# Resultados caso hiperelástico

*30 mayo 2027*

## Introducción

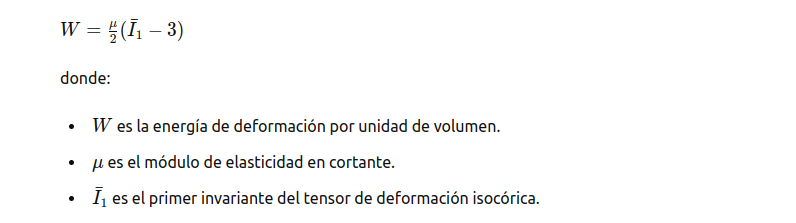
Se ha ejecutado el código diseñado para el caso hiperelástico según el modelo de Neohook, y estos son los resultados. A continuación se expone el proceso, las ecuaciones utilizadas, los resultados obtenidos, las conclusiones y futuros cambios que favorecerían para obtener mejores resultados.

## Teoría

**Material hiperelástico**

Los materiales hiperelásticos son aquellos que exhiben una gran capacidad de deformación reversible bajo cargas externas. Estos materiales pueden soportar grandes deformaciones y recuperar su forma original al eliminar la carga.

El modelo de Neo-Hooke es uno de los modelos constitutivos más simples para describir el comportamiento de los materiales hiperelásticos. Este modelo se basa en la energía de deformación, que se define en términos del tensor de deformación de Green-Lagrange. La función de energía de deformación para un material Neo-Hookeano puede expresarse como:



Este modelo se utiliza frecuentemente debido a su simplicidad y su capacidad para capturar comportamientos esenciales de los materiales hiperelásticos bajo deformaciones moderadas.

**Las PINNs**

Las redes neuronales de ecuaciones diferenciales físicas (Physics-Informed Neural Networks, PINNs) son una clase de redes neuronales diseñadas para resolver problemas de ecuaciones diferenciales parciales (PDEs). Las PINNs incorporan conocimientos físicos directamente en la estructura de la red mediante la inclusión de términos de pérdida que penalizan las soluciones que no satisfacen las leyes físicas gobernantes.

En el contexto de la inferencia de deformaciones en materiales hiperelásticos, las PINNs pueden usarse para predecir el comportamiento de deformación basado en el modelo de Neo-Hooke. La red neuronal se entrena no solo para ajustar los datos experimentales, sino también para satisfacer las ecuaciones de equilibrio y las condiciones de contorno del problema.

1. **Definición del Problema**: Se establece el problema de deformación en términos de las ecuaciones diferenciales que describen el equilibrio de fuerzas en el material, así como las condiciones de contorno y las propiedades del material (en este caso, el modelo de Neo-Hooke).

2. **Estructura de la Red Neurona**: Se diseña una red neuronal con una arquitectura adecuada para aproximar las funciones de desplazamiento o deformación en base a las posiciones iniciales y unos parámetros materiales. Como regla general, en este tipo de redes, no hace falta que lleguemos a los millones de parámetros, y será más fácil entrenar redes mas profundas que anchas.

3. **Función de Pérdida**\*\* La función de pérdida de la PINN incluye varios términos:

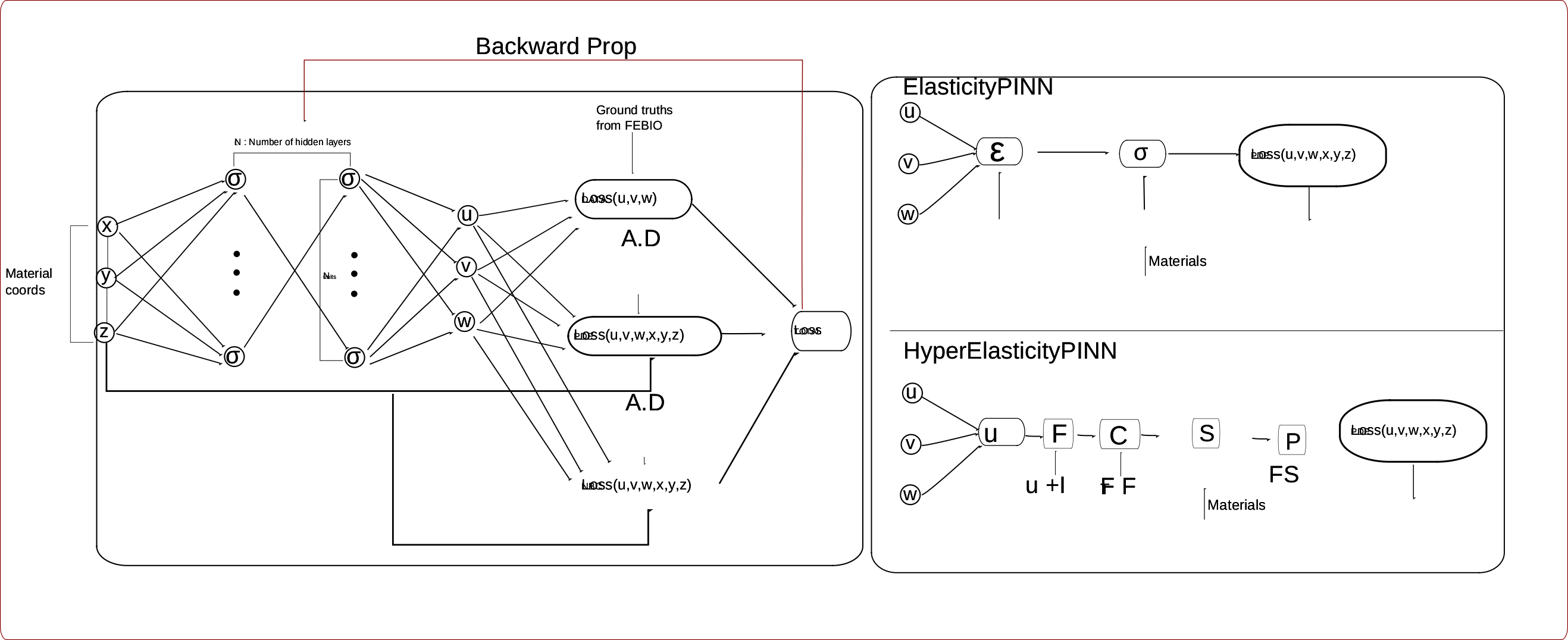
- Pérdida de Datos: Penaliza la diferencia entre las predicciones de la red y los datos experimentales.

- Pérdida Física: Penaliza las violaciones de las ecuaciones de equilibrio y las condiciones de contorno. Esto se evalua en todo el dominio en que queremos que la física se válida, que no tiene que coincidir con todo el dominio donde tenemos datos experimentales.

- Pérdida condiciones de contorno: Impone las condiciones de contorno que hacen única la solución de la física.

- Pérdida Regularizadora: Ayuda a prevenir el sobreajuste de la red según se requiera. En nuestro caso esto no se hizo, sin embargo Christian si aplicó una perdida del estilo si los valores de los parámetros a inferir tomaban valores extremos.

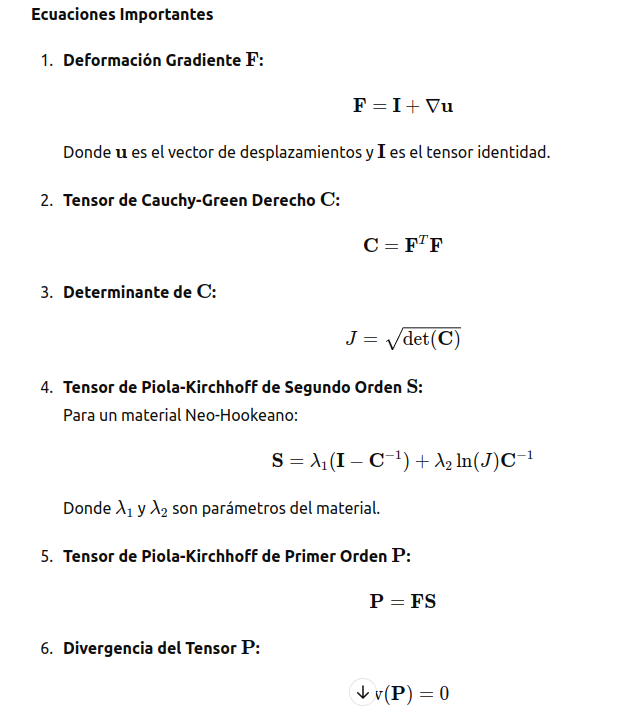
4. **Entrenamiento**: La red se entrena utilizando técnicas de optimización (por ejemplo, el descenso de gradiente) para minimizar la función de pérdida.



### Ecuaciones

En el caso elástico lineal, tenemos una dependencia lineal entre las tensiones y las deformaciones, y los desplazamientos eran fáciles de obtener partiendo de las deformaciones.

Sin embargo, en este caso no lineal, se complica un poco más el problema:



Y para hacer el cálculo de esto mediante la red neuronal:

1. **Inicialización del Modelo**:

- Definir la arquitectura de la red neuronal.

- Inicializar los parámetros del material lambda\_1 y lambda\_2 .

2. **Cálculo de la Pérdida de Datos:**

- Comparar los desplazamientos predichos **u** con los desplazamientos reales **u\_real** usando la función de pérdida de mínimos cuadrados.

3. **Cálculo de la Pérdida Física:**

- Calcular los desplazamientos (u,v,w) en las posiciones dadas ( X, Y, Z ).

- Calcular el gradiente de desplazamientos ∇**u**.

- Construir el tensor de deformación gradiente **F** y el tensor de Cauchy-Green **C**.

- Calcular el tensor de Piola-Kirchhoff de segundo orden **S** y de primer orden **P**.

- Calcular la divergencia del tensor **P** y el residual de equilibrio div(**P**).

- Comparar el residual de equilibrio con cero usando la función de pérdida de mínimos cuadrados.

4. **Cálculo de la Pérdida de Simetría:** Este es un paso extra opcional que se añade a modo de regularización, para imponer más condiciones sobre la red

- Calcular la pérdida de simetría comparando



5. **Función de Pérdida Total**:

- La pérdida total es la suma de la pérdida de datos, la pérdida de equilibrio y la pérdida de simetría.

Este flujo de trabajo y las ecuaciones proporcionan una visión general del enfoque PINN para modelar materiales hiperelásticos según el modelo Neo-Hookeano, combinando datos experimentales y principios físicos en el entrenamiento de la red neuronal, además de las condiciones de contorno.

**Nota:** El modelo hiperelástico planteado está imponiendo unas condiciones de contorno de Dirichlet, estas condiciones están directamente contenidas en la loss de la DATA. Son condiciones sobre la posiciones de los nodos fijos. O limitaciones unidireccionales sobre ciertos nodos.

## Código

De manera muy sintetizada, este es el código:



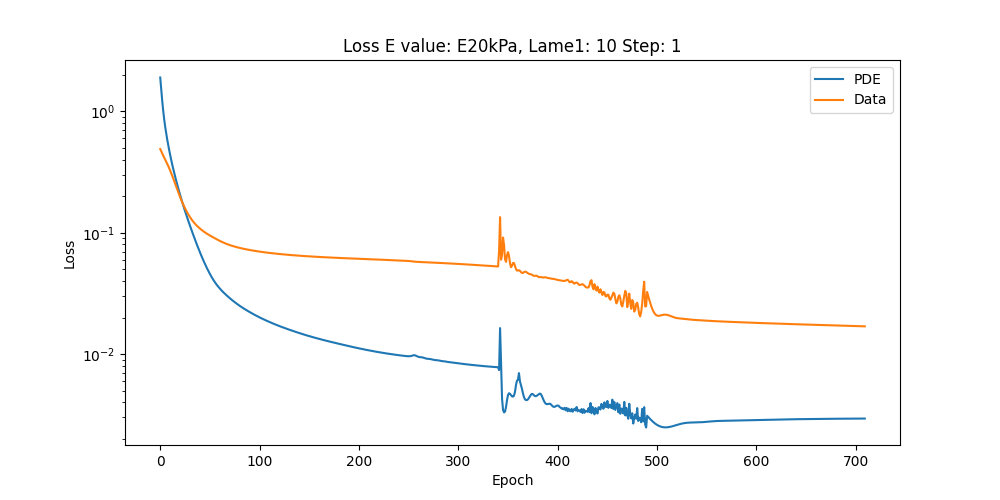
En el código podemos observar lo siguiente:

* La declaración de los parámetros de lame como parámetros entrenables de la red.
* La definición de las funciones que serán la funcion loss usada por la red para aprender de los datos y de la física.
  + La loss de la data no tiene mucha complicación será la función elegida para comparar entre los datos predichos y los reales.
  + La loss de la PDE es un poco mas compleja y responde al cálculo de la ecuación diferencial que rige el comportamiendo de los materiales según el modelo de Neohook. En este flujo de cálculos, partimos de las posiciones X,Y,Z y se calculan los desplazamientos por medio del modelo PINN. Una vez obtenidas U,V,W , podemos obtener F, con ello, C y por ultimo S. Mediante F y S obtenemos P, y es la divergencia de P la que igualamos a 0. Además podemos calcular una simetria que debe cumplirse entre P y F, e imponerla como una loss extra que añade más valor a la física.

**El código completo con más detalle se puede encontrar en el repositorio:**[**https://github.com/ArturoSirvent/DeepElasticity**](https://github.com/ArturoSirvent/DeepElasticity)

## Resultados

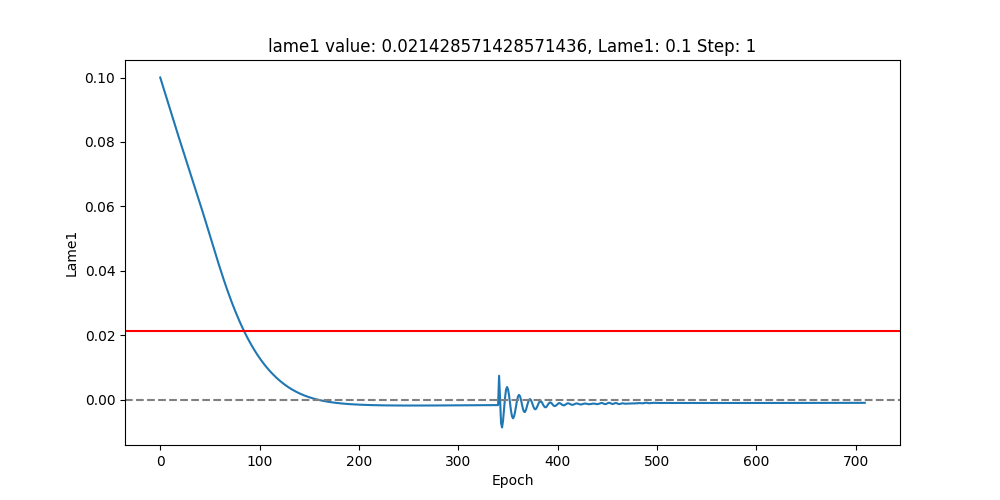
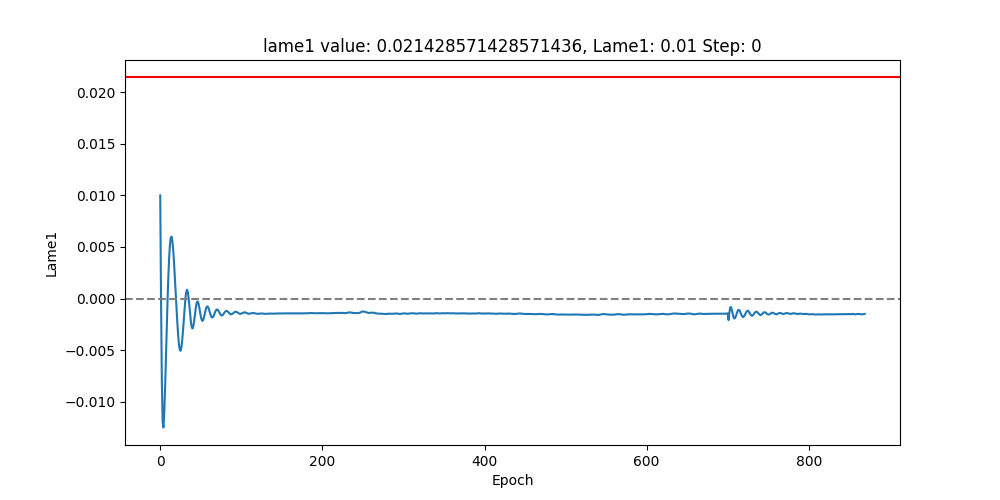
Los resultados obtenidos tiene buena forma en lo respectivo al proceso de entrenamiento. Las curvas obtenidas presentan una buena progresión descendente tanto en el término de data, como para el término de la física (PDE).



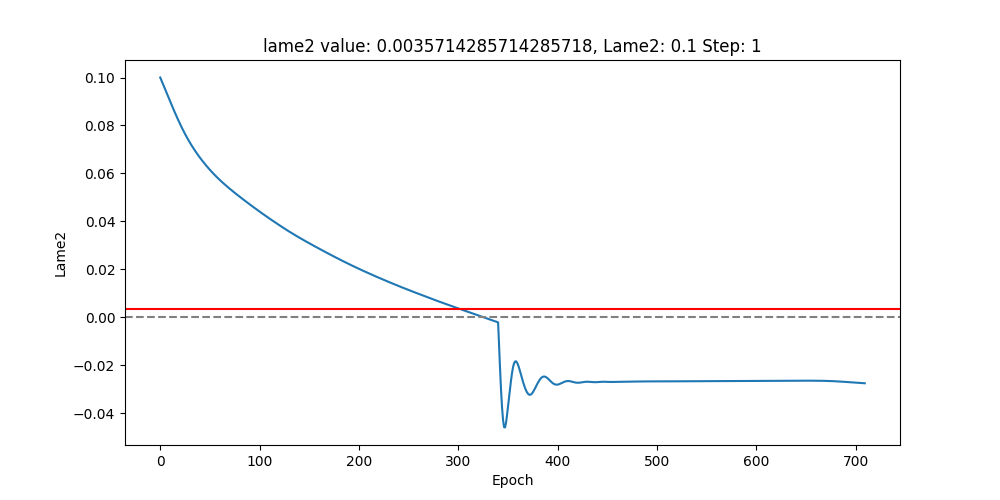
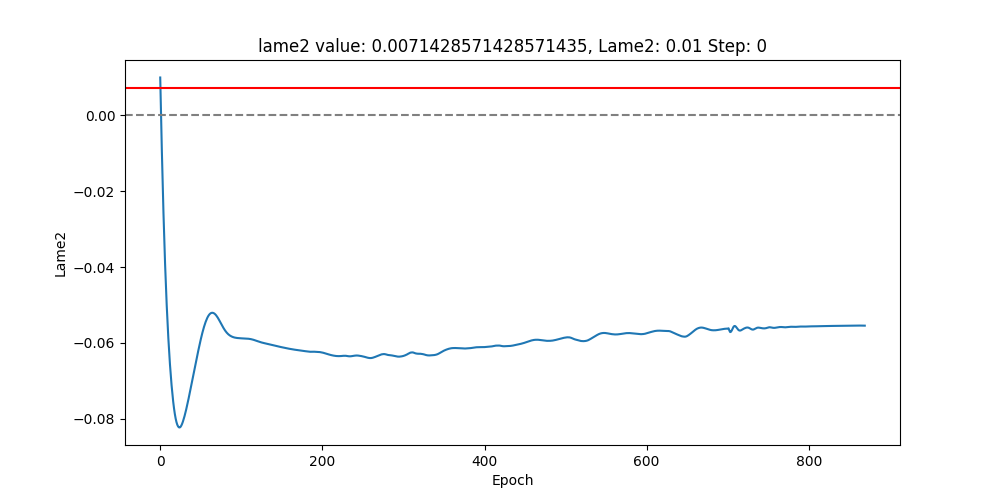
En estos casos, las condiciones de contorno que estamos considerando, han sido únicamente las de Dirichlet y no las de Neumann.

Sin embargo la convergencia en estos casos a resultado ser a valores erróneos de ambos parámetros. El problema inverso se ha planteada cada vez para uno de los parámetros, y algunos de los resultados son los siguientes:

Lame 1:



Lame 2:



**Nota**: En todos los casos las convergencias fueron a valores negativos y bastante estables, por ejemplo el segundo parámetro de lame, como se observa en los gráficos anteriores mostró convergencias a valores como -0.58 y - 0.36

**Nota 2**: El “step” mencionado en el título de las imágenes no tiene relación con el step de los métodos finitos, es un “flujo de entrenamiento” definido para probar diferentes estrategias de optimización.

En el repositorio se pueden encontrar todas las gráficas del bucle de entrenamiento (https://github.com/ArturoSirvent/DeepElasticity/tree/main/notebooks/002-HyperElasticity/results), además de los objetos “pickle” con los datos concretos del entrenamiento, y el propio script con el que se hizo el bucle, por si se quisieran probar más combinaciones. Y el modelo usado su puede encontrar en src/models (https://github.com/ArturoSirvent/DeepElasticity/blob/main/src/models.py) en la clase “PINN\_NeoHook”.

Una de los posibles soluciones que se plantean en la sección de conclusiones, sobre la convergencia errónea del problema inverso, es la poca importancia que pueda tener el término de la PDE frente a de la data o la regularización de la simetría mencionada en la teoría.

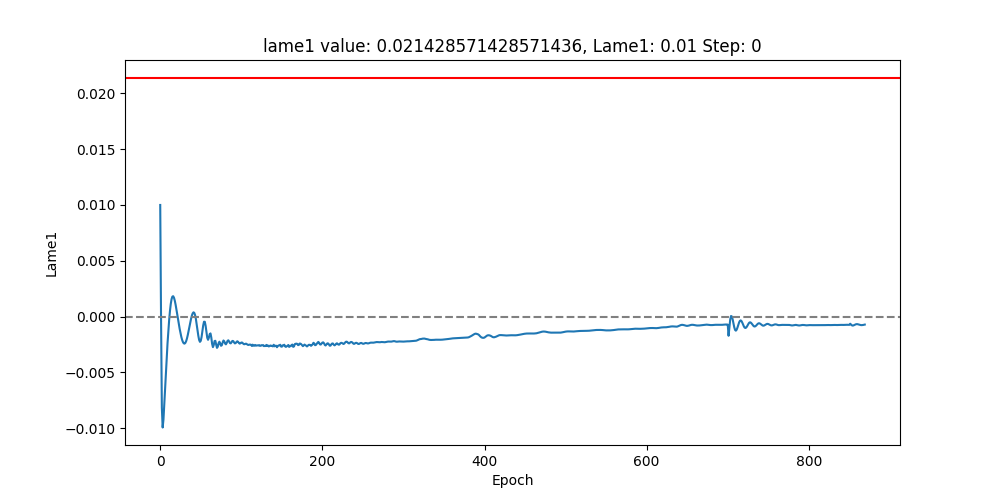
Loss=loss\_PDE + loss\_Data + loss\_symetry

Tras una prueba y error, se han determinado unos pesos estáticos que asociaremos a cada uno de los términos para intentar igualar sus importancias. Los pesos son los siguientes:

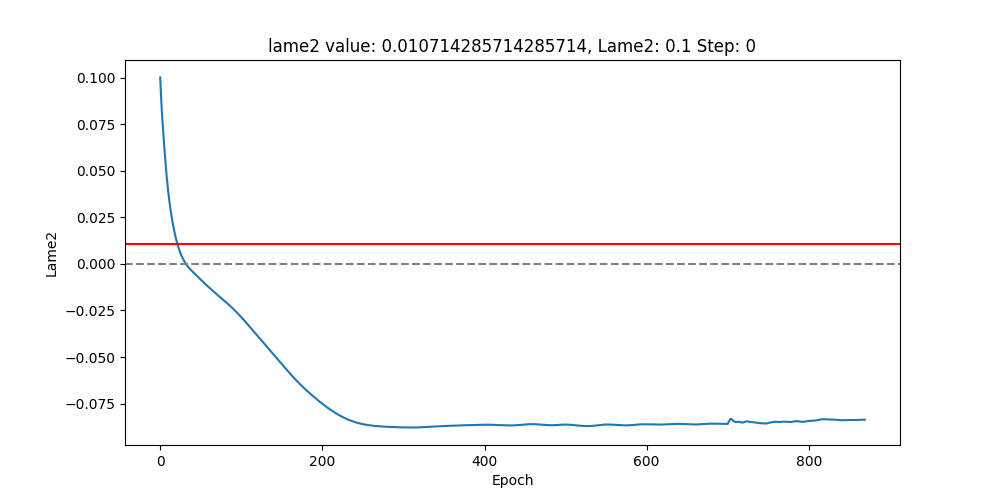
Loss= 1e5 \* loss\_PDE + loss\_Data + 1e13 \* loss\_symetry

Con este cambio, los nuevos resultados son los siguientes:

Lame 1:



Lame 2:



Nota: Se realizó el mismo bucle para la prueba de hiperparámetros que en el caso anterior, y todas las gráficas y datos obtenidos (en formato pickle) están disponibles en el repositorio en la carpeta results\_2 (https://github.com/ArturoSirvent/DeepElasticity/tree/main/notebooks/002-HyperElasticity/results\_2).

## Conclusiones

Los resultados obtenidos muestran una prometedora forma respecto al progreso del entrenamiento, y la convergencia de valores. Sin embargo, la convergencia de los parámetros estimados se produce a valores erróneos no justificables dentro de un intervalo de error.

El entrenamiento parece haberse realizado de manera correcta, y obteniendo curvas de entrenamiento correctas (al contrario de algunas pruebas realizadas con el caso lineal, donde la priorización de una loss daba lugar a descuidar las otras). A continuación expongo los puntos de mejora que se podrían abordar para investigar y solventar los problemas sobre la convergencia, sin embargo, cada unos de estos puntos corresponde a un estudio más en profundidad del comportamiento del modelo.

**Opción 1:** Balanceo de las funciones de pérdida.

Un posible motivo de que no se esté realizando correctamente el aprendizaje en el problema inverso, podría deberse a que la física no está siendo aprendida correctamente, y eso podría deberse a su vez, a un desbalanceo entre las funciones de pérdida. Tal y como se menciona en la sección de resultados, hasta el momento no se ha observado que ese juego dé lugar a mejores resultados, pero quizá un balanceo dinámico, o estrategias más avanzadas, puedan ayudar al problema, aunque muy dificilmente resolverlo.

**Opción 2:** Revisión de la teoría física subyacente.

Un motivo de posible fallo, es la manera en que la PINN está implementando la teoria del material de Neohook y la manera en que fue simulada por FEM. En este caso, la teoría implementada (y explicada en la sección de Teoría) corresponde a la que se implementa directamente en FEBIO. Sin embargo, cabe la posibilidad de las simulaciones de ANSYS no correspondan con la misma manera de calcularse o presente diferencias, lo cual daría lugar a calcular unos parámetros que se relacionan con las deformaciones diferente a como la simulación de estas deformaciones se produjo.

**Opcion 3:** Inicialización de la red, y los parámetros materiales.

En algunos casos se ha observado como el valor inicial de los parámetros materiales es relevante para la estabilidad y convergencia de la red. En algunos casos tenemos que un valor inicial muy alejado del real, le hace tomar una inercia progresiva y correcta hacia el valor real, y para cuando la red ya está más entrenada, se da la convergencia de manera suave. Por otro lado, se ha observado como comenzar con valores iniciales muy próximo al real, o el propio real, puede dar lugar a inestabilidades y divergencias de este.   
Al respecto se proponen ciertas estrategias que podrían ayudar. La primera es la de adecuar primero la red a la ley física (con los parámetros congelado sin variar a pesar de ser erróneos), para más tarde descongelar los parámetros y permitirles converger. Otra posible estrategia, es aplicar una regularización sobre un rango de posibles valores de dicho parámetro (por ejemplo, para evitar valores negativos). Esto se lograría mediante una función a modo “pozo de potencial”.