Departamento de Matemáticas, Facultad de Ciencias
Universidad Autónoma de Madrid

Redes Neuronales: aproximación de EDPs

Trabajo de fin de grado

Grado en Matemáticas

Autor: Luis Hebrero Garicano
Tutor: Julia Novo

Curso 2024-2025

Resumen

Las redes neuronales son un modelo matemático inspirado en el funcionamiento cerebral que, en esencia, se utiliza para encontrar funciones: funciones que clasifican datos, que predicen valores o incluso que anticipan la siguiente palabra en una frase. En este trabajo, se estudiará cómo se puede aplicar esta capacidad de aproximación de funciones para resolver ecuaciones en derivadas parciales. Exploraremos distintas estrategias para construir estas redes y analizaremos su aplicación en problemas concretos, centrándonos en las ventajas que aportan con respecto a los métodos numéricos tradicionales, así como en los casos en los que una aproximación mediante redes neuronales no resulta efectiva.

Abstract

Neural networks are a mathematical model inspired by the brain's functioning, primarily used to find functions: functions that classify data, predict values, or even anticipate the next word in a sentence. This work will explore how this function approximation capability can be applied to solve partial differential equations. We will investigate different strategies for constructing these networks and analyze their application to specific problems, focusing on the advantages they offer over traditional numerical methods, as well as the cases where a neural network-based approach proves ineffective.

Índice general

CAPÍTULO 1
Introducción y preliminares

Para poder entender las aproximaciones a las EDPs mediante redes neuronales, es necesario tener un conocimiento previo de las redes neuronales y de las ecuaciones en derivadas parciales. En este capítulo, se introducirán los conceptos básicos de ambos temas, así como las herramientas matemáticas necesarias para comprender el resto del trabajo.

1.1. Introducción a las redes neuronales

Una red neuronal, de forma abstracta, es simplemente una función que toma una entrada y produce una salida. Es decir, una red neuronal, es una función F que toma un vector de entrada x y produce un vector de salida y, siendo $F : \mathbb{R}^n \to \mathbb{R}^m$. La red neuronal se compone de una serie de capas, cada una de las cuales está formada por un conjunto de neuronas. Cada neurona de una capa recibe una serie de entradas, las procesa y produce una salida. La salida de cada neurona se calcula mediante una función de activación, que puede ser de distintos tipos, como la función sigmoide, la función tangente hiperbólica o la función ReLU. Para entender este concepto nos vamos a centrar en el caso concreto de la red neuronal de la Figura 1.1.

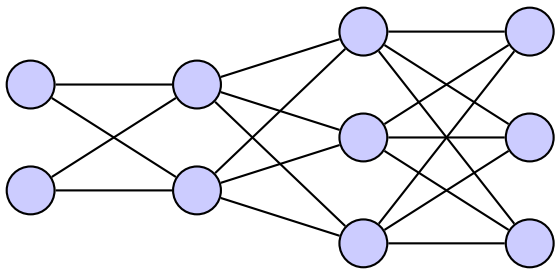


Figura 1.1: Esquema de una red neuronal con 4 capas.

Como se ve en la Figura 1.1, la entrada en nuestra función está representada por los dos círculos de la izquierda, que representan los valores de entrada x_1 y x_2, siendo así la entrada $x \in \mathbb{R}^2$. Estos valores se multiplican por unos pesos ($W^{[2]}$)y se suman a un sesgo b. La salida de esta neurona se calcula mediante una función de activación.

Así, los valores que "llegan" a la segunda capa de nuestra red neuronal serán de la forma

$$\sigma(W^{[2]}x + b^{[2]}) \in \mathbb{R}^2,$$

Siendo $W^{[2]} \in \mathbb{R}^{2\times 2}$ y el vector $b^{[2]} \in \mathbb{R}^2$. A partir de aquí, se repite el proceso para cada capa de la red neuronal, hasta llegar a la capa de salida, que nos dará el valor de salida de nuestra red neuronal. De forma visual, se pueden interpretar las flechas de la Figura 1.1 como los pesos por los que se va multiplicando.

En la tercera capa de la red neuronal, vemos que los valores que llegan de la capa 2, estos $\sigma(W^{[2]}x + b^{[2]})$, pertenecen a $\mathbb{R}^2$. De este modo, como tenemos 3 neuronas en la tercera capa, para obtener un valor perteneciente a $\mathbb{R}^3$, necesitamos una matriz $W^{[3]} \in \mathbb{R}^{3\times 2}$ y un vector $b^{[3]} \in \mathbb{R}^3$. Así, el valor de nuestra red neuronal en la tercera capa será

$$\sigma(W^{[3]}\sigma(W^{[2]}x + b^{[2]}) + b^{[3]}) \in \mathbb{R}^3.$$

Finalmente, la capa de salida recibirá de la tercera capa un vector perteneciente a $\mathbb{R}^3$, por lo que necesitaremos una matriz $W^{[4]} \in \mathbb{R}^{3\times 3}$ y un vector $b^{[4]} \in \mathbb{R}^3$. Así, el valor de salida de nuestra red neuronal, esa F de la que habíamos hablado al principio, será

$$F(x) = \sigma(W^{[4]}\sigma(W^{[3]}\sigma(W^{[2]}x + b^{[2]}) + b^{[3]}) + b^{[4]}) \in \mathbb{R}^3. \tag{1.1}$$

En general, una red neuronal se puede representar como una composición de funciones, donde cada función es una capa de la red neuronal.

Nuestra intención con este tipo de funciones es ir variando los valores de las matrices W, también conocidos como pesos, y los vectores b también conocidos como sesgos, para que la salida de nuestra red neuronal se acerque lo máximo posible a la salida deseada.

Para entender esto, vamos a utilizar la red neuronal de la Figura 1.1 para resolver un problema concreto muy sencillo de clasificación. Supongamos que tenemos una serie de puntos en el plano, de tres tipos distintos, como los de la Figura 1.2, y queremos clasificarlos en tres grupos, los puntos de tipo azul, rojo y amarillo.

De este modo, nuestra red neuronal recibirá como entrada un punto del plano, y nos dirá a qué categoría pertenece, devolviendo $(1,0,0)^T$ si es de la categoría azul, $(0,1,0)^T$ si es de la categoría roja y $(0,0,1)^T$ si es de la categoría amarilla.

Lo siguiente que queremos hacer será entrenar la red neuronal, es decir, ajustar los pesos y sesgos de la red neuronal para que la salida se acerque lo máximo posible a la salida deseada. Es decir, que cuando se introduzca un punto de un tipo concreto, la salida lo asigne a la categoría adecuada.

Designamos a $y(x)$ como la salida deseada de nuestra red neuronal, y a $F(x)$ como la salida real. Así, el error vendrá dado en función de los pesos y sesgos de la siguiente forma

$$E(W^{[2]}, W^{[3]}, W^{[4]}, b^{[2]}, b^{[3]}, b^{[4]}) = \frac{1}{2}\sum_{x\in X} \|y(x) - F(x)\|^2,$$

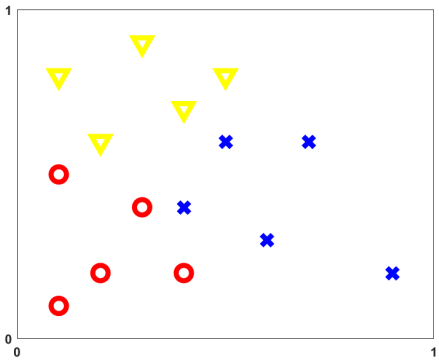


Figura 1.2: Puntos en el plano que marcan las tres categorías

donde X es el conjunto de puntos que tenemos para entrenar la red neuronal, en nuestro caso, serán 15. Esta función es conocida como función de coste.

Así, lo que queremos hacer es minimizar esta función, es decir, encontrar los pesos que minimicen la función E. Para ello, se utilizan algoritmos de optimización, como el descenso del gradiente. Este proceso es conocido como el entrenamiento de la red neuronal. Si se logra con éxito, la red neuronal será capaz de clasificar correctamente los puntos en el plano. En este caso concreto, al entrenar la red neuronal, se obtiene la clasificación de la Figura 1.3, en la que simplemente hemos aplicado nuestra función para cada punto del plano y lo hemos sombreado de acuerdo a la clasificación que se le ha dado.

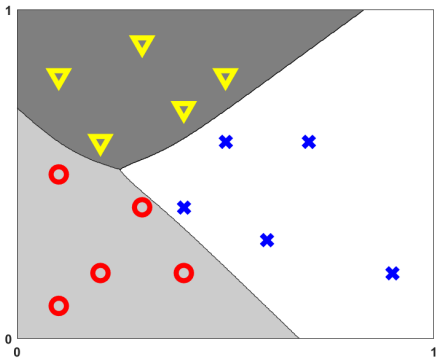


Figura 1.3: Puntos en el plano que marcan las tres categorías

Más adelante entenderemos con más detalle como es este proceso de entrenamiento.

1.1.1. Comentarios sobre la función sigmoide

Como hemos visto con el ejemplo anterior, las redes neuronales se pueden utilizar para generar funciones. A su vez, estas funciones se utilizan para resolver problemas

de forma aproximada. En el ejemplo de la Figura 1.1, la función red neuronal clasifica cualquier parte del plano en una de las tres categorías sombreadas a partir del entrenamiento. Dicho entrenamiento fija los valores de los pesos y sesgos, y por tanto define la función red neuronal (en el ejemplo define la función (1.1)).

Como las redes neuronales se utilizan para aproximar problemas de tipos muy distintos, una de las características deseables es que sean capaces de aproximar el conjunto de funciones "mayor posible".

Para poder lograr este objetivo, se necesitan las funciones de activación. Como ejercicio, supongamos que en nuestra ecuación (1.1) en lugar de utilizar la función sigmoide, no utilizamos ninguna función de activación, es decir, que nuestra red neuronal es simplemente una composición de funciones lineales. En este caso, nuestra red tendría la siguiente forma

$$F(x) = W^{[4]}(W^{[3]}(W^{[2]}x + b^{[2]}) + b^{[3]}) + b^{[4]} \in \mathbb{R}^3.$$

Claramente, una función lineal definida de forma global puede estar muy lejos de aproximar bien funciones generales. De este modo, parece por tanto razonable utilizar funciones de activación no lineales. La función sigmoide es una de las más utilizadas, pero existen otras como la función tangente hiperbólica o la función ReLU. En este trabajo, nos centraremos en la función sigmoide, que se define como

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

Esta función tiene la ventaja de que su derivada es fácil de calcular, y es precisamente esta derivada la que se utiliza en el algoritmo de entrenamiento de la red neuronal.

Un resultado relevante para justificar el uso de funciones de activación no lineales es el siguiente propuesto por Pinkus [8, Theorem 3.1]. Este resultado es para redes neuronales de una sola capa.

Teorema 1.1 (Pinkus)**.** *Sea $\sigma \in C(\mathbb{R})$ y sea conjunto $\mathcal{M}(\sigma) = span\{\sigma(w \cdot x + b) : b \in \mathbb{R}, w \in \mathbb{R}^n\}$, se cumple que*

Para cualquier $f \in C(\mathbb{R}^n)$, cualquier conjunto compacto $K \in \mathbb{R}^n$ y cualquier $\epsilon > 0$, existe una función $g \in \mathcal{M}(\sigma)$ tal que $\max_{x \in K} |f(x) - g(x)| < \epsilon$ si y solo si σ no es una función polinómica.

De forma más intuitiva, cualquier función $f \in C(\mathbb{R}^n)$, se puede aproximar tan bien como se quiera con una red neuronal de una sola capa si y solo si la función de activación no es polinómica.

Este resultado nos aporta una intuición de por qué las funciones de activación no lineales son necesarias para aproximar funciones de forma general pues, de no ser funciones no lineales, habría muchas funciones que no podríamos aproximar.

1.2. Conceptos preliminares sobre las ecuaciones en derivadas parciales

En este trabajo nos vamos a centrar en las ecuaciones en derivadas parciales elípticas, definidas con la siguiente forma general.

Definición 1.2. Dado un conjunto Ω, acotado y abierto en $\mathbb{R}^n$, decimos que una ecuación en derivadas parciales es elíptica si es de la siguiente forma:

$$-\sum_{i,j=1}^{n} \frac{\partial}{\partial x_j}\left(a_{ij}(x)\frac{\partial u}{\partial x_i}\right) + \sum_{i=1}^{n} b_i(x)\frac{\partial u}{\partial x_i} + c(x)u = f(x), \qquad x \in \Omega. \tag{1.2}$$

Donde los coeficientes $a_{ij}(x)$, $b_i(x)$, $c(x)$ y f son funciones que satisfacen las siguientes condiciones

$$a_{ij} \in C^1(\overline{\Omega}), \qquad i,j = 1,\dots,n \tag{1.3}$$

$$b_i, c \in C(\overline{\Omega}), \qquad i = 1,\dots,n \tag{1.4}$$

$$c \in C(\overline{\Omega}), \tag{1.5}$$

$$f \in C(\overline{\Omega}) \tag{1.6}$$

y además cumplen la condición de elipticidad uniforme, es decir

$$\sum_{i,j=1}^{n} a_{ij}(x)\xi_i\xi_j \geq c_0 \sum_{i=1}^{n} \xi_i^2, \quad \forall \xi = (\xi_1,\dots,\xi_n) \in \mathbb{R}^n, \quad x \in \overline{\Omega}; \tag{1.7}$$

Concretamente, a lo largo del trabajo, nos centraremos en problemas de condición de frontera de Dirichlet, es decir, problemas en los que se cumple que

$$u(x) = g(x), \qquad \forall x \in \partial\Omega. \tag{1.8}$$

En la formulación que hemos dado de las ecuaciones en derivadas parciales elípticas, nos estábamos refiriendo a su formulación en forma fuerte. No obstante, en muchos casos, no es posible encontrar una solución en forma fuerte, es decir, una función que cumpla la ecuación en todo el dominio y que además cumpla las condiciones de frontera. En estos casos, se recurre a la formulación débil de la ecuación, que permite encontrar una solución en un espacio de funciones más amplio.

Definición 1.3. Sea Ω un conjunto abierto de $\mathbb{R}^n$. Definimos el espacio de Sobolev $H^1(\Omega)$ como el conjunto de funciones en el siguiente conjunto

$$H^1(\Omega) = \{u \in L^2(\Omega) : \frac{\partial u}{\partial x_i} \in L^2(\Omega),\ i = 1,\dots,n\}. \tag{1.9}$$

En este espacio, se define la siguiente norma

$$\|u\|_{H^1(\Omega)} = \left(\|u\|^2_{L^2(\Omega)} + \sum_{i=1}^{n} \left\| \frac{\partial u}{\partial x_i} \right\|^2_{L^2(\Omega)} \right)^{1/2}. \tag{1.10}$$

Definimos demás el espacio $H^1_0(\Omega)$. Este espacio es el cierre de las funciones de $C^\infty_0(\Omega)$ en la norma de $H^1(\Omega)$. Es decir, $H^1_0(\Omega)$ es el conjunto de funciones $u \in H^1(\Omega)$ que se obtienen como límite en $H^1(\Omega)$ de una serie de funciones $\{u_m\}_{m=1}^\infty$ todas ellas en $C^\infty_0(\Omega)$. De este modo, si $\partial\Omega$ es suficientemente regular, $H^1_0(\Omega)$ es el siguiente conjunto:

$$H^1_0(\Omega) = \{u \in H^1(\Omega) : u = 0 \text{ en } \partial\Omega\}. \tag{1.11}$$

Con esto, podemos definir la solución débil de una ecuación en derivadas parciales elíptica de la siguiente forma.

Definición 1.4. Dadas las funciones $a_{ij} \in L^\infty(\Omega)$, $i, j = 1, \ldots, n$, $b_i \in L^\infty(\Omega), i = 1, \ldots, n$, $c \in L^\infty(\Omega)$, y la función $f \in L^2(\Omega)$. Decimos que la función $u \in H^1_0(\Omega)$ es una solución débil de la ecuación (1.2) si se cumple que,

$$\begin{aligned} &\sum_{i,j=1}^{n} \int_\Omega a_{ij}(x) \frac{\partial u}{\partial x_i} \frac{\partial \varphi}{\partial x_j}\, dx + \sum_{i=1}^{n} \int_\Omega b_i(x) \frac{\partial u}{\partial x_i} \varphi\, dx \\ &+ \int_\Omega c(x) u \varphi\, dx = \int_\Omega f(x)\varphi(x) dx, \qquad \forall \varphi \in H^1_0(\Omega). \end{aligned} \tag{1.12}$$

Para simplificar esta engorrosa notación, nos interesa definir la siguiente forma bilineal y lineal.

$$a(u, \varphi) = \sum_{i,j=1}^{n} \int_\Omega a_{ij}(x) \frac{\partial u}{\partial x_i} \frac{\partial \varphi}{\partial x_j}\, dx + \sum_{i=1}^{n} \int_\Omega b_i(x) \frac{\partial u}{\partial x_i} \varphi\, dx + \int_\Omega c(x) u \varphi\, dx, \tag{1.13}$$

y el funcional lineal,

$$l(\varphi) = \int_\Omega f(x)\varphi(x) dx. \tag{1.14}$$

De este modo, nuestro problema elíptico en forma debil se puede expresar de la siguiente manera.

$$a(u, \varphi) = l(\varphi) \quad \forall \varphi \in H^1_0(\Omega). \tag{1.15}$$

1.3. Métodos numéricos tradicionales: el método de los elementos finitos

El método de los elementos finitos (MEF) es una técnica numérica para resolver ecuaciones en derivadas parciales. Este método es ampliamente utilizado en ingeniería y ciencias aplicadas porque permite aproximar soluciones en dominios complejos.

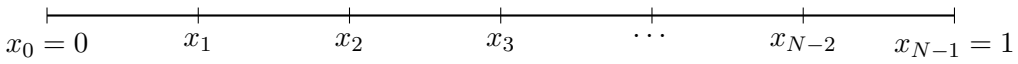


Figura 1.4: División de $\Omega = (0, 1)$ en N elementos finitos

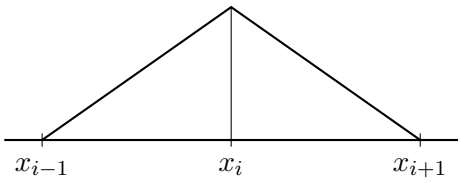


Figura 1.5: Polinomio ϕ_i

Para un problema elíptico con condiciones de frontera, se parte de la ecuación en su formulación débil. Como hemos visto en el apartado anterior, este problema se puede formular de forma muy simplificada como:

$$a(u, \varphi) = l(\varphi) \quad \forall \varphi \in V,$$

donde V es el espacio de funciones en el que están nuestras soluciones. En el caso del problema con condiciones homogéneas, este espacio será $V = H_0^1(\Omega)$.

Lo siguiente que se hace es dividir el dominio Ω en un conjunto de subdominios más pequeños, llamados elementos finitos. Por ejemplo, si estamos trabajando con un problema de una dimensión, con $\Omega = (0, 1)$, dividimos $\overline{\Omega} = [0, 1]$ en N subintervalos $[x_i, x_{i+1}]$, $i = 1, \ldots, N - 1$, como se hace en la Figura 1.3.

A esta subdivision, se le asocia una base de polinomios a trozos. Así, se reemplaza el subespacio de funciones que habíamos llamado V por un subespacio de dimensión finita V_h formado por polinomios a trozos de grado fijo. Siguiendo nuestro ejemplo de antes, la base de polinomios que vamos a utilizar será la de los polinomios como los vistos en la Figura 1.3. Definidos de la siguiente forma. condiciones de contorno

$$\phi_i(x) = \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}}, & \text{si } x \in [x_{i-1}, x_i], \\ \frac{x_{i+1} - x}{x_{i+1} - x_i}, & \text{si } x \in [x_i, x_{i+1}], \\ 0, & \text{en otro caso.} \end{cases}$$

En este ejemplo, nuestro espacio de funciones será $V_h = \text{span}\{\phi_1, \ldots, \phi_{N-1}\}$, donde N es el número de elementos finitos en los que hemos dividido el dominio. De este modo, al buscar soluciones en este nuevo espacio de funciones, nuestra ecuación diferencial elíptica se convertiría en encontrar $u_h \in V_h$ tal que falta condiciones de contorno

$$a(u_h, v_h) = l(v_h) \quad \forall v_h \in V_h,$$

De forma intuitiva, lo que se hace es que en vez de buscar una solución en un espacio de funciones de dimensión infinita, como es V, se busca una solución en un espacio de funciones de dimensión finita, como es V_h. Para poder hacer eso, que nos simplifica mucho el problema, lo que hemos hecho es que nuestro dominio, lo hemos

dividido en subdominios más pequeños a partir de los cuales hemos construido nuestro nuevo espacio de funciones. falta LEX-MILGRAM

Ahora, encontrar una solución se convierte en encontrar los coeficientes en la base de polinomios a trozos, es decir, los U_i en la siguiente ecuación

$$u_h = \sum_{i=1}^{N-1} U_i \phi_i. \tag{1.16}$$

Es decir, resolver la ecuación diferencial se convierte en encontrar $U = (U_1, \ldots, U_N) \in \mathbb{R}^{N-1}$ que cumplen:

$$\sum_{i=1}^{N-1} a\left(\phi_i, \phi_j\right) U_i = l(\phi_j) \quad \forall j = 1, \ldots, N-1. \tag{1.17}$$

Esto se traduce en un sistema lineal de la forma $AU = b$, donde A es la matriz de rigidez (con entradas $A_{ij} = a\left(\phi_i, \phi_j\right)$), y b es el vector de términos fuente (con entradas $b_j = l(\phi_j)$). A este nuevo sistema lineal se le pueden aplicar métodos numéricos tradicionales para resolverlo, como la factorización LU, encontrando así los coeficientes U que nos dan u_h, la aproximación de u. generalizar el problema en 2D

Poner esto como un ejemplo de la generalización.

Hay más información en el libro de como se plantea este problema

Ponerlo como ejemplo de lex milgram para más dimensiones.

Ejemplo 1.5. Veamos ahora un ejemplo concreto de la aplicación del FEM a través de la ecuación de Poisson. El objetivo es encontrar un campo escalar u sobre un dominio $\Omega \subset \mathbb{R}^n$ que satisfaga:

$$\begin{aligned} -\nabla^2 u &= f \quad \text{en } \Omega, \\ u &= 0 \quad \text{en } \Gamma_D, \\ \nabla u \cdot n &= g \quad \text{en } \Gamma_N, \end{aligned} \tag{1.18}$$

donde f es un término fuente, g es un flujo en la frontera, n es el vector normal unitario, Γ_D y Γ_N son partes de la frontera donde se aplican las condiciones de contorno de Dirichlet y Neumann, respectivamente.

El método de elementos finitos (FEM) aproxima u transformando el problema en su forma débil y resolviéndolo en un espacio de funciones de dimensión finita. La forma fuerte del problema se convierte en: decir en que espacio esta u

$$a(u, v) = l(v) \quad \forall\, v \in V, \tag{1.19}$$

donde:

- $a(u, v) = \int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}x$

- $l(v) = \int_\Omega fv\,\mathrm{d}x + \int_{\Gamma_N} gv\,\mathrm{d}s$

Al discretizar el espacio V, el FEM transforma esta ecuación en un sistema de ecuaciones lineales que se puede resolver numéricamente. En este caso concreto usaremos un dominio rectangular $\Omega = [0,2] \times [0,1]$, en el que se imponen las siguientes condiciones:

- Frontera de Dirichlet Γ_D en los bordes verticales ($x = 0$ y $x = 2$) con $u = 0$,
- Frontera de Neumann Γ_N en los bordes horizontales ($y = 0$ y $y = 1$) con flujo $g(x) = \sin(5x)$.
- Término fuente $f(x,y) = 10 \cdot \exp\left(-\frac{(x-0{,}5)^2+(y-0{,}5)^2}{0{,}02}\right)$.

Bajo estas condiciones, si usamos el paquete de python FEniCS, podemos resolver el problema obteniendo el siguiente resultado.

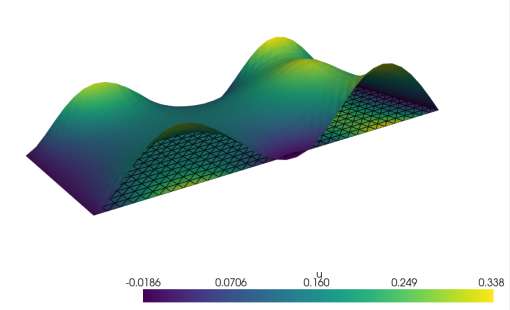


Figura 1.6: Solución de la ecuación de Poisson

Se puede apreciar en la Figura 1.6 la malla de elementos finitos utilizada para la aproximación de la solución, siendo esta la que da lugar a la base de polinomios a trozos que hemos mencionado anteriormente.

1.4. El problema elíptico autoadjunto

Vamos a considerar ahora un caso particular que es el llamado problema elíptico autoadjunto, el cual permite caracterizar las soluciones de las EDP elípticas como el mínimo de un funcional. Esto conecta de manera lógica con el enfoque de redes neuronales para aproximar soluciones, donde la función de coste de la red neuronal se puede definir como esta función. Al minimizarla, se obtiene una solución aproximada del problema original. No obstante, esto lo veremos más adelante en detalle. De momento, se define el problema elíptico autoadjunto de la siguiente forma.

Definición 1.6. Dado un conjunto Ω, acotado y abierto en $\mathbb{R}^n$, decimos que una ecuación en derivadas parciales elíptica autoadjunta, y además se cumplen las

siguientes condiciones de simetría en los coeficientes.

$$\begin{aligned} a_{ij} &= a_{ji}, \qquad i,j = 1,\dots,n, \\ b_i &= 0, \qquad i = 1,\dots,n, \end{aligned} \tag{1.20}$$

Bajo estas condiciones, nuestro problema se puede formular de la siguiente forma.

$$\begin{cases} -\sum_{i,j=1}^{n} \frac{\partial}{\partial x_j}\left(a_{ij}(x)\frac{\partial u}{\partial x_i}\right) + c(x)u = f(x), & x \in \Omega, \\ u(x) = 0, & x \in \partial\Omega. \end{cases} \tag{1.21}$$

En un problema autoadjunto se pueden caracterizar las soluciones débiles como el mínimo de un funcional. Esto es así por el siguiente resultado.

Lema 1.7. *hacerla más rigurosa. Tengo que hablar de que es la solución única*

Definir el J antes

Las siguientes afirmaciones son equivalentes:

1. *Encontrar* $u \in H_0^1(\Omega)$ *tal que* $a(u,\varphi) = l(\varphi) \quad \forall \varphi \in H_0^1(\Omega)$
2. *Encontrar* $u \in H_0^1(\Omega)$ *tal que* $J(u) \leq J(\varphi) \quad \forall \varphi \in H_0^1(\Omega)$, *siendo* $J(u)$ *la siguiente función*
$$J(u) = \frac{1}{2}a(u,u) - l(u), \quad u \in H_0^1(\Omega)$$

CAPÍTULO 2
Aproximación de EDPs mediante redes neuronales

2.1. PINN: Physics-Informed Neural Networks

2.1.1. Introducción

Las PINN, o Physics-Informed Neural Networks, son una técnica que combina la resolución de ecuaciones en derivadas parciales con redes neuronales. La idea es que, en vez de resolver la ecuación diferencial directamente o con métodos numéricos tradicionales, se entrena una red neuronal para que aproxime la solución de la ecuación.

Como habíamos comentado en la Sección 1.1, las redes neuronales son capaces de aproximar cualquier función, por lo que, en teoría, pueden aproximar cualquier solución de una ecuación diferencial. De este modo, las PINN van a aprovechar esto, definiendo la aproximación de nuestra solución u, como la salida de la función red neuronal $\hat{u}(x;\theta)$, donde θ hace referencia a los pesos y sesgos vistos en la Sección 1.1. Así, la función en la Figura 2.1.1 sería la aproximación de la solución de la ecuación diferencial.

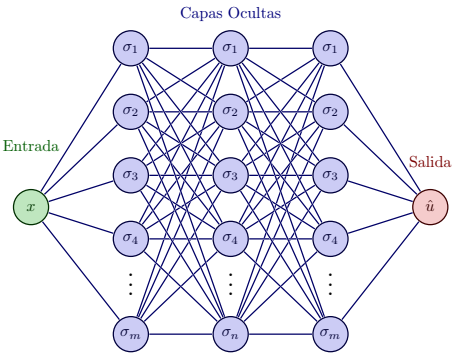


Figura 2.1: Esquema de una PINN

2.1.2. Formulación de las PINN

Consideramos una EDP de la forma vista en la Sección 1.2, es decir, una EDP elíptica sujeta a condiciones de contorno de Dirichlet. De modo que la ecuación a resolver es la siguiente:

$$\begin{cases} -\sum_{i,j=1}^{n} \frac{\partial}{\partial x_j}\left(a_{ij}(\mathbf{x})\frac{\partial u}{\partial x_i}\right) + \sum_{i=1}^{n} b_i(\mathbf{x})\frac{\partial u}{\partial x_i} + c(\mathbf{x})u = f(\mathbf{x}), & \mathbf{x} \in \Omega, \\ u(\mathbf{x}) = g(\mathbf{x}), & \mathbf{x} \in \partial\Omega. \end{cases} \tag{2.1}$$

donde $\Omega \subset \mathbb{R}^n$ es un dominio y las funciones $a_{ij}(\mathbf{x})$, $b_i(\mathbf{x})$, $c(\mathbf{x})$, $f(\mathbf{x})$ y $g(\mathbf{x})$ son conocidas. La incógnita es la función $u : \Omega \to \mathbb{R}$ que satisface la ecuación diferencial y las condiciones de contorno. Para poder construir y ajustar los pesos de una red neuronal que aproxime u es esencial tener una función de coste, es decir, una función que nos indique la calidad de nuestra aproximación. En el caso de las PINN, la función de coste se define como la suma ponderada de dos términos. El primero de ellos garantiza que la función aproximada por la red neuronal satisface la ecuación diferencial en todo el dominio. El segundo término asegura que la aproximación satisface las condiciones de contorno de Dirichlet

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{T}) = w_f \mathcal{L}_f(\boldsymbol{\theta}; \mathcal{T}_f) + w_b \mathcal{L}_b(\boldsymbol{\theta}; \mathcal{T}_b), \tag{2.2}$$

donde w_f y w_b son pesos que se ajustan para dar más importancia a un término u otro y ((No encuentro casi nada de como se obtienen estos pesos))

$$\mathcal{L}_f(\boldsymbol{\theta}; \mathcal{T}_f) = \frac{1}{|\mathcal{T}_f|} \sum_{\mathbf{x} \in \mathcal{T}_f} \left\| -\sum_{i,j=1}^{n} \frac{\partial}{\partial x_j}\left(a_{ij}(\mathbf{x})\frac{\partial \hat{u}}{\partial x_i}\right) + \sum_{i=1}^{n} b_i(\mathbf{x})\frac{\partial \hat{u}}{\partial x_i} + c(\mathbf{x})\hat{u} - f(\mathbf{x}) \right\|_2^2,$$

$$\mathcal{L}_b(\boldsymbol{\theta}; \mathcal{T}_b) = \frac{1}{|\mathcal{T}_b|} \sum_{\mathbf{x} \in \mathcal{T}_b} \|\hat{u}(\mathbf{x}) - g(\mathbf{x})\|_2^2,$$

En estas ecuaciones, $\mathcal{T}_f$ y $\mathcal{T}_b$ son conjuntos de puntos en los que se evaluarán las funciones de coste, siendo $\mathcal{T} = \mathcal{T}_f \cup \mathcal{T}_b$, conocidos como puntos de colocación o "collocation points". En el caso de $\mathcal{L}_f$, se evaluará en puntos del dominio pues $\mathcal{T}_f \subset \Omega$, mientras que en $\mathcal{L}_b$ se evaluará en puntos de la frontera, pues $\mathcal{T}_f \subset \partial\Omega$. Obviamente, estos conjuntos son necesarios pues a nivel computacional es imposible calcular derivadas e integrales en un espacio continuo. Existe mucha literatura sobre como tomar estos puntos pues, al igual que en FEM la malla impacta el resultado, en las PINN, el conjunto $\mathcal{T}$ determina como de bien nuestra red neuronal se ajusta a la solución [7][1][6][9][3].

Dada la función de coste definida y la distribución de los puntos de colocación, el entrenamiento de la red neuronal se reduce a resolver un problema de optimización: encontrar los parámetros θ que minimizan la función de coste $\mathcal{L}(\boldsymbol{\theta}; \mathcal{T})$. Para ello, es habitual emplear métodos de optimización basados en gradientes, como Adam o L-BFGS.

Este párrafo lo dejaría si explico antes que es el AD, sino queda muy al aire Una ventaja fundamental del uso de redes neuronales en este contexto es la diferenciación

automática, una técnica numérica muy potente que facilita la obtención de las derivadas de $\hat{u}$ con respecto a sus entradas de forma exacta y eficiente. A continuación, vamos a ver dos ejemplos para ver como se aplican las PINN a problemas concretos.

Ejemplo 2.1. Consideramos una ecuación de Poisson:

$$-\Delta u = \pi^2 \sin(\pi x), \quad x \in [-1, 1],$$

con las condiciones de contorno de Dirichlet

$$u(-1) = 0, \quad u(1) = 0.$$

En esta ecuación la solución exacta es conocida $u(x) = \sin(\pi x)$. Para resolver este problema con una PINN, nos asistimos de la librería de python DeepXDE [4], la cual tiene implementada la funcionalidad necesaria para entrenar una red neuronal que aproxime EDPs. Así, comenzamos definiendo el intervalo en el que se encuentra el dominio y los puntos de colocación

```
geom = dde.geometry.Interval(-1, 1)
```

definimos la EDP

```
def pde(x, y):
    dy_xx = dde.grad.hessian(y, x)
    return -dy_xx - np.pi ** 2 * tf.sin(np.pi * x)
```

indicamos las condiciones de contorno de Dirichlet

```
bc = dde.icbc.DirichletBC(geom, func, boundary)
```

y, con todo, se define el problema de EDPs

```
data = dde.data.PDE(geom, pde, bc, 16, 2, solution=func,
num_test=100)
```

Gracias a la librería DeepXDE, podemos entrenar la red neuronal con el siguiente código

```
Losshistory, train_state = model.train(
    iterations=10000, callbacks=[checkpointer, movie]
)
```

Por detrás, esta librería se encargará de definir la función de coste asociada, que en este caso será

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{T}) = w_f \mathcal{L}_f(\boldsymbol{\theta}; \mathcal{T}_f) + w_b \mathcal{L}_b(\boldsymbol{\theta}; \mathcal{T}_b),$$

donde

$$\mathcal{L}_f(\boldsymbol{\theta}; \mathcal{T}_f) = \frac{1}{|\mathcal{T}_f|} \sum_{\mathbf{x} \in \mathcal{T}_f} \left\| \Delta \hat{u} + \pi^2 \sin(\pi x) \right\|_2^2,$$

$$\mathcal{L}_b(\boldsymbol{\theta}; \mathcal{T}_b) = \frac{1}{|\mathcal{T}_b|} \sum_{\mathbf{x} \in \mathcal{T}_b} \left\| \hat{u}(\mathbf{x})) \right\|_2^2,$$

por los argumentos con los que se han inicializado las funciones, $\mathcal{T}_f$ serán 16 puntos aleatorios en el intervalo $[-1, 1]$ y $\mathcal{T}_b = \{0, 1\}$. Con todo, el resultado de ejecutar este código será el de la Figura 2.2.

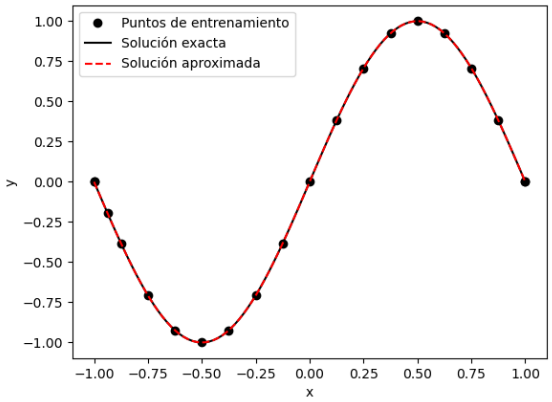


Figura 2.2: Solución de la ecuación de Poisson

DUDAS PARA JULIA No sé si poner el código en python o no, si explicar en más detalle cada instrucción o si entrar en como se entrena la red neuronal. Tengo miedo de que todo esto poco matemático. Como el objetivo del trabajo son las ecuaciones autoadjuntas igual esto solo es dar unas pinceladas del estado del arte. También en todo esto parece interesante hablar de como se eligen los hiperparámetros.

No obstante, las PINN pueden presentar errores elevados incluso en sistemas relativamente simples. A continuación, vemos uno de los ejemplos que presentan en Luo et al. en [5] en los que las PINN fallan.

Ejemplo 2.2. La ecuación considerada es unidimensional y se expresa como

$$\begin{cases} Lu = -D_x(AD_x u) = f & \text{en } \Omega = (-1, 1), \\ u = 0 & \text{en } \partial\Omega = \{-1, 1\}, \end{cases} \tag{2.3}$$

donde la función coeficiente A y f son ambas funciones por tramos discontinuas y se expresan como

$$A(x) = \begin{cases} \frac{1}{2}, & x \in (-1, 0), \\ 1, & x \in [0, 1), \end{cases} \qquad f(x) = \begin{cases} 0, & x \in (-1, 0), \\ -2, & x \in [0, 1). \end{cases} \tag{2.4}$$

Claramente, no existe una solución fuerte, porque u tiene que ser derivable, y en tal caso, la parte izquierda de la ecuación no podría ser igual a una función

discontinua. Sin embargo la solución débil $u \in H^1((-1,1))$ para esta ecuación es

$$u(x) = \begin{cases} -\frac{2}{3}x - \frac{2}{3}, & x \in (-1,0), \\ x^2 - \frac{1}{3}x - \frac{2}{3}, & x \in [0,1). \end{cases} \tag{2.5}$$

En la serie de experimentos numéricos, utilizamos una red neuronal con la siguiente configuración 1-256-256-256-1. Así, la función de coste vendrá dada por la siguiente función.

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{T}) = w_f \mathcal{L}_f(\boldsymbol{\theta}; \mathcal{T}_f) + w_b \mathcal{L}_b(\boldsymbol{\theta}; \mathcal{T}_b),$$

donde

$$\mathcal{L}_f(\boldsymbol{\theta}; \mathcal{T}_f) = \frac{1}{|\mathcal{T}_f|} \sum_{\mathbf{x} \in \mathcal{T}_f} \|D_x(A(x)D_x w(x)) + f(x)\|_2^2,$$
$$\mathcal{L}_b(\boldsymbol{\theta}; \mathcal{T}_b) = \frac{1}{|\mathcal{T}_b|} \sum_{\mathbf{x} \in \mathcal{T}_b} \|\hat{u}(\mathbf{x})\|_2^2,$$

En este caso, $\mathcal{T}_b = \{-1, 1\}$ y $\mathcal{T}_f$ es un conjunto de 1000 puntos muestreados uniformemente en el intervalo $(-1, 1)$. Es decir, $|\mathcal{T}_f| = 1000$ y $|\mathcal{T}_b| = 2$.

Bajo esta configuración, obtenemos la función de red $\hat{u}(\mathbf{x}; \theta)$ numéricamente a través del método PINN y la comparamos con la solución débil en la Figura 2.3.

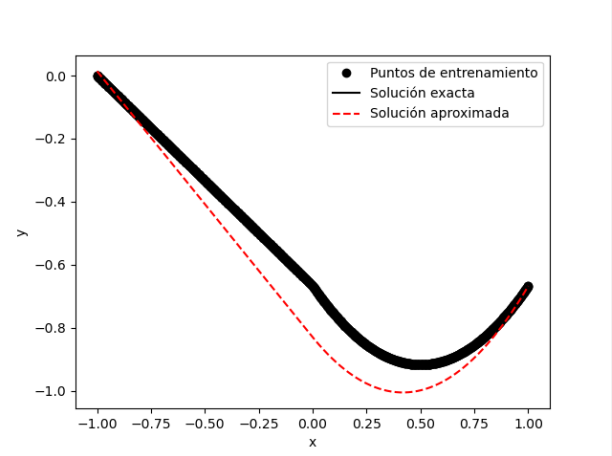


Figura 2.3: Solución de la ecuación

Obviamente, el método no logra encontrar la solución exacta u. La diferencia entre u y $\hat{u}(\mathbf{x}; \theta)$ es tan grande como la magnitud de u. Además, si nos fijamos en la gráfica de entrenamiento y test, vemos que el error no disminuye con el número de iteraciones.

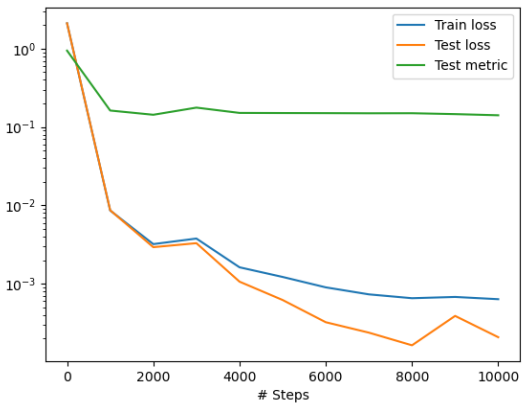


Figura 2.4: Error de la red neuronal

Como hemos visto, las PINN presentan errores significativos en ciertos problemas, lo que puede atribuirse a diversos factores. Uno de los principales desafíos de las PINN es la falta de garantía de unicidad en la solución. A diferencia de los métodos numéricos tradicionales, las PINN resuelven problemas de optimización no convexos, los cuales, por naturaleza, no aseguran una solución única. Esto implica que la red neuronal no tiene garantía de converger a la solución correcta y, de hecho, en muchos casos no lo hace. Es más, al igual que ocurre con otros algoritmos de inteligencia artificial, la red puede converger con la misma certeza (o error) hacia una solución incorrecta que hacia la solución real.

Otro obstáculo importante es la ausencia de una justificación teórica que determine cuáles son los hiperparámetros óptimos. Este problema añade incertidumbre al proceso de ajuste de la red, lo que complica la obtención de resultados precisos. Finalmente, otro aspecto problemático de las PINN es que aproximan el residuo de forma fuerte. Esto significa que, si no existe una solución en el sentido fuerte (como ocurre en algunos casos), la red neuronal no puede aproximar de forma adecuada la solución débil.

De estos problemas, los dos primeros son característicos del uso de redes neuronales, mientras que el tercero es específico del método. Con esto en mente, exploraremos un enfoque alternativo que considere la forma débil en la composición del residuo, lo que podría permitir superar esta limitación.

2.2. El “Deep Ritz Method”

El método “Deep Ritz” es una técnica numérica basada en las redes neuronales, que busca encontrar una solución a una ecuación en derivadas parciales utilizando su forma débil. Este método es más coherente con la formulación matemática del problema, y evita las restricciones demasiado fuertes que presentan las PINN.

CAPÍTULO 3

Resultados

CAPÍTULO 4
Partes eliminadas - NO IMPRIMIR

4.1. Espacios de funciones

Para poder entender y formular de forma matemática el problema de aproximación de EDPs mediante redes neuronales, es necesario tener un conocimiento previo de los espacios de funciones en los que trabajamos. Como veremos más adelante al estudiar los problemas de frontera para EDPs elípticas, es importante tener una caracterización del espacio H_1^0. Para entenderlo, empezaremos por los espacios de Hilbert Soobolev.

Definición 4.1. Seam Ω un conjunto abierto de $\mathbb{R}^n$. Definimos el espacio de Sobolev $H^1(\Omega)$ como el conjunto de funciones en el siguiente conjunto

$$H^1(\Omega) = \{u \in L^2(\Omega) : \frac{\partial u}{\partial x_i} \in L^2(\Omega),\, i = 1, \ldots, n\}. \tag{4.1}$$

En este espacio, se define la norma de funciones de con la siguiente ecuación

$$\|u\|_{H^1(\Omega)} = \left(\|u\|^2_{L^2(\Omega)} + \sum_{i=1}^{n} \left\| \frac{\partial u}{\partial x_i} \right\|^2_{L^2(\Omega)} \right)^{1/2}. \tag{4.2}$$

Dada esta definición, se define el espacio $H_0^1(\Omega)$ como el cierre de las funciones de $C_0^\infty(\Omega)$ en la norma de $H^1(\Omega)$. Es decir, $H_0^1(\Omega)$ es el conjunto de funciones $u \in H^1(\Omega)$ que se obtienen como límite en $H^1(\Omega)$ de una serie de funciones $\{u_m\}_{m=1}^\infty$ todas ellas en $C_0^\infty(\Omega)$. Así, de forma más simple, se puede demostrar que $H_0^1(\Omega)$ se trata del siguiente conjunto.

$$H_0^1(\Omega) = \{u \in H^1(\Omega) : u = 0 \text{ en } \partial\Omega\}. \tag{4.3}$$

Nótese que $H_0^1(\Omega)$ es un espacio de Hilbert con la misma norma y producto interno que $H^1(\Omega)$.

Ejemplo 4.2. Es fácil ver por ejemplo que la función $u = x^2 + y^2 - 4$ pertenece a $H_0^1(C)$, siendo $C = \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 < 4\}$ el círculo abierto de radio 2 centrada en el origen. Esto se debe a que $u \in L^2(C)$ y además:

$$\frac{\partial u}{\partial x} = 2x \in L^2(C), \quad \frac{\partial u}{\partial y} = 2y \in L^2(C), \quad u = 0 \text{ en } \partial C.$$

Por tanto, $u \in H_0^1(C)$.

4.2. Introducción a las ecuaciones en derivadas parciales

Las ecuaciones en derivadas parciales (EDPs) son ecuaciones que relacionan una función desconocida con sus derivadas parciales. Son fundamentales en la física y en la ingeniería, ya que permiten modelar fenómenos físicos y predecir su evolución en el tiempo.

Existen varios tipos de EDPs, no obstante, nosotros nos centraremos en los problemas de frontera para ecuaciones en derivadas parciales elípticas. Estas se utilizan para resolver problemas de equilibrio, que implican encontrar la solución de una ecuación diferencial en un dominio acotado con condiciones de frontera específicas. Estos problemas incluyen la distribución estacionaria de temperatura, el flujo de fluidos incompresibles no viscosos, la distribución de tensiones en sólidos en equilibrio, y el cálculo de campos eléctricos en regiones con densidad de carga. En general, se aplican cuando se busca determinar un potencial en situaciones estacionarias.

Lo he sacado de: este enlace, este tipo de cosas se citan?

Uno de los primero ejemplos de este tipo de ecuaciones es la ecuación de Poisson,

$$-\Delta u = f,$$

donde Δ es el operador laplaciano, que se define como la suma de las segundas derivadas parciales de la función u:

$$\Delta u = \sum_{i=1}^{n} \frac{\partial^2 u}{\partial x_i^2}$$

Esta ecuación es una EDP elíptica pues cumple la siguiente definición general.

Definición 4.3. Dado un conjunto Ω, acotado y abierto en $\mathbb{R}^n$, decimos que una ecuación en derivadas parciales es elíptica si:

$$-\sum_{i,j=1}^{n} \frac{\partial}{\partial x_j}\left(a_{ij}(x)\frac{\partial u}{\partial x_i}\right) + \sum_{i=1}^{n} b_i(x)\frac{\partial u}{\partial x_i} + c(x)u = f(x), \qquad x \in \Omega. \tag{4.4}$$

Donde los coeficientes $a_{ij}(x)$, $b_i(x)$, $c(x)$ y f son funciones que satisfacen las siguientes condiciones

$$a_{ij} \in C^1(\overline{\Omega}), \qquad i,j = 1,\dots,n \tag{4.5}$$

$$b_i, c \in C(\overline{\Omega}), \qquad i = 1,\dots,n \tag{4.6}$$

$$c \in C(\overline{\Omega}), \tag{4.7}$$

$$f \in C(\overline{\Omega}) \tag{4.8}$$

De entre los problemas elípticos definidos en 1.2, como comentábamos, nos interesan las ecuaciones en derivadas parciales elípticas con condiciones de frontera, en concreto, las condiciones de frontera de Dirichlet.

Definición 4.4. El problema de condición de frontera de Dirichlet es concretamente el que tenemos una EDP elíptica como la definida en 1.2, tal que, nuestra solución u, además de cumplir las ecuación (1.2), cumple la siguiente condición de frontera

$$u(x) = g(x), \qquad \forall x \in \partial\Omega. \tag{4.9}$$

Asimismo, el problema homogéneo de Dirichlet es aquel en el que $g = 0$. A lo largo del trabajo nos centraremos principalmente en es te tipo de EDPs.

Con todo, el tipo de ecuaciones elípticas en el que nos vamos a centrar (entendiendo que se cumplen las condiciones de (1.3) - (1.6)) serán de la siguiente forma.

$$\begin{cases} -\sum_{i,j=1}^{n} \frac{\partial}{\partial x_j}\left(a_{ij}(x)\frac{\partial u}{\partial x_i}\right) + \sum_{i=1}^{n} b_i(x)\frac{\partial u}{\partial x_i} + c(x)u = f(x), & x \in \Omega \\ u(x) = 0, & x \in \partial\Omega. \end{cases} \tag{4.10}$$

4.3. Forma fuerte y débil de una EDP

En la formulación que hemos dado de las ecuaciones en derivadas parciales elípticas, nos estábamos refiriendo a su formulación en forma fuerte. De forma intuitiva, esta es la que se obtiene directamente de la definición de la ecuación, y es la que relaciona la función desconocida con sus derivadas parciales. La solución de estas ecuaciones en forma fuerte es la que se conoce como solución clásica. De esto surge la siguiente definición.

Definición 4.5. Una función $u \in C^2(\Omega)$ que cumples las condiciones de frontera de Dirichlet y satisface la ecuación (1.2) en Ω se dice que es una solución clásica o fuerte de la ecuación (1.2).

No obstante, en muchos casos, no es posible encontrar una solución en forma fuerte, es decir, una función que cumpla la ecuación en todo el dominio y que además cumpla las condiciones de frontera. En estos casos, se recurre a métodos alternativos, como

la formulación débil de la ecuación, que permite encontrar una solución en un espacio de funciones más amplio. A continuación, vemos un ejemplo que motiva la necesidad de recurrir a la formulación débil de una ecuación.

Ejemplo 4.6. Supongamos que tenemos la siguiente ecuación de Poisson con una condición de frontera de Dirichlet homogénea,

$$\begin{cases} -\Delta u = f, & \text{en } \Omega, \\ u = 0, & \text{en } \partial\Omega. \end{cases}$$

Donde Ω es un conjunto acotado y abierto en $\mathbb{R}^n$. En este caso, no siempre es posible encontrar una solución en forma fuerte, es decir, una función u que cumpla la ecuación en todo el dominio Ω y que además cumpla la condición de frontera. Por ejemplo, si f no es una función suave, no se puede garantizar la existencia de una solución en forma fuerte pues estaríamos rompiendo la condición de (1.6). En estos casos, se recurre a métodos alternativos, como la formulación débil de la ecuación, que permite encontrar una solución en un espacio de funciones más amplio.

Para pasar de forma fuerte a débil, lo que hacemos es seguir el siguiente proceso

1. Multiplicamos a ambos lados de la igualdad por una función $\varphi \in C_0^\infty(\Omega)$ e integramos.

$$\int_\Omega \left(-\sum_{i,j=1}^n \frac{\partial}{\partial x_j}\left(a_{ij}(x)\frac{\partial u}{\partial x_i}\right) + \sum_{i=1}^n b_i(x)\frac{\partial u}{\partial x_i} + c(x)u \right) \varphi \, dx = \int_\Omega f(x)\varphi \, dx$$

2. Expandimos la multiplicación e integramos por partes en la primera integral, llegando a

$$\sum_{i,j=1}^n \int_\Omega a_{ij}(x)\frac{\partial \varphi}{\partial x_j}\frac{\partial u}{\partial x_i} + \sum_{i=1}^n \int_\Omega b_i(x)\frac{\partial u}{\partial x_i}\varphi + \int_\Omega c(x)\varphi u \, dx = \int_\Omega f(x)\varphi \, dx$$

   Con esta manipulación hemos conseguido una cosa muy interesante: ya no tenemos segundas derivadas de u, es decir, ya no necesitamos que $u \in C^2$. Para que esta igualdad tenga sentido, solo hace falta que $u \in L^2(\Omega)$ y que $\partial y/\partial x_i \in L^2(\Omega)$, $i = 1, \ldots, n$. Nótese además que, para que se cumpla la condición de frontera de Dirichlet, $u = 0 \in \partial\Omega$. Todo esto se traduce a que $u \in H_0^1(\Omega)$ Esto simplifica las condiciones del problema pues el espacio de funciones en el que puede estar u, ahora es mucho más amplio.

3. Para simplificar aun más el problema, nótese que a_{ij} ya no aparecen tras derivadas de ningun tipo, luego no es necesario asumir que $a_{ij} \in C^1(\overline{\Omega})$, basta con que $a_{ij} \in L^\infty(\Omega)$. Por lo mismo, b_i, $c \in L^\infty(\Omega), i = 1, \ldots, n$ es suficiente.

4. Por último, nótese que $C_0^\infty(\Omega) \subset H_0^1(\Omega)$, luego se puede ver que teniendo $u, v \in H_0^1(\Omega)$, nuestra ecuación sigue teniendo sentido. Con todo esto, surge la siguiente definición de forma débil.

Ejemplo 4.7. Resolveremos una ecuación de calor:

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}, \quad x \in [0,1], \quad t \in [0,1]$$

donde $\alpha = 0{,}4$ es la constante de difusividad térmica. Con condiciones de frontera de Dirichlet:

$$u(0,t) = u(1,t) = 0,$$

y condición inicial periódica (sinusoidal):

$$u(x,0) = \sin\left(\frac{n\pi x}{L}\right), \quad 0 < x < L, \quad n = 1, 2, \ldots$$

donde $L = 1$ es la longitud de la barra y $n = 1$ es la frecuencia de la condición inicial sinusoidal. Para este problema, sabemos que la solución exacta es:

$$u(x,t) = e^{-\frac{n^2\pi^2\alpha t}{L^2}} \sin\left(\frac{n\pi x}{L}\right).$$

Que si la vemos de forma gráfica, obtenemos la Figura 4.1.

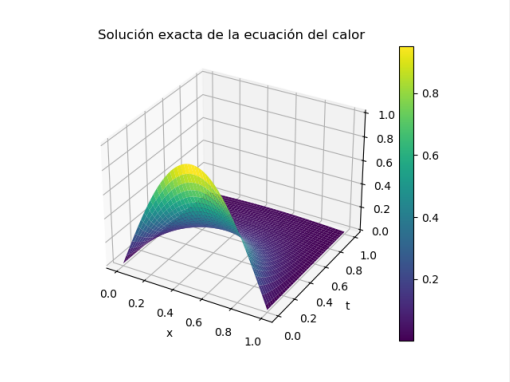


Figura 4.1: Solución de la ecuación de calor

Si nos ayudamos de la librería DeepXDE, podemos programar la red neuronal y entrenarla para que aproxime la solución de la ecuación de calor. El resultado obtenido es el obtenido en la Figura 4.2.

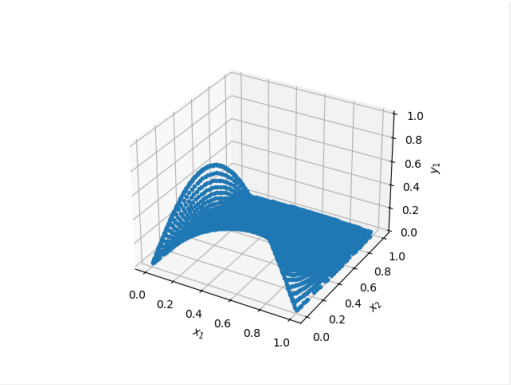


Figura 4.2: Aproximación de la solución de la ecuación de calor

Ejemplo 4.8. Consideramos un problema de convección unidimensional, una EDP hiperbólica:

$$\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} = 0, \quad x \in \Omega,\ t \in [0, T],$$
$$u(x, 0) = h(x), \quad x \in \Omega.$$

Aquí, β es el coeficiente de convección y $h(x)$ es la condición inicial. Para un β constante y condiciones de contorno periódicas, este problema tiene una solución analítica simple:

$$u_{\text{analytical}}(x, t) = \mathcal{F}^{-1}\big(\mathcal{F}(h(x))e^{-ik\beta t}\big),$$

donde $\mathcal{F}$ es la transformada de Fourier, $i = \sqrt{-1}$ y k denota la frecuencia en el dominio de Fourier. La función de pérdida general para este problema (correspondiente a la Ecuación (2.2)) es

$$\mathcal{L}(\theta) = \frac{1}{N_u}\sum_{i=1}^{N_u}\left(\hat{u} - u_i^0\right)^2 + \frac{1}{N_f}\sum_{i=1}^{N_f}\lambda_i\left(\frac{\partial \hat{u}}{\partial t} + \beta\frac{\partial \hat{u}}{\partial x}\right)^2 + \mathcal{L}_B,$$

donde $\hat{u}$ es la salida de la red neuronal, y $\mathcal{L}_B$ es la pérdida de contorno. Para condiciones de contorno periódicas con $\Omega = [0, 2\pi]$, esta pérdida es:

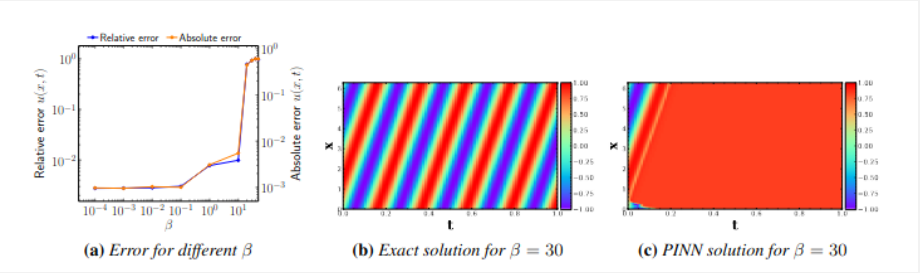
$$\mathcal{L}_B = \frac{1}{N_B}\sum_{i=1}^{N_B}\left(\hat{u}(\theta, 0, t) - \hat{u}(\theta, 2\pi, t)\right)^2.$$

Usamos las siguientes condiciones iniciales y de contorno periódico simples:

$$u(x, 0) = \sin(x),$$
$$u(0, t) = u(2\pi, t).$$

Si usamos la librería propuesta por Xu et al. en el Repositorio [2] para resolver este problema, obtenemos la aproximación se puede ver en la Figura 4.3. En esta,

se puede observar que la aproximación y la solución exacta no coinciden y además, el error es muy elevado para $\beta > 1$.



(a) *Error for different* β **(b)** *Exact solution for* $\beta = 30$ **(c)** *PINN solution for* $\beta = 30$

Figura 4.3: Solución de la ecuación de convección

Bibliografía

[1] Yuri Aikawa, Naonori Ueda y Toshiyuki Tanaka. "Improving the efficiency of training physics-informed neural networks using active learning". En: *New Generation Computing* (2024), págs. 1-22.

[2] C. *Possible Failure Modes in Physics-Informed Neural Networks*. https://github.com/a1k12/characterizing-pinns-failure-modes. 2021.

[3] Jie Hou, Ying Li y Shihui Ying. "Enhancing PINNs for solving PDEs via adaptive collocation point movement and adaptive loss weighting". En: *Nonlinear Dynamics* 111.16 (2023), págs. 15233-15261.

[4] Lu Lu et al. "DeepXDE: A deep learning library for solving differential equations". En: *SIAM Review* 63.1 (2021), págs. 208-228. DOI: 10.1137/19M1274067.

[5] Tao Luo y Qixuan Zhou. *On Residual Minimization for PDEs: Failure of PINN, Modified Equation, and Implicit Bias*. 2023. arXiv: 2310.18201 [math.AP]. URL: https://arxiv.org/abs/2310.18201.

[6] Takashi Matsubara y Takaharu Yaguchi. *Good Lattice Training: Physics-Informed Neural Networks Accelerated by Number Theory*. 2023. arXiv: 2307.13869 [cs.LG]. URL: https://arxiv.org/abs/2307.13869.

[7] Marcus Münzer y Chris Bard. *A Curriculum-Training-Based Strategy for Distributing Collocation Points during Physics-Informed Neural Network Training*. 2022. arXiv: 2211.11396 [cs.LG]. URL: https://arxiv.org/abs/2211.11396.

[8] Allan Pinkus. "Approximation theory of the MLP model in neural networks". En: *Acta numerica* 8 (1999), págs. 143-195.

[9] Shashank Subramanian et al. *Adaptive Self-supervision Algorithms for Physics-informed Neural Networks*. 2022. arXiv: 2207.04084 [cs.LG]. URL: https://arxiv.org/abs/2207.04084.