Una exploración de las Redes Neuronales Físicamente Informadas (PINN)

Sebastian Gaviria Giraldo*

*sebastian.gaviriag@udea.edu.co

ABSTRACT

Ante el desafío de explorar problemas complejos de la física (o de los sistemas dinámicos en general) el Aprendizaje Estadístico entra en juego con el Machine Learning Físicamente Informado. Es a partir de esta visión más general que surgen las PINN, como un método para agregar un sesgo físico a una red neuronal que es entrenada para aproximar una función en específico, mediante sus ecuaciones dinámicas, y condiciones iniciales y de frontera.

El presente estudio busca entender los fundamentos teóricos, y aplicar las PINNs a problemas directos relacionados con la ecuación del calor.

1. Introducción

Actualmente disponemos de ingentes volúmenes de observaciones —por ejemplo, sensores terrestres y satelitales— pero las herramientas clásicas de simulación basada en PDEs (elementos finitos, diferencias finitas, métodos espectrales) naufragan ante sistemas no lineales multiescala, con incertidumbres, geometrías complejas o condiciones de frontera ruidosas.

Ante este panorama, la solución propuesta por diversos académicos es "enseñar" a las redes neuronales las leyes físicas que rigen el sistema, proporcionando un sesgo inductivo que refuerce la interpolación y extienda la extrapolación de los modelos ML más allá de meros datos.

Como ejemplo básico de aplicación de las PINN, podemos obtener la función solución de la ecuación del calor, con unas condiciones específicas. Además, podemos extraer información oculta de datos experimentales. Sin embargo, antes de entrar de lleno con las PINN, entendamos un poco más qué es esto del PIML.

2. Physics Informed Machine Learning (PIML)

El aprendizaje automático informado por la física, combina la solidez de las ecuaciones que describen la naturaleza con la flexibilidad de las redes neuronales profundas. Su objetivo no es sustituir los métodos clásicos de simulación, sino potenciarlos: aprovechar tanto los datos como el conocimiento físico para entrenar modelos que requieran menos ejemplos y ofrezcan predicciones más fiables, incluso fuera del rango observado.

Para ello, PIML recurre a tres "palancas" que inyectan información física en el proceso de aprendizaje:

- **Datos con estructura física (Observational Biases):** Al entrenar con resultados de simulaciones o mediciones que ya cumplen leyes de conservación o simetrías (por ejemplo, flujos de fluidos o difusión térmica), la red aprende patrones que serían muy costosos de descubrir solo a partir de datos crudos. Esto reduce la necesidad de grandes volúmenes de muestras y mejora la calidad de la interpolación.
- **Arquitecturas alineadas con la física (Inductive Biases):** Algunas propiedades pueden 'programarse' directamente en el diseño de la red. Por ejemplo, las convoluciones garantizan la invarianza traslacional; también están las Redes sobre grafos que respetan la permutación de nodos (empleadas en mallas de elementos finitos o estructuras moleculares); Las capas específicas (como en FermiNet) que imponen antisimetría en funciones de onda cuántica.
- **Ecuaciones en la función de pérdida (Learning Biases):** Aquí, la red minimiza al mismo tiempo el error frente a los datos y el residual de la ecuación diferencial que rige el fenómeno (por ejemplo, la ecuación de calor o Navier–Stokes). Si la red propone una solución que no satisface la ley física, ese fallo se traduce en un aumento del "castigo" en la pérdida, obligándola a corregirse en cada iteración.

Se podría decir que los tres tipos de enfoques que se pueden plantear en el aprendizaje estadístico físicamente informado se basan en añadir el conocimiento físico de tres formas diferentes: añadirlos a los datos de entrenamiento directamente, añadirlos a la arquitectura de la red misma, o añadirlos durante el proceso de entrenamiento vía función de pérdida; respectivamente.

Mi estudio se centrará en el último método de añadir conocimiento físico a modelos de aprendizaje estadístico, haciendo que la red ya montada aprenda a ajustarse a la ecuación diferencial.

2.1. Physics Informed Neural Networks (PINN)

Vamos a discutir un poco más en profundidad el sesgo por aprendizaje, que es el método que vio nacer las PINN. La base fundamental de las PINN es la conocida propiedad de aproximador universal de funciones que tienen las redes neuronales en general. Con esto, y asumiendo la ecuación dinámica de un sistema como una ecuación diferencial parametrizada y no lineal, se puede hacer todo el trabajo.

$$\mathscr{F} = \left\{ x \mapsto \sum_{j=1}^{N} \alpha_j \sigma(w_j^T x + b_j) \;\middle|\; N \in \mathbb{N},\ \alpha_j \in \mathbb{R},\ w_j \in \mathbb{R}^n,\ b_j \in \mathbb{R} \right\}$$

$$\frac{\partial u(x,t)}{\partial t} + \mathscr{N}[u(x,t);\lambda] = 0, \quad x \in \Omega, \quad t \in [0,T]$$

Para entender bien el concepto, nos podemos imaginar a estas NN como una red cualquiera, pero cuyo resultado no se limita únicamente a un output entrenado, sino que a ese output se le aplican una serie de operadores diferenciales (en un método que se llama diferenciación automática o auto-diferenciación). Posteriormente, a partir del output transformado, se calcula una diferencia que representa la ecuación diferencial, y cuyo resultado se añade a la función de pérdida total, ajustando aun más el output. La figura muestra un diagrama clásico de la arquitectura típica de una PINN.

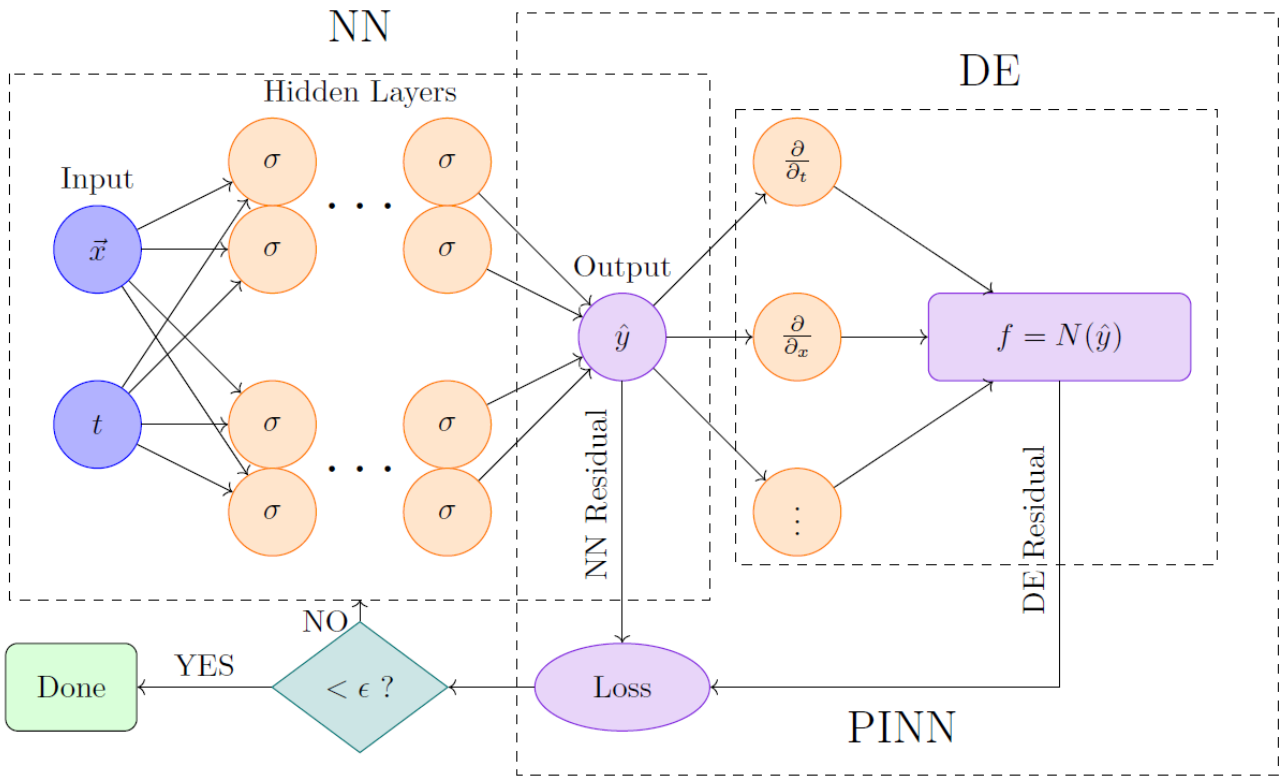

Figure 1. Arquitectura de una FNN convertida en PINN

Al final, la función de coste total de la PINN quedaría algo tal que así:

$$\mathscr{L}(\theta) = \lambda_{data}\mathscr{L}_{data} + \lambda_{IC}\mathscr{L}_{IC} + \lambda_{BC}\mathscr{L}_{BC} + \lambda_f \mathscr{L}_f$$

con

$$\mathscr{L}_{\text{data}} = \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} \left| u_\theta(x_i,t_i) - u^{\text{obs}}(x_i,t_i) \right|^2 \tag{1}$$

$$\mathscr{L}_{\text{IC}} = \frac{1}{N_{\text{IC}}} \sum_{i=1}^{N_{\text{IC}}} |u_\theta(x_i,0) - u_0(x_i)|^2 \tag{2}$$

$$\mathscr{L}_{\text{BC}} = \frac{1}{N_{\text{BC}}} \sum_{i=1}^{N_{\text{BC}}} |u_\theta(x_i,t_i) - g(x_i,t_i)|^2 \tag{3}$$

$$\mathscr{L}_f = \frac{1}{N_f} \sum_{i=1}^{N_f} \left| \frac{\partial u_\theta}{\partial t}(x_i,t_i) + \mathscr{N}[u_\theta(x_i,t_i);\lambda] \right|^2 \tag{4}$$

Entonces, de esta manera, aprovechando el hecho de que se cuenta con cierta arbitrariedad a la hora de establecer una función de coste en cualquier red neuronal, se puede definir una, utilizando la física para corregir la red en cada paso.

3. Aplicaciones de las PINN

Mi proyecto se centró en dos aplicaciones concretas, resolver la ecuación de difusión del calor, y resolver la ecuación de la aleta (relacionada con los disipadores de calor), obteniendo resultados interesantes.

3.1. Ecuación de difusión del calor

Pensemos por ejemplo, en la ecuación que modela la propagación del calor en el espacio y a lo largo del tiempo. Este sería un ejemplo perfecto para observar el uso de las PINN, pues esta es una ecuación diferencial en derivadas parciales dependiente del tiempo. La forma de esta ecuación es:

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u + \frac{Q}{\rho c_p}, \qquad \text{Con generación interna de calor (fuentes)} \tag{5}$$

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u, \qquad \text{Sin fuentes} \tag{6}$$

Donde $u = u(\vec{r},t)$ siendo el campo de temperaturas. Las funciones solución de esta ecuación son conocidas como funciones calóricas. Si pensamos en este campo de temperaturas como una hiper-superficie mapeada en el dominio, podemos extender la ecuación del calor a diferentes situaciones, con geometrías planas, e incluso esféricas. También con dominios de geometrías irregulares como los disipadores de calor y las aletas de transferencia de calor.

En esta parte, mi enfoque se basó en resolver la ecuación del calor sin fuentes, para una geometría unidimensional (barra de grosor infinitesimal) y una geometría bidimensional (placa rectangular de grosor infinitesimal). Es decir, resolver la ecuación del calor 1D y 2D

3.2. Ecuación de la aleta (o ecuación de convección)

Por otro lado, está la llamada ecuación de la aleta (o del disipador) que modela un proceso térmico en el que se busca propagar el calor de forma estratégica hacia el ambiente (disipación). Esta ecuación es similar a la ecuación general del calor, pero se le añade un término de decaimiento. Además, las condiciones de frontera que se imponen a este problema son del tercer tipo, es decir, se combinan restricciones sobre la solución misma, y sobre su derivada normal. Estas condiciones de frontera se llaman condiciones de Robin o de Convección.

$$\frac{\partial u}{\partial t} = a\nabla^2 u - \beta u, \qquad \text{Ecuación del disipador con decaimiento térmico} \tag{7}$$

$$\frac{h}{k}u + \frac{\partial u}{\partial n} = 0, \qquad \text{Condiciones de frontera de Robin (convección en aletas)} \tag{8}$$

En esta parte, me enfoqué en resolver la ecuación de la aleta dependiente del tiempo para un disipador de calor con base rectangular y aletas también rectangulares.

Aquí se nota el gran poder que tienen las PINN para resolver EDP en geometrías complejas.

4. Métodos

Para implementar la solución a cada uno de los problemas, usé la librería DeepXDE para la construcción de PINNs. Con ella, construí un pipeline completo de adecuación geométrica del problema, imposición de condiciones iniciales y de frontera, generación de datos sintéticos, entrenamiento y visualización de resultados.

De esta librería utilicé directamente los módulos `geometry` y con sus métodos construí una clase llamada `Geometry()` que me sirvió para todas las adecuaciones geométricas de los problemas. También usé el módulo `icbc`, y construí la clase `ICBC()` para la imposición de condiciones iniciales y de frontera en los datos. El módulo `data` de la librería, me sirvió para construir la clase `loadData()`, que genera los datos sintéticos para el entrenamiento de la PINN. Por otro lado, construí una clase llamada `PINN()`, con el uso de los módulos `dde.nn` y `dde.Model`. Esta clase me sirvió para construir, compilar y entrenar la red neuronal físicamente informada y guardarle como un modelo que resuelve ecuaciones diferenciales.

Por último, creé dos clases más para todo el proceso de graficación y visualización. `Makegrid()` para la creación de mallas de dominio adecuadas para matplotlib, y la clase `Ploter()` que se encarga de todas las graficaciones en sí.

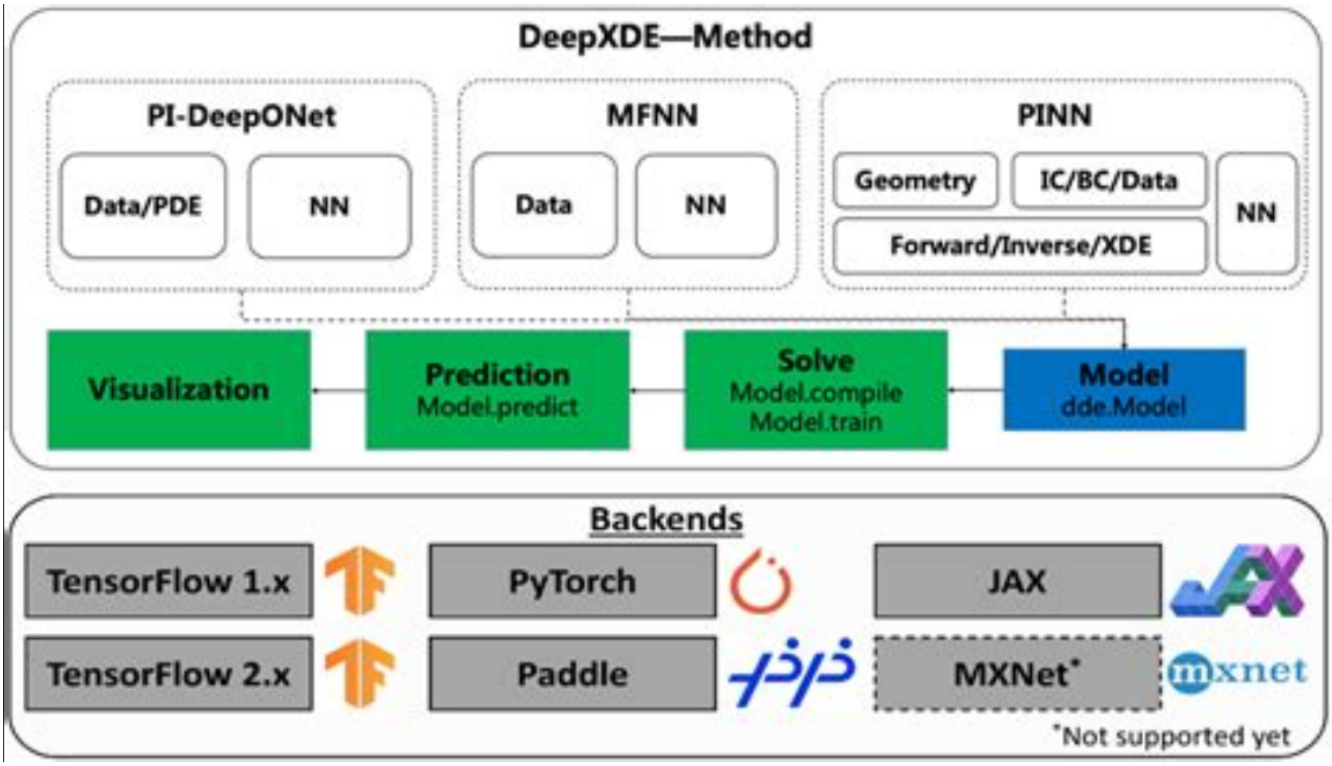

Figure 2. Librería DeepXDE para la construcción y uso de PINN

La red construida fue una FeedForward con una capa de entrada con el número de variables del dominio de la EDP, tiene 90 capas ocultas, y una capa de salida con un solo output que representa la solución de la ecuación diferencial. Se logró una convergencia óptima del entrenamiento en una cantidad de 2000 épocas. Se utilizaron la función de activación tanh y el inicializador `Glorot normal`, y como optimizador se utilizó `Adam` y refinamiento con `L-BFGS-B` en la compilación. El learning rate usado fue de 0.001. En la figura 3 se ve un diagrama completo de la lógica empleada en el diseño de la resolución del problema.

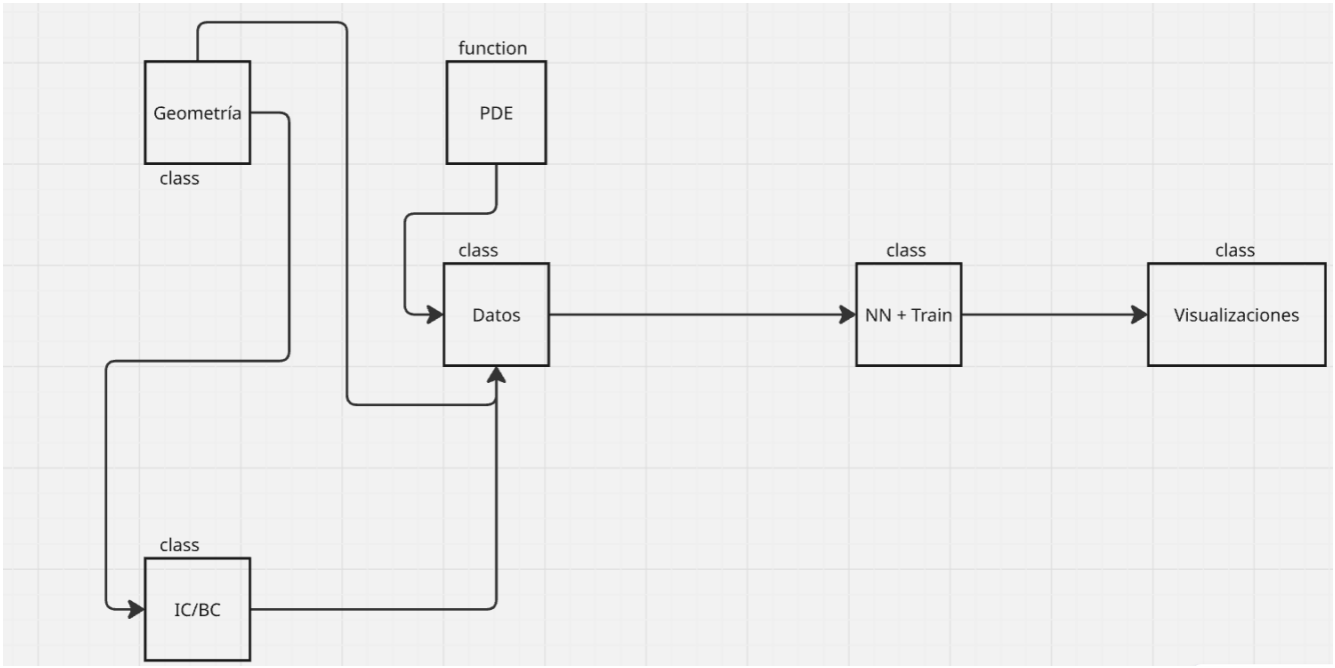

Figure 3. Lógica empleada en la solución del problema

5. Resultados

5.1. Resolución de la ecuación del calor unidimensional

Se obtuvieron los siguientes resultados para una barra de largo 1 y de grosor infinitesimal. Se tienen, además, sus comparaciones con la solución de diferencias finitas y la solución exacta.

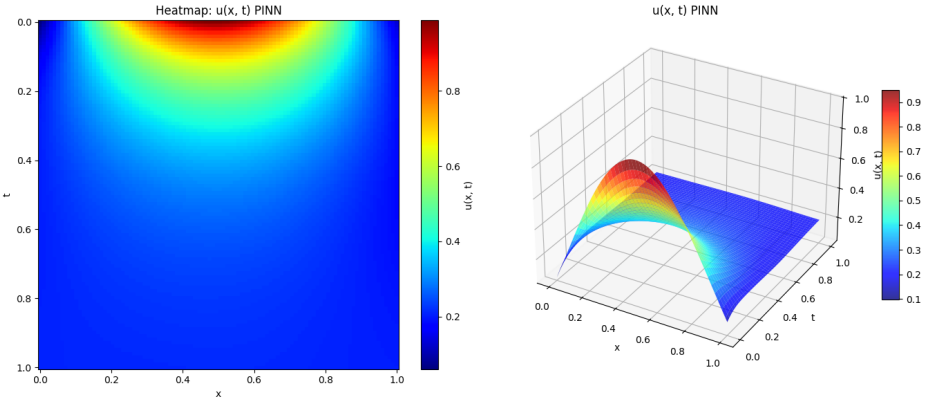

Figure 4. Solución empleando la PINN

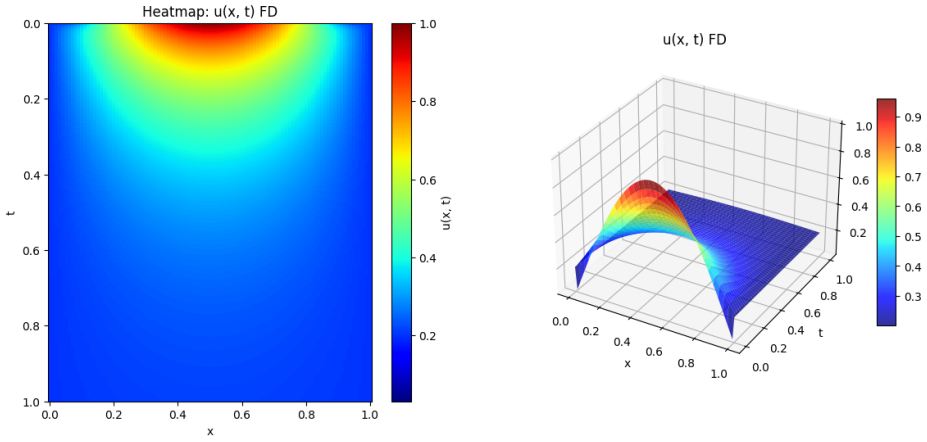

Figure 5. Solución empleando el método de diferencias finitas

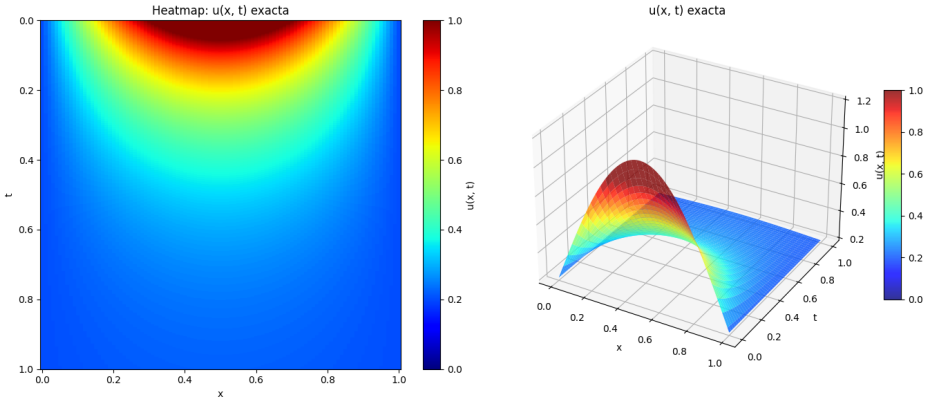

Figure 6. Solución exacta

5.2. Resolución de la ecuación del calor bidimensional

Por otro lado, se tienen los siguientes resultados para una placa cuadrada de área 1 y grosor infinitesimal. Se muestran los resultados para el primer instante de tiempo.

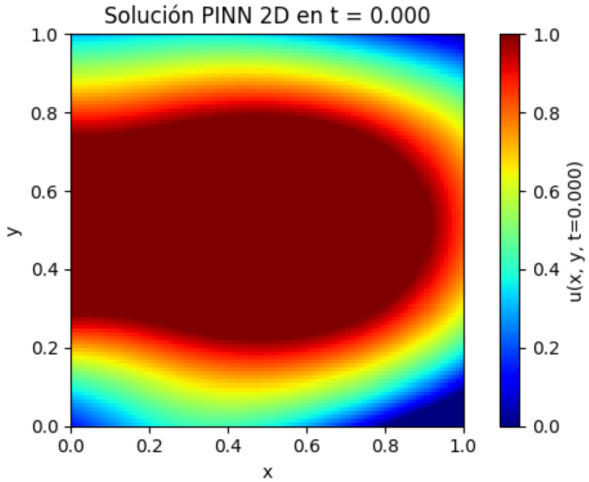

Figure 7. Mapa de calor de la placa

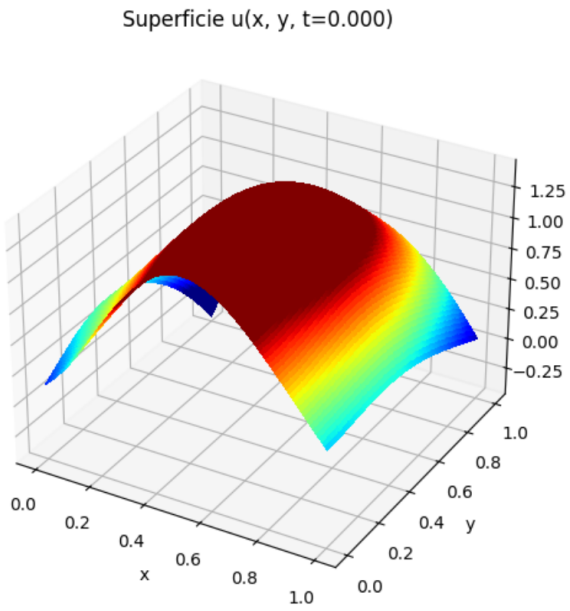

Figure 8. Superficie solución de la placa

5.3. Resolución de la ecuación de la aleta para un disipador de base y aletas rectangulares

Por último, se tienen los siguientes resultados para un disipador de base rectangular y 7 aletas rectangulares. La solución mostrada es en estado estacionario dos segundos después de iniciada la disipación.

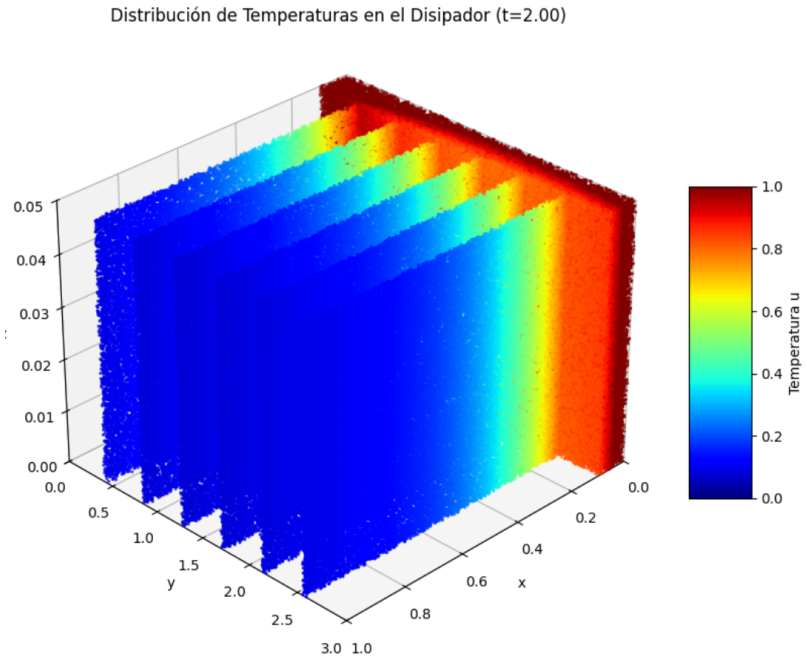

Figure 9. Solución estacionaria para un disipador de calor

6. Conclusiones

Durante este estudio y aplicación de las PINN, pude llegar a varias conclusiones:

- Las PINN permiten integrar directamente la ecuación diferencial parcial (EDP) en el entrenamiento de la red, evitando depender solo de datos experimentales.

- Las PINN lograron aproximar soluciones similares a las obtenidas con el método clásico usado (diferencias finitas).
- Las PINN tienen capacidad de representar condiciones iniciales variadas: polinómicas, sinusoidales, localizadas. Lo mismo para las condiciones de frontera.
- Las PINN ofrecen la ventaja de manejar geometrías irregulares mediante funciones de dominio en lugar de mallado explícito.

References

1. Farea, A.; Yli-Harja, O.; Emmert-Streib, F. **Understanding Physics-Informed Neural Networks: Techniques, Applications, Trends, and Challenges**. *AI*, **2024**, 5 (3), 1534–1557. doi:10.3390/ai5030074.
2. Raissi, M.; Perdikaris, P.; Karniadakis, G. E. **Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations**. *Journal of Computational Physics*, **2019**, 378, 686–707. doi:10.1016/j.jcp.2018.10.045.
3. Karniadakis, G. E.; Kevrekidis, I. G.; Lu, L.; Perdikaris, P.; Wang, S.; Yang, L. **Physics-informed machine learning**. *Nature Reviews Physics*, **2021**, 3 (6), 422–440. doi:10.1038/s42254-021-00314-5.