

PINNs y Δ -PINNs

Aplicado Mapas de Activación, Disipador de Calor, Ecuación de Onda

Allen Arroyo, Isidora Miranda - Grupo 1

FCFM, Universidad de Chile
MA5307 - Análisis Numérico de Ecuaciones en Derivadas Parciales
Pr. Axel Osses A. Auxiliares : Tomás Banduc & Cristóbal Godoy

Miercoles 2 de Julio, 2025

Contenidos

① Motivación

¿Por qué usamos PINNs y Δ – PINNs?

② Teoría PINNs

¿Qué es una Red Neuronal?

¿Qué es una PINNs?

Aplicación de PINNs - Mapeo de Activación Cardíaca Paper [2.]

③ Teoría Δ –PINNs

④ Aplicación Disipador Δ -PINNs Paper [1.]

Resolución con JAX Disipador

⑤ Aplicación Dominio L para Ec. de Ondas

Resolución FEM

Resolución PINNs

Resolución Δ -PINNs

⑥ Conclusión

Motivación : ¿Por qué usamos PINNs y $\Delta - PINNs$?

Las redes neuronales informadas por la física (PINNs) han emergido como una herramienta poderosa para resolver ecuaciones diferenciales, especialmente en problemas inversos con datos parciales o cuando las leyes físicas se conocen solo de forma aproximada. Sin embargo, su desempeño en problemas directos aún es superado por métodos clásicos como los elementos finitos.

A pesar de los avances recientes, muchas implementaciones de PINNs siguen limitándose a dominios geométricos simples. Esto restringe su aplicabilidad en contextos reales, donde las geometrías suelen ser complejas y de topología irregular.

Motivación: ¿Por qué usamos PINNs y Δ -PINNs?

Para abordar esta limitación, Δ -PINNs introducen una representación del dominio basada en los autovectores del operador de Laplace-Beltrami, capturando la estructura geométrica intrínseca del dominio. Esta codificación posicional:

- Preserva relaciones de proximidad geodésica.
- Permite trabajar con geometrías arbitrarias.
- Facilita la generalización a contextos tridimensionales.

Además, mediante el uso de elementos finitos sobre la malla, es posible aproximar eficientemente operadores diferenciales como el gradiente y el laplaciano, integrando así lo mejor del aprendizaje automático con las técnicas numéricas tradicionales

¿Que es una Red Neuronal?

Una red neuronal es una función parametrizada que aproxima relaciones complejas entre entradas y salidas. Que consiste de

- Capas Ocultas $H \in \mathbb{N}$
- Capa de entrada
- Capa de salida
- Dimensiones de las capas K_0, \dots, K_{H+1}
- Función de activación σ , como sigmoide, tanh, ReLU, etc.

La teoría inicialmente se base en las **Fully Connected feed forward neural network**

Funcionamiento Red Neuronal

Donde el funcionamiento es

- Capa de partida: Input $x_0 \in \mathbb{R}^{K_0}$
- 1° Capa $x_1 = \sigma(W_1x_0 + b_1) \in \mathbb{R}^{K_1}$
- 2° Capa $x_2 = \sigma(W_2x_1 + b_2) \in \mathbb{R}^{K_2}$
- ...
- H° Capa $x_H = \sigma(W_Hx_{H-1} + b_H) \in \mathbb{R}^{K_H}$
- Capa de salida $x_{H+1} = W_Hx_{H-1} + b_H \in \mathbb{R}^{K_{H+1}}$

Asi la red neuronal se representa de la siguiente manera

$$\text{NN}(x; \theta) = f_H \circ f_{H-1} \circ \cdots \circ f_1(x)$$

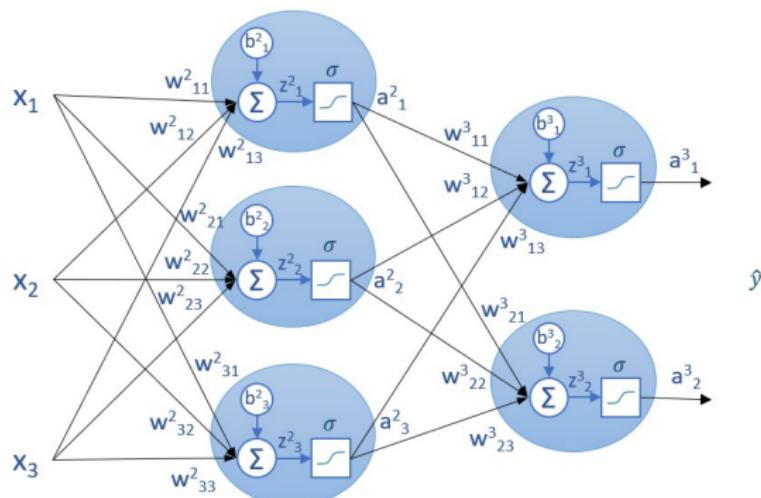
Se puede interpretar como una secuencia de matrices(weights) y vectores(biases)

donde cada capa $f_\ell(x) = \sigma(W_\ell x + b_\ell)$ aplica una transformación lineal seguida por una activación posiblemente no lineal.

Aproximación de datos red neuronal

El conjunto de todos los pesos y sesgos $\theta = \{W_\ell, b_\ell\}_{\ell=1}^H$ se ajusta durante un entrenamiento, minimizando una **función de pérdida** a elección $\mathcal{L}(\theta)$, que mide qué tan bien la red approxima los datos disponibles. Por lo que existe un problema de minimización :

$$\arg \min_{\theta} \mathcal{L}(\theta)$$



¿Qué es una PINNs?

Una Physics-Informed Neural Network (PINN) es una red neuronal que se entrena no solo con datos observacionales, sino también con conocimiento físico en forma de ecuaciones diferenciales parciales.

Supongamos que queremos aproximar una función desconocida $u(x)$, definida sobre un dominio de entrada

$$x \in \mathcal{B}$$

sujeta a una EDP:

$$\mathcal{N}[u; \lambda] = 0$$

donde \mathcal{N} es un operador diferencial y condiciones de frontera:

$$\nabla u \cdot \mathbf{n} = g_i, \quad x_i^b \in \partial\mathcal{B}$$

Se tiene que puede ser un dominio abierto y acotado $\mathcal{B} \subset \mathbb{R}^d$ ($d = 2, 3$) o puede ser una variedad suave (como una superficie).

¿Como funciona un PINNs?

Las observaciones parciales u_i estan disponibles en puntos $x_i \in \mathcal{B}$ con $i = 1, \dots, N$, algunas de estas observaciones pueden incluir puntos en el borde $\partial\mathcal{B}$

Para resolver este problema se aproxima $u(x)$ por una red neuronal

$$\hat{u}(x) = NN(x; \theta)$$

donde θ son los parámetros entrenables. El entrenamiento de estos parámetros consiste en minimizar la pérdida de tres componentes.

Función de pérdida total:

$$\mathcal{L}(\theta, \lambda) = MSE_{data} + MSE_{PDE} + MSE_b$$

Entrenamiento PINNs

- Datos observados: Este término fuerza a la red a ajustar los valores conocidos:

$$\text{MSE}_{\text{data}} = \frac{1}{N} \sum_{i=1}^N (u_i - \hat{u}(x_i; \theta))^2$$

- Residuo de la EDP:

$$\text{MSE}_{\text{PDE}} = \frac{1}{R} \sum_{i=1}^R (\mathcal{N}[\hat{u}(x_i^r; \theta); \lambda])^2$$

- Condiciones de Neumann: Se impone que el flujo a través del borde se ajuste a valores prescritos:

$$\text{MSE}_b = \frac{1}{B} \sum_{i=1}^B \left(\nabla \hat{u}(x_i^b; \theta) \cdot \mathbf{n} - g_i \right)^2$$

Aplicación de PINNs - Mapeo de activación cardíaca -

Francisco Sahli C.

[2.] (2020) Un procedimiento crucial en el diagnóstico de la fibrilación auricular es la **creación de mapas de activación** electroanatómicos.

Los métodos actuales generan estos mapas mediante interpolación utilizando unos pocos puntos de datos dispersos registrados dentro de las aurículas; no incluyen el conocimiento previo de la física subyacente ni la incertidumbre de estos registros.

En este estudio, se propone una **PINNs** para el mapeo de la activación cardíaca que considera la dinámica subyacente de propagación de ondas :

$$\|\nabla T(x)\| = \frac{1}{V(x)}$$

El mapa de activación eléctrica del corazón puede relacionarse con una onda viajera, donde el frente de onda representa la ubicación de las células que se despolarizan [2].

El tiempo de despolarización celular se denomina **tiempo de activación** y corresponde a un aumento del potencial transmembrana por encima de un umbral determinado y al inicio de la contracción celular. Los tiempos de activación de la onda viajera deben cumplir la **ecuación de Eikonal**:

$$\|\nabla T(x)\| = \frac{1}{V(x)}$$

Donde $T(x)$ es el **tiempo de activación** en un punto x y $V(x)$ la velocidad local de la onda en la misma ubicación, denominada **velocidad de conducción**.

Aplicación de PINNs-Problema de referencia sintético

Para estudiar el uso de PINNs en [2.] se resuelve un problema en 2D y 3D.

Para una malla espaciada cuadrada $[0, 1]^2$ se tiene para $\|\nabla T(x)\| \cdot V(x) - 1 = 0$ la posible solución :

$$T(x, y) = \min \left(\sqrt{x^2 + y^2}, 0.7\sqrt{(x - 1)^2 + (y - 1)^2} \right) \quad (11)$$

$$V(x, y) = \begin{cases} 1.0 & \text{if } \sqrt{x^2 + y^2} < 0.7\sqrt{(x - 1)^2 + (y - 1)^2} \\ \frac{1.0}{0.7} & \text{otro caso} \end{cases} \quad (12)$$

Que introduce una discontinuidad en la velocidad de conducción y la colisión de dos frentes de onda.

Esto gráficamente es :

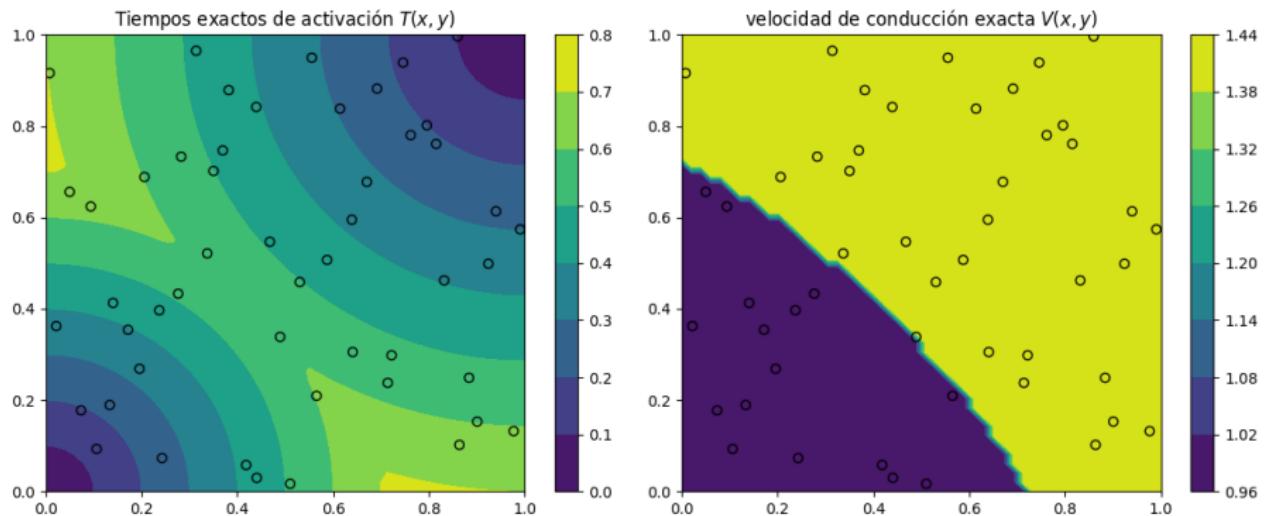


Figure: Solución exacta

Ideas con PINNs - Para resolver Ec. Eikonal

El “residuo” será :

$$R(x) := \|\nabla T(x)\| \cdot V(x) - 1 = 0$$

Se aproxima por dos redes neuronales :

$$T \approx NN_T(x, \theta_T)$$

$$V \approx NN_V(x, \theta_V)$$

Con restricción física $V_{max} = 1.5$:

$$V(x) := V_{max} \cdot \sigma(NN_V(x))$$

La función de perdida se define por :

$$\begin{aligned} \mathcal{L}(\theta_T, \theta_V) = & \frac{1}{N_T} \sum_{i=1}^{N_T} (T(x_i) - \hat{T}_i)^2 \\ & + \frac{1}{N_R} \sum_{i=1}^{N_R} R(x_i)^2 \\ & + \alpha_{TV} \frac{1}{N_R} \sum_{i=1}^{N_R} \|\nabla V(x_i)\| \\ & + \alpha_{L2} \sum_{i=1}^{N_{\theta_T}} \theta_{T_i}^2. \end{aligned}$$

La función de perdida se define por :

$$\begin{aligned}\mathcal{L}(\theta_T, \theta_V) = & \frac{1}{N_T} \sum_{i=1}^{N_T} (T(x_i) - \hat{T}_i)^2 \\ & + \frac{1}{N_R} \sum_{i=1}^{N_R} R(x_i)^2 \\ & + \alpha_{TV} \frac{1}{N_R} \sum_{i=1}^{N_R} \|\nabla V(x_i)\| \\ & + \alpha_{L2} \sum_{i=1}^{N_{\theta_T}} \theta_{T_i}^2.\end{aligned}$$

Existen N_i mediciones de tiempos \hat{T}_i . α_{TV}, α_{L2} son hiperparámetros.

- ① 1er y 2do término son MSE_{data} y MSE_{PDE}
- ② El 3er y 4to término son una regularización para el problema inverso.
- ③ 3er término permite saltos discretos en la solución, sirve para modelar regiones de conducción lenta, como parches fibróticos
- ④ 4to término regula los pesos de la red para $T(x)$.

Resumen PINNs -Ec. Eikonal - Mapeo de activación cardíaca

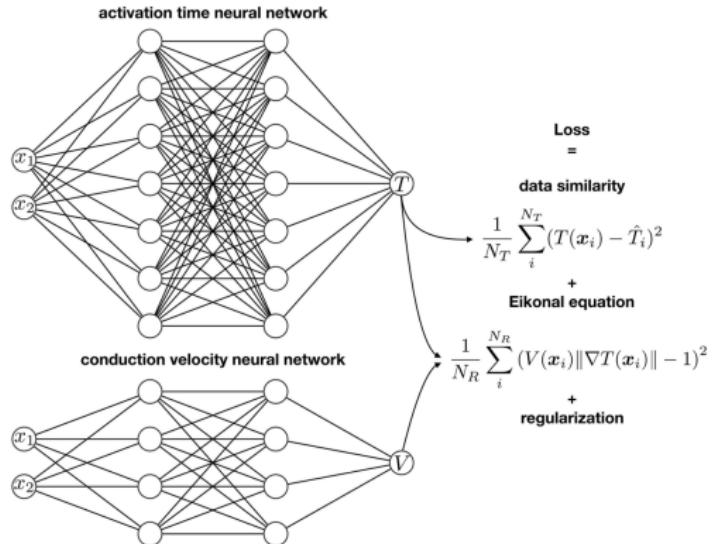


FIGURE 1 | Physics-informed neural networks for activation mapping. We use two neural networks to approximate the activation time T and the conduction velocity V . We train the networks with a loss function that accounts for the similarity between the output of the network and the data, the physics of the problem using the Eikonal equation, and the regularization terms.

Figure: Imagen original de [2.]

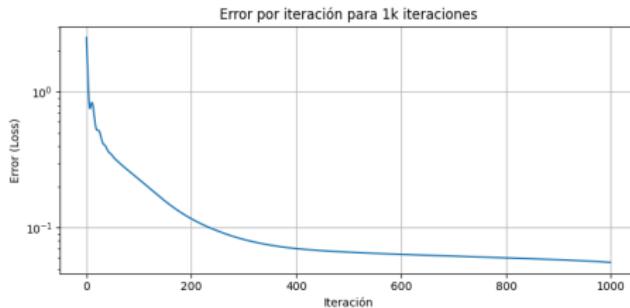
Setup - Entrenamiento de PINNs

- ① Se trabaja en tensorflow en Colab con CPU o GPU-T4 si es posible.
- ② Se generan $N = 50$ datos con un diseño de hipercubo latino(lhs) para $[0, 1]^2$, solo datos sobre los tiempos de activación \hat{T}_i .
- ③ $T_{layers} = [2, 20, 20, 20, 20, 20, 1]$, $V_{layers} = [2, 5, 5, 5, 5, 1]$

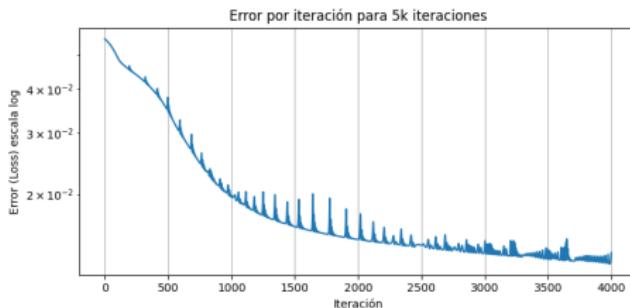
Se probó también con $T_{layers} = [2, 50, 50, 50, 50, 50, 1]$,
 $V_{layers} = [2, 10, 10, 10, 10, 1]$

- ④ $\alpha_{TV} = 10^{-7}$ y $\alpha_{L2} = 10^{-9}$
- ⑤ Entrenamos la red para 50.000 iteraciones con optimizador ADAM y $batch_{size} = 100$
- ⑥ Se agrega entrenamiento *L-BFGS* en el paper [2].
- ⑦ Para 5.000 iteraciones en Colab(solo con CPU) se demora aproximadamente 20 minutos para configuración (3).

Resultados : Error dominio cuadrado en 2D

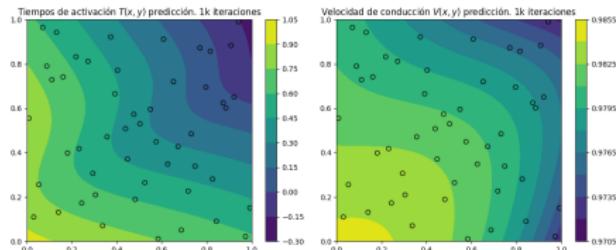


(a) Predicción 1k iteraciones escala log

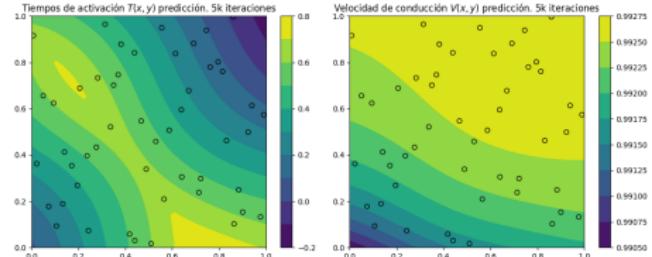


(b) Predicción 1k-5k iteraciones escala log

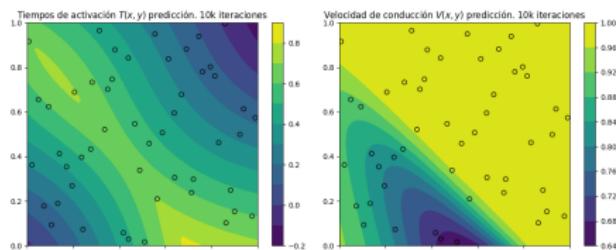
Resultados dominio cuadrado en 2D



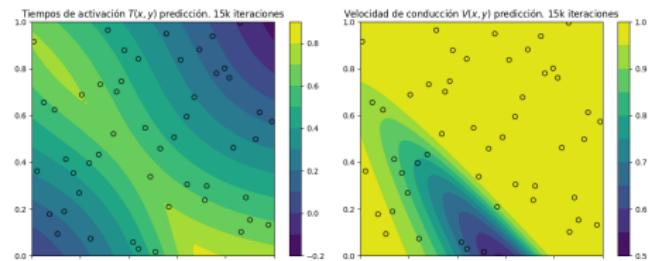
(a) Predicción 1k iteraciones



(b) Predicción 5k iteraciones

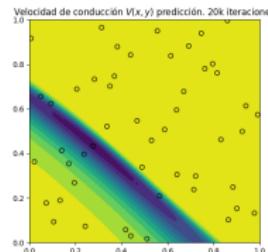
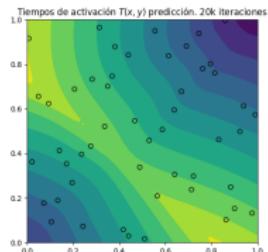


(c) Predicción 10k iteraciones

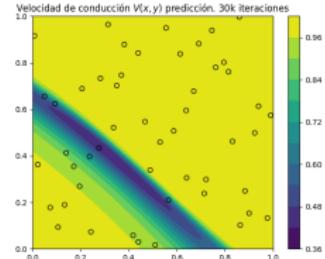
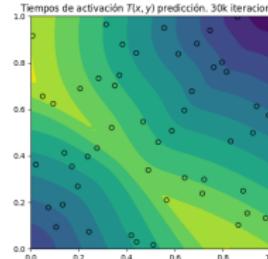


(d) Predicción 15k iteraciones

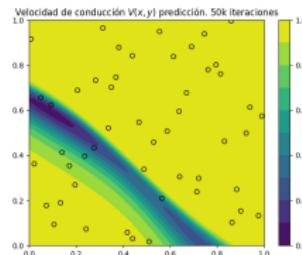
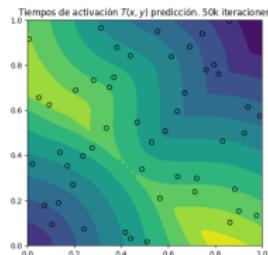
Resultados dominio cuadrado 2D



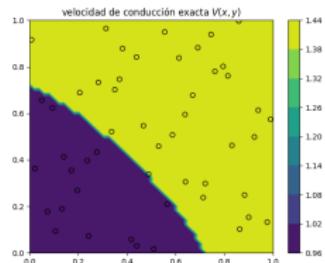
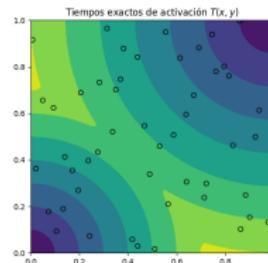
(a) Predicción 20k iteraciones



(b) Predicción 30k iteraciones



(c) Predicción 50k iteraciones



(d) Valores exactos

Comparación de métodos - Problema tiempos de activación

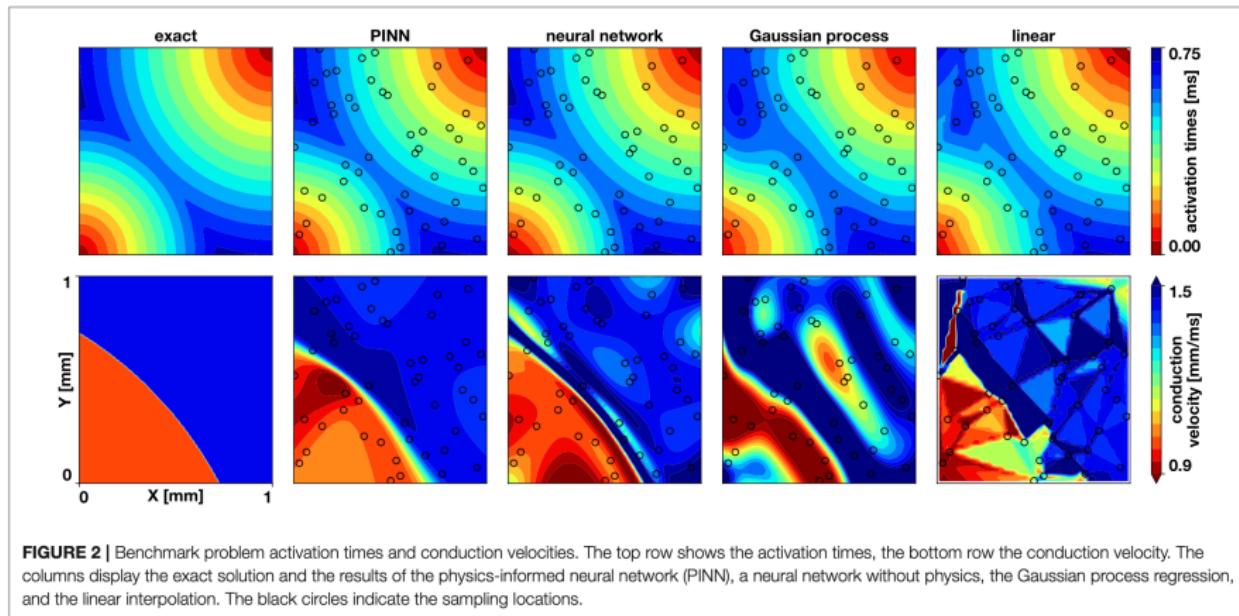


Figure: Resultados originales de [2.]

Introducción : Teoría Δ -PINNs

Para una variedad suave (smooth manifold) \mathcal{B} abierta y acotada en \mathbb{R}^d buscamos representar sus coordenadas geométricas con una codificación posicional basada en las funciones propias del operador Laplace-Beltrami $\Delta_{\mathcal{B}}(\cdot)$. Aquél se sabe [1] que codifica toda la información sobre la geometría de \mathcal{B} , hasta incluso la isometría.

De esta manera, los puntos que están cerca en la geometría permanecerán cerca en el espacio de codificación posicional.

Introducción : Teoría Δ -PINNs

El operador Laplace-Beltrami $\Delta_B(\cdot)$ es intrínseco, no requiere para su definición del espacio original y es invariante a las transformaciones isométricas (traslación y rotación) [1]. las funciones propias (eigenfunctions) del operador ν_i cumplen el siguiente problema simplificado :

$$-\Delta_S \nu_i(x) = \lambda_i \nu_i(x) \quad \text{con } x \in \mathcal{B}$$

Este problema suele ir acompañado de condiciones de contorno homogéneas de Dirichlet o Neumann.

Los valores y funciones propias son invariantes a las isometrías [1.]

Introducción : Teoría Δ -PINNs

Teorema 1 (Teorema espectral)

Por lo menos para una variedad riemanniana (es suave) compacta, el operador laplaciano con condiciones de borde Dirichlet homogéneas induce una base ortonormal de $L^2(\mathcal{B})$ dada por sus funciones propias normalizadas $\{\nu_i\}_{i=0}^\infty$

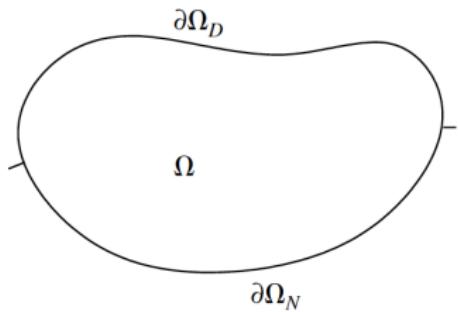
$$\begin{cases} -\Delta_S \nu_i = \lambda_i \nu_i & \text{en } \mathcal{B}, \\ \nu_i = 0 & \text{en } \partial\mathcal{B}, \end{cases}$$

donde $\{\lambda_i\}_{i=0}^\infty$ es una sucesión no-decreciente tal que $0 < \lambda_0$ y $\lambda_i \rightarrow +\infty$.

Corolario 1 (7.3.7 Allaire)

[3.] Sea Ω un conjunto abierto regular y acotado de \mathbb{R}^N cuya frontera $\partial\Omega$ se descompone en dos partes regulares disjuntas $\partial\Omega_N$ y $\partial\Omega_D$ (ver Figura 4.1). Existe una sucesión creciente $(\lambda_k)_{k \geq 1}$ de números reales positivos o nulos que tienden a infinito, y existe una base hilbertiana de $L^2(\Omega)$, $(u_k)_{k \geq 1}$, tal que cada u_k pertenece a $H^1(\Omega)$ y satisface

$$\begin{cases} -\Delta u_k = \lambda_k u_k & \text{en } \Omega \\ u_k = 0 & \text{en } \partial\Omega_D \\ \frac{\partial u_k}{\partial n} = 0 & \text{en } \partial\Omega_N. \end{cases}$$



Introducción : Teoría Δ -PINNs

De hecho la distancia geodesica $d_S(x, y)$ entre dos puntos de la variedad $(x, y) \in \mathcal{B}$ se relaciona con el núcleo de calor mediante la fórmula de Varadhan, que a su vez puede definirse mediante el operador de Laplace-Beltrami por [1.] :

$$d_S(x, y) = \lim_{t \rightarrow 0^+} \sqrt{-4t \log k_t(x, y)}, \quad k_t(x, y) = \sum_i e^{-\lambda_i t} \nu_i(x) \nu_i(y).$$

Obtención del Operador Δ_S - Utilizando FEM

Se calculan las eigenfunctions del operador de Laplace-Beltrami utilizando elementos finitos(FEM), lo que corresponde a resolver el siguiente tipo de problema con condición de borde :

$$\begin{cases} -\Delta_S \nu(x) = \lambda \nu(x) & \text{en } x \in \mathcal{B}, \\ -\nabla \nu \cdot n = 0 & \text{en } x \in \partial \mathcal{B}. \end{cases}$$

Representamos el dominio \mathcal{B} usando una malla con nodos(vértices) y conectividades(usual triángulos simples). Como se ha visto en el curso se obtiene la formulación débil del problema [1.Pag 19][3.Pag 218]

Rápidamente multiplicando por una función test $w(x)$ obtenemos :

$$-\int_{\mathcal{B}} w \left(\Delta \nu(x) + \lambda \nu(x) \right) dx = 0$$

Malla generada con Triangle

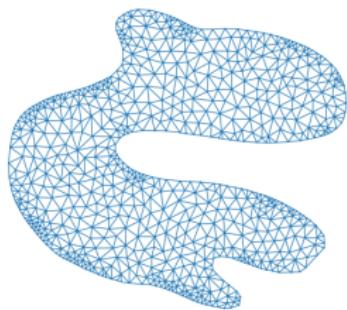


Figure:
Desde dibujo a mano

Malla generada con Triangle

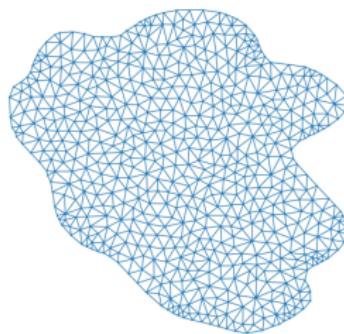


Figure:
Desde dibujo a mano

Dominio en L desde VTU

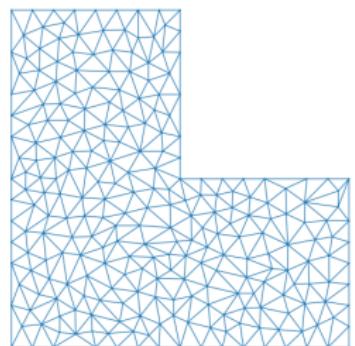


Figure: Creación manual

Aplicando formulas de Green's (Evans apéndice) tenemos la forma variacional :

$$\int_{\mathcal{B}} (\nabla w \cdot \nabla \nu + \lambda w \nu) dx - \int_{\partial \mathcal{B}} (w \nabla u \cdot n) d\sigma = 0$$

Luego de discretizar el espacio [1.Pag 20] con el método de elementos finitos [3.Pag 153] el problema se reduce a :

$$K\nu = \lambda M\nu$$

dado un espacio de aproximación de un Sobolev adecuado(Ejemplo $H_1^0(\mathcal{B})$). La matriz de rigidez K y matriz de masa M vienen definidas según:

$$\begin{cases} K_{ij} = \int_{\mathcal{B}} \nabla \nu_i \cdot \nabla \nu_j, \\ M_{ij} = \int_{\mathcal{B}} \nu_i \nu_j, \end{cases} \quad (1)$$

¿Qué hacemos ahora con los eigenfunctions?

Luego de obtener K, M con FEM se puede simplemente obtener $eigvals, eigvecs = \text{eigh}(K, M)$ con *Scipy*(tarda segundos).

Si poseemos una malla de n_{nodos} (ejemplo 1289 nodos), obtenemos

$$eigvecs = \begin{bmatrix} \vdots & \vdots & \ddots & \vdots \\ v_1 & v_2 & \cdots & v_{n_{nodos}} \\ \vdots & \vdots & \ddots & \vdots \end{bmatrix} \in \mathbb{R}^{n_{nodos} \times n_{nodos}}$$

Solo se utilizan $n_{eigs} = 50$ vectores, definiendo $eigfuncs := eigvecs|_{n_{eigs}}$

- Para cada nodo de la malla x_i con $i \in \{0, \dots, n_{nodos}\}$ tendremos 50 valores asociados respecto a los vectores propios, correspondientes a las coordenadas $v_j[i]$ para cada vector propio con $j \in \{0, \dots, n_{eigs}\}$. Es decir,

$$x_i \rightarrow X_i = [v_1[i], v_2[i], \dots, v_{50}[i]]$$

¿Qué hacemos ahora con los eigenfunctions?

Para cada indice de nodo $i \in \{0, \dots, n_{nodos}\}$ usualmente la **input** de la red (para solo una entrada) es :

$$x_i \rightarrow \hat{u}(x_i) =: u_{pre}$$

ahora se utiliza

$$X_i = [v_1[i], v_2[i], \dots, v_{50}[i]] \rightarrow \hat{u}(X_i) =: u_{pred}$$

En el algoritmo, aleatoriamente escoge una cantidad igual al tamaño $batch_size = 32$ indices idx , para luego introducir en paralelo(vectorialmente) a la red algo de la forma : $\hat{u}(X)$.

Ejemplo sencillo : si $n_{nodos} = 3$ y tenemos 3 *eigfuncs*

$$eigfuncs = \begin{pmatrix} | & | & | \\ v_1 & v_2 & v_3 \\ | & | & | \end{pmatrix} = \begin{pmatrix} 0.9 & 9 & -4 \\ 0.6 & 8 & 1.3 \\ 0.5 & 6 & -0.5 \end{pmatrix}$$

Entonces en **JAX**, si $idx = [1, 3]$ resulta :

$$X[idx] := eigufuncs[idx] = \begin{pmatrix} 0.9 & 9 & -4 \\ 0.5 & 6 & -0.5 \end{pmatrix}$$

¿Qué hacemos ahora con los eigenfunctions?

$$X[idx] := eigufuncs[idx] = \begin{pmatrix} 0.9 & 9 & -4 \\ 0.5 & 6 & -0.5 \end{pmatrix}$$

Luego este $X[idx]$ se ingresa a la red $\hat{u}(\cdot, \theta)$ y dado una matriz de pesos W_θ de dimensiones adecuada, vector bias b_θ y función de activación $\sigma := \tanh(\cdot)$ obtenemos :

$$u_{pred} := \hat{u}(X[idx], \theta) = \sigma(W_\theta X[idx] + b_\theta)$$

Que corresponde a un vector de $len(idx)$ (cantidad de nodos en aprendizaje) filas y una columna, con esto podemos calcular para **un batch** por ejemplo :

$$MSE_{data} = \frac{1}{len(idx)} \sum_{k=1}^{len(idx)} (u_{pred}[k] - Y[k])^2$$

Donde $Y := Y[idx]$ vector de soluciones en los nodos idx

Δ -PINNs Aplicado a transferencia de calor en un disipador

[1.] (Feb. 2025) Se simulama un disipador de calor con pérdidas por convección en 2D y en estado estacionario. Esto se puede expresar como :

$$\begin{cases} -\Delta u(x) = 0, & x \in \mathcal{B}, \\ u(x) = 1, & x \in \Gamma_D, \\ \nabla u(x) \cdot \mathbf{n} = 0.1 u(x), & x \in \Gamma_C, \\ \nabla u(x) \cdot \mathbf{n} = 0, & x \in \Gamma_N. \end{cases}$$

La condición **Dirichlet** representa otro cuerpo con temperatura constante, superior a la ambiente, que el disipador de calor \mathcal{B} intenta disipar. La condición **Robin** (en Γ_C) representa un flujo convectivo proporcional a la temperatura u . La condición **Neumann** se utiliza para representar la simetría del dominio.

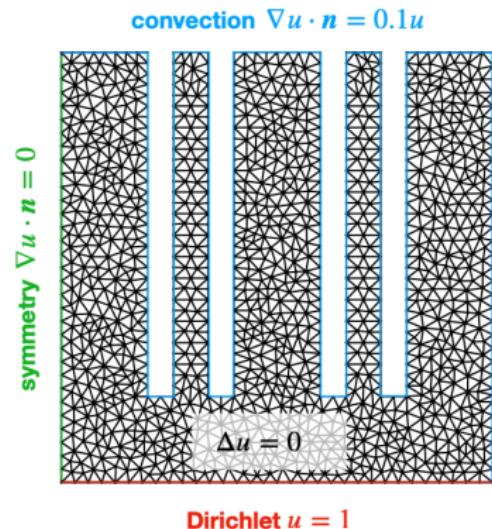


Figure: Original de [1.]

Resultados

Existen 1289 *verts* y 2183 *connectivity*(triángulos)

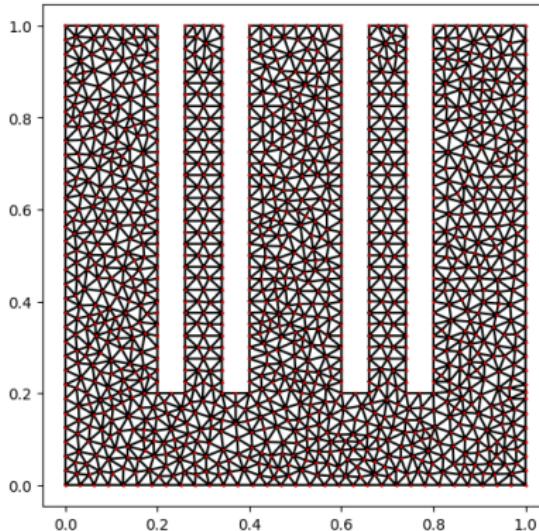


Figure: Mallado con matplotlib

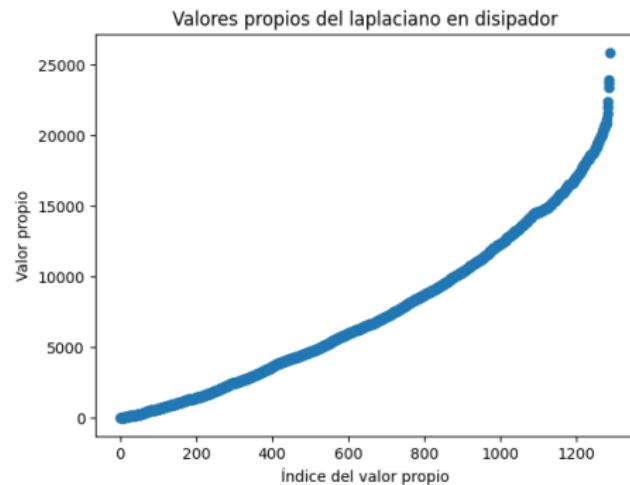


Figure: Valores propios son positivos y crecientes

Para resolver el problema se dispone inicialmente :

- ① Las soluciones Y_{borde} de $u(x)$ para todo el borde
 - ② Se define

$$n_{int} := \text{indice nodos interiores y}$$

$$n_{borde} := \text{indice nodos del borde}$$
 - ③ Se escoge la función de perdida

$$\mathcal{L}(\theta) = MSE_{PDE} + MSE_{data, borde}$$
 - ④ Se utiliza la matriz K definida antes tal que $K \approx \Delta(\cdot)$.

Con $u(x_i) \approx \hat{u}(X_i) := NN(X_i, \theta)$ para vértex x_i de la malla con $i \in \{0, \dots, \text{len}(\text{verts})\}$:

$$MSE_{PDE} = \|\Delta \hat{u}(X_{n_{int}})\|^2$$

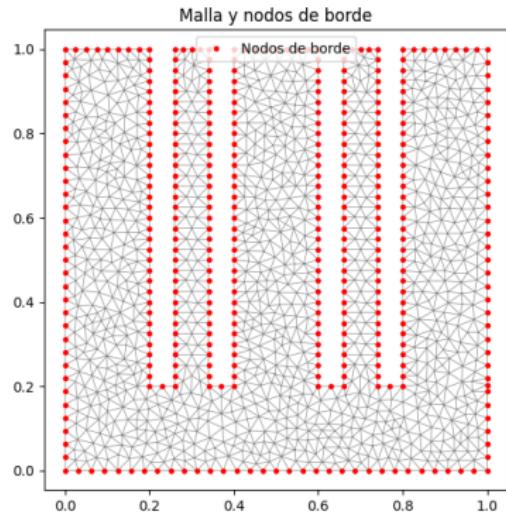
$$\approx \sum_{i \in n_{int}} \left(\sum_{j \in V(i)} K_{ij} \hat{u}_j \right)^2$$

$$MSE_{data,borde} = \|\hat{u}(X_{n_{borde}}) - Y_{borde}\|^2$$

$V(i)$ son los índices del nodo i y sus vecinos

Obs: Notar que se omite MSE_{borde} , es decir, el calculo de $\nabla \hat{u}(X_i) \cdot n$. La condición de Dirichlet se ingresa a MSE_{data} .

Tenemos 393 nodos borde y 896 interiores.



Setup - Entrenamiento Δ -PINNs aplicado a disipador

- ① Se trabaja en **JAX 0.5.2** o superior en **Colab** con CPU o GPU-T4 si es posible.
- ② Se utilizan los *verts*(nodos) y *connectivity*(triángulos). **1289** nodos y **2183** triángulos lineales.
- ③ la solución de referencia Y en el borde se puede obtener con FEM usando **FEniCS**.
- ④ $layers = [n_{eigs}, 100, 100, 100, 1]$, variamos $n_{eigs} \in \{2, 3, 5, 10, 50\}$
- ⑤ Entrenamos la red para 50.000 iteraciones con optimizador **ADAM** y $batch_{size} = 30$
- ⑥ Se toma **learning rate** de decaimiento exponencial con $\eta_0 = 10^{-3}$

$$\eta_k = \eta_0 \cdot (decay_{rate})^{\frac{k}{decay_{steps}}} ; decay_{rate} = 0.99, decay_{steps} = 100$$

- ⑦ Para 50.000 iteraciones en Colab(con CPU) se demora aproximadamente 8 minutos.

Resultados Δ -PINNs disipador $n_{eigs} = 50$

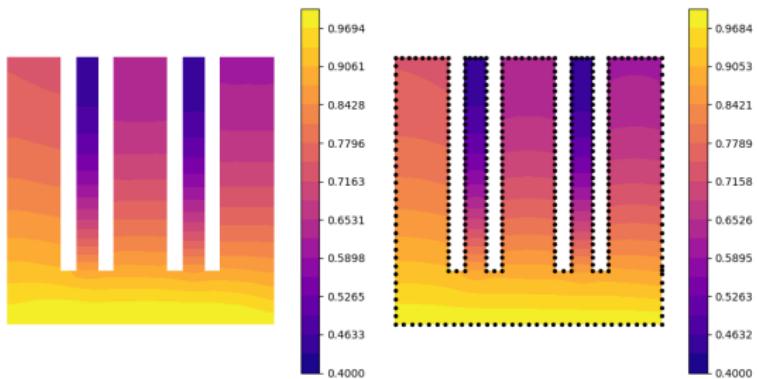
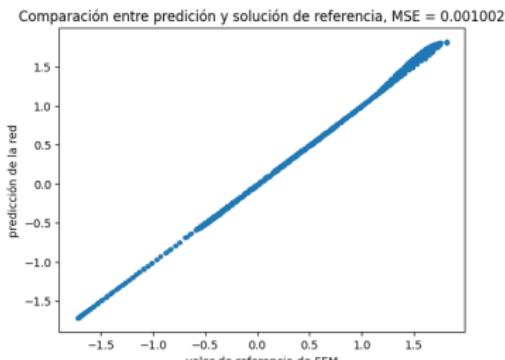


Figure: Resultado replicación experimento



Resultados Δ -PINNs disipador JAX en Colab

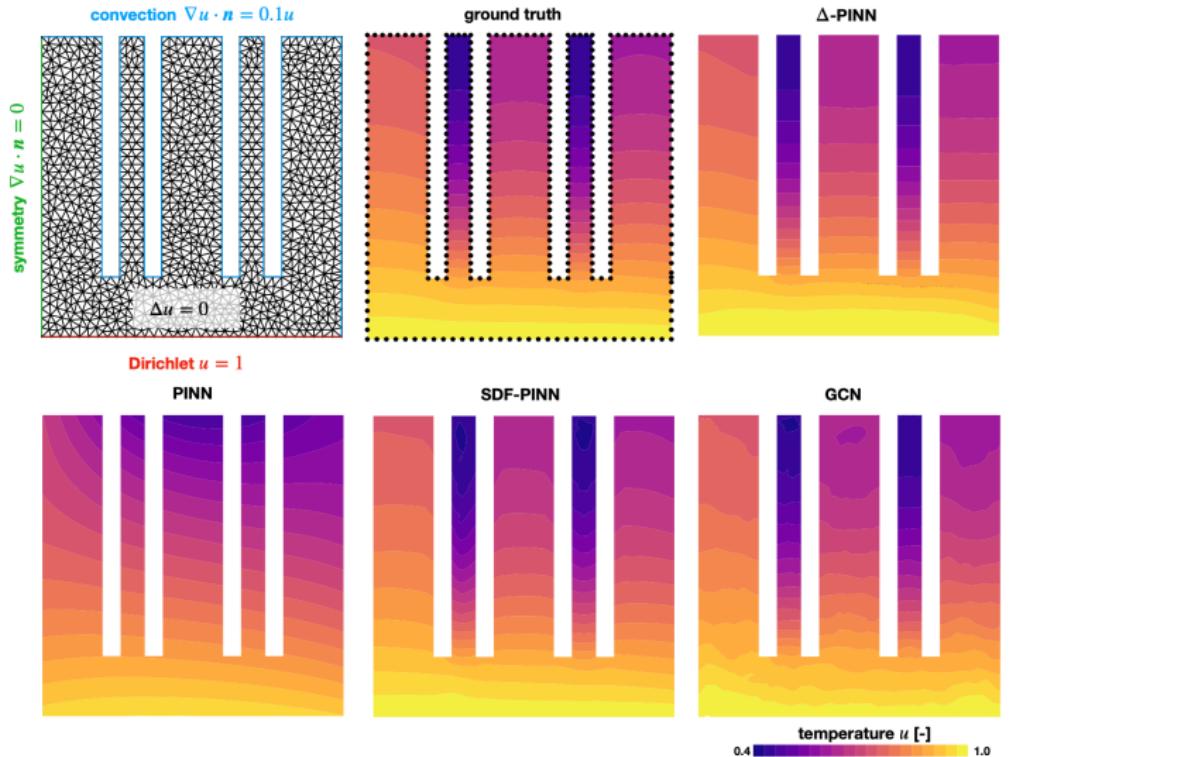
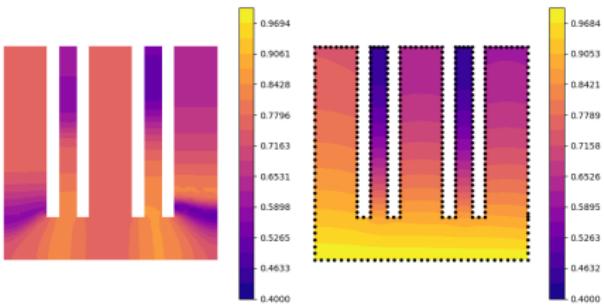
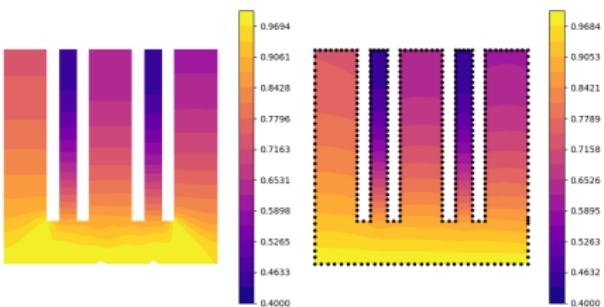


Figure: Resultado de [1.]

Para $n_{eigs} \in \{2, 3\}$ - Resultado disipador



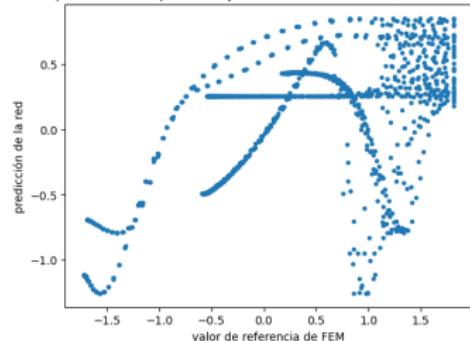
(a) Predicción 2 eigfuncs



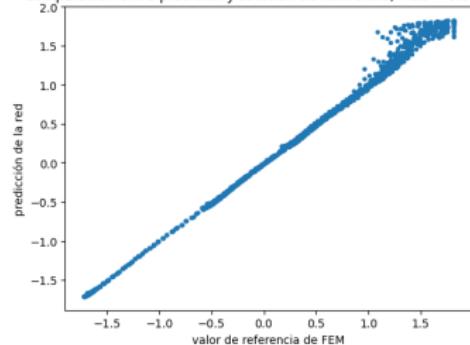
(b) Predicción 3 eigfuncs

(FCFM)

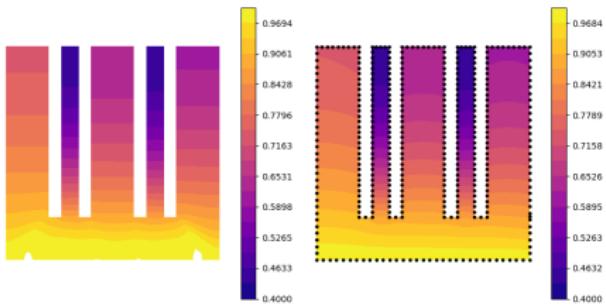
Comparación entre predicción y solución de referencia, MSE = 0.640548



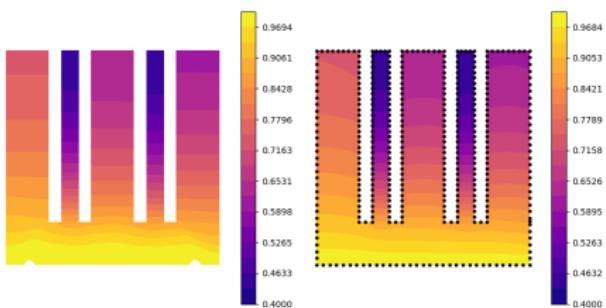
Comparación entre predicción y solución de referencia, MSE = 0.005541



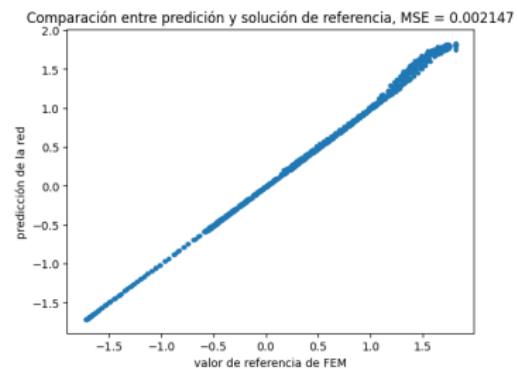
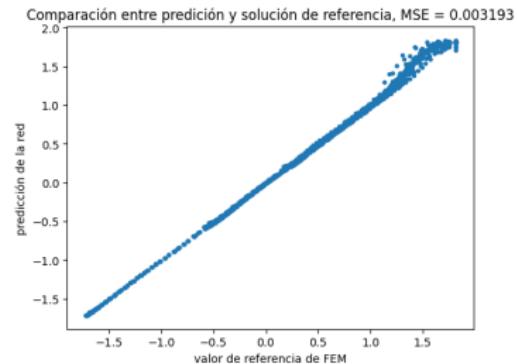
Para $n_{eigs} \in \{5, 10\}$ - Resultado disipador



(a) Predicción 5 eigfuncs



(b) Predicción 10 eigfuncs



Δ -PINNs Ec. de Calor con fuente en 3D Paper [1.]

Para una fuente de calor $f = (\sin(3\pi x) + \cos(3\pi y) + \sin(3\pi z))/3$, en la EDP $\Delta_S u + u = f$, con 99.970 triángulos y 49.987 nodos utilizando 100 *eigfucns*, dos capas de 100 neuronas.

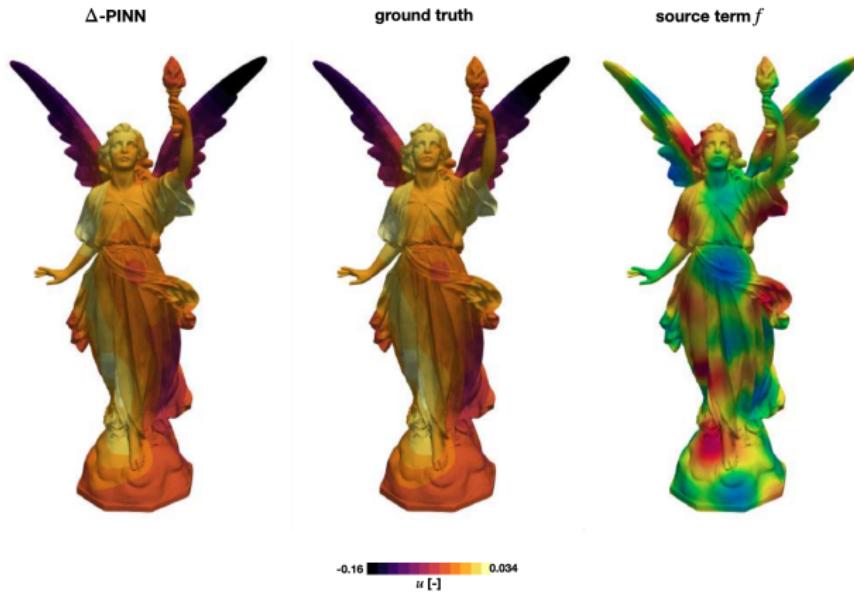


Figure 9: Solving a partial differential equation on a large mesh. We train Δ -PINN on a mesh with 99,970 triangles and 49,987 nodes. On the left, is the approximation of Δ -PINN of Eq. (16), center is the finite element solution, considered the ground truth, and right the source term form the screened Poisson Eq. (16).

Figure: Resultado de [1.]

Aplicación a EDP temporal en 2D en dominio L

Sea la siguiente EDP

$$\partial_{tt}u(x, y, t) = c^2 \Delta u(x, y, t), \quad (x, y) \in \mathcal{B}, \quad t > 0$$

$$\frac{\partial u}{\partial n}(x, y, t) = \sin(2\pi t) \quad \text{en el borde superior (top)}$$

$$\frac{\partial u}{\partial n}(x, y, t) = 0 \quad \text{en el resto de los bordes}$$

$$u(x, y, 0) = 0$$

$$\partial_t u(x, y, 0) = 0$$

Dominio EDP

Donde el dominio del problema \mathcal{B} generado con la librería **gmsh** corresponde a

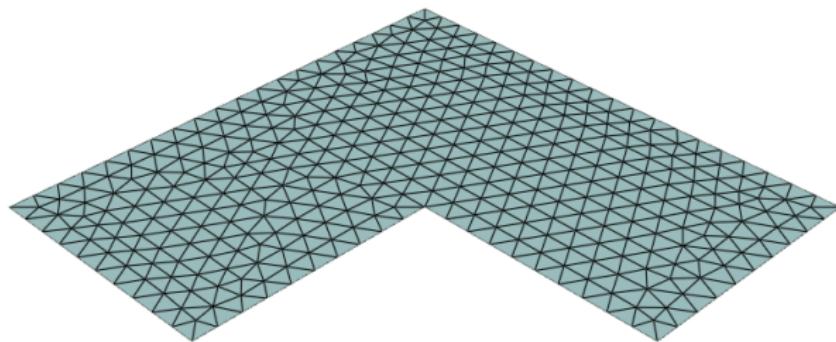


Figure: Dominio L

Donde la malla tiene 408 nodos. De los cuales 328 son interiores

Restricción de Neumann

En el top tenemos 11 nodos. La restricción de Neumann en el top se representa por la siguiente imagen

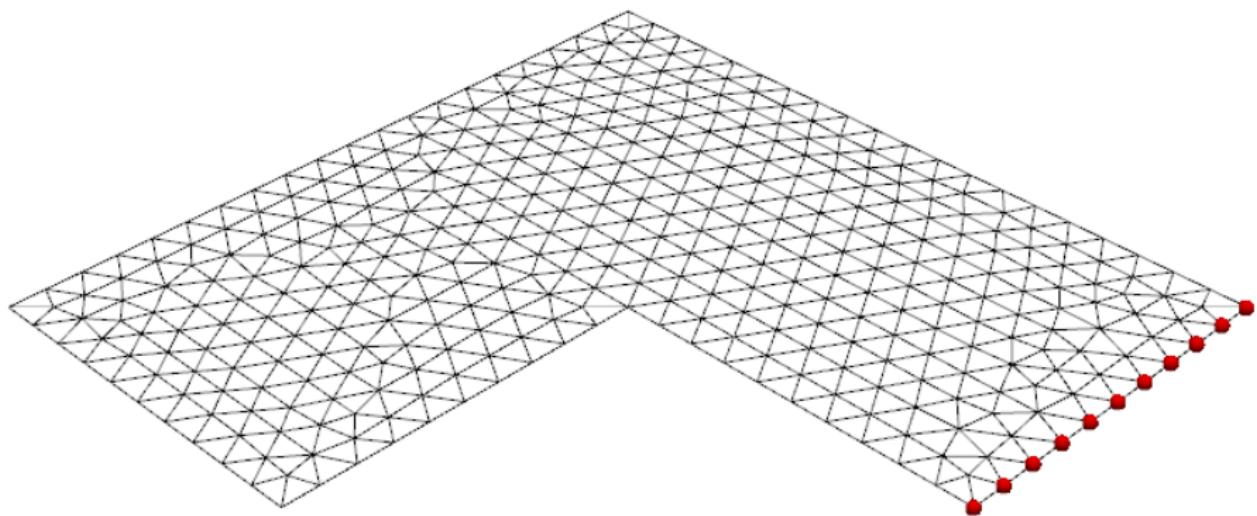


Figure: Restricción $\sin(2\pi t)$

Gif de la restricción de Neumann

Elementos finitos(FEM) para la ecuación de Ondas

Hacemos la formulación débil de nuestro problema, sea $v \in D(\mathcal{B})$

$$\begin{aligned}\int_{\mathcal{B}} (\partial_{tt} u)v dx &= c^2 \int_{\mathcal{B}} (\Delta u)v dx \\ \int_{\mathcal{B}} (\partial_{tt} u)v dx &= -c^2 \int_{\mathcal{B}} \nabla u \nabla v dx + c^2 \int_{\mathcal{B}} v \frac{\partial u}{\partial n} ds \\ \int_{\mathcal{B}} (\partial_{tt} u)v dx + c^2 \int_{\mathcal{B}} \nabla u \nabla v dx &= c^2 \int_{\mathcal{B}} v \frac{\partial u}{\partial n} ds\end{aligned}$$

Luego usando elementos finitos, hacemos una discretización espacial. Aproximamos la solución $u(x,y,t)$ como una combinación lineal de funciones base $\phi_i(x, y)$, así obtenemos

$$u(x, y, t) \approx \sum_j u_j(t) \phi_j(x, y)$$

Establecemos que $u(t) = [u_j(t)]$, luego así obtenemos un sistema matricial

$$M\ddot{u}(t) + Ku(t) = f(t)$$

Donde tenemos que

- M matriz de masa corresponde a

$$M_{i,j} = \int_{\mathcal{B}} \phi_i(x, y)\phi_j(x, y)dx$$

- K matriz de rigidez corresponde a

$$K_{i,j} = c^2 \int_{\mathcal{B}} \nabla \phi_i(x, y)\nabla \phi_j(x, y)dx$$

- f(t) es la contribución de la condición de Neumann

Hacemos discretización temporal usando el método de Leapfrog donde usamos $t_n = n\Delta t$ y escribimos

$$u^n \approx u(t_n)$$

Luego usando diferencias finitas obtenemos

$$\ddot{u}(t_n) \approx \frac{u^{n+1} - 2u^n + u^{n-1}}{\Delta t^2}$$

Remplazamos esto en el sistema matricial

$$M(u^{n+1} - 2u^n + u^{n-1}) + \Delta t^2 Ku^n = \Delta t^2 f^n$$

Así obtenemos

$$u^{n+1} = 2u^n - u^{n-1} + \Delta t^2 M^{-1} (f^n - Ku^n)$$

Resolución Solución Elementos Finitos

Para resolver nuestro problema usamos las librerías, **dolfinx.fem**, **ufl**, **petsc4py**, **numpy**. Ademas en la implementacion del método se fijo

$$u^0(x, y) = 0.5$$

$$u^{-1}(x, y) = 0.5$$

Lo cual implica que

$$\frac{\partial u}{\partial t}|_{t=0} \approx \frac{u^1 - u^{-1}}{2\Delta t} = 0$$

Solución Elementos Finitos

Donde obtenemos lo siguiente

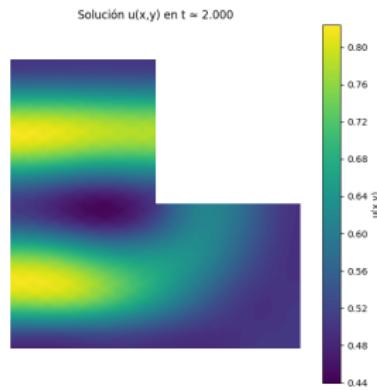
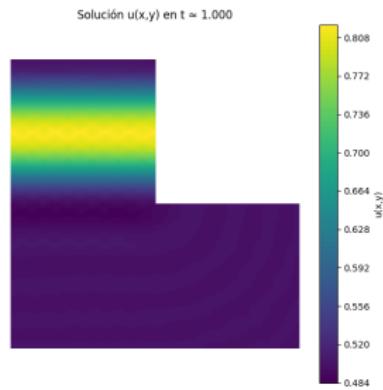
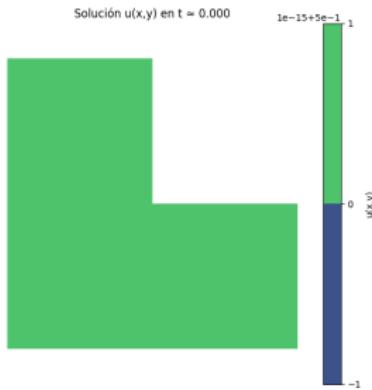
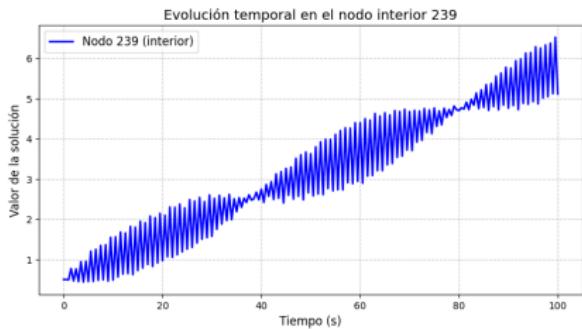


Figure: Solucion $t=0$

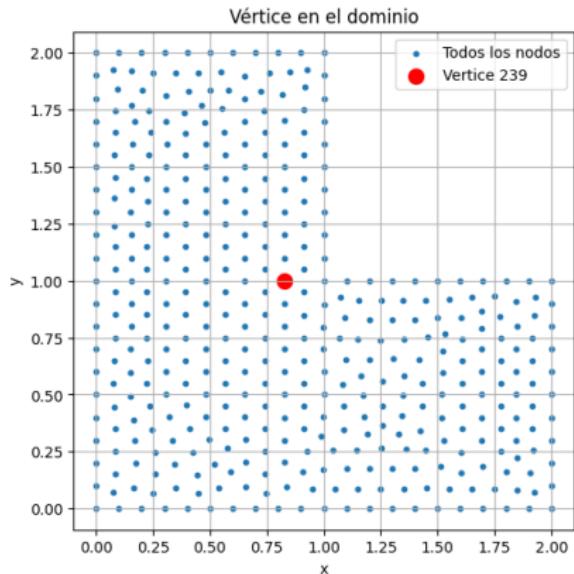
Figure: Solucion $t=1$

Figure: Solucion $t=2$

Evolución de la solución en un nodo interior



Evolución temporal de la solución en el nodo 239



Ubicación del nodo 239 en el dominio

Resolución PINNs

Al tratar de resolver el problema, entrenando la PINNs se noto que la condición de Neumann en el borde superior era muy difícil de aprender.

Para facilitar el entrenamiento, transformamos la condición de Neumann no homogénea, en una fuente localizada espacialmente en la parte superior del dominio y oscilando en el tiempo con $\sin(2\pi t)$.

Así la reformulacion de la EDP corresponde a la siguiente ecuacion

$$\partial_{tt}u = c^2 \Delta u + \sin(2\pi t) \mathbf{1}_{\{(x,y) \text{ en } \text{borde}_{top}\}}(x, y)$$

Manteniendo de las condiciones de Neumann cero en los bordes respectivos.

Generación de datos entrenamiento

Las soluciones utilizadas para entrenar la PINNs provienen del método de elementos finitos (FEM), resuelto en el mismo dominio.

El resultado FEM nos entrega una matriz $U(i, t_j)$ donde cada columna representa la solución de los nodos espaciales en el instante t_j . Se considera una discretización temporal $T=201$ pasos: t_0, t_1, \dots, t_{200} .

Para el entrenamiento consideramos conocidas las soluciones de u en los índices de tiempo

$$Indices_{tiempo} = [0, 1, 4, 5, 15, 25, 35, 50, 60, 70, 80, 85, 100]$$

En la simulación mediante elementos finitos se estableció la condición inicial $u(x, y, 0) = 0.5$, la cual fue incorporada de forma consistente en los datos utilizados para entrenar la PINN.

Dado que en nuestra red PINN la condición de Neumann no homogénea en el borde superior fue modelada como una fuente localizada dentro del término residual de la ecuación, se impusieron condiciones de Neumann homogéneas (cero) explícitamente en todos los bordes del dominio excepto en el top durante la generación de datos.

Considerando que la condición impuesta en el borde superior es especialmente difícil de aprender, se seleccionaron aleatoriamente en todo el tiempo $N_r = 10.000$ nodos interiores y del borde superior para el cálculo del residuo. Para reforzar el aprendizaje en dicha región, se concentró un 95% de esos puntos en el borde superior.

Aprendizaje PINNs

Para el aprendizaje de la PINNs usamos un learning rate de 10^{-3} , trabajamos con 3 capas de 100 neuronas y usamos como función de activación la función $\sin(\tanh(\circ))$. El comportamiento de aprendizaje de la red es el siguiente

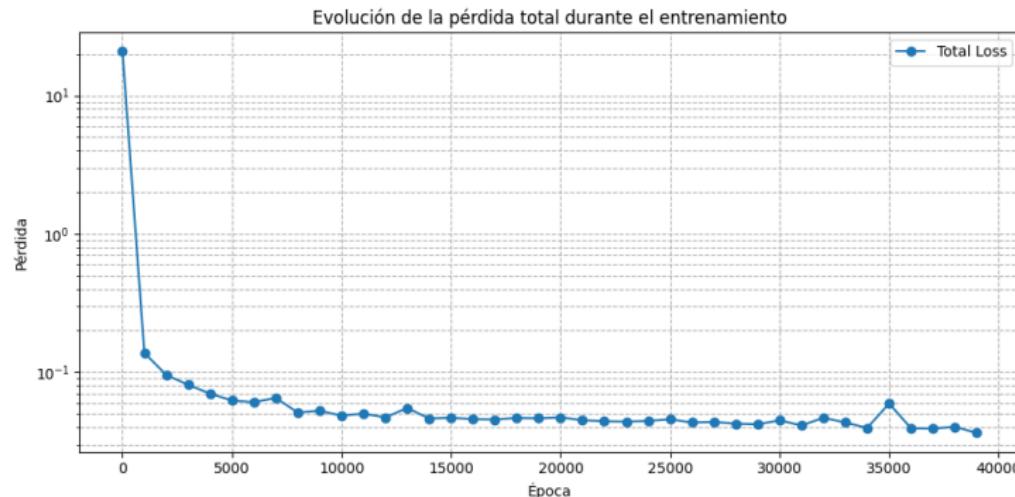
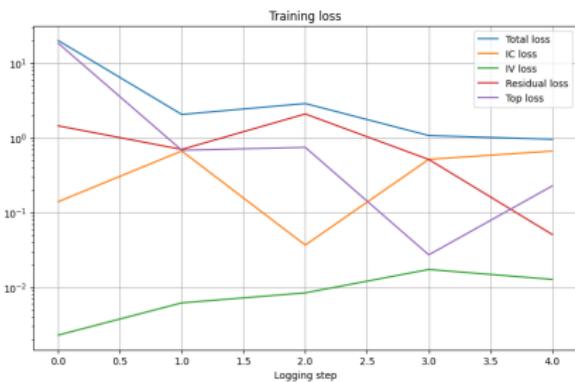
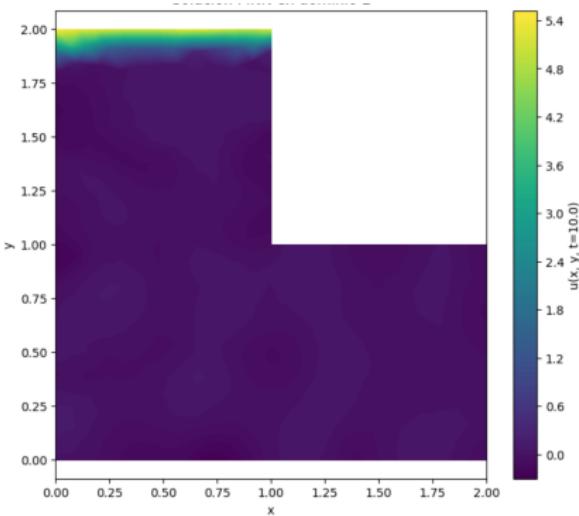


Figure: Restricción $\sin(2\pi t)$

Resolución Δ -PINNs



Errores en entrenamiento escala log



Solución en tiempo fijo $t = 10$

Conclusión

- ① Dados los problemas de la bibliografía Δ -PINNs funciona en $2D$ y $3D$ de manera eficiente cuando existen operadores $\nabla(\cdot)$ y $\Delta(\cdot)$, los cuales independientes de la entrada de la red pueden computarse utilizando métodos tipo FEM.
- ② Agregar el tiempo no es directo y existe la duda de como calcular correctamente derivadas como $\partial_{tt}\hat{u}(X, t)$.
- ③ En un apartado del Github [1.] se define la Clase [dtPINN](#) que utiliza metodos IRK(Fully implicit Runge-Kutta).
- ④ Problemas que inicialmente se ven “sencillos” pueden tener complicaciones para PINNs.

Referencias

- [1.] Sahli Costabal, F., Pezzuto, S., & Perdikaris, P. (2025, February 17). *Δ -PINNs: Physics-informed neural networks on complex geometries*. Pontificia Universidad Católica de Chile; Università di Trento; Università della Svizzera italiana; University of Pennsylvania.

Code available at: <https://github.com/fsahli/Delta-PINNs>

- [2.] Sahli Costabal, F., Yang, Y., Perdikaris, P., Hurtado, D. E., & Kuhl, E. (2020, February 28). *Physics-Informed Neural Networks for Cardiac Activation Mapping*. *Frontiers in Physics*. <https://doi.org/10.3389/fphy.2020.00042>

Code available at: <https://github.com/fsahli/EikonalNet>

- [3.] Allaire, G. (2007). Chapter 7: Eigenvalue problems. In *Numerical analysis and optimization: An introduction to mathematical modelling and numerical simulation* (A. Craig, Trans., pp. 205–221). Oxford University Press.