COMILLAS
UNIVERSIDAD PONTIFICIA

ICAI

GRADO EN INGENIERÍA MATEMÁTICA E INTELIGENCIA ARTIFICIAL

TRABAJO FIN DE GRADO

ENFOQUES NUMÉRICOS Y VARIACIONALES PARA LA RESOLUCIÓN DE ECUACIONES DIFERENCIALES EN DERIVADAS PARCIALES

Autor: Miguel Ara Adánez

Director: Manuel Villanueva Pesqueira

Co-Director: Emanuel Gastón Mompó Pavesi

Madrid, Junio de 2025

UNIVERSIDAD PONTIFICIA COMILLAS
Escuela Técnica Superior de Ingeniería (ICAI)
Grado en Ingeniería Matemática e Inteligencia Artificial

Declaración responsable de autoría y originalidad

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título **Enfoques Numéricos y Variacionales para la Resolución de Ecuaciones Diferenciales en Derivadas Parciales** en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2024/25 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: **Miguel Ara Adánez** Fecha: 10/06/2025

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: **Manuel Villanueva Pesqueira** Fecha: 10/06/2025

AGRADECIMIENTOS

A Manu, por acompañarme desde el primer día hasta el último.

A mi familia por apoyarme.

A mis amigos de la primera promoción de IMAT.

UNIVERSIDAD PONTIFICIA COMILLAS
Escuela Técnica Superior de Ingeniería (ICAI)
Grado en Ingeniería Matemática e Inteligencia Artificial

Enfoques Numéricos y Variacionales para la Resolución de Ecuaciones Diferenciales en Derivadas Parciales

Autor: Miguel Ara Adánez
Director: Manuel Villanueva Pesqueira
Co-Director: Emanuel Gastón Mompó Pavesi
Collaborating Institution: ICAI – Comillas Pontifical University

Abstract

This Bachelor's Thesis studies a subset of differential equations, the partial differential equations (PDEs), from their theoretical foundations through their solution using advanced numerical methods. The project is divided into several stages, beginning with a review of classical theory and an introduction to modern PDE theory, distributional derivatives, and the weak formulation, followed by a survey of classical numerical methods such as finite differences (FD) and finite elements (FEM). Finally, innovative numerical methods, Physics-Informed Neural Networks (PINNs), are explored. These approaches are applied to a practical example involving heat transfer.

Keywords: PDE, FD, FEM, PINN

1. Introduction

Partial differential equations (PDEs) extend ordinary differential equations (ODEs) by involving functions of multiple independent variables, making them more complex and requiring advanced analysis techniques and numerical methods for their solution. Nowadays, PDEs are essential to model phenomena across engineering disciplines such as material elasticity, fluid dynamics, and diffusion processes [1][2]. They describe systems evolving over more than one independent variable, such as the time and space case. Notwithstanding, finding explicit, exact solutions to these type of equations is generally intractable and too complex.

Traditionally, finite differences (FD) and finite elements (FEM) have been the most widely used tools for these problems. These methods rely on subdividing the domain into small elements and storing the resulting mesh, an approach that becomes challenging in complex geometries or high-dimensional settings. Moreover, efficient implementation of the required numerical schemes can be difficult.

Recently, PINNs have emerged as an alternative to classical methods by integrating machine learning concepts and eliminating the need for mesh generation. By embedding the governing physical laws directly into their loss function, these networks offer greater flexibility and lower computational cost in atypical domains, providing a more adaptable framework for solving PDEs [3, 4, 5].

2. Project Definition

In this thesis, we aim to analyze and compare the effectiveness of different numerical and variational approaches, including the use of PINNs, for solving PDEs:

1. **Variational approach and classical methods**: Formulate the problem in its strong form and weak form on discrete Sobolev spaces, leading to finite difference (FD) and finite element (FEM) techniques.
2. **Deep learning with PINNs**: Develop a PINN that incorporates the PDE, initial, and boundary conditions into its loss function. This eliminates the dependence on a mesh, allows handling time-dependent domains, and enables extrapolation beyond the training data.
3. **Practical case study and comparison**: Apply both approaches to solve the one-dimensional heat equation, comparing errors (offgrid MSE via natural cubic spline interpolation) and PINN loss, as well as computational times.

3. System Description

The implemented prototype is organized into 3 configurable modules for the same scenario. Full code is available in a Github Repository [6].

3.1. Classical Module

- *Finite Differences (FD)*: explicit scheme on an evenly spaced mesh with stability analysis via the Courant–Friedrichs–Lewy (CFL) condition.
- *Finite Elements (FEM)*: weak formulation of the problem, assembly of stiffness and mass matrices, and the load vector.

3.2. PINN Module

Architecture: a multi-layer network that takes coordinates (x, t) as input and through the minimizaton of the loss function by SGD (Stochastic Gradient Descent) methods, returns the approximation $u_\theta(x, t)$.

3.3. Combined Module

Thanks to its modular design, the system can switch between methods, adjust parameters easily, collect error and timing metrics, and plot both solutions together, settling the groundwork for future hybrid strategies.

4. Results

- **General comparison.** Classical methods deliver high accuracy with evenly spaced meshes and very low computation times for moderate dimensions. FD is the fastest but requires meeting the CFL condition; FEM, although slightly slower, always maintains stability. PINNs generally require more expensive training and achieve comparable errors only when data is well distributed across the domain, but they have the advantage of not needing a structured mesh.
- **Sampling effect in PINNs.** With evenly spaced sampling, a PINN needs most of the points to reduce its error to levels comparable with classical methods, whereas random sampling achieves faster and more precise convergence with fewer data. This shows that point distribution is as important as quantity: covering the entire domain without a fixed structure allows the network to generalize better.
- **Application to time-dependent domains.** Unlike FD and FEM, which require complex remeshing for each deformation, PINNs handle variable boundaries by adjusting their loss function $\mathcal{L}(\theta)$ and concentrating points in boundary regions. They work in any domain, limited only by compatibility between the initial and boundary conditions.

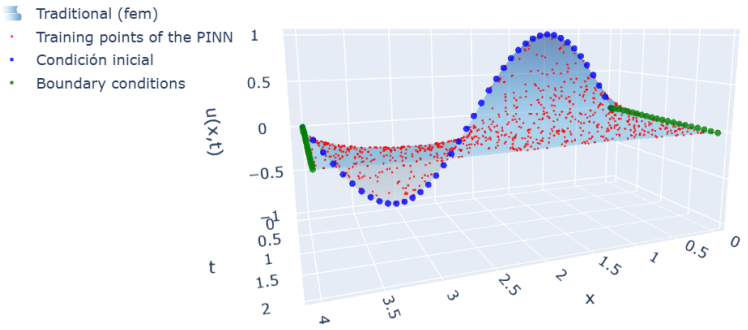

Figura 1: Solution $u(x,t)$ of the heat equation obtained with FEM and the random-sampling training data points from the PINN (2000 epochs).

5. Conclusions

This work thoroughly compared classical numerical solvers (FD, FEM) with Physics-Informed Neural Networks for the one-dimensional heat equation, meeting all stated objectives: theoretical analysis of PDE formulations, implementation and accuracy assessment of FD and FEM (using spline-based offgrid MSE), exploration of PINNs on time-dependent domains via boundary-point reinforcement, joint 3D visualizations and convergence studies for PINNs with evenly spaced and random sampling. The latter is reminiscent of compressed sensing, a technique that profits from unstructured sampling in such a way that signal reconstruction can be done overcoming the Shannon-Nyquist sampling theorem [7]. Another key contributions include demonstrating the benefit of random sampling for PINN convergence, and validating PINN flexibility in complex geometries without explicit meshing.

UNIVERSIDAD PONTIFICIA COMILLAS
Escuela Técnica Superior de Ingeniería (ICAI)
Grado en Ingeniería Matemática e Inteligencia Artificial

Autor: Miguel Ara Adánez
Director: Manuel Villanueva Pesqueira
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

Resumen

Este trabajo de fin de grado (TFG) aborda el estudio de un subcojunto de las ecuaciones diferenciales, las ecuaciones en derivadas parciales (EDPs), desde sus fundamentos teóricos hasta su resolución utilizando métodos numéricos avanzados. El proyecto se divide en varias fases, comenzando con un repaso de la teoría clásica y una introducción a la teoría moderna de las EDPs, la derivada distribucional y la formulación débil, seguido de una revisión de los métodos numéricos clásicos como las diferencias finitas (FD) y los elementos finitos (FEM). Finalmente, se explorarán métodos numéricos innovadores como son las Redes Neuronales Físicamente Informadas o *Physics-Informed Neural Networks* (PINNs). Estos enfoques se aplicarán a un ejemplo práctico relacionado con la transferencia de calor.

Palabras clave: EDP, FD, FEM, PINN

1. Introducción

Las ecuaciones en derivadas parciales (EDPs) son una extensión de las ecuaciones diferenciales ordinarias (EDOs). Al involucrar funciones de múltiples variables independientes son más complejas, y requieren técnicas avanzadas de análisis y métodos numéricos para su resolución. Hoy en día, las EDPs son esenciales para entender y modelar fenómenos en numerosos campos dentro de la ingeniería como la elasticidad de materiales, los fluidos o los procesos de difusión [1][2]. Estas ecuaciones permiten describir sistemas complejos que cambian respecto a más de una variable independiente, como es el caso del tiempo y del espacio. No obstante, resolver este tipo de ecuaciones de forma explícita y exacta es demasiado complejo.

Tradicionalmente, las diferencias finitas (FD) y los elementos finitos (FEM) han sido las herramientas más empleadas para abordar estos problemas. Estos enfoques se basan en dividir el dominio de la ecuación en pequeñas partes y almacenar el mallado empleado, algo difícil de lograr cuando se trabaja en geometrías complejas o en problemas de alta dimensionalidad. Además, los esquemas numéricos necesarios son difíciles de implementar de forma eficiente.

Recientemente se han comenzado a desarrollar las PINNs como una alternativa frente a los métodos tradicionales que emplea conceptos del aprendizaje automático y que elimina la necesidad de un mallado. Al integrar directamente las leyes de la física en su propia función de pérdida, estas redes poseen mayor flexibilidad y menor coste computacional en dominios atípicos, proporcionando un enfoque más adaptable en la resolución de EDPs [3, 4, 5].

2. Definición del proyecto

En este TFG, la intención es analizar y comparar la efectividad de los diferentes enfoques numéricos y variacionales, incluyendo el uso de PINNs, para resolver EDPs:

1. **Enfoque variacional y métodos clásicos**: Plantear el problema en su forma fuerte y en su formulación débil sobre espacios discretos de Sobolev, lo que conduce a técnicas de diferencias finitas y elementos finitos.

2. **Aprendizaje profundo con PINNs**: Desarrollar una PINN que incorpore en su función de pérdida la ecuación de la EDP junto con las condiciones iniciales y de contorno. Se eliminará la dependencia de un mallado, se permitirá trabajar con dominios no constantes o variantes en el tiempo además de la extrapolación fuera de los datos de entrenamiento.

3. **Caso práctico y comparación**: Aplicar ambos métodos a la resolución de la ecuación del calor unidimensional, comparando los errores (MSE *offgrid* mediante interpolación cúbica natural con *splines*, y la pérdida de la PINN) y los tiempos de computación.

3. Descripción del sistema

El prototipo implementado se organiza en tres módulos configurables para el mismo caso. El código completo se encuentra disponible en un repositorio de Github [6].

3.1. Módulo clásico

- *Diferencias finitas (FD)*: esquema explícito sobre malla equiespaciada, con análisis de estabilidad por el criterio CFL (*Courant-Friedrichs-Lewy condition*)

- *Elementos finitos (FEM)*: formulación débil del problema, ensamblaje de las matrices de rigidez y masa con el vector de carga.

3.2. Módulo PINN

Arquitectura: red de varias capas que toma como entrada las coordenadas (x, t) y, a través de una optimización con métodos estocásticos de descenso de gradiente, va disminuyendo la función de pérdida y devuelve la aproximación $u_\theta(x, t)$:

3.3. Combinación del módulo clásico y del módulo PINN

Gracias a su diseño modular, el sistema permite alternar entre ambos métodos, ajustar parámetros de forma sencilla, recoger métricas de error y tiempo de cómputo, y graficar de forma conjunta ambas soluciones, sentando así las bases para futuras estrategias híbridas.

4. Resultados

- **Comparativa general.** Los métodos clásicos proporcionan alta precisión con mallas equiespaciadas y tiempos de cómputo muy bajos cuando las dimensiones son moderadas. FD es el más rápido, pero exige cumplir la condición CFL; FEM, aunque algo más lento, mantiene siempre estabilidad. Las PINNs generalmente necesitan un entrenamiento más costoso y logran errores comparables solo cuando se dispone de datos distribuidos por todo el dominio, pero no requieren de un mallado estructurado.
- **Efecto del muestreo en PINN.** Con muestreo equiespaciado, la PINN requiere la mayor parte de los puntos para disminuir su error a niveles comparables a los métodos clásicos, mientras que el muestreo aleatorio logra una convergencia más rápida y precisa con menos datos. Esto demuestra que la distribución de puntos es tan importante como su cantidad: cubrir todo el dominio sin estructura fija permite a la red generalizar mejor.
- **Aplicación en dominios variantes en el tiempo.** A diferencia de FD y FEM, que requieren mallados complejos para cada deformación, las PINNs manejan fronteras variables ajustando su función de pérdida $\mathcal{L}(\theta)$ y concentrando puntos en las zonas de contorno. Se logra reproducir correctamente la solución incluso en dominios variantes, quedando limitada únicamente por la compatibilidad entre condición inicial y contorno.

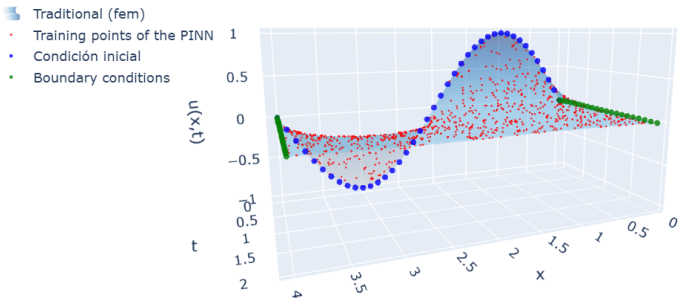

Figura 2: Solución $u(x,t)$ de la ecuación del calor obtenida con el método FEM junto con los datos de entrenamiento (2000 épocas) aleatorios de una PINN

5. Conclusiones

En este proyecto se ha comparado rigurosamente la resolución de la ecuación del calor 1D mediante métodos clásicos (FD y FEM) y PINN, alcanzando todos los objetivos planteados: se evaluó la precisión y tiempo de cómputo de FD y FEM con interpolación mediante *splines*, se demostró la ventaja de las PINN en dominios no constantes reforzando los puntos y pesos de contorno, se visualizó la solución conjunta en 3D y se analizó la convergencia de la PINN con muestreos equiespaciados y aleatorios. Esto último recuerda al *compressed sensing*, una técnica que se beneficia del muestreo no estructurado de tal forma que la reconstrucción de la señal puede llevarse a cabo superando el teorema de muestreo de Shannon-Nyquist [7]. Se evidencia que el muestreo aleatorio reduce sustancialmente el error en la PINN, y que esta ofrece flexibilidad para dominios variables sin necesidad de mallado explícito.

6. Referencias

[1] Evans, L. C. (1994). Partial differential equations/ by Lawrence C. Evans. https://ci.nii.ac.jp/ncid/BA27007864

[2] Haberman, R. (1998). Elementary Applied partial differential equations: With Fourier Series and Boundary Value Problems.

[3] Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2018, 3 noviembre). A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. https://www.sciencedirect.com/science/article/

[4] Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2017, 28 noviembre). Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. arXiv.org. https://arxiv.org/abs/1711.10561

[5] Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2017, 28 noviembre). Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations. arXiv.org. https://arxiv.org/abs/1711.10566

[6] Repositorio GitHub con el código del TFG https://github.com/miguel-ara/TFG

[7] Blind Multiband Signal Reconstruction: Compressed Sensing for Analog Signals. (2009, 1 marzo). IEEE Journals & Magazine — IEEE Xplore. https://ieeexplore.ieee.org/document/4749297

Índice

1. Introducción

1.1. Contexto y Motivación

Las ecuaciones en derivadas parciales (EDP) constituyen una herramienta fundamental para modelar fenómenos físicos y de ingeniería en los que intervienen múltiples variables independientes. Por ejemplo, la ecuación de difusión del calor en un sólido se puede expresar como

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u,$$

donde $u(x,t)$ es la temperatura en el punto x y en el instante t, y α es el coeficiente de difusividad térmica. De forma más general, las EDP pueden escribirse en la forma

$$\mathcal{N}[u](x,t) = f(x,t), \quad (x,t) \in \Omega \times (0,T],$$

acompañadas de la condición inicial en $t = 0$ y de las condiciones de contorno apropiadas sobre la frontera del dominio ($\partial\Omega$) [1]. Estos modelos son esenciales para describir fenómenos tales como la mecánica de fluidos, la transferencia de calor, y la propagación de ondas.

$\mathcal{N}$ es un operador diferencial que actúa sobre la función desconocida u y, al evaluarlo en cada punto x del dominio Ω, debe igualar el valor dado por $f(x)$. De nuevo, centrándonos en la ecuación del calor el operador sería:

$$\mathcal{N} = \partial_t - \alpha\, \Delta.$$

El cual, al desarrollarlo, nos deja la expresión final de la EDP acompañada de condiciones de contorno $u(x,t) = g(x,t)$ en $\partial\Omega$ y condición inicial $u(x,0) = u_0(x)$, imprescindibles para que exista solución de la EDP.

$$\mathcal{N}[u](x,t) = \frac{\partial u}{\partial t}(x,t) - \alpha\, \Delta u(x,t),$$

$$\mathcal{N}[u](x,t) = f(x,t) \implies \frac{\partial u}{\partial t}(x,t) - \alpha\, \Delta u(x,t) = f(x,t), \quad (x,t) \in \Omega \times (0,T],$$

Así, $\mathcal{N}[u] = f$ es una forma compacta de expresar que una combinación diferenciable de u debe coincidir con la fuente f en todo el dominio Ω.

La resolución numérica de las EDP plantea importantes retos. Los métodos clásicos como diferencias finitas o elementos finitos requieren en primer lugar una discretización o mallado equiespaciado del dominio Ω. En geometrías complejas o en problemas de alta dimensionalidad, construir este mallado se vuelve costoso, tanto en tiempo de cálculo como en memoria [2]. En este contexto, las Redes Neuronales Físicamente Informadas o *Physics-Informed Neural Networks* (PINNs) han surgido como una alternativa prometedora. La idea básica es

aproximar la solución $u(x,t)$ mediante una red neuronal $u_\theta(x,t)$, cuyos parámetros θ se optimizan minimizando una función de pérdida que incluye un término de residuo de la EDP, un término para la condición inicial y un término para las condiciones de contorno:

$$\mathcal{L}(\theta) = \underbrace{\sum_{\substack{x_i^{bc}\in\partial\Omega \\ t_j^{bc}\in[0,T]}} \left|u_\theta(x_i^{bc},t_j^{bc}) - g(x_i^{bc},t_j^{bc})\right|^2}_{\text{error en condiciones de contorno}} + \underbrace{\sum_{x_i^0\in\Omega} \left|u_\theta(x_i^0,0) - u_0(x_i^0)\right|^2}_{\text{error en condición inicial}} + \underbrace{\sum_{\substack{x_i\in\Omega \\ t_j\in[0,T]}} \left|\mathcal{N}[u_\theta](x_i,t_j) - f(x_i,t_j)\right|^2}_{\text{residuo de la EDP en el dominio}}$$

- $\partial\Omega$ es la frontera del dominio, el conjunto de puntos (x_i^{bc}, t_j^{bc}) donde se imponen las condiciones de contorno $u(x,t) = g(x,t)$.
- $\{x_i^0 \in \Omega\}$ es el conjunto de puntos donde se cumple la condición inicial $u(x,0) = u_0(x)$.
- (x_j, t_j) es el conjunto de puntos del dominio con los que se entrena la PINN.
- $\mathcal{N}[u]$ es el operador diferencial que define la EDP, y $f(x,t)$ es el término fuente.

Minimizando $\mathcal{L}(\theta)$, la red aprende simultáneamente a respetar las condiciones de contorno y la condición inicial, mientras disminuye el residuo de la EDP en todo el dominio haciendo que la solución $u(x,t)$ verifique la ecuación de la EDP. De esta forma, la red incorpora directamente la información física del problema sin necesidad de un mallado explícito, lo que puede mejorar la flexibilidad y reducir el coste computacional en dominios no constantes en el tiempo o de alta dimensión [4, 8].

La motivación de este trabajo radica en comparar rigurosamente los métodos numéricos clásicos con las PINNs, con el fin de identificar escenarios en los que cada enfoque sea más adecuado. Por un lado, los métodos de diferencias finitas o elementos finitos cuentan con un amplio respaldo teórico respecto a convergencia y estabilidad. Por otro lado, las PINNs demuestran su potencial y flexibilidad cuando el mallado es complejo. De este modo, se obtendrá una visión completa de las ventajas, limitaciones y alcances de cada estrategia aplicada a la resolución de EDPs.

1.2. Objetivos

Los objetivos de este proyecto incluyen los siguientes aspectos:

- **Fundamentos teóricos de las EDP.** Revisar la formulación clásica o fuerte de las EDPs y la formulación moderna o débil, derivadas distribucionales y condiciones de contorno/inicial para problemas de Poisson, Laplace y la ecuación del calor.

- **Implementación de métodos numéricos clásicos.** Desarrollar en Python los esquemas de diferencias finitas y elementos finitos para la ecuación del calor (módulo `traditional.py`, funciones `heat_equation_fd` y `heat_equation_fem`), así como para Poisson/Laplace en 1D, evaluando su convergencia y precisión. Implementar un método de evaluación mediante interpolación de la malla con *splines* cúbicos (función `residual_MSE_offgrid_two_stage`).

- **Construcción y entrenamiento de una PINN.** Implementar una PINN para la ecuación del calor (clase `HeatEquationPINN` en `pinn.py`), definiendo la función de pérdida completa, con los términos PDE (*Partial Differential Equation*), IC (*Initial Condition*) y BC (*Boundary Conditions*), utilizando `torch.autograd` para calcular derivadas y detallando la estrategia de entrenamiento. Implementar dominios variantes, es decir, con condiciones de contorno no constantes en forma de sierra (*sawtooth*), senoidal, o de onda cuadrada.

- **Resolución de un caso práctico y comparación.** Generar un caso de prueba concreto (función fuente f, condición inicial u_0 y contornos g) y resolverlo con ambos enfoques:

  - El método clásico seleccionado (`heat_equation_fd` o `heat_equation_fem`).
  - La PINN entrenada (`HeatEquationPINN`).

  Emplear `combination.py` para superponer soluciones, comparar tiempos de cómputo y error numérico como el error cuadrático medio o MSE (*Mean Squared Error*) del residuo, e identificar ventajas y limitaciones de cada enfoque.

1.3. Planificación

A continuación se presenta la planificación detallada del TFG, organizada en fases secuenciales. En la Figura 3 se muestra el diagrama de Gantt que ilustra la distribución temporal de cada fase a lo largo del calendario del proyecto.

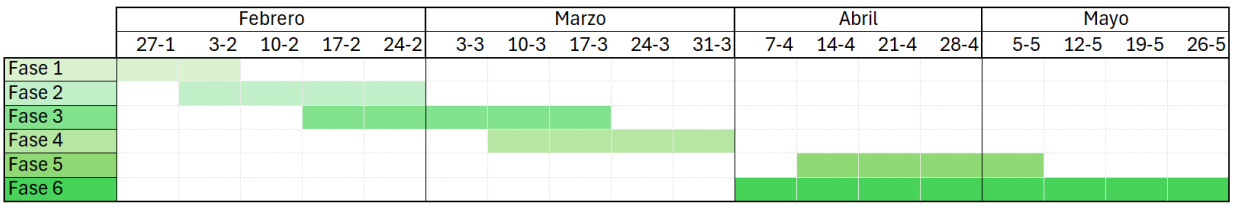

Figura 3: Diagrama de Gantt del proyecto

Fase 1: Introducción y planteamiento

- Familiarizarse con los objetivos y alcance del TFG.
- Revisar bibliografía inicial sobre EDP, métodos clásicos y PINNs.
- Redactar el marco teórico introductorio, explicando la importancia de las EDP en ciencia e ingeniería y motivando el uso de PINNs.

Fase 2: Fundamentos teóricos

- Estudiar la formulación clásica o fuerte de las EDP con las series de Fourier y la formulación moderna o débil, con la derivada distribucional y espacios de Sobolev
- Profundizar en los métodos numéricos tradicionales (diferencias finitas y elementos finitos) para Poisson, Laplace y la ecuación del calor.

Fase 3: Métodos numéricos clásicos

- Implementar en Python los esquemas de diferencias finitas (FD) y elementos finitos (FEM) para la ecuación del calor y para Poisson/Laplace en 1D (módulo `traditional.py`).
- Verificar convergencia y precisión calculando el error MSE del residuo y realizando pruebas *offgrid* con interpolación (*splines* cúbicos).
- Documentar los resultados, generar tablas de error frente a refinamiento de malla y redactar el apartado correspondiente.

Fase 4: Introducción a las PINNs

- Estudiar las bases teóricas de las PINNs y su integración con las EDPs, tal y como se describe en los artículos [3, 4, 5]
- Implementar la clase `HeatEquationPINN` en el módulo `pinn.py`: definir arquitectura, la función de pérdida completa (PDE, IC y BC) y configurar el flujo de entrenamiento.
- Realizar experimentos iniciales en un caso sencillo de la ecuación del calor, ajustando hiperparámetros y analizando la evolución de cada término de la pérdida.

Fase 5: Desarrollo de aplicaciones prácticas

- Resolver el problema de transferencia de calor con fuente dependiente de x y t usando:
  - El método clásico elegido (`heat_equation_fd` o `heat_equation_fem`).
  - La PINN entrenada (`HeatEquationPINN`).

- Emplear `combination.py` para superponer soluciones, comparar gráficas y calcular métricas de error y tiempo de ejecución.
- Documentar resultados y redactar la correspondiente sección

Fase 6: Revisión y redacción final

- Revisar y pulir todos los capítulos: introducción 1, estado del arte 2, marco teórico 3, metodología 4, experimentos 5, resultados 6 y conclusiones 7.
- Incluir reflexiones críticas sobre las fortalezas y limitaciones de cada enfoque, así como posibles líneas futuras como la extensión a dimensiones superiores.
- Preparar la presentación para la defensa del TFG.

De esta forma, la planificación queda distribuida a lo largo de aproximadamente 20 semanas, garantizando un progreso equilibrado entre la teoría matemática, la implementación, el análisis de resultados y la redacción del documento final.

1.4. Viabilidad Económica

El desarrollo de este TFG se basa en herramientas de código abierto y en recursos informáticos ya disponibles, por lo que los costes adicionales son mínimos. A continuación se detallan los principales conceptos que afectan a la viabilidad económica:

- **Hardware**
  - Se utilizará un ordenador personal con CPU moderna y GPU compatible con la librería `PyTorch` de `Python`. No se requerirá la adquisición de nuevo equipamiento, pues el equipo con el que ya se trabaja satisface los requisitos de entrenamiento de la PINN (8 GB de GPU y 16 GB de RAM).
  - El consumo eléctrico derivado de las etapas de entrenamiento y ejecución de esquemas numéricos es moderado. Estimando un uso intensivo de la GPU durante 2 horas diarias en el pico de la fase de experimentación, a 0,20 €/kWh, el coste total de electricidad no superará los 10 € para todo el proyecto.
- **Software**
  - Todos los paquetes empleados (`Python`, `NumPy`, `SciPy`, `PyTorch`, `Matplotlib`, `Plotly`) son de código abierto, por lo que no hay coste de licencias.
  - El entorno de edición también es gratuito, sin necesidad de licencias comerciales.

En conjunto, se estima que el coste total adicional para llevar a cabo el proyecto es mínimo, resulta asumible dentro del marco académico y no supone un obstáculo para la viabilidad económica del TFG.

1.5. Estructura del Trabajo

La memoria comienza situando al lector en el contexto de las EDP 1, motivando su estudio y definiendo objetivos claros, seguido de un análisis del estado del arte 2 que sintetiza tanto los métodos clásicos (FD y FEM) como los más recientes basados en PINNs. A continuación, el marco teórico 3 profundiza en los fundamentos matemáticos: formulación fuerte y débil, espacios de Sobolev, esquemas de diferencias finitas y elementos finitos, y la construcción de las PINN con su arquitectura y función de pérdida que integra PDE, condiciones iniciales y de contorno.

La sección de metodología 4 describe de forma concisa cómo se implementan los métodos clásicos y la PINN, y cómo se combinan ambas aproximaciones para comparar precisión y eficiencia. En experimentos 5 se fijan la ecuación de calor, la configuración de mallas y muestreos (equiespaciado y aleatorio), los parámetros de la red y los criterios de evaluación (tiempo de cómputo, MSE offgrid y visualizaciones 3D). Los resultados 6 muestran el comportamiento relativo de FD, FEM y PINN, y las conclusiones 7 recapitulan aportaciones, limitaciones y plantean líneas de trabajo futuro como extensiones a dimensiones superiores, soluciones a problemas inversos o propuestas híbridas avanzadas.

2. Estado del Arte

La resolución numérica de EDP ha estado dominada tradicionalmente por métodos como diferencias finitas (FD) y elementos finitos (FEM), que poseen un importante respaldo teórico en términos de convergencia y estabilidad. Haberman (1998) presenta distintos esquemas explícitos e implícitos para la ecuación del calor en 1D, estableciendo condiciones CFL que garantizan la estabilidad en FD [2]. Por otro lado, Evans (1994) detalla la formulación de FEM en 1D y 2D, con el ensamblado de matrices de rigidez y masa que se resuelven mediante métodos directos o iterativos para obtener soluciones con un error controlado [1]. No obstante, el mallado en dominios no constantes y la escalabilidad de estos métodos en dimensiones superiores continúan siendo desafíos relevantes.

Entre los años 2017 y 2018, Raissi et al. introdujeron las *Physics-Informed Neural Networks* (PINNs) como un nuevo paradigma que incorpora la EDP en la función de pérdida de una red neuronal. En su artículo de 2017 se define la estructura básica de PINNs, donde el término de residuo PDE+IC+BC permite entrenar la red sin requerir un mallado explícito [4]. En 2018 amplían esta idea para abordar problemas inversos, demostrando que las PINNs pueden estimar parámetros desconocidos de la EDP a partir de datos parciales [3].

A la hora de comparar directamente las PINNs con los métodos tradicionales FD y FEM, Grossmann et al. (2023) documentan un estudio en el que se resuelve la ecuación de Poisson 1D con ambas metodologías y comparan costos computacionales y precisión frente a la solución analítica. Sus resultados indican que, si bien algunas arquitecturas PINN pueden

aproximar con error comparable a FEM, los tiempos de entrenamiento superan a FEM en dos o tres órdenes de magnitud [11]. Sacchetti et al. (2022) realizan una comparación análoga para la ecuación del calor en 2D, concluyendo que FEM ofrece mayor precisión y eficiencia computacional que las PINNs para resoluciones finas [12].

Para superar las limitaciones de cada enfoque, han surgido métodos híbridos que combinan FEM o FD con PINNs. Sobh et al. (2025) presentan *PINN-FEM*, un enfoque que introduce elementos finitos cerca de las fronteras para imponer condiciones de Dirichlet de forma fuerte, mientras la PINN se encarga de aproximar el dominio interior. Sus experimentos muestran una mejor precisión y convergencia en problemas lineales y no lineales comparados con las PINNs estándar [13]. Chen et al. (2025) proponen un método *PINN-enriched FEM*, donde la aproximación obtenida por una PINN se enriquece en el espacio FEM mediante sumas o productos, logrando órdenes de convergencia superiores al FEM convencional para ecuaciones elípticas 1D–3D [14].

Del mismo modo, Kumar et al. (2024) introducen "PINNacle", un *benchmarking* que evalúa más de 20 EDPs sobre distintas arquitecturas PINN y los compara con esquemas clásicos. Sus resultados resaltan la importancia del diseño de la red y del muestreo de puntos para garantizar precisión y eficiencia, señalando que en muchos casos las PINNs aún no igualan a FEM en tiempo de cómputo [15]. En cuanto a problemas con condiciones de frontera especiales, Oh y Jo (2025) desarrollan una PINN para la ecuación del calor con contacto imperfecto en interfaces, demostrando que, mediante una extensión de variables, es posible acotar el error y mejorar la convergencia en dominios con discontinuidades [16].

A pesar de estos avances, persisten áreas por explorar: no hay estudios exhaustivos que comparen PINNs y métodos clásicos específicamente para la ecuación del calor 1D en diversos regímenes de discretización, ni trabajos que analicen sistemáticamente el impacto de la selección de puntos de entrenamiento en PINNs frente a criterios CFL en FD. Este trabajo se propone llenar esa laguna implementando y comparando ambos métodos FD y FEM con PINNs en la ecuación del calor 1D, midiendo tiempos junto con el MSE del residuo, e identificando escenarios donde cada estrategia resulte más ventajosa.

3. Marco Teórico

3.1. Fundamentos de las EDPs

3.1.1. Definición de EDP (forma fuerte).

Sea $\Omega \subset \mathbb{R}^d$ un dominio (conjunto abierto y acotado) con frontera $\partial\Omega$. Una ecuación en derivadas parciales (EDP) se expresa en términos de un operador diferencial $\mathcal{N}$ aplicado a una función desconocida $u : \Omega \to \mathbb{R}$. Como ya se ha mencionado en la Introducción 1, el

problema clásico en forma fuerte es:

$$\mathcal{N}[u](x) \;=\; f(x), \quad x \in \Omega,$$

sujeto a condiciones de contorno (Dirichlet, Neumann o mixtas) sobre $\partial\Omega$ y, en problemas evolutivos, condiciones iniciales en $t = 0$. Por ejemplo, la **ecuación de Poisson** en Ω es

$$-\Delta u(x) \;=\; f(x), \quad x \in \Omega, \quad u(x)|_{\partial\Omega} \;=\; g(x),$$

donde $\Delta = \sum_{i=1}^{d} \partial^2_{x_i}$ es el operador laplaciano [9]. El caso homogéneo ($f \equiv 0$) se conoce como **ecuación de Laplace**:

$$-\Delta u(x) = 0, \quad x \in \Omega, \quad u|_{\partial\Omega} = g.$$

Para la **ecuación del calor** en un sólido, con coeficiente de difusividad $\alpha > 0$, se considera

$$\frac{\partial u}{\partial t}(x,t) - \alpha\, \Delta u(x,t) \;=\; f(x,t), \quad (x,t) \in \Omega \times (0,T],$$

acompañada de

$$u(x,0) = u_0(x), \quad x \in \Omega, \quad u(x,t) = g(x,t), \quad (x,t) \in \partial\Omega \times (0,T]\,.$$

Una forma clásica de obtener soluciones fuertes es recurrir a la separación de variables y la expansión en series de Fourier. Se supone

$$u(t,x) = T(t)\, X(x),$$

con X y T suficientemente suaves ($u \in C^2$) que satisfacen las condiciones de contorno, lo que reduce la EDP a un problema de Sturm–Liouville para $X(x)$ y a una EDO (Ecuación Diferencial Ordinaria) para $T(t)$, dando lugar a la solución en serie de autofunciones ortogonales. Esta formulación exige que la solución sea C^2 sin discontinuidades ni singularidades, pues las derivadas de segundo orden deben existir punto a punto.

En cambio, como veremos a continuación, la formulación débil o moderna admite soluciones $u \in H^1(\Omega)$ (espacio que se definirá más adelante en la sección 3.1.3, permitiendo saltos o picos cuando la fuente no es continua. Cualquier solución fuerte (C^2) es también solución débil, pero no todas las soluciones débiles poseen la regularidad necesaria para ser fuertes.

3.1.2. Derivadas distribucionales.

Para extender la noción de derivada a funciones que no son puntualmente diferenciables, se usan distribuciones. En este nuevo enfoque no identificamos a la función en sí misma, sino que la estudiamos por el efecto que tiene sobre otras funciones, de forma similar a como ocurre con los sistemas electrónicos al intentar caracterizar un circuito según cómo reacciona ante impulsos. Sea $\mathcal{D}(\Omega)$ el espacio de funciones de prueba, es decir, infinitamente diferenciables

con soporte compacto en Ω. Una función generalizada u define una distribución $\langle u, \varphi \rangle$ para $\varphi \in \mathcal{D}(\Omega)$. La derivada distribucional $\partial_{x_i} u$ se define por

$$\langle \partial_{x_i} u, \varphi \rangle \;=\; -\langle u, \partial_{x_i} \varphi \rangle, \quad \forall\, \varphi \in \mathcal{D}(\Omega).$$

Esta definición coincide con la derivada clásica cuando u es suavemente diferenciable, pero además nos permite tratar funciones con singularidades [10].

3.1.3. Espacios de Sobolev.

Para poder formular problemas de EDP en su forma débil, es necesario trabajar con espacios de funciones que admitan derivadas "en sentido débil" y sean estables bajo ciertas normas. El espacio de Sobolev más básico es

$$H^1(\Omega) \;=\; \big\{\, u \in L^2(\Omega) \;:\; \partial_{x_i} u \in L^2(\Omega) \text{ (en sentido débil) para } i = 1, \ldots, d \big\},$$

donde:

- $L^2(\Omega)$ es el conjunto de funciones u de cuadrado integrable, es decir, $\displaystyle\int_\Omega |u(x)|^2\, dx < \infty$.
- Decir que $\partial_{x_i} u$ existe "en sentido débil" implica que existe una función $v_i \in L^2(\Omega)$ que cumple con
$$\int_\Omega u\, \frac{\partial \varphi}{\partial x_i}\, dx \;=\; -\int_\Omega v_i\, \varphi\, dx \quad \forall\, \varphi \in \mathcal{D}(\Omega),$$
en cuyo caso se define $\partial_{x_i} u = v_i$.
- La norma en $H^1(\Omega)$ se define como
$$\|u\|_{H^1(\Omega)} = \left(\int_\Omega |u(x)|^2\, dx \;+\; \sum_{i=1}^{d} \int_\Omega \big|\partial_{x_i} u(x)\big|^2\, dx \right)^{1/2}.$$

El subespacio

$$H_0^1(\Omega) \;=\; \overline{C_c^\infty(\Omega)}^{\|\cdot\|_{H^1}}$$

está formado por aquellas funciones de $H^1(\Omega)$ cuya traza en $\partial\Omega$ es cero, es decir que se anulan en la frontera en sentido débil. Estos espacios de Sobolev son fundamentales porque:

- Permiten formular la versión débil de EDP, ya que se pueden integrar por partes sin exigir derivabilidad clásica.
- Garantizan existencia y unicidad de la solución mediante teoremas de Lax–Milgram y propiedades de compactación (Rellich–Kondrachov).
- A través de desigualdades (Poincaré, Sobolev) se controla la norma L^2 en función de la norma de las derivadas, esencial para análisis de convergencia y estabilidad.

3.1.4. Formulación débil.

Partiendo de la ecuación de Poisson

$$-\Delta u = f \quad \text{en } \Omega, \qquad u|_{\partial\Omega} = g,$$

buscamos u en el espacio de Sobolev $H^1(\Omega)$, las funciones con derivadas débiles que son además de cuadrado integrable. Definimos entonces el espacio $H^1_g(\Omega)$, además del $H^1_0(\Omega)$ que ya teníamos.

$$H^1_g(\Omega) \;=\; \{\, v \in H^1(\Omega) \mid v|_{\partial\Omega} = g\}, \quad H^1_0(\Omega) \;=\; \{\, v \in H^1(\Omega) \mid v|_{\partial\Omega} = 0\}.$$

Multiplicamos la ecuación fuerte por una función de test $v \in H^1_0(\Omega)$ y aplicamos integración por partes:

$$\int_\Omega \big(-\Delta u(x)\big)\, v(x)\,\mathrm{d}x = \int_\Omega f(x)\, v(x)\,\mathrm{d}x \quad \Longrightarrow \quad \int_\Omega \nabla u(x)\cdot\nabla v(x)\,\mathrm{d}x = \int_\Omega f(x)\, v(x)\,\mathrm{d}x.$$

La formulación débil, también conocida como variacional, queda: encontrar $u \in H^1_g(\Omega)$ tal que

$$a(u,v) \;=\; \ell(v), \quad \forall\, v \in H^1_0(\Omega),$$

donde

$$a(u,v) = \int_\Omega \nabla u\cdot\nabla v\,\mathrm{d}x, \quad \ell(v) = \int_\Omega f\, v\,\mathrm{d}x.$$

En la ecuación del calor, la formulación débil en cada paso temporal (método de Rothe) o directamente en espacio-tiempo se construye de manera análoga, integrando también el término $\int_\Omega u_t\, v$, y requiere funciones $u(\cdot,t) \in H^1(\Omega)$ con continuidad en t en $L^2(\Omega)$ [1].

3.2. Métodos Numéricos Tradicionales

3.2.1. Diferencias finitas (FD)

En el método de diferencias finitas se aproximan las derivadas parciales en puntos de una malla equiespaciada. Sea $\{x_i\}_{i=0}^{N-1}$ con

$$x_i = a + i\,h, \qquad h = \frac{b-a}{N-1},$$

donde N es el número total de puntos en la malla. Para la derivada segunda en x, usamos la fórmula de diferencias centradas de segundo orden:

$$\frac{\mathrm{d}^2u}{\mathrm{d}x^2}(x_i) \;\approx\; \frac{u_{i-1} - 2\,u_i + u_{i+1}}{h^2}, \qquad i = 1, 2, \ldots, N-2.$$

Para resolver la ecuación de Poisson en 1D se toma

$$-\,u''(x) = f(x), \quad x \in [a,b], \quad u(a) = \alpha,\; u(b) = \beta,$$

y se aproxima $-u''(x_i) = f(x_i)$ mediante

$$-\frac{u_{i-1} - 2\,u_i + u_{i+1}}{h^2} \;=\; f_i, \qquad i = 1, \ldots, N-2,$$

con $u_0 = \alpha$ y $u_{N-1} = \beta$. De esta forma se obtiene un sistema lineal tridiagonal $A\,\mathbf{u} \;=\; \mathbf{b}$ de tamaño $N \times N$ donde $\mathbf{u} = (u_0,\, u_1,\, \ldots,\, u_{N-1})^T$ y $\mathbf{b} = (b_0,\, b_1,\, \ldots,\, b_{N-1})^T$ se construyen como sigue:

$$A \;=\; \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} & \cdots & 0 & 0 \\ 0 & \frac{1}{h^2} & -\frac{2}{h^2} & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}, \qquad \mathbf{b} \;=\; \begin{pmatrix} \alpha \\ f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{N-2}) \\ \beta \end{pmatrix}.$$

En este sistema:

- La primera fila $A_{0,*} = (1,\, 0,\, 0,\, \ldots,\, 0)$ y $b_0 = \alpha$ impone $u_0 = u(a) = \alpha$.
- La última fila $A_{N-1,*} = (0,\, 0,\, \ldots,\, 0,\, 1)$ y $b_{N-1} = \beta$ impone $u_{N-1} = u(b) = \beta$.
- Para $i = 1, \ldots, N-2$, la fila i de A es

  $$A_{i,i-1} = \tfrac{1}{h^2}, \quad A_{i,i} = -\tfrac{2}{h^2}, \quad A_{i,i+1} = \tfrac{1}{h^2}, \quad b_i = f(x_i).$$

  Esto corresponde a $-\frac{u_{i-1} - 2\,u_i + u_{i+1}}{h^2} = f(x_i)$.

Por tanto, el sistema tridiagonal $A\,\mathbf{u} = \mathbf{b}$ representa exactamente la discretización de orden $O(h^2)$ en espacio para la ecuación $-u'' = f$ con contornos $u(a) = \alpha$ y $u(b) = \beta$

Para la ecuación de Laplace el razonamiento es idéntico considerando $f(x) = 0$

Para la ecuación del calor en 1D se considera

$$u_t = u_{xx} + f(x,t), \quad x \in [a,b],\; t \in [0,T], \quad u(a,t) = \alpha(t),\; u(b,t) = \beta(t), \quad u(x,0) = u_0(x).$$

Se define $u_i^n \approx u(x_i, t^n)$, donde $t^n = n\,\Delta t$ para $n = 0, 1, \ldots, N_t$. Con

$$r = \frac{\Delta t}{h^2},$$

el esquema explícito (Euler hacia adelante en tiempo y diferencias centradas en espacio) es

$$u_i^{n+1} = u_i^n + r\left(u_{i-1}^n - 2\,u_i^n + u_{i+1}^n\right) + \Delta t\, f(x_i, t^n), \quad i = 1, \ldots, N-2,$$

imponiendo en cada paso temporal

$$u_0^{n+1} = \alpha(t^{n+1}), \quad u_{N-1}^{n+1} = \beta(t^{n+1}),$$

y con condición inicial

$$u_i^0 = u_0(x_i), \quad i = 0, \dots, N-1.$$

Convergencia y estabilidad en FD. El esquema FD para Poisson y Laplace es consistente de orden $O(h^2)$ en espacio. Para la ecuación del calor, el esquema explícito tiene orden de consistencia $O(h^2)$ en espacio y $O(\Delta t)$ en tiempo. Por el teorema de Lax, un esquema lineal que sea consistente y estable (en el sentido de Von Neumann) converge a la solución exacta cuando $h, \Delta t \to 0$. Para garantizar la estabilidad del método explícito se debe comprobar que $r \leq 1/2$, coincidiendo con el criterio CFL estándar.

3.2.2. Elementos finitos (FEM)

El método de elementos finitos en 1D parte de la formulación débil de la ecuación diferencial y de una discretización equiespaciada de $[a, b]$ idéntica a la del método de diferencias finitas, en $N-1$ subintervalos de longitud

$$h = \frac{b-a}{N-1}, \qquad x_i = a + i\,h, \quad i = 0, \dots, N-1.$$

Se construye por tanto el espacio finito

$$V_h = \operatorname{span}\{\varphi_0, \dots, \varphi_{N-1}\} \subset H^1(a, b),$$

donde cada φ_i es la función *hat* lineal que vale 1 en x_i y 0 en los demás nodos.

Problema de Poisson y matriz de rigidez

Consideremos

$$\begin{cases} -u''(x) = f(x), & x \in [a, b], \\ u(a) = \alpha, \quad u(b) = \beta. \end{cases}$$

La formulación débil consiste en buscar $u \in H^1(a, b)$ con $u(a) = \alpha$, $u(b) = \beta$ tal que $\forall v \in H_0^1(a, b)$,

$$\int_a^b u'(x)\, v'(x)\, dx \;=\; \int_a^b f(x)\, v(x)\, dx.$$

Aproximamos $u(x) \approx u_h(x) = \sum_{j=0}^{N-1} U_j\, \varphi_j(x)$, con $U_0 = \alpha$, $U_{N-1} = \beta$. Para $i = 1, \dots, N-2$, imponemos

$$\int_a^b \Big(\sum_{j=0}^{N-1} U_j\, \varphi_j'(x)\Big)\, \varphi_i'(x)\, dx = \int_a^b f(x)\, \varphi_i(x)\, dx.$$

Definimos la *matriz de rigidez* $A \in \mathbb{R}^{N\times N}$ con entradas

$$A_{ij} = \int_a^b \varphi_j'(x)\,\varphi_i'(x)\,dx.$$

Dado que φ_i' es constante en cada subintervalo, se obtiene la forma tridiagonal

$$A_{i,i-1} = -\tfrac{1}{h}, \quad A_{i,i} = \tfrac{2}{h}, \quad A_{i,i+1} = -\tfrac{1}{h}, \qquad i = 1,\dots,N-2,$$

y $A_{i,j} = 0$ si $|i-j| > 1$. Para imponer Dirichlet, las filas $i = 0$ y $i = N-1$ se convierten en vectores unitarios:

$$A_{0,0} = 1,\ A_{0,j} = 0\ (j \neq 0), \quad A_{N-1,N-1} = 1,\ A_{N-1,j} = 0\ (j \neq N-1).$$

El *vector de carga* $\mathbf{b} \in \mathbb{R}^N$ tiene componentes

$$b_i = \int_a^b f(x)\,\varphi_i(x)\,dx, \quad i = 1,\dots,N-2,$$

y $b_0 = \alpha,\ b_{N-1} = \beta$. Cada b_i se calcula aproximando siguiendo la regla de Simpson en $[x_{i-1}, x_{i+1}]$

$$\int_{x_{i-1}}^{x_{i+1}} f(x)\,\varphi_i(x)\,dx \approx \frac{h}{6}\,f(x_{i-1}) + \frac{4h}{6}\,f(x_i) + \frac{h}{6}\,f(x_{i+1}),$$

En consecuencia, el sistema lineal para Poisson es

$$A\,\mathbf{U} = \mathbf{b}, \quad \mathbf{U} = (U_0,\dots,U_{N-1})^T, \quad \mathbf{b} = (b_0,\dots,b_{N-1})^T.$$

Ecuación del calor: matrices de masa y rigidez

Para

$$\begin{cases} u_t - u_{xx} = f(x,t), & x \in [a,b],\ 0 < t \le T, \\ u(a,t) = \alpha(t),\ u(b,t) = \beta(t), & 0 < t \le T, \\ u(x,0) = u_0(x), & x \in [a,b], \end{cases}$$

buscamos $u_h(x,t) = \sum_{j=0}^{N-1} U_j(t)\,\varphi_j(x)$. Al multiplicar por φ_i y aplicar integración por partes en el espacio, se definen:

$$M_{ij} = \int_a^b \varphi_j(x)\,\varphi_i(x)\,dx, \qquad K_{ij} = \int_a^b \varphi_j'(x)\,\varphi_i'(x)\,dx, \quad i,j = 0,\dots,N-1.$$

En 1D con base lineal, las entradas no nulas de M y K son:

$$M_{i,i} = \frac{2h}{3}, \quad M_{i,i-1} = \frac{h}{6}, \quad M_{i,i+1} = \frac{h}{6}, \quad i = 1,\dots,N-2,$$

$$K_{i,i} = \frac{2}{h}, \quad K_{i,i-1} = -\frac{1}{h}, \quad K_{i,i+1} = -\frac{1}{h}, \quad i = 1,\dots,N-2,$$

con filas de identidad para $i = 0$ y $i = N-1$ a fin de imponer Dirichlet:

$$M_{0,0} = 1,\ M_{0,j} = 0\ (j \neq 0), \quad M_{N-1,N-1} = 1,\ M_{N-1,j} = 0\ (j \neq N-1),$$

$$K_{0,0} = 1,\ K_{0,j} = 0\ (j \neq 0), \quad K_{N-1,N-1} = 1,\ K_{N-1,j} = 0\ (j \neq N-1).$$

El *vector de carga temporal* $\mathbf{F}(t)$ tiene componentes

$$F_i(t) = \int_a^b f(x,t)\,\varphi_i(x)\,dx, \quad i = 1, \ldots, N-2,$$

aproximándose de nuevo por el método de Simpson en $[x_{i-1}, x_{i+1}]$ por

$$F_i(t) \approx \frac{h}{6}\,f(x_{i-1},t) + \frac{4h}{6}\,f(x_i,t) + \frac{h}{6}\,f(x_{i+1},t),$$

y $F_0(t) = \alpha(t),\ F_{N-1}(t) = \beta(t)$.

Esquema implícito de Euler en el tiempo

Sean

$$t^n = n\,\Delta t \quad n = 0, \ldots, N_t \quad \mathbf{U}^n = (U_0(t^n), \ldots, U_{N-1}(t^n))^T$$

Aproximando

$$\partial_t u_h(x, t^{n+1}) \approx \frac{u_h(x,t^{n+1}) - u_h(x,t^n)}{\Delta t}.$$

la formulación variacional lleva al sistema

$$\left(M + \Delta t\,K\right)\mathbf{U}^{n+1} \;=\; M\,\mathbf{U}^n \;+\; \Delta t\,\mathbf{F}^{n+1},$$

con las condiciones $U_0^{n+1} = \alpha(t^{n+1}),\ U_{N-1}^{n+1} = \beta(t^{n+1})$ impuestas en las filas correspondientes en $M + \Delta t\,K$ y en el vector derecho.

Convergencia y estabilidad

- Para el problema de Poisson o Laplace, el teorema de Céa garantiza que la aproximación u_h queda tan cerca de la solución exacta u como lo permita el mejor elemento de V_h:

  $$\|u - u_h\|_{H^1(a,b)} \;\leq\; C \inf_{v_h \in V_h} \|u - v_h\|_{H^1(a,b)}.$$

  Con funciones base lineales en 1D esto implica que, al refinar la malla ($h \to 0$), el error en norma H^1 decrece como $O(h)$ y, en norma L^2, como $O(h^2)$.

- Para la ecuación del calor, al combinar Euler implícito en el tiempo con FEM en el espacio, el método resulta incondicionalmente estable, es decir, no impone ninguna restricción del tipo $\Delta t \leq C\, h^2$. Bajo hipótesis razonables de regularidad para u y f, la aproximación satisface

$$\max_{0\leq n\leq N_t} \|u(\cdot,t^n) - u_h(\cdot,t^n)\|_{L^2(a,b)} = O\big(h^2 + \Delta t\big),$$

lo que indica que el error en L^2 se reduce cuadráticamente al refinar la malla y linealmente al disminuir el paso temporal. De forma similar,

$$\|u(\cdot,t^n) - u_h(\cdot,t^n)\|_{H^1(a,b)} = O\big(h + \sqrt{\Delta t}\big),$$

lo que refleja que, en norma H^1, el método converge de orden 1 en el espacio y de orden $1/2$ en el tiempo. En la práctica, esto significa que la precisión aumenta al hacer h y Δt más pequeños aunque sea de forma independiente, sin vincular estrictamente ambos parámetros.

3.3. Fundamentos de las PINNs

Las *Physics-Informed Neural Networks* (PINNs) aproximan la solución $u(x)$ (o $u(x,t)$) de una EDP mediante una red neuronal profunda u_θ con parámetros θ que realiza el siguiente mapeo

$$u_\theta : \mathbb{R}^2 \to \mathbb{R}, \quad (x,t) \mapsto u_\theta(x,t).$$

El enfoque consiste en definir una función de pérdida que penalice simultáneamente: (i) el residuo de la EDP en puntos interiores del dominio, (ii) la condición inicial y (iii) las condiciones de contorno. Es decir, el objetivo es que u_θ verifique:

$$u_t - u_{xx} = f(x,t) \quad \text{en } \Omega\times(0,T], \quad u(x,0) = u_0(x), \quad u|_{\partial\Omega} = g(x,t).$$

Arquitectura y función de activación

El aproximador u_θ se construye como una red *feed-forward* de L capas ocultas, donde cada capa realiza una combinación lineal seguida de una activación suave:

$$h^{(\ell+1)} = \sigma\big(W^{(\ell)}\, h^{(\ell)} + b^{(\ell)}\big), \quad \ell = 0,\dots,L-2,$$
$$u_\theta(x,t) = w^{(L-1)} \cdot h^{(L-1)} + b^{(L-1)}.$$

- $h^{(0)} = [x,\, t]^T \in \mathbb{R}^2$ es el vector de entrada; $h^{(\ell)} \in \mathbb{R}^{n_\ell}$ es la activación de la capa ℓ.
- $W^{(\ell)} \in \mathbb{R}^{n_{\ell+1}\times n_\ell}$ es la matriz de pesos que conecta la capa ℓ (con n_ℓ neuronas) a la capa $\ell+1$ (con $n_{\ell+1}$ neuronas).
- $b^{(\ell)} \in \mathbb{R}^{n_{\ell+1}}$ es el vector de sesgos (*bias*) de la capa $\ell+1$.

- $\sigma(z) = \text{SiLU}(z) = \frac{z}{1+e^{-z}}$ es la activación *sigmoid-linear* 4 aplicada elemento a elemento, elegida por ser suave y poseer derivadas continuas.
- En la capa de salida, $w^{(L-1)} \in \mathbb{R}^{1\times n_{L-1}}$ y $b^{(L-1)} \in \mathbb{R}$ producen la combinación lineal final sin activación, por lo que $u_\theta(x,t)$ es un escalar.

De este modo, u_θ es una función continuamente diferenciable en (x,t), lo cual permite definir y evaluar derivadas parciales $\partial_t u_\theta$ y $\partial_{xx} u_\theta$.

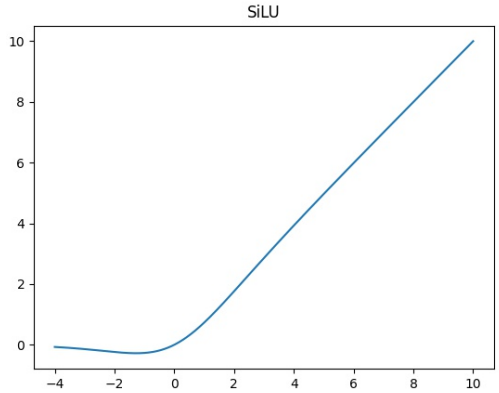

Figura 4: Función de activación SiLU

Definición de la función de pérdida

Se eligen tres conjuntos de puntos:

$$\{(x_j^r, t_j^r)\}_{j=1}^{N_r} \subset \Omega \times (0,T] \quad \text{(puntos interiores para PDE)},$$

$$\{x_k^0\}_{k=1}^{N_0} \subset \Omega \quad \text{(puntos iniciales en } t=0),$$

$$\{t_\ell^{bc}\}_{\ell=1}^{N_{bc}} \subset [0,T] \quad (\textit{instantes} \text{ para frontera } x = x_{\text{mín}},\, x_{\text{máx}}).$$

Denotamos $g_{\text{mín}}(t)$ y $g_{\text{máx}}(t)$ los valores de contorno en $x_{\text{mín}}$ y $x_{\text{máx}}$, respectivamente. Para cada tipo de punto, se define un término de pérdida como promedio de errores cuadráticos:

$$L_{pde}(\theta) = \frac{1}{N_r}\sum_{j=1}^{N_r}\left[\partial_t u_\theta(x_j^r, t_j^r) \;-\; \partial_{xx} u_\theta(x_j^r, t_j^r) \;-\; f(x_j^r, t_j^r)\right]^2,$$

$$L_{ic}(\theta) = \frac{1}{N_0}\sum_{k=1}^{N_0}\left[u_\theta(x_k^0, 0) - u_0(x_k^0)\right]^2,$$

$$L_{bc}(\theta) = \frac{1}{N_{bc}}\sum_{\ell=1}^{N_{bc}}\Big[\big(u_\theta(x_{\text{mín}}, t_\ell^{bc}) - g_{\text{mín}}(t_\ell^{bc})\big)^2 + \big(u_\theta(x_{\text{máx}}, t_\ell^{bc}) - g_{\text{máx}}(t_\ell^{bc})\big)^2\Big].$$

La pérdida total se define como combinación lineal:

$$\mathcal{L}(\theta) = w_{pde}\, L_{pde}(\theta) + w_{ic}\, L_{ic}(\theta) + w_{bc}\, L_{bc}(\theta),$$

donde $w_{pde}, w_{ic}, w_{bc} > 0$ son factores de peso que permiten equilibrar la contribución de cada término.

Interpretación

- El término L_{pde} fuerza que u_θ satisfaga $\partial_t u - \partial_{xx} u = f$ en media sobre los N_r puntos interiores, incorporando la física sin necesidad de un mallado.
- El término L_{ic} asegura que la aproximación coincida con la condición inicial impuesta $u_0(x)$.
- El término L_{bc} garantiza la imposición de los valores de contorno $g_{\text{mín}}, g_{\text{máx}}$ a lo largo de todos los instantes de tiempo seleccionados.

Minimizando $\mathcal{L}(\theta)$ se obtiene un θ^* tal que u_{θ^*} aproxima la solución de la ecuación del calor, respetando simultáneamente la EDP, la condición inicial y las condiciones de contorno, todo ello sin recurrir necesariamente a una malla discreta del dominio.

En conclusión, el marco teórico expuesto establece las bases matemáticas de las EDP en formulación fuerte y débil, detalla los fundamentos de FD y FEM (incluyendo convergencia y estabilidad) y presenta los principios esenciales de PINNs.

4. Metodología

En este apartado se describe de forma concisa el procedimiento seguido para implementar y comparar los métodos clásicos, la PINN y la combinación de ambos. Se incluye también una explicación de los hiperparámetros que pueden ser modificados en el `main` de cada uno de los archivos python. Además en cada archivo se incluyen diversas formas de visualizar el proceso de entrenamiento y la solución final, empleando las librerías de `matplotlib` y `plotly`.

4.1. Implementación de métodos clásicos (`traditional.py`)

Para los métodos tradicionales se parte de la formulación discreta de la EDP sobre una malla equiespaciada en $[a, b]$. En FD, se elige un número de puntos N en x, se define $h = (b-a)/(N-1)$ y se aplica un esquema explícito de Euler hacia adelante en el tiempo con diferencias centradas de segundo orden en el espacio, asegurando la condición CFL ($r = \Delta t/h^2 \leq 1/2$). En FEM, se toma el mismo mallado, se construyen las matrices de rigidez y masa usando funciones base lineales, y se emplea un esquema implícito de Euler en

el tiempo para el sistema $(M + \Delta t\, K)\, U^{n+1} = M\, U^n + \Delta t\, F$.

En ambos casos, tras ensamblar la matriz y el vector de carga, se resuelve el sistema lineal resultante y se evalúa el MSE del residuo de la ecuación en la propia malla (*ongrid*) y, también en puntos interpolados a partir de la malla *offgrid*. Esto se hará bien mediante interpolación simple bilineal y diferencias centradas, o bien con *splines* cúbicos que aportan mayor precisión. Es decir, para cada solución numérica se interpola en una malla fina $\{(x_i, t_j)\}$ y se utiliza un procedimiento de dos etapas para estimar $\partial_{xx} u$ y $\partial_t u$. La métrica de interés es

$$\mathrm{MSE}_{\mathrm{off}} \;=\; \frac{1}{M} \sum_{k=1}^{M} \Big[\, u_t(x_k, t_k) - u_{xx}(x_k, t_k) - f(x_k, t_k) \Big]^2,$$

donde $\{(x_k, t_k)\}$ son M puntos muestreados de forma equiespaciada en el dominio. Esta métrica se empleará también para las PINNs, cómo veremos en los Experimentos 5.

Los parámetros que controlan estos métodos, en el mismo orden en que aparecen al configurar la llamada, son:

- Intervalo espacial en x, es decir $[a, b]$.
- Diferencial de tiempo Δt.
- Tiempo máximo $t_{\mathrm{máx}}$.
- Condiciones de contorno Dirichlet $\{g_{\mathrm{mín}}(a),\, g_{\mathrm{máx}}(b)\}$ o $\{g_{\mathrm{mín}}(a, t),\, g_{\mathrm{máx}}(b, t)\}$.
- Número de puntos (*mesh_points*) en los que se divide la componente espacial.
- Función fuente $f(x)$ o $f(x, t)$ según corresponda.
- Función de condición inicial $u_0(x)$ (solo para la ecuación del calor).
- Método a emplear, diferencias finitas o elementos finitos
- Número de muestras para calcular el MSE

4.2. Implementación de la PINN (`pinn.py`)

Para evaluar el residuo $\mathcal{N}[u_\theta](x, t)$ ($u_t - \alpha\, u_{xx}$ en el caso de la ecuación del calor), se emplea diferenciación automática. Así, $\partial_t u_\theta$ y $\partial_{xx} u_\theta$ se obtienen llamando a la rutina de `autograd.grad`, evitando cálculos simbólicos.[13].

La red u_θ es una red *feed-forward* (perceptrón multicapa) que emplea funciones de activación suaves en las capas ocultas como SiLU o tanh, decantándonos finalmente por la primera, y una salida lineal. Se eligen los siguientes hiperparámetros en orden de aparición en el código del archivo `pinn.py`:

- Condiciones de contorno Dirichlet $\{g_{\text{mín}}(a),\, g_{\text{máx}}(b)\}$ o $\{g_{\text{mín}}(a,t),\, g_{\text{máx}}(b,t)\}$.
- Función fuente $f(x)$ o $f(x,t)$ según corresponda.
- Función de condición inicial $u_0(x)$ (solo para la ecuación del calor).
- Intervalo espacial en x, es decir $[a,b]$.
- Diferencial de tiempo Δt.
- Tiempo máximo $t_{\text{máx}}$
- Número de puntos (*mesh points*) para la componente espacial.
- Porcentaje de datos (porcentaje del dominio) con el que se entrena la red.
- Multiplicadores para incluir más datos específicamente en condiciones de contorno o en la condición inicial.
- Tipo de dominio (*sawtooth*, senoidal , onda cuadrada).
- Modo de muestreo de datos: equiespaciado o aleatorio.
- Parámetros (amplitud y periodo) de las funciones que definen el dominio variante $(g_{\text{mín}},\, g_{\text{máx}})$.
- Número de capas L junto con número de neuronas n_ℓ en cada capa ℓ.
- Tasa de aprendizaje.
- Número de épocas de entrenamiento.
- Ponderaciones de los términos de la función de pérdida.

Dado el conjunto de puntos interiores y puntos de contorno/inicial, la función de pérdida los combina cómo hemos visto anteriormente en la sección 3.3

$$\mathcal{L}(\theta) = w_{pde}\, L_{pde}(\theta) + w_{ic}\, L_{ic}(\theta) + w_{bc}\, L_{bc}(\theta),$$

donde $w_{pde}, w_{ic}, w_{bc} > 0$ son las ponderaciones. Esta función se minimiza con un optimizador estocástico como Adam.

4.3. Comparación de ambos enfoques (`combination.py`)

Los hiperparámetros a elegir son los mismos que en los dos apartados anteriores.

- *Solución clásica de referencia:* Se obtiene u_c con FD o FEM en una malla fina, midiendo el tiempo de ensamblado y resolución.
- *Predicción con PINN:* La red entrenada se evalúa en los mismos nodos de la malla clásica, registrando el tiempo de predicción.
- *Medición de errores y tiempos:* Se compara el MSE del residuo en la malla y el coste computacional (entrenamiento frente a ensamblado y solución directa) para distintos números de puntos interiores y épocas de entrenamiento de la PINN.

5. Experimentos

En primer lugar, cabe destacar que para el desarrollo de todos los experimentos se han definido semillas fijas (Python 12.2) con el objetivo de permitir la reproducibilidad del código. Se emplearon las librerías `torch`, `numpy`, `scipy`, `matplotlib` y `plotly`, como se indica en el archivo de `requirements.txt`

5.1. Objetivos experimentales

La ecuación en la que nos centraremos será la ecuación del calor unidimensional, que es en la que se han basado nuestros análisis matemáticos y experimentos. Además, salvo que se especifique lo contrario en algún experimento puntual, los intervalos de espacio y tiempo serán siempre los siguientes,

$$u_t \;-\; u_{xx} \;=\; f(x,t), \quad (x,t) \in [0,4] \times [0,2],$$

junto con las condiciones de contorno homogéneas $u(0,t) = 0$, $u(4,t) = 0$ y la condición inicial

$$u(x,0) \;=\; u_0(x), \qquad u_0(x) \;=\; \sin\!\left(\tfrac{\pi}{2}x\right).$$

Por otro lado, la fuente también permanecerá invariante en nuestros análisis y se define como

$$f(x,t) \;=\; \sin\!\left(\tfrac{\pi}{2}x\right)\, \exp(-10\,t).$$

Por último, la arquitectura de la PINN será siempre $\big[2,\ 10,\ 10,\ 1\big]$ con activación SiLU en las capas ocultas y salida lineal, y con una tasa de aprendizaje (*learning rate - lr*) de 10^{-2}. Estos parámetros se mantienen constantes puesto que al aumentar el número de capas, aumenta el tiempo de cómputo sin observar una mejora en la predicción. Lo mismo ocurre con la tasa de aprendizaje, el modelo se comporta muy bien al resolver la ecuación sin necesidad de un ajuste fino del *lr*. Para que el modelo aprenda más, se puede simplemente entrenar

durante más épocas.

A pesar de que el código está diseñado para soportar: cualquier función regular para la fuente $f(x,t)$ y para la condición inicial $u_0(x)$, cualquier arquitectura de red, y cualquier intervalo de tiempo y espacio, los experimentos se han centrado en comparar otros aspectos, manteniendo los anteriores invariantes. Por tanto, los objetivos de los experimentos son:

1. **Comparar la precisión de los métodos clásicos (FD y FEM)** mediante el cálculo del MSE *ongrid* y *offgrid* con 400 puntos tanto con interpolación bilineal como con *splines* cúbicos, junto con los tiempos de cómputo necesarios para cada método, en función de la refinación espacial y temporal.

2. **Evaluar la convergencia de la PINN** con el error MSE al variar el porcentaje de datos y las épocas de entrenamiento, tanto con muestreos equiespaciados como aleatorios. Mostrar la evolución de la función de pérdida $\mathcal{L}(\theta)$ con sus tres componentes $w_{pde}\,L_{pde}(\theta), w_{ic}\,L_{ic}(\theta)$ y $w_{bc}\,L_{bc}(\theta)$.

3. Observar cómo se comporta la **PINN en dominios no constantes en el tiempo** que los métodos clásicos no pueden abordar, prestando atención a las variaciones en los pesos de la función de pérdida.

4. Visualizar en **gráficos 3D conjuntamente la solución obtenida por los métodos clásicos y la aproximación de la PINN**, comparando el error MSE.

5.2. Precisión de los métodos clásicos

En las siguientes tablas Tabla 1 y Tabla 2 podemos observar las métricas de tiempo y de MSE para los métodos de diferencias finitas y elementos finitos respectivamente. El número de puntos que aparece en las cabaceras de las columnas se corresponde con el número de combinaciones de instantes de tiempo (`time_steps` $= t_{\text{máx}}/\Delta t + 1$) y de puntos en el espacio o `mesh_points` para los cuales se está calculando el valor final de la función $u(x,t)$. Δt y `mesh_points` serán los parámetros que iremos modificando para lograr un número de puntos diferentes al multiplicar `time_steps` y `mesh_points`.

- $\Delta t = 0{,}001$, `mesh_points` $= 50$, `time_steps` $= 2001 \longrightarrow 2001 \times 50 = 100\,050$
- $\Delta t = 0{,}001$, `mesh_points` $= 100$, `time_steps` $= 2001 \longrightarrow 2001 \times 100 = 200\,100$
- $\Delta t = 0{,}0001$, `mesh_points` $= 100$, `time_steps` $= 20\,001 \longrightarrow 20\,001 \times 100 = 2\,000\,100$
- $\Delta t = 0{,}0001$, `mesh_points` $= 200$, `time_steps` $= 20\,001 \longrightarrow 20\,001 \times 200 = 4\,000\,200$

Número de Puntos	**100050**	**200100**	**2000100**	**4000200**
Tiempo (s)	0.2139	N/A (CFL)	5.1625	11.3656
MSE *ongrid*	0	N/A (CFL)	0	0
MSE *offgrid*	0.1016	N/A (CFL)	0.1027	0.0998
MSE *offgrid (splines)*	5.9081×10^{-7}	N/A (CFL)	3.7795×10^{-8}	2.6147×10^{-9}

Tabla 1: MSE medido para FD con 400 muestras con distintas densidades de mallado

Número de Puntos	**100050**	**200100**	**2000100**	**4000200**
Tiempo (s)	0.3981	0.8135	8.9064	21.1811
MSE *ongrid*	0	0	0	0
MSE *offgrid*	0.0991	0.1002	0.1002	0.0998
MSE *offgrid (splines)*	3.9894×10^{-5}	9.8188×10^{-6}	9.8314×10^{-6}	2.3822×10^{-6}

Tabla 2: MSE medido para FEM con 400 muestras con distintas densidades de mallado

El MSE *ongrid* es una métrica que no es representativa, puesto que se basa en evaluar la ecuación diferencial en exactamente los mismos puntos con los que se ha ejecutado el método correspondiente, devolviendo siempre un error nulo. El MSE *offgrid* es el que emplea interpolación bilineal, el cual comete graves errores con la segunda derivada, otorgándole valores 100 veces por encima de lo matemáticamente correcto. Es por ello que se ha implementado el MSE *offgrid* con interpolación a través de *splines* cúbicos, para garantizar una aproximación precisa de la primera y segunda derivada, y con ello, un resultado representativo del error. Para el resto de comparaciones de aquí en adelante, solo se empleará esta métrica, el MSE *offgrid* interpolado con *splines*

Sin necesidad de reducir drásticamente Δt, se observa que el MSE *offgrid* apenas disminuye al bajar Δt, mientras que sí mejora claramente al incrementar el número de `mesh_points`. Por tanto, usar un Δt muy pequeño solo aumenta el tiempo de cálculo sin aportar ganancia real en precisión a nivel de MSE.

Por otro lado, FEM requiere más tiempo de ejecución que FD, pero es importante destacar que funciona en cualquier caso sin tener que prestar atención a la condición CFL. Aunque en estos ejemplos su MSE está a la par o ligeramente por encima del método de diferencias finitas, su solidez hace que, si contamos con suficiente capacidad de cómputo, sea una opción más segura para garantizar estabilidad y convergencia en todo tipo de mallados aunque tarde algo más de tiempo. Además, gracias a la formulación débil, FEM permite trabajar con fuentes discontinuas o picudas, convirtiéndolo en un método más versátil.

5.3. Precisión de la PINN.

En este caso, la malla de referencia contiene 41 `mesh_points` y 21 `time_steps` ($\Delta t = 0{,}1$), dejando una malla de $41 \times 21 = 861$ puntos para el muestreo equiespaciado, y obteniendo 861 puntos en el muestreo aleatorio, aunque no en una malla equiespaciada como tal. A pesar de existir la posibilidad de incrementar de forma independiente el número de puntos específicamente en las condiciones de contorno o en la condición inicial, haremos uso de esto en la siguiente sección 5.4. Del mismo modo, las ponderaciones de la función de pérdida también se mantendrán en el caso básico hasta la siguiente sección $\{w_{\text{PDE}}, w_{\text{IC}}, w_{\text{BC}}\} = \{1, 1, 1\}$. Por tanto, los únicos parámetros que variarán en esta sección serán:

- **Porcentaje de datos de entrenamiento:** El porcentaje de datos empleados en el entrenamiento de la PINN del total de 861 puntos de los que se parte. Para el caso de muestreo equiespaciado, este porcentaje coincidirá con el porcentaje del dominio en el que se entrena la PINN, como se puede apreciar en la Figura 7. Sin embargo, para el muestreo aleatorio, siempre se cubrirá todo el dominio y solamente se reducirá la cantidad de puntos muestreados, como se puede observar en la Figura 8
- **Épocas de entrenamiento de la PINN**
- **Modos de muestreo:** equiespaciado y aleatorio (puntos sin estructura de malla).

En consecuencia, en las siguientes tablas Tabla 3 y Tabla 4, se muestra el MSE para cada combinación de porcentaje de datos y épocas, habiendo una tabla para cada modo de muestreo. Además, con estos mismos datos se han generado mapas de calor en escala logarítmica que muestran las regiones de mayor y menor error en función de esos parámetros.

% de datos	Épocas				
	400	**800**	**1200**	**1600**	**2000**
100	$2{,}239 \times 10^{-3}$	$1{,}190 \times 10^{-3}$	$7{,}728 \times 10^{-4}$	$5{,}059 \times 10^{-4}$	$3{,}286 \times 10^{-4}$
80	$8{,}400 \times 10^{-3}$	$3{,}667 \times 10^{-3}$	$3{,}078 \times 10^{-3}$	$2{,}483 \times 10^{-3}$	$1{,}772 \times 10^{-3}$
60	$1{,}402 \times 10^{-1}$	$8{,}135 \times 10^{-2}$	$4{,}265 \times 10^{-2}$	$1{,}965 \times 10^{-2}$	$1{,}054 \times 10^{-2}$
40	$3{,}392 \times 10^{-1}$	$2{,}773 \times 10^{-1}$	$2{,}333 \times 10^{-1}$	$2{,}106 \times 10^{-1}$	$2{,}047 \times 10^{-1}$
20	$6{,}489 \times 10^{-1}$	$6{,}327 \times 10^{-1}$	$5{,}658 \times 10^{-1}$	$4{,}929 \times 10^{-1}$	$4{,}394 \times 10^{-1}$

Tabla 3: MSE *offgrid* con *splines* para una PINN con diferentes porcentajes de datos **equiespaciados** y épocas de entrenamiento

% de datos	Épocas				
	400	**800**	**1200**	**1600**	**2000**
100	$8{,}447 \times 10^{-4}$	$3{,}266 \times 10^{-4}$	$2{,}043 \times 10^{-4}$	$1{,}630 \times 10^{-4}$	$1{,}423 \times 10^{-4}$
80	$2{,}257 \times 10^{-3}$	$9{,}117 \times 10^{-4}$	$6{,}622 \times 10^{-4}$	$4{,}999 \times 10^{-4}$	$1{,}800 \times 10^{-4}$
60	$1{,}321 \times 10^{-3}$	$3{,}557 \times 10^{-4}$	$2{,}302 \times 10^{-4}$	$2{,}001 \times 10^{-4}$	$1{,}774 \times 10^{-4}$
40	$1{,}842 \times 10^{-3}$	$7{,}857 \times 10^{-4}$	$5{,}741 \times 10^{-4}$	$4{,}717 \times 10^{-4}$	$3{,}666 \times 10^{-4}$
20	$6{,}820 \times 10^{-3}$	$3{,}662 \times 10^{-3}$	$2{,}120 \times 10^{-3}$	$1{,}815 \times 10^{-3}$	$1{,}704 \times 10^{-3}$

Tabla 4: MSE *offgrid* con *splines* para una PINN con diferentes porcentajes de datos **aleatorios** y épocas de entrenamiento

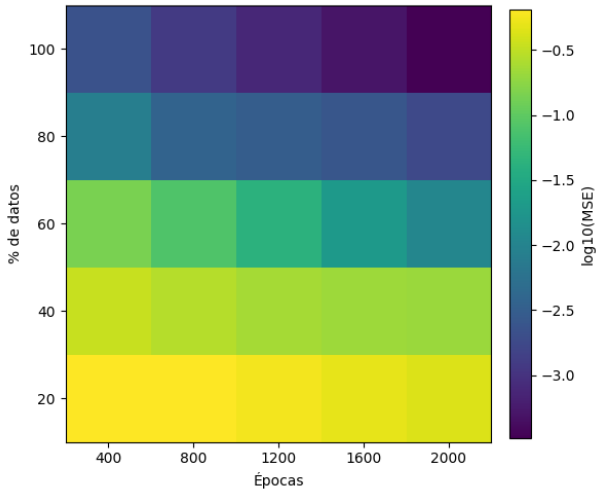

Figura 5: Mapa de calor del MSE *offgrid* con *splines* en escala logarítmica para una PINN con diferentes porcentajes de datos **equiespaciados** y épocas de entrenamiento

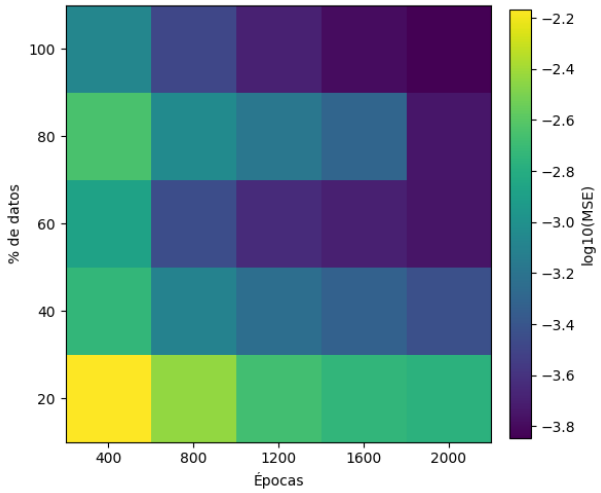

Figura 6: Mapa de calor del MSE *offgrid* con *splines* en escala logarítmica para una PINN con diferentes porcentajes de datos **aleatorios** y épocas de entrenamiento

Como cabría esperar, tanto en las Tablas 3 y 4 como en las Figuras 5 y 6 se aprecia claramente cómo el MSE disminuye a medida que aumentan tanto el porcentaje de datos empleados como el número de épocas de entrenamiento, sobre todo con el muestreo equiespaciado. A continuación se ofrece un análisis de cada tabla individualmente y cómo conjunto.

- **Muestreo equiespaciado (Tabla 3).**
  - Con el 100 % de datos, el error cae de $2{,}239 \times 10^{-3}$ a $3{,}286 \times 10^{-4}$ cuando las épocas pasan de 400 a 2000.
  - Para 80 %, el MSE inicial de $8{,}400 \times 10^{-3}$ se reduce a $1{,}772 \times 10^{-3}$ tras 2000 épocas, siguiendo la misma tendencia de mejora.

  - En porcentajes inferiores (60 %, 40 %, 20 %), el error permanece en $O(10^{-1})$ a pocas épocas y solo baja a $O(10^{-2})$ o $O(10^{-1})$ incluso tras 2000 épocas, lo que indica que con muestreo equiespaciado hace falta un porcentaje elevado de datos para lograr un error pequeño. Por debajo del 80 %, la red no dispone de suficiente información para reducir el error por debajo de $\sim 10^{-1}$.

- **Muestreo aleatorio (Tabla 4).**
  - Con el 100 % de datos, el MSE parte de $8{,}447 \times 10^{-4}$ a 400 épocas y llega a $1{,}423 \times 10^{-4}$ a 2000 épocas, mostrando mayor rapidez en la convergencia que en el caso equiespaciado.
  - Para 80 %, el error baja a $1{,}800 \times 10^{-4}$ tras 2000 épocas, notablemente mejor que los $1{,}772 \times 10^{-3}$ del muestreo equiespaciado.
  - Incluso con 60 % o 40 %, el MSE logra valores de orden 10^{-4} y 10^{-3} respectivamente al alcanzar 2000 épocas, lo que indica que el muestreo aleatorio permite a la red beneficiarse de la variabilidad espacial para generalizar mejor con menos datos. Es decir, no es que sea tan importante el número de datos, sino que estos estén distribuidos por todo el dominio y no solamente en una sección, como ocurre en el muestreo equiespaciado con porcentajes menores al 100 %.
  - Con tan solo 20 % de datos, el error se sitúa en $1{,}704 \times 10^{-3}$ a 2000 épocas; aunque mayor que con porcentajes superiores, sigue siendo mucho menor que en el muestreo equiespaciado con el mismo porcentaje.

- **Comparación general:**
  - El muestreo aleatorio consigue MSE de $O(10^{-4})$ con menos datos y menos épocas, mientras que el muestreo equiespaciado requiere mayor densidad de datos para alcanzar errores similares.
  - En ambos modos, a partir de aproximadamente 1600–2000 épocas la mejora en MSE se atenúa, lo cual sugiere que las redes están llegando a un límite impuesto por la cantidad de datos disponibles.
  - Queda demostrado que el muestreo aleatorio presenta ventajas significativas, ya que incluso cubriendo en ambos modos el 100 % del dominio y usando todos los datos, el muestreo aleatorio logra un menor MSE.

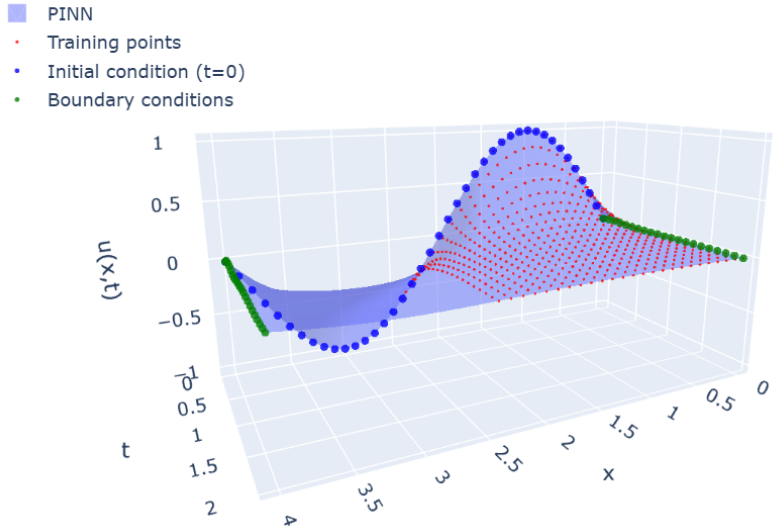

Figura 7: Evolución de $u(x,t)$ obtenida con una PINN entrenada durante 2000 épocas con un 60 % de los datos con muestreo equiespaciado alcanzando un MSE de $1{,}054 \times 10^{-2}$

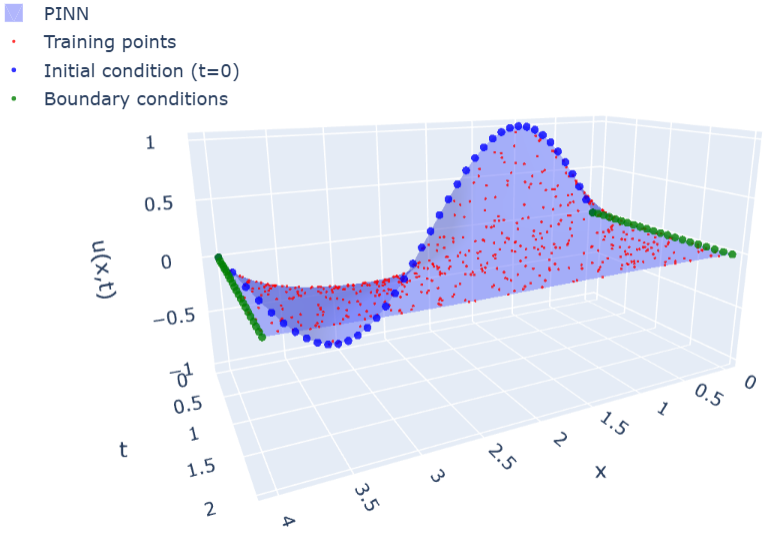

Figura 8: Evolución de $u(x,t)$ obtenida con una PINN entrenada durante 2000 épocas con un 60 % de los datos con muestreo aleatorio alcanzando un MSE de $1{,}774 \times 10^{-4}$

Siempre que se entrena una PINN quedan almacenados los datos de la evolución de la función de pérdida $\mathcal{L}(\theta)$. A continuación, en las Figuras 9 y 10 se muestra dicha evolución en escala logarítmica para los casos entrenados de las Figuras 7 y 8 respectivamente. A pesar de las acusadas oscilaciones, sobre todo en la segunda gráfica, en líneas generales la función de pérdida decrece, por lo que la red está entrenando y convergiendo de forma equilibrada.

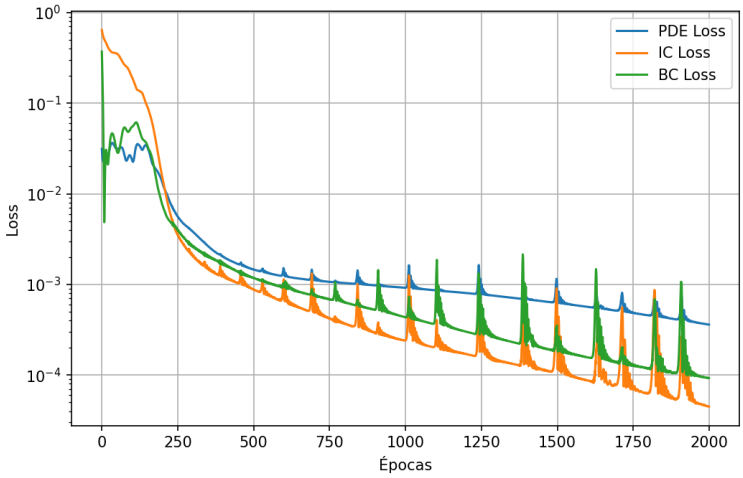

Figura 9: Evolución por épocas de la función de pérdida $\mathcal{L}(\theta)$ con sus tres componentes $w_{pde}\, L_{pde}(\theta), w_{ic}\, L_{ic}(\theta)$ y $w_{bc}\, L_{bc}(\theta)$ para el caso de la Figura 7

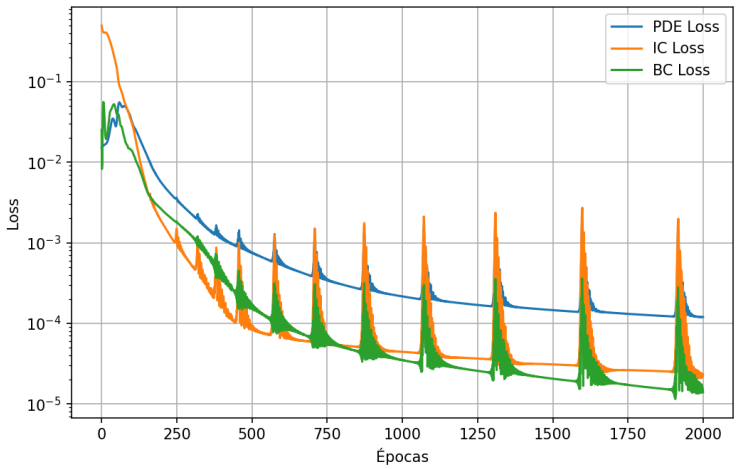

Figura 10: Evolución por épocas de la función de pérdida $\mathcal{L}(\theta)$ con sus tres componentes $w_{pde}\, L_{pde}(\theta), w_{ic}\, L_{ic}(\theta)$ y $w_{bc}\, L_{bc}(\theta)$ para el caso de la Figura 8

5.4. PINN en dominios variantes en el tiempo

Como ya se ha mencionado previamente, una de las ventajas más sorprendentes de las PINNs es que pueden entrenarse sobre puntos aleatorios en lugar de mallas equiespaciadas, y esto cobra especial relevancia en dominios variantes. Al permitir que los puntos interiores de entrenamiento se distribuyan de forma no estructurada, la red aprende la solución aun cuando el contorno del dominio cambia según una función compleja. Siendo A la amplitud (desplazamiento máximo del borde) y λ el periodo, para este apartado consideramos tres tipos de fronteras variables definidas por:

- **Dominio senoidal:**

$$g_{\text{mín}}(t) \;=\; x_0 + A\,\sin\!\big(2\pi\,t/\lambda\big), \quad g_{\text{máx}}(t) \;=\; x_L + A\,\sin\!\big(2\pi\,t/\lambda\big).$$

  Aquí la frontera oscila suavemente entre $x_0 - A$ y $x_0 + A$ con periodo λ.

- **Dominio sawtooth:**

$$g_{\text{mín}}(t) \;=\; x_0 + A\,\Big(2\big(\{\,t/\lambda\,\} - 0{,}5\big)\Big), \quad g_{\text{máx}}(t) \;=\; x_L + A\,\Big(2\big(\{\,t/\lambda\,\} - 0{,}5\big)\Big).$$

  Con $\{\,t/\lambda\,\} = t/\lambda - \lfloor t/\lambda \rfloor$. En cada periodo λ, $\{t/\lambda\}$ recorre $[0, 1)$, de modo que

$$2\big(\{\,t/\lambda\,\} - 0{,}5\big) \quad \text{crece linealmente de } -1 \text{ a } +1.$$

- **Dominio *square wave*:**

$$g_{\text{mín}}(t) \;=\; x_0 + A\begin{cases} +1, & \{\,t/\lambda\,\} < 0{,}5, \\ -1, & \{\,t/\lambda\,\} \geq 0{,}5, \end{cases} \quad g_{\text{máx}}(t) \;=\; x_L + A\begin{cases} +1, & \{\,t/\lambda\,\} < 0{,}5, \\ -1, & \{\,t/\lambda\,\} \geq 0{,}5. \end{cases}$$

  Aquí la frontera salta entre $x_0 - A$ y $x_0 + A$ cada media onda $(\lambda/2)$, permaneciendo constante en cada tramo.

A la hora de entrenar la PINN en estos dominios no constantes, se vuelve crucial:

- Colocar un mayor número de puntos sobre las condiciones de contorno móviles $g_{\text{mín}}(t)$ y $g_{\text{máx}}(t)$. Ya que las fronteras varían con t, necesitamos hacer un muestreo más denso allí para que la red capture bien la geometría. Lo controlamos con un multiplicador de los puntos que haya en el contorno.

- Asignar más peso al término de pérdida correspondiente a las condiciones de contorno (L_{bc}) en la función total

$$\mathcal{L}(\theta) \;=\; w_{pde}\,L_{pde} \;+\; w_{ic}\,L_{ic} \;+\; w_{bc}\,L_{bc},$$

  de modo que la PINN preste especial atención a satisfacer las fronteras variables.

Como en este caso no disponemos de una malla equiespaciada de referencia, ya que el dominio se deforma continuamente, no podemos calcular un MSE *offgrid* convencional. En su lugar, mostramos:

1. El valor final de la función de pérdida $\mathcal{L}(\theta)$ tras el entrenamiento completo.

2. Representaciones gráficas 3D donde se superpone la aproximación de la PINN con los puntos de la frontera en cada instante t. Si la superficie predicha se ajusta sin saltos ni solapamientos a la región variante, inferimos que la PINN ha aprendido correctamente la forma del dominio y satisface la ecuación del calor.

A continuación presentamos algunos ejemplos visuales para cada uno de los tres tipos de frontera (*senoidal*, *sawtooth* y *square wave*). Hemos elegido A y λ de tamaño pronunciado ($A \approx 0{,}25$, $\lambda \approx 1$) para que los límites definidos por $g_{\text{mín}}(t)$ y $g_{\text{máx}}(t)$ resulten muy ondulados o dentados y la irregularidad resulte muy evidente. Pero cabe mencionar que, ajustando A y λ se pueden simular deformaciones más precisas, en las cuales solo necesitaremos aún más puntos de entrenamiento para capturarlas en posibles futuras implementaciones

Para el primer ejemplo, seleccionaremos el dominio *senoidal* y además, para hacer más notable el efecto de añadir puntos en la condición de contorno ($\times 3$) y aumentar el peso del término correspondiente en la función de pérdida ($\times 10$), modificaremos ligeramente el problema inicial. Por supuesto, ahora los contornos son funciones en vez de valores constantes de x, pero además en lugar de mantener las condiciones de contorno de Dirichlet homogéneas, vamos a hacer que tengan los siguientes valores $u(g_{\text{mín}}(t), t) = 0$ y $u(g_{\text{máx}}(t), t) = -1$ para cada instante de tiempo. Consecuentemente, para que la solución de la EDP sea compatible matemáticamente, debemos hacer que las condiciones de contorno y la condición inicial coincidan en las esquinas del dominio, es por ello que debemos reducir el espacio al intervalo $[0, 3]$.

En la Figura 11 podemos observar cómo la frontera amarilla del contorno no está verificando las condiciones impuestas, a pesar de acabar con una *loss* menor que en la Figura 12. Esto es debido a que en la segunda Figura al darle más peso a las condiciones de contorno en la función de pérdida, estamos esencialmente reescalando el término de la *loss*. Es decir, aunque realmente se puede apreciar en las gráficas que el segundo caso está respetando mejor las fronteras gracias a haber añadido más puntos en ellas y haberles dado mayor peso en la pérdida, la *loss* nos va a devolver un mayor valor. Este análisis sirve para resaltar que el valor de la función de pérdida no lo es todo en las redes neuronales, sino que también hay que saber interpretar el proceso de entrenamiento al completo.

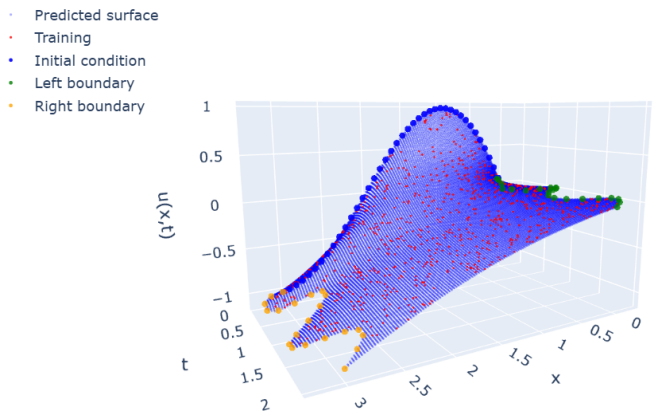

Figura 11: Evolución de $u(x, t)$ obtenida con una PINN entrenada durante 2000 épocas en un dominio *senoidal* con una *loss* final de $\mathcal{L}(\theta) = 1{,}1785 \times 10^{-2}$ con ponderaciones y número de puntos en las condiciones de contorno originales.

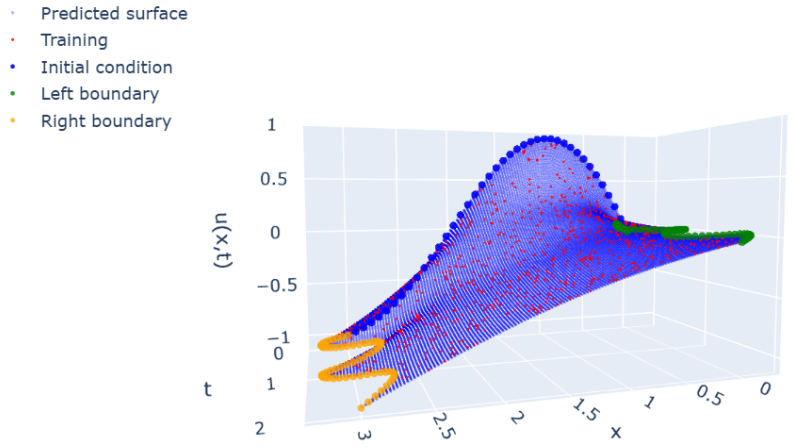

Figura 12: Evolución de $u(x,t)$ obtenida con una PINN entrenada durante 2000 épocas en un dominio *senoidal* con una *loss* final de $\mathcal{L}(\theta)$ =5,0352 $\times 10^{-2}$ ponderando $\times 10$ (L_{bc}) y número de puntos en las condiciones de contorno $\times 3$.

A continuación, volvemos a modificar el problema original, ahora en el intervalo [0.25, 4.25] y con $u(g_{\text{mín}}(t),t) = 0$ y $u(g_{\text{máx}}(t),t) = 0$ para cada instante de tiempo, mostramos dos ejemplos más con los dominios *sawtooth* y *square wave* en las Figuras 13 y 14 respectivamente. En ellas mantenemos un mayor número de puntos en los contornos ($\times 3$) y una mayor ponderación en la función de pérdida ($\times 10$). Como podemos observar en la función de pérdida y en las propias gráficas, en el intervalo seleccionado, la frontera *sawtooth* 13 se comporta bien, pero la frontera *square wave* es incompatible con la condición inicial. En este caso, la PINN intenta ajustarse sin éxito a ambas condiciones y queda una zona irregular 14.

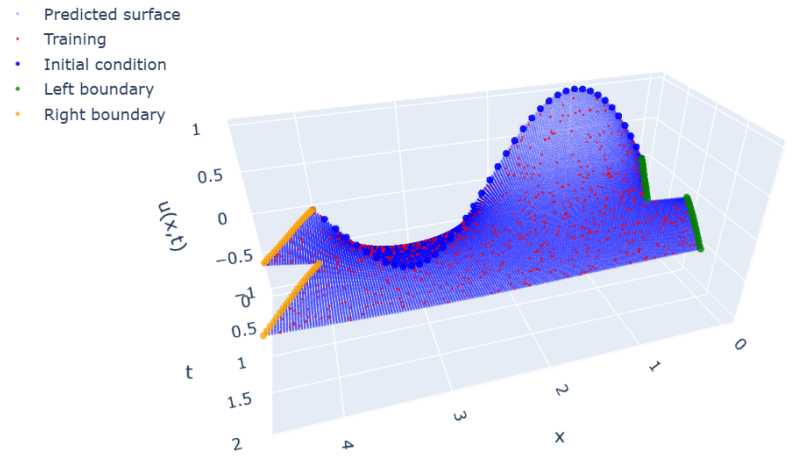

Figura 13: Evolución de $u(x,t)$ obtenida con una PINN entrenada durante 2000 épocas en un dominio *sawtooth* con una *loss* final de $\mathcal{L}(\theta)$ =8,55 $\times 10^{-4}$ ponderando $\times 10$ (L_{bc}) y número de puntos en las condiciones de contorno $\times 3$.

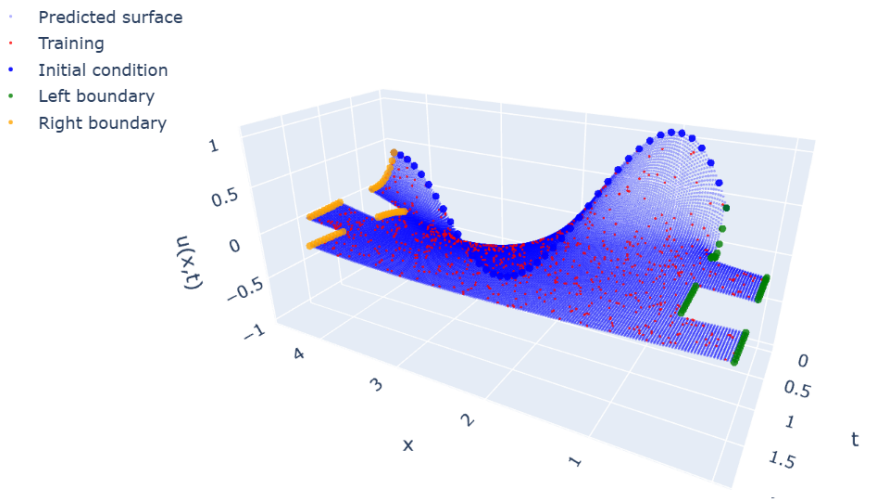

Figura 14: Evolución de $u(x,t)$ obtenida con una PINN entrenada durante 2000 épocas en un dominio *square wave* con una *loss* final de $\mathcal{L}(\theta)$ =2,5956 $\times 10^{-2}$ ponderando $\times 10$ (L_{bc}) y número de puntos en las condiciones de contorno $\times 3$.

En resumen, los resultados demuestran que:

- Con muestreo *aleatorio* y un número elevado de puntos en las fronteras $u(g_{\text{mín}}(t),t) = 0$ y $u(g_{\text{máx}}(t),t) = -1$, la PINN reproduce con fidelidad tanto la forma variante del dominio como la evolución en (x,t) de la temperatura. Hay que tener en cuenta el valor en $t = 0$ de la función que describe la frontera del dominio, y hacer que coincida con la condición inicial.
- Dar mayor *peso* a L_{bc} y reforzar con más puntos las condiciones de contorno mejora notablemente la estabilidad y la exactitud en los márgenes del dominio.
- Aunque no podemos cuantificar el error con un MSE clásico debido a no poseer una malla equiespaciada, la convergencia de $\mathcal{L}(\theta)$ a valores cada vez más pequeños y sobre todo, las gráficas 3D de la solución muestran que la PINN funciona correctamente en dominios que cambian con el tiempo.

5.5. Visualización conjunta de métodos clásicos y PINNs

En este apartado mostraremos dos gráficos 3D conjuntos de la solución obtenida por los métodos clásicos y la aproximación de la PINN, comparando el error MSE. Ya hemos comprobado que tanto diferencias finitas como elementos finitos consiguen una aproximación muy buena en términos de MSE. Sin embargo, para no tener que preocuparnos de la condición CFL tomaremos como referencia para realizar las comparaciones al método de elementos finitos, que es el más empleado de los métodos clásicos.

En la primera Figura 15, para el método clásico seleccionado, se grafica $u(x,t)$ como superficie 3D en una malla. Para la PINN, se añade sobre esa misma superficie los puntos con los que ha sido entrenada, que cómo podemos comprobar quedan ligeramente desviados de la superficie semitransparente del método clásico. Los parámetros son todos los definidos para el caso base, salvo el porcentaje de datos con los que entrenaremos que será 50 % con el muestreo equiespaciado. En la segunda Figura 16 se muestra la predicción que realiza la PINN en todo el dominio, en lugar de solamente mostrar el valor de los puntos con los que ha sido entrenada como en la figura anterior. Los resultados obtenidos son muy favorables hacia los métodos clásicos ya que tardan menos tiempo y consiguen un mejor MSE:

- Tiempo de entrenamiento de la PINN: 8.4840 segundos.
- Tiempo empleado por el método tradicional (FEM): 0.4376 segundos.
- MSE del método tradicional FEM: $5{,}448 \times 10^{-5}$
- MSE de la PINN con un 50 % de puntos con muestreo equiespaciado: $6{,}754 \times 10^{-2}$

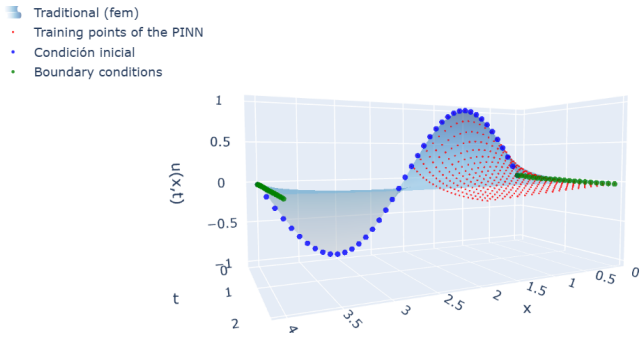

Figura 15: Evolución de $u(x,t)$ obtenida con el método FEM junto con los datos de entrenamiento (2000 épocas) de una PINN con un 50 % de los datos con muestreo equiespaciado

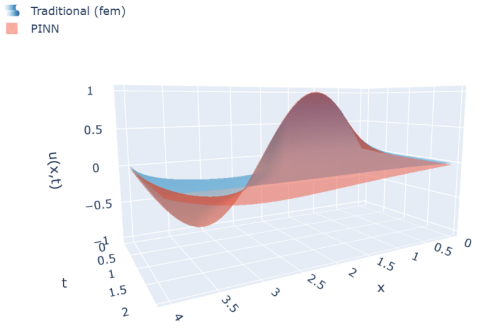

Figura 16: Evolución de $u(x,t)$ obtenida con el método FEM y con una PINN entrenada durante 2000 épocas con un 50 % de los datos con muestreo equiespaciado

Sin embargo, veamos lo que ocurre si entrenamos la PINN con el 100 % de los datos muestreados aleatoriamente y si densificamos ligeramente la malla del método tradicional (disminuyendo Δt a 0.0001 y aumentando los `mesh_points` a 100). Los resultados obtenidos se equilibran bastante y las figuras 17 y 18 muestran que la diferencia entre un método y otro es prácticamente imperceptible:

- Tiempo de entrenamiento de la PINN: 9.1558 segundos.
- Tiempo empleado por el método tradicional (FEM): 9.3588 segundos.
- MSE del método tradicional FEM: $1{,}355 \times 10^{-5}$
- MSE de la PINN con un 100 % de puntos con muestreo aleatorio: $1{,}204 \times 10^{-4}$

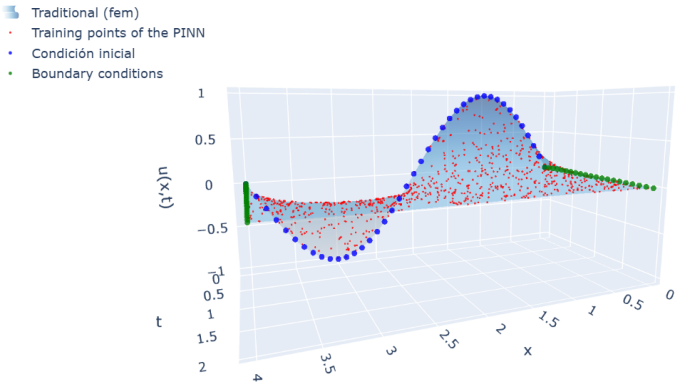

Figura 17: Evolución de $u(x,t)$ obtenida con el método FEM junto con los datos con los que una PINN entrenada durante 2000 épocas con un 100 % de los datos con muestreo aleatorio

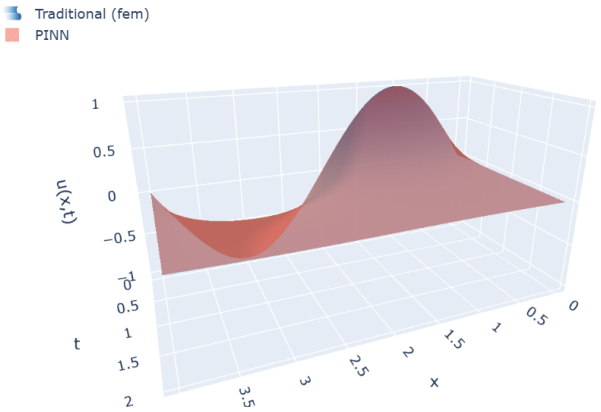

Figura 18: Evolución de $u(x,t)$ obtenida con el método FEM y con una PINN entrenada durante 2000 épocas con un 100 % de los datos con muestreo aleatorio

6. Resultados

En esta sección se presenta un análisis los experimentos descritos, comparando el comportamiento y la precisión de los métodos clásicos (FD y FEM) frente a las PINN.

6.1. Métodos clásicos

- **Diferencias finitas (FD).**
  - Con mallados moderados ($\Delta t = 10^{-3}$, 50 `mesh points`), FD resuelve el problema en 0.2s, alcanzando un MSE del orden de 10^{-7}.
  - Al refinar temporalmente ($\Delta t = 10^{-4}$) o espacialmente (100 `mesh points`), el tiempo crece hasta varios segundos, pero el MSE solo mejora ligeramente (10^{-8}–10^{-9}).
  - Por tanto, refinar Δt más allá de cierto umbral no justifica el coste adicional, ya que la principal reducción de error proviene de aumentar la densidad espacial.
- **Elementos finitos (FEM).**
  - FEM es más lento que FD con mallados equivalentes (0.4s vs 0.2s para 50 `mesh points`) y $\Delta t = 10^{-3}$), pero mantiene estabilidad sin restricciones de CFL.
  - Su MSE es peor, aunque comparable al de FD (10^{-6}–10^{-9}) para mallados finos.
  - Aumentar la resolución temporal o espacial incrementa el tiempo de cálculo a decenas de segundos, y no mejora demasiado el MSE.
- **Ventajas comparativas.**
  - Para problemas con dominio fijo y recursos limitados, FD ofrece la máxima velocidad con error muy pequeño.
  - Si se prefiere evitar restricciones de estabilidad o se cuenta con potencia de cómputo suficiente, FEM es más fiable.
  - En ambos casos, el uso de *splines* cúbicos para estimar derivadas *offgrid* es esencial para obtener errores representativos.

6.2. PINN en dominios constantes

- **Convergencia y muestreo.**
  - Con 861 puntos de referencia (malla 41×21) y arquitectura $[2, 10, 10, 1]$, la PINN tarda entre 8 y 9s en entrenamiento completo (2000 épocas).
  - Con muestreo equiespaciado, el MSE *offgrid* disminuye de 10^{-3} a 10^{-4} solamente si se usan al menos el 80 %–100 % de los puntos y alrededor de 2000 épocas. Con menos datos, el error se mantiene en torno a 10^{-1}.

  - Con muestreo aleatorio, la PINN aprovecha mejor la información espacial: alcanza MSE de 10^{-4} con entre 60 % y 80 % de puntos y 2000 épocas, y sigue mejorando ligeramente con 100 % de datos.
  - En resumen, el muestreo aleatorio logra convergencia más rápida y precisa con menos datos que el muestreo equiespaciado.

- **Comparación de eficiencia.**
  - Con 50 % de datos equiespaciados, la PINN alcanza un MSE del orden de 10^{-2} en 8.5s. Mientras, FEM sobre la misma malla tarda 0.44s con MSE del orden de 10^{-5}.
  - Con una malla más fina ($\Delta t = 10^{-4}$, 100 `mesh points`)), FEM invierte 9.4s con un MSE de $\approx 10^{-5}$, mientras que la PINN con 100 % de puntos aleatorios tarda 9.2s con un MSE $\approx 10^{-4}$.
  - Así, para mallados densos, la brecha temporal entre FEM y PINN se reduce, y ambas ofrecen soluciones muy similares en calidad.

6.3. PINN en dominios variantes en el tiempo

- Las PINN demuestran flexibilidad para resolver la ecuación del calor en fronteras variables (*senoidales*, *sawtooth* o *square wave*), donde los métodos clásicos no pueden usarse sin una estrategia de mallado más compleja.
- Con un adecuado reforzamiento en las regiones de frontera, multiplicando el número de puntos en $g_{\text{mín}}(t)$, $g_{\text{máx}}(t)$ y elevando el peso de L_{bc}, la PINN es capaz de aproximar con precisión la forma del dominio y la solución interior.
- Es de vital importancia a la hora de seleccionar la función $u_0(x)$, que determinará la condición inicial, y las condiciones de contorno, tener en cuenta la incompatibilidad que puede surgir si no coinciden en las esquinas del dominio. Esto se aplica tanto a dominios constantes en el tiempo como no constantes, aunque es más visible en estos últimos.
- Aunque no disponemos de un MSE *offgrid* estándar para estos dominios, la convergencia de $\mathcal{L}(\theta)$ a valores de 10^{-5}–10^{-6} y las gráficas 3D muestran que la PINN puede "encajar" la solución en formas no constantes en el tiempo.

7. Conclusiones y Trabajos Futuros

En este proyecto se ha estudiado de forma exhaustiva la teoría de las ecuaciones en derivadas parciales (EDP) y su resolución mediante métodos clásicos, diferencias finitas (FD) y elementos finitos (FEM), y redes neuronales físicamente informadas (PINN). Se abordó

primero un análisis matemático de las EDPs, incluyendo formulaciones fuertes y débiles, derivadas distribucionales y espacios de Sobolev, además de ejemplos estándar (Poisson, Laplace, calor). A continuación, se detallaron los fundamentos de los métodos numéricos tradicionales: se mostró cómo FD y FEM discretizan la EDP, garantizan convergencia y estabilidad, y se compararon sus ventajas en términos de precisión y robustez. Entre las aportaciones más destacadas se incluyen:

- La comprobación de que FD y FEM siguen siendo referencia en dominios constantes y sencillos: FD es muy eficiente en mallados equiespaciados, mientras que FEM, aunque más costoso, garantiza estabilidad sin restricción CFL.
- La validación de que el muestreo aleatorio mejora considerablemente la convergencia de la PINN con menos datos que el muestreo equiespaciado, lo que resalta la importancia de la distribución de puntos en la red.
- La demostración de que la PINN puede incorporar fronteras móviles (*senoides*, *saw-tooth*, *square wave*) sin definir mallas específicas, simplemente ajustando la función de pérdida y concentrando puntos en las condiciones de contorno variables.
- La generación de comparativas gráficas 3D que ilustran la coincidencia entre la solución FEM y la aproximación de la PINN, evidenciando que ambas alcanzan calidad similar cuando se refina la malla o se dispone de suficientes datos de entrenamiento.

El proyecto cumple por tanto con los objetivos propuestos: establecer un marco teórico sólido para las EDPs y sus métodos clásicos, comparar rigurosamente FD, FEM y PINN en dominios constantes, explorar el uso de PINN en dominios variantes y ofrecer herramientas de visualización unificadas. Como trabajos futuros se plantean varias direcciones:

- Extender el estudio a EDP de mayor complejidad, como sistemas de reacción-difusión o ecuaciones de Navier-Stokes, donde las PINN podrían aportar ventajas en problemas inversos y modelado de parámetros desconocidos.
- Realizar un estudio para obtener una regla respecto al número de puntos necesarios que hay que tomar para que la PINN converja a la solución de la EDP, teniendo en cuenta el número de puntos tomados en las fronteras, condiciones de contorno e inicial.
- Considerar dominios de más alta dimensionalidad (2D/3D), donde el coste de mallado de FD/FEM crece sustancialmente y las PINN ofrecen flexibilidad sin mallado explícito.
- Diseñar métodos híbridos que combinen aproximaciones clásicas y PINN, como usar soluciones FEM como pre-entrenamiento de la red, para acelerar la convergencia y mejorar la precisión en zonas complejas.

Estos desarrollos permitirían ampliar el alcance de las PINN a problemas más realistas y de dimensión superior, aprovechando su capacidad para integrar información física y manejar geometrías no constantes sin necesidad de mallado explícito.

8. Bibliografía

[1] Evans, L. C. (1994). Partial differential equations by Lawrence C. Evans. https://ci.nii.ac.jp/ncid/BA27007864

[2] Haberman, R. (1998). Elementary Applied partial differential equations: With Fourier Series and Boundary Value Problems.

[3] Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2018, 3 noviembre). A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. https://www.sciencedirect.com/science/article/

[4] Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2017, 28 noviembre). Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. arXiv.org. https://arxiv.org/abs/1711.10561

[5] Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2017, 28 noviembre). Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations. arXiv.org. https://arxiv.org/abs/1711.10566

[6] Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. (2018, 19 junio). Neural ordinary differential equations. arXiv.org. https://arxiv.org/abs/1806.07366

[7] Baty, H. (2024, 1 marzo). A hands-on introduction to Physics-Informed Neural Networks for solving partial differential equations with benchmark tests taken from astrophysics and plasma physics. arXiv.org. https://arxiv.org/abs/2403.00599

[8] Karniadakis, George and Kevrekidis, Yannis and Lu, Lu and Perdikaris, Paris and Wang, Sifan and Yang, Liu. (2021). Physics-informed machine learning. Nature Reviews Physics. https://www.nature.com/articles/s42254-021-00314-5

[9] Jhssyb. (s. f.). GitHub - jhssyb/PINN-master-Rassi: PINN to infer solutions to partial differential equations. GitHub. https://github.com/jhssyb/PINN-master-Rassi

[10] Haghighat, E., Raissi, M., Moure, A., Gomez, H., and Juanes, R. (2020, 14 febrero). A deep learning framework for solution and discovery in solid mechanics. arXiv.org. https://arxiv.org/abs/2003.027sol51

[11] Grossmann, T. G., et al. (2023). Can Physics-Informed Neural Networks beat the Finite Element Method? *IMA Journal of Numerical Analysis*.https://arxiv.org/abs/2302.04107

[12] Sacchetti, A., Bachmann, B., Löffel, K., Künzi, U.-M., & Paoli, B. (2022). Neural Networks to solve Partial Differential Equations: a Comparison with Finite Elements. https://arxiv.org/abs/2201.03269

[13] Sobh, N., Gladstone, R. J., & Meidani, H. (2025). PINN-FEM: A Hybrid Approach for Enforcing Dirichlet Boundary Conditions in Physics-Informed Neural Networks. https://arxiv.org/abs/2501.07765

[14] Chen, X., Luo, Y., & Chen, J. (2025). A PINN-enriched finite element method for linear elliptic problems. https://arxiv.org/abs/2503.14913

[15] Kumar, A., Liu, Y., & García, F. (2024). PINNacle: A Comprehensive Benchmark of Physics-Informed Neural Networks. *NeurIPS 2024 Datasets and Benchmarks Track*. https://arxiv.org/pdf/2306.08827

[16] Hansaem Oh and Gwanghyun Jo (2025). Physics-informed neural network for the heat equation under imperfect contact conditions and its error analysis. *AIMS Mathematics* https://www.aimspress.com/article/doi/10.3934/math.2025364

[17] Blind Multiband Signal Reconstruction: Compressed Sensing for Analog Signals. (2009, 1 marzo). IEEE Journals & Magazine — IEEE Xplore. https://ieeexplore.ieee.org/document/4749297

[18] Repositorio GitHub con el código del TFG https://github.com/miguel-ara/TFG

[19] PyTorch documentation — PyTorch 2.6 documentation. (s.f.). https://docs.pytorch.org/docs/2.6/

[20] SciPy documentation — SciPy v1.14.1 Manual. (s.f.). https://docs.scipy.org/doc/scipy-1.14.1/index.html

[21] Plotly package — 6.0.0 documentation. (s.f.). https://plotly.com/python-api-reference/generated/plotly.html

[22] NumPy documentation — NumPy v2.0 Manual. (s.f.). https://numpy.org/doc/2.0/

[23] Matplotlib documentation — Matplotlib 3.9.2 documentation. (s.f.). https://matplotlib.org/3.9.2/