UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Desarrollo de una herramienta de simulación para modelar el efecto piezoeléctrico utilizando *PINNs*

Trabajo de graduación presentado por José Daniel González Carrillo para optar al grado académico de Licenciado en Ingeniería en Ciencias de la Computación y Tecnologías de la Información

Guatemala,

2024

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Desarrollo de una herramienta de simulación para modelar el efecto piezoeléctrico utilizando *PINNs*

Trabajo de graduación presentado por José Daniel González Carrillo para optar al grado académico de Licenciado en Ingeniería en Ciencias de la Computación y Tecnologías de la Información

Guatemala,

2024

Vo.Bo.:

(f) LUKE DANIEL SHAW Firmado digitalmente por LUKE DANIEL|SHAW Fecha: 2025.03.17 21:31:57 +01'00'

PhD. Luke Daniel Shaw

Tribunal Examinador:

(f)

PhD. Guadalupe Barrios

(f)

PhD. Gabriel Barrientos

(f)

MSc. Luis Furlán

Fecha de aprobación: Guatemala, 4 de diciembre de 2024.

Dedicatoria

Dedico este proyecto a Dios, porque sin su bendición no hubiera podido realizar absolutamente nada de lo que he plasmado aquí. Porque siempre me dio su mano en los momentos más difíciles y me iluminó cuando más lo necesité.

A mi mamá, porque es la que siempre ha estado a mi lado, mi apoyo más grande. Ella siempre se ha preocupado porque yo salga adelante, me ha inspirado a superarme y ha sido un gran ejemplo a seguir. Es la que me ha dado su hombro en los momentos más difíciles y su sonrisa más brillante en las alegrías más grandes. Soportó mi mal humor y mis frustraciones, y siempre estuvo ahí para darme un abrazo. Es incondicional para mí y su sabiduría me ayuda a dilucidar el camino siempre que lo necesito. Mi gran amiga.

A mi papá, porque ve un potencial en mí que no logro ver. Porque siempre me empuja a dar lo mejor de mí. Porque ha sudado para siempre darme las mejores oportunidades y porque ha sido un ejemplo de trabajo duro. Porque siempre ha estado ahí para darme un consejo cuando lo necesito y más que un padre, ha sido un amigo. Porque se ha emocionado tanto o más que yo por mis logros y siempre está orgulloso de mí. Por estar presente.

A mi novia Natalia, porque en este último año ha sido fuente de mis alegrías más grandes. Porque definitivamente, ha hecho este camino más suave. Por estar ahí para apoyarme en mis días, dándome todo su amor y por recordarme de lo que soy capaz. Porque ha volado a mi lado y ha quedado marcada en mi corazón.

A mi perro Charlie, porque la suavidad de su amor ha aliviado la carga y el estrés de los días. Porque me ha enseñado a tomarme la vida con más alegría y a disfrutar con intensidad todos los momentos.

Agradecimientos

En primer lugar, agradezco a mi supervisor, Ph.D. Gabriel Barrientos. Él fue el que me propuso el desarrollo de este proyecto tan retador. Siempre me instó a empujar mis límites y me acompañó en todo el proceso. Fue un verdadero guía, brindándome mucha información e ideas valiosas, sin las cuales definitivamente no hubiera podido completar este trabajo. Siempre estuvo para responder mis preguntas, hasta las más obvias y me brindó su tiempo cuando yo lo necesité. Su apoyo y seguimiento del proyecto fueron cruciales.

También agradezco a mi asesor, M.Sc. Luke Shaw, quien me ayudó a desentrañar los hilos más complejos de este trabajo, especialmente en la parte matemática y de programación. Con mucha paciencia siempre leyó mi trabajo y me hizo pulirlo, así yo ya no quisiera. Él tomó la responsabilidad de asesorarme, a pesar de que tenía que entregar su tesis de Ph.D. Fue una gran ayuda para completar el proyecto.

Finalmente, agradezco al Centro de Estudios en Informática Aplicada (CEIA) de la Universidad del Valle de Guatemala, por haberme permitido realizar mis prácticas. Esto me facilitó mucho el desarrollo de este trabajo, permitiéndome dedicarle todo mi tiempo.

Índice

Lista de figuras

Lista de tablas

Resumen

El presente proyecto aprovecha el reciente desarrollo de un paradigma de redes neuronales orientado a la simulación de sistemas descritos por ecuaciones diferenciales, conocido como Redes Neuronales Informadas por la Física (PINNs por sus siglas en inglés). Las PINNs han sido utilizadas para solucionar distintos problemas que involucran sistemas físicos, tales como mecánica de fluidos o transmisión de calor. Este trabajo extiende su aplicación a la modelación del fenómeno piezoeléctrico.

Los materiales piezoeléctricos presentan una relación electro-mecánica, lo cuál les permite generar electricidad al deformarse, conocido como el efecto piezoeléctrico directo, o deformarse cuando se les aplica electricidad, siendo este el efecto piezoeléctrico indirecto. Durante esta investigación se identificó el conjunto de ecuaciones diferenciales que permiten describir el comportamiento piezoeléctrico. A partir de esto, la metodología propuesta incorpora este conjunto de ecuaciones en una red neuronal, junto con las condiciones específicas necesarias para modelar un dispositivo conocido como viga en voladizo piezoeléctrica, generando una PINN.

Inicialmente la PINN fue utilizada para modelar el efecto piezoeléctrico indirecto para la viga en voladizo. Para evaluar la efectividad del modelo, se compararon los resultados obtenidos con simulaciones obtenidas del Método de los Elementos Finitos (FEM por sus siglas en inglés). A partir de esto se obtuvieron bajos errores, validando la capacidad de la PINN para modelar este fenómeno. Posteriormente se realizó una variación de la arquitectura original de la PINN, que resultó en una optimización en tiempo y precisión de la solución al problema. Finalmente, esta arquitectura optimizada fue empleada para modelar el fenómeno piezoeléctrico directo, igualmente para una viga en voladizo.

Este proyecto representa, según el conocimiento del autor, una de las primeras implementaciones exitosas de una PINN para la modelación del efecto piezoeléctrico, tanto directo como indirecto. Los resultados obtenidos demuestran la efectividad del acercamiento empleado. Además, la metolodogía utilizada sienta una base que puede permitir la modelación de dispositivos piezoeléctricos más complejos en el futuro.

Abstract

This thesis leverages the recent development of a neural network paradigm known as Physics-Informed Neural Networks (PINNs), used to simulate systems described by differential equations. PINNs have been demonstrated to be effective at solving various problems involving physical systems, such as fluid mechanics and heat transfer. This work extends their application to the modeling of the piezoelectric effect.

Piezoelectric materials exhibit an electro-mechanical relationship, which allows them to generate electricity when deformed, known as the direct piezoelectric effect, or to deform when electricity is applied, known as the indirect piezoelectric effect. During this research, the set of differential equations that describe piezoelectric behavior was identified. Based on this, the proposed methodology incorporates these equations into a neural network, along with the specific conditions necessary to model a device known as a piezoelectric cantilever beam, generating a PINN.

Initially, the PINN was used to model the indirect piezoelectric effect for the cantilever beam. To evaluate the model's effectiveness, the results were compared with simulations obtained from the Finite Element Method (FEM). This comparison yielded low errors, validating the PINN's capability to model this phenomenon. Subsequently, a variation of the original PINN architecture was implemented, resulting in optimization of both computational time and solution accuracy. Finally, this optimized architecture was employed to model the direct piezoelectric effect, also for a cantilever beam.

This project represents, to the author's knowledge, one of the first successful implementations of a PINN for modeling the direct and indirect piezoelectric effect. The results obtained demonstrate the effectiveness of the employed approach. Furthermore, the methodology establishes a foundation that may enable the modeling of more complex piezoelectric devices in the future.

CAPÍTULO 1

Introducción

Las ecuaciones diferenciales parciales (PDEs por sus siglas en inglés) son una herramienta matemática que permite modelar funciones que dependen de distintas variables independientes al mismo tiempo. Su flexibilidad y expresividad hace que sean ampliamente utilizadas en distintas áreas de ciencia y tecnología [1]. Su principal aplicación es en física, donde modela fenómenos en óptica, aerodinámica, física cuántica, entre otros. El proceso para encontrar las soluciones de estas ecuaciones muchas veces no es trivial, e incluso a veces imposible de forma analítica. Por lo tanto, en muchos escenarios se requiere de métodos computacionales que ayuden a encontrar aproximaciones de la solución de las ecuaciones diferenciales que modelan problemas físicos [2].

Entre los distintos algoritmos creados para la modelación y resolución de PDEs de problemas físicos, el Método de los Elementos Finitos (FEM por sus siglas en inglés) se perfila como uno de los más populares. Este se ha destacado por su eficiencia, precisión y madurez para resolver problemas científicos e industriales complejos. Además, a lo largo de su historia, se ha presentado como un método de vanguardia tecnológica [3]. Sin embargo, a pesar de su eficacia, también presenta limitaciones en cuanto a flexibilidad y generalización de las soluciones, especialmente en escenarios de alta complejidad [4]. Entonces, para fomentar nuevos avances y descubrimientos científicos, el desarrollo de nuevas herramientas que usen tecnologías innovadoras es imprescindible.

Con los recientes avances en Aprendizaje Automático, particularmente en Aprendizaje Profundo o *Deep Learning*, se ha abierto la oportunidad de solucionar problemas anteriormente imposibles [5]. Por esta razón se ha adoptado en distintas áreas de la ciencia y la industria, entre las que destacan visión por computadora, medicina, modelos de lenguaje natural e Inteligencia Artificial Generativa. La intersección entre esta tecnología innovadora y la computación científica se puede encontrar en las Redes Neuronales Informadas por la Física (PINNs por sus siglas en inglés), un área de rápido crecimiento. Este término acuñado tan solo en el 2019 plantea un nuevo paradigma para abordar fenómenos científicos, empleando redes neuronales que integran las ecuaciones diferenciales que los describen, la capacidad de usar observaciones experimentales y el poder computacional del *Deep Lear-*

ning [6]. Esta técnica tiene un potencial para revolucionar la modelación y simulación de problemas científicos, así como nuevos descubrimientos y aplicaciones.

Una de estas aplicaciones es el uso de dispositivos piezoeléctricos. La piezoelectricidad es una propiedad de ciertos cristales, que les permite generar una carga eléctrica cuando se les aplica una fuerza mecánica (efecto piezoeléctrico directo) y viceversa (efecto piezoeléctrico indirecto) [7]. Esta capacidad ha causado que puedan ser usados en distintas áreas. Por ejemplo, forman parte de equipo médico para extracción de imágenes [8]. También pueden ser usados en la recolección de energía eléctrica a partir de energía mecánica, área en la que se han investigado innovaciones como pavimentos que generen electricidad a partir de la fuerza de los vehículos [9]. En general, tienen un gran potencial para ser uno de los materiales del futuro, sin embargo, su diseño puede ser complejo y requerir múltiples iteraciones [10]. Por lo que una herramienta de simulación eficiente es necesaria para poder modelar nuevos dispositivos piezoeléctricos que permitan innovar de distintas maneras. De esta forma, este proyecto explorará el uso de *Deep Learning*, para crear una herramienta que permita la simulación del fenómeno piezoeléctrico. Se hará uso de PINNs, que integren las leyes físicas de la piezoelectricidad, en forma de ecuaciones diferenciales parciales. A partir de esto, se espera obtener un modelo que genere soluciones adecuadas para el fenómeno piezoeléctrico.

CAPÍTULO 2

Justificación

Las PDEs son de vital importancia en el estudio de fenómenos que involucran múltiples variables, como tiempo y espacio, permitiendo modelar comportamientos muy complejos. Las PDEs proporcionan un mecanismo matemático que permite describir estos fenómenos con bastante precisión. Es posible comprender el comportamiento de los problemas estudiados, al resolver las PDEs, pudiendo predecir y optimizar sistemas que dependan de ellas [1]. Esto hace que sean ampliamente usadas en áreas de la física, biología, economía, entre otras, convirtiéndose en herramientas que potencian avances científicos y tecnológicos [11].

Por otra parte, los materiales piezoeléctricos son cruciales en una amplia gama de aplicaciones tecnológicas. En primer lugar, los actuadores hacen uso del efecto piezoeléctrico indirecto. Estos tienen una gran precisión, por lo que son usados en sistemas robóticos, medicina, biotecnología, industria aeroespacial, entre otros [12]. En segundo lugar, los sensores piezoeléctricos son utilizados en evaluación de turbinas y motores de jets, así como biosensores para diagnóstico de problemas médicos, entre otras aplicaciones. [13], [14]. Finalmente, se puede mencionar el uso de piezoelectricidad en la conversión de energía del cinética a energía eléctrica. Esta capacidad les da el potencial para hacer dispositivos electrónicos más baratos y eficientes, así como incrementar el uso de energía renovable [15]. Es evidente entonces que la piezoelectricidad tiene un papel fundamental en diversas áreas tecnológicas. Sus características la posicionan como una tecnología clave en el desarrollo de soluciones sostenibles para el futuro.

El potencial de las PINNs para resolver PDEs que tienen una alta complejidad, además de su capacidad de integrar datos experimentales al proceso de simulación, hacen que sea un paradigma prometedor para el desarrollo de la computación científica. A pesar de no ser el único acercamiento que hace uso de redes neuronales para resolver problemas físicos, tiene ciertas ventajas. Una de ellas, es que la integración de las PDEs en la simulación asegura resultados que son consistentes con las leyes físicas fundamentales, mejorando así la validez y la fiabilidad de las simulaciones. Otra ventaja, es la relativa facilidad del uso de las PINNs, debido a que se pueden basar en arquitecturas de redes neuronales relativamente simples. Además, integrar y expresar las PDEs en las PINNs es relativamente sencillo, haciendo uso

de la autodiferenciación. [4], [16]

A partir de esto, surge la necesidad de realizar un proyecto que aproveche el impulso inicial de métodos innovadores, para poder crear una herramienta que permitan superar los límites y mejorar los métodos de computación científica para resolución de PDEs [3]. Además, el uso de PINNs en este contexto, puede ser un paso inicial hacia una herramienta más económica y eficiente para investigar dispositivos piezoeléctricos, proveyendo una mayor flexibilidad en el desarrollo de estos, permitiendo iterar y perfeccionarlos más rápidamente, fomentando avances tecnológicos.

CAPÍTULO 3

Objetivos

3.1. Objetivo general

Desarrollar una herramienta de simulación que permita modelar el comportamiento piezoeléctrico, usando *Deep Learning*.

3.2. Objetivos específicos

- Identificar las bases teóricas necesarias para comprender el fenómeno físico piezoeléctrico que se busca modelar.
- Desarrollar una Red Neuronal Informada por la Física (PINN) que tenga aplicación con piezoelectricidad.
- Optimizar los hiperparámetros de la PINN para incrementar la precisión y eficiencia del modelo.
- Validar la precisión del modelo.

CAPÍTULO 4

Aprendizaje automático informado por la física

4.1. Ecuaciones diferenciales

En esta sección se presenta una base teórica breve para las ecuaciones diferenciales. Se introduce el concepto y definición de una ecuación diferencial, una clasificación de estas y algunos operadores vectoriales diferenciales que se emplean comúnmente. En este contexto, un operador es una herramienta matemática que actúa sobre funciones o campos vectoriales para transformarlos, obteniendo información clave sobre sus propiedades. Entre los operadores más utilizados se encuentran el gradiente, que indica la dirección y magnitud del cambio más rápido de una función escalar, y la divergencia, que mide la tendencia de un campo vectorial a expandirse o contraerse en un punto. [17]

4.1.1. Concepto y definición

Una ecuación diferencial, es una ecuación que relaciona una o más funciones y sus derivadas. Las incógnitas de la ecuación son las funciones mencionadas y uno de los principales objetivos es encontrar la solución de estas [18]. Las ecuaciones diferenciales son usadas para describir matemáticamente un cambio continuo de un sistema físico. Comúnmente son utilizadas en campos como biología, física, economía y ciencias sociales [1]. Dos ejemplos de ecuaciones diferenciales son los siguientes,

$$m\frac{d^2u}{dt^2} = F, \tag{1}$$

$$\frac{\partial U}{\partial t} = k\frac{\partial^2 U}{\partial x^2}. \tag{2}$$

La Ecuación 1, busca explicar el movimiento de un cuerpo cuando se le aplica una fuerza. Describe como cambia la velocidad del cuando se le aplica una fuerza. Para esto relaciona la

segunda derivada del desplazamiento u, la masa m del cuerpo y una fuerza F aplicada a este [19]. La Ecuación 2, describe el cambio de la temperatura U en un cuerpo unidimensional a lo largo del tiempo y de la posición x a lo largo del cuerpo, así como de una constante de difusión térmica k [1].

Es importante evidenciar que se pueden relacionar distintas funciones y que las derivadas de estas pueden darse con respecto a una o más variables. Por lo tanto, usando una definición más formal, una ecuación que contiene a las derivadas de una o más funciones desconocidas (o variables dependientes) respecto a una o más variables independientes, se conoce como ecuación diferencial [18].

Existen distintas notaciones para expresar las derivadas en las ecuaciones diferenciales [18]. Se utilizarán tres notaciones a lo largo de este trabajo. La principal será la **notación de Leibniz**,

$$\frac{dy}{dx}, \quad \frac{d^2y}{dx^2}, \quad \frac{d^3y}{dx^3}, \ldots,$$

$$\frac{\partial y}{\partial x}, \quad \frac{\partial^2 y}{\partial x^2}, \quad \frac{\partial^3 y}{\partial x^3}, \ldots,$$

para derivadas totales, o para derivadas parciales, respectivamente. En la Ecuación 2 se observa el uso de esta notación, donde la función U se puede derivar con respecto a dos distintas variables independientes. Es importante evidenciar, que, con derivadas parciales, las segundas derivadas se pueden realizar con respecto a la misma variable de la primera derivada, o con respecto a otra variable independiente.

También se puede emplear la **notación prima**, donde el número de apóstrofes representa el grado de la derivada,

$$y', \quad y'', \quad y''', \ldots$$

Finalmente, se tiene la **notación de subíndice**, que indica las variables independientes con respecto a las que se está derivando,

$$u_x, u_y, u_{xx}, u_{yy}, u_{xy}, u_{yx}, \ldots$$

Esta notación sirve para proporcionar una forma más compacta de escribir derivadas parciales. Cuando se usa doble subíndice, significa una derivada de grado 2, con respecto a las variables indicadas en el subíndice. El orden de las derivadas parciales van de acuerdo con los subíndices es de izquierda a derecha.

4.1.2. Clasificación de ecuaciones diferenciales

Según su definición, una ecuación diferencial no solo involucra funciones cuyas derivadas se dan solo con respecto a una variable independiente, sino que puede ser respecto a más de

una. Por esto, se genera una clasificación dependiendo de las derivadas presentes. Una ecuación diferencial ordinaria (ODE por sus siglas en inglés) es aquella que contiene derivadas totales de posiblemente distintas funciones, con respecto a una única variable independiente. Las ODEs son usadas principalmente en la modelación de sistemas mecánicos, reacciones químicas, circuitos eléctricos y dinámicas poblacionales. Un ejemplo de ODE es la Ecuación 1. El orden de una ecuación diferencial es el grado de la mayor derivada presente en la ecuación. Simbólicamente, una ODE de n-ésimo orden se expresa de la siguiente manera [20]

$$F(x, y, y', \ldots, y^{(n)}) = 0.$$

En contraste, una ecuación diferencial parcial (PDE por sus siglas en inglés), es una ecuación diferencial que involucra una o más funciones de dos o más variables y sus derivadas parciales. Esto quiere decir que las funciones dependen de dos o más variables independientes. Un ejemplo de una PDE es la Ecuación 2. El orden de una PDE significa lo mismo que para una ODE, es decir, al grado de la derivada más alta presente en la ecuación. [1]

La forma general de una PDE escalar de primer orden es [18]

$$F(x, y, u, u_x, u_y) = 0,$$

donde u es la función desconocida, y x e y son las variables independientes. Para una PDE escalar de segundo orden, su forma más general es [1]

$$F(x, y, u, u_x, u_y, u_{xx}, u_{xy}, u_{yy}) = 0.$$

Por otra parte, algunas PDEs populares son las siguientes [21]:

- **Ecuaciones de Maxwell**: Las ecuaciones de Maxwell describen el comportamiento de los campos eléctricos y magnéticos, estableciendo la relación entre las fuerzas de ambos campos, con la carga eléctrica y la densidad de flujo magnética.

- **Ecuaciones de Navier-Stokes**: Las ecuaciones de Navier-Stokes describen el movimiento de los fluidos, relacionando la velocidad, densidad, presión y viscosidad de los fluidos.

- **Ecuación de Schrödinger**: Fundamental en la mecánica cuántica, describe cómo evoluciona en el tiempo el estado cuántico de un sistema, al relacionar la masa, energía potencial y energía total de una partícula.

- **Ecuaciones de la relatividad general de Einstein**: Describen cómo la masa y la energía influyen en la curvatura del espacio-tiempo, ayudando a explicar fenómenos gravitacionales como la formación de agujeros negros y la expansión del universo.

4.1.3. Condiciones iniciales y de frontera

Las ecuaciones diferenciales suelen tener familias de funciones como soluciones, las cuales pueden ser múltiples e incluso infinitas. Por lo tanto, para encontrar una solución específica para la ecuación diferencial, se definen condiciones extra, que la función debe satisfacer [20]. Estas condiciones se clasifican como condiciones iniciales y de frontera.

4.1.3.1. Condiciones iniciales

Al estudiar ecuaciones diferenciales relacionadas con fenómenos físicos, una condición inicial establece el estado del sistema en algún punto particular del tiempo t_0 [22]. En este caso, el tiempo, una variable independiente, es una forma efectiva y sencilla de comprender la condición. Sin embargo, se podrían dar condiciones iniciales para otras variables independientes. Las condiciones iniciales se pueden dar para valores de la función y también para derivadas de esta.

Siguiendo con la Ecuación 2, un ejemplo de condición inicial es

$$U(x, 0) = f(x).$$

Lo cual quiere significa que en el tiempo inicial (en este caso cero), el objeto tiene una distribución de temperatura, es decir, cada punto del objeto estudiado tiene una temperatura dada por la función $f(x)$. Conforme pase el tiempo, la temperatura irá cambiando, pero ese es el estado inicial del sistema.

4.1.3.2. Condiciones de frontera

Cuando se modela un sistema usando una ecuación diferencial parcial (PDE), es necesario definir el problema en un dominio específico Ω. Este dominio puede ser, por ejemplo, una varilla unidimensional limitada por un intervalo, una región en el plano, o un volumen tridimensional [1]. Definir el funcionamiento de la función o funciones de la ecuación diferencial en Ω, es necesario para encontrar una solución única del problema y se describe mediante las condiciones de frontera [18]. Estas condiciones pueden ser de varios tipos, las cuales se describen a continuación [1]:

- **Condición de Dirichlet**: Se especifican los valores de la función o funciones solución en distintos puntos de la frontera. La forma en la que se define esto es

  $$u(x) = g(x) \quad \text{para } x \in \partial\Omega,$$

  donde $g(x)$ es una función dada y $\partial\Omega$ es la frontera del dominio Ω .

- **Condición de Neumann**: El valor de la derivada normal de la función o funciones es especificado en los puntos de la frontera. Esto se define como

  $$\frac{\partial u}{\partial n}(x) = h(x) \quad \text{para } x \in \partial\Omega,$$

donde $h(x)$ es una función preestablecida y $\frac{\partial u}{\partial n}$ es la derivada de u en la dirección normal a la frontera.

- **Condición de Robin**: Es una combinación de las condiciones de Dirichlet y Neumann. Su forma es

$$\alpha u(x) + \beta \frac{\partial u}{\partial n}(x) = q(x) \quad \text{para } x \in \partial\Omega,$$

donde $\alpha \in \mathbb{R}$, $\beta \in \mathbb{R}$ y $q(x) \in \mathbb{R}$ son especificados en el problema.

4.1.4. Operadores comunes de identidades de cálculo vectorial

A continuación, se mencionarán algunos operadores comunes que se encuentran al expresar ecuaciones diferenciales, cuya definición se dió al inicio de este capítulo. [17]

4.1.4.1. Divergencia

La divergencia de un campo vectorial es una operación escalar que indica si las líneas de flujo en el campo son paralelas o no. Un ejemplo es el flujo de un gas en un tubo, donde las líneas permanecen paralelas, si el tubo es uniforme. Sin embargo, si el tubo se estrecha, las líneas de flujo convergen. La divergencia ayuda a entender si un campo vectorial se expande o converge hacia cierto punto. Es una medida importante en áreas de la física, como en el análisis de flujos de fluidos y campos electromagnéticos.

La divergencia en d dimensiones se expresa de la siguiente forma,

$$\nabla \cdot \mathbf{F} = \sum_{i=1}^{d} \frac{\partial F_i}{\partial x_i},$$

donde $\mathbf{F} = (F_1, F_2, \ldots, F_d)$ es el campo vectorial en d dimensiones y F_i es la componente del campo en la dirección x_i con $i = 1, 2, \ldots, d$.

4.1.4.2. Gradiente

El gradiente es la derivada de una función escalar. Cuando se aplica produce un vector. Este describe la dirección y magnitud del mayor cambio de una función escalar en un espacio dado. Esto quiere decir que este vector apunta en la dirección donde está la pendiente más pronunciada de la función.

Para una función escalar en d dimensiones f, el gradiente se expresa de la siguiente forma,

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_d} \right),$$

donde f es una función escalar de d variables $x_1, x_2, \ldots, x_d$, y el gradiente ∇f es un vector cuyas componentes son las derivadas parciales de f respecto a cada variable.

4.1.4.3. Laplaciano

El Laplaciano es un operador que se aplica a funciones escalares o vectoriales. En el caso de una función escalar, el Laplaciano mide la suma de las segundas derivadas parciales respecto a cada una de las variables espaciales, proporcionando información sobre la curvatura de la función en un punto dado. Es utilizado en diversas áreas de la física y las matemáticas, especialmente en ecuaciones diferenciales parciales, como en la ecuación de calor, la ecuación de Poisson y la ecuación de ondas.

Para una función escalar f en d dimensiones, el Laplaciano se puede expresar de la siguiente forma,

$$\Delta f = \sum_{i=1}^{d} \frac{\partial^2 f}{\partial x_i^2},$$

donde f es una función escalar de d variables $x_1, x_2, \ldots, x_d$, y Δf representa la suma de las segundas derivadas parciales de f respecto a cada variable x_i.

4.2. Piezoelectricidad

En esta sección se da una introducción a la piezoelectricidad, mencionando su historia, su funcionamiento físico y su descripción matemática.

4.2.1. Historia

La piezoelectricidad (del griego *piezein*, πιέζω, que significa estrujar o apretar) es un fenómeno en el cual ciertos cristales generan una carga eléctrica cuando se les aplica una presión mecánica. La historia de los materiales piezoeléctricos se remonta siglos atrás, cuando nativos de India se percataron de que un cristal conocido como turmalina, atraía a cenizas calientes cuando se ponía en contacto con ellas. A partir de esto, en 1756, el físico alemán Aepinus describió el origen eléctrico de este comportamiento. Se nombró piroelectricidad en 1824 y se trata de una polarización en el material debido a la absorción de energía calorífica. Esta propiedad está estrechamente relacionada con la piezoelectricidad ya que ambos fenómenos involucran la generación de carga eléctrica en cristales bajo diferentes estímulos. [23]

El primer artículo científico publicado donde se menciona el efecto piezoeléctrico y la cristalografía fue publicado por los hermanos Pierre y Jacques Curie en 1880. Encontraron carga en cristales como la mencionada turmalina, cuarzo y sal de Seignette, al aplicarles una fuerza mecánica. Ellos fueron los descubridores del efecto piezoeléctrico directo. El efecto

inverso, conocido como el efecto piezoeléctrico indirecto, donde se genera una deformación en presencia de una carga o voltaje, fue descrito por Lippman, un físico francés, en 1881, al hacer una deducción matemática a partir de principios termodinámicos. [24]

Hasta inicios del siglo XX se empezaron a ver las primeras aplicaciones de estos materiales. La primera fue el sonar, en donde cuarzo piezoeléctrico se usaba para producir ondas ultrasónicas y como sensor. Luego de la segunda guerra mundial, se empezaron a realizar dispositivos más industriales que hicieron uso de materiales piezoeléctricos. Entre estos se encuentran pequeños micrófonos, acelerómetros, transductores cerámicos, entre otros. En la década de 1960, científicos japoneses inventaron nuevas aplicaciones, entre las que se encuentran filtros de señales, encendedores piezo cerámicos y transductores ultrasónicos de aire usados en medicina. [25]

Actualmente, la investigación se orienta al desarrollo de dispositivos piezoeléctricos compuestos, los cuales tienen múltiples capas piezoeléctricas muy delgadas, que permiten mayor deformación. También son ampliamente usados en control de sonido y vibración, donde dos materiales predominan. Uno es el Titanato de Zirconato de Plomo (PZT por sus siglas en inglés), que es usado comúnmente como actuador, teniendo alta precisión. Por otra parte, está el Fluoruro de Polivinilideno (PVDF por sus siglas en inglés), usado como sensor. [7], [24]

4.2.2. Descripción física del efecto piezoeléctrico

El efecto piezoeléctrico tiene dos formas de presentarse, como se mencionó previamente. El primero es el efecto piezoeléctrico directo, que se observa en la Figura 1. Acá, se genera un voltaje a partir de la aplicación de una fuerza mecánica al material, que es una deformación de este. El segundo es el efecto indirecto, en donde el material se deforma a partir de una aplicación de voltaje. [7]

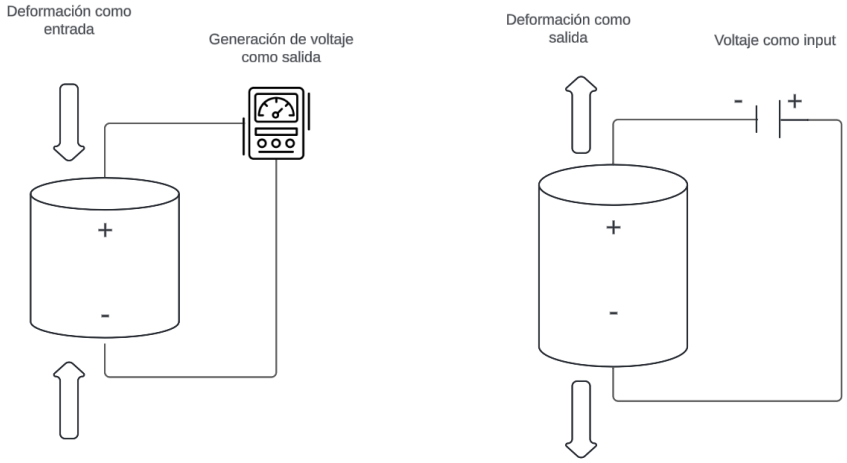


Figura 1: Efecto piezoeléctrico. Figura de la izquierda representa una generación de voltaje a partir de deformación, efecto directo. Figura de la derecha representa el efecto indirecto, que es una deformación por aplicación de voltaje.

Los materiales piezoeléctricos son cristales. Los cristales son materiales sólidos, cuyas moléculas, átomos o iones, están altamente ordenados microscópicamente [26]. Los cristales se pueden clasificar según su simetría, lo que se refiere a las formas en las cuales puede ser manipulado y seguir teniendo la misma apariencia. Un cristal podría ser rotado o reflejado y mantener su misma apariencia [27]. A partir de esto surgen 32 grupos de simetría, a los cuales pertenecen distintos cristales según la simetría que mantienen en un punto fijo de su estructura. De los 32 grupos de cristales, 11 grupos tienen centros de simetría y 21 no. De estos últimos, uno tiene una simetría especial. Los cristales de estos 20 grupos pueden generar electricidad por una fuerza mecánica, siendo piezoeléctricos [7]. De estos, 10 grupos son los cristales piroeléctricos.

Los cristales piezoeléctricos tienen una estructura característica. Cuando no sufren ninguna deformación, tienen una distribución balanceada de cargas positivas y negativas. Sin embargo, estos cristales son no centrosimétricos, lo cual significa que no existe un punto en el cristal que lo divida en dos mitades equivalentes. Esto implica que a pesar de este balance cuando no hay deformación, cuando se aplica una fuerza al material, las moléculas, que son las que tienen las cargas, cambian su posición. Este cambio, modifica la distribución de las cargas y se genera una separación entre las cargas positivas y negativas, generando un potencial eléctrico en el cristal. Esto se observa en la Figura 2, donde están las cargas distribuidas aleatoriamente cuando el material está sin perturbación. Al momento de ser comprimido, las cargas de mueven y se alinean, generando el potencial mencionado. [28]

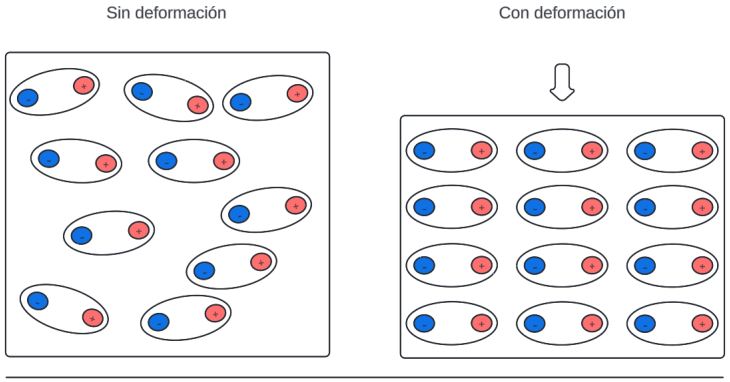


Figura 2: Modificación en la estructura de un material piezoeléctrico, cuando una fuerza es aplicada.

4.2.3. Descripción matemática del efecto piezoeléctrico

La relación entre los distintos factores involucrados en interacciones termodinámicas entre las propiedades térmicas, mecánicas y eléctricas de un cristal se puede observar en la Figura 3. De este diagrama se puede observar la relación entre los factores que generan el efecto piezoeléctrico directo y el indirecto. Para el directo, una deformación genera un desplazamiento dieléctrico, que a su vez genera un campo eléctrico. En el efecto indirecto, un campo eléctrico genera una deformación mecánica en el cristal. En las aplicaciones consideradas para el presente trabajo, no se toman en cuenta efectos térmicos, debido a que se asumen procesos adiabáticos. [7]

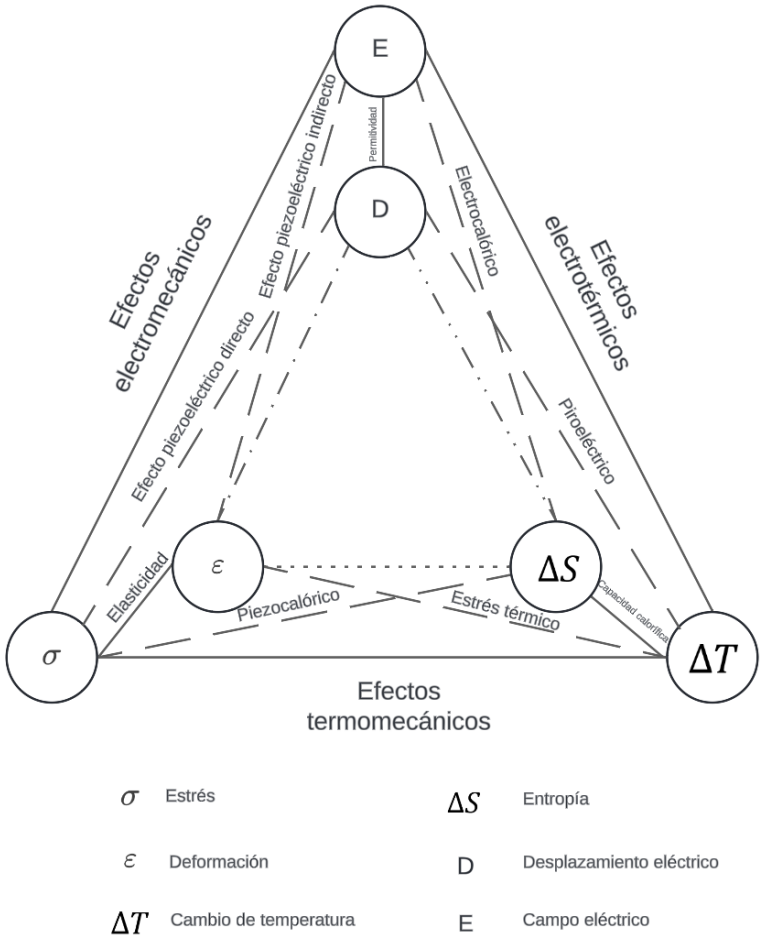


Figura 3: Diagrama de Heckmann, que relaciona las propiedades térmicas, mecánicas y eléctricas de un cristal en interacciones termodinámicas [7].

El fenómeno piezoeléctrico es modelado por medio de ecuaciones constitutivas, que permiten expresar las relaciones entre las propiedades del diagrama de la Figura 3 [23]. Estas ecuaciones constitutivas pueden tener la siguiente forma,

$$\vec{\sigma} = c^E \vec{S} - e^T \vec{E} \tag{3}$$

$$\vec{D} = e\vec{S} + \epsilon^S \vec{E} \tag{4}$$

En estas ecuaciones, $\vec{\sigma}$ corresponde al vector de estrés mecánico. Luego se tiene c^E, que es un tensor de elasticidad propio del material piezoeléctrico que se está estudiando. El tensor $\vec{S}$ corresponde a la deformación del material, e^T es el tensor de coeficientes piezoeléctricos, $\vec{E}$ es el vector de campo eléctrico y finalmente $\vec{D}$ es el vector de desplazamiento dieléctrico. La primera ecuación se refiere al efecto piezoeléctrico indirecto, en donde se genera un estrés mecánico $\vec{\sigma}$, en respuesta a un campo eléctrico aplicado $\vec{E}$. La segunda ecuación describe el efecto piezoeléctrico directo, donde se genera un desplazamiento eléctrico $\vec{D}$, en respuesta a un estrés mecánico $\vec{\sigma}$ que está relacionado con la deformación $\vec{S}$. [7]

En tres dimensiones, la relación entre estos campos se puede expresar de forma tensorial,

$$\begin{pmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \\ D_1 \\ D_2 \\ D_3 \end{pmatrix} = \begin{pmatrix} c_{11}^E & c_{12}^E & c_{13}^E & \cdot & \cdot & \cdot & \cdot & \cdot & -e_{13} \\ c_{12}^E & c_{11}^E & c_{13}^E & \cdot & \cdot & \cdot & \cdot & \cdot & -e_{13} \\ c_{13}^E & c_{13}^E & c_{33}^E & \cdot & \cdot & \cdot & \cdot & \cdot & -e_{33} \\ \cdot & \cdot & \cdot & c_{44}^E & \cdot & \cdot & \cdot & -e_{15} & \cdot \\ \cdot & \cdot & \cdot & \cdot & c_{44}^E & \cdot & -e_{15} & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & c_{66}^E & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & e_{15} & \cdot & \epsilon_{11}^S & \cdot & \cdot \\ \cdot & \cdot & \cdot & e_{15} & \cdot & \cdot & \cdot & \epsilon_{11}^S & \cdot \\ e_{13} & e_{13} & e_{33} & \cdot & \cdot & \cdot & \cdot & \cdot & \epsilon_{33}^S \end{pmatrix} \begin{pmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \\ E_1 \\ E_2 \\ E_3 \end{pmatrix} \tag{5}$$

Existen distintas formas de expresar las ecuaciones constitutivas, sin embargo, esta forma es la más utilizada para realizar ciertas simulaciones [7].

4.2.4. Viga en voladizo piezoeléctrica

Existen distintos dispositivos que pueden ser creados a partir de un material piezoeléctrico. De particular interés para este trabajo, se busca estudiar una viga en voladizo piezoeléctrica, como la que se observa en la Figura 4. Esta viga está hecha de un material piezoeléctrico, el cual puede deformarse por una carga eléctrica o generar una cuando se deforma. La viga se modela en dos dimensiones.

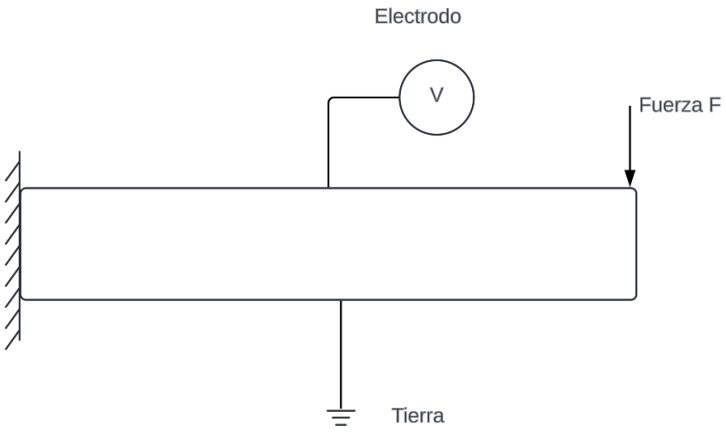


Figura 4: Viga en voladizo piezoeléctrica.

Para modelar esta viga en voladizo, se pueden usar las ecuaciones constitutivas de la piezoelectricidad. Estas son expresadas por la Ecuación 3 y la Ecuación 4. Además, se introduce otro conjunto de ecuaciones diferenciales para poder modelar el sistema completo [29],

$$\nabla \cdot \sigma + \mathbf{f} = 0, \tag{6}$$
$$\nabla \cdot D = 0. \tag{7}$$

Donde $\mathbf{f}$ es un vector que indica una posible fuerza aplicada a la viga, lo cual puede servir para el efecto piezoeléctrico directo. Esta fuerza se puede observar en la Figura 4. Es opcional, en caso se modele el efecto directo, lo cual genera voltaje. Si se modela el indirecto, la fuerza es 0 y se aplica un voltaje. Estas ecuaciones son elípticas, debido a que se asume un estado estable o estacionario. Sin embargo, se podrían añadir términos para modelar el sistema respecto al tiempo [7]. Las funciones para este caso solo tienen dos entradas, que son las coordenadas del objeto en dos dimensiones. Este conjunto de ecuaciones no es aplicable únicamente para la viga en voladizo, puede ser para cualquier dispositivo piezoeléctrico. Se incluyen acá para poder integrarlas con las condiciones de frontera propias de esta viga en voladizo.

De la Figura 4, también se puede observar que la viga está fija del lado izquierdo y está libre del lado derecho. Esto permite un movimiento oscilante de la viga. Como es un sistema en estado estacionario, solo se tienen condiciones de frontera [29]

$$\begin{cases} u(x) = \bar{u}, & \forall x \in \Omega_u, \\ \varphi(x) = \bar{\varphi}, & \forall x \in \Omega_\varphi, \\ \sigma(x) \cdot \mathbf{n} = \mathbf{0}, & \forall x \in \Omega_t, \\ D \cdot \mathbf{n} = 0 & \forall x \in \Omega_p. \end{cases} \tag{8}$$

Donde $\bar{u}$ es un desplazamiento definido, $\bar{\varphi}$ un potencial eléctrico definido y $\mathbf{n}$ el vector

normal en las fronteras. También, Ω_u corresponde a todos los puntos donde se desea definir un desplazamiento con cierto valor; Ω_φ se refiere a la frontera donde se define un potencial establecido que debe cumplir la función φ; Ω_t se refiere a la frontera donde se busca aplicar la derivada normal del estrés; Ω_p son aquellos puntos donde se debe aplicar la condición de frontera para el desplazamiento dieléctrico.

La viga en voladizo puede constar de distintas capas de materiales piezoeléctricos, como se observa en la Figura 5. En dado caso, es necesario especificar las propiedades de estos materiales. Esta configuración puede causar que una capa se comprima y la otra se estire. Se pueden seguir aplicando las mismas condiciones de frontera y ecuaciones. [29]

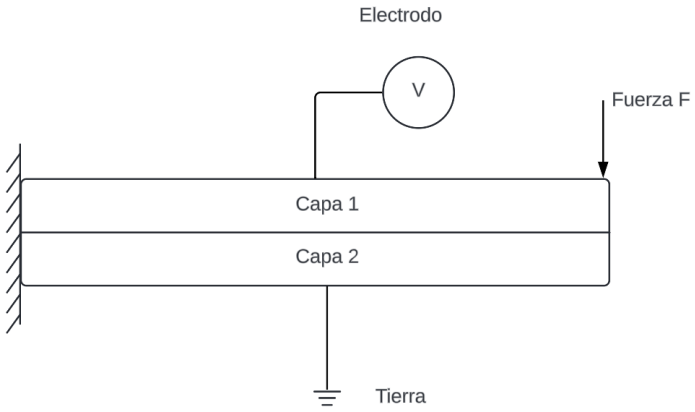


Figura 5: Viga en voladizo con dos capas piezoeléctricas.

La viga se modela en dos dimensiones, por lo que los tensores de las ecuaciones constitutivas de la Ecuación 5 cambian, debido a que ya no se manejan en tres dimensiones,

$$\begin{pmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \\ D_1 \\ D_2 \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & 0 & e_{11} & e_{31} \\ c_{12} & c_{22} & 0 & e_{13} & e_{33} \\ 0 & 0 & c_{33} & e_{14} & e_{34} \\ e_{11} & e_{13} & e_{14} & -\epsilon_1 & 0 \\ e_{31} & e_{33} & e_{34} & 0 & -\epsilon_2 \end{pmatrix} \begin{pmatrix} S_1 \\ S_2 \\ S_3 \\ -E_1 \\ -E_2 \end{pmatrix}. \tag{9}$$

En este caso se modificaron ligeramente las notaciones de algunos elementos.

Este problema se puede modelar en términos de desplazamiento y de potencial eléctrico, para tener concordancia con las condiciones de frontera. Por lo tanto, se pueden hacer sustituciones por la deformación S y el campo eléctrico E, en la matriz presentada anteriormente. [30]

Para hacer estas sustituciones se pueden usar las siguientes igualdades,

$$\begin{pmatrix} S_1 \\ S_2 \\ S_3 \end{pmatrix} = \begin{pmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \\ \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \end{pmatrix}, \qquad \begin{pmatrix} E_1 \\ E_2 \end{pmatrix} = -\begin{pmatrix} \frac{\partial \varphi}{\partial x} \\ \frac{\partial \varphi}{\partial y}. \end{pmatrix} \tag{10}$$

4.3. Introducción a *Machine Learning* (ML)

En esta sección se presenta una descripción básica del concepto de *machine learning* o aprendizaje automático y una clasificación de este.

4.3.1. Concepto

La inteligencia artificial (IA) es una ciencia que estudia el desarrollo de acciones racionales por parte de agentes inteligentes. Estos últimos generalmente se implementan usando máquinas o computadoras, buscando simular la inteligencia humana y la capacidad de solución de problemas. La IA abarca distintos campos, que van desde lógica, decisiones en problemas de juegos como ajedrez, demostración de teoremas matemáticos, aprendizaje, robótica, procesamiento de lenguaje natural, entre otros. Entre estas áreas, se encuentra una que se ha vuelto popular recientemente y es el *machine learning*. [31]

El *machine learning* se enfoca en el uso de algoritmos o modelos que involucran aprendizaje estadístico y optimización. Estos permiten a las máquinas identificar patrones en conjuntos de datos, usualmente para generar una predicción o clasificación, sin necesidad de una intervención manual humana. El proceso típico de aprendizaje toma tres pasos. El primero es un proceso de decisión, donde, a partir de los datos y una serie de pasos, se genera una conjetura del patrón que se busca. Luego, se usa una función de pérdida, que mide que tan buena o mala es la conjetura realizada con respecto al patrón real. Finalmente, se usa un proceso de optimización, que modifican parámetros del modelo de ML para que se ajuste al patrón buscado. [32]

Los algoritmos de ML relativamente básicos, en muchas ocasiones requieren de la creación de características de los datos [33]. Por ejemplo, es muy probable que si los datos son audio y el modelo los recibe sin preprocesar, no podrá aprender ningún patrón. Sin embargo, si se obtienen características específicas del audio, como sus frecuencias o su amplitud, es más factible que el modelo genere buenos resultados. Este proceso puede ser manual y complejo, requiriendo conocimiento del dominio, dando espacio a que el conjunto de características elegido no sea el adecuado y no se obtenga un buen modelo.

El aprendizaje por representación es un conjunto de métodos que permiten a una máquina aprender a partir de datos sin preprocesar y encontrar características necesarias para encontrar patrones en los datos, sin intervención humana. Esto fue la base para una nueva subrama de ML, conocida como aprendizaje profundo o *deep learning* (DL). En la Figura 6, se puede observar la jerarquía de la inteligencia artificial. El *deep learning* usa múltiples capas de representación para encontrar las características que permiten describir un patrón o función muy compleja en un conjunto de datos. Las características elegidas para esta representación se eligen automáticamente, eliminando el problema de la elección manual. [34]

El *deep learning*, se apoya en el uso de redes neuronales para sus capas de representación. Estas, brevemente, se pueden describir como un conjunto de nodos que buscan simular el comportamiento de las neuronas humanas [31]. Las redes neuronales se ordenan por capas que tienen múltiples nodos, los cuales se conectan con los de las capas adyacentes y se basan

Figura 6: Jerarquía de la inteligencia artificial [35]

.

en el paso de información entre estas capas [34]. El término *deep* en DL, se refiere a la cantidad de capas que hay en una red neuronal. Un diagrama básico de una red neuronal se encuentra en la Figura 7.

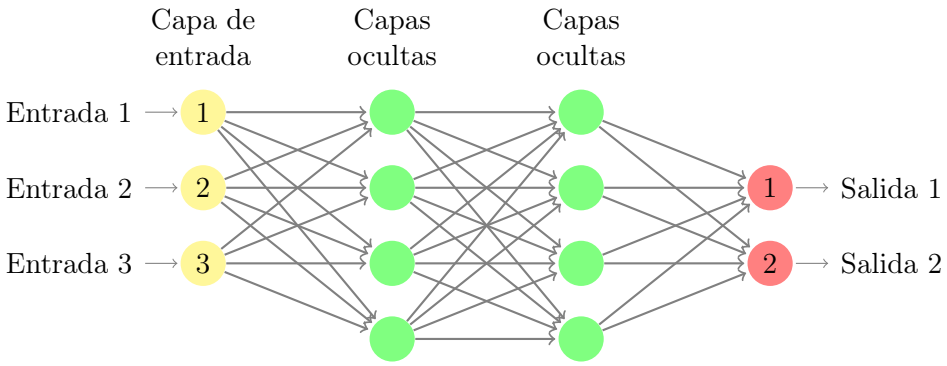


Figura 7: Red neuronal con capas de entrada, ocultas y de salida.

Las redes neuronales y el *deep learning*, han demostrado ser una herramienta muy efectiva para resolver distintos tipos de problemas. Un ejemplo de esto es el reconocimiento de imágenes de texto escrito a mano y la identificación de elementos de un video en tiempo real para vehículos autónomos, como transeúntes u otros vehículos [34]. También, recientemente una de sus aplicaciones más populares es la inteligencia artificial generativa y procesamiento de lenguaje natural, como ChatGPT [36].

4.3.2. Clasificación

Los algoritmos de *machine learning* se pueden clasificar según el tipo de problema que buscan resolver y el uso que le dan a los datos que tienen a su disposición [33]. Esta clasificación se aplica a todos los algoritmos que engloba el ML según la Figura 6.

- **Aprendizaje supervisado**

  El aprendizaje supervisado hace uso de un conjunto de datos previamente etiquetados. Esto quiere decir, que los datos traen un conjunto de características y una variable cuyos valores son correctos. Esta comúnmente se conoce como verdad fundamental o *ground truth*. El objetivo del modelo es ajustar sus parámetros hasta encontrar el patrón en las características de los datos, según la verdad fundamental. Una vez se logre esto, se puede hacer clasificación o predicciones con nuevas observaciones de los datos.

  Dos de los principales problemas que se buscan resolver con este enfoque son la regresión y la clasificación. La regresión consiste en usar modelos de ML para encontrar una función que mapee las características del conjunto de datos, a un valor continuo, según la *ground truth*. Por ejemplo, se puede predecir el valor de una vivienda o la temperatura futura de un sistema. Por otra parte, en la clasificación, se busca predecir una variable categórica. Esto significa que hay un conjunto limitado de clases que

se pueden obtener como salida del modelo [32]. Se puede clasificar, por ejemplo, una grabación del corazón como anómala o normal.

- **Aprendizaje no supervisado**

  El aprendizaje no supervisado usa algoritmos que permiten analizar conjuntos de datos sin etiquetar. Se busca encontrar patrones ocultos o agrupar los datos, sin la necesidad de una intervención humana. Es muy útil para una exploración profunda de datos, permitiendo encontrar características o relaciones sutiles. También puede servir para una segmentación de clientes en una empresa o la recomendación de productos a clientes.

- **Aprendizaje semisupervisado**

  El aprendizaje semisupervisado es un punto medio entre el aprendizaje supervisado y el aprendizaje no supervisado. Estos algoritmos utilizan un conjunto de datos relativamente pequeño el cuál está etiquetado. Este guía el aprendizaje para clasificación y extracción de características, de un conjunto de datos mucho más grande. Si es muy costoso etiquetar los conjuntos de datos o no se tienen suficientes, este tipo de aprendizaje puede ser una solución efectiva.

4.4. Aspectos básicos de redes neuronales

A continuación, se presenta el concepto de las redes neuronales básicas, ciertos elementos que las conforman y que permiten su funcionamiento y aprendizaje. El enfoque principal será en las redes neuronales más básicas conocidas como *feedforward*.

4.4.1. Red Neuronal *Feedforward*

Una neurona es una célula cerebral que procesa información. Para esto la recopila, la analiza y la transmite a otras neuronas, usando señales eléctricas. Inspirándose en este funcionamiento, la neurociencia computacional y la inteligencia artificial han desarrollado modelos matemáticos conocidos como **neuronas artificiales** o **nodos**, términos que se usarán intercambiadamente a lo largo del trabajo, los cuales son las unidades básicas de las redes neuronales actuales. El funcionamiento de los nodos está definido por una función lineal que busca aprender una discriminación entre un conjunto de datos. [31]

Una variación específica de nodo, conocida como perceptrón, puede tener un conjunto de una o más entradas, a las cuales están asociados pesos, que son parámetros que se ajustarán de acuerdo con algún tipo de los aprendizajes mencionados en secciones anteriores. Además, se tiene un sesgo en la función lineal que ayuda a aprender funciones más complejas.[34].

Para un solo nodo, la función es de la siguiente forma,

$$z = \sum_{i=1}^{d} w_i x_i + b, \tag{11}$$

donde:

- w_i son los pesos que dan la importancia a cada característica de la entrada x_i del conjunto de datos,
- x_i es la i-ésima entrada,
- b es el sesgo que permite ajustar la función lineal,
- d es la cantidad de entradas,
- z es la salida de la neurona, que es un valor escalar.

Generalmente, se busca el aprendizaje de funciones complejas por medio de los nodos, por lo que se aplica una función de activación al resultado de la ecuación anterior. La función de activación introduce una no linealidad en la salida de la neurona, lo cual le permite aprender patrones más complejos.[34]

La aplicación de la función de activación es de la siguiente forma,

$$a = \sigma(z), \tag{12}$$

donde σ es la función de activación aplicada al nodo. En este trabajo, entiende que la función de activación σ puede aplicarse tanto a escalares como a vectores. Esto significa que, cuando se utilice $\sigma(z)$, se entenderá que σ se aplica a cada componente de z si este es un vector, o directamente si z es un escalar. Se darán más detalles en las siguientes secciones sobre funciones de activación.

Una forma gráfica de un nodo se puede observar en la Figura 8. Donde se observan las distintas entradas, la suma de estas con los pesos y sesgos y luego la aplicación de la función de activación para generar la salida.

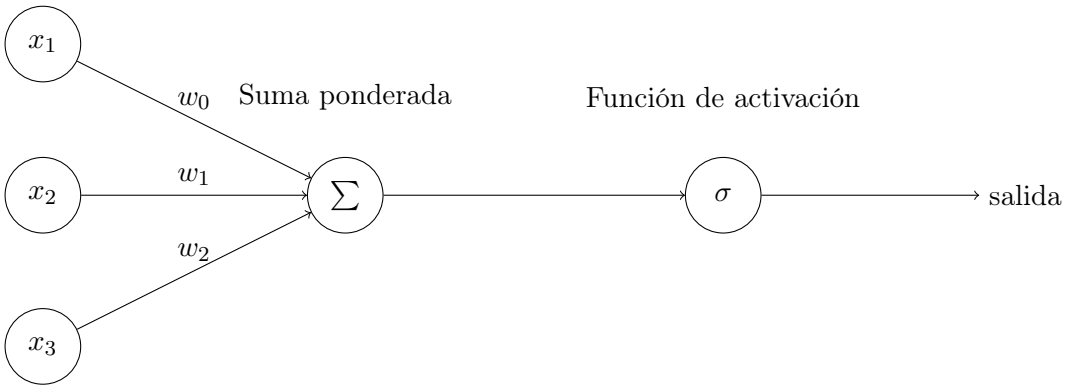


Figura 8: Diagrama de un perceptrón.

Cuando los perceptrones se combinan y se organizan en capas, se forma una red neuronal *feedforward*. En su forma más básica, esto es un perceptrón multicapa (MLP por sus siglas en inglés). Las capas del MLP son consecutivas y se caracterizan por tener a todas las unidades

de una capa, totalmente conectadas a las neuronas de la capa adyacente siguiente, como se observa en la Figura 7. Esto quiere decir que un nodo de una capa está conectada a todos los nodos de la siguiente capa. [31]

Para una red neuronal *feedforward*, se tienen tres tipos de capas, que son la capa de entrada, las capas ocultas y la capa de salida [34].

En la capa de entrada es donde inicia el flujo de información, siendo este el lugar donde ingresan los datos, de los cuales se desea encontrar un patrón o aproximar una función objetivo.

Luego de la capa de entrada, se tiene un conjunto de capas ocultas que contienen múltiples nodos o neuronas, cuya cantidad es flexible y se puede ajustar de acuerdo a las necesidades del problema. Se pueden tener tantas capas ocultas como se necesite al definir la arquitectura. Estas capas, sirven para aprender automáticamente características de los datos, como se mencionó en el aprendizaje por representación. Cada capa, puede aprender distintas características de los datos, permitiendo a la red crear abstracciones complejas de la información.

Finalmente se llega a la capa de salida, que es la responsable de producir el resultado final de la red. Puede servir para clasificar datos o realizar una regresión de estos. El nombre *feedforward* se da debido a que la información viaja desde la capa de entrada hasta la de salida de forma secuencial, sin ciclos ni retornos, en un paso hacia adelante.

Basado en la representación por capas, se puede usar notación vectorial para definir la salida de cada capa i del MLP como a_i y los sesgos b_i. Los pesos que conectan los nodos de la capa $(i-1)$ y los de la capa i se pueden denotar como la matriz W_i. La capa de entrada tiene un índice 0 y la de salida un índice k, donde $k+1$ representa la cantidad de capas en la red neuronal. Entonces a_i tiene la siguiente forma,

$$a_i(x) = \begin{cases} x, & i = 0 \\ \sigma_i(W_i a_{i-1} + b_i), & i = 1, 2, \ldots, k-1 \\ W_i a_{i-1} + b_i, & i = k. \end{cases}$$

Donde las dimensiones de cada variable están definidas de la siguiente forma,

- $x \in \mathbb{R}^{d_0}$ es el vector de entrada, donde d_0 es la dimensión de los datos de entrada,
- $W_i \in \mathbb{R}^{d_i \times d_{i-1}}$ es la matriz de pesos de la capa i, donde d_i es la cantidad de nodos en la capa i y d_{i-1} es la cantidad de nodos en la capa $(i-1)$,
- $b_i \in \mathbb{R}^{d_i}$ es el vector de sesgos para la capa i,
- $a_i(x) \in \mathbb{R}^{d_i}$ es el vector de salida de la capa i,
- σ_i es la función de activación aplicada en la capa i.

El objetivo principal de una red neuronal feedforward es aproximar una función f^*. Esta función podría ser un clasificador de aprendizaje supervisado, que busca asignar una

categoría a una entrada x, donde $y = f^*(x)$. Una red neuronal feedforward define una relación $y = f(x;\theta)$, donde se busca aprender los valores de los parámetros θ, siendo $\theta = \{W_i, b_i\}_{i=1}^{k}$, que resulten en la mejor función de aproximación para un patrón dado en los datos. Estos parámetros θ son el conjunto de pesos y sesgos de cada uno de los nodos de la red neuronal. [34]

4.4.2. Funciones de activación

Una red neuronal que no haga uso de funciones de activación se comporta de forma muy similar a un modelo de regresión lineal [31]. Sin funciones de activación, las combinaciones de los nodos solo pueden aproximar relaciones lineales entre las entradas y las salidas, pudiendo predecir funciones relativamente simples. Una regresión lineal es como la de la Figura 9, la cuál es un método estadístico que busca encontrar la relación entre una variable dependiente y una o más independientes [33]. El objetivo es encontrar la línea recta que más se ajusta a la relación entre los datos.

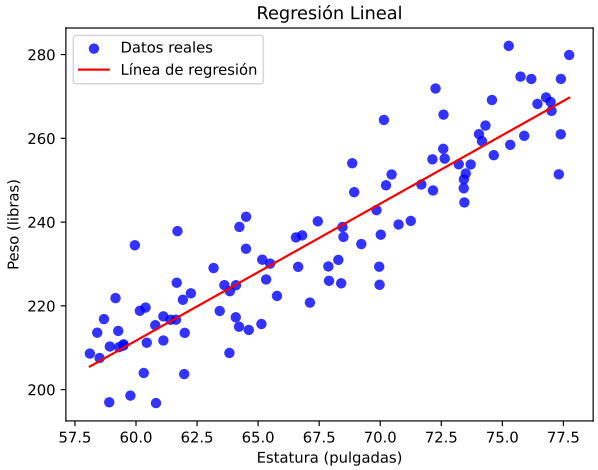


Figura 9: Ajuste de regresión lineal.

Al introducir funciones de activación a las capas de una red neuronal, se da la capacidad de capturar relaciones no lineales entre las variables de entrada y salida. De esta forma, la red neuronal puede aprender y representar patrones complejos, permitiendo modelar y predecir fenómenos más sofisticados. [31]

Las funciones de activación buscan imitar un mecanismo de las neuronas biológicas, donde estas se activan dependiendo de la intensidad de los estímulos nerviosos y el paso de información [31]. Se usan para computar un valor en base a la Ecuación 12, el cual se usa para decidir si el nodo se activa, es decir, si pasa información. Son principalmente empleadas por los nodos de las capas ocultas y existen diferentes tipos que pueden producir diversos resultados. Los casos de uso de estas son un área de investigación activa [34].

Existen funciones de activación binarias, las cuales indican si un nodo debe ser activado o no, en base a cierto valor límite [37]. Un ejemplo puede ser,

$$\sigma(z) = \begin{cases} 0 & \text{si } z < 0 \\ 1 & \text{si } z \geq 0 \end{cases}.$$

También están las funciones de activación lineales, donde el valor de activación es proporcional al input, lo cual, como su nombre lo indica, no añade ningún tipo de no linealidad a la red neuronal [37]. Si se usa esta función en todos los nodos, es equivalente a tener una red neuronal de una sola capa. Su forma es la siguiente,

$$\sigma(z) = z.$$

Finalmente, se encuentran las funciones de activación no lineales, las cuales introducen la no linealidad deseada para poder aprender patrones complejos en los datos [31]. A continuación, se presentan las más comunes.

Aunque se definen para un escalar x, es importante recordar que estas funciones pueden aplicarse de manera componente a componente cuando la entrada es un vector $\mathbf{z} \in \mathbb{R}^d$, es decir, cada componente del vector se procesa individualmente por la función de activación. Esto también es aplicable para las funciones de activación no lineales que se verán a continuación.

De forma que, la función de activación definida para un escalar z, es fácilmente extendible al caso vectorial. Para un vector $\mathbf{z} \in \mathbb{R}^d$, la función se aplica componente a componente, es decir,

$$\sigma(\mathbf{z}) = \begin{bmatrix} \sigma(z_1) \\ \sigma(z_2) \\ \vdots \\ \sigma(z_d) \end{bmatrix}$$

donde z_i es la i-ésima componente del vector $\mathbf{z}$.

Las funciones de activación que se verán a continuación, entran en el conjunto de funciones comúnmente conocidas como funciones de activación *squashing*. Esto quiere decir que comprimen el rango de las entradas a un intervalo finito, generalmente entre 0 y 1, o entre -1 y 1. Este comportamiento es útil para mantener los valores de salida dentro de un rango controlado, lo que facilita el entrenamiento de la red neuronal y previene que las activaciones crezcan de manera descontrolada. [34]

4.4.2.1. ReLU

Esta función de activación, conocida como Unidad Lineal Rectificada (ReLU por sus siglas en inglés), es de las más usadas actualmente y parece que tuviera el mismo mecanismo que una función de activación lineal. Sin embargo, debido a que divide el dominio en dos partes, como se observa en la Figura 10, permite no activar a todas las neuronas al mismo

tiempo. Es computacionalmente eficiente comparada con las siguientes funciones que se mencionarán y ayuda a la convergencia del aprendizaje de la red neuronal.

Se puede expresar como,

$$ReLU(z) = \text{máx}(0, z)$$

.

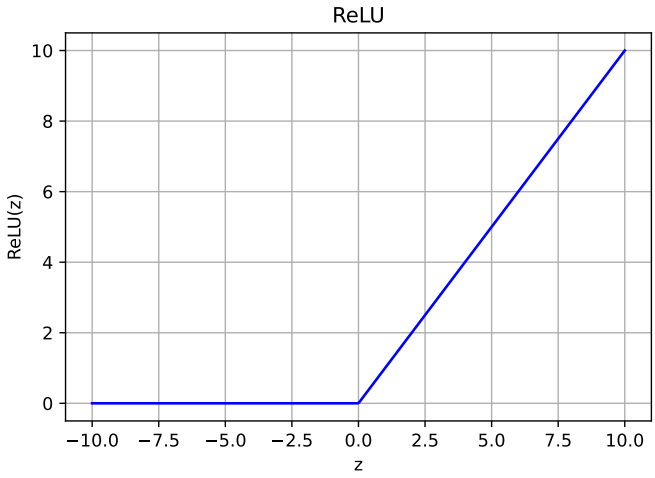


Figura 10: Función de activación ReLU.

4.4.2.2. Sigmoide

La función de activación sigmoide es muy común y era la más utilizada antes de ReLU. En la capa de salida es comúnmente usada para dar una probabilidad de que una observación pertenezca a una de dos clases. Su forma se puede observar en la Figura 11.

Se puede expresar de la forma,

$$Sigmoide(z) = \frac{1}{1 + e^{-z}}.$$

Al ser usadas en las capas ocultas, tienden a tener valores muy altos cuando x es muy positivo y valores muy bajos cuando x es muy negativo. Esto puede causar problemas en el aprendizaje basado en gradiente, que se verá más adelante. Por esto, es necesario evaluar en qué problemas es viable utilizarla.

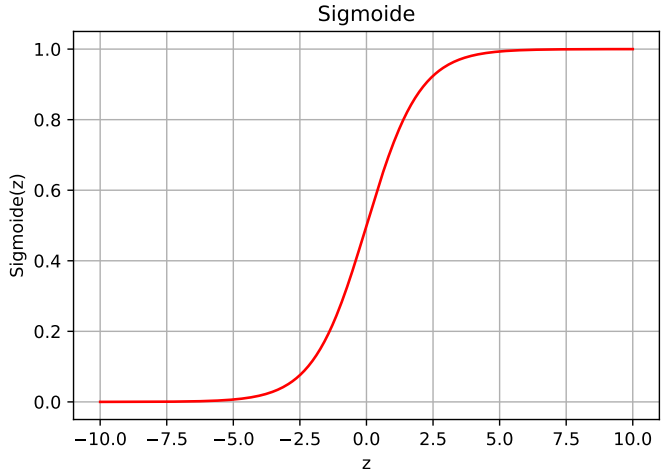


Figura 11: Función de activación sigmoide.

4.4.2.3. Tanh

La función de activación tanh, puede facilitar un aprendizaje más rápido a la red neuronal, debido a que está acotada entre -1 y 1, debido al equilibrio de los valores positivos y negativos. Se usa comúnmente en las capas ocultas de las redes neuronales feedforward. Puede llevar a una mejor convergencia que la función sigmoide. Su forma se puede observar en la Figura 12.

Su ecuación es la siguiente,

$$tanh(x) = \frac{e^z - e^{-z}}{e^z + e^{-z}}.$$

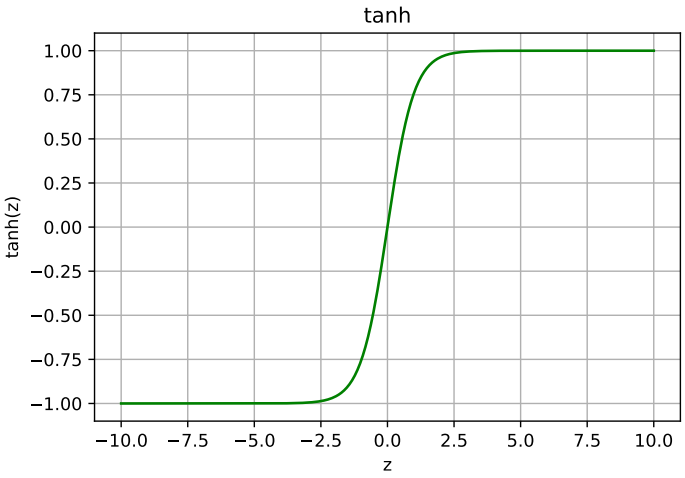


Figura 12: Función de activación tanh.

4.4.3. Elementos para el entrenamiento de una red neuronal

En un paso hacia adelante, los datos de entrada de la red neuronal viajan a través de todas las capas de una red neuronal y esta produce un resultado [34]. Sea una clasificación o una regresión, es necesario determinar qué tan bueno fue el resultado y optimizar la red para que se ajuste mejor a la función objetivo. Este proceso se repite por un número determinado de épocas o hasta cumplir con una métrica establecida. Cada época corresponde a un ciclo completo a través del conjunto de datos de entrenamiento, lo que permite que el modelo ajuste sus parámetros iterativamente a medida que aprende de los errores cometidos en las predicciones. El uso de múltiples épocas mejora la capacidad del modelo para generalizar, ya que permite la optimización progresiva de los pesos y sesgos de la red.

Durante el entrenamiento, en cada iteración dentro de una época, se calcula una función de pérdida para determinar qué tan bueno fue el resultado, y con base en esa métrica, se ajustan los parámetros del modelo para acercarse más a la función objetivo, por medio de *backpropagation*. Este proceso se repite por el número de épocas determinado, hasta que los parámetros converjan a un valor óptimo o se alcance una condición de parada, como la mejora en una métrica específica o la estabilización de la pérdida.

Para lograr el entrenamiento de la red, se utilizan distintos mecanismos en conjunto, que permiten iterar sobre los parámetros del modelo para ajustarlos. A continuación, consideramos estos elementos bajo un enfoque de aprendizaje supervisado, en el cual contamos con un conjunto de datos de entrenamiento $\{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$, donde $x^{(i)}$ representa las entradas y $y^{(i)}$ las etiquetas o valores objetivo correspondientes, y N es la cantidad total de ejemplos en el conjunto de datos.

4.4.3.1. Funciones de pérdida

El primero de estos mecanismos, es una función de pérdida, el cual es un método para evaluar que tan bien se ajusta la red neuronal a la función objetivo. Si las predicciones son totalmente erróneas, la función de pérdida será muy alta. Si son buenas, será lo contrario, debido a que la diferencia entre los datos reales y los predichos, es baja. En el entrenamiento de la red neuronal, se busca optimizar los parámetros mencionados, en relación con la pérdida obtenida en cada paso hacia adelante. En cada iteración de entrenamiento, se espera que la pérdida se reduzca, indicando que la red neuronal está aprendiendo los patrones en los datos de entrada. [34]

Generalmente, la función de pérdida computa la diferencia entre los valores predichos por la red y los valores reales que se desea obtener, para cada observación. El valor obtenido es conocido como error de predicción y el objetivo del modelo es minimizarlo.

Existen diferentes tipos de funciones de pérdida que aplican para distintos casos.

- ***Mean Squared Error*** **(MSE)**

  Esta función de pérdida cuantifica la magnitud del error a partir del promedio de las diferencias cuadradas entre los valores predichos y los reales. Al usar los valores al cuadrado, se busca penalizar fuertemente las desviaciones de los resultados erróneos. Se

usa principalmente en tareas de regresión, donde pueden existir muchos datos atípicos y se busca penalizarlos.

Se expresa de la siguiente manera,

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)}\|_2, \tag{13}$$

donde:

- N es la cantidad de ejemplos en el conjunto de datos de entrenamiento,
- $\mathbf{y}_i \in \mathbb{R}^d$ es el vector de valores predichos para la observación i,
- $\hat{\mathbf{y}}_i \in \mathbb{R}^d$ es el vector de valores reales para la observación i,
- d es la dimensión de los vectores $\mathbf{y}_i$ y $\hat{\mathbf{y}}_i$, es decir, la cantidad de salidas de la red neuronal,
- $\|\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)}\|_2$ denota la norma ℓ_2, es decir, la suma de los cuadrados de las diferencias entre cada componente del vector predicho con el real.

- ***Mean Absolute Error* (MAE)**

  Esta función de pérdida calcula el promedio del valor absoluto de las diferencias entre las predicciones y los valores reales. A diferencia de MSE, trata todos los errores por igual sin importar su magnitud. Se usa principalmente para tareas de regresión, donde no se desea penalizar con tanta fuerza los datos atípicos.

  Se expresa de la siguiente manera,

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)}\|_1, \tag{14}$$

  donde:

  - N es la cantidad de ejemplos en el conjunto de datos de entrenamiento,
  - $\mathbf{y}_i \in \mathbb{R}^d$ es el vector de valores predichos para la observación i,
  - $\hat{\mathbf{y}}_i \in \mathbb{R}^d$ es el vector de valores reales para la observación i,
  - d es la dimensión de los vectores $\mathbf{y}_i$ y $\hat{\mathbf{y}}_i$, es decir, la cantidad de salidas de la red neuronal,
  - $\|\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)}\|_1$ denota la norma ℓ_1, es decir, la suma de los valores absolutos de las diferencias entre cada componente del vector predicho con el real.

- ***Binary Cross-Entropy Loss***

  A diferencia de las anteriores, esta función de pérdida se utiliza para problemas de clasificación binaria. Se asume que el modelo genera una predicción de la probabilidad de que una observación pertenezca o no a una de las clases. A partir de esto, se cuantifica la diferencia entre las predicciones generadas por el modelo y las clases reales.

  Se expresa de la siguiente forma,

$$\text{BCE} = -\frac{1}{N}\left(\mathbf{y}^T \log(\hat{\mathbf{y}}) + (1-\mathbf{y})^T \log(1-\hat{\mathbf{y}})\right), \tag{15}$$

donde:

- N es la cantidad de ejemplos en el conjunto de datos de entrenamiento,
- $\mathbf{y} = [y_1, \ldots, y_N]^T$ es el vector de etiquetas binarias reales para cada observación, donde $y_i \in \{0, 1\}$,
- $\hat{\mathbf{y}} = [\hat{y}_1, \ldots, \hat{y}_N]^T$ es el vector de probabilidades predichas por el modelo para la clase 1 en cada observación, donde $\hat{y}_i \in [0, 1]$.

Estas funciones de pérdida son comúnmente utilizadas en los problemas de ML. Sin embargo, es posible personalizar una función de pérdida en base a los objetivos del aprendizaje, como se verá más adelante. Es posible crear funciones más complejas, basándose en el uso de funciones más simples como las presentadas en esta sección.

4.4.3.2. Descenso de gradiente

Ahora que ya se tiene como objetivo minimizar el error calculado, respecto los parámetros θ de la red neuronal, usando la función de pérdida, es necesario un método para realizar la optimización. La forma más común de hacer esta optimización en redes neuronales es por medio del algoritmo de descenso de gradiente, que se observa en el Algoritmo 1. Este algoritmo busca encontrar el valor mínimo de la función por medio del ajuste de θ. [34]

Como ya se mencionó anteriormente, un gradiente es una forma de medir qué tan inclinada es una línea o una curva en cierto punto y la dirección de esta inclinación. Este gradiente es aplicado a la función de pérdida que permite guiar al algoritmo de forma iterativa para encontrar el mínimo de la función de pérdida, por medio de la actualización de θ. En el algoritmo se busca encontrar puntos mínimos globales de la función, aunque en ocasiones es suficiente con encontrar mínimos locales [34]. Un punto mínimo global es aquel donde la función tiene su menor valor en todo el dominio [38]. Un punto mínimo local es aquel en donde la función tiene un valor menor en comparación con otros en su vecindad [38].

Para ejecutar el algoritmo es necesario definir la cantidad de iteraciones que se realizarán de este. Además, también es necesario definir una tasa de aprendizaje o *learning rate*, lo cual indica qué porcentaje se usará del gradiente computado para ajustar los pesos de la red neuronal. Una mayor tasa de aprendizaje puede acelerar la convergencia y aprendizaje, pero puede causar un estancamiento en mínimos locales. Una menor tasa de aprendizaje puede ser más lenta, pero puede llevar a encontrar el mínimo global. [34]

Algoritmo 1 Descenso de gradiente para redes neuronales

1: Inicializar los parámetros θ de la red neuronal (incluyendo pesos y sesgos) con valores aleatorios.
2: **Repetir**
3: Calcular el gradiente de la función de pérdida $L(\theta)$ con respecto a θ, utilizando *backpropagation*.
4: Actualizar los parámetros θ usando la tasa de aprendizaje η:

$$\theta = \theta - \eta \nabla_\theta L(\theta)$$

5: **Hasta que** El número de iteraciones se cumple o el valor de la pérdida $L(\theta)$ esté por debajo de un umbral ϵ.

4.4.3.3. *Backpropagation*

El gradiente de la pérdida, que se observa en el paso 4 del Algoritmo 1, contiene las derivadas de la función de pérdida con relación a cada uno de los pesos y sesgos de la red neuronal. Para un modelo con muchos nodos, esto puede volverse computacionalmente pesado. Por lo que es necesario un algoritmo eficiente que pueda calcular las derivadas. Este algoritmo es conocido como *backpropagation* (retropropagación en español). Cada una de estas derivadas puede servir para entender qué tanta responsabilidad tiene cada uno de los parámetros en la pérdida o error de la red neuronal, en la iteración actual. [39]

Para entender el funcionamiento de *backpropagation*, es necesario primero comprender la regla de la cadena en cálculo diferencial. Esta regla se aplica cuando se tiene una composición de funciones, es decir, el resultado de una función se utiliza como entrada para otra función. Esto es precisamente lo que ocurre en el funcionamiento de una red neuronal, donde la salida de cada capa se convierte en la entrada para la siguiente. [34]

La regla de la cadena expresa la derivada de una composición de funciones de la siguiente manera,

$$\frac{da}{db} = \frac{da}{dc} \cdot \frac{dc}{db}. \tag{16}$$

En este caso, la función a depende de c, que a su vez depende de b. Esta dependencia se puede expresar como $a(c(b))$. La Ecuación 16 significa que la derivada de a con respecto a b se calcula multiplicando la derivada de a con respecto a c por la derivada de c con respecto a v. De manera similar, en una red neuronal, las funciones que componen el modelo están encadenadas, y para calcular la derivada de la función de pérdida con respecto a los parámetros, se deben multiplicar todas las derivadas intermedias.

Ahora, es necesario calcular las derivadas parciales de la pérdida L, con respecto a los parámetros W, que son los pesos, y b, que son los sesgos. Por simplicidad, se explicará el funcionamiento del algoritmo para la capa de salida k que se observa en la Figura 13, y luego se extenderá a las demás capas.

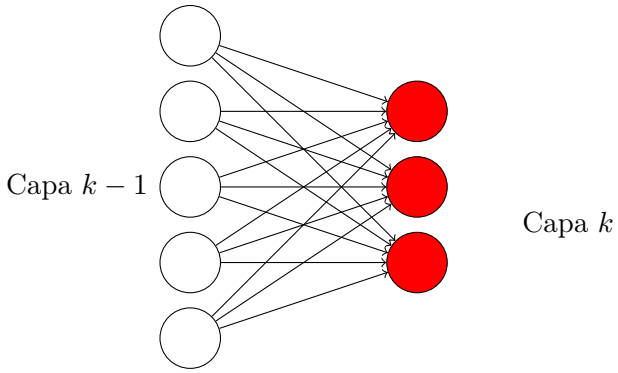


Figura 13: Capa de salida de red neuronal (Capa k) conectada a una capa anterior $k-1$.

El objetivo es realizar el cálculo de las derivadas de la pérdida con respecto a los parámetros de la última capa k,

$$\frac{\partial L}{\partial b_k}, \quad \frac{\partial L}{\partial W_k}, \tag{17}$$

donde:

- b_k es el vector de sesgos de la capa k,
- W_k es la matriz de pesos de la capa k.

Para calcular estas derivadas, es necesario seguir el flujo de la información desde los parámetros hasta el coste final, lo que implica una composición de funciones a lo largo de las capas de la red. Esta composición se da de la siguiente forma,

$$\begin{gathered} z_k = W_k a_{k-1} + b_k, \\ a_k = \sigma(z_k), \\ \text{Error de la red neuronal} = L\left(a_k\right). \end{gathered}$$

Donde z_k es el vector resultante de la suma ponderada de las activaciones de la capa anterior; a_k el resultado de aplicar la función de activación σ a la suma ponderada; el error de la red neuronal es el resultado de aplicar la función de pérdida a a_k. Este es el funcionamiento del paso hacia adelante en una red neuronal. Primero se computa la suma ponderada de las entradas de la red, luego se aplica una función de activación y finalmente en la capa de salida, se evalúa la función de pérdida.

Ahora, para encontrar las derivadas respecto a los parámetros, se puede aplicar la regla de la cadena de la siguiente forma,

$$\frac{\partial L}{\partial b_k} = \frac{\partial L}{\partial a_k} \cdot \frac{\partial a_k}{\partial z_k} \cdot \frac{\partial z_k}{\partial b_k}, \tag{18}$$

$$\frac{\partial L}{\partial W_k} = \frac{\partial L}{\partial a_k} \cdot \frac{\partial a_k}{\partial z_k} \cdot \frac{\partial z_k}{\partial W_k}. \tag{19}$$

El cálculo de las derivadas intermedias es relativamente simple. La primera $\frac{\partial L}{\partial a_k}$, significa cómo varía el error de la red con respecto a la salida. Se asume que se está usando MSE en la red neuronal para realizar el cálculo del error. La pérdida expresada en términos de a_k, que son las predicciones de la red neuronal, expresada en la Ecuación 13 como $\hat{\mathbf{y}}$, es de la siguiente forma,

$$L(a_k) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - a_k^{(i)})^2.$$

A pesar de que el siguiente análisis puede ser extendido para una cantidad arbitraria de ejemplos, el siguiente análisis se realizará con un solo ejemplo de aprendizaje (i.e. $N = 1$), por simplicidad. Por lo tanto, MSE para un solo ejemplo es de la siguiente manera,

$$L(a_k) = (y - a_k)^2.$$

Es posible encontrar la derivada de L respecto a a_k de la siguiente forma,

$$\frac{\partial L}{\partial a_k} = -2(y - a_k).$$

También para entender como varía la función de activación a_k en relación con la suma ponderada z_k, suponiendo que la función de activación es la sigmoide,

$$a_k(z_k) = \frac{1}{1 + e^{-z_k}}.$$

Por lo que,

$$\frac{\partial a_k}{\partial z_k} = a_k(z_k) \cdot (1 - a_k(z_k)).$$

Finalmente,

$$\frac{\partial z_k}{\partial b_k} = 1, \quad \frac{\partial z_k}{\partial W_k} = a_{k-1}.$$

Luego es posible multiplicar cada una de las derivadas obtenidas para la regla de la cadena establecida anteriormente en la Ecuación 18 y la Ecuación 19. De estas mismas ecuaciones, el término

$$\frac{\partial L}{\partial a_k} \cdot \frac{\partial a_k}{\partial z_k},$$

se puede decir que es $\frac{\partial L}{\partial z_k}$, por la regla de la cadena. Lo cual mide en que grado se modifica el error de la red neuronal, en base al cambio en la suma ponderada de las neuronas. Esto se conoce como error imputado de la neurona y se representa de la siguiente forma,

$$\delta_k = \frac{\partial L}{\partial a_k} \cdot \frac{\partial a_k}{\partial z_k}.$$

Así que la Ecuación 18 y la Ecuación 19 se simplifican a,

$$\frac{\partial L}{\partial b_k} = \delta_k, \quad \frac{\partial L}{\partial w_k} = \delta_k \cdot a_{k-1}.$$

El cálculo de las derivadas de los parámetros del resto de las capas se simplifica. Siguen siendo composiciones que involucran más funciones. Para la capa oculta, previa a la de salida, de nuevo se necesitan las derivadas por la regla de la cadena,

$$\frac{\partial L}{\partial b_{k-1}} = \frac{\partial L}{\partial a_k} \cdot \frac{\partial a_k}{\partial z_k} \cdot \frac{\partial z_k}{\partial a_{k-1}} \cdot \frac{\partial a_{k-1}}{\partial z_{k-1}} \cdot \frac{\partial z_{k-1}}{\partial b_{k-1}},$$

$$\frac{\partial L}{\partial W_{k-1}} = \frac{\partial L}{\partial a_k} \cdot \frac{\partial a_k}{\partial z_k} \cdot \frac{\partial z_k}{\partial a_{k-1}} \cdot \frac{\partial a_{k-1}}{\partial z_{k-1}} \cdot \frac{\partial z_{k-1}}{\partial W_{k-1}}.$$

Sin embargo, es posible simplificar estas expresiones utilizando valores computados previamente y derivadas conocidas,

$$\frac{\partial L}{\partial b_{k-1}} = \delta_k \cdot \frac{\partial z_k}{\partial a_{k-1}} \cdot \frac{\partial a_{k-1}}{\partial z_{k-1}},$$

$$\frac{\partial L}{\partial W_{k-1}} = \delta_k \cdot \frac{\partial z_k}{\partial a_{k-1}} \cdot \frac{\partial a_{k-1}}{\partial z_{k-1}} \cdot a_{k-2}.$$

Además, se puede obtener información útil de la siguiente relación,

$$\delta_{k-1} = \delta_k \cdot \frac{\partial z_k}{\partial a_{k-1}} \cdot \frac{\partial a_{k-1}}{\partial z_{k-1}},$$

donde δ_{k-1} representa el gradiente propagado hacia atrás para la capa $k-1$.

Luego, las derivadas finales respecto a los parámetros b_{k-1} y W_{k-1} se simplifican de la siguiente manera,

$$\frac{\partial L}{\partial b_{k-1}} = \delta_{k-1}, \quad \frac{\partial L}{\partial W_{k-1}} = \delta_{k-1} \cdot a_{k-2}.$$

De esta forma, será posible encontrar las derivadas de los parámetros de la red, usando información de las derivadas de las capas siguientes, para las cuales ya se computaron las respectivas derivadas. Este algoritmo es relativamente sencillo de implementar, además de ser bastante eficiente para la computación de las derivadas. Es generalizable para distintos tipos de redes neuronales y potencia el aprendizaje de estas con respecto a los datos. Es bastante escalable en relación con la cantidad de parámetros de la red neuronal. [39]

4.4.3.4. Optimizadores

Los optimizadores son métodos o algoritmos que se usan para minimizar el error que se obtiene a través de la función de pérdida. Ya se describió el algoritmo de descenso de gradiente, el cuál es un optimizador. Sin embargo, existen algunos otros que son más sofisticados y pueden ayudar a la red neuronal a aprender más rápidamente. A continuación, se detallan brevemente los dos optimizadores que se utilizan en este trabajo.

- **ADAM**
  Adaptive Moment Estimation (ADAM) es un algoritmo de optimización que ajusta la tasa de aprendizaje para cada parámetro de la red neuronal, permitiendo una actualización más precisa y eficiente de los pesos y sesgos. Permite manejar eficientemente grandes volúmenes de datos y modelos con muchos parámetros. [40]

  Usa un mecanismo conocido como *Momentum*, el cual implica mantener un promedio móvil exponencialmente ponderado de los gradientes computados anteriormente. Involucra este valor en la actualización de los pesos, ayudando a que la convergencia sea más rápida y estable.

  Es útil en escenarios complejos, donde existen muchos datos que potencialmente puedan tener ruido. Es usado ampliamente en tareas de *Deep Learning*, visión por computadora y modelos de procesamiento de lenguaje natural.

- **L-BFGS**
  Los métodos como descenso de gradiente hacen uso de derivadas parciales de primer orden para encontrar el mínimo de la función. Sin embargo, existen otros métodos como el método de Newton que hacen uso de derivadas parciales de segundo orden. Estos métodos pueden ser más efectivos para encontrar el mínimo de la función, debido a que no solo dan la dirección en la que debe moverse en la función, si no que también el tamaño del paso que se debe tomar. [41]

  El método de Newton necesita calcular la matriz Hessiana, la cuál contiene todas las segundas derivadas de una función con respecto a las variables independientes. También para su funcionamiento necesita calcular su inversa, lo cual puede ser altamente costoso en términos computacionales. En BFGS, que se basa en el método de Newton, la Hessiana puede incrementar su tamaño en base a la cantidad de variables independientes del problema. Por lo que actualizarla puede ser muy ineficiente.

Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) se clasifica como Quasi-Newtoniano, puesto que aproxima la inversa de la Hessiana por medio de un método de gradiente, lo cuál es menos costoso. Además, esto también soluciona el problema del incremento de tamaño de la Hessiana.

Entre sus beneficios es que se puede tener una convergencia en el entrenamiento rápida, debido a la información que incluye de la función. Sin embargo, puede ser más intensa computacionalmente que un método basado en descenso de gradiente sencillo.

4.4.4. Diferenciación automática

Como se pudo observar en la sección anterior, la computación automática de las derivadas es de vital importancia para el entrenamiento de las redes neuronales. En muchas ocasiones, al introducir nuevos modelos de ML que requieren el cálculo de derivadas, esto se ha realizado de forma manual. Sin embargo, este proceso es costoso y complejo. También se ha buscado hacer empleando métodos numéricos, pero ha generado errores de aproximación. Finalmente se ha buscado realizar con diferenciación simbólica, pero esta ha sido demasiado engorrosa. La diferenciación automática (AD, por sus siglas en inglés) se ha vuelto muy popular, porque permite propagar las derivadas a través de los cálculos necesarios en un modelo de *machine learning*, usando la regla de la cadena. De hecho, el algoritmo de descenso de gradiente que se explicó anteriormente hace uso de la diferenciación automática. [42]

Como se mencionó, distintos métodos de diferenciación han fallado al momento de computar gradientes en modelos de ML. A continuación, se da una breve explicación de los métodos de diferenciación numérica y simbólica y la razón de sus fallas:

- **Diferenciación numérica**

  Esta es una técnica para computar los gradientes para optimizar una red neuronal, usando el límite de la definición de una derivada [38],

  $$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}. \tag{20}$$

  Para computar un gradiente de una función, se puede usar de la siguiente forma,

  $$\frac{\partial f}{\partial x_i} = \frac{f(x + he_i) - f(x)}{h} \quad \text{para } i = 1, 2, \ldots, d, \tag{21}$$

  donde se computa la derivada de la función objetivo, con respecto a todas las variables independientes de interés. Acá, e_i es un vector unitario para la dimensión i, y h es un escalar que representa un cambio pequeño, acorde a la definición de la derivada [42].

  Su implementación no es difícil, pero computar el gradiente se convierte en un algoritmo de complejidad $O(d)$. Considerando que en muchas ocasiones las entradas de una red neuronal son grandes, el tiempo para computar los gradientes aumenta considerablemente. Además, la necesidad de un valor de h pequeño puede causar inestabilidad. Finalmente, la representación numérica por parte de las computadoras es limitada. Generalmente para este proceso se usan valores de punto flotante, los cuales tienen cierto

límite de representación y esto puede generar imprecisiones en los cálculos, además de requerir mucha memoria. De esta forma se reduce la escalabilidad del algoritmo y se pueden generar errores en la optimización de la red neuronal. [43]

- **Diferenciación simbólica**

  La diferenciación simbólica es un proceso sistemático por el cual se transforma una expresión que la componen símbolos y operadores aritméticos, en otra expresión que representa su derivada. Esto es similar al proceso manual, pero se usa un algoritmo para automatizarlo. Acá no se obtienen las imprecisiones de diferenciación numérica, porque se obtiene una expresión que es exactamente la derivada de la función. Sin embargo, el mayor problema proviene de un crecimiento de las expresiones simbólicas, que se puede salir de control. Considerando que, como se mencionó, las redes neuronales usan composición de funciones, hacer derivadas de estas puede resultar en un proceso impráctico. También puede ser computacionalmente costoso evaluar las derivadas simbólicas, debido a que estas pueden ser expresiones inmanejables. [43]

La diferenciación automática, permite expresar las composiciones de funciones con las variables que las componen y operaciones elementales, que son suma, resta, substracción, división y multiplicación. Toda la computación numérica se desarrolla alrededor de estas operaciones, concatenando los valores de derivadas conocidas de las cuales depende el cálculo de una nueva derivada. La AD tiene dos formas principales, el modo hacia adelante y modo reverso. [42]

- **Modo hacia adelante**

  Esta técnica evalúa simultáneamente el valor de la función y sus derivadas, utilizando pares de valores conocidos como *dual numbers*. Estos valores son propagados hacia adelante, en cada operación aritmética de la cadena, hasta la salida de la función. Es útil cuando se desea una precisión exacta de las derivadas y es eficiente para funciones con pocas entradas.

- **Modo reverso**

  Todo el proceso de AD parece muy conocido a una sección previa de este trabajo y es que como se mencionó, el descenso de gradiente hace uso de él. La forma en la que lo usa es con *backpropagation*, método que se explicó anteriormente. El modo reverso de la diferenciación automática esencialmente es *backpropagation*. Es el método más utilizado para el cálculo de derivadas en ML.

4.5. PINNs

Como se mencionó anteriormente, el entrenamiento de un modelo de ML generalmente consta de un conjunto de características de datos que son ingresadas como entradas. A partir de estas se genera una salida, basada en un algoritmo que ha aprendido un patrón en los datos. Particularmente en sistemas físicos o ingenieriles, este acercamiento puede ser complejo, debido a que la adquisición de datos relacionados al problema puede ser muy difícil. Debido a esta falta de datos, muchos algoritmos de ML y particularmente de *deep*

learning, fallan al momento de hacer predicciones acertadas. Sin embargo, hay un factor extra que no siempre se toma en cuenta y puede ser de gran ayuda en la resolución de estos problemas, el conocimiento teórico. [44]

Existe una buena cantidad de conocimiento teórico de diversos sistemas. Este conocimiento muchas veces se da en forma de leyes o principios físicos, que ayudan a describir los problemas, los cuales pueden expresarse como PDEs. Sin embargo, estas no suelen ser incorporadas en los métodos de aprendizaje tradicionales. A pesar de esto, integrarlas puede traer un gran beneficio, debido a que pueden actuar como un regularizador. Esto puede ayudar a forzar a que el modelo aprenda la función o funciones que modelan el comportamiento del sistema y que genere soluciones correctas al problema. Al integrar esta información en un modelo, se puede aumentar la información que este tiene para generar estas soluciones. [6]

A partir de esto, surgen las Redes Neuronales Informadas por la Física (PINNs por sus siglas en inglés). Este modelo permite integrar el conocimiento teórico existente, incorporándolo en su arquitectura y proceso de aprendizaje. Combinan la potencia de las redes neuronales con la precisión de las ecuaciones físicas. Esto permite modelar problemas que no son triviales y brindan soluciones para escenarios de los cuales previamente no se tenía información, por su capacidad de generalización. Se basan principalmente en el teorema de aproximación universal de las redes neuronales. Este y las características básicas del modelo se describirán a continuación. [6]

4.5.1. Teorema de aproximación universal

El teorema de aproximación universal es una propiedad de las redes neuronales feedforward, sobre la cual se basa el uso de PINNs. Este establece que una red neuronal feedforward que tenga una capa de salida lineal y al menos una capa oculta con cualquier función de activación *squashing*, como la sigmoide o tanh, puede aproximar cualquier función continua en un subconjunto de $\mathbb{R}^n$, con un factor de error arbitrario, dado que tenga suficientes unidades o nodos ocultos. Las derivadas de la red neuronal también pueden aproximar las derivadas de la función con precisión arbitraria.[34]

Este teorema quiere decir que, sin importar qué función se busque aproximar o aprender, una red neuronal feedforward lo suficientemente grande podrá hacerlo. Al decir *suficientemente grande*, se refiere a tener una cantidad suficiente de capas ocultas y de nodos en estas. Es importante notar que el teorema indica que existe una red neuronal suficientemente grande, pero no indica el tamaño. Podría requerirse una red de tamaño exponencial o prohibitivamente grande, que haga imposible el aprendizaje y su implementación.

El teorema es de gran importancia, debido a que, al poder aproximar cualquier función, la red neuronal puede aprender las soluciones de cualquier PDE o conjunto de estas, con alta precisión. Por lo tanto, esto proporciona una base teórica para poder usar redes neuronales en la resolución de problemas físicos.

4.5.2. Implementación

Considerando una ecuación diferencial parcial parametrizada para una función $u = u(t, x)$, donde

$$u : \mathbb{R}^{d+1} \to \mathbb{R},$$

donde d se refiere a la cantidad de dimensiones espaciales y se le suma la dimensión temporal. Esta función tiene un codominio escalar. Además, $t \in [0, T]$ y $x \in \Omega \subset \mathbb{R}^d$.

La ecuación diferencial parcial toma la forma,

$$\frac{\partial u}{\partial t} + \mathcal{N}[u] = 0, \quad t \in [0, T], \quad x \in \Omega, \tag{22}$$

donde $\mathcal{N}[u]$ es un operador diferencial en u con respecto a las variables espaciales. Esto quiere decir, posiblemente un operador de divergencia, gradiente, laplaciano, entre otros.

La ecuación está sujeta a las condiciones iniciales y de frontera,

$$u(0, x) = g(x), \quad x \in \Omega, \tag{23}$$
$$B[u] = 0, \quad t \in [0, T], \quad x \in \partial\Omega. \tag{24}$$

donde $B[u]$ es un operador de frontera que corresponde a condiciones de frontera de Dirichlet, Neumann o Robin.

La función u es la solución de la PDE. En este caso, u es dependiente del tiempo y de una variable de posición x. Sin embargo, u puede depender de las variables necesarias del problema. [6]

Para aproximar la solución de la función descrita por medio de una PINN, se debe construir una red neuronal feedforward $\hat{u}_\theta(t; x)$ con parámetros θ, que son los pesos y sesgos. La salida de esta red neuronal representa la solución de la PDE de la Ecuación 22. Para poder encontrar la solución por medio de la red neuronal, se necesitan distintos elementos. [45]

El primer y más importante elemento necesario para entrenar la red neuronal, es la definición de una función de pérdida personalizada. Esta permitirá integrar información del sistema. Esto lo hará por medio de términos en la pérdida que involucran las condiciones iniciales, las condiciones de frontera y la PDE.

El primer término es una función que computa la pérdida para los puntos de condiciones iniciales. En esta pérdida se emplean puntos definidos en el subdominio del problema, donde se desean establecer valores iniciales, los cuáles se evalúan la red neuronal. Se busca hacer que, en el tiempo inicial, la red neuronal cumpla con los valores preestablecidos para estos puntos.

Se usa una función de pérdida existente como MSE, para cuantificar la diferencia entre las predicciones de la red neuronal, y los valores reales de la función objetivo en las condiciones iniciales, de la siguiente forma,

$$L_{IC}(\theta) = \frac{1}{N_{IC}} \sum_{i=1}^{N_{IC}} \|u_\theta(0, \mathbf{x}_{IC}^{(i)}) - \mathbf{g}(\mathbf{x}_{IC}^{(i)})\|^2. \tag{25}$$

Acá N_{IC} se refiere a la cantidad de puntos evaluados, $x_{IC}^{(i)}$ a cada punto evaluado y g es una función que da el valor inicial a cada punto. Estos puntos evaluados en las condiciones iniciales son muestreados del dominio espacial donde se impone la condición, como se mencionó. Comúnmente se usa un muestreo uniforme de los puntos y se busca tener una cantidad representativa de estos, para que la red neuronal pueda aprender a satisfacerlos adecuadamente.

Para el segundo término se busca que la red neuronal cumpla con las condiciones de frontera. Para esto se evalúan los puntos en los subdominios de frontera en la red neuronal y se busca que esta cumpla con los valores preestablecidos.

Se usa una función de pérdida preexistente como MSE, de nuevo,

$$L_{BC}(\theta) = \frac{1}{N_{BC}} \sum_{i=1}^{N_{BC}} \|B[u_\theta](t_{BC}^i, x_{BC}^i)\|^2. \tag{26}$$

Donde N_{BC} es la cantidad de puntos evaluados de frontera, $t_{BC}^{(i)}$ los puntos temporales en el contorno y $x_{BC}^{(i)}$ los puntos espaciales en el contorno. Al igual que con las condiciones iniciales, se busca tener una muestra representativa de los puntos en el subdominio de frontera correspondiente.

Hasta el momento, se han definido dos términos en la función de pérdida, los cuales corresponden a un enfoque de aprendizaje supervisado convencional. En este caso, se tienen las características de las observaciones, que son las coordenadas espacio-temporales de los puntos evaluados en las condiciones iniciales y de frontera, junto con los valores reales (*ground truth*) asociados a cada uno de estos puntos. Estos valores permiten calcular el error entre la salida de la red neuronal y los datos esperados, lo que permite ajustar los parámetros de la red para que esta aprenda a satisfacer las condiciones impuestas de manera adecuada. Sin embargo, a continuación se definirá un término de la pérdida que corresponde a un aprendizaje no supervisado, puesto que no se tienen valores reales contra los cuales se puedan comparar los resultados de la red neuronal. Por esto, las PINNs son conocidas como un método de aprendizaje semi-supervisado.

Para el tercer término involucrado, es necesario definir los residuales que son funciones que miden el error en la solución aproximada de la PDE, por medio de la red neuronal. Estos ayudan a que la red cumpla con la restricción dada por la PDE, asegurando que esté modelando adecuadamente la función solución.

El residual de la PDE de la Ecuación 22 se puede definir de la siguiente forma,

$$R_\theta(t,x) = \frac{\partial u_\theta}{\partial t}(t,x) + N[u_\theta](t,x). \tag{27}$$

Se busca que este residual sea 0 en todo el dominio interior de la función, es decir, donde no hay condiciones de contorno. Para esto, se hace un muestreo, posiblemente aleatorio, de puntos en todo el dominio, exceptuando las condiciones de frontera. Estos puntos se conocen como puntos de colocación y sirven para ser evaluados en la red neuronal y sus derivadas. En este caso, no hay un valor con el cual comparar los puntos, simplemente se busca reducir a 0 el residual. El cálculo de estas derivadas se hace por medio de diferenciación automática. En esencia, esto lo que significa es integrar la PDE en la red neuronal de una forma explícita. De forma que, se puede escribir la forma de la PDE, con sus respectivas derivadas en la función de pérdida y como se mencionó, se toma a la red neuronal como la función solución.

La forma de esta función de pérdida, una vez más usando MSE es,

$$L_{\text{Física}}(\theta) = \frac{1}{N_r} \sum_{i=1}^{N_r} \|R_\theta(\mathbf{t}_r^{(i)}, \mathbf{x}_r^{(i)})\|^2 \tag{28}$$

Donde N_r es la cantidad de puntos de colocación, $t_r^{(i)}$ son los puntos de colocación temporales y $x_r^{(i)}$ son los puntos de colocación espaciales. En el presente trabajo se usa el término $L_{\text{Física}}$ debido a que esta es la parte donde se integra la mayor cantidad de información del funcionamiento del sistema físico que se modela.

Finalmente, se define una función de pérdida conjunta basada en todos los términos anteriores:

$$L(\theta) = L_{IC}(\theta) + L_{BC}(\theta) + L_{\text{Física}}(\theta) \tag{29}$$

Esta función de pérdida es usada para entrenar la red neuronal, buscando cumplir no solo con las condiciones de frontera e iniciales, sino que también con la forma de la PDE. Debido a que para los puntos de frontera e iniciales existen valores objetivo que se busca cumplir, pero para los puntos de colocación no, las PINN se consideran un tipo de aprendizaje semi-supervisado. [6]

En la Figura 14, se puede observar la arquitectura de una PINN. El primer componente que tiene es una red neuronal feedforward, que busca aproximar una función definida. A partir del resultado de esta red neuronal, se computan las derivadas necesarias expresadas en la PDE. Usando estas, se calcula la pérdida del residual, de las condiciones iniciales y de las condiciones de frontera y se integra. Usando esta pérdida, se realiza todo el proceso de optimización de la red, con los algoritmos mencionados en secciones anteriores. Esto se realiza hasta llegar a una solución suficientemente buena. Generalmente, las PINNs usan tanh de función de activación, debido a que es posible computar todas sus derivadas durante el *backpropagation*, a diferencia de funciones como ReLU. [44]

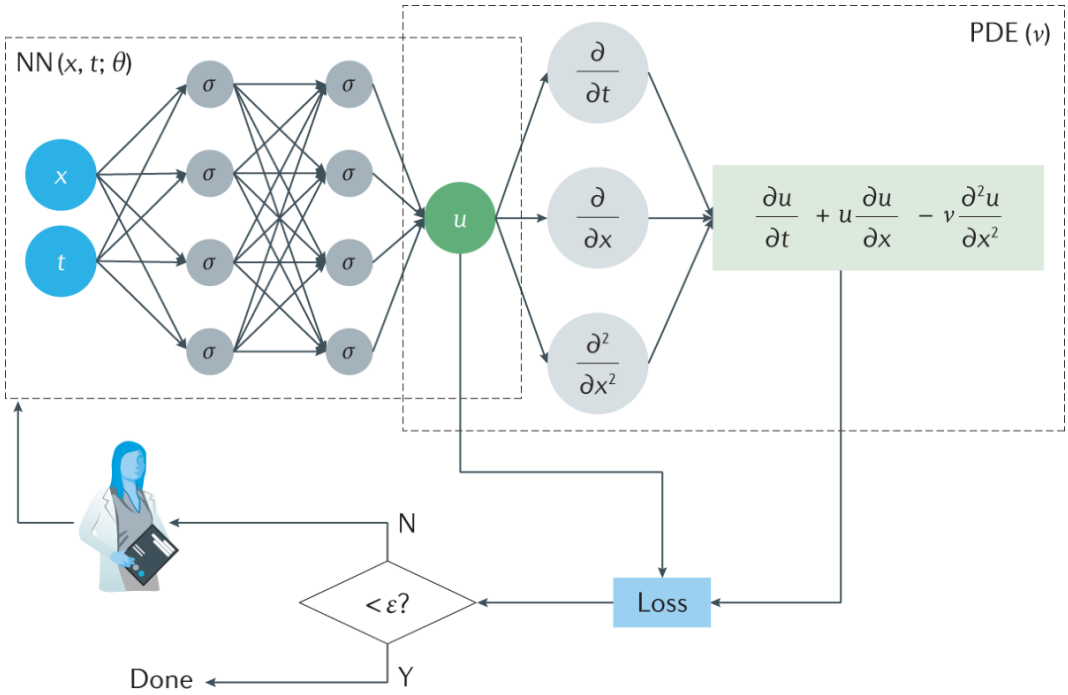


Figura 14: Diagrama de arquitectura de una PINN [44].

CAPÍTULO 5

PINN para modelar piezoelectricidad

El presente capítulo describe la metodología utilizada para modelar el efecto piezoeléctrico utilizando PINNs.

5.1. Exploración del funcionamiento de PINNs

La primera parte para llevar a cabo el desarrollo del proyecto fue entender el funcionamiento de las PINNs. Para esto se realizaron dos problemas previamente solucionados. En el primero se modeló un problema descrito por la ecuación de calor, que se mostrará más adelante. El otro consistió en la modelación de un sistema de fluidos descrito por las ecuaciones de Navier-Stokes.

5.1.1. Ecuación de calor

Inicialmente se modeló la ecuación de calor, como se mencionó, para comprender el esquema general empleado para la resolución de PDEs usando redes neuronales. Se utilizó un problema relativamente simple, para poder obtener un conocimiento del funcionamiento de las PINNs de forma rápida [46].

5.1.1.1. Definición del problema

Se modeló la ecuación de calor en una dimensión como problema físico. Para este fin se usó una varilla de una dimensión con un largo establecido. La PDE de este sistema está dada por la Ecuación 30.

$$\frac{\partial U}{\partial t} = \alpha \frac{\partial^2 U}{\partial x^2}, \tag{30}$$

$$f(x) = 50x(1-x), \tag{31}$$

$$U(x,0) = f(x), \tag{32}$$

$$U(0,t) = U(L,t) = 0. \tag{33}$$

Se definió la condición inicial de la PDE por medio de la Ecuación 32. Esta quiere decir que la temperatura en cada punto de la varilla está inicialmente en una configuración que conforme a la función $f(x)$. En este caso, esta función está dada en la Ecuación 31.

Además también se definieron las condiciones de frontera en la Ecuación 33. Para estas, L es el largo de la varilla. Esto quiere decir, que, en los extremos de la varilla, la temperatura es cero, por lo que no hay ninguna fuente de calor adicional. Las propiedades del material usado se encuentran en la Tabla 1.

Características del problema	**Valor**
Largo de varilla (L)	1.0 metro
Tiempo (T)	100.0 segundos
Constante de difusión de calor (α)	0.003 m^2/s

Tabla 1: Propiedades del material utilizado.

5.1.1.2. Tecnologías utilizadas y entorno de ejecución

El experimento se llevó a cabo en una computadora Alienware m15 R6 con las siguientes especificaciones:

- **Procesador:** 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz, 2304 MHz, 8 núcleos, 16 procesadores lógicos.
- **Memoria RAM:** 32 GB.

Para la implementación de la red neuronal y el análisis de datos, se utilizaron las siguientes librerías de Python:

- **PyTorch:** Es una de las bibliotecas más populares para el *deep learning*. Ofrece la capacidad de crear distintos tipos de arquitecturas de redes neuronales, por medio de métodos y clases. Facilita el entrenamiento de las redes neuronales, debido a que incluye optimizadores, la capacidad de definir funciones de pérdida flexibles y el ajuste automático de pesos con *backpropagation*. Además, provee la posibilidad de hacer diferenciación automática. Por lo tanto, en el proyecto fue usada para crear la red neuronal *feedforward* y para el cálculo de derivadas incluidas en las PDEs. [47]

- **Numpy:** Es una biblioteca orientada a la computación científica con Python. Proporciona soporte para arrays y matrices grandes y multidimensionales, junto con una colección de funciones matemáticas para operar en estos arrays. En el proyecto, fue utilizada para la generación de los datos para alimentar a la red neuronal. [48]
- **Matplotlib:** Es una biblioteca de visualización con Python. Se utiliza para crear figuras y gráficos. En el proyecto, Matplotlib se usó para visualizar los resultados del modelo. [49]
- **DeepXDE:** Es una biblioteca orientada a la facilitación de la creación y entrenamiento de PINNs. En este proyecto fue utilizada para hacer diferenciación automática para la ecuación de calor únicamente. [50]
- **PyDOE:** Es una biblioteca que permite realizar muestreo de datos, lo cual fue útil para el entrenamiento de las PINNs en este proyecto.

5.1.1.3. Generación de datos

Para las condiciones iniciales y de frontera se generaron 300 puntos espaciales, muestreados uniformemente, en el rango de 0 a 1.0 (perteneciendo a números reales) y lo mismo para los puntos temporales, pero en el rango de 0 a 100.0 segundos.

Luego se creó una *meshgrid* de estos, que es una estructura de datos que relaciona el espacio con el tiempo. Para la condición inicial, se usaron todos aquellos puntos donde el tiempo fuera 0 y se calculó el resultado de la función $f(x)$ mencionada, siendo este la temperatura correspondiente. Para las condiciones de frontera, se usaron todos los puntos que tuvieran tiempo 0 y punto espacial 0 o L y se les asignó como resultado, una temperatura de 0.

Finalmente, se generaron 300 puntos espaciales y 300 temporales, muestreados uniformemente en todo el dominio, haciendo igualmente una *meshgrid*, para un total de 9000 puntos. Estos fueron los puntos de colocación utilizados en el problema. Además, se generaron 500 puntos espaciales y 500 puntos temporales como conjunto de prueba, para evaluar las predicciones de la PINN.

5.1.1.4. Arquitectura de PINN

Para resolver la ecuación diferencial, se utilizó una PINN basada en una *Feed Forward Neural Network*. Los detalles de la arquitectura de una PINN utilizada para modela la ecuación de calor se pueden encontrar en la Tabla 2.

La arquitectura de forma gráfica se puede observar en la Figura 15. Se usó una capa de entrada de 2 unidades, debido a que las variables independientes del problema son posición en una dimensión y tiempo. Luego se usaron 2 capas ocultas de 20 nodos cada una y entre ellas se como función de activación tanh. Finalmente, en la capa de salida se tuvo un solo nodo, debido a que este modela la función de calor U, presentada en la Ecuación 30.

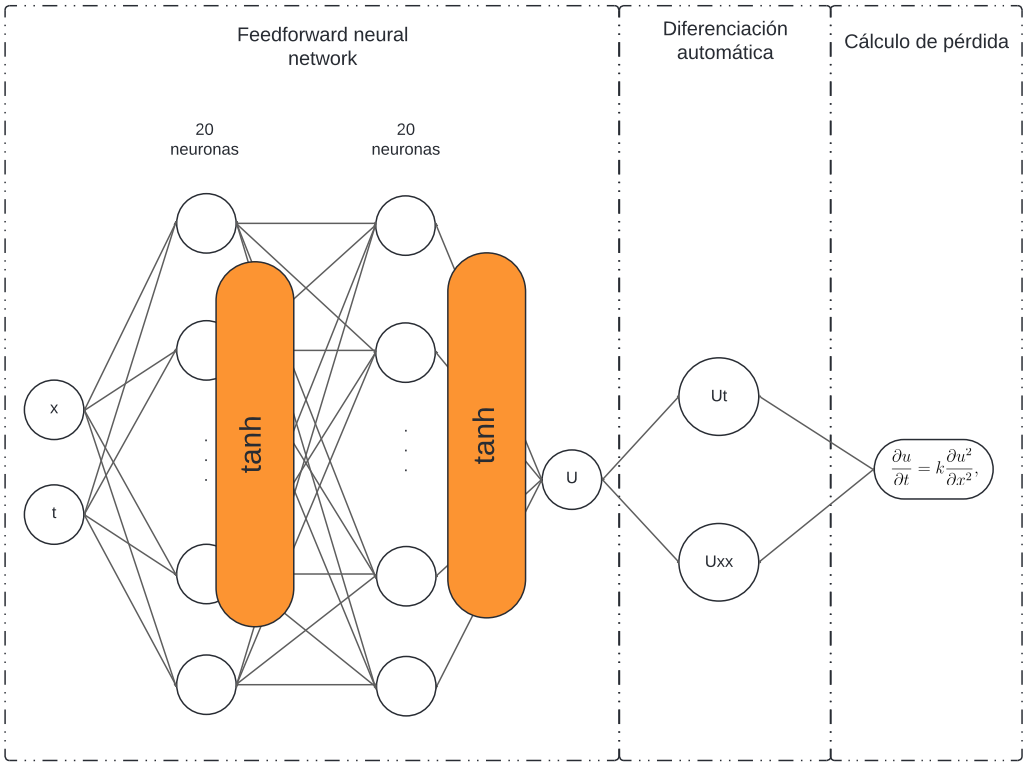


Figura 15: Arquitectura de PINN para ecuación de calor.

Número de capa	Nombre de capa	Descripción de capa
1	Input/Dense layer 1	Tamaño de capa de entrada de 2; 20 nodos en capa oculta.
2	tanh 1	Función de activación tanh.
3	Dense layer 2	20 nodos en capa oculta.
4	tanh 2	Función de activación tanh.
5	Output/Dense layer 4	20 nodos en capa oculta; tamaño de capa de salida de 1.

Tabla 2: Arquitectura de la red neuronal.

5.1.1.5. Función de pérdida

La función de pérdida se divide en dos partes. Por un lado, se tiene la pérdida física, que está basada en la PDE de la Ecuación 30 y que tiene la forma,

$$\mathcal{L}_{\text{física}} = \frac{1}{M}\sum_{i=1}^{M}\left\|\frac{\partial U_\theta(x^{(i)},t^{(i)})}{\partial t_i} - \alpha\frac{\partial^2 U_\theta(x^{(i)},t^{(i)})}{\partial x_i^2}\right\|^2, \tag{34}$$

donde M es la cantidad de puntos de colocación usados, y $U_\theta(x^{(i)},t^{(i)})$ es el resultado de la red neuronal. Entonces, se busca reducir a 0 la PDE, optimizando la red para que modele la función que se desea aproximar. El cálculo de las derivadas de la función $U_\theta(x^{(i)},t^{(i)})$, que es la red neuronal, se realiza por medio de autodiferenciación en reversa o *backpropagation*.

Luego, se utiliza la pérdida de la condición inicial y de las condiciones de frontera. En este caso, se usa una pérdida MSE, comparando los puntos predichos por la red neuronal y los resultados verdaderos, definidos en la creación de los datos,

$$\mathcal{L}_{\text{IC}} = \frac{1}{N_{\text{IC}}}\sum_{i=1}^{N_{\text{IC}}}\|U_\theta(x_i,t_i) - y_{\text{IC},i}\|^2 \tag{35}$$

$$\mathcal{L}_{\text{BC}} = \frac{1}{N_{\text{BC}}}\sum_{i=1}^{N_{\text{BC}}}\|U_\theta(x_i,t_i) - y_{\text{BC},i}\|^2 \tag{36}$$

En la Ecuación 32, $y_{\text{IC},i}$ se refiere al resultado esperado de la función para la condición inicial en los puntos de este subdominio. En la Ecuación 33, $y_{\text{BC},i}$ se refiere al resultado esperado de la función para las condiciones de frontera en este subdominio.

Finalmente, se utiliza una pérdida en conjunto,

$$\mathcal{L} = \mathcal{L}_{\text{física}} + \mathcal{L}_{\text{IC}} + \mathcal{L}_{\text{BC}}. \tag{37}$$

5.1.1.6. Entrenamiento

Para el entrenamiento de la PINN se usaron 5000 épocas, durante las cuales se alimentó el modelo con los puntos de colocación, de condiciones iniciales y de frontera. Se utilizó la función de pérdida establecida para hacer el backpropagation en la red, buscando satisfacer las condiciones y que el resiudal de la PDE convergiera hacia 0. Se utilizó ADAM como optimizador con una tasa de aprendizaje de 0.01, siendo este una opción eficiente para manejar la optimización de la PINN [45].

5.1.1.7. Resultados

En la Figura 16 se pueden observar los resultados de la modelación de la ecuación de calor utilizando PINNs. Se utilizó el método de las diferencias finitas (FDM por sus siglas en inglés), para hacer una modelación clásica del problema y poder realizar una comparación con lo obtenido de la red neuronal. Los resultados de este método se pueden observar en la Figura 17.

Se puede observar que la forma de la función obtenida de la PINN es bastante similar a la de la función obtenida por la FDM. Para sustentar esto, también se realizó el cálculo del error relativo medio L2, cuya fórmula se encuentra en la Ecuación 38, entre los resultados de la PINN y FDM usado como valor real. Se obtuvo un error relativo medio L2 de 0.609.

$$\text{Error Relativo Medio L2} = \frac{\|u_\theta - u_{\text{FDM}}\|_2}{\|u_{\text{FDM}}\|_2} \tag{38}$$

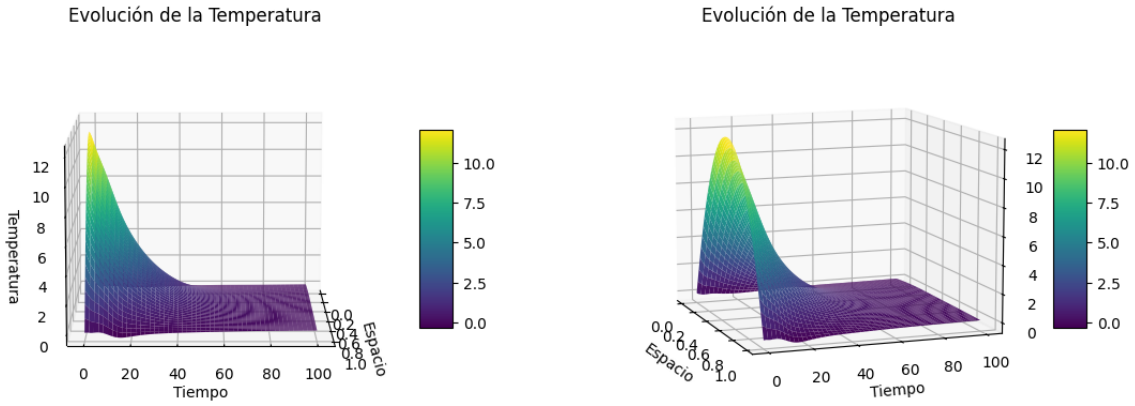


Figura 16: Resultado de PINN para ecuación de calor.

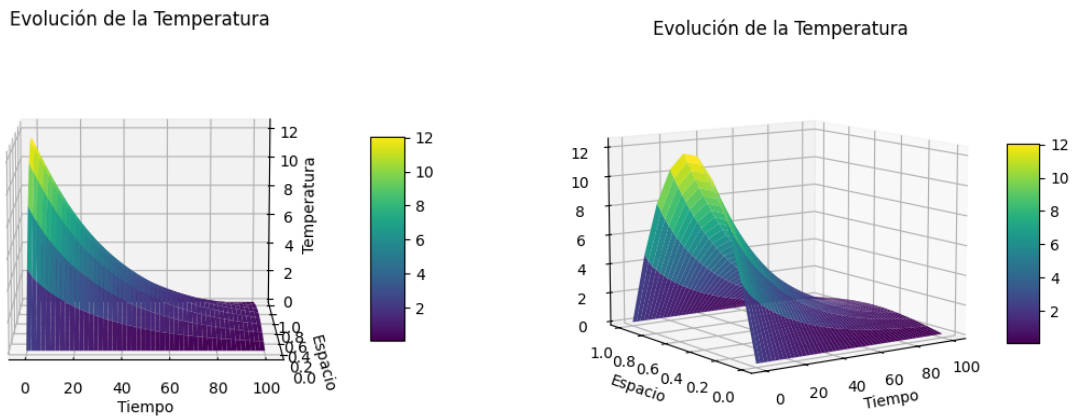


Figura 17: Resultado de FDM para ecuación de calor.

5.1.2. Navier-Stokes

Para continuar con el entendimiento de PINNs, se procedió a modelar un problema de mecánica de fluidos descrito por las ecuaciones de Navier-Stokes. Este escenario es más retador que el anterior, debido a que las ecuaciones son más complejas, además de que es un sistema de PDEs.

5.1.2.1. Definición del problema

El problema consiste en un fluido atravesando un canal que tiene un cilindro circular en medio, en dos dimensiones. Esta configuración se puede observar en la Figura 18. El canal tiene un alto de 0.41 unidades y un ancho de 2.2 unidades. El cilindro tiene un radio de 0.05 y su centro está en el punto (0.2, 0.2). El dominio del problema consiste en todas las coordenadas espaciales dentro del canal, exceptuando aquellas dentro del cilindro.

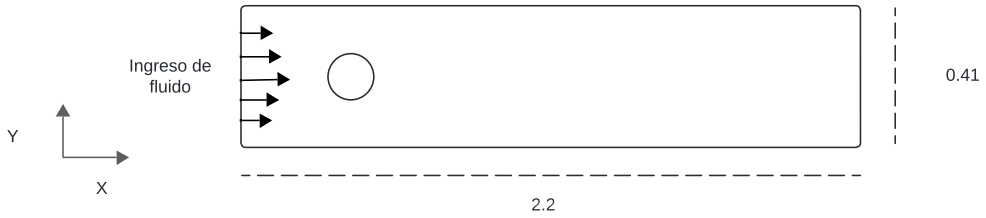


Figura 18: Configuración de problema de mecánica de fluidos.

Se modeló el estado estacionario, que está descrito por la Ecuación 39 y la Ecuación 40. Acá ν representa una viscosidad cinemática, propia del fluido, cuyo valor es 0.001. Por otra

parte, $\mathbf{u}$ es el vector de velocidad del fluido, en este caso en dos dimensiones, y p la presión.

$$\nabla \cdot \mathbf{u} = 0, \tag{39}$$

$$-\nu \Delta \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p = 0. \tag{40}$$

Se definieron distintas condiciones de frontera. En las paredes superior e inferior, así como en la circunferencia del cilindro, se usa una condición de frontera de no deslizamiento. Esto quiere decir que la velocidad $\mathbf{u}$, tanto en x como en y, debe ser 0 en estos subdominios. Además, en el extremo izquierdo se define un flujo del fluido, dado por la Ecuación 41.

$$\mathbf{u}(0, y) = \left(\frac{4Uy(0.41 - y)}{0.41^2}, 0 \right). \tag{41}$$

5.1.2.2. Tecnologías utilizadas y entorno de ejecución

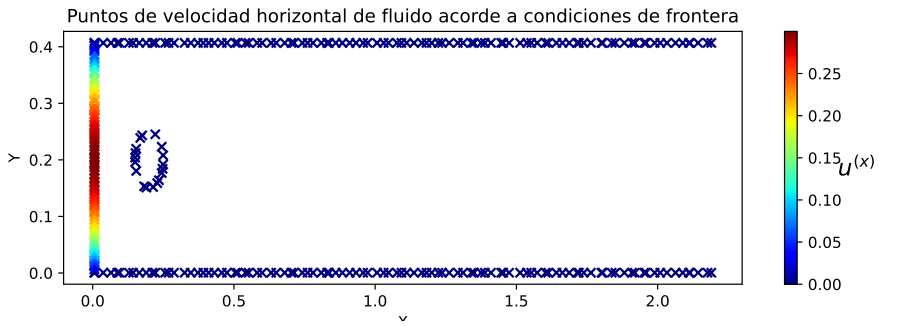
El experimento se llevó a cabo en la plataforma *Google Colab* con una suscripción *pro* para poder utilizar tarjetas gráficas. Se usó una tarjeta gráfica **Tesla T4** con 15 GB de RAM y un CPU con 51 GB de RAM.

Para la implementación de la red neuronal y el análisis de datos, se utilizaron las siguientes bibliotecas de Python:

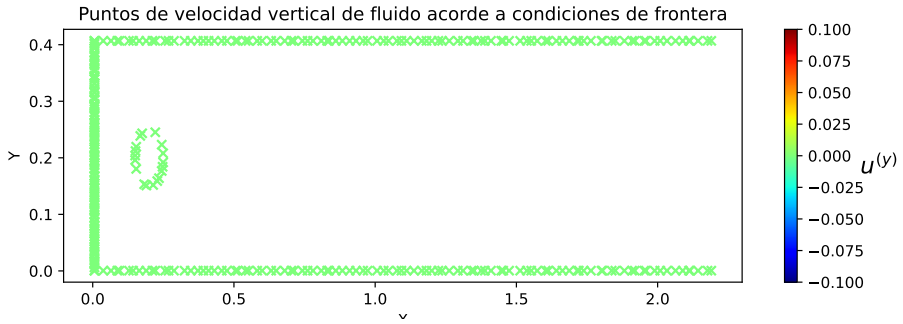
- PyTorch
- Numpy
- Matplotlib
- PyDOE

5.1.2.3. Generación de datos

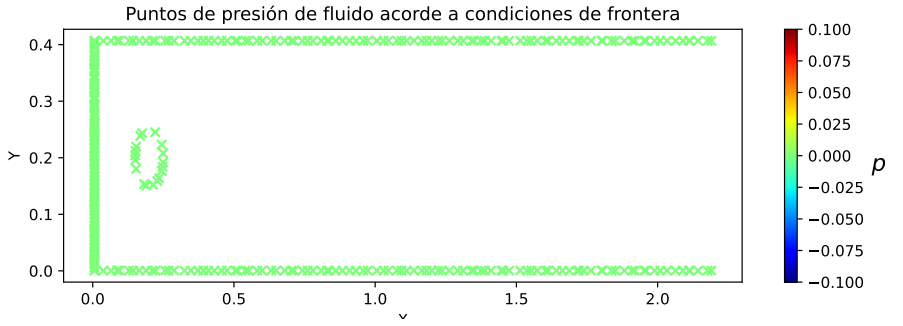
Para las condiciones de frontera se generaron 318 puntos en total, que incluyen todas las condiciones de frontera mencionadas. Cada punto se asocia con sus respectivas coordenadas x e y, además de incluir los valores objetivo para la velocidad en x, en y y la presión del fluido. La generación de estos puntos se realizó mediante un proceso de muestreo conocido como Latin Hypercube Sampling (LHS), el cual asegura una cobertura uniforme y eficiente del espacio de las fronteras, y es un método comúnmente empleado para la resolución de PDEs. En la Figura 19, se pueden observar los puntos obtenidos y los valores de estos para la velocidad del fluido en x, y y la presión. Se puede observar claramente los valores del flujo de entrada, dados por la Ecuación 41.



(a)



(b)



(c)

Figura 19: Condiciones de frontera para Navier-Stokes: (a) condiciones de frontera para velocidad en eje horizontal, (b) condiciones de frontera para velocidad en eje vertical, (c) condiciones de frontera para presión.

Adicionalmente, se generaron 22,345 puntos de colocación con los cuales se buscó cubrir todo el dominio. Estos se generaron usando igualmente LHS. Estos se pueden observar en la Figura 20. No se generaron puntos dentro del cilindro, debido a que está fuera del dominio del problema. La velocidad y presión deben a ser 0 dentro de este, ya que es un sólido por el cual no pasa el fluido.

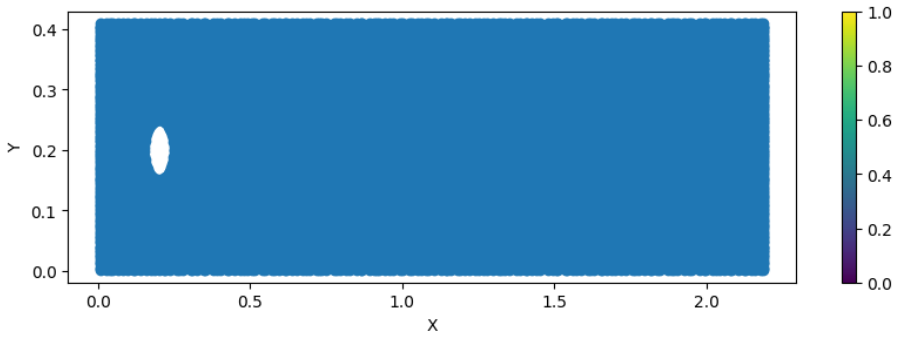


Figura 20: Puntos de colocación para Navier-Stokes.

5.1.2.4. Arquitectura de PINN

La arquitectura utilizada para la PINN para resolver el problema de mecánica de fluidos se encuentra en la Tabla 3. Además, también se puede observar un diagrama de esta en la Figura 21. Se usaron 2 datos de entrada, que fueron las coordenadas x e y de los puntos. Se utilizaron 4 capas ocultas con 50 neuronas cada una. Se empleó la función de activación tanh. En la capa de salida se usaron 3 neuronas, debido a que se buscó modelar la velocidad del fluido en el eje horizontal, vertical y la presión. En la Figura 21 se puede apreciar cómo se tiene una red neuronal feedforward que tiene como salidas la velocidad y la presión. A partir de estas se computan las primeras y segundas derivadas, con diferenciación automática, que luego son empleadas para calcular la función de pérdida física.

Número de capa	**Nombre de capa**	**Descripción de capa**
1	Input/Dense layer 1	Tamaño de capa de entrada de 2; 50 nodos en capa oculta.
2	tanh 1	Función de activación tanh.
3	Dense layer 2	50 nodos en capa oculta.
4	tanh 2	Función de activación tanh.
5	Dense layer 3	50 nodos en capa oculta.
6	tanh 3	Función de activación tanh.
7	Dense layer 4	50 nodos en capa oculta.
8	tanh 4	Función de activación tanh.
9	Output/Dense layer 5	50 nodos en capa oculta; tamaño de capa de salida de 3.

Tabla 3: Arquitectura de la red neuronal con 4 capas ocultas y 50 neuronas cada una.

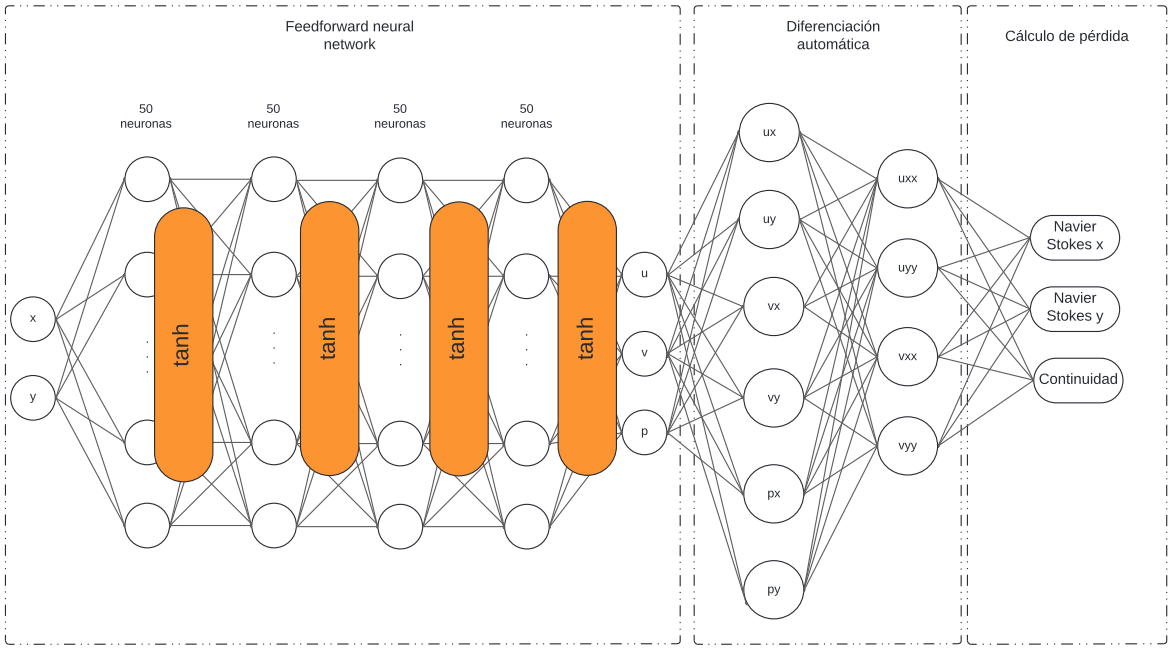


Figura 21: Esquema de arquitectura de PINN para Navier-Stokes.

5.1.2.5. Función de pérdida y entrenamiento

$$\mathrm{NS}_x = -\nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} \tag{42}$$

$$\mathrm{NS}_y = -\nu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} \tag{43}$$

$$\text{continuity} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \tag{44}$$

$$\mathcal{L}_{\text{física}} = \mathrm{MSE}(\mathrm{NS}_x, 0) + \mathrm{MSE}(\mathrm{NS}_y, 0) + \mathrm{MSE}(\text{continuity}, 0) \tag{45}$$

La función de pérdida utilizada está dividida en dos partes, al igual que con la ecuación de calor. La pérdida física se definió como se observa en la Ecuación 45, donde cada uno de los términos está definido en la Ecuación 42, la Ecuación 43 y la Ecuación 44. En este caso, se utiliza MSE para buscar reducir los resiudales del sistema de PDEs de Navier-Stokes a 0.

$$\mathcal{L}_{\mathrm{BC}} = \mathrm{MSE}(u_{\mathrm{BC}}, u^*_{\mathrm{BC}}) + \mathrm{MSE}(v_{\mathrm{BC}}, v^*_{\mathrm{BC}}) + \mathrm{MSE}(p_{\mathrm{BC}}, p^*_{\mathrm{BC}}) \tag{46}$$

La función de pérdida para las condiciones de frontera se observa en la Ecuación 46, donde u_{BC}, v_{BC} y p_{BC} representan los valores de las velocidades y la presión en los puntos de la frontera, y u^*_{BC}, v^*_{BC} y p^*_{BC} son los valores esperados en esas fronteras.

En esta expresión, $\mathcal{L}_{\text{BC}}$ se calcula como la suma de los errores cuadrados medios entre los valores predichos y los valores esperados de las variables en los puntos de la frontera. Esto asegura que la red neuronal cumpla con las condiciones de frontera impuestas por el problema físico. La función de pérdida usada por la red neuronal se compuso de la suma de $\mathcal{L}_{\text{BC}}$ con $\mathcal{L}_{\text{física}}$.

Para el entrenamiento se utilizó una combinación de ADAM y L-BFGS, una técnica bastante utilizada en diferentes experimentos [51]. Se usaron 2000 épocas con ADAM y un *learning rate* de 0.001. Con L-BFGS se usaron 1000 épocas y un *learning rate* de 0.1.

5.1.2.6. Resultados

En la Figura 22 se puede observar la pérdida obtenida durante el entrenamiento de la red neuronal. Es posible evidenciar que llegó a valores relativamente cercanos a 0. Esto indica que se redujeron los residuales de la PDE y además se ajustó la red neuronal a los valors de las condiciones de contorno. La velocidad horizontal, vertical y la presión del fluido se pueden apreciar en la Figura 23.

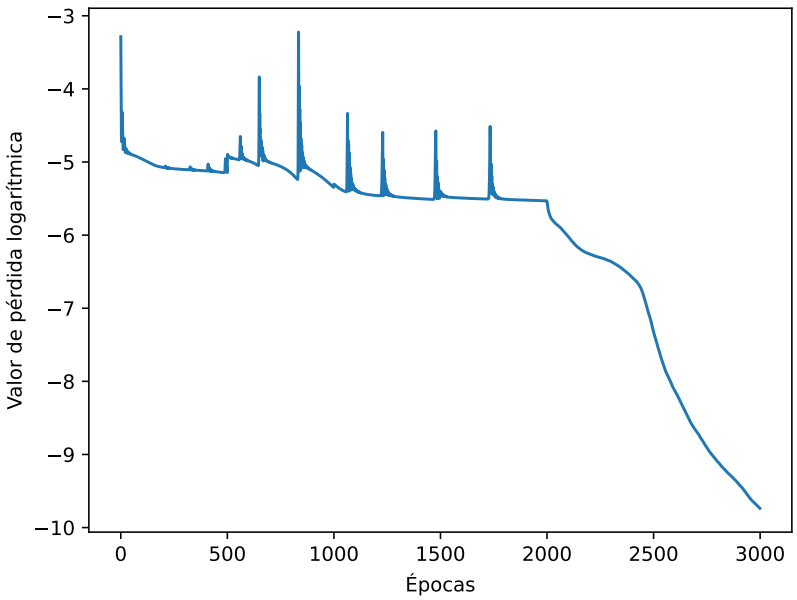
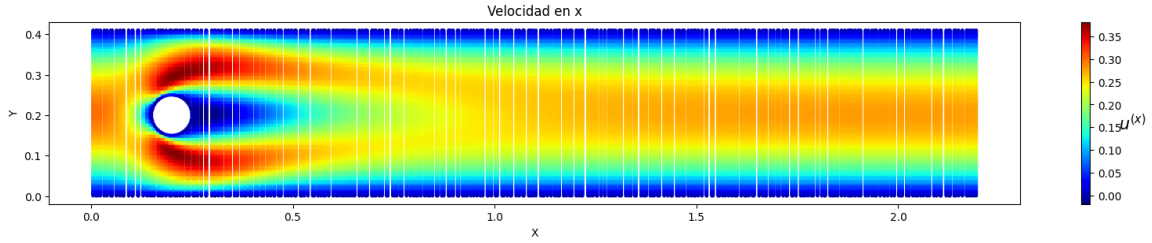
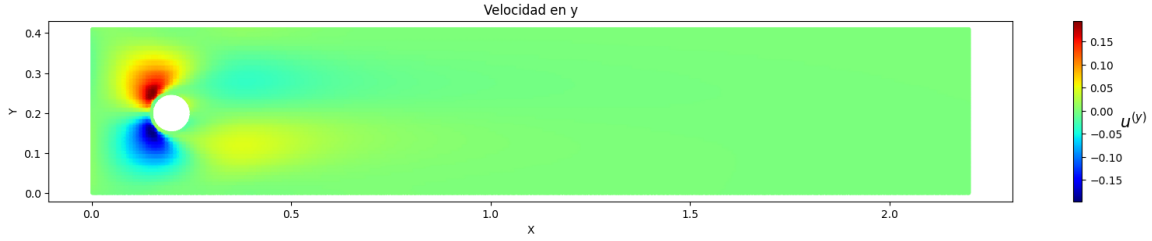


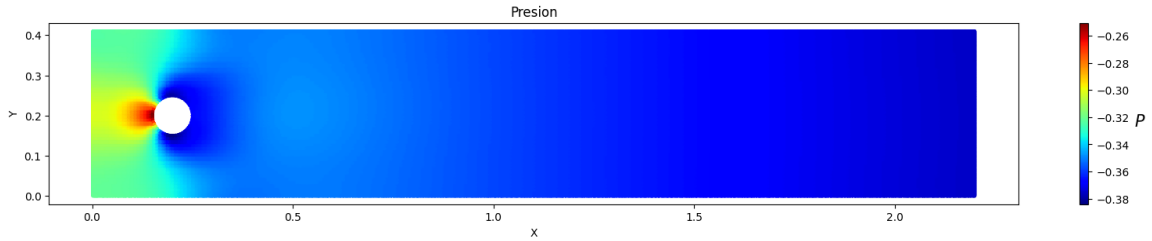
Figura 22: Pérdida de red neuronal para Navier-Stokes en escala logarítmica.



(a)



(b)



(c)

Figura 23: Resultados de PINN para Navier-Stokes. a) Velocidad horizontal de fluido a partir de Navier-Stokes modelada con PINN; b) Velocidad vertical de fluido a partir de Navier-Stokes modelada con PINN; c) Presión de fluido a partir de Navier-Stokes modelada con PINN.

5.2. Aplicación a piezoelectricidad

Luego de entender los componentes y funcionamiento básico de la PINN, se procedió a aplicarla en un nuevo problema, el efecto piezoeléctrico.

5.2.1. Efecto piezoeléctrico indirecto

En esta sección se evidencia el método utilizado para modelar el efecto piezoeléctrico indirecto. Para esto se introduce la definición del problema modelado y el enfoque utilizado para solucionarlo por medio de PINNs.

5.2.1.1. Definición del problema modelado

Se modeló una viga en voladizo con dos capas en dos dimensiones. Esto debido a que es una de las configuraciones más comunes al momento de evaluar la piezoelectricidad [52]. Las PDEs utilizadas para modelar este problema fueron las presentadas en la Ecuación 6 y la Ecuación 7, además de las ecuaciones constitutivas de la piezoelectricidad, observadas en la Ecuación 3 y la Ecuación 4. Para la configuración de la viga piezoeléctrica, el largo de la viga es de 0.1 metros y el alto de 0.001 metros. Para ambas capas de la viga se utilizó el material PVDF. Los coeficientes del material para la capa superior de la viga con dimensiones en metros se encuentra en la Tabla 4. Los coeficientes del material para la capa inferior de la viga con dimensiones en metros se encuentran en la Tabla 5.

Propiedad de material	**Valor**
c_{11}	$2.1836 \times 10^9\,\mathrm{N/m^2}$
c_{12}	$0.6332 \times 10^9\,\mathrm{N/m^2}$
c_{22}	$2.1836 \times 10^9\,\mathrm{N/m^2}$
c_{33}	$0.775 \times 10^9\,\mathrm{N/m^2}$
e_{11}	$0\,\mathrm{C/m^2}$
e_{13}	$0\,\mathrm{C/m^2}$
e_{31}	$-2.904 \times 10^{-2}\,\mathrm{C/m^2}$
e_{33}	$-5.157 \times 10^{-2}\,\mathrm{C/m^2}$
e_{14}	$0\,\mathrm{C/m^2}$
e_{34}	$0\,\mathrm{C/m^2}$
ϵ_1	$-1.062 \times 10^{-10}\,\mathrm{C/N/m^2}$
ϵ_2	$-1.041 \times 10^{-10}\,\mathrm{C/N/m^2}$

Tabla 4: Propiedades de material PVDF para capa superior de viga en voladizo paralela piezoeléctrica en metros [29].

En la Figura 24, se pueden observar las condiciones de frontera utilizadas para el dispositivo seleccionado. Para el potencial eléctrico se utilizaron las condiciones de frontera de los bordes superior e inferior de la viga, donde el valor de cada una es de 100 voltios y

Propiedad de material	**Valor**
c_{11}	$2.1836 \times 10^9\,\mathrm{N/m^2}$
c_{12}	$0.6332 \times 10^9\,\mathrm{N/m^2}$
c_{22}	$2.1836 \times 10^9\,\mathrm{N/m^2}$
c_{33}	$0.775 \times 10^9\,\mathrm{N/m^2}$
e_{11}	$0\,\mathrm{C/m^2}$
e_{13}	$0\,\mathrm{C/m^2}$
e_{31}	$2.904 \times 10^{-2}\,\mathrm{C/m^2}$
e_{33}	$5.157 \times 10^{-2}\,\mathrm{C/m^2}$
e_{14}	$0\,\mathrm{C/m^2}$
e_{34}	$0\,\mathrm{C/m^2}$
ϵ_1	$-1.062 \times 10^{-10}\,\mathrm{C/N/m^2}$
ϵ_2	$-1.041 \times 10^{-10}\,\mathrm{C/N/m^2}$

Tabla 5: Propiedades de material PVDF para capa inferior de viga en voladizo paralela piezoeléctrica en metros [29].

0 voltios respectivamente. La viga en voladizo está fija en su lado izquierdo, por lo que el desplazamiento en x y y es 0 en este subdominio. Por otra parte, se asume que no hay estrés sobre el extremo libre de la viga (extremo derecho), por lo que la derivada normal del estrés en este punto es 0. Finalmente se asume que la viga es aislante, por lo que no sale carga de ella, así que, en los extremos derecho e izquierdo, la derivada normal del desplazamiento eléctrico es 0.

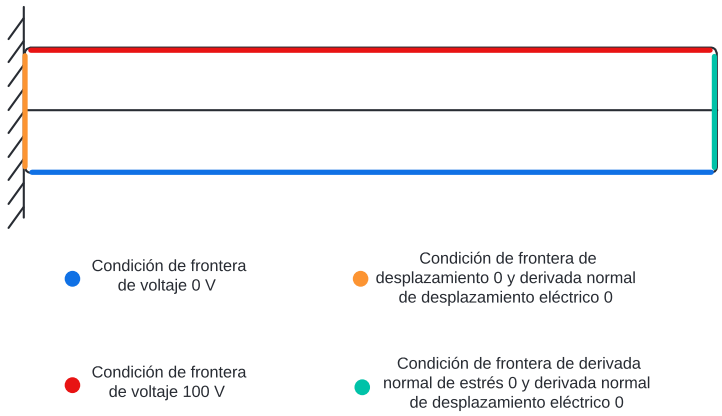


Figura 24: Condiciones de frontera para viga en voladizo piezoeléctrica paralela.

5.2.1.2. Tecnologías utilizadas y entorno de ejecución

El experimento se llevó a cabo en la plataforma *Google Colab* con una suscripción *pro* para poder utilizar tarjetas gráficas. Se usó una tarjeta gráfica **Tesla T4** con 15 GB de RAM y un CPU con 51 GB de RAM.

Se definió el uso de un servidor de MLflow [53] remoto, por medio de una plataforma web llamada *DagsHub*, para poder hacer seguimiento de los experimentos relacionados con el entrenamiento de la PINN, incluyendo el registro de métricas de rendimiento, gráficos de pérdidas, y gráficos de resultados del modelo.

Para la implementación de la red neuronal y el análisis de datos, se utilizaron las siguientes bibliotecas de Python:

- PyTorch
- Numpy
- Matplotlib
- PyDOE

5.2.1.3. Generación de datos

Debido a que para las PINNs se incluyen términos de pérdida de condiciones de contorno, se deben generar datos para poder evaluarlas. En este caso, las condiciones de desplazamiento y voltaje, mencionadas, se integraron directamente en la arquitectura de la red, como se verá más adelante. Por esto, solo se generaron datos para evaluar las condiciones de frontera de estrés y desplazamiento eléctrico, como se observa en la Figura 25. Se generaron 150 puntos en las fronteras. Acá se tienen las coordenadas x e y de cada punto. La generación de estos puntos se realizó mediante un proceso de muestreo conocido como *Latin Hypercube Sampling* (LHS), el cual asegura una cobertura uniforme y eficiente del espacio de las fronteras, y es un método comúnmente empleado para la resolución de PDEs [54].

Por otra parte, se generaron 22500 puntos de colocación en el dominio del problema para entrenamiento y 160000 puntos de prueba. La generación de estos se realizó igualmente usando LHS. En general, al usar una cantidad grande de datos en DL, ayuda a la generalización de la red neuronal [34]. Este escenario no es la excepción, la cual es una de las razones de usar esa cantidad de puntos. Múltiples aproximaciones a distintos problemas físicos usando PINNs utilizan cantidades de puntos de colocación similares [6], [55]. El uso de puntos de colocación puede aumentar la estabilidad del entrenamiento y capturar con mayor precisión variaciones en el dominio y el comportamiento de las PDEs en este [56]. Esta selección de puntos se puede ver en la Figura 26a.

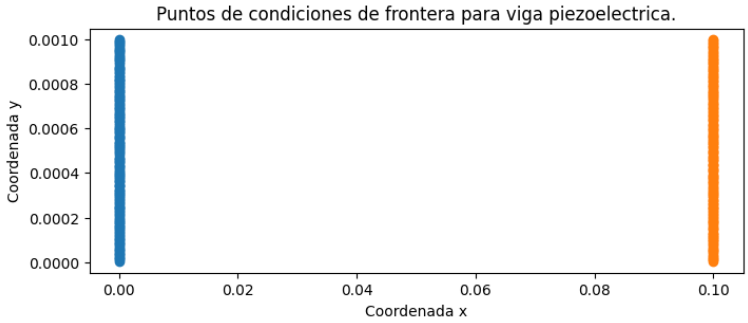
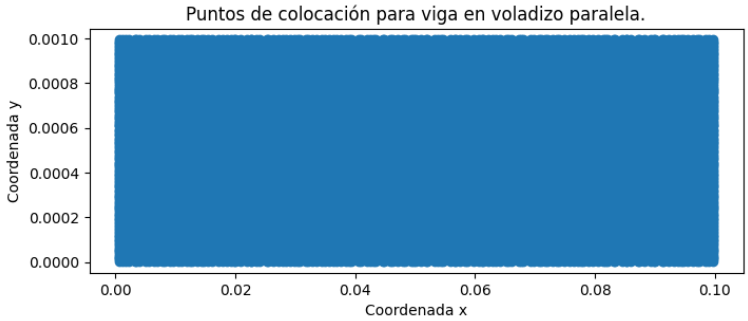
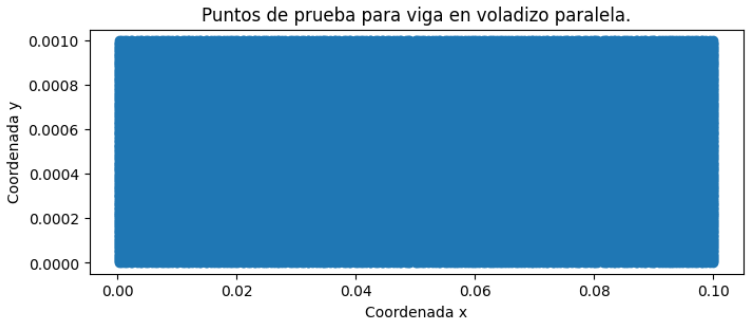


Figura 25: Puntos empleados para satisfacer las condiciones de frontera de estrés y desplazamiento.



(a) Puntos de colocación para modelación de efecto piezoeléctrico indirecto para configuración de viga en voladizo.



(b) Puntos de prueba para modelación de efecto piezoeléctrico indirecto para configuración de viga en voladizo.

Figura 26: Puntos generados para evaluar en PINN en la configuración de la viga en voladizo bajo el efecto piezoeléctrico indirecto.

Adicionalmente a las coordenadas x e y de los puntos de colocación, se añadieron los coeficientes correspondientes de cada punto de la Tabla 4 y la Tabla 5. Esto se realizó debido a que se modelan dos capas piezoeléctricas que tienen propiedades ligeramente distintas, por lo que es práctico añadir los coeficientes como parte del conjunto de datos. Estos fueron útiles al definir la función de pérdida. En la Figura 27, como un ejemplo de lo mencionado, se puede observar el valor de e_{33} para cada punto de colocación.

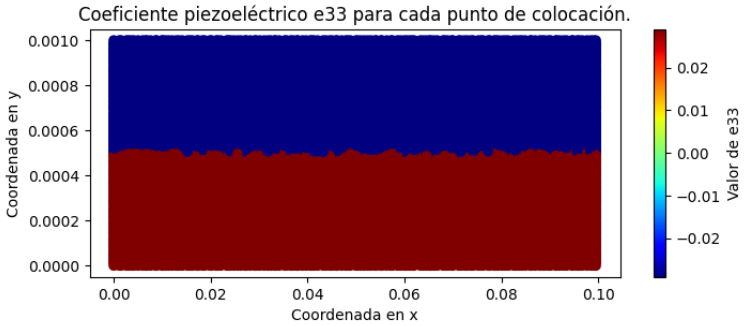


Figura 27: Ejemplo de inclusión de coeficientes de material en conjunto de datos para cada punto de colocación en configuración de viga.

5.2.1.4. Arquitecturas de PINN empleadas

Para modelar el efecto piezoeléctrico indirecto, se emplearon dos arquitecturas diferentes de PINN. El propósito de esto fue evaluar su rendimiento al momento de modelar el problema y determinar cuál lo hizo de forma más eficiente.

La primera arquitectura empleada, denominada en este trabajo como uniforme, se puede resumir en la Tabla 6. Además, en la Figura 28 se puede observar un diagrama de cómo se modeló y de cómo se obtuvieron las derivadas de las funciones. Se evidencia que es una arquitectura con una cantidad uniforme de neuronas por capa. Es importante destacar que, para solventar las PDEs de piezoelectricidad indirecta, se utilizaron más neuronas en la red neuronal. También, se modelaron múltiples salidas, que incluyeron el estrés y el desplazamiento eléctrico, lo cual permitió que se usaran solo derivadas de primer orden [55].

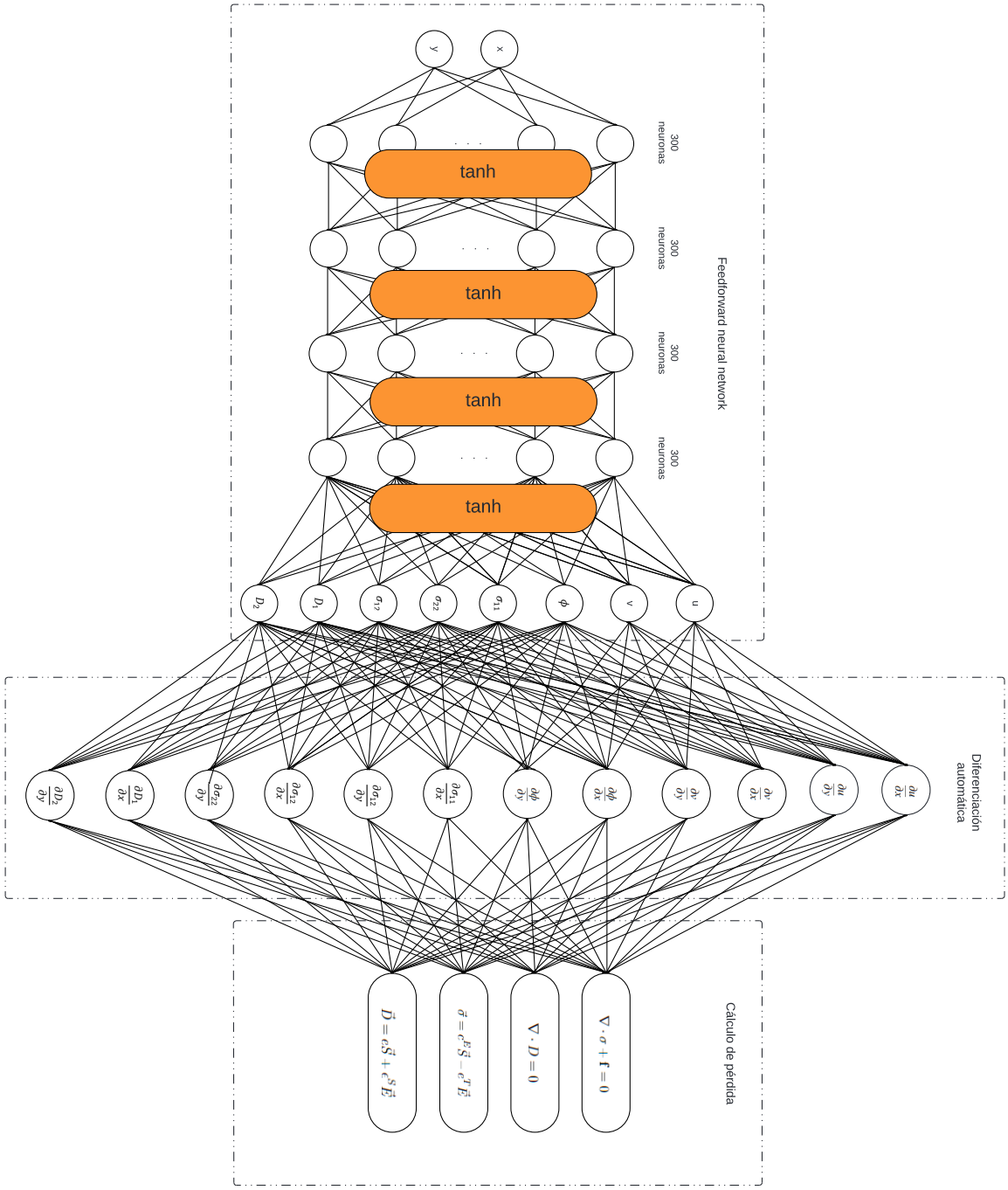


Figura 28: Esquema de arquitectura uniforme de PINN para efecto piezoeléctrico indirecto.

Número de capa	Nombre de capa	Descripción de capa
1	Input/Dense layer 1	Tamaño de capa de entrada de 2; 300 nodos en capa oculta.
2	tanh 1	Función de activación tanh.
3	Dense layer 2	300 nodos en capa oculta.
4	tanh 2	Función de activación tanh.
5	Dense layer 3	300 nodos en capa oculta.
6	tanh 3	Función de activación tanh.
7	Dense layer 4	300 nodos en capa oculta.
8	tanh 4	Función de activación tanh.
9	Output/Dense layer 5	300 nodos en capa oculta; tamaño de capa de salida de 8.

Tabla 6: Arquitectura de la red neuronal con 4 capas ocultas y 300 neuronas cada una.

Para la segunda arquitectura evaluada, denominada piramidal en este trabajo, se utilizó una distribución de nodos por capa de manera incremental y simétrica. A medida que se avanza en las capas, el número de nodos aumenta hasta llegar a un punto máximo, tras el cual se comienza a decrementar siguiendo el mismo patrón simétrico. Esto se observa en el resumen de la arquitectura en la Tabla 7. Se puede observar un diagrama de la arquitectura en la Figura 29

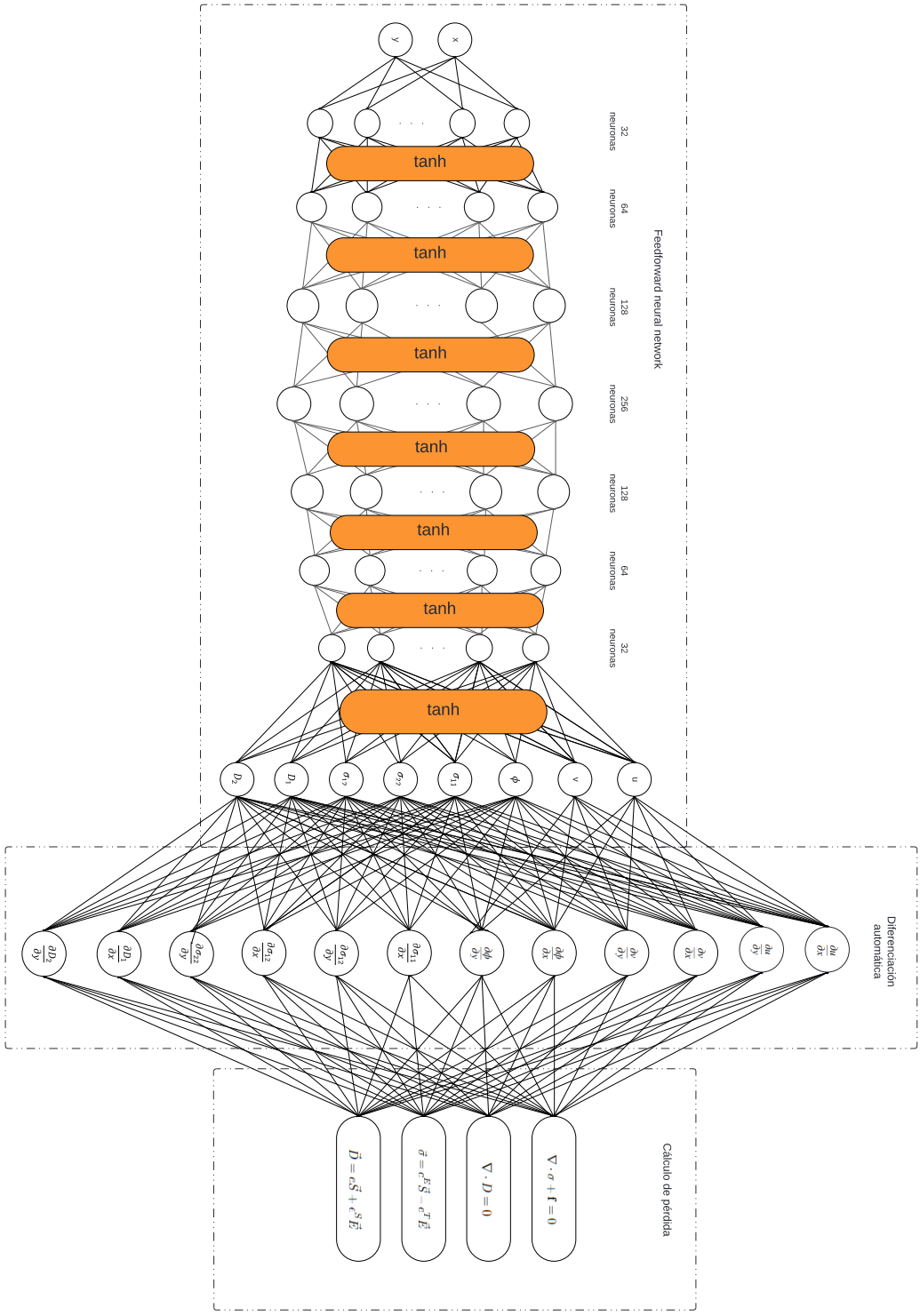


Figura 29: Esquema de arquitectura piramidal de PINN para efecto piezoeléctrico indirecto.

Número de capa	Nombre de capa	Descripción de capa
1	Input/Dense layer 1	Tamaño de capa de entrada de 2; 32 nodos en capa oculta.
2	tanh 1	Función de activación tanh.
3	Dense layer 2	32 nodos en capa oculta.
4	tanh 2	Función de activación tanh.
5	Dense layer 3	64 nodos en capa oculta.
6	tanh 3	Función de activación tanh.
7	Dense layer 4	128 nodos en capa oculta.
8	tanh 4	Función de activación tanh.
9	Dense layer 5	256 nodos en capa oculta.
10	tanh 5	Función de activación tanh.
11	Dense layer 6	128 nodos en capa oculta.
12	tanh 6	Función de activación tanh.
13	Dense layer 7	64 nodos en capa oculta.
14	tanh 7	Función de activación tanh.
15	Dense layer 8	32 nodos en capa oculta.
16	tanh 8	Función de activación tanh.
17	Output/Dense layer 9	32 nodos en capa oculta; tamaño de capa de salida de 8.

Tabla 7: Arquitectura de la red neuronal piramidal con múltiples capas ocultas.

5.2.1.5. Función de pérdida y entrenamiento

Para ambas arquitecturas, la función de pérdida total es la misma y consta de dos partes. La primera es la pérdida física $\mathcal{L}_{\text{física}}$, que se calcula como la suma de las pérdidas mecánicas, eléctricas y de divergencia, donde cada término de pérdida es un error cuadrático medio (MSE) de los residuales correspondientes,

$$\begin{aligned}
\sigma_{11} &= c_{11}u_x + c_{12}v_y + e_{11}\phi_x + e_{31}\phi_y, \\
\sigma_{22} &= c_{12}u_x + c_{22}v_y + e_{13}\phi_x + e_{33}\phi_y, \\
\sigma_{12} &= c_{33}\frac{1}{2}(u_y + v_x) + e_{14}\phi_x + e_{34}\phi_y, \\
D_x &= e_{11}u_x + e_{13}v_y + e_{14}(u_y + v_x) - \epsilon_1\phi_x, \\
D_y &= e_{31}u_x + e_{33}v_y + e_{34}(u_y + v_x) - \epsilon_2\phi_y.
\end{aligned} \tag{47}$$

$$
\begin{aligned}
\text{residual}_{\sigma_{11}} &= \sigma_{11} - \sigma_{11,\text{pred}}, \\
\text{residual}_{\sigma_{22}} &= \sigma_{22} - \sigma_{22,\text{pred}}, \\
\text{residual}_{\sigma_{12}} &= \sigma_{12} - \sigma_{12,\text{pred}}, \\
\text{residual}_{D_x} &= D_x - D_{x,\text{pred}}, \\
\text{residual}_{D_y} &= D_y - D_{y,\text{pred}}.
\end{aligned} \tag{48}
$$

$$
\begin{aligned}
\text{divergencia}_{\sigma_1} &= \frac{\partial \sigma_{11}}{\partial x} + \frac{\partial \sigma_{12}}{\partial y}, \\
\text{divergencia}_{\sigma_2} &= \frac{\partial \sigma_{12}}{\partial x} + \frac{\partial \sigma_{22}}{\partial y}, \\
\text{divergencia}_{D} &= \frac{\partial D_x}{\partial x} + \frac{\partial D_y}{\partial y}.
\end{aligned} \tag{49}
$$

$$
\begin{aligned}
\text{MSE}_{\text{mec}} &= \frac{1}{N} \sum_{i=1}^{N} \left(\text{residual}^2_{\sigma_{11},i} + \text{residual}^2_{\sigma_{22},i} + \text{residual}^2_{\sigma_{12},i} \right), \\
\text{MSE}_{\text{eléct}} &= \frac{1}{N} \sum_{i=1}^{N} \left(\text{residual}^2_{D_{x,i}} + \text{residual}^2_{D_{y,i}} \right), \\
\text{MSE}_{\text{div}} &= \frac{1}{N} \sum_{i=1}^{N} \left(\text{divergencia}^2_{\sigma_{1,i}} + \text{divergencia}^2_{\sigma_{2,i}} + \text{divergencia}^2_{D_i} \right).
\end{aligned}
$$

$$
\mathcal{L}_{\text{física}} = \text{MSE}_{\text{mec}} + \text{MSE}_{\text{eléct}} + \text{MSE}_{\text{div}}. \tag{50}
$$

Debido a que se está modelando tanto desplazamiento y potencial, como estrés y desplazamiento eléctrico, estos se deben usar para cumplir con las ecuaciones constitutivas piezoeléctricas. Esto se puede realizar, calculando el estrés y el desplazamiento dieléctrico por medio de la Ecuación 3 y la Ecuación 4. Este cálculo se realiza por medio de la Ecuación 47, donde se usa la notación de subíndice para las derivadas. Luego, se realiza una comparación de estos valores calculados usando desplazamientos y potencial eléctrico, con las predicciones de estrés y divergencia eléctrica dadas por la red neuronal en sus nodos de salida, lo que se observa en la Ecuación 48. En esta ecuacion, donde se tiene el subíndice "pred", se refiere a las predicciones directas de los nodos de la PINN.

Por otra parte, se calculan las divergencias de estrés y desplazamiento eléctrico en la Ecuación 49, donde se usa notación de Leibniz para las derivadas. Finalmente se calcula MSE para los residuales y las divergencias, los cuales conforman la pérdida física en la Ecuación 50. En esta última ecuación el subíndice i se refiere a cada punto de colocación evaluado en el dominio del problema y N a la cantidad de puntos de colocación empleados.

Por otra parte, se tiene la función de pérdida de condiciones de frontera $\mathcal{L}_{\text{BC}}$, donde se busca cumplir con las dos condiciones de frontera definidas en la Ecuación 51.

$$
\begin{aligned}
\sigma(\mathbf{x}) \cdot \mathbf{n} &= 0, \qquad \forall \mathbf{x} \in \Omega_t, \\
\mathbf{D} \cdot \mathbf{n} &= 0, \qquad \forall \mathbf{x} \in \Omega_p.
\end{aligned} \tag{51}
$$

Para garantizar que estas condiciones se cumplan, se define la pérdida en los subdominios Ω_t y Ω_p como la suma de los errores cuadráticos medios (MSE) en estos subdominios,

$$\mathcal{L}_{\text{BC}} = \text{MSE}_{\sigma} + \text{MSE}_{\mathbf{D}}, \tag{52}$$

donde,

$$\text{MSE}_{\sigma} = \frac{1}{N_{\Omega_t}} \sum_{i=1}^{N_{\Omega_t}} \left(\sigma_{pred}(\mathbf{x}_i) \cdot \mathbf{n}_i\right)^2, \tag{53}$$

$$\text{MSE}_{\mathbf{D}} = \frac{1}{N_{\Omega_p}} \sum_{i=1}^{N_{\Omega_p}} \left(\mathbf{D}_{pred}(\mathbf{x}_i) \cdot \mathbf{n}_i\right)^2. \tag{54}$$

Aquí, N_{Ω_t} y N_{Ω_p} son el número de puntos de colocación en los subdominios Ω_t y Ω_p respectivamente, y $\sigma_{pred}(\mathbf{x}_i) \cdot \mathbf{n}_i$ y $\mathbf{D}_{pred}(\mathbf{x}_i) \cdot \mathbf{n}_i$ son los valores calculados en los puntos de frontera. El término de la función de pérdida $\mathcal{L}_{\text{BC}}$ minimiza la discrepancia en estas condiciones de frontera durante el entrenamiento de la red.

Finalmente, se unen el término físico de la función de pérdida con el término de condiciones de contorno, obteniendo la siguiente expresión para la función de pérdida total,

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{física}} + \mathcal{L}_{\text{BC}}. \tag{55}$$

Para implementar las condiciones de frontera para el desplazamiento y el potencial eléctrico, estos se integraron directamente en la arquitectura de la red como restricciones rígidas. De esta forma, se busca que las condiciones de frontera formen parte de la estructura de la red, satisfaciéndolas automáticamente, en lugar de usar datos para satisfacerla [45]. La forma de imponer estas condiciones se observa en las siguientes ecuaciones [29],

$$u_{\text{modified}} = x \cdot u, \tag{56}$$

$$v_{\text{modified}} = x \cdot v, \tag{57}$$

$$\phi_{\text{constraint}} = \frac{y}{\text{altura}} \cdot \text{voltaje}, \tag{58}$$

$$\phi_{\text{modified}} = y \cdot (y-1) \cdot \phi + \phi_{\text{constraint}}. \tag{59}$$

En estas ecuaciones:

- u_{modified} y v_{modified} representan los desplazamientos modificados que incorporan las condiciones de frontera, donde u y v son las predicciones originales de la red para el desplazamiento, y x es la coordenada espacial en la dirección correspondiente. Multiplicar u y v por x asegura que estos desplazamientos se anulen en $x = 0$, cumpliendo con la condición de frontera.

- ϕ_{modified} es el potencial eléctrico modificado, donde ϕ es la predicción original del potencial, y y es la coordenada espacial en la dirección correspondiente. El término $\phi_{\text{constraint}}(x, y)$ ajusta el potencial para que cumpla con un gradiente esperado en la viga, basado en el voltaje máximo y la altura de la viga.

Estas expresiones aseguran que las condiciones de frontera se integran de manera natural en la salida de la red neuronal, eliminando la necesidad de aplicar un término adicional de penalización en la función de pérdida para satisfacer estas condiciones.

Finalmente, el entrenamiento se realizó con 1000 épocas usando ADAM como optimizador, con un *learning rate* de 0.001, seguido de 200 épocas de L-BFGS como optimizador, con *learning rate* de 0.01.

5.2.1.6. Validación de resultados

Para validardar la precisión de los resultados obtenidos con la Red Neuronal Informada por la Física (PINN), se tomó como referencia un ejercicio previamente resuelto utilizando el método de elementos finitos (FEM) [57]. Este enfoque permitió una comparación directa entre las predicciones de la PINN y los resultados ya validados por FEM. Se empleó la métrica previamente usada, Error Relativo Medio L2, definida en la Ecuación 38.

5.2.2. Efecto piezoeléctrico directo

A continuación se modeló el efecto piezoeléctrico directo con una configuración similar al efecto piezoeléctrico indirecto previamente modelado.

5.2.2.1. Definición del problema modelado

Al igual que en el efecto piezoeléctrico indirecto, se modeló una viga en voladizo con dos capas en dos dimensiones. Las PDEs utilizadas para modelar el problema fueron, igualmente, las presentadas en la Ecuación 6 y la Ecuación 7. Además, también se usaron las ecuaciones constitutivas de la piezoelectricidad presentadas en la Ecuación 3 y la Ecuación 4.

Las dimensiones de la viga fueron las mismas que para el problema anterior, teniendo 0.1 metros de largo y 0.001 metros de alto. Los coeficientes del material piezoeléctrico para la capa superior de la viga se encuentran en la Tabla 4 y para la inferior en Tabla 5.

La variación más importante de este problema, es que en lugar de aplicar un voltaje a la viga que genere una deformación, se aplica una fuerza que debe generar un voltaje. Por esto, las condiciones de frontera para este problema se modifican ligeramente en comparación con el efecto piezoeléctrico indirecto. En este caso, como se observa en la Figura 30, ya no se tiene una condición de frontera de 100 voltios en la parte superior de la viga. Solo se fuerza a que la parte inferior tenga 100 voltios. Además, la fuerza mencionada que se aplica a la viga es de 1 N.

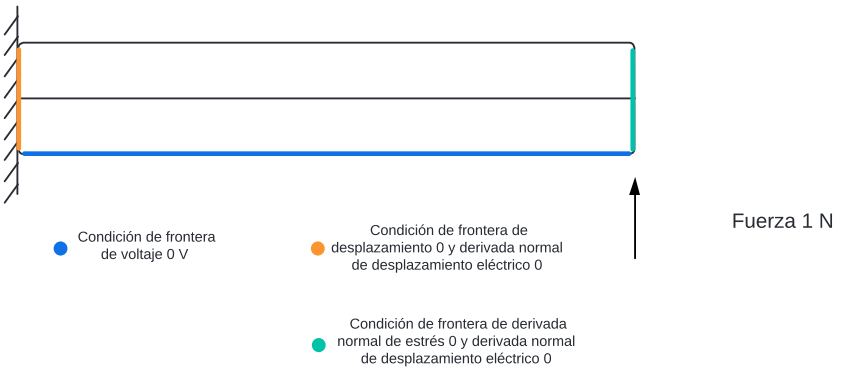


Figura 30: Configuración de sistema con condiciones de frontera para viga en voladizo paralela bajo efecto piezoeléctrico directo.

5.2.2.2. Tecnologías utilizadas y entorno de ejecución

Las tecnologías utilizadas y el entorno de ejecución fueron exactamente los mismos empleados para modelar el efecto piezoeléctrico indirecto, en la Subsubsección 5.2.1.2.

5.2.2.3. Generación de datos

Para modelar el efecto piezoeléctrico directo se utilizó el mismo conjunto de datos utilizado para el efecto piezoeléctrico indirecto, presentado en la Subsubsección 5.2.1.3. Esto debido a que solo se generaron puntos de evaluación del dominio, que es el mismo para ambos problemas. Por lo tanto, no fue necesario cambiar valores. La cantidad de puntos de colocación y de prueba es la misma.

5.2.2.4. Arquitectura de PINN empleada

En la modelación del efecto piezoeléctrico directo, se utilizó únicamente una de las arquitecturas propuestas. Se utilizó la arquitectura piramidal previamente mencionada, cuya organización se observa en la Tabla 7 y cuyo diagrama se observa en la Figura 29.

5.2.2.5. Función de pérdida y entrenamiento

La función de pérdida para el efecto piezoeléctrico directo sigue la misma estructura que la presentada en la Subsubsección 5.2.1.5, en la cual se consideran los términos mecánicos, eléctricos y de divergencia. Sin embargo, se realizó la incorporación de una fuerza adicional

en los términos de divergencia para reflejar el efecto piezoeléctrico directo. Por lo tanto, estas ecuaciones se modifican de la siguiente manera,

$$\text{divergencia}_{\sigma_1} = \frac{\partial \sigma_{11}}{\partial x} + \frac{\partial \sigma_{12}}{\partial y} + F_x,$$
$$\text{divergencia}_{\sigma_2} = \frac{\partial \sigma_{12}}{\partial x} + \frac{\partial \sigma_{22}}{\partial y} + F_y.$$

Acá el valor de F_x es de 0 N y el valor de F_y es de 1 N.

Adicionalmente, se realizó una modificación a la imposición de condiciones de frontera en la red neuronal. Particularmente, se realizó en la condición para el potencial eléctrico. Ahora se define como

$$\phi_{\text{modified}} = y \cdot \phi,$$

donde y representa la coordenada espacial en la dirección vertical. De esta manera, se busca asegurar que en el inferior de la viga, el voltaje sea 0.

Se utilizaron 1000 épocas de ADAM como optimizador, con *learning rate* de 0.001, seguidas de 180 épocas de L-BFGS como optimizador, con un *learning rate* de 0.01.

CAPÍTULO 6

Resultados

A continuación, se presentan los resultados de las PINN para la modelación de la viga en voladizo bajo el efecto piezoeléctrico.

6.1. Gestión y seguimiento de experimentos

Como se mencionó, para el registro de los experimentos, se configuró un servidor de MLflow remoto. En dicha herramienta se almacenaron los resultados de los experimentos realizados con las PINNs. Estos experimentos se pueden observar en la Figura 31. Por otra parte, en la Figura 32 se observan los detalles correspondientes a un experimento, donde se almacenaron parámetros utilizados en el entrenamiento, así como las métricas obtenidas. En la Figura 33 se puedem observar de nuevo la las métricas relacionadas al experimento, así como la pérdida de la red neuronal en el entrenamiento. Finalmente, en la Figura 34 se observa que se pudieron almacenar los gráficos, como el potencial eléctrico, de los resultados del experimento dentro de MLflow.

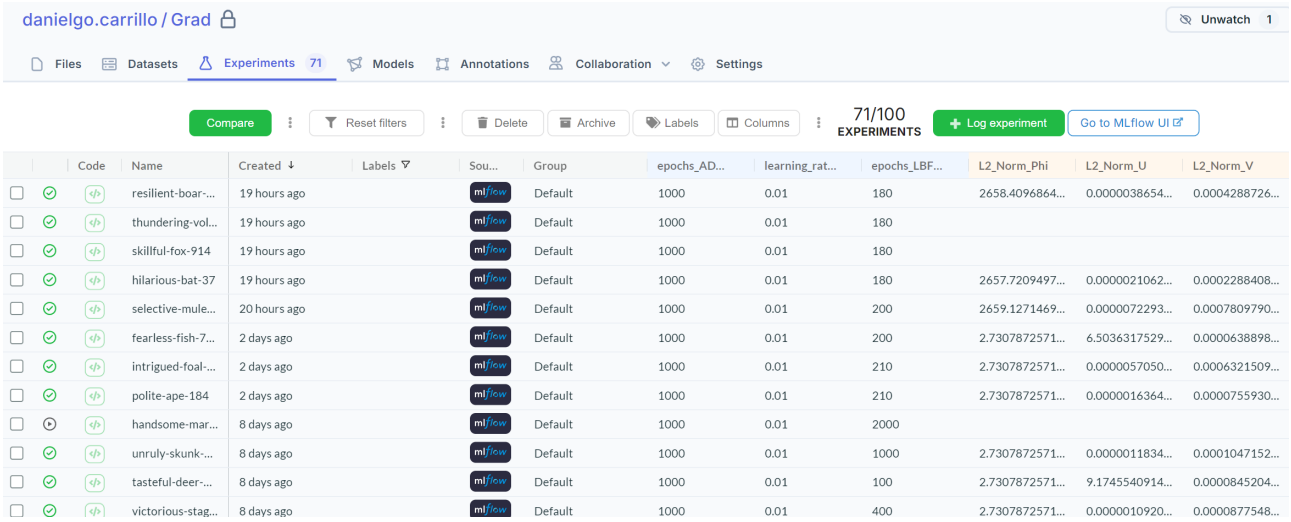
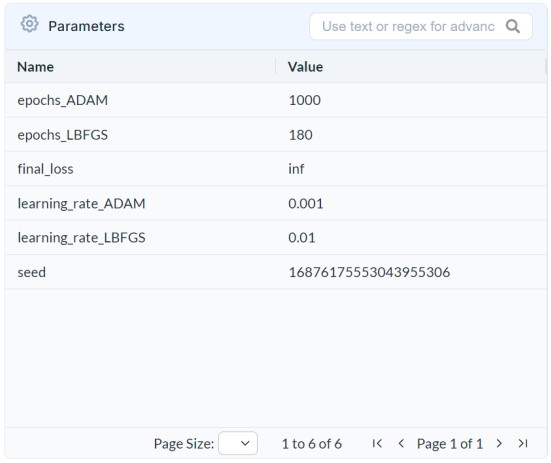
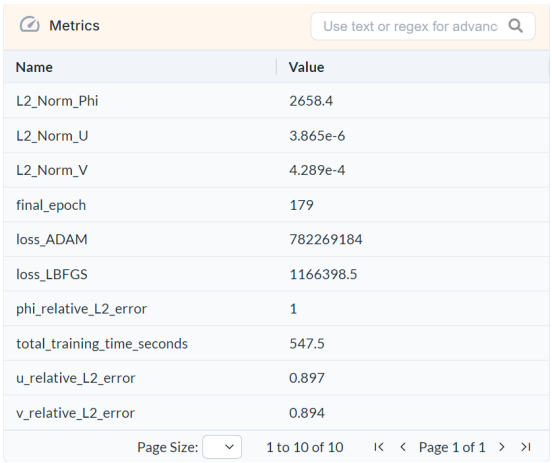


Figura 31: Vista de herramienta de almacenamiento de experimentos en MLflow



(a)



(b)

Figura 32: Ejemplo de detalles del experimento en MLflow. a) parámetros registrados de experimento; b) métricas registradas de experimento.

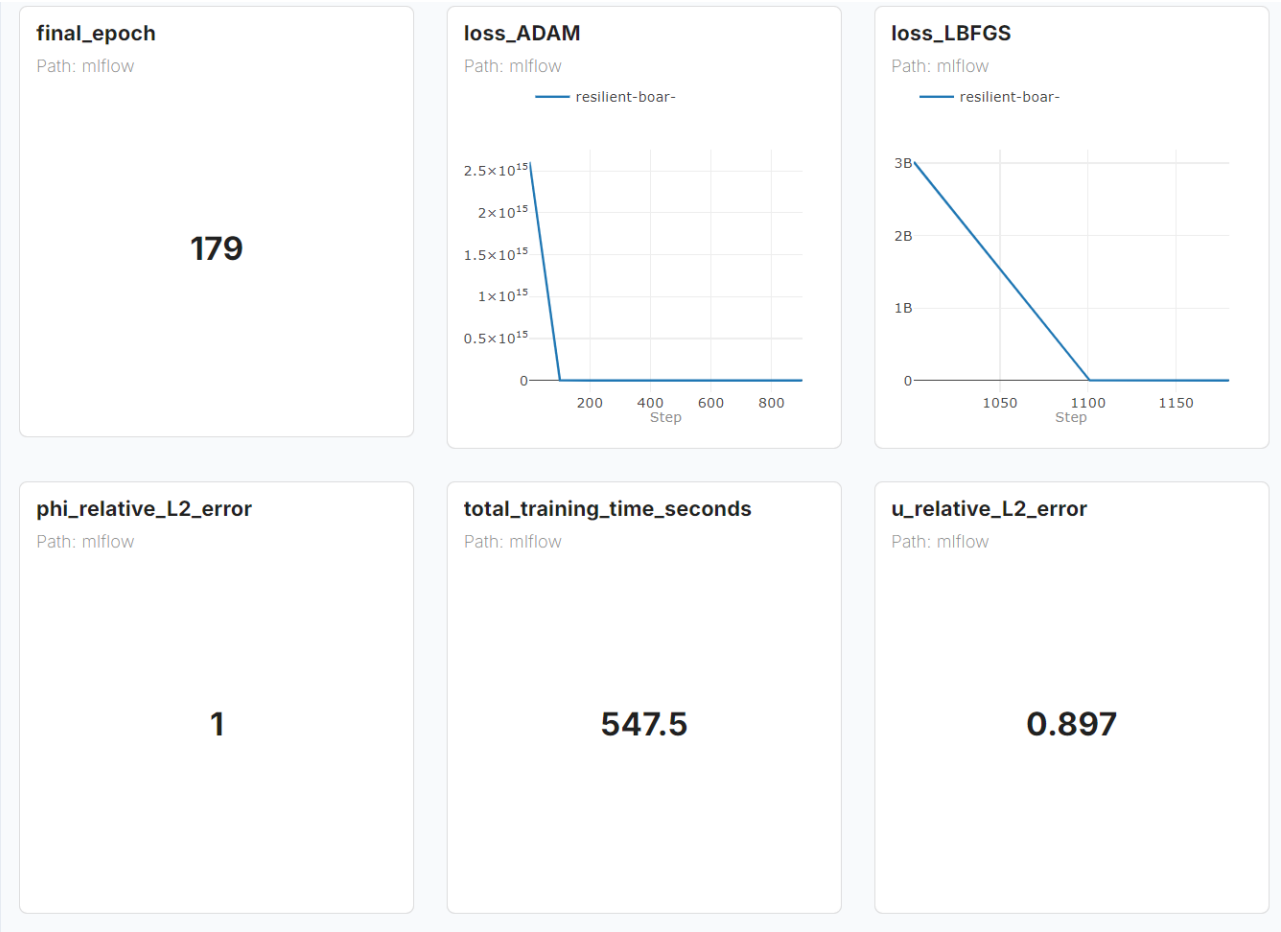


Figura 33: Métricas importantes y pérdida almacenada por experimento en MLflow.

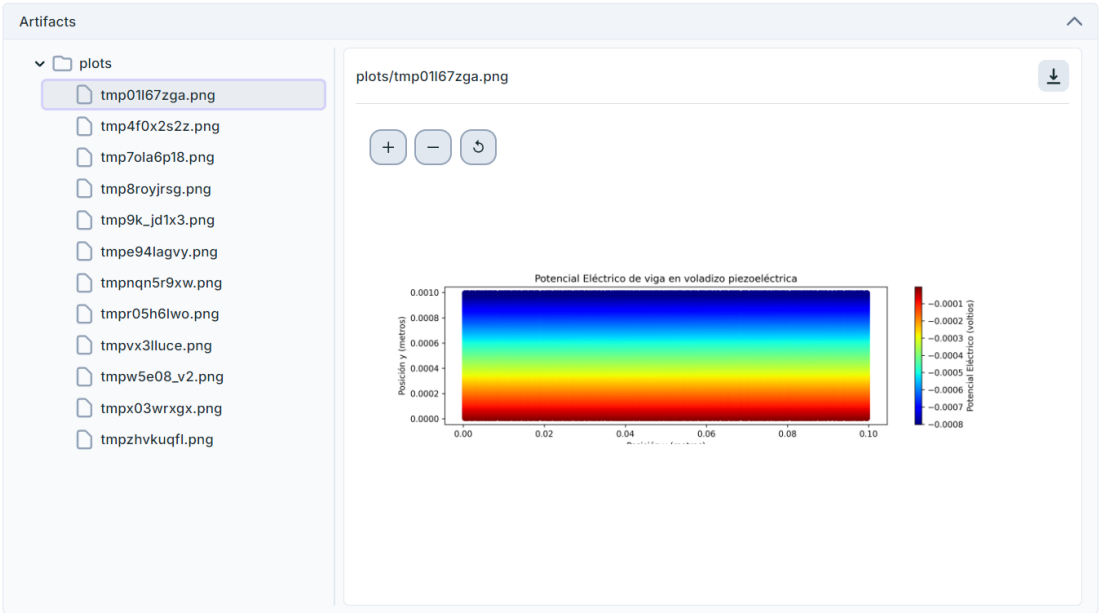


Figura 34: Gráficos almacenados en experimento de MLflow.

6.2. Efecto piezoeléctrico indirecto

A continuación se presentan los resultados obtenidos de la modelación del efecto piezoeléctrico indirecto, con las dos arquitecturas propuestas.

6.2.1. Resultados del entrenamiento de las redes neuronales

En la Figura 35 se puede observar la pérdida de la red neuronal con arquitectura uniforme, a lo largo de las épocas de entrenamiento en escala logarítmica, para la viga en voladizo, con dimensiones en metros. Se hizo una escala logarítmica debido a que el valor de la pérdida inicial es muy alta en comparación a los valores subsiguientes. En la Figura 36 se puede observar la pérdida, igualmente en escala logarítmica, para la red neuronal con arquitectura piramidal.

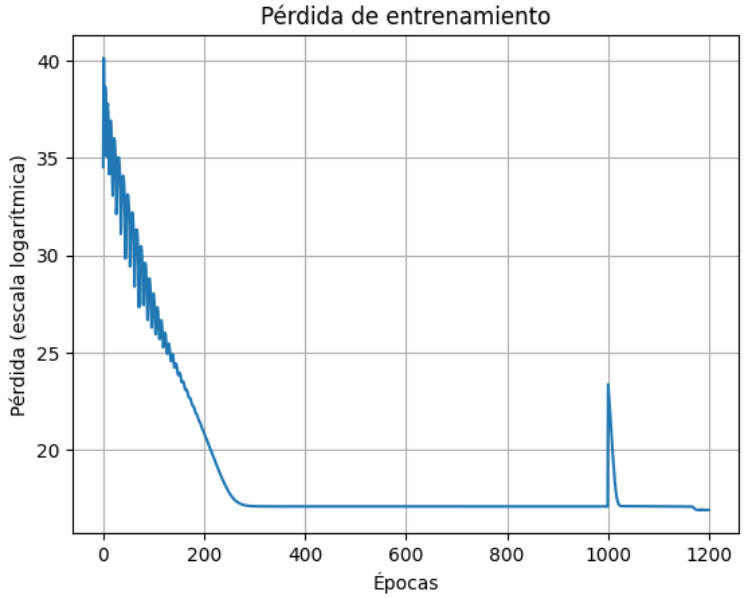


Figura 35: Pérdida de entrenamiento de PINN para viga en voladizo con dimensiones en metros para arquitectura uniforme de red neuronal.

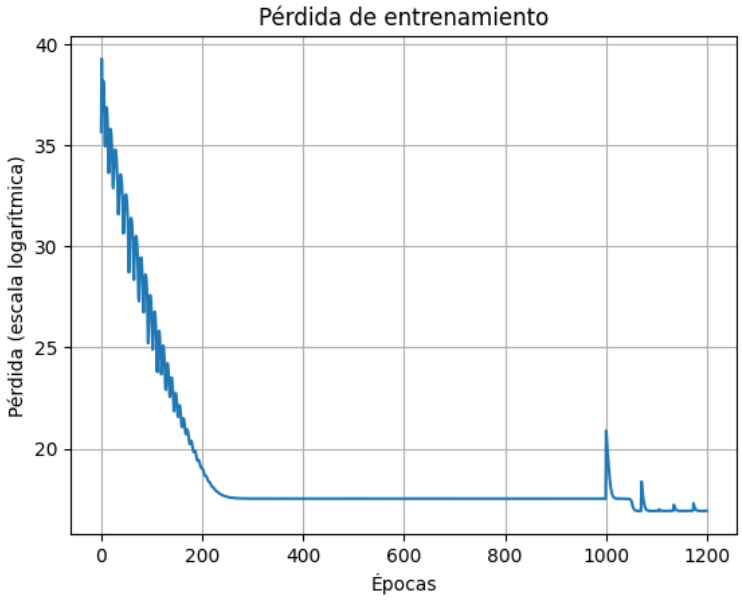


Figura 36: Pérdida de entrenamiento de PINN para viga en voladizo con dimensiones en metros para arquitectura piramidal de red neuronal.

Además, en la Tabla 8 se puede observar una compracaión entre las arquitecturas utilizadas, para el valor final de la pérdida luego de las épocas de entrenamiento y el tiempo que tomó entrenar la red neuronal.

Arquitectura	**Parámetros**	**Épocas**	**Pérdida Final**	**Tiempo (min)**
Uniforme	274,208	1200	2.19e+7	17.73
Piramidal	87,048	1200	2.24e+7	10.54

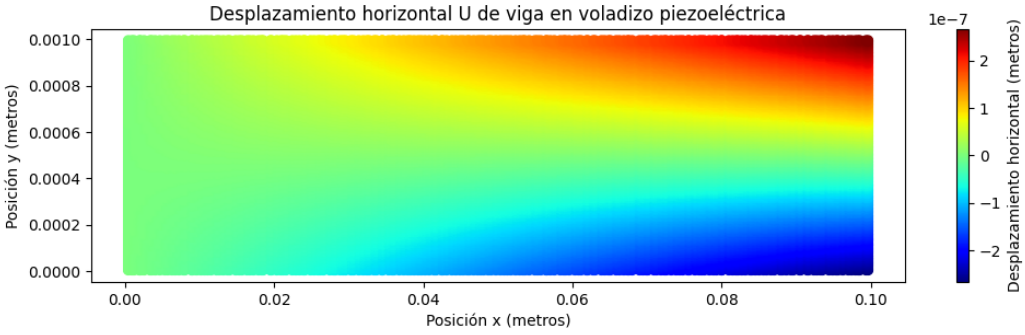
Tabla 8: Resultados de pérdida, tiempo de entrenamiento y cantidad de parámetros para arquitecturas de PINN para efecto piezoeléctrico indirecto.

6.2.2. Resultados de PINN y comparación con FEM para viga en voladizo bajo efecto piezoeléctrico indirecto

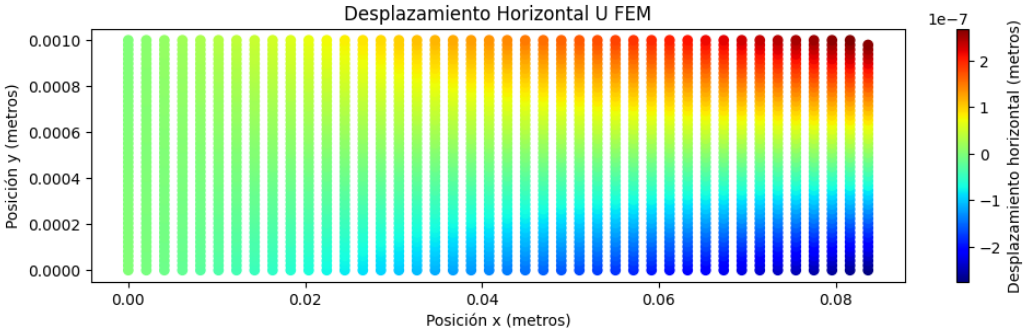
A continuación se muestran los resultados obtenidos por las dos arquitecturas de PINN comparadas, para el efecto piezoeléctrico indirecto.

6.2.2.1. Arquitectura uniforme

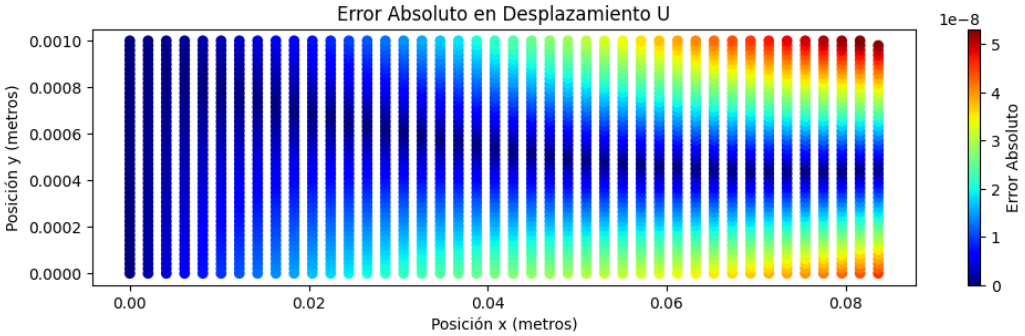
En la Figura 37 se puede observar el desplazamiento horizontal de la viga en voladizo con sus dimensiones en metros para la arquitectura de PINN presentada en la Tabla 6. Este resultado significa que la capa superior tuvo un desplazamiento negativo y la capa inferior un desplazamiento positivo. Esto indica una elongación en la parte inferior y una contracción en la parte superior. En la misma figura se observa el resultado esperado por parte de FEM para el desplazamiento horizontal y el error absoluto obtenido entre PINN y FEM.



(a)



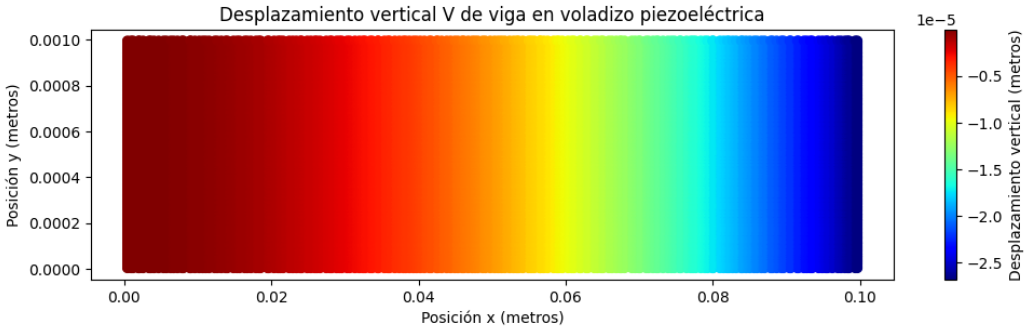
(b)



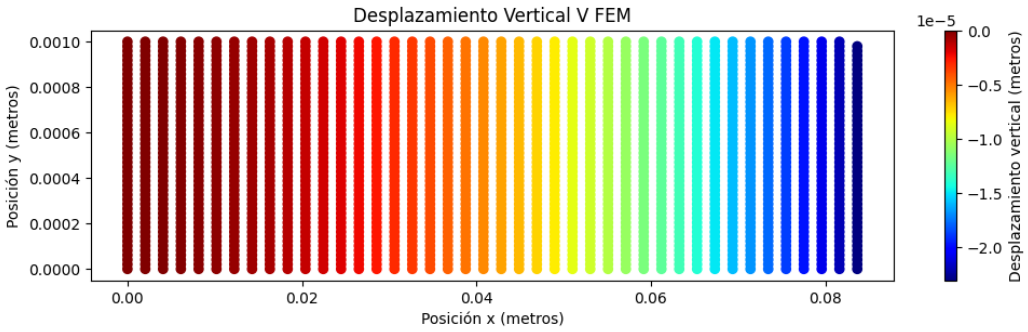
(c)

Figura 37: Resultado para el desplazamiento horizontal obtenido con PINN uniforme. (a) Desplazamiento horizontal (PINN); (b) Desplazamiento horizontal (FEM); (c) Error absoluto entre PINN y FEM.

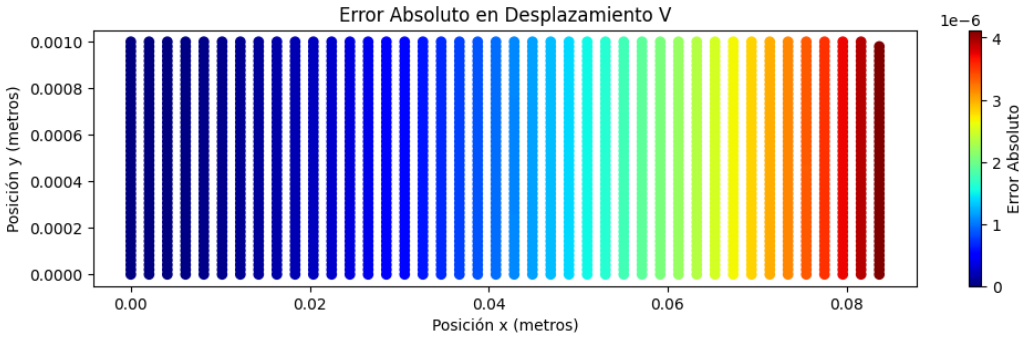
Por otra parte, en la Figura 38 se observa el desplazamiento vertical de la viga modelado por la PINN. Esto indica que la sección cercana al extremo derecho de la viga fue la que tuvo un mayor desplazamiento positivo, mientras que el resto de la viga no se desplazó tanto. Igualmente se muestra el desplazamiento vertical esperado, modelado con FEM y el error absoluto entre PINN y FEM. Lo mismo sucede en la Figura 39 para el potencial eléctrico de la viga.



(a)

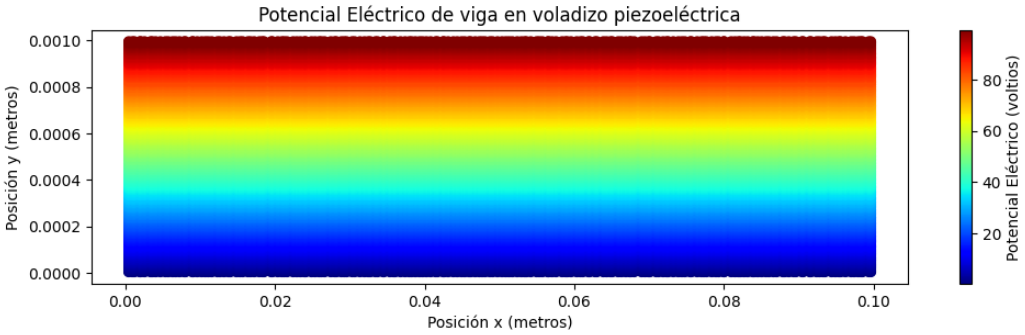


(b)

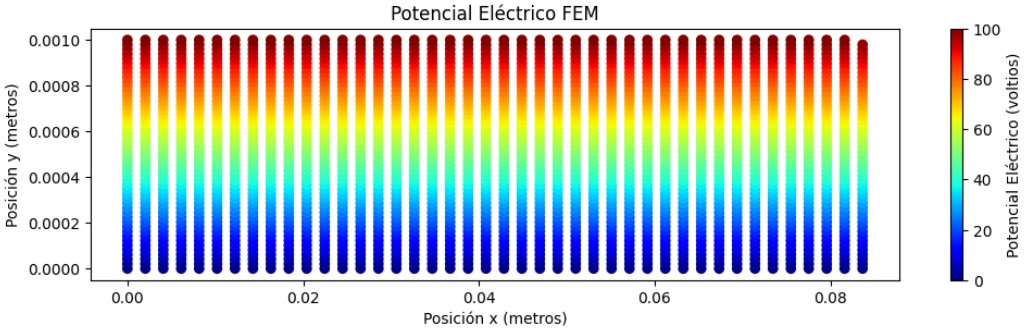


(c)

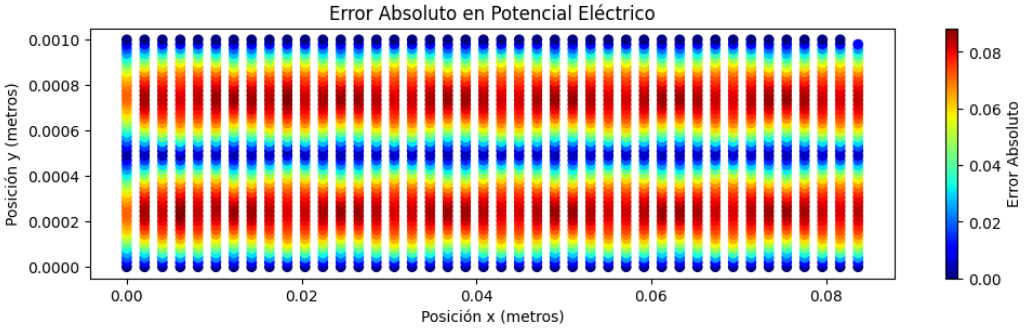
Figura 38: Resultado para el desplazamiento vertical obtenido con PINN uniforme. (a) Desplazamiento vertical (PINN); (b) Desplazamiento vertical (FEM); (c) Error absoluto entre PINN y FEM.



(a)



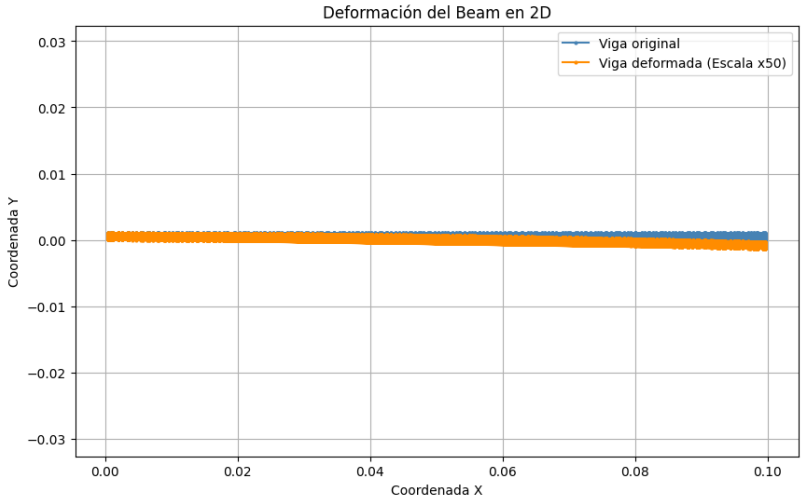
(b)



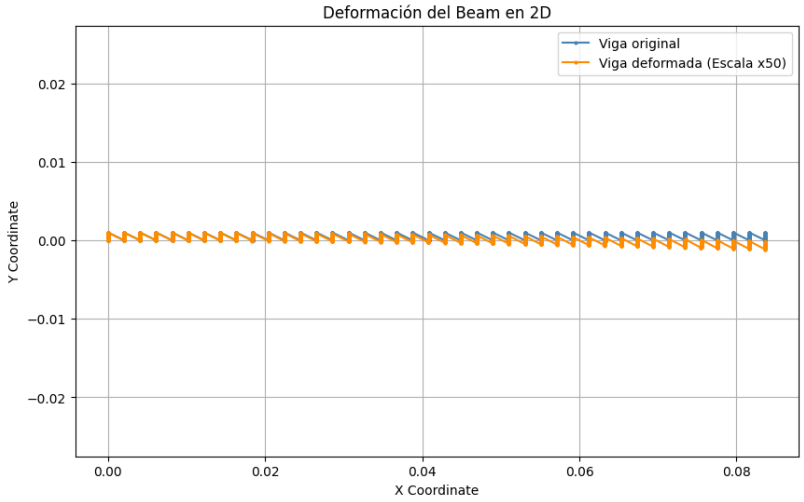
(c)

Figura 39: Resultado para el potencial eléctrico obtenido con PINN uniforme. (a) Potencial eléctrico (PINN); (b) Potencial eléctrico (FEM); (c) Error absoluto entre PINN y FEM.

La Figura 40a muestra cómo se deforma la viga bajo el efecto piezoeléctrico indirecto modelado por la PINN uniforme. En contraste, se muestra la deformación de la viga por FEM en la Figura 40b.



(a)

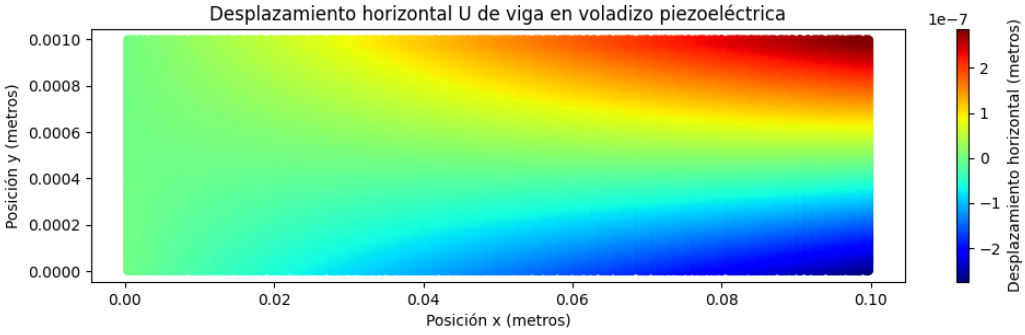


(b)

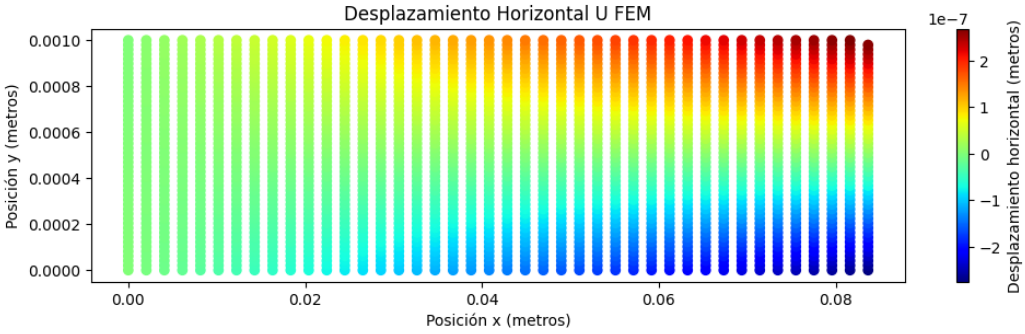
Figura 40: Comparación de la deformación de la viga piezoeléctrica bajo el efecto indirecto modelada con PINN y FEM. (a) Deformación de la viga con PINN; (b) Deformación de la viga con FEM.

6.2.2.2. Arquitectura piramidal

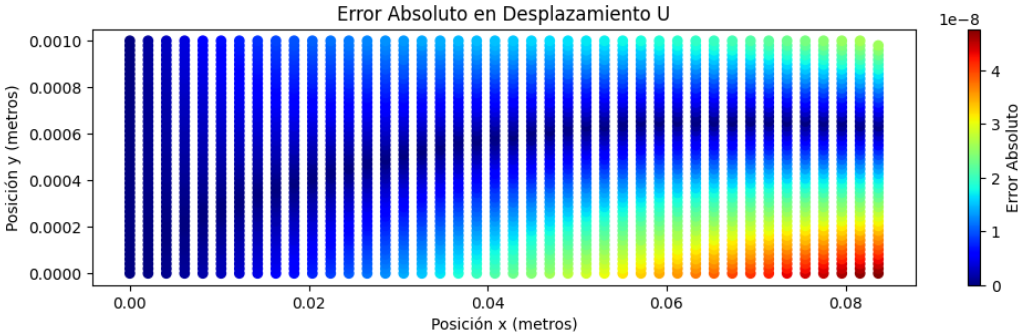
Similar a la arquitectura uniforme, en la Figura 41 se puede observar el resultado del desplazamiento horizontal modelado con la PINN y su comparación con FEM. Lo mismo sucede en la Figura 42 para el desplazamiento vertical y en la Figura 43 para el potencial eléctrico.



(a)

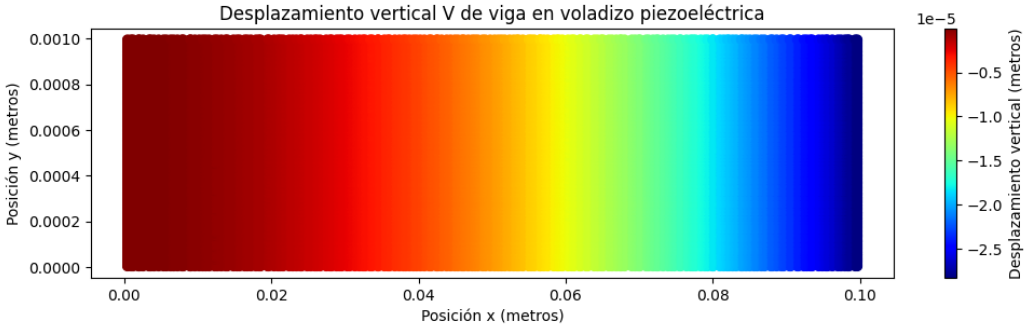


(b)

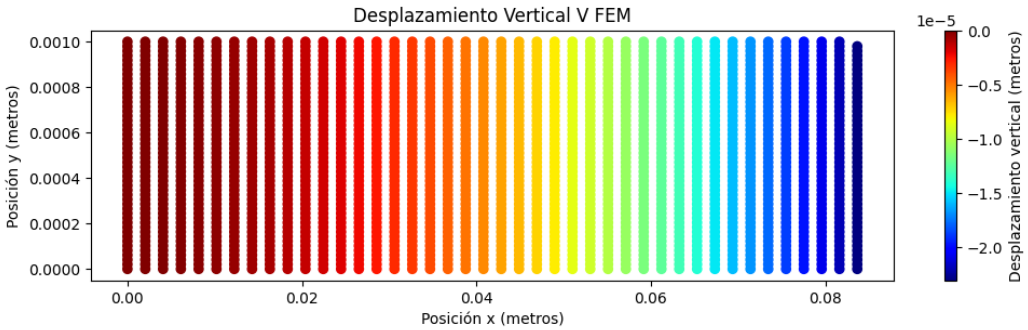


(c)

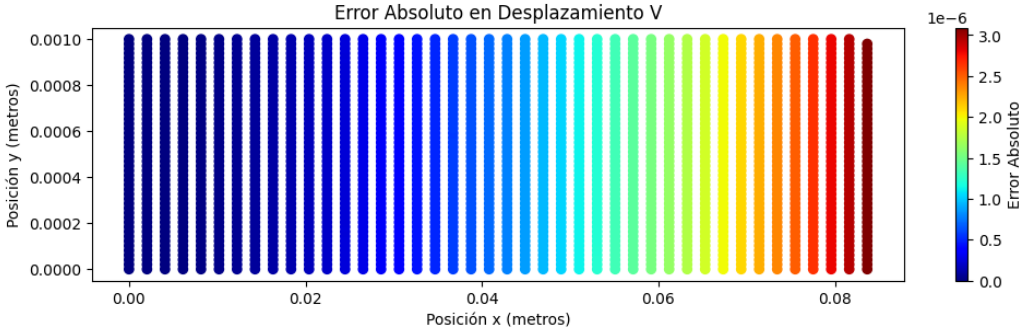
Figura 41: Resultado para el desplazamiento horizontal obtenido con PINN piramidal, y comparación con FEM. (a) Desplazamiento horizontal (PINN); (b) Desplazamiento horizontal (FEM); (c) Error absoluto entre PINN y FEM.



(a)

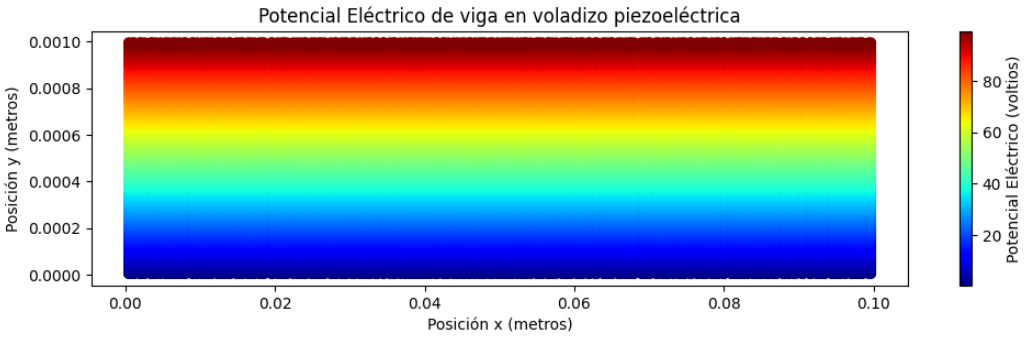


(b)

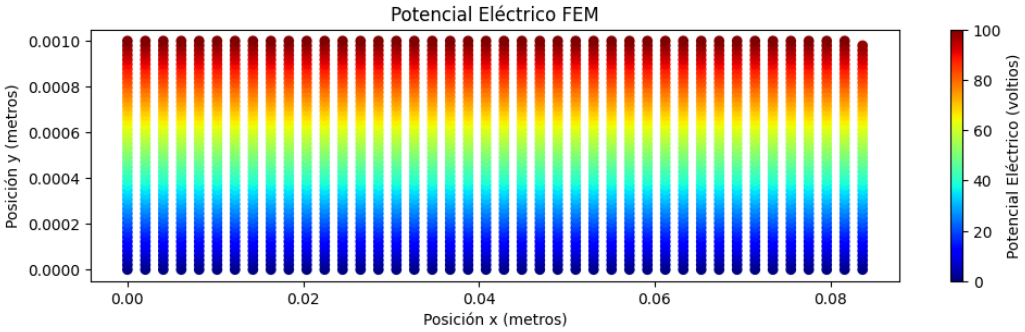


(c)

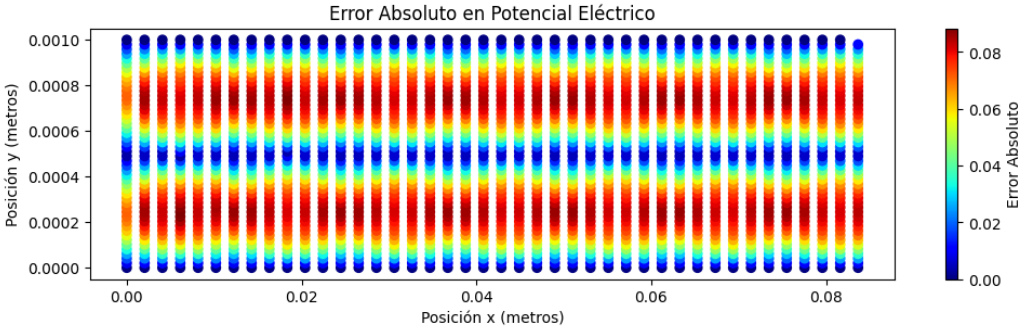
Figura 42: Resultado para el desplazamiento vertical obtenido con PINN piramidal, y comparación con FEM. (a) Desplazamiento vertical (PINN); (b) Desplazamiento vertical (FEM); (c) Error absoluto entre PINN y FEM.



(a)



(b)



(c)

Figura 43: Resultado para el potencial eléctrico obtenido con PINN piramidal, y comparación con FEM. (a) Potencial eléctrico (PINN); (b) Potencial eléctrico (FEM); (c) Error absoluto entre PINN y FEM.

También, la Figura 44a muestra cómo se deforma la viga bajo el efecto piezoeléctrico indirecto modelado por la PINN piramidal y se contrasta con la deformación de la viga por FEM en la Figura 44b.

6.2.3. Resultados de error relativo L2 para arquitecturas de PINN usadas para efecto piezoeléctrico indirecto

En la Tabla 9 se puede observar una comparación de los valores del error relativo L2 entre cada arquitectura de PINN utilizada en relación a los resultados esperados por parte de FEM.

Magnitud	**Norma L2 Arquitectura Uniforme**	**Norma L2 Arquitectura Piramidal**
Desplazamiento horizontal (m)	7.92e-7	6.504e-7
Desplazamiento vertical (m)	8.55e-5	6.389e-5
Potencial eléctrico (V)	2.731	2.731

Tabla 9: Comparación de los valores de la norma L2 entre las predicciones de las arquitecturas de PINN y los resultados de FEM.

6.3. Efecto piezoeléctrico directo

A continuación se presentan los resultados obtenidos para la modelación del efecto piezoeléctrico directo para una viga en voladizo paralela.

6.3.1. Resultados del entrenamiento de red neuronal

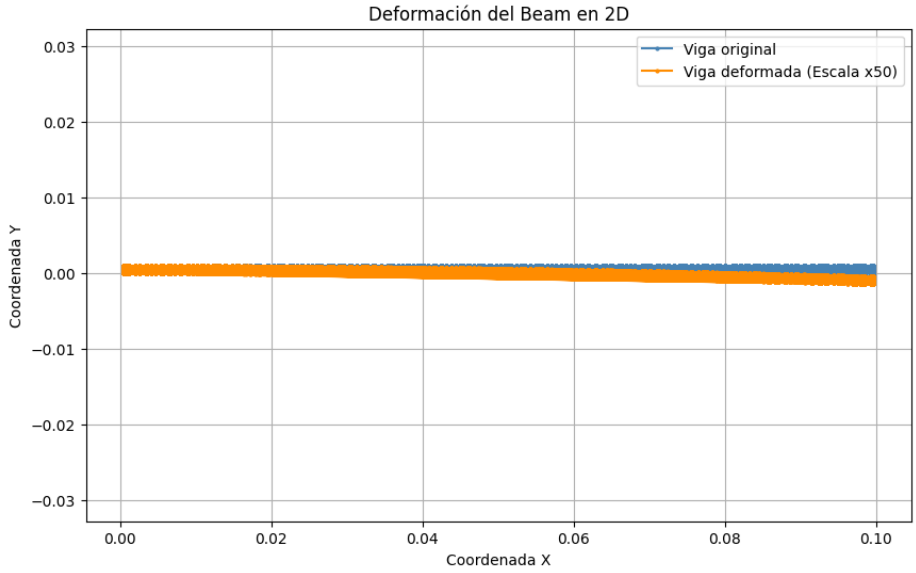
En la Tabla 10 se puede observar la pérdida final obtenida durante el entrenamiento, así como el tiempo que tomó en realizarse. Además, en la Figura 45 se puede observar la pérdida obtenida durante el entrenamiento, para la PINN que modeló el efecto piezoeléctrico directo.

Épocas	**Pérdida Final**	**Tiempo de Entrenamiento (min)**
1200	1.21e+7	9.097

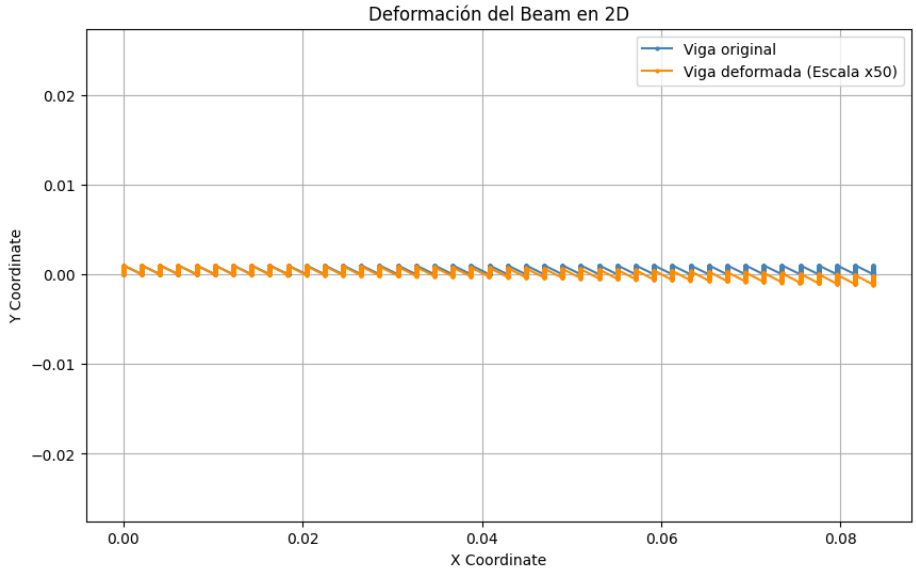
Tabla 10: Resultados de pérdida y tiempo de entrenamiento para PINN para efecto piezoeléctrico directo.

6.3.1.1. Resultados de PINN para efecto piezoeléctrico directo

A continuación se presentan los resultados de la PINN para modelar el efecto piezoeléctrico directo, por medio de una fuerza vertical de 1 N. A diferencia del efecto piezoeléctrico indirecto, acá no se tiene una comparación con FEM.



(a)



(b)

Figura 44: Comparación de la deformación de la viga piezoeléctrica bajo el efecto indirecto modelada con PINN y FEM. (a) Deformación de la viga con PINN; (b) Deformación de la viga con FEM.

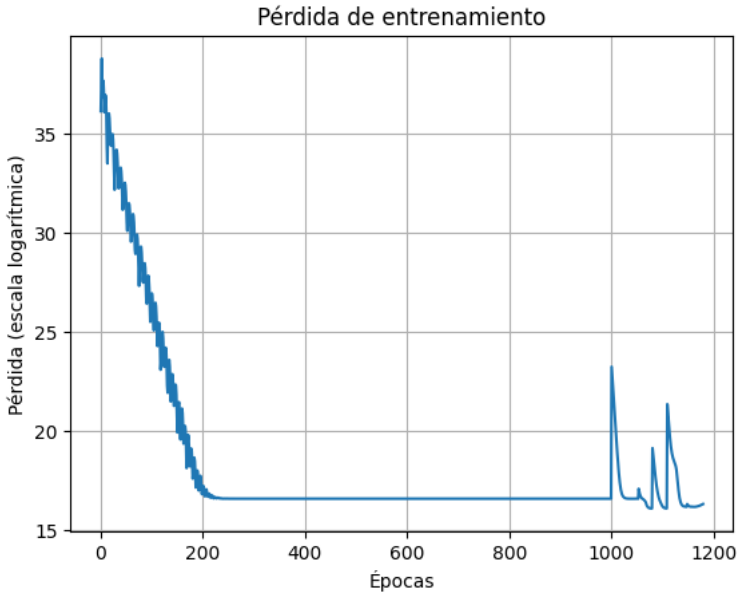


Figura 45: Pérdida de entrenamiento en escala logarítmica de PINN para viga en voladizo con dimensiones en metros bajo efecto piezoeléctrico directo para arquitectura uniforme de red neuronal.

En la Figura 46 se puede observar el desplazamiento horizontal obtenido por la PINN. En este caso, para el desplazamiento horizontal, se genera una contracción de la capa superior y una elongación de la capa inferior. Igualmente en la Figura 47 se puede observar el desplazamiento vertical modelado y en la Figura 48 el potencial eléctrico. Además, en la Figura 49 se observa como se deforma la viga al aplicarle la fuerza mencionada.

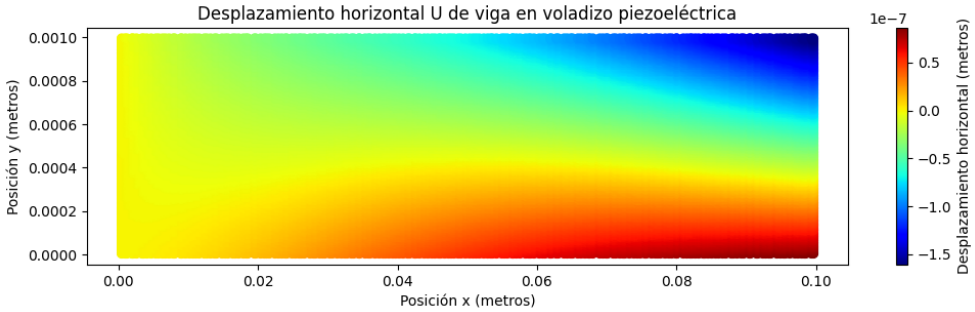


Figura 46: Desplazamiento horizontal de viga en voladizo bajo efecto piezoeléctrico directo modelado por PINN.

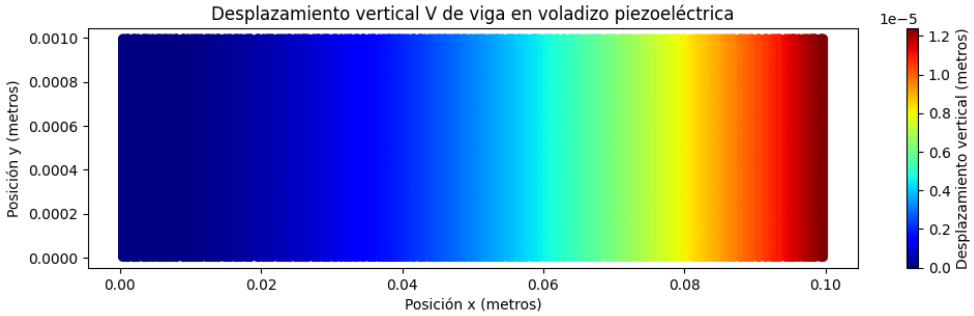


Figura 47: Desplazamiento vertical de viga en voladizo bajo efecto piezoeléctrico directo modelado por PINN.

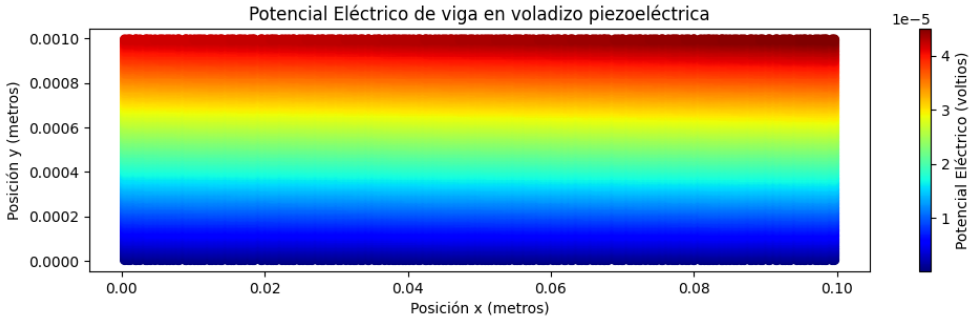


Figura 48: Potencial eléctrico de viga en voladizo bajo efecto piezoeléctrico directo modelado por PINN.

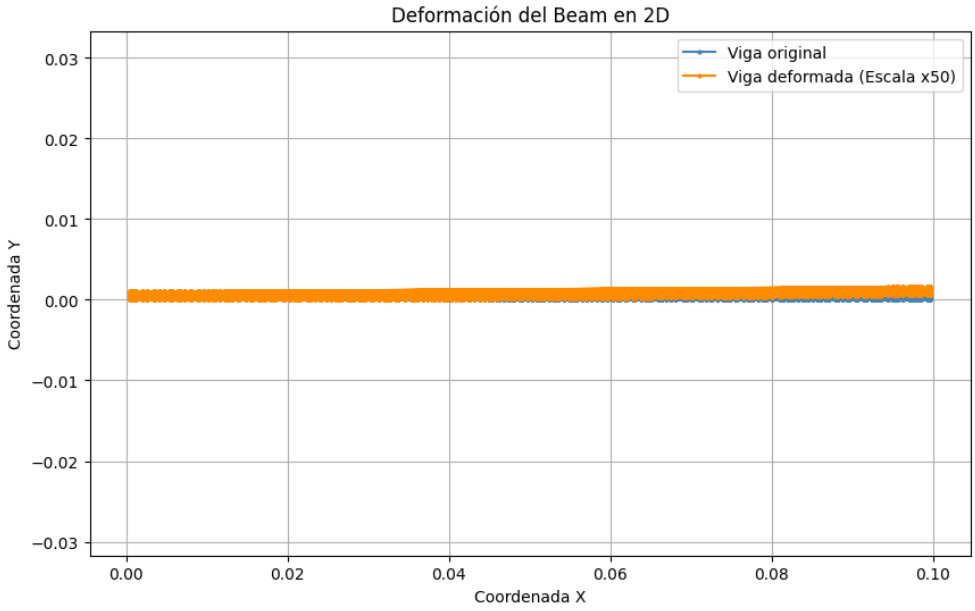


Figura 49: Deformación de viga en voladizo bajo efecto piezoeléctrico directo modelado por PINN.

CAPÍTULO 7

Discusión

La presente investigación tuvo como objetivo principal el desarrollo de una herramienta de simulación que permitiera modelar el efecto piezoeléctrico utilizando *Deep Learning*. Para lograrlo, primero se debieron identificar las bases teóricas que rigen la piezoelectricidad, en forma de un conjunto de ecuaciones diferenciales parciales (PDEs por sus siglas en inglés). Estas ecuaciones fueron integradas en una Red Neuronal Informada por la Física (PINN por sus siglas en inglés). Se implmementaron dos arquitecturas de la PINN con ligeras variaciones, que lograron capturar adecuadamente el efecto piezoeléctrico indirecto. Posteriormente, se usó una de estas arquitecturas para modelar el efecto piezoeléctrico directo. A continuación se evaluarán los resultados obtenidos a partir de las PINNs para modelar el efecto piezoeléctrico.

Al modelar el problema del efecto piezoeléctrico por medio de las arquitecturas de red neuronal en la Subsubsección 5.2.1.4 e integrar las PDEs correspondientes en la función de pérdida en la Subsubsección 5.2.1.5, se logró generar una PINN. En este proceso, se tomó la decisión de modelar las tensiones y desplazamientos eléctricos como salidas de la red neuronal, en lugar de expresarlas en términos de las derivadas de desplazamiento mecánico y potencial eléctrico, como permitiría la Ecuación 9. Esto incrementó la complejidad de la red neuronal por tener más salidas. Sin embargo, eliminó la necesidad de computar segundas derivadas de desplazamiento mecácnico y potencial eléctrico. Esta necesidad hubiera sido causada por la Ecuación 6 y la Ecuación 7. Esta decisión permitió computar solo primeras derivadas de todas las funciones objetivo. Además, permitió aplicar las condiciones de frontera de la Ecuación 51 directamente, sin necesidad de computar derivadas al momento de reforzarlas. Esto hizo más eficiente el entrenamiento de la red neuronal.

De la Figura 36 y de la Figura 35 se pueden observar las funciónes de pérdida para las PINN definidas en la Subsubsección 5.2.1.4 durante el entrenamiento. Se puede evidenciar que el comportamiento de la pérdida de ambas PINN es bastante similar, donde en las primeras 200 épocas se tiene una reducción considerable de la pérdida y luego se estanca

hasta llegar a la época 1000. Acá se da un punto de inflexión causado por el cambio de optimizador a L-BFGS. Esto causa un incremento momentáneo en la pérdida y luego una reducción posterior significativa. Esto puede deberse a que L-BFGS usa más información para el proceso de optimización de la función de pérdida y que usa un *learning rate* mayor que ADAM. Esto último pudo haber permitido que escapara de un mínimo local de la función de pérdida que causaba el estancamiento.

De la Tabla 8 se puede hacer una comparación de las arquitecturas mencionadas en términos de eficiencia. Se puede evidenciar que la arquitectura uniforme tiene una mayor cantidad de parámetros que la arquitectura piramidal. Esto causa que en la misma cantidad de épocas, el tiempo de entrenamiento de la arquitectura uniforme sea considerablemente mayor, que el de la piramidal. Esto se debe a que al momento de hacer la autodiferenciación para cumplir con las PDEs piezoeléctricas y al momento de hacer el *backpropagation* es necesario tomar en cuenta una mayor cantidad de información, reduciendo la eficiencia del entrenamiento.

Con respecto a los resultados de las PINN, en la Figura 37 se puede observar el desplazamiento horizontal modelado por la arquitectura uniforme para la viga en voladizo bajo el efecto piezoeléctrico indirecto y una comparación con los resultados de FEM. Visualmente es posible evidenciar que la forma de la función obtenida, es bastante similar entre ambos métodos. También se evidencia que la escala del desplazamiento es la misma, estando en $10^{-}7$. Además, de la Figura 37c se puede observar la diferencia entre ambos métodos por medio del error absoluto. La predominancia de tonos azules indica que en general el error entre los métodos es bajo. Las mayores diferencias se dan en el desplazamiento de los bordes y esquinas de la viga, pero estas no son tan significativas, debido a que estan en el orden de $10^{-}8$. Esto indica que la red neuronal pudo modelar adecuadamente el desplazamiento horizontal.

En la Figura 38 se puede observar igualmente, la comparación entre el desplazamiento vertical modelado por la PINN uniforme y FEM. Se evidencia que la forma de ambas funciones es bastante similar, igualmente la magnitud del desplazamiento. De la Figura 38c se observa que el error entre la PINN y FEM es bastante bajo, siendo más notoria la diferencia en el extremo derecho de la viga. Este error está en el orden de $10^{-}6$, lo que indica que se pudo modelar adecuadamente el desplazamiento vertical con la PINN.

En cuanto al potencial eléctrico modelado con la PINN uniforme, que se observa en la Figura 39, visualmente es bastante similar al potencial indicado por FEM. Sin embargo, al momento de comparar el error absoluto entre ambos métodos en la Figura 39c, las diferencias son significativas. Se puede observar que la mayor fuente de error se produce en la mitad inferior y en la mitad superior de las capas de la viga. Acá es donde predomina el patrón de color rojo. Esta error por parte de la PINN se puede deber a una imposición errónea de la condición de frontera en la red neuronal con la Ecuación 59. Esta función espera cierto tipo de relación lineal entre la coordenada vertical de la viga y el potencial. Sin embargo, por el error observado este no es el comportamiento adecuado. La PINN uniforme no logró capturar adecuadamente el potencial eléctrico en este caso.

La PINN piramidal modeló mejor el desplazamiento horizontal que la PINN uniforme con respecto a FEM, lo cual se evidencia en la Figura 41c. Acá se observa una mayor área azul, indicando un menor error. Además, la diferencia principal se da en la esquina inferior derecha.

En contraste para la PINN uniforme, las diferencias se dieron en ambas esquinas. La PINN piramidal también modeló adecuadamente el desplazamiento vertical, incluso teniendo un error absoluto ligeramente más bajo que la PINN uniforme, como se aprecia en la Figura 42c. Finalmente, esta arquitectura también tuvo problemas para modelar el potencial eléctrico en la viga, muy similar a la arquitectura uniforme, como se observa en la Figura 43c.

De la Tabla 9, es evidente que la arquitectura piramidal modeló ligeramente mejor el efecto piezoeléctrico indirecto para la viga en voladizo, al tener un error relativo L2 menor para los desplazamientos. Por otra parte, como se mencionó, de la Tabla 8, el tiempo de entrenamiento de la arquitectura piramidal es menor, al igual que la cantidad de parámetros. Esta mejora en el rendimiento de la PINN piramidal, respecto a la uniforme, puede deberse a distintos factores. Por un lado, es una red más profunda, es decir con más capas, lo que le puede permitir capturar una jerarquía de relaciones más compleja al tener una mayor capacidad de representación. Además, expandir y estrechar la cantidad de neuronas, puede ayudar a que el flujo de información de la red fluya más adecuadamente, pudiendo extraer las características clave de las funciones con mayor facilidad. Por otra parte, una menor cantidad de parámetros pudo haber permitido una mejor generalización de las funciones objetivo al dominio del problema. También es posible que permita mayor estabilidad en los gradientes computados, al necesitar menos información.

También de la Tabla 9 se puede observar que ambas arquitecturas tienen valores de error relativo L2 bajos para desplazamiento y potencial eléctrico, con respecto a FEM. Esto apoya las observaciones previamente realizadas con respecto a las funciones resultantes de las arquitecturas evaluadas. De esta forma es posible decir que, independientemente de la distribución de las capas y cantidad de neuronas de la red neuronal, si fue posible implementar una PINN para modelar el efecto piezoeléctrico indirecto.

Un detalle importante por resaltar, es que la pérdida final presentada en la Tabla 8, con un valor en el orden de 10^7 y la pérdida en la Figura 36 y la Figura 35 tienen valores muy altos. Esto se atribuye a la integración de las PDEs en la red neuronal. Específicamente al uso de los coeficientes elásticos del material de la viga, presentados en la Tabla 5 y la Tabla 4. Estos coeficientes se encuentran en el orden de 10^9 y son necesarios para reforzar las PDEs en la pérdida, como se definió en la Subsubsección 5.2.1.5. Entonces, al momento de usar estos valores en conjunto con MSE, el valor de la pérdida se vuelve bastante alto. A pesar de esto, como se evidenció, se logró modelar adecuadamente el efecto piezoeléctrico indirecto, sin necesidad de llevar la pérdida a 0. Por lo tanto, al solventar un problema utilizando una PINN, no siempre es necesario reducir los residuales de las PDEs a 0. Es necesario modelar los problemas, bajo una interpretación en el contexto de las propiedades del sistema o del dominio.

Debido a que la arquitectura piramidal generó mejores resultados para el efecto piezoeléctrico indirecto, se utilizó para modelar el efecto piezoeléctrico directo. En la Figura 45 se puede observar que el comportamiento de la pérdida para el entrenamiento de este modelo fue bastante similar que en el efecto piezoeléctrico indirecto. La Tabla 10 indica que el valor de la pérdida final y el tiempo de entrenamiento también son similares.

En cuanto a los resultados obtenidos para el efecto piezoeléctrico directo, se puede observar de la figura Figura 46 que el desplazamiento horizontal es similar al efecto piezoeléctrico indirecto. En este caso, la contracción en la capa superior y la elongación en la inferior,

corresponden a la fuerza de 1 N aplicada sobre la viga. Esto también causa que la viga suba verticalmente, como se observa en la Figura 49. El resultado más importante de la PINN aplicada al efecto piezoeléctrico directo, es el potencial eléctrico de la Figura 48. Acá se puede observar claramente, que la deformación de la viga generó una diferencia de potencial entre los extremos inferior y superior. Esto corresponde a la generación de electricidad, a partir de una deformación, propio del efecto piezoeléctrico directo.

En conclusión se realizó el análisis de los resultados obtenidos al modelar una viga en voladizo bajo el efecto piezoeléctrico directo e indirecto. Se pudo evaluar y validar la efectividad de la PINN para encontrar las soluciones al problema del efecto piezoeléctrico indirecto, al compararlo con FEM. Se mencionaron las ventajas y desventajas de la arquitectura de PINN implementada. Además, se compararon los resultados obtenidos para el efecto piezoeléctrico indirecto, por dos arquitecturas distintas de PINN. Se pudo evidenciar la arquitectura que obtuvo mejor rendimiento computacional y más precisión en la solución. También se pudo discutir la causa de una pérdida de entrenamiento alta. Finalmente se analizaron los resultados de la aplicación de la PINN al efecto piezoeléctrico directo.

CAPÍTULO 8

Conclusiones

1. Se logró desarrollar con éxito una herramienta de simulación que permitió modelar el efecto piezoeléctrico utilizando *Deep Learning*.

2. Se identificaron las bases teóricas del fenómeno físico piezoeléctrico, lo cuál resultó en la recopilación de un conjunto de ecuaciones diferenciales y constitutivas que permiten describir el acomplamiento electromecánico de dispositivos piezoeléctricos. Esto proporcionó un fundamento matemático necesario para el desarrollo de un modelo computacional.

3. Se logró desarrollar una Red Neuronal Informada por la Física (PINN por sus siglas en inglés), mediante la integración de las ecuaciones diferenciales y constitutivas de la piezoelectricidad, en la arquitectura de la red neuronal. La PINN fue aplicada con éxito al efecto piezoeléctrico directo e indirecto para una viga en voladizo en dos dimensiones.

4. Se optimizó la PINN por medio de una comparación del rendimiento de dos arquitecturas distintas. Se identificó que el uso de una arquitectura con una cantidad variable de neuronas por capa y menos parámetros, obtuvo una mejor precisión y menor tiempo de entrenamiento, en comparación con una arquitectura que tuvo la misma cantidad de neuronas en todas sus capas y una mayor cantidad de parámetros.

5. Se validó la precisión de la PINN mediante su comparación con los resultados obtenidos por el Método de los Elementos Finitos (FEM por sus siglas en inglés). Al comparar el error relativo medio L2, se obtuvieron valores bajos en relación a los desplazamientos mecánicos y el potencial eléctrico modelado. Esto confirma la capacidad de la PINN de modelar adecuadamente el problema piezoeléctrico indirecto para una viga en voladizo en dos dimensiones.

6. Se evidenció que, aunque teóricamente para cumplir con las regulaciones de los problemas modelados por las PINN, es necesario minimizar los residuales de las ecuaciones diferenciales hasta cero, es necesario interpretar los resultados de la pérdida en el contexto del problema. Para la configuración del efecto piezoeléctrico presentada en este

trabajo, la magnitud de los coeficientes elásticos del material modelado no permiten reducir la pérdida a cero. Sin embargo, la validación de los modelos indicó que se pudo solucionar satisfactoriamente el problema.

CAPÍTULO 9

Recomendaciones

1. A pesar de que se logró modelar adecuadamente el efecto piezoeléctrico, los coeficientes de elasticidad jugaron un papel importante en la estabilidad de la pérdida de las redes neuronales, debido a sus valores altos. Se recomienda realizar una adimensionalización de las ecuaciones diferenciales del problema. Esto permitirá expresar las relaciones de las funciones físicas modeladas, sin la influencia de constantes que pueden reducir la eficiencia del aprendizaje de los modelos.

2. Se recomienda realizar pruebas exhaustivas con otros materiales, configuraciones del problema y dispositivos. Esto permitirá validar que la PINN propuesta pueda ser aplicada a diferentes escenarios y entender las limitaciones que puede presentar.

3. Se recomienda tener un conjunto de datos extraído de un método tradicional como FEM, para poder validar el funcionamiento de la PINN aplicada al efecto piezoelectrico directo.

4. Se recomienda extender el estudio a distintas formas de las ecuaciones diferenciales piezoeléctricas. Esto quiere decir, poder evaluarlas con respecto al tiempo o realizar un análisis de frecuencias. Esto permitirá la modelación de distintos dispositivos piezoeléctricos.

CAPÍTULO 10

Bibliografía

[1] P. K. Jakobsen, "An Introduction to Partial Differential Equations," *arXiv (Cornell University)*, ene. de 2019. DOI: 10.48550/arxiv.1901.03022. (visitado 07-05-2024).

[2] B. S. Grewal, *Numerical Methods in Engineering and Science : C, C++, and MATLAB*. Mercury Learning, 2019.

[3] W. K. Liu, S. Li y H. S. Park, "Eighty Years of the Finite Element Method: Birth, Evolution, and Future," *Archives of Computational Methods in Engineering*, vol. 29, págs. 4431-4453, jun. de 2022. DOI: 10.1007/s11831-022-09740-9.

[4] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi y F. Piccialli, "Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next," *Journal of Scientific Computing*, vol. 92, jul. de 2022. DOI: 10.1007/s10915-022-01939-z.

[5] Y. LeCun, Y. Bengio y G. Hinton, "Deep Learning," *Nature*, vol. 521, págs. 436-444, mayo de 2015. DOI: 10.1038/nature14539.

[6] M. Raissi, P. Perdikaris y G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, págs. 686-707, feb. de 2019. DOI: 10.1016/j.jcp.2018.10.045.

[7] T. Lahmer, "Forward and Inverse Problems in Piezoelectricity," Tesis doct., mayo de 2008. dirección: https://d-nb.info/989762831/34.

[8] T. L. Szabo y P. A. Lewin, "Piezoelectric Materials for Imaging," *Journal of Ultrasound in Medicine*, vol. 26, págs. 283-288, mar. de 2007. DOI: 10.7863/jum.2007.26.3.283.

[9] L. Walubita, D. Sohoulande Djebou, A. Faruk, S. Lee, S. Dessouky y X. Hu, "Prospective of Societal and Environmental Benefits of Piezoelectric Technology in Road Energy Harvesting," *Sustainability*, vol. 10, pág. 383, feb. de 2018. DOI: 10.3390/su10020383.

[10] J. Song, G. Sun, X. Zeng, X. Li, Q. Bai y X. Zheng, "Piezoelectric energy harvester with double cantilever beam undergoing coupled bending-torsion vibrations by width-splitting method," *Scientific Reports*, vol. 12, pág. 583, ene. de 2022. DOI: 10.1038/s41598-021-04476-1. dirección: https://www.nature.com/articles/s41598-021-04476-1 (visitado 14-12-2022).

[11] Y. Pinchover y J. Rubinstein, *AN INTRODUCTION TO PARTIAL DIFFERENTIAL EQUATIONS*. Cambridge University Press, 2005.

[12] X. Gao, J. Yang, J. Wu et al., "Piezoelectric Actuators and Motors: Materials, Designs, and Applications," *Advanced Materials Technologies*, vol. 5, pág. 1 900 716, nov. de 2019. DOI: 10.1002/admt.201900716.

[13] S. Zhang y F. Yu, "Piezoelectric Materials for High Temperature Sensors," *Journal of the American Ceramic Society*, vol. 94, D. J. Green, ed., págs. 3153-3170, ago. de 2011. DOI: 10.1111/j.1551-2916.2011.04792.x.

[14] M. T. Chorsi, E. J. Curry, H. T. Chorsi et al., "Piezoelectric Biomaterials for Sensors and Actuators," *Advanced Materials*, vol. 31, pág. 1 802 084, oct. de 2018. DOI: 10.1002/adma.201802084.

[15] N. Sezer y M. Koç, "A comprehensive review on the state-of-the-art of piezoelectric energy harvesting," *Nano Energy*, vol. 80, pág. 105 567, feb. de 2021. DOI: 10.1016/j.nanoen.2020.105567.

[16] Z. Hao, S. Liu, Y. Zhang et al., *Physics-Informed Machine Learning: A Survey on Problems, Methods and Applications*, arXiv.org, mar. de 2023. DOI: 10.48550/arXiv.2211.08064. dirección: https://arxiv.org/abs/2211.08064 (visitado 15-12-2023).

[17] J. Stewart, *Multivariable calculus*. Brooks/Cole, 2012.

[18] D. G. Zill, *A First Course in Differential Equations with Modeling Applications*. Cengage, 1997.

[19] H. D. Young, P. W. Adams y R. J. Chastain, *Sears Zemansky's college physics. Volume 1*. Pearson, 2016.

[20] W. E. Boyce, R. C. Diprima y A. Firm, *Elementary differential equations and boundary value problems*. Wiley, 2009.

[21] M. T. Heath, *Scientific computing : an introductory survey*. Philadelphia Society For Industrial y Applied Mathematics, 2018.

[22] P. Blanchard, R. L. Devaney y G. R. Hall, *Differential Equations*. Cengage Learning, jul. de 2012.

[23] V. Piefort, "Finite Element Modelling of Piezoelectric Active Structures," Tesis doct., abr. de 2001.

[24] W. P. Mason, "Piezoelectricity, its history and applications," *The Journal of the Acoustical Society of America*, vol. 70, págs. 1561-1566, dic. de 1981. DOI: 10.1121/1.387221.

[25] I. PIEZO SYSTEMS, *History of Piezoelectricity*, PIEZO.COM, 2007. dirección: https://piezo.com/pages/history-of-piezoelectricity.

[26] R. Chang y K. A. Goldsby, *Chemistry*. Mcgraw-Hill Education, 2014.

[27] A. Safari y E. K. Akdoğan, *Piezoelectric and Acoustic Materials for Transducer Applications*. New York Springer, 2008.

[28] J. Tichý, J. Erhart, E. Kittinger y J. Přívratská, *Fundamentals of Piezoelectric Sensorics*. Berlin, Heidelberg Springer Berlin Heidelberg, 2010.

[29] K.-C. Lin, C.-H. Hu y K.-C. Wang, "Innovative deep energy method for piezoelectricity problems," *Applied mathematical modelling*, vol. 126, págs. 405-419, feb. de 2024. DOI: 10.1016/j.apm.2023.11.006. (visitado 02-08-2024).

[30] P. Kelly, *Solid Mechanics Part II: Engineering Solid Mechanics*, 2013.

[31] S. J. Russell y P. Norvig, *Artificial intelligence : a modern approach*. Pearson India Education Services Pvt. Ltd, 2018.

[32] M. Tamir, *What Is Machine Learning? - I School Online*, Berkeley School of Information, jun. de 2020. dirección: https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/.

[33] A. Burkov, *THE HUNDRED-PAGE MACHINE LEARNING BOOK*. Andriy Burkov, 2019.

[34] I. Goodfellow, Y. Bengio y A. Courville, *Deep Learning*. The Mit Press, 2016.

[35] G. Delanerolle, X. Yang, S. Shetty et al., "Artificial intelligence: A rapid case for advancement in the personalization of Gynaecology/Obstetric and Mental Health care," *Women's Health*, vol. 17, pág. 174 550 652 110 181, ene. de 2021. DOI: 10.1177/17455065211018111.

[36] S. Feuerriegel, J. Hartmann, C. Janiesch y P. Zschech, "Generative AI," *Business Information Systems Engineering*, vol. 66, págs. 111-126, sep. de 2023. DOI: 10.1007/s12599-023-00834-7. dirección: https://link.springer.com/article/10.1007/s12599-023-00834-7.

[37] P. Baheti, *12 Types of Neural Networks Activation Functions: How to Choose?* www.v7labs.com, mar. de 2022. dirección: https://www.v7labs.com/blog/neural-networks-activation-functions.

[38] J. Stewart, *Single Variable Calculus*. Brooks Cole, jun. de 2015.

[39] S. Kostadinov, *Understanding Backpropagation Algorithm*, Medium, ago. de 2019. dirección: https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd.

[40] S. Doshi, *Various Optimization Algorithms For Training Neural Network*, Medium, ago. de 2020. dirección: https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6.

[41] Taorui$_W$*ang*, *L-BFGS algorithm*, Medium, abr. de 2022. dirección: https://medium.com/@tru11631/l-bfgs-algorithm-ecfb832a4176.

[42] A. G. Baydin, B. A. Pearlmutter, A. A. Radul y J. M. Siskind, *Automatic differentiation in machine learning: a survey*, arXiv.org, feb. de 2018. DOI: 10.48550/arXiv.1502.05767. dirección: https://arxiv.org/abs/1502.05767 (visitado 14-06-2024).

[43] A. Holmes, *What's Automatic Differentiation?* huggingface.co, mar. de 2024. dirección: https://huggingface.co/blog/andmholm/what-is-automatic-differentiation (visitado 03-08-2024).

[44] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang y L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, vol. 3, págs. 422-440, jun. de 2021. DOI: 10.1038/s42254-021-00314-5. dirección: https://www.nature.com/articles/s42254-021-00314-5.

[45] S. Wang, S. Sankaran, H. Wang y P. Perdikaris, *An Expert's Guide to Training Physics-informed Neural Networks*, arXiv.org, ago. de 2023. DOI: 10.48550/arXiv.2308.08468. dirección: https://arxiv.org/abs/2308.08468 (visitado 10-12-2023).

[46] Z. Wang y M. Hernández Salinas, *PINNs Introductory Code for the Heat Equation*, Nat.fau.eu, dic. de 2023. dirección: https://dcn.nat.fau.eu/pinns-introductory-code-for-the-heat-equation/ (visitado 15-08-2024).

[47] A. Paszke, S. Gross, F. Massa et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," en *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, págs. 8024-8035. dirección: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[48] C. R. Harris, K. J. Millman, S. J. van der Walt et al., "Array programming with NumPy," *Nature*, vol. 585, n.º 7825, págs. 357-362, sep. de 2020. DOI: 10.1038/s41586-020-2649-2. dirección: https://doi.org/10.1038/s41586-020-2649-2.

[49] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, n.º 3, págs. 90-95, 2007. DOI: 10.1109/MCSE.2007.55.

[50] L. Lu, X. Meng, Z. Mao y G. E. Karniadakis, "DeepXDE: A deep learning library for solving differential equations," *SIAM Review*, vol. 63, n.º 1, págs. 208-228, 2021. DOI: 10.1137/19M1274067.

[51] X. Jin, S. Cai, H. Li y G. E. Karniadakis, "NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations," *Journal of Computational Physics*, pág. 109 951, nov. de 2020. DOI: 10.1016/j.jcp.2020.109951.

[52] E. Samaniego, C. Anitescu, S. Goswami et al., "An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications," *Computer Methods in Applied Mechanics and Engineering*, vol. 362, pág. 112 790, abr. de 2020. DOI: 10.1016/j.cma.2019.112790. (visitado 24-11-2020).

[53] A. Chen, A. Chow, A. Davidson et al., "Developments in MLflow," *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning*, jun. de 2020. DOI: 10.1145/3399579.3399867.

[54] Q. Fang, X. Mou y S. Li, "A physics-informed neural network based on mixed data sampling for solving modified diffusion equations," *Scientific reports*, vol. 13, feb. de 2023. DOI: 10.1038/s41598-023-29822-3.

[55] C. Rao, H. Sun e Y. Liu, "Physics-Informed Deep Learning for Computational Elastodynamics without Labeled Data," *Journal of Engineering Mechanics*, vol. 147, pág. 04 021 043, ago. de 2021. DOI: 10.1061/(asce)em.1943-7889.0001947. (visitado 15-06-2021).

[56] A. Sholokhov, Y. Liu, H. Mansour y S. Nabi, "Physics-informed neural ODE (PINODE): embedding physics into models using collocation points," *Scientific Reports*, vol. 13, pág. 10 166, jun. de 2023. DOI: 10.1038/s41598-023-36799-6. dirección: https://www.nature.com/articles/s41598-023-36799-6 (visitado 30-04-2024).

[57] Mathworks, *Deflection of Piezoelectric Actuator*, Mathworks.com, 2024. dirección: https://es.mathworks.com/help/pde/ug/deflection-of-a-piezoelectric-actuator.html (visitado 30-08-2024).

CAPÍTULO 11

Anexos

Implementación de PINNs para metodología y piezoelectricidad

Se puede acceder al código realizado para solucionar los distintos problemas presentados en este proyecto en el siguiente repositorio: https://github.com/Daniel14gonc/piezonet.git.