento de la red. El nodo del que extraeremos los datos de la evolución temporal de las diferentes variables será el que este expuesto al mayor desplazamiento (Fig. 3.4), para hacer el análisis en el punto más crítico.

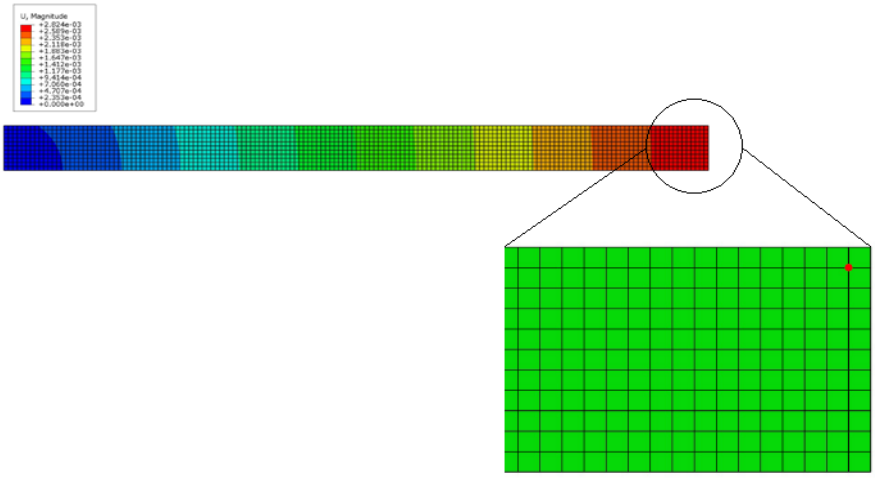

Figura 3.4: Nodo escogido del modelo para la extracción de datos.

Las variables de estado escogidas, que no únicas, para sacar la expresión de GENERIC de este problema serán las siguientes:

- Desplazamientos en X e Y. (U1 y U2 en Abaqus, respectivamente.)
- Velocidades en X e Y. (Calculadas a partir de los desplazamientos.)
- Tensiones principales en X e Y. (S11 y S22 en Abaqus, respectivamente.)
- Tensión tangencial. (S12 en Abaqus.)

– Energía total de deformación elástica (ELSE en Abaqus.)

Por lo tanto, $\mathbf{z}$ quedaría como:

$$\mathbf{z} = (\mathbf{u}, \mathbf{v}, e, \boldsymbol{\sigma}, \tau) \in (\mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R} \times \mathbb{R}^2 \times \mathbb{R}) \tag{3.3}$$

Las matrices $\mathbf{M}$ y $\mathbf{L}$, matrices de fricción y Poisson respectivamente, y los gradientes de entropía (DS) y energía (DE) serán optimizados por la red. Sabemos que el material tiene un comportamiento hiperelástico por lo que se podría haber calculado solo el potencial conservativo pero se ha decidido aprender ambos suponiendo que no conocemos estas características. Con esto ya se tiene descrito el problema con el formalismo GENERIC.

3.1.2. Extracción de datos.

Una vez terminadas las simulaciones en el programa Abaqus, se realiza la extracción de datos. Las variables de estado que necesitamos extraer para este problema son los desplazamientos en las direcciones x e y, velocidades en estas mismas direcciones, las tensiones principales, la tensión tangencial y la energía. Seis de estas ocho variables las podemos coger directamente de Abaqus y las dos velocidades las calcularemos como velocidad en ese punto según la ecuación:

$$V_m = \frac{\Delta x}{\Delta t} \tag{3.4}$$

Mediante un script de pyhton se extraen los datos del programa Abaqus en un archivo de texto. Con la ayuda del programa MatLab se preparan los datos en un formato válido para Python, lenguaje empleado en la programación de la red. Estos datos que estaban en "crudo" los tendríamos finalmente en formato de matrices:

$$\begin{bmatrix} | & | & & | \\ \mathbf{z_1} & \mathbf{z_2} & \cdots & \mathbf{z_n} \\ | & | & & | \end{bmatrix} = \mathbf{Z} \in (\mathbb{R}^{D \times n}) \tag{3.5}$$

Donde $\mathbf{z_n}$ será el vector de variables de estado en el instante de tiempo n y D es el numero de variables de estado de nuestro problema.

Como paso previo en el procesamiento de los datos aplicamos una normalización de estos con el objetivo de tener rangos homogéneos de los valores de las diferentes

variables de estado para que aprenda todas por igual y filtrar el posible ruido que tuvieran. La normalización que se utiliza es de 0 a 1 y desviación 0 de las diferentes variables.

En total tenemos 5 simulaciones, cada una discretizada en 101 instantes de tiempo para los cuales hemos evaluado las variables de estado previamente descritas

3.1.3. Implementación con redes SPNN y resultados.

Una vez obtenidos todos los datos para el entrenamiento de la red estos se dividen en dos subgrupos. Para ello se seleccionan 4 simulaciones de entrenamiento y 1 de validación de manera aleatoria. Además, de las simulaciones seleccionadas para el entrenamiento, 80 % de los instantes de tiempo son para el entrenamiento de la red, y 20 % de test para controlar el aprendizaje.

El tamaño de la entrada para la red será el número de variables de estado ($dim_{in} = 8$), el tamaño de salida de la red viene dada por la expresión $dim_{out} = dim_{in} \cdot (dim_{in} + 3) = 88$, el número de capas ocultas será de $N_{Ocultas} = 6$ con 88 neuronas cada capa, la función de activación para las capas ocultas será del tipo ReLU y lineal en la capa de salida. La red se inicializa según el método de Kaiming [21] con una distribución normal y el optimizador usado es el Adam [22] con una tasa de aprendizaje (Learning rate) $L_r = 10^-5$ y el peso (Weight decay) $\lambda_r = 10^-5$.

Dando como entrada a la red el vector de variables de estado en el primer instante de tiempo los resultados para este caso 2D son los presentados en las siguientes figuras 3.5 y 3.6.

Como se puede observar en la tabla 3.2 los errores medios cuadráticos para estas reproducciones temporales de las variables de estado son bajos, teniendo un mínimo de 5.86E-06 y un máximo de 2.10E-03 al final de la simulación solamente en la energía para el entrenamiento y un mínimo de 7.86E-06 y un máximo de 2.90E-03 para la prueba de validación.

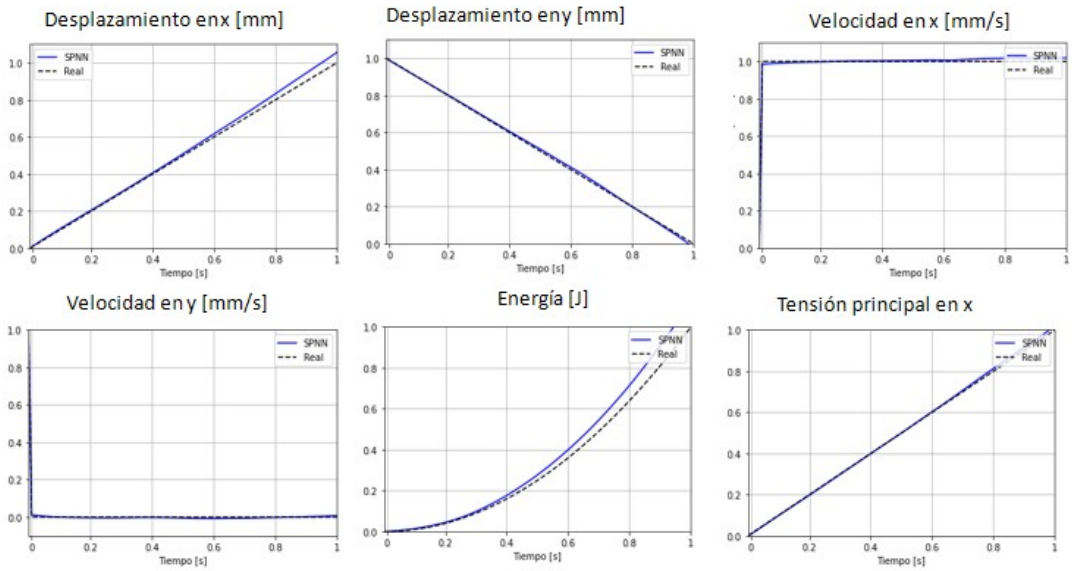

Figura 3.5: Gráfica de la evolución temporal de las variables de estado de la red frente a la realidad en el entrenamiento.

Variable	MSE Simulación de entrenamiento	MSE Simulación de Validación
u_x	6.00E-04	6.00E-04
u_y	6.04E-05	6.00E-05
v_x	1.00E-04	1.00E-04
v_y	2.15E-05	2.15E-05
e	2.10E-03	2.90E-03
σ_x	6.19E-05	6.19E-05
σ_y	5.86E-06	7.86E-06
τ_{xy}	5.00E-04	5.00E-04

Tabla 3.2: Error medio cuadrático para cada variable de estado en el entrenamiento y en la validación.

En la figura 3.7 podemos ver el diagrama de caja la comparación gráfica de los resultados para el entrenamiento como para la prueba de validación, se observa que los errores de las simulaciones de entrenamiento y validación son muy similares para ambas reproducciones debido a que ambos conjuntos pertenecen a la misma distribución y tienen la misma complejidad.

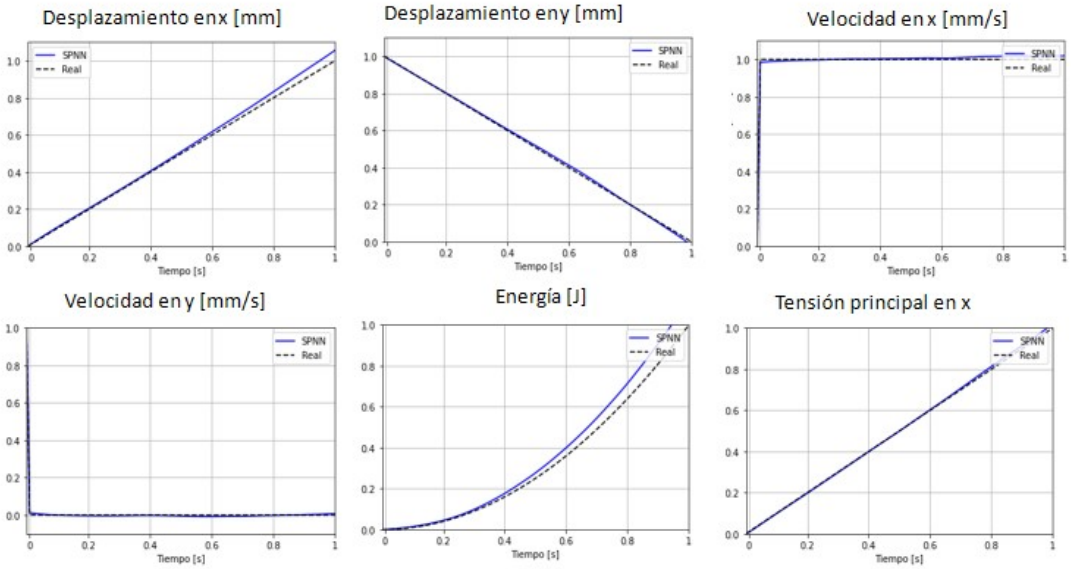

Figura 3.6: Gráfica de la evolución temporal de las variables de estado de la red frente a la realidad en la validación.

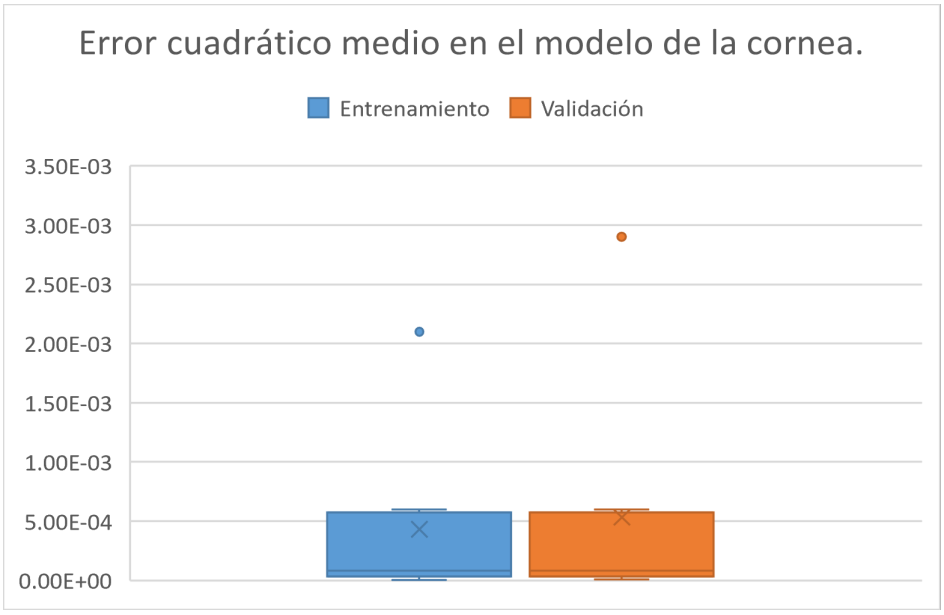

Figura 3.7: Diagrama de caja del error medio cuadrático para los resultados del entrenamiento y de la validación.

3.2. Aplicación de la red SPNN en un modelo hiperelástico de la capa adventicia de la arteria aorta ilíaca.

En esta sección aplicaremos las ya mencionadas redes SPNN a un caso de material biológico blando como es la capa adventicia de la arteria aorta ilíaca. Explicaremos como son las simulaciones que usamos para conseguir los datos, la extracción y "limpieza" de estos y luego su implementación en la red en diferentes modelos.

3.2.1. Descripción del modelo.

En este segundo modelo en el que se implementarán las redes SPNN se modelará el comportamiento de la capa adventicia de la arteria ilíaca humana (Fig. 3.8). Este problema ha sido analizado numéricamente por Gasser, Holzapfel y Ogden [25] y se han extraído los modelos de los recursos públicos de Abaqus [26].

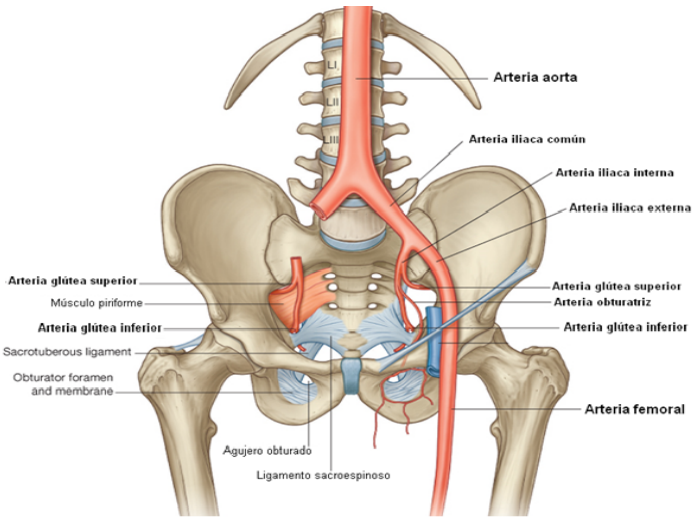

Figura 3.8: Ubicación de la arteria ilíaca humana (Fuente: Internet).

Las arterias poseen diferentes capas como podemos ver en la figura 3.9 y la capa adventicia es la más exterior de estas. Para este modelo se ha considerado un ensayo de tracción uniaxial en muestras en diferentes direcciones, a lo largo de la dirección axial, en la dirección circunferencial y con un ángulo de 15^{o} respecto a la dirección circunferencial como podemos ver en la figura 3.10.

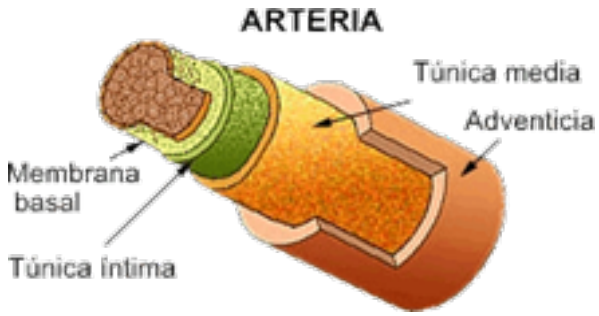

Figura 3.9: Diferentes capas de una arteria (Fuente: Internet).

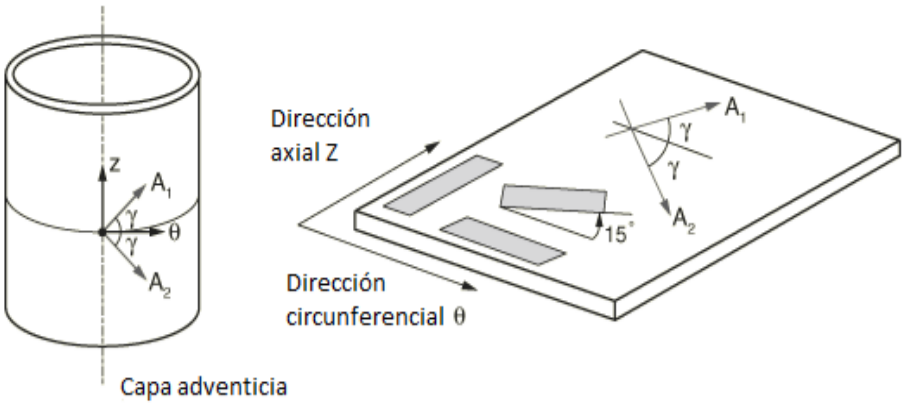

Figura 3.10: Capa adventicia con las diferentes direcciones principales de fibras de colágeno (izquierda) y las diferentes orientaciones de las muestras ensayadas (Fuente: [25]).

Estas muestras son rectangulares y tienen unas dimensiones de 10 mm de largo, 3 mm de ancho y 0.5 mm de grosor y están libres de tensiones en la configuración inicial. Hay dos familias de fibras de colágeno embebidas en las muestras dispuestas simétricamente respecto a las direcciones axial y circunferencial como podemos ver en la figura 3.10. El ángulo entre la orientación media de las fibras y la dirección circunferencial es $\gamma = 49{,}98^o$. Las probetas se cargan en la dirección longitudinal y la deformación en las caras está restringida.

Para las muestras en dirección axial y circunferencial se modela un octavo de la geometría gracias a las condiciones de contorno impuestas (Fig. 3.11) y para la muestra cortada a 15^{o} se crea una geometría completa (Fig. 3.12).

Para modelar la capa adventicia se supone compuesta por dos familias de fibras de colágeno embebidas en una matriz blanda e incompresible. Ambas familias tienen orientaciones medias caracterizada por los vectores $\vec{A1}$ y $\vec{A2}$ en la configuración de referencia, pero las orientaciones de las fibras dentro de cada familia están dispersas. El modelo incorpora el parámetro κ $(0 \leq \kappa \leq 1/3)$ que caracteriza el nivel de dispersión

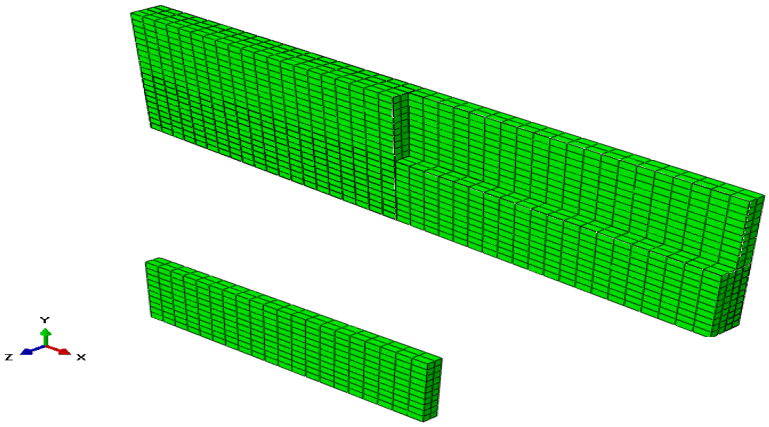

Figura 3.11: Modelo de un octavo de la geometría de la probeta para el ensayo de tracción.

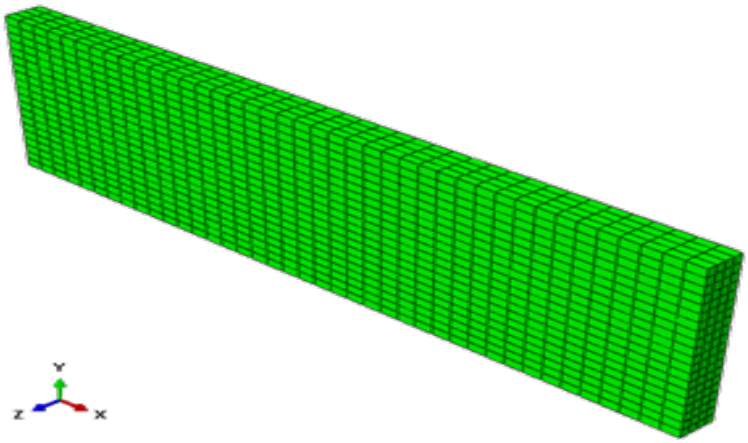

Figura 3.12: Modelo completo de la probeta para el ensayo de tracción.

de la orientación de las fibras de colageno. Por lo tanto, cuando $\kappa = 0$ las fibras están completamente alineadas y cuando $\kappa = \frac{1}{3}$ las fibras se distribuyen aleatoriamente (material isótropo).

Las propiedades del material se han tomado del trabajo de Gasser, Holzapfel y Ogden [25] (ver en la tabla 3.3).

La forma del potencial de energía de deformación para modelar capas arteriales con orientaciones de fibras de colágeno distribuidas es:

$$U = C_{10}(\bar{I}_1 - 3) + \frac{1}{D}(\frac{(J^{el})^2 - 1)}{2} - lnJ^{el}) + \frac{k_1}{2k_2}\sum_{\alpha=1}^{N}\{exp[k_2\langle\bar{E}_\alpha\rangle^2]\}, \tag{3.6}$$

con

$$\bar{E_\alpha} = \kappa(\bar{I}_1 - 3) + (1 - 3\kappa)(\bar{I}_{4(\alpha\alpha)} - 1), \tag{3.7}$$

donde U es la energía de deformación por unidad de volumen de referencia, C_{10}, D, κ_1, κ_2 y κ son parámetros del material que depende de la temperatura, N es el numero de familias de fibras ($N \leq 3$); $\bar{I}_1$ es el primer invariante de deformación desviadora; J^{el} es la relación de volumen elástica en expansión térmica y $\bar{I}_{4(\alpha\alpha)}$ son los pseudo-invariantes de $\bar{\mathbf{C}}$ parte distorsionada del tensor Cauchy-Green por la derecha.

C_{10}	3.82 kPa
k_1	996.6 kPa
k_2	524.6
κ	0 - 0.333
D	0

Tabla 3.3: Parámetros del material tipo Holzapfel para el modelo de la capa adventicia de la arteria ilíaca.

Para la generación de la base de datos necesaria para la red se han hecho cinco simulaciones con diferentes cargas entre 1.0 N y 5.0 N, cada una de estas simulaciones tiene una duración de 1 segundo discretizado en intervalos de tiempo de 0.01 segundos ($\Delta t = 0{,}01s$) por lo que tendremos un total de 101 instantes de tiempo.

El problema se describe a través de las siguientes variables de estado: desplazamientos en las tres direcciones principales, velocidades en las tres direcciones principales, energía de deformación, las tres tensiones principales y las tres tensiones tangenciales. Con esta aproximación queremos caracterizar no solo el comportamiento hiperelástico del material, sino también posibles desviaciones de dicho comportamiento. A pesar de considerarlo un tejido de estas características, los tejidos biológicos vivos reales no suelen ajustarse perfectamente a estos comportamientos. Por eso, aprenderemos los dos potenciales, el referido a la conservación de la energía y el de disipación suponiendo que no conocemos su condición hiperelástica.

Por tanto $\mathbf{z}$ quedará como:

$$\mathbf{z} = (\mathbf{u}, \mathbf{v}, e, \boldsymbol{\sigma}, \boldsymbol{\tau}) \in (\mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R} \times \mathbb{R}^3 \times \mathbb{R}^3) \tag{3.8}$$

Además, como para esta elección de variables no tenemos datos en la literatura de las matrices de Poisson ($\mathbf{L}$) y de Fricción ($\mathbf{M}$), no las consideraremos conocidas. Por lo

tanto, la red aprenderá estas dos matrices y el gradiente de entropía (DS) y el gradiente de energía (DE).

3.2.2. Extracción de datos.

Una vez terminadas las simulaciones se aplicará el script de pyhton de extracción de datos explicado anteriormente.

El nodo seleccionado para la extracción de datos será el que mostramos en la figura 3.13 como uno de los puntos críticos del experimento encontrandose en el medio de la muestra. De este nodo extraeremos la evolución temporal de las variables de estado necesarias.

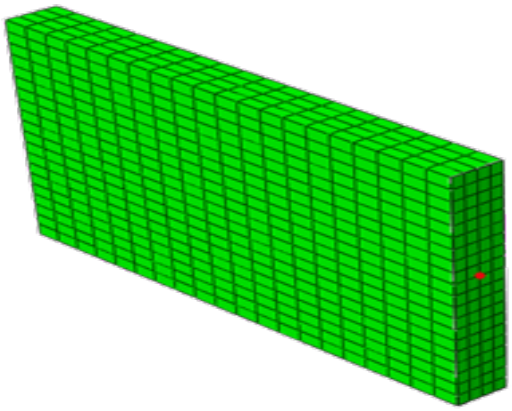

Figura 3.13: Nodo seleccionado para el entrenamiento de la red.

Las variables de estado para este problema, como ya se ha mencionado antes, son los desplazamientos en las tres direcciones, velocidades en estas mismas direcciones, las tensiones principales, las tensiones tangenciales y la energía. Todas las variables las podemos coger directamente del programa Abaqus, excepto las velocidades que las calcularemos como hemos mencionado en el capítulo anterior (Ecuación 3.4).

Seguiremos el mismo método explicado en el apartado 3.2 de este documento para realizar la extracción completa de los datos y conseguir los datos de 5 simulaciones cada cual con 13 variables de estado y una evolución temporal de 101 instantes de tiempo.

3.2.3. Implementación con redes SPNN y resultados.

En los siguientes apartados se explica los parámetros escogidos en cada caso para los diferentes modelos 3D usados y los resultados obtenidos. Para todos los siguientes casos el vector de variables de estado que se ha escogido, ya mencionado antes, es de 13 variables (desplazamientos, velocidades, energía y tensor de tensiones).

Modelo básico.

A continuación, se describirá como se ha implementado la red en el modelo de capa adventicia de la arteria aorta ilíaca. Para tomar un punto de partida se ha escogido la muestra a 15^{o} (ya que es la geometría completa) y la alineación de fibras de $\kappa = 0{,}333$ (material isótropo).

Como en el caso anterior tendremos 4 simulaciones para el entrenamiento y una para la prueba de validación. Además, en las simulaciones de entrenamiento se dividirán los instantes de tiempo, un 80 % para el entrenamiento y el 20 % para la prueba dentro del entrenamiento. Esto se aplicará también a todas las redes para los siguientes modelos.

El tamaño de la entrada para la red, por lo tanto, será el número de variables de estado ($dim_{in} = 13$), el tamaño de salida de la red viene dada por la expresión $dim_{out} = dim_{in} \cdot (dim_{in} + 3) = 208$, el número de capas ocultas será de $N_{Ocultas} = 7$ con 208 neuronas cada capa (esta red es un poco más profunda que para los casos anteriores), la función de activación para las capas ocultas será del tipo ReLU y lineal en la capa de salida. La red se inicializa según el método de Kaiming con una distribución normal y el optimizador usado es el Adam con una tasa de aprendizaje (Learning rate") $L_r = 10^{-}5$ y el peso (Weight decay) $\lambda_r = 10^{-}5$ y las épocas de entrenamiento para este caso serán de 200 épocas.

Las gráficas 3.14 y 3.15 muestran los resultados de la evolución temporal de todas las variables a partir del vector variables de estado en el momento inicial ($t = 0$). Se puede observar que la red aprende bien las tendencias de la evolución temporal para las diferentes variables y se consigue una estabilidad temporal gracias a los sesgos inductivos de estas redes.

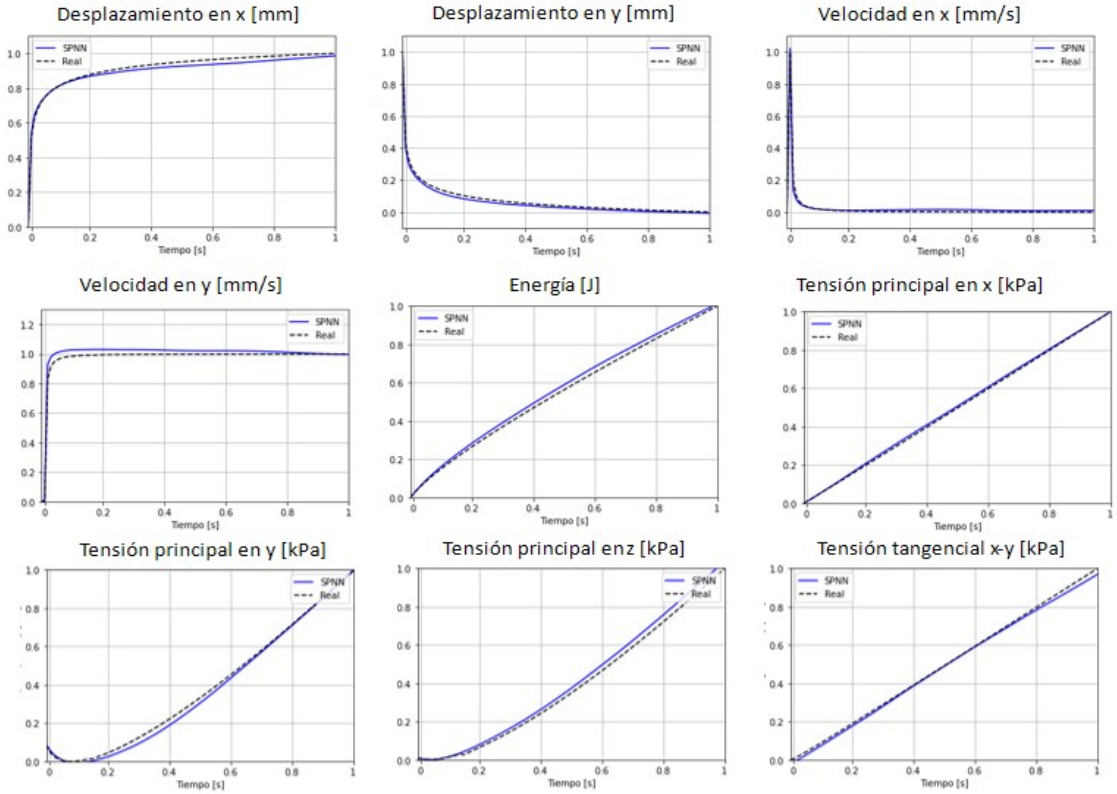

Figura 3.14: Gráficas de la evolución temporal de las variables de estado de la red frente a la realidad en el entrenamiento.

En la tabla 3.4 se puede ver el error cuadrático medio para cada una de las diferentes variables de estado tanto en el entrenamiento como en el test y en la figura 3.16 se ve el diagrama de caja de estos errores. En dicha figura se aprecia a simple vista que los errores de ambas reproducciones se mantienen en un rango bajo. En el entrenamiento el valor de error más alto lo tiene el desplazamiento en x y en el test la velocidad en y, pero estos errores son muy próximos en todas las variables. Cabe destacar que, en la integración de la simulación de entrenamiento sólo se cuenta con un 80 % de la información, y que la simulación de validación es completamente desconocida. Aún así, la red ha aprendido una estructura estable del problema que permite reproducir, además de los casos de entrenamiento, otros completamente nuevos. Esta estabilidad se obtiene gracias a la guía de la física en esta técnica.

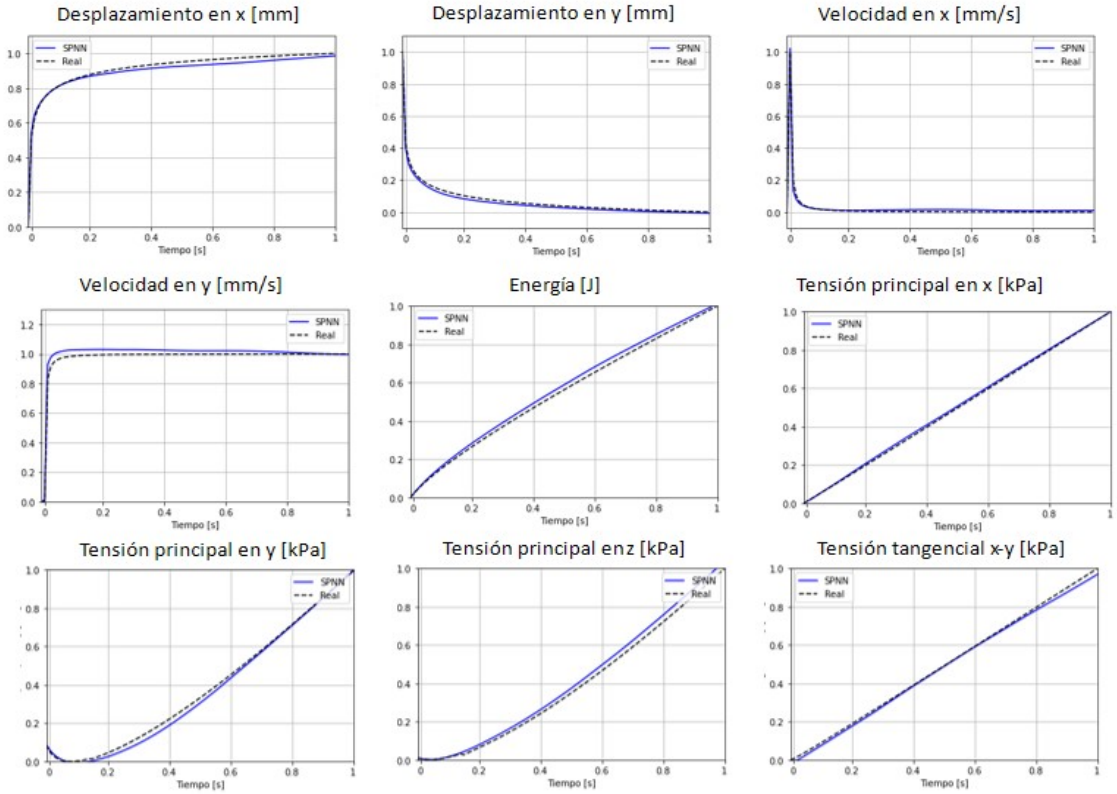

Figura 3.15: Gráficas de la evolución temporal de las variables de estado de la red frente a la realidad en la validación.

Variable	MSE Simulación de entrenamiento	MSE Simulación de Validación
u_x	9.41E-05	4.00E-04
u_y	1.00E-04	2.00E-04
u_z	4.00E-04	6.00E-04
v_x	7.01E-05	1.00E-04
v_y	2.00E-04	8.00E-04
v_z	2.03E-05	4.00E-04
e	2.00E-04	4.00E-04
σ_x	1.00E-04	6.83E-05
σ_y	6.82E-05	4.00E-04
σ_z	6.30E-05	7.00E-04
τ_{xy}	4.00E-04	2.00E-04
τ_{yz}	2.06E-05	2.28E-05
τ_{xz}	1.93E-05	2.00E-04

Tabla 3.4: Error cuadrático medio para cada variable de estado en la prueba de validación para el modelo básico.

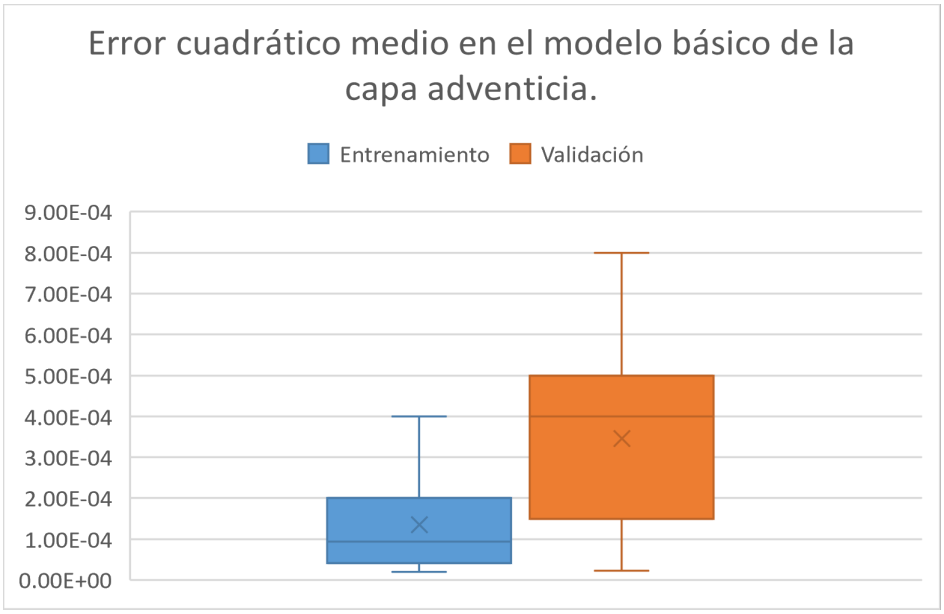

Figura 3.16: Diagrama de caja del error medio cuadrático para los resultados del entrenamiento y de la validación del modelo básico.

En el anexo A se encuentra la implementación y los resultados para otros modelos: diferente valor de κ (anexo A.1), diferente orientación de la muestra (anexo A.2), y para la aplicación de la red a otro nodo de la geometría (anexo A.3). En ellos se puede ver que las redes SPNN son eficaces en diferentes modelos de la capa adventicia.

Modelo con diferentes nodos y misma fuerza.

Para este modelo se hará sólo una simulación con una carga de $1N$ y se cogerán 5 nodos consecutivos como se puede ver en la figura 3.17. Se ha modificado el script de pyhton de extracción de datos para sacar los diferentes nodos y se ha seguido el mismo método relatado anteriormente para extraer los datos y dejarlos limpios para la implementación en la red. Hay que destacar que la red será entrenada con 4 nodos y en la prueba de validación reproducirá uno completamente nuevo a partir del vector de variables de estado en el primer instante de tiempo.

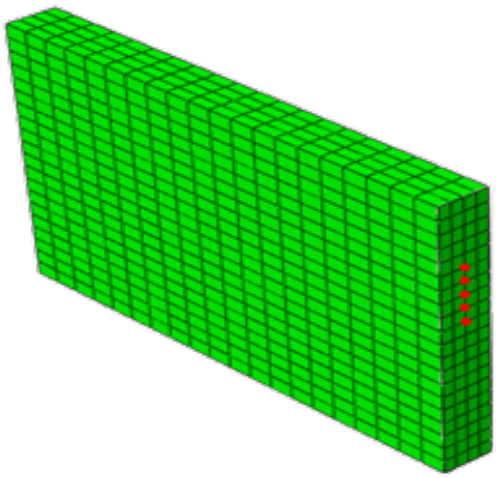

Figura 3.17: Nodos seleccionados para la implementación de la red con diferentes nodos.

El vector de variables de estado será el mismo que en el caso 3D de todos los anteriores modelos y los hiperparámetros de la red serán los siguientes:

- $dim_{in} = 13$
- $dim_{out} = 208$
- $N_{ocultas} = 7$
- $L_r = 10^-5$
- $\lambda_r = 10^-5$
- Épocas de entrenamiento = 200

Las gráficas 3.18 y 3.19 nos muestran que se consigue reproducir fielmente la evolución temporal de las variables de estado tanto para el entrenamiento como para la prueba de validación. Además si analizamos la tabla con el error cuadrático medio por variable

(Tablas 3.5) se observa que el máximo para la prueba de validación se encuentra en la tensión principal en y. Por último el diagrama de caja se puede ver que estos errores están agrupados entorno a $1{,}00E-03$ excepto para la tensión principal en y de la validación que se nos va a valores mas altos ($7{,}60E-03$).

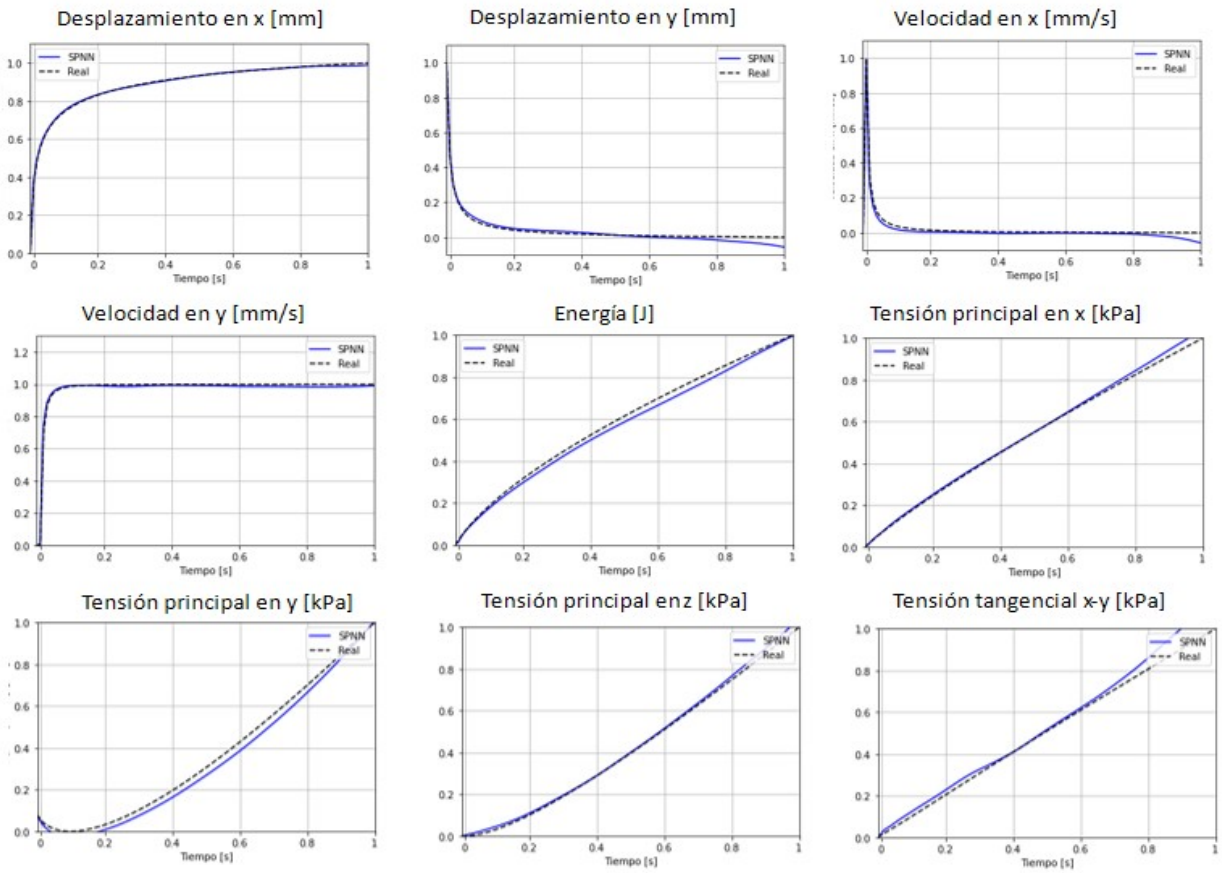

Figura 3.18: Gráficas de la evolución temporal de las variables de estado de la red frente a la realidad en el entrenamiento.

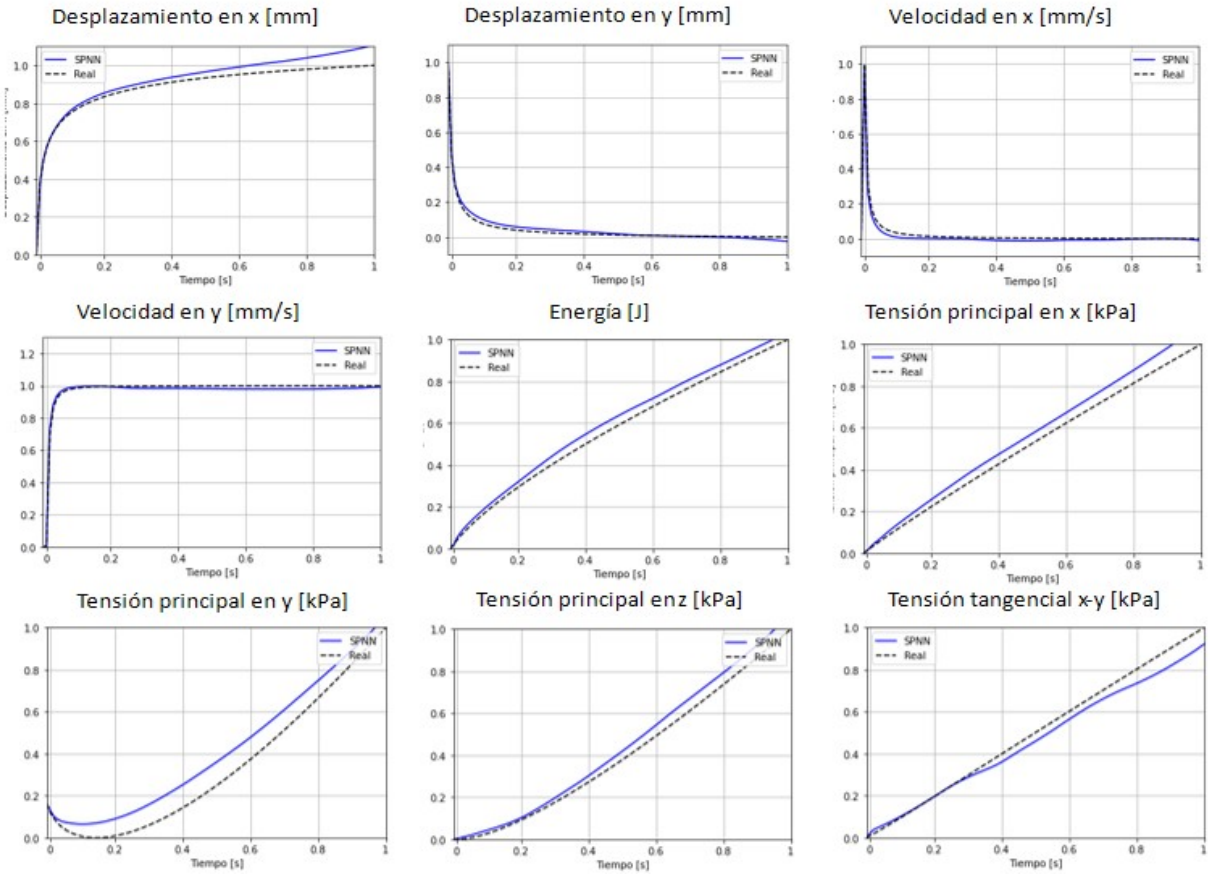

Figura 3.19: Gráficas de la evolución temporal de las variables de estado de la red frente a la realidad en la validación.

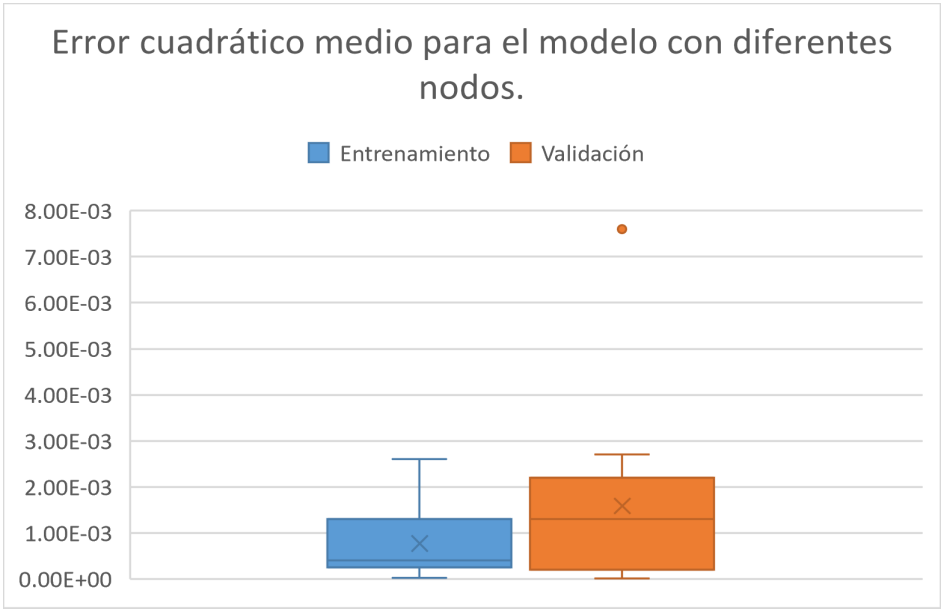

Figura 3.20: Diagrama de caja del error medio cuadrático para los resultados del entrenamiento y de la validación del modelo aplicado a diferentes nodos.

Variable	MSE Simulación de entrenamiento	MSE Simulación de Validación
u_x	1.99E-05	2.20E-03
u_y	4.00E-04	2.00E-04
u_z	1.10E-03	5.10E-06
v_x	3.00E-04	2.00E-04
v_y	3.00E-04	2.00E-04
v_z	3.00E-04	7.54E-05
e	5.00E-04	1.30E-03
σ_x	2.00E-04	2.70E-03
σ_y	9.00E-04	7.60E-03
σ_z	2.00E-04	1.80E-03
τ_{xy}	2.60E-03	2.20E-03
τ_{yz}	1.70E-03	1.70E-03
τ_{xz}	1.50E-03	4.00E-04

Tabla 3.5: Error cuadrático medio para cada variable de estado para el modelo aplicado a diferentes nodos.

Con este último caso se puede afirmar que las redes SPNN tienen una adaptabilidad a entrenar la red con diferentes nodos y reproducir una evolución temporal de otro nodo distinto por la estabilidad que le confiere la física, que lo ha guiado a aprender una estructura correcta del problema estudiado.

Capítulo 4

Aplicación de la red SPNN a un caso pseudo-experimental.

4.1. Descripción del experimento.

Como paso final, se aprenderá un modelo de comportamiento de material suponiendo que solo se tiene acceso a datos pseudo-experimentales y, por ende, a variables observables como si estuviéramos en un ensayo real.

El experimento que se emulará es un ensayo de tracción uniaxial (Fig. 4.1) con diferentes muestras la capa adventicia de la arteria aorta ilíaca. Para esto nos basamos en la simulación del capítulo 3. El objetivo es tener experimentos con muestras a diferentes cargas y cada una con diferentes alineaciones de fibras, $0 \leq \kappa \leq 1/3$, así nos generaremos una base de datos para entrenar la red y comprobar su eficacia en este ensayo.

Figura 4.1: Ejemplo de un ensayo de tracción en un tejido vivo (Fuente: Internet).

Con el fin de que sea una caso más real cambiaremos ligeramente las constantes del material tipo Holzapfel para añadir ruido a nuestro experimento. Las constantes para este caso se pueden ver en la tabla 4.1.

C_{10}	6.21 kPa
k_1	766 kPa
k_2	618
κ	0 - 0.333
D	0

Tabla 4.1: Nuevas constantes del material tipo Holzapfel para el modelo de la capa adventicia de la arteria ilíaca pseudo-experimental.

En este caso adaptaremos el modelo de la capa adventicia de la arteria ilíaca pero las variables de estado que usaremos serán variables observables como si fuera un ensayo de tracción real. Las variables que se han elegido, por lo tanto, son la deformación nominal y la tensión principal en lo ejes x e y. Por lo que $\mathbf{z}$ queda como:

$$\mathbf{z} = (\boldsymbol{\epsilon}, \boldsymbol{\sigma}) \in (\mathbb{R}^2 \times \mathbb{R}^2) \tag{4.1}$$

Para la elección del nodo y la extracción de datos se ha procedido igual que en lo ya relatado en el capítulo 3. Las matrices de Poisson ($\mathbf{L}$) y de Fricción ($\mathbf{M}$), el gradiente de energía (DE) y el gradiente de entropía (DS) son objeto de la optimización. En este caso, por la selección de variables de estado que hay disponibles, perdemos información y por tanto no se podría hacer una descripción hiperelástica. La falta de información se traduce en disipación por el teorema de fluctuación-disipación, apareciendo por tanto el potencial disipativo [27].

4.2. Implementación con redes SPNN y resultados.

4.2.1. Modelo básico con variables observables.

Primero emularemos el ensayo en el modelo básico del capítulo 3 con las variables observables para confirmar que estas variables sirven para entrenar la red. Por lo tanto, como en el caso básico tendremos 5 simulaciones con diferentes cargas, cuatro de ellas las usaremos para el entrenamiento y una para la prueba de validación.

La entrada de la red será el número de variables de estado ($dim_{in} = 4$), la dimensión de salida $dim_{out} = 28$, el número de capas ocultas $N_{Ocultas} = 4$. La función de activación para las capas ocultas será del tipo ReLU y lineal en la capa de salida. La red se inicializa según el método de Kaiming con una distribución normal y el optimizador usado es el Adam con una tasa de aprendizaje (Learning rate) $L_r = 10^{-5}$ y el peso (Weight decay) $\lambda_r = 10^{-5}$ y las épocas de entrenamiento $N épocas = 200$.

Las gráficas 4.2 y 4.3 muestran los resultados de la evolución temporal de todas las variables a partir del vector variables de estado en el momento inicial ($t = 0$), se observa que la red reproduce las diferentes variables eficazmente pero con mayor error que para la definición completa del capítulo anterior.

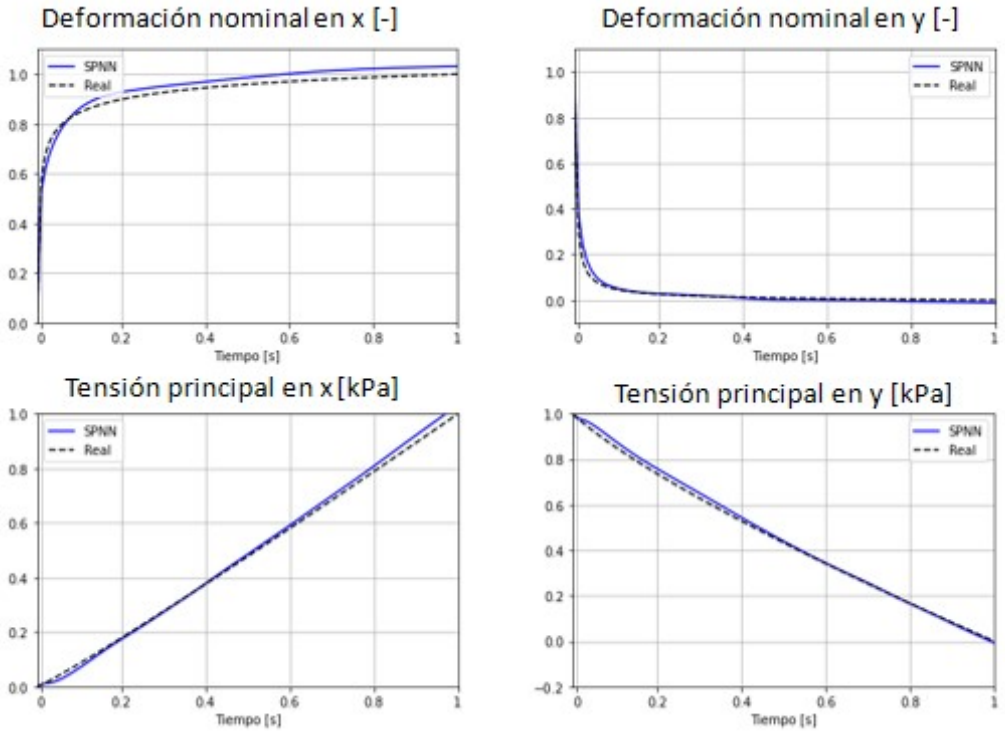

Figura 4.2: Gráficas de la evolución temporal de las variables de estado de la red frente a la realidad en el entrenamiento.

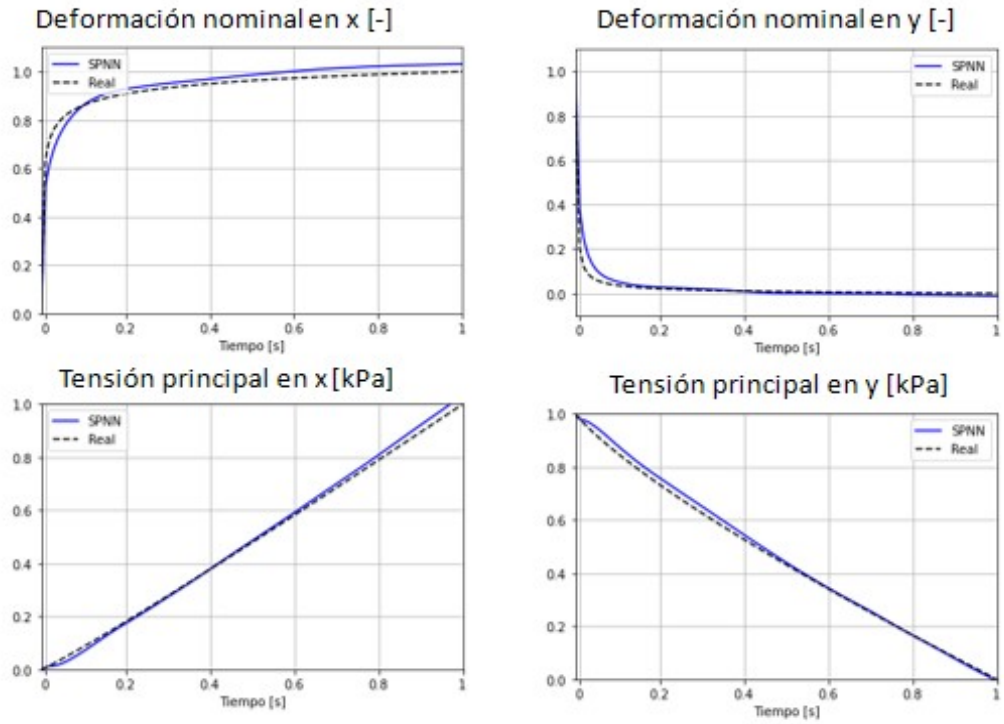

Figura 4.3: Gráficas de la evolución temporal de las variables de estado de la red frente a la realidad en la validación.

En la tabla 4.2 y en la figura 4.4 se pueden ver el error cuadrático medio para el entrenamiento y la validación. Se aprecia que estos valores son bajos y son mejores para el entrenamiento y que en ambos casos los máximos se dan para la deformación nominal en la dirección x.

Variable	MSE Simulación de entrenamiento	MSE Simulación de Validación
ϵ_x	9.00E-04	1.00E-03
ϵ_y	2.00E-04	7.00E-04
σ_x	2.00E-04	2.00E-04
σ_y	2.00E-04	3.00E-04

Tabla 4.2: Error cuadrático medio para cada variable de estado para el modelo básico con variables observables.

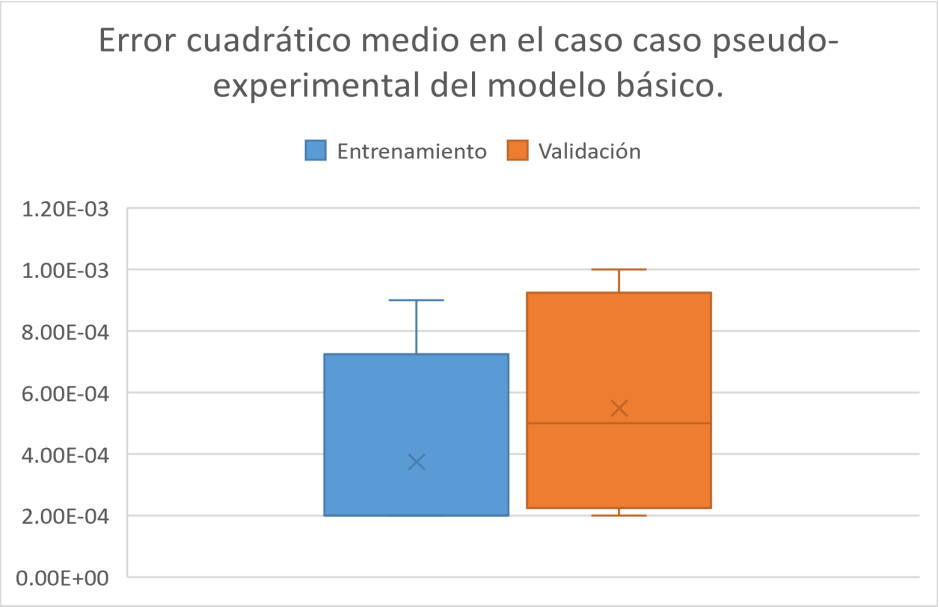

Figura 4.4: Diagrama de caja del error medio cuadrático para los resultados del entrenamiento y de la validación del modelo básico con variables observables.

4.2.2. Modelo con diferentes κ y diferentes cargas.

A continuación, se ampliará la variabilidad de las muestras eligiendo aquellas con una orientación de 15^{o} pero con κ (alineación de fibras) comprendida entre $\kappa = 0{,}270$ y $\kappa = 0{,}333$, por tanto, la red tiene que aprender la variabilidad procedente de dos parámetros en vez de uno solo como en casos anteriores. Se harán un total de 9 simulaciones con diferentes cargas entre $1N$ y $3N$, ocho de ellas las usaremos para entrenar la red y una para la prueba de validación. Además, se coge el 80 % de instantes de tiempo para el entrenamiento y un 20 % para el test.

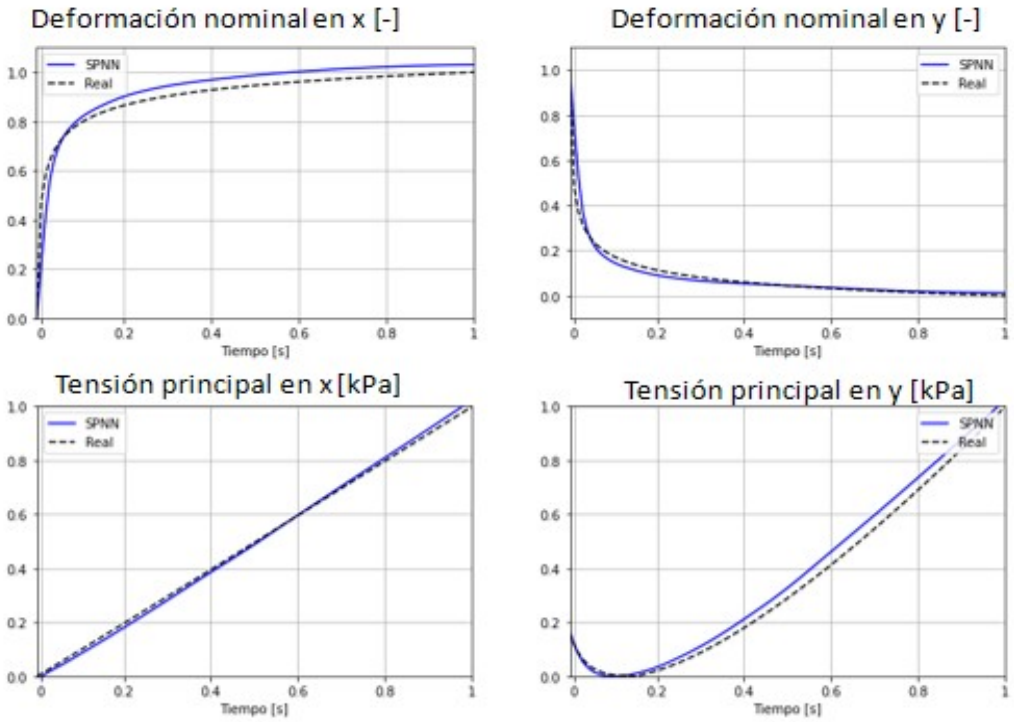

Figura 4.5: Gráficas de la evolución temporal de las variables de estado de la red frente a la realidad en el entrenamiento.

Los hiperparámetros para la red serán:

- $dim_{in} = 4$
- $dim_{out} = 28$
- $N_{ocultas} = 7$
- $L_r = 10^{-5}$
- $\lambda_r = 10^{-5}$
- Épocas de entrenamiento = 200

Los resultado de la evolución temporal para este modelo se pueden ver en las figuras 4.5 y 4.6 para el entrenamiento y la validación respectivamente. En ellos se ve que para este caso más general los resultados siguen siendo aceptables en ambos casos.

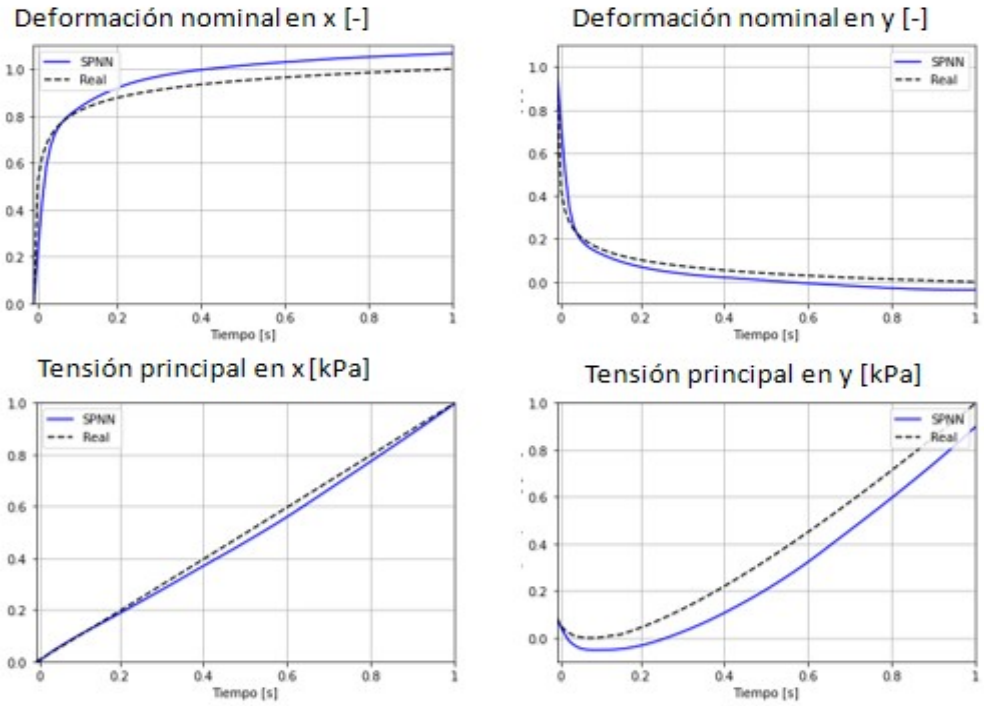

Figura 4.6: Gráfica de la evolución temporal de las variables de estado de la red frente a la realidad en la validación.

En la tabla 4.3 y el diagrama de caja (Fig. 4.7) se puede ver el error cuadrático medio de cada una de las variables de estado. En este caso se ve que para el entrenamiento los resultados son mejores especialmente en la tensión principal en y debido a que es la variable que más dispersión tiene entre simulaciones. Aun así los errores se mantienen bajos y se reproduce la evolución temporal eficazmente. Estos resultados reflejan que, para una muestra de un paciente nuevo para el que no sabemos su distribución de fibras, podríamos reproducir el ensayo completo para distintas cargas.

Variable	MSE Simulación de entrenamiento	MSE Simulación de Validación
ϵ_x	2.30E-03	4.70E-03
ϵ_y	1.10E-03	2.50E-03
σ_x	2.00E-04	5.00E-04
σ_y	1.20E-03	1.09E-02

Tabla 4.3: Error cuadrático medio para cada variable de estado para el modelo pseudo-experimental con diferentes κ.

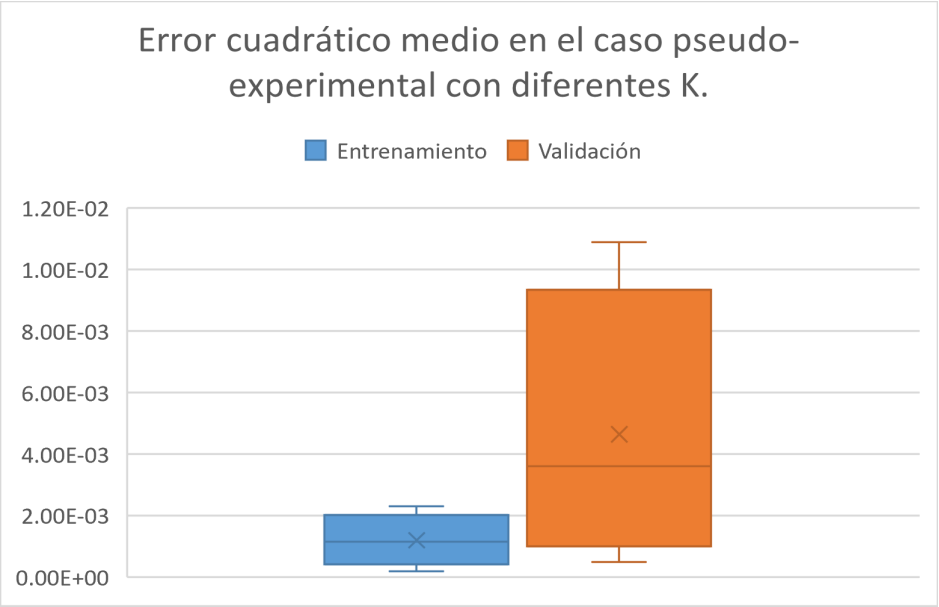

Figura 4.7: Diagrama de caja del error medio cuadrático para los resultados del entrenamiento y de la validación del modelo pseudo-experimental con diferentes κ.

En estos ejemplos podemos ver la adaptabilidad de las redes SPNN a un caso con pocos datos, condiciones que suele darse en casos reales. Con mayor disponibilidad de datos podríamos ir aumentado en la complejidad del experimento cambiando más variables y haciendo el problema más genérico.

Capítulo 5

Conclusiones y lineas futuras.

En el capítulo final haremos un análisis de los resultados frente a otros tipos de redes y sacaremos las conclusiones más relevantes del proyecto, así como las líneas futuras de trabajo.

5.1. Comparativa de redes.

Uno de los objetivos era el análisis y comparación de redes guiadas por sesgos inductivos frente a otros modelos que no incorporen ese conocimiento. Por ello realizamos una comparativa con otras dos redes:

- En la primera, red sin función de pérdida por degeneración, quitaremos al modelo básico, explicado en el capítulo 3, el cálculo de la función de pérdida de las condiciones de degeneración, así como su aportación a la función de pérdida total
- En la segunda, red "Caja negra" quitaremos toda la implementación del formalismo GENERIC a la red neuronal.

Para estas 2 redes se han utilizado los mismos hiperparámetros, misma base de datos y mismas épocas de entrenamiento que en el modelo básico visto en el capitulo 3.2 para que la comparación sea en condiciones exactas para los tres casos.

Los resultados obtenidos para estas dos nuevas redes y comparados con el modelo básico se encuentra en la figura 5.1

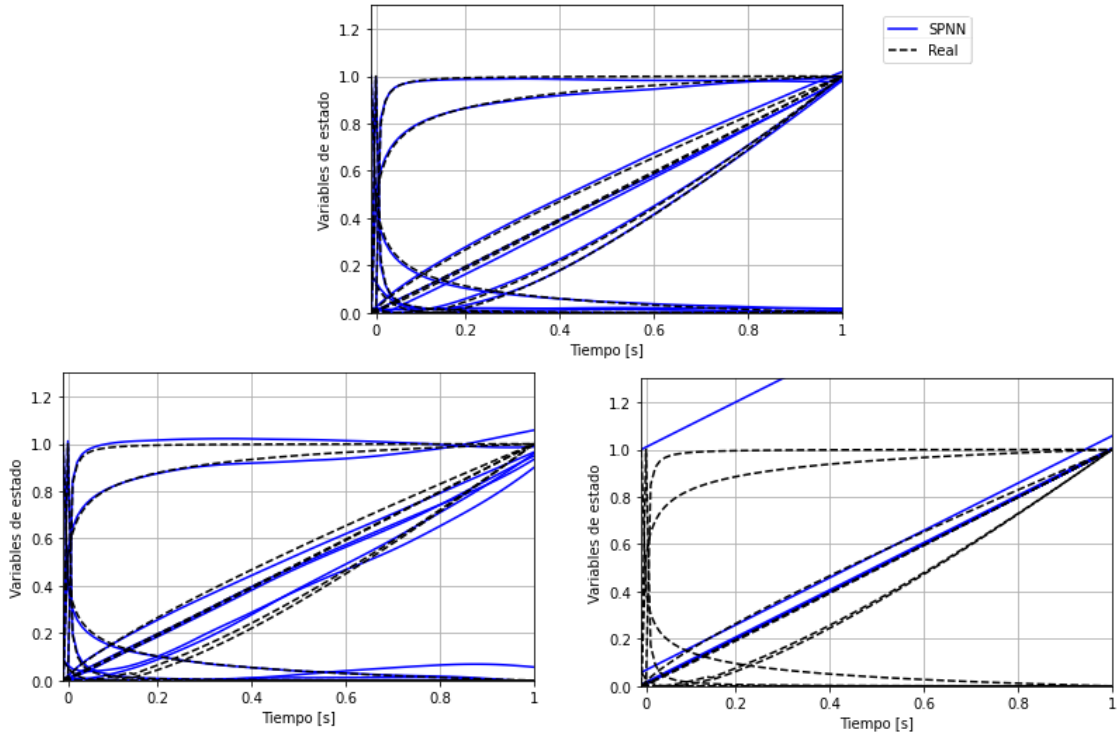

Figura 5.1: Gráficas de la evolución temporal de las variables de estado de la red SPNN (arriba) frente a la red sin pérdida por degeneración (abajo izquierda) y la red caja negra (abajo derecha).

En la tabla 5.1 podemos ver el error medio cuadrático de las tres diferentes redes en todas las variables de estado en el entrenamiento y en la tabla 5.2 para la validación.

En las figuras 5.3 y 5.4 están los diagramas de caja del error cuadrático medio para las tres redes, con esto se puede ver a simple vista que la red caja negra, con estos parámetros, no consigue reproducir la evolución temporal de las diferentes variables y solo aprende una evolución en linea recta de todas las variables dando por ello unos valores de error muy altos. También podemos ver en el detalle de los diagramas de caja de la red SPNN frente a la red sin la aportación de la pérdida por degeneración que hay valores que tienen mucho error. Esto es debido a que, conforme quitamos información, la red necesita más datos, mayor profundidad y mayor ajuste para alcanzar los mismos niveles de precisión. Además, como se puede ver en la figura 5.2 el gradiente de energía se mantiene en torno a cero y el gradiente de entropía se mantiene positivo en toda su evolución temporal del modelo básico. Esto representa el cumplimiento de las restricciones termodinámicas a lo largo de toda la simulación gracias a los sesgos inductivos de las redes SPNN.

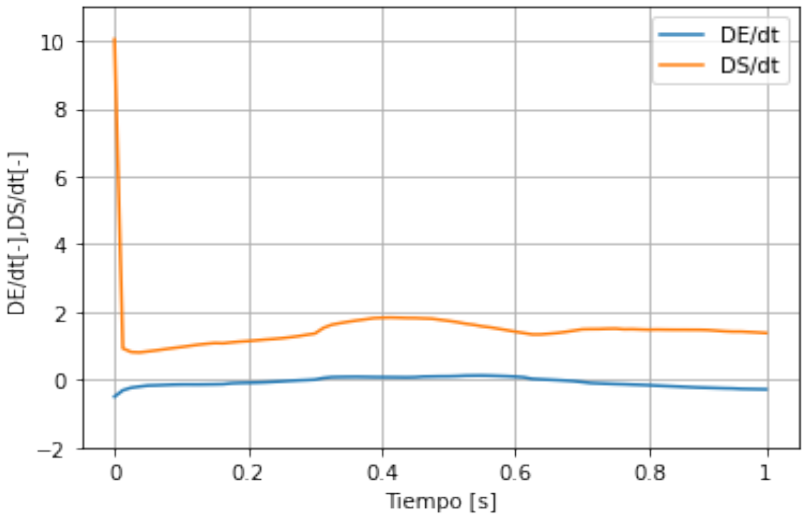

Figura 5.2: Evolución temporal del gradiente energía y entropía.

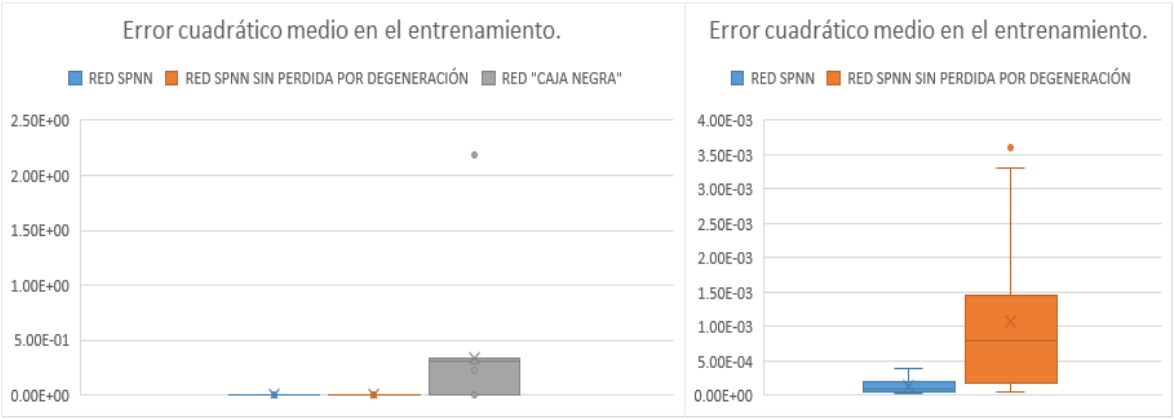

Figura 5.3: Diagramas de caja para del error cuadrático medio en el entrenamiento en los tres tipos de redes.

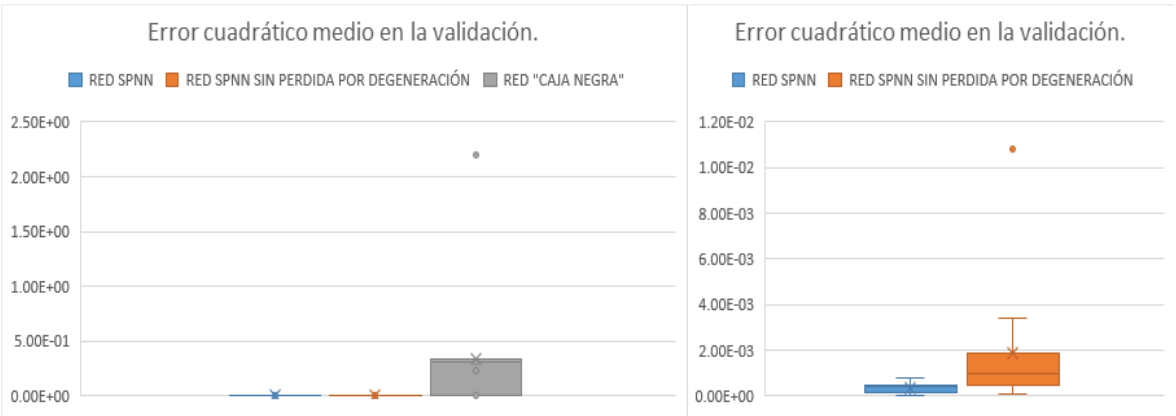

Figura 5.4: Diagramas de caja para del error cuadrático medio en la prueba de validación en los tres tipos de redes.

Variable	MSE Red SPNN	MSE Red sin degeneración	MSE Red caja negra
u_x	9.41E-05	4.00E-04	2.29E-01
u_y	1.00E-04	4.69E-05	2.19E+00
u_z	4.00E-04	5.47E-05	3.35E-01
v_x	7.01E-05	6.79E-05	3.44E-01
v_y	2.00E-04	3.00E-04	3.06E-01
v_z	2.03E-05	3.30E-03	3.35E-01
e	2.00E-04	8.00E-04	2.40E-03
σ_x	1.00E-04	1.30E-03	9.14E-06
σ_y	6.82E-05	3.60E-03	2.81E-02
σ_z	6.30E-05	7.00E-04	1.23E-02
τ_{xy}	4.00E-04	1.00E-03	4.75E-05
τ_{yz}	2.06E-05	8.00E-04	3.35E-01
τ_{xz}	1.93E-05	1.60E-03	3.35E-01

Tabla 5.1: Error medio cuadrático para cada variable de estado en el entrenamiento en los tres tipos de redes.

Variable	MSE Red SPNN	MSE Red sin degeneración	MSE Red caja negra
u_x	4.00E-04	6.00E-04	2.33E-01
u_y	2.00E-04	2.00E-04	2.19E+00
u_z	6.00E-04	8.00E-04	3.35E-01
v_x	1.00E-04	1.00E-04	3.44E-01
v_y	8.00E-04	1.00E-03	3.07E-01
v_z	4.00E-04	1.08E-02	3.35E-01
e	4.00E-04	3.00E-04	2.30E-03
σ_x	6.83E-05	8.00E-04	8.19E-06
σ_y	4.00E-04	3.40E-03	2.02E-02
σ_z	7.00E-04	1.10E-03	1.04E-02
τ_{xy}	2.00E-04	1.80E-03	3.43E-05
τ_{yz}	2.28E-05	1.50E-03	3.35E-01
τ_{xz}	2.00E-04	2.00E-03	3.35E-01

Tabla 5.2: Error medio cuadrático para cada variable de estado en la prueba de validación en los tres tipos de redes.

5.2. Conclusiones.

Las conclusiones obtenidas de este trabajo son las siguientes:

- La principal conclusión a la que llegamos es que se han podido implementar las redes SPNN en tejidos biológicos blandos obteniendo por tanto una reproducción fiel de la evolución temporal de las variables de estado que describían cada modelo.

- Se ha comprobado que la implementación de GENERIC en las redes SPNN mejora los resultados en un orden muy considerable frente a modelos tradicionales como el de caja negra debido a que las redes SPNN aprenden la física del problema dando interpretabilidad y generalidad a los modelos aprendidos algo especialmente deseable en al ámbito de la bioingeniería.

- La implementación de las redes SPNN realizada es lo suficientemente robusta como para adaptarse a diferentes modelos (Anexo A), modelo de córnea y modelo de arteria ilíaca. Además, se ha comprobado que la red es capaz de abordar dos diferentes problemas, el primero, entrenar con diferentes fuerzas y reproducir otra diferentes y el segundo, ser entrenada con diferentes nodos para la misma fuerza y reproducir otro nodo diferente.

- Se ha implementado la red en datos que imitan a los experimentales, obteniendo resultados satisfactorios para, con datos procedentes de un paciente para el cual no sabemos la dispersión de las fibras, reproducir la respuesta mecánica del tejido.

- En los modelos pseudo-experimentales cuanto mayor dispersión hay en los datos, se necesita una mayor profundidad en la red y mayor numero de épocas para poder llegar a resultados favorables. Así como un mayor número de datos para aprender correctamente las estructuras de la mecánica subyacente. Sin embargo, gracias a los sesgos inductivos, se consigue aprender el modelo con un margen de error menor que si no se aplicaran las restricciones físicas.

5.3. Líneas futuras.

Este proyecto ha implementado las redes SPNN en tejidos biológicos vivos a partir de datos que provienen de simulaciones computacionales. Una de las líneas futuras de trabajo es entrenar las redes SPNN con datos experimentales para predecir comportamientos de ensayos reales, o probar esta implementación entrenada con datos computacionales para predecir un ensayo real. Además, se podría entrenar la red con más datos computacionales para mejorar los resultados y para abordar casos específicos con una mejor aproximación, especialmente aquellos que presentan una mayor dispersión en su comportamiento. Por ejemplo, se podrían añadir más datos a los casos pseudo-experimentales variando cargas, orientación de la muestra y alineación de las fibras.

Otra linea de trabajo sería relacionar las medidas parciales con el modelo completo, así entrenando la red con una menor cantidad de datos se podría llegar a recuperar más información del material.

Otra posibilidad interesante sería aplicar técnicas de reducción de orden para poder reproducir la evolución temporal de toda la geometría del modelo con un coste computacional menor que aplicando las redes directamente a todo el modelo. Además, a través de estos métodos, se conseguiría mejorar la robustez y la estabilidad del esquema de entrenamiento ya que, así, cambios en la entrada original no condicionarían de una manera tan fuerte la simulación.

Capítulo 6

Bibliografía

[1] H. C. Öttinger and M. Grmela, "Dynamics and thermodynamics of complex fluids. i. development of a general formalism," *Phys. Rev. E*, vol. 56, pp. 6620–6632, Dec 1997.

[2] H. C. Öttinger and M. Grmela, "Dynamics and thermodynamics of complex fluids. ii. illustrations of a general formalism," *Phys. Rev. E*, vol. 56, pp. 6633–6655, Dec 1997.

[3] Q. Hernández, A. Badías, D. González, F. Chinesta, and E. Cueto, "Structure-preserving neural networks," *Journal of Computational Physics*, vol. 426, p. 109950, Feb 2021.

[4] Q. Hernandez, A. Badías, D. González, F. Chinesta, and E. Cueto, "Deep learning of thermodynamics-aware reduced-order models from data," *Computer Methods in Applied Mechanics and Engineering*, vol. 379, p. 113763, Jun 2021.

[5] D. González, A. García-González, F. Chinesta, and E. Cueto, "A data-driven learning method for constitutive modeling: Application to vascular hyperelastic soft tissues," *Materials*, vol. 13, no. 10, 2020.

[6] D. González, F. Chinesta, and E. Cueto, "Thermodynamically consistent data-driven computational mechanics," *Continuum Mechanics and Thermodynamics*, 2018.

[7] P. Werbos and P. John, "Beyond regression : new tools for prediction and analysis in the behavioral sciences /," 01 1974.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A

large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[9] M. Fernández, F. Fritzen, and O. Weeger, "Material modeling for parametric, anisotropic finite strain hyperelasticity based on machine learning with application in optimization of metamaterials," *International Journal for Numerical Methods in Engineering*, vol. n/a, no. n/a.

[10] F. Chinesta, E. Cueto, and B. Klusemann, "Empowering materials processing and performance from data and ai," *Materials*, vol. 14, no. 16, 2021.

[11] D. Klein, M. Fernández, R. Martin, P. Neff, and O. Weeger, "Polyconvex anisotropic hyperelasticity with neural networks," 06 2021.

[12] T. Gärtner, M. Fernández, and O. Weeger, "Nonlinear multiscale simulation of elastic beam lattices with anisotropic homogenized constitutive models based on artificial neural networks," *Computational Mechanics*, vol. 68, 11 2021.

[13] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, "Relational inductive biases, deep learning, and graph networks," 2018.

[14] M. Pfeiffer, C. Riediger, J. Weitz, and S. Speidel, "Learning soft tissue behavior of organs for surgical navigation with convolutional neural networks," *CoRR*, vol. abs/1904.00722, 2019.

[15] F. Regazzoni, M. Salvador, L. Dedè, and A. Quarteroni, "A machine learning method for real-time numerical simulations of cardiac electromechanics," 2021.

[16] J. Zhang, Y. Zhong, and C. Gu, "Neural network modelling of soft tissue deformation for surgical simulation," *Artificial Intelligence in Medicine*, vol. 97, pp. 61–70, 11 2018.

[17] E. Zhang, M. Yin, and G. E. Karniadakis, "Physics-informed neural networks for nonhomogeneous material identification in elasticity imaging," 2020.

[18] T. Grandits, S. Pezzuto, F. S. Costabal, P. Perdikaris, T. Pock, G. Plank, and R. Krause, "Learning atrial fiber orientations and conductivity tensors from

intracardiac maps using physics-informed neural networks," 2021.

[19] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS 2017 Workshop on Autodiff*, 2017.

[20] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, 2015.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[23] M. Ariza-Gracia, J. Flecha-Lescún, P. Büchler, and B. Calvo, "Corneal Biomechanics After Intrastromal Ring Surgery: Optomechanical In Silico Assessment," *Translational Vision Science Technology*, vol. 9, pp. 26–26, 10 2020.

[24] "Hyperelastic behavior of rubberlike materials." `https://abaqus-docs.mit.edu/2017/English/SIMACAEMATRefMap/simamat-c-hyperelastic.htm`.

[25] T. Gasser, R. Ogden, and G. Holzapfel, "Hyperelastic modelling of arterial layers with distributed collagen fibre orientations," *Journal of the Royal Society - Interface*, vol. 3, no. 6, pp. 15–35, 2006.

[26] "Anisotropic hyperelastic modeling of arterial layers." `https://abaqus-docs.mit.edu/2017/English/SIMACAEBMKRefMap/simabmk-c-anisohyperelastic.htm`.

[27] M. Mehboudi, A. Sanpera, and J. M. R. Parrondo, "Fluctuation-dissipation theorem for non-equilibrium quantum systems," *Quantum*, vol. 2, p. 66, May 2018.

[28] G. Holzapfel, T. Gasser, and R. Ogden, "A new constitutive framework for arterial wall mechanics and a comparative study of material models," *Journal of Elasticity*, vol. 61, pp. 1–48, 04 2012.

[29] M. Liu, L. Liang, and W. Sun, "A generic physics-informed neural network-based constitutive model for soft biological tissues," *Computer Methods in Applied Mechanics and Engineering*, vol. 372, p. 113402, 2020.

[30] "Anisotropic hyperelastic behavior." `https://abaqus-docs.mit.edu/`

2017/English/SIMACAEMATRefMap/simamat-c-anisohyperelastic.htm#simamat-c-anisohyperelastic-holzapfel.

Lista de Figuras

Lista de Tablas

Anexos

Anexos A

Otros modelos de la capa adventicia de la arteria aorta ilíaca.

Como se ha mencionado en la memoria la red también se puede adaptar a otras orientaciones de la muestra, otra alineación de las fibras y a otro nodo de la geometría. En este anexo veremos su implementación y resultados en esos casos.

A.1. Diferentes alineaciones de fibras.

Se puede tener diferente alineación de fibras dependiendo del valor de la variable κ:

- $\kappa = 0$ (Fibras alineadas)
- $\kappa = 0{,}333$ (Material isótropo)

En este apartado se adaptará la red para aprender y reproducir el caso de $\kappa = 0$, para ello se ha hecho otra base de datos con los mismos criterios y simulaciones que hemos explicado en el capítulo 3.2. En este caso la orientación de la muestra es la del modelo básico de 15^{o}.

Los hiperparámetros de la red son los mismo que en el modelo básico exceptuando el número de épocas de entrenamiento que para este caso ha sido de 150 épocas.

Los resultados de la evolución temporal dada por la red en el entrenamiento y en la validación para este caso se puede ver en las figuras A.1 y A.2. Se puede ver que se consigue reproducir la evolución de las variables tanto en el entrenamiento como en la prueba de validación.

En la tabla A.1 y en la figura A.3 se pueden ver el error cuadrático medio para el

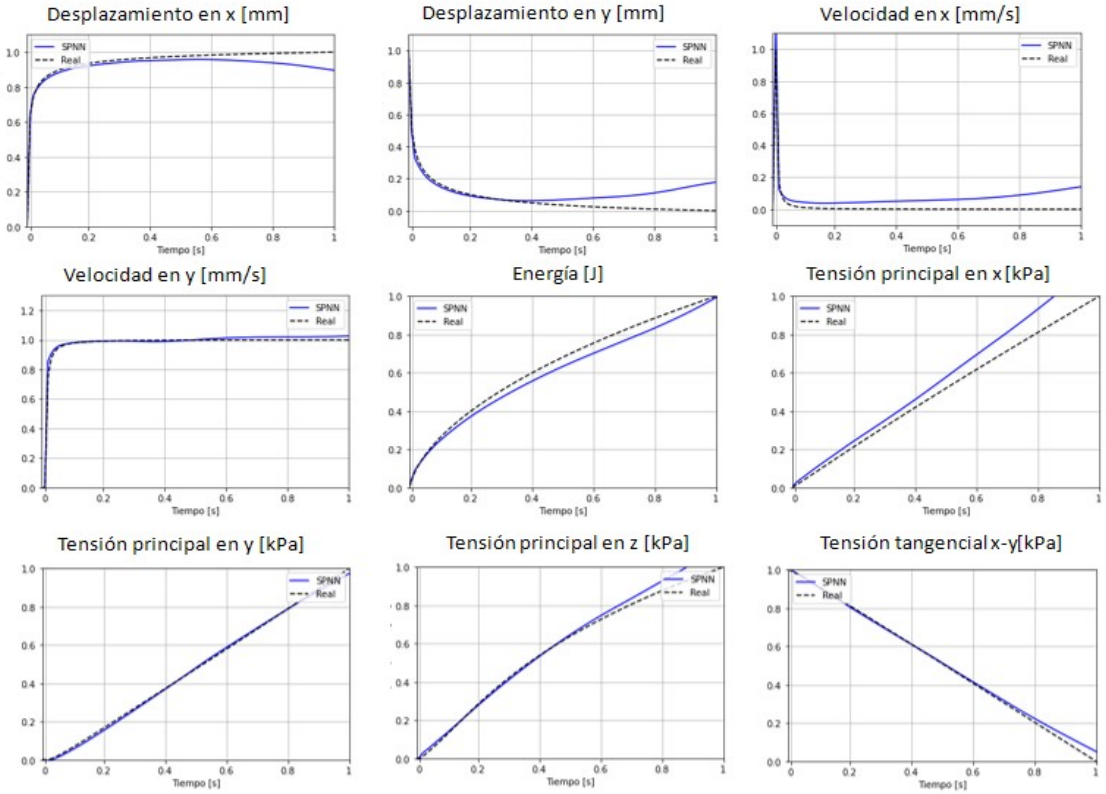

Figura A.1: Gráfica de la evolución temporal de las variables de estado de la red frente a la realidad en el entrenamiento.

entrenamiento y la validación. Se aprecia que estos valores son bajos, del orden de $1{,}00E-03$, y que los máximos se dan para la tensión principal en x para ambos casos.

Variable	MSE Simulación de entrenamiento	MSE Simulación de Validación
u_x	1.70E-03	2.20E-03
u_y	5.60E-03	7.20E-03
u_z	5.00E-04	4.00E-04
v_x	5.00E-03	5.40E-03
v_y	3.00E-04	1.20E-03
v_z	2.20E-03	2.10E-03
e	1.60E-03	1.10E-03
σ_x	8.30E-03	1.03E-02
σ_y	1.00E-04	6.00E-04
σ_z	1.50E-03	6.40E-03
τ_{xy}	3.00E-04	4.00E-04
τ_{yz}	1.00E-04	1.00E-04
τ_{xz}	2.00E-04	2.00E-04

Tabla A.1: Error cuadrático medio para cada variable de estado en la prueba de validación para el modelo $\kappa = 0$

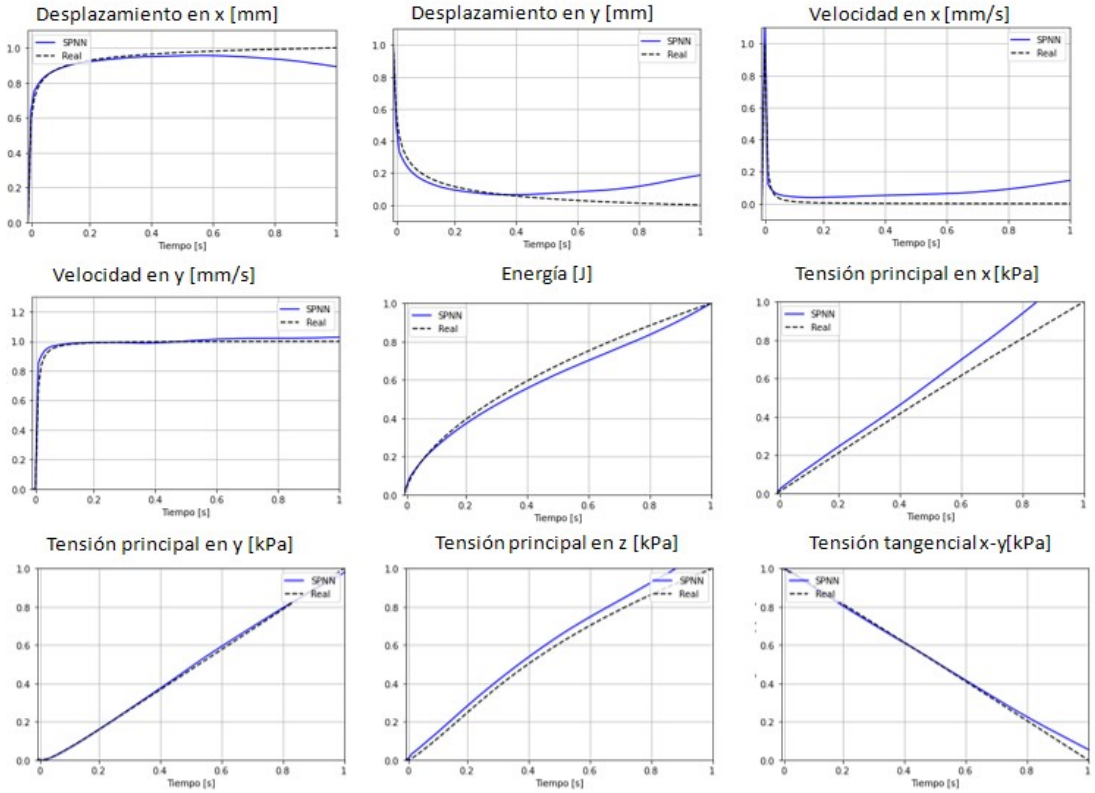

Figura A.2: Gráficas de la evolución temporal de las variables de estado de la red frente a la realidad en la validación.

Se puede comprobar que nuestra red es capaz de adaptarse eficazmente a modelos con diferentes alineaciones de fibras y reproducir la evolución temporal de todas las variables de estado con errores bajos.

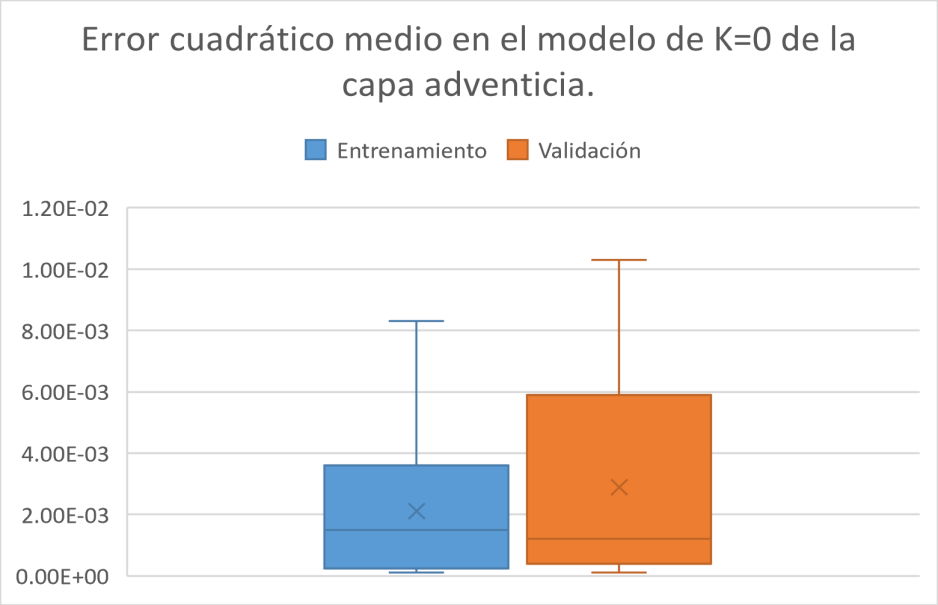

Figura A.3: Diagrama de caja del error medio cuadrático para los resultados del entrenamiento y de la validación del modelo $\kappa = 0$.

A.2. Diferente orientación de la muestra.

También se ha aplicado la red en otra orientación de la muestra (axial). La alineación de fibras será como en el modelo básico $\kappa = 0{,}333$ y los hiperparámetros de la red serán los mismos

En las gráficas A.4 y A.5 se ven los resultados de la evolución temporal de las distintas variables de la red tanto en el entrenamiento como en el test para el caso axial.

La tabla A.2 y el diagrama de caja (Figura A.6) muestran el error cuadrático medio para este caso. Se observa que los valores tanto para el entrenamiento como para el test están alrededor de 2.10E-03, exceptuando el máximo del de la validación que se da en el desplazamiento en z con un valor de 1.08E-02.

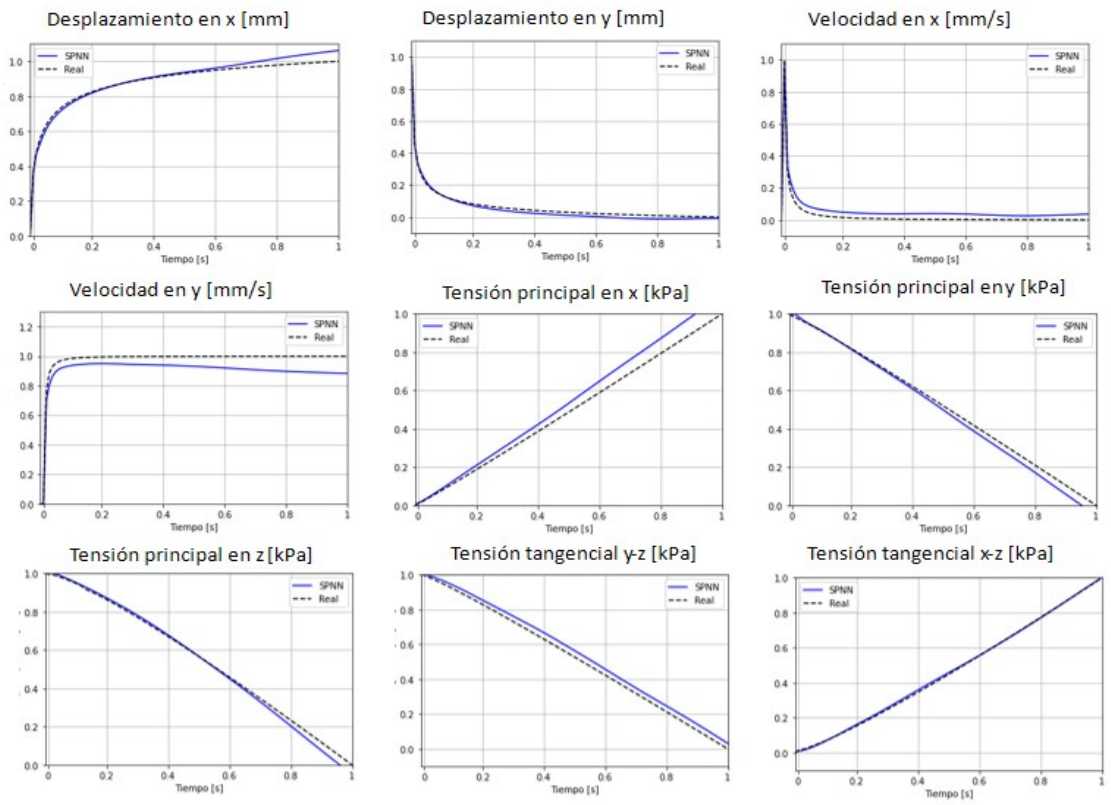

Figura A.4: Gráficas de la evolución temporal de las variables de estado de la red frente a la realidad en el entrenamiento.

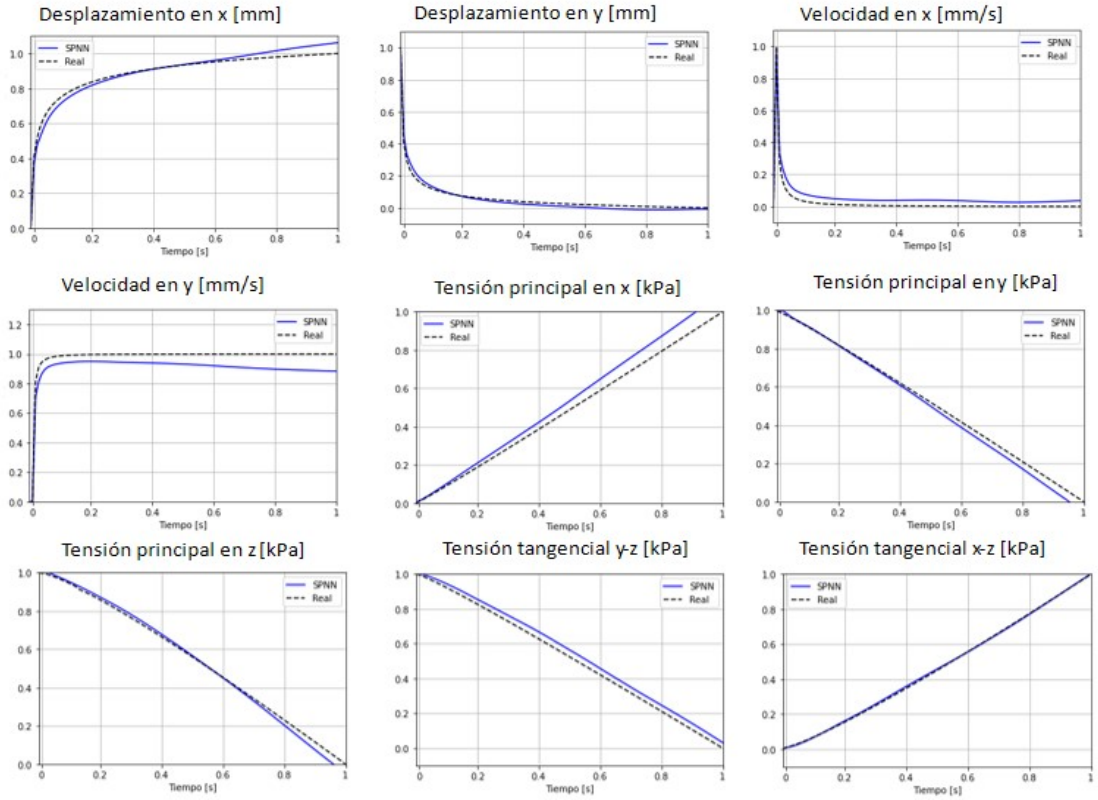

Figura A.5: Gráficas de la evolución temporal de las variables de estado de la red frente a la realidad en la validación.

Variable	MSE Simulación de entrenamiento	MSE Simulación de Validación
u_x	7.00E-04	9.00E-04
u_y	3.00E-04	3.00E-04
u_z	5.00E-03	1.08E-02
v_x	1.10E-03	1.30E-03
v_y	6.10E-03	6.30E-03
v_z	6.60E-03	7.10E-03
e	4.40E-03	4.40E-03
σ_x	3.20E-03	3.20E-03
σ_y	8.00E-04	7.00E-04
σ_z	5.00E-04	5.00E-04
τ_{xy}	2.10E-03	2.00E-03
τ_{yz}	1.00E-03	1.10E-03
τ_{xz}	3.89E-05	2.82E-05

Tabla A.2: Error cuadrático medio para cada variable de estado en la prueba de validación para el modelo axial.

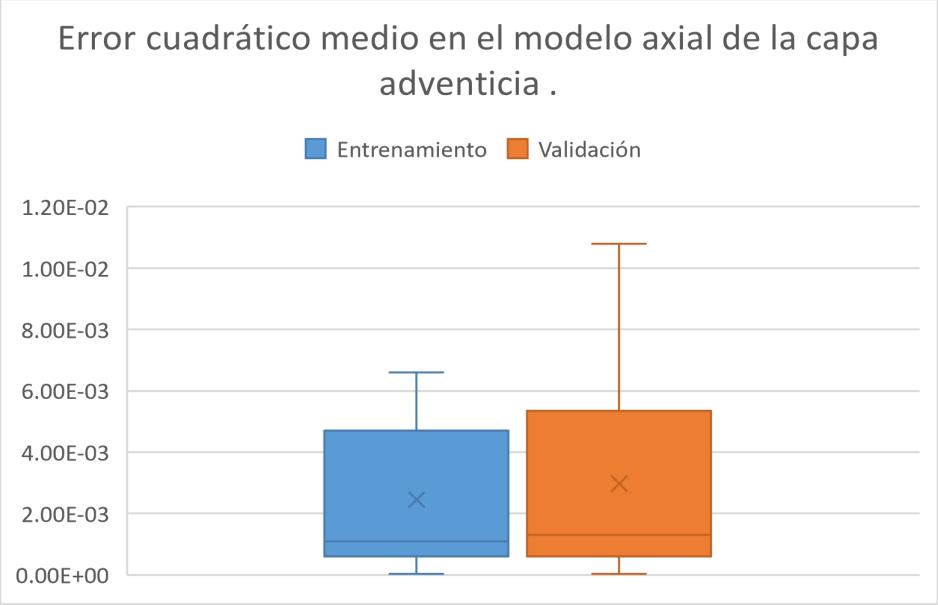

Figura A.6: Diagrama de caja del error medio cuadrático para los resultados del entrenamiento y de la validación del modelo axial.

A.3. Aplicación a otro nodo.

Ya se ha comprobado que en el mismo nodo la red es capaz de aprender diferentes modelos con diferente comportamiento, ahora veremos si nuestra red es capaz de aprender en otro nodo de la malla, en este caso nos iremos a una zona donde haya mayor solicitación de tensiones, por lo tanto el nodo que se ha escogido se puede ver en la figura A.7.

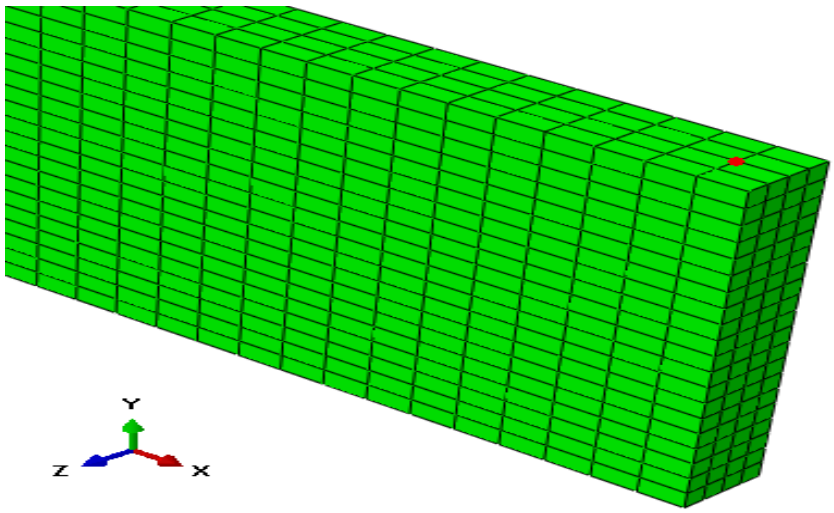

Figura A.7: Nuevo nodo escogido para la aplicación de la red a otro punto de la geometría.

Los resultados de le evolución temporal para este nodo se pueden ver en las figuras A.8 y A.9, se aprecia que para este cambio de nodo la red consigue una reproducción fiel de las diferentes variables.

Por ultimo, vemos que el error cuadrático medio tiene valores similares a los diferentes modelos que hemos ya hemos tratado, teniendo máximos en la tensión principal en z en el entrenamiento (4.70E-03) y en la velocidad en z en la prueba de validación (8.50E-03). (Tabla A.3 y Fig. A.10)

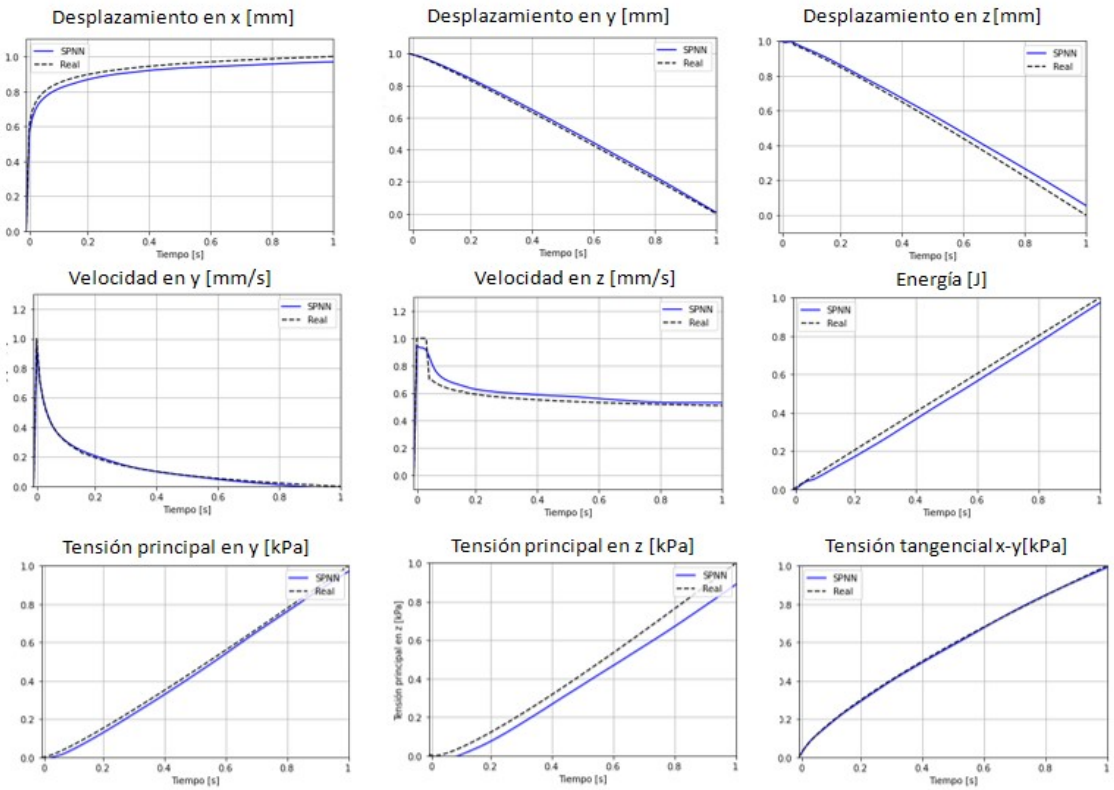

Figura A.8: Gráficas de la evolución temporal de las variables de estado de la red frente a la realidad en el entrenamiento.

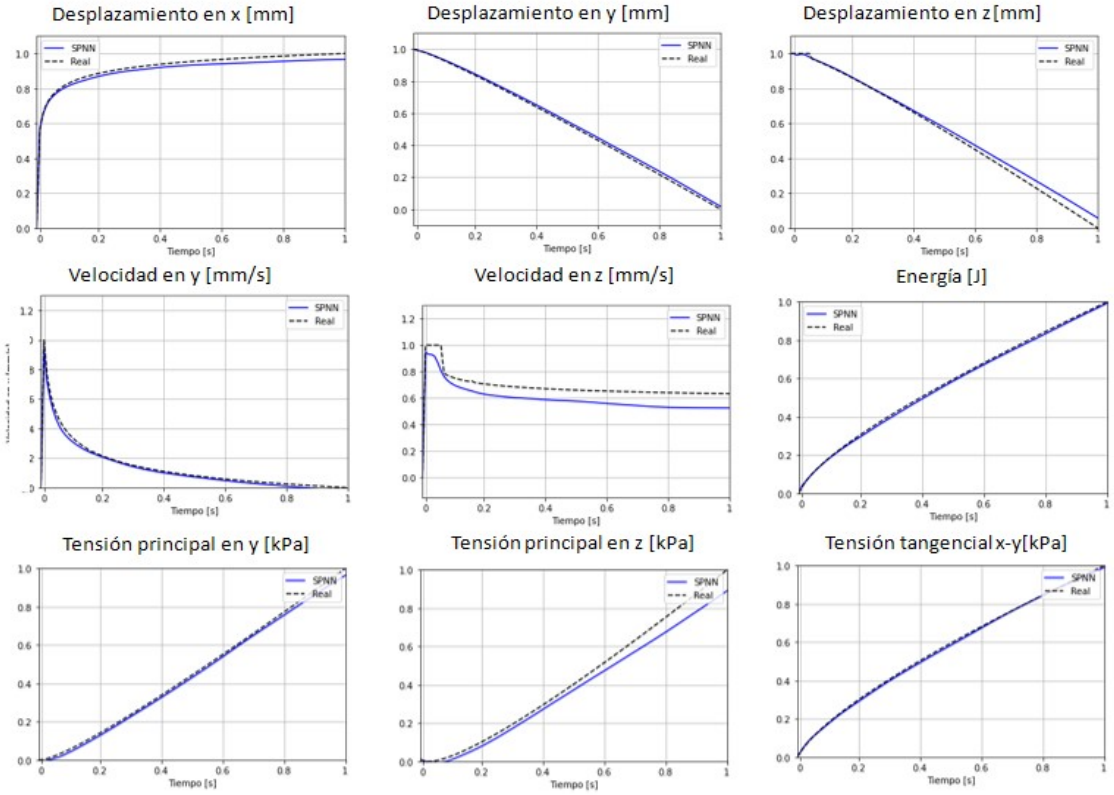

Figura A.9: Gráficas de la evolución temporal de las variables de estado de la red frente a la realidad en la validación.

Variable	MSE Simulación de entrenamiento	MSE Simulación de Validación
u_x	8.00E-04	5.00E-04
u_y	2.00E-04	2.00E-04
u_z	1.00E-03	8.00E-04
v_x	6.00E-04	5.00E-04
v_y	1.00E-04	3.00E-04
v_z	1.60E-03	8.50E-03
e	2.51E-05	9.21E-05
σ_x	1.20E-03	1.30E-03
σ_y	4.00E-04	2.00E-04
σ_z	4.70E-03	2.90E-03
τ_{xy}	2.30E-05	5.15E-05
τ_{yz}	1.60E-03	1.60E-03
τ_{xz}	5.00E-04	5.00E-04

Tabla A.3: Error cuadrático medio para cada variable de estado en la prueba de validación para el modelo aplicado a otro nodo.

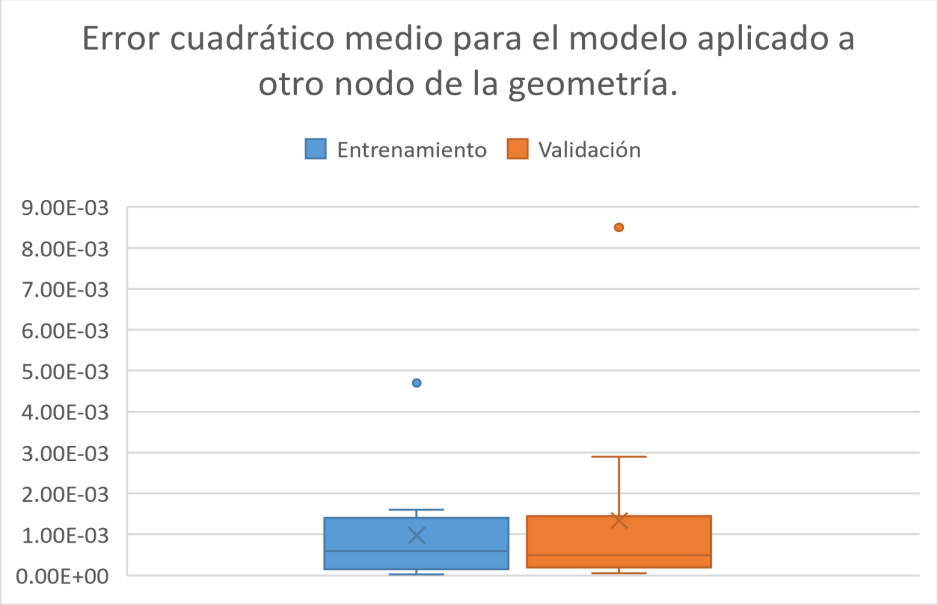

Figura A.10: Diagrama de caja del error medio cuadrático para los resultados del entrenamiento y de la validación del modelo aplicado a otro nodo.

Este modelo sirve para comprobar que la red puede ser implementada en diferentes nodos de la geometría teniendo reproducciones fieles de la evolución temporal de las variables de estado.