Evaluación de Desempeño: Diferencias Finitas vs Diferenciación Automática

Fabricio Lopretto (a1616)

Quinto Bimestre 2024

Resumen

En este trabajo se abordan dos problemas numéricos mediante el uso de métodos de redes neuronales físicas (PINN) y el método tradicional de diferencias finitas. El primer problema consiste en resolver la ecuación de Laplace con un término sinusoidal en un dominio unitario, mientras que el segundo trata la ecuación de conducción de calor con un término fuente no lineal. Ambos problemas fueron resueltos utilizando redes neuronales con diferentes arquitecturas y utilizando grillas de diferentes tamaños (5×5, 10×10, 20×20) para el método de diferencias finitas. En el caso de PINN, se utilizaron redes neuronales con configuraciones de capas [2, 3, 3, 1], [2, 5, 5, 1] y [2, 10, 10, 1]. Se compararon las soluciones obtenidas con las soluciones exactas de los problemas, observando que, para la ecuación sinusoidal, las soluciones por diferencias finitas tienden a sobrestimar la intensidad, especialmente en el centro del dominio, mientras que las soluciones por PINN subestiman la intensidad. Además, se discutió el efecto de la arquitectura de la red en la precisión de las soluciones, encontrando que la implementación de condiciones de frontera estrictas y la adimensionalización de la ecuación mejora la precisión de las soluciones. En el segundo problema, se observó que la morfología de las soluciones obtenidas mediante PINN y diferencias finitas es similar, pero con diferencias en la extensión del mínimo en la esquina superior derecha del dominio. Finalmente, se observó que la distribución de los puntos de colocación no tuvo un impacto significativo en los resultados, independientemente de la distribución utilizada o el tamaño de la red neuronal. Estos resultados destacan la capacidad de PINN para abordar problemas complejos, pero también muestran las limitaciones en cuanto a la precisión y la dependencia de la resolución espacial y la configuración de la red.

Palabras clave: diferencias finitas, PINN, EDP.

1. OBJETIVOS

El objetivo de este trabajo es comparar el desempeño de dos métodos numéricos, diferencias finitas y redes neuronales informadas por física (PINNs, por sus siglas en inglés), en la resolución de ecuaciones diferenciales seleccionadas. Para evaluar la precisión de ambos enfoques, se emplea la solución exacta heurística como referencia cuando esta esté disponible.

a) Hallar la solución por ambos métodos para tres tamaños diferentes de grilla, proponiendo diferentes arquitecturas para el método con redes neuronales informadas por física.
b) Utilizar muestreos aleatorios de los puntos de colocación con la misma cantidad de muestras que en el caso de las grillas uniformes cuando corresponda.

2. INTRODUCCIÓN

La resolución de ecuaciones en derivadas parciales (EDPs) es fundamental en el modelado de fenómenos físicos y procesos complejos en diversas áreas de la ciencia e ingeniería. Tradicionalmente, métodos numéricos como las diferencias finitas (FDM) han sido ampliamente utilizados para aproximar soluciones a este tipo de ecuaciones, ofreciendo una precisión que depende del tamaño de la grilla y del orden de la aproximación [2]. Sin embargo, en los últimos años ha surgido una nueva metodología que aprovecha el poder de las redes neuronales: las Redes Neuronales Informadas por Física (PINN). Las PINNs permiten abordar EDPs mediante la incorporación de las ecuaciones de gobierno en el entrenamiento de la red, logrando así soluciones que respetan los principios físicos del problema [3].

En este trabajo, se propone un análisis comparativo entre dos enfoques de diferenciación: la diferenciación automática (autodiff) y el método de diferencias finitas, aplicados a la resolución de problemas en estado estacionario. La diferenciación automática facilita el cálculo de derivadas parciales de manera precisa y eficiente mediante la construcción de grafos computacionales [1], siendo una herramienta fundamental para entrenar PINNs. Por otro lado, el método de diferencias finitas utiliza aproximaciones incrementales para estimar derivadas, lo que permite su implementación en una variedad de problemas discretizados.

Para esta comparación, se emplean dos problemas representativos: uno relacionado con una fuente lineal y otro con un modelo de conducción de calor con un término fuente no lineal. Cada problema se resuelve tanto con una PINN como con el método de diferencias finitas sobre grillas de distintos tamaños y configuraciones de red neuronal, y los resultados se contrastan con soluciones exactas. A través de este enfoque, se busca evaluar la precisión y las características computacionales de cada método, identificando ventajas y limitaciones en función de la configuración y la naturaleza del problema.

Este estudio tiene como objetivo proporcionar una visión integral sobre la efectividad de las PINNs combinadas con autodiff frente al método clásico de diferencias finitas, formando parte de la base para futuras investigaciones en la resolución eficiente de EDPs mediante herramientas de inteligencia artificial.

3. METODOLOGÍA

En este trabajo se proponen dos formas de la ecuación de Poisson, una dada por la ecuación:

$$\nabla^2 u = sin(\pi x)\, sin(\pi y);\ 0 < x < 1, 0 < y < 1;\ (1)$$

Con las siguientes condiciones de borde:

$$u(0, y) = u(1, y) = 0; \text{ y } u(x, 0) = u(x, 1) = 0,$$

Por otro lado, se utiliza:

$$\nabla^2 u = 0{,}5e^u;\ 0 < x < 1, 0 < y < 1;\ (2)$$

Con las siguientes condiciones de borde:

$$u(0, y) = u(x, 0) = 0; \text{ y } \partial u(1, y)/\partial x = \partial u(x, 1)/\partial y = 0,$$

Para ambos casos, se describen a continuación métodos para hallar la solución *u* mediante diferencias finitas, PINN y, para el primer caso, de forma analítica.

3.1 Método de diferencias finitas

El método se basa en el polinomio de Taylor, que su expresión para una función suave $f: R \to R$ de una variable afirma que:

$$f(xo + h) = f(xo) + f'(xo).h + (1/2).f''(xo).h^2 + \ldots + [1/(k!)].f^{(k)}(xo).h^k + R(xo, h)$$

A partir de la misma, es posible expandir $f(x)$ hasta segundo orden alrededor de un punto x evaluándose en $x + \Delta x$ y $x - \Delta x$:

$$f(x + \Delta x) = f(x) + f'(x).\Delta x + (1/2).f''(x).\Delta x^2 \ (3)$$

$$f(x - \Delta x) = f(x) - f'(x).\Delta x + (1/2).f''(x).\Delta x^2 \quad (4)$$

Luego, para encontrar una aproximación de $f'(x)$, se restan las ecuaciones anteriores de la siguiente manera:

$$f(x+\Delta x) - f(x-\Delta x) = [f(x) + f'(x).\Delta x + (1/2).f''(x).\Delta x^2] - [f(x) - f'(x).\Delta x + (1/2).f''(x).\Delta x^2]$$

Quedando:

$$f(x+\Delta x) - f(x-\Delta x) = 2.f'(x).\Delta x$$

$$\boldsymbol{f'(x) = [f(x+\Delta x) - f(x-\Delta x)]/(2.\Delta x)} \quad (5)$$

De manera similar, para obtener una aproximación de $f''(x)$, se suman las ecuaciones (3) y (4):

$$f(x+\Delta x) + f(x-\Delta x) = [f(x) + f'(x).\Delta x + (1/2).f''(x).\Delta x^2] + [f(x) - f'(x).\Delta x + (1/2).f''(x).\Delta x^2]$$

Resultando:

$$f''(x) = [f(x+\Delta x) - 2f(x) + f(x-\Delta x)]/(\Delta x^2) \quad (6)$$

La ecuación de Poisson, en ambos casos propuestos por (1) y (2), se describe como $\nabla^2 u = \partial u^2/\partial^2 x + \partial u^2/\partial^2 y$, donde $u(x,y)$ es la función desconocida que se quiere calcular. El término$\nabla^2 u$ corresponde al operador Laplaciano bidimensional, que involucra las segundas derivadas parciales de *u* con respecto a *x* e *y*. Además, este operador puede interpretarse como la divergencia del gradiente de *u*, siguiendo las definiciones planteadas en (5) y (6).

Es posible utilizar la ecuación (6) para aproximar las expresiones de $\partial u^2/\partial^2 x$ y $\partial u^2/\partial^2 y$.

Usando una grilla uniforme con espaciado *dx* en la dirección *x*, la segunda derivada de *u* respecto a *x* en el punto *(i,j)* (es decir, en el punto con coordenadas *xi* e *yj*) se aproxima por:

$$\partial^2 u/\partial x^2 = [u(i+1,j) - 2u(i,j) + u(i-1,j)]/dx^2 \quad (7)$$

De manera similar, para la segunda derivada de *u* respecto a *y*, usando un espaciado *dy* en la dirección *y*, se tiene:

$$\partial^2 u/\partial y^2 = [u(i,j+1) - 2u(i,j) + u(i,j-1)]/dy^2 \quad (8)$$

La ecuación de Poisson puede expresarse como la suma de las discretizaciones de las segundas derivadas en las direcciones *x* e *y*, obtenidas a partir de la combinación de las ecuaciones (7) y (8):

$$\nabla^2 u(x,y) = [u(i+1,j) - 2u(i,j) + u(i-1,j)]/dx^2 + [u(i,j+1) - 2u(i,j) + u(i,j-1)]/dy^2$$

Resolviendo por diferencias finitas se tiene que *dx→Δx* y *dy→Δy*. Además, al usar una grilla de puntos equiespaciada se asume que *Δx=Δy.* A partir de esto, se tiene que:

$$\nabla^2 u(x,y) = \partial u^2/\partial^2 x + \partial u^2/\partial^2 y = [u(i+1,j) + u(i-1,j) + u(i,j+1) + u(i,j-1) - 4u(i,j)]/\Delta x^2$$

Despejando *u(x,y)*, resulta:

$$u(i,j) = (1/4).[u(i+1,j) + u(i-1,j) + u(i,j+1) + u(i,j-1) - \Delta x^2 \cdot \nabla^2 u(x,y)] \quad (9)$$

Se deberá reemplazar $\nabla^2 u(x,y)$ según corresponda el caso (5) o (6).

Utilizando la ecuación (9), se realizaron tres experimentos en los que se buscó la solución de *u(x,y)* mediante este método en tres grillas equiespaciadas: 5x5, 10x10 y 20x20. Para ello, se inicializó la solución *u* en cero en cada punto de la grilla y luego se aplicó la ecuación (9) en cada uno de ellos mediante un bucle. En cada iteración, se recorrieron todos los puntos de la grilla para calcular *u* y verificar un criterio de convergencia basado en un umbral determinado por la diferencia entre el valor calculado y el valor de la iteración anterior. Si esta diferencia cae por debajo del umbral, el bucle finaliza y se considera que se ha alcanzado la solución. En caso contrario, si no se logra convergencia en la solución, el bucle finaliza tras un máximo de 10.000 iteraciones.

3.2 Método mediante redes neuronales informadas por física

En este trabajo, se emplea el enfoque de redes neuronales informadas por física (PINN) para resolver la ecuación diferencial parcial de Poisson en el dominio $\Omega \subset R^n$. Este método aprovecha el poder de las redes neuronales para aproximar soluciones a ecuaciones diferenciales, incorporando las leyes físicas que gobiernan el problema directamente en el proceso de entrenamiento de la red.

La ecuación de Poisson, $\nabla^2 u = f$, donde *u* es la solución buscada y *f* es una función dada que actúa como fuente, es un problema clásico en matemáticas aplicadas, especialmente en campos como la física y la ingeniería. El objetivo es encontrar la función *u(x)*, que satisface esta ecuación en el dominio Ω, bajo condiciones iniciales y de frontera adecuadas.

En lugar de discretizar el dominio y resolver la ecuación de Poisson utilizando métodos numéricos tradicionales, como el método de diferencias finitas, se emplea una red neuronal profunda entrenada para aproximar la solución *u(x)* de manera directa. En este enfoque, la red neuronal no solo ajusta sus parámetros a partir de los datos etiquetados si los hubiese, sino que también es informada por las ecuaciones físicas que rigen el problema.

La red neuronal está configurada para minimizar una función de pérdida que combina dos términos principales en este trabajo:

1. El término de la ecuación de Poisson: La red aprende a aproximar la solución *u(x)* tal que cumpla con la ecuación $\nabla^2 u(x) = f(x)$ en todo el dominio. Para ello, la red estima *u(x)* en puntos de malla distribuidos en Ω, y la derivada segunda de *u(x)* se calcula utilizando la diferenciación automática, luego se calcula el gradiente de la función de pérdida con respecto a los parámetros de la red neuronal.

2. El término de las condiciones de frontera: Las condiciones de frontera se imponen explícitamente en el entrenamiento de la red. Estas condiciones son cruciales para asegurar que la solución tiene solución única y que satisface las restricciones físicas en el límite del dominio. Para un dominio con frontera $\partial\Omega$, estas condiciones pueden ser Dirichlet (valores conocidos de *u* en la frontera) o Neumann (valores conocidos de la derivada normal de *u* en la frontera).

En este enfoque, se pueden incluir datos etiquetados en forma de puntos de entrenamiento donde la solución *u* es conocida (por ejemplo, en ciertos puntos del dominio o en la frontera). Estos datos etiquetados se incorporan en la función de pérdida para forzar a la red neuronal a aproximar correctamente los valores en esos puntos. Sin embargo, el mayor poder de las PINNs proviene de su capacidad para resolver las EDPs (como la ecuación de Poisson) a partir de la información inherente a la ecuación misma y las condiciones de frontera, sin necesidad de datos adicionales.

Las condiciones iniciales y de frontera se implementan de la siguiente manera:

- Condiciones de frontera de Dirichlet: Si la condición en la frontera es de Dirichlet $u = g(x)$ en $\partial\Omega$, la red neuronal penaliza las discrepancias entre su estimación de *u(x)* en la frontera y los valores *g(x)* dados.
- Condiciones de frontera de Neumann: Si la condición en la frontera es de Neumann, es decir, $\partial u/\partial n = g(x)$ en $\partial\Omega$, se penaliza la discrepancia entre el valor estimado de la derivada normal de *u(x)* y el valor *g(x)* en la frontera.

La función de pérdida final se combina en un solo término que incluye tanto la penalización por el error en la ecuación de Poisson como el error en las condiciones de frontera.

La función de pérdida *L* se compone de dos términos principales en este trabajo:

1. Pérdida de la ecuación de Poisson: Se calcula como el error entre el valor estimado de $\nabla^2\hat{u}(x)$ y la fuente *f(x)* en puntos de entrenamiento dentro del dominio Ω, es decir,

$$L_{\text{Poisson}} = \frac{1}{N_\Omega}\sum_{i=1}^{N_\Omega}\left|\nabla^2 u(x_i) - f(x_i)\right|^2,$$

   donde N_Ω es el número de puntos de malla dentro del dominio Ω y los x_i son los puntos en los que se evalúa la ecuación. La segunda derivada de *u(x)* se obtiene mediante diferenciación automática utilizando las herramientas de retropropagación de gradientes.

2. Pérdida de las condiciones de frontera: Se calcula como el error entre los valores de *u(x)* en la frontera y las condiciones de frontera especificadas *g(x)* (en el caso de Dirichlet) o el error en la derivada normal de *u(x)* (en el caso de Neumann), es decir,

$$L_{\text{BC}} = \frac{1}{N_{\partial\Omega}}\sum_{i=1}^{N_{\partial\Omega}}|u(x_i) - g(x_i)|^2 \quad \text{(para Dirichlet)}$$

   o

$$L_{\text{BC}} = \frac{1}{N_{\partial\Omega}}\sum_{i=1}^{N_{\partial\Omega}}\left|\frac{\partial u(x_i)}{\partial n} - g(x_i)\right|^2 \quad \text{(para Neumann).}$$

Aquí, $N_{\partial\Omega}$ es el número de puntos en la frontera, y los x_i son los puntos en los que se aplican las condiciones de frontera.

La función de pérdida total se obtiene combinando ambos términos:

$$L = L\ Poisson + L\ BC$$

El proceso de entrenamiento de la red neuronal sigue el algoritmo estándar de retropropagación para optimizar la función de pérdida.

1. Cálculo de gradientes: Para cada iteración de entrenamiento, la red neuronal calcula el valor de la salida *u(x)* para un conjunto de puntos de entrenamiento en el dominio Ω y en la frontera $\partial\Omega$. Luego, se calcula el error entre la salida de la red y los valores deseados según la ecuación de Poisson y las condiciones de frontera.
2. Propagación hacia atrás: Usando el algoritmo de retropropagación, los gradientes de la función de pérdida respecto a los pesos de la red se calculan. Esto se logra mediante la aplicación de la regla de la cadena, que permite calcular cómo cada peso de la red contribuye al error final. Los gradientes se calculan tanto para el término relacionado con la ecuación de Poisson como para las condiciones de frontera.
3. Actualización de pesos: Los gradientes calculados se utilizan para actualizar los pesos de la red utilizando un algoritmo de optimización, como el optimizador Adam. El proceso de retropropagación continúa hasta que la función de pérdida se minimiza hasta que cae por debajo de una cota aceptable, lo que resulta en una aproximación de la solución *u(x)* que satisface la ecuación de Poisson y las condiciones de frontera.

No obstante, se pueden implementar técnicas de mejora sobre las PINN, como por ejemplo la implementación estricta de las condiciones de borde. En este caso, en vez de entrenar a la red neuronal para que aprenda las mismas, se la entrena solo para la ecuación de gobierno (y sobre datos etiquetados y condiciones iniciales si los hubiera). De forma tal que el método queda implementado redefiniendo la solución $u\theta(x, y)$ como:

$$u\theta(x, y) = g(x, y) + \phi(x, y) \cdot u'\theta(x, y)$$

donde:

1. *g(x,y)* es la función que describe la condición de borde, es decir, *u(x,y) = g(x,y)* en $\partial\Omega$.
2. $\phi(x,y)$ es una función *clipping* que es igual a 1 en el borde donde se impone la condición de borde, y menor a 1 en el interior del dominio.
3. $u'\theta(x,y)$ es la solución entrenable que se ajustará para satisfacer sólo la EDP en el interior del dominio.

De esta forma:

- En el borde ($\phi(x,y) = 0$), la solución es exactamente *g(x,y)*, garantizando que la condición se cumpla estrictamente.
- En el interior ($0<\phi(x,y) < 1$), la red neuronal ajusta los valores para satisfacer la ecuación diferencial.

En la ecuación diferencial sinusoidal utilizada en este trabajo: $\nabla^2 u = sin(\pi.x).sin(\pi.y)$, se puede elegir $\phi(x,y)$ como un polinomio que se anule en el interior: $\phi(x,y) = x.(1-x).y.(1-y)$.

Otra técnica de mejora implementada es la adimensionalización de las EDP. En el caso de $\nabla^2 u = sin(\pi.x).sin(\pi.y)$ se puede realizar de la siguiente manera:

En las direcciones *x* e *y*, las longitudes están en el intervalo *0 < x < 1* y *0 < y < 1*. Como el intervalo ya es de longitud unitaria, se puede usar *L = 1* como una escala adimensional para ambas direcciones. La solución *u* se puede tomar el órden de magnitud *Uo* hallado en la solución exacta, definiendo una variable adimensional *u'(x,y)* que sea *u(x,y) = Uo.u'(x,y)*.

Aplicado a la EDP, resulta que el operador Laplaciano en términos de u'(x,y) es

$$\nabla^2 u(x,y) = Uo.(\partial u'^2/\partial^2 x + \partial u'^2/\partial^2 y).$$

En el segundo miembro de la ecuación original $sin(\pi.x).sin(\pi.y)$, las variables *x* e *y* están en el intervalo [0,1][0,1], por ende ya es adimensional. Con lo cual, la ecuación adimensionalizada resulta:

$$(\partial u'^2/\partial^2 x + \partial u'^2/\partial^2 y = (1/Uo).sin(\pi.x).sin(\pi.y)$$

3.3 Solución exacta

En el caso de la búsqueda de la solución de la ecuación (5), se puede describir la solución exacta de la misma mediante la solución analítica de *u* dada por *u(x,y) = −[1/(2.π²)].sin(□.□).sin(□.□)*. Para corroborar que es solución de la ecuación (5), se opera derivando.

En primer lugar, se trabaja con las derivadas primera y segunda de *u* con respecto a *x*:

$$\partial u/\partial x = -[1/(2.\pi^2)].cos(\pi.x).sin(\pi.y).\pi = -[1/(2.\pi)].cos(\pi.x).sin(\pi.y)$$

$$\partial^2 u/\partial x^2 = -[1/(2.\pi)].[-sin(\pi.x).\pi].sin(\pi.y) = (1/2).sin(\pi.x).sin(\pi.y)$$

De manera similar, se trabaja con las derivadas primera y segunda de *u* con respecto a *y*:

$$\partial u/\partial y = -[1/(2.\pi^2)].sin(\pi.x).cos(\pi.y).\pi = -[1/(2.\pi)].sin(\pi.x).cos(\pi.y)$$

$$\partial^2 u/\partial y^2 = -[1/(2.\pi)].sin(\pi.x).[-sin(\pi.y).\pi] = (1/2).sin(\pi.x).sin(\pi.y)$$

Realizando la suma, se tiene que:

$$\partial^2 u/\partial x^2 + \partial^2 u/\partial y^2 = (1/2).sin(\pi.x).sin(\pi.y) + (1/2).sin(\pi.x).sin(\pi.y) = sin(\pi.x).sin(\pi.y)$$

Que es igual a la expresión dada por la ecuación (5) de $\nabla^2 u$.

Al igual que métodos previos, se calculó el valor de *u(x,y)* con la solución exacta para cada punto de grilla, utilizando las resoluciones espaciales de 5x5, 10x10 y 20x20.

4. RESULTADOS

4.1 Soluciones de la ecuación sinusoidal

4.1.1 Utilizando una grilla de 5x5

Luego de realizar los cálculos por los métodos propuestos indicados en la sección 3, se obtuvieron los siguientes campos escalares de *u*.

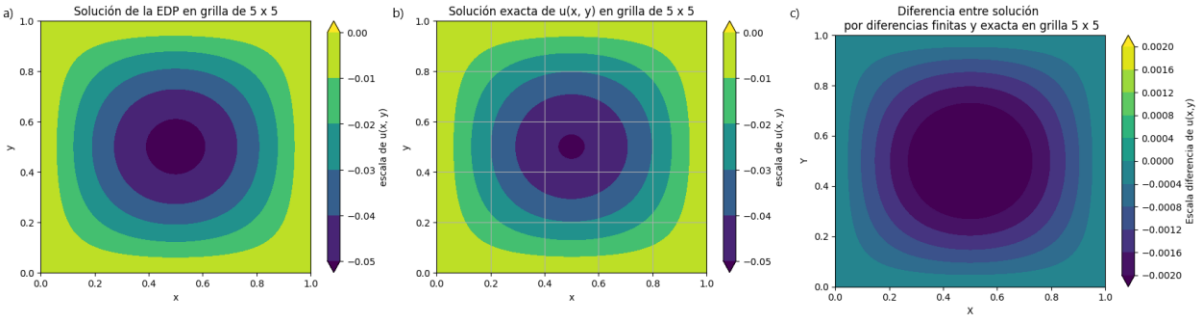


***Fig.1**: Campos de de solución de u en grilla de 5x5. a) diferencias finitas, b) solución exacta, c) diferencia entre soluciones.*

Se observó que el método de diferencias finitas arrojó buenos resultados, con la misma morfología del campo escalar y orden de magnitud dado por la solución exacta. Además, en todo el dominio los valores que toma *u* son negativos con un mínimo absoluto en el centro de la región. No obstante, a medida que se desplaza al centro del dominio, se identificó valores negativos de diferencia de soluciones. Esto mostró que hacia esa zona del dominio, la solución dada por diferencias finitas tiende a sobrestimar, en intensidad, el valor de *u*.

Por otro lado, se buscó la solución utilizando PINN. Para esto se utilizó tres diferentes configuraciones de las capas ocultas, trabajando con 3, 5 y 10 neuronas por cada capa oculta en cada caso. En la figura 2 se muestran los resultados de la red con estas diferentes configuraciones, y sus diferencias con la solución exacta.

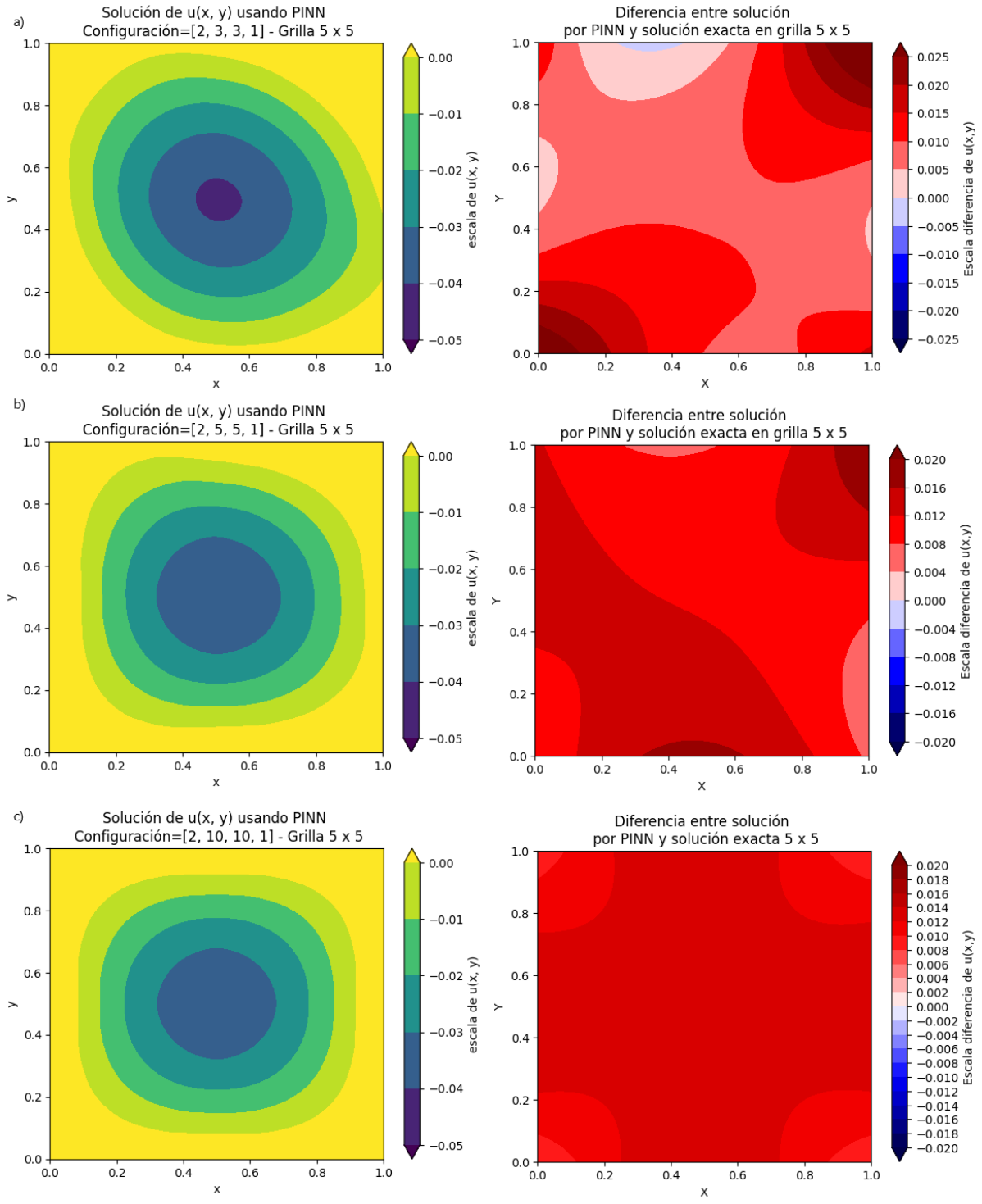


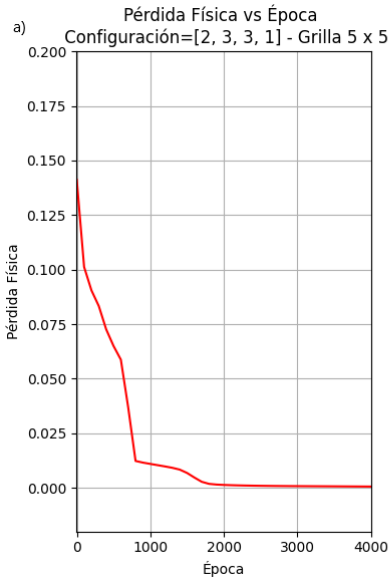
***Fig.2**: Campos de solución de u(x,y) por PINN, de solución exacta y de diferencia entre estos en grilla de 5x5 para configuración PINN de: a) [2, 3, 3, 1], b) [2, 5, 5, 1] y c) [2, 10, 10, 1].*

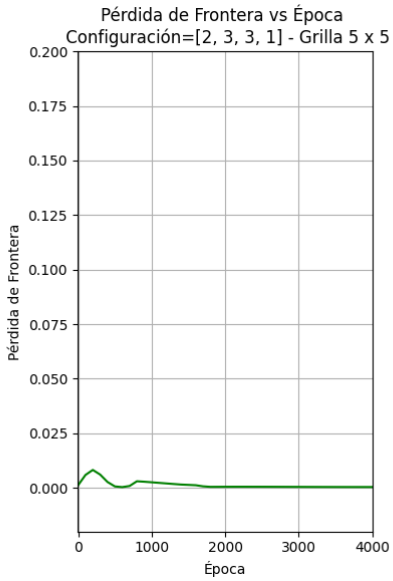
Se observó que la solución obtenida mediante el método PINN reproduce la misma estructura que la solución exacta, presentando un mínimo en el centro del dominio y valores que tienden a cero en los contornos, en concordancia con las condiciones de borde impuestas. Sin embargo, en el caso de configuraciones con tres neuronas por capa oculta, la solución no se ajusta completamente a dichas

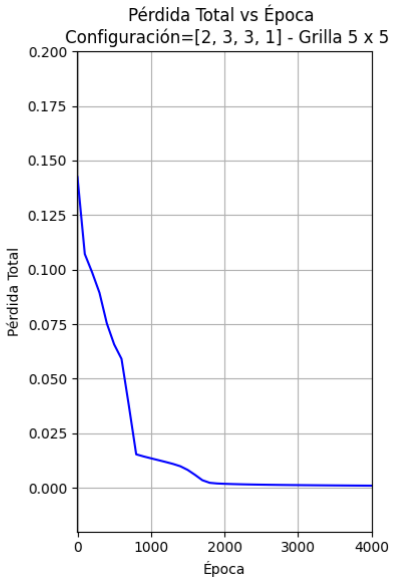
condiciones de borde. Además, al analizar la diferencia entre la solución obtenida por PINN y la solución exacta, se identificaron valores positivos distribuidos en todo el dominio, lo que indica que el método PINN tiende a subestimar la magnitud de *u*. Por otro lado, las diferencias entre ambas soluciones tienden a disminuir y homogeneizar al modificar la cantidad de neuronas en las capas ocultas, lo que sugiere una limitada sensibilidad de la solución a este parámetro de configuración.

Por otro lado, se generaron las curvas de pérdida para cada componente y configuración de la red.

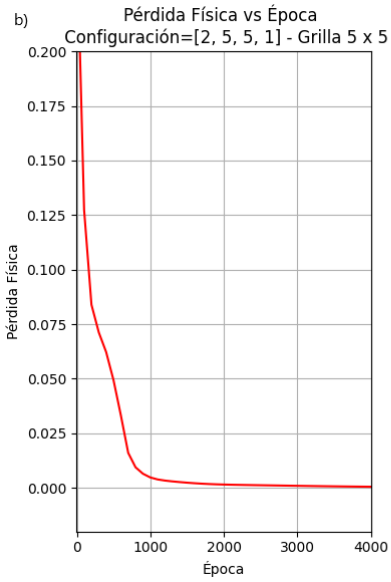
a)

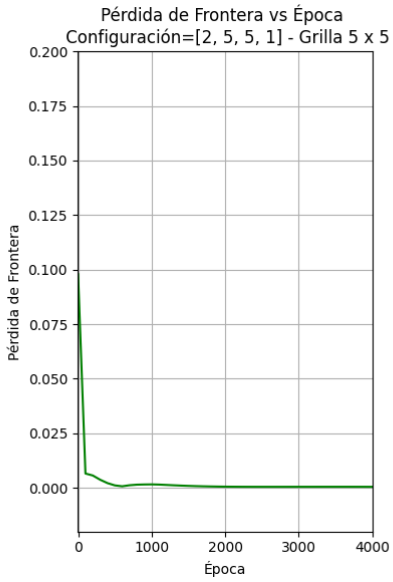

Pérdida Física vs Época
Configuración=[2, 3, 3, 1] - Grilla 5 x 5
Pérdida Física
Época

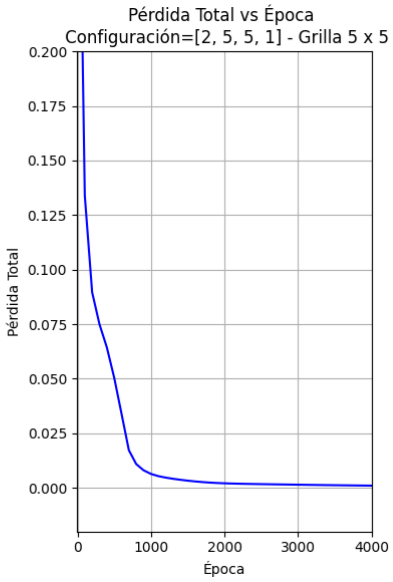

Pérdida de Frontera vs Época
Configuración=[2, 3, 3, 1] - Grilla 5 x 5
Pérdida de Frontera
Época


Pérdida Total vs Época
Configuración=[2, 3, 3, 1] - Grilla 5 x 5
Pérdida Total
Época

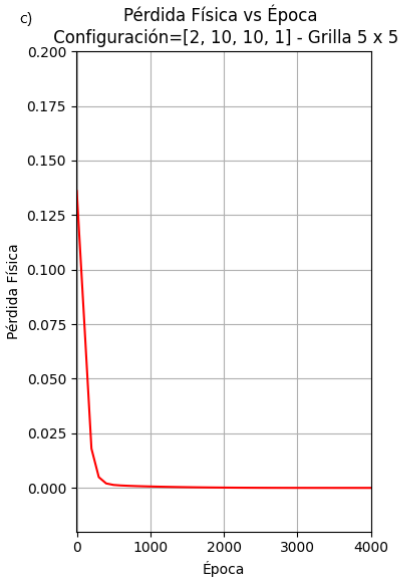
b)

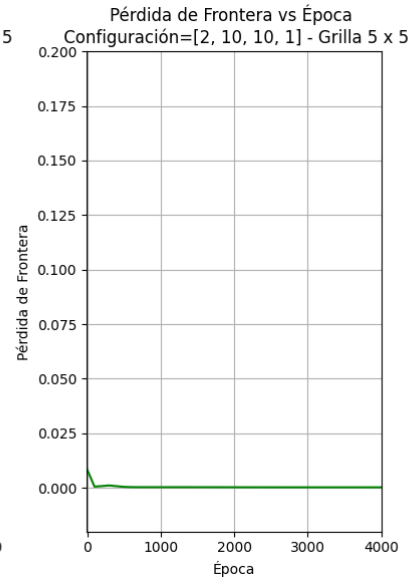

Pérdida Física vs Época
Configuración=[2, 5, 5, 1] - Grilla 5 x 5
Pérdida Física
Época


Pérdida de Frontera vs Época
Configuración=[2, 5, 5, 1] - Grilla 5 x 5
Pérdida de Frontera
Época


Pérdida Total vs Época
Configuración=[2, 5, 5, 1] - Grilla 5 x 5
Pérdida Total
Época

c)


Pérdida Física vs Época
Configuración=[2, 10, 10, 1] - Grilla 5 x 5
Pérdida Física
Época


Pérdida de Frontera vs Época
Configuración=[2, 10, 10, 1] - Grilla 5 x 5
Pérdida de Frontera
Época

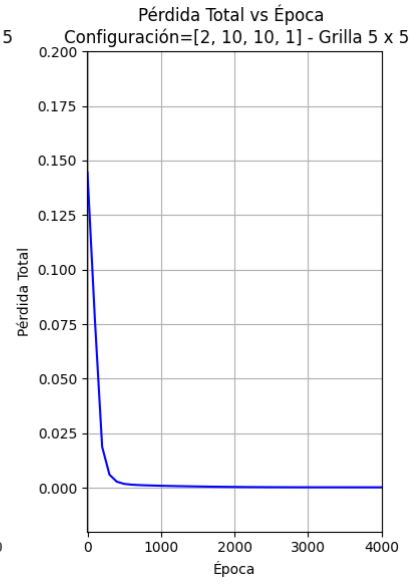

Pérdida Total vs Época
Configuración=[2, 10, 10, 1] - Grilla 5 x 5
Pérdida Total
Época

***Fig 3.**: Curvas de pérdida total, de función de gobierno y de condiciones de borde en grilla de 5x5 para diferentes configuraciones de PINN: a) [2, 3, 3, 1], b) [2, 5, 5, 1] y a) [2, 10, 10, 1].*

En el caso de una configuración con tres neuronas por capa oculta, se observó que el modelo requiere aproximadamente 2000 épocas para alcanzar el mínimo de la función de pérdida. En contraste, con cinco neuronas por capa oculta, el número de épocas necesarias disminuye a aproximadamente 1200, mientras que con diez neuronas por capa oculta se reduce aún más, a cerca de 500 épocas. Estos resultados destacan la importancia de aumentar el ancho de las capas en la red PINN para acelerar la convergencia. Asimismo, en todos los casos analizados, se observó que el término de pérdida asociado a la función de gobierno requiere más épocas para converger al mínimo en comparación con el término de pérdida correspondiente a las condiciones de borde.

4.1.2 Utilizando una grilla de 10x10

De manera similar, luego de realizar los cálculos de la solución por diferencias finitas, de la solución exacta y su diferencia, se generaron los campos respectivos.

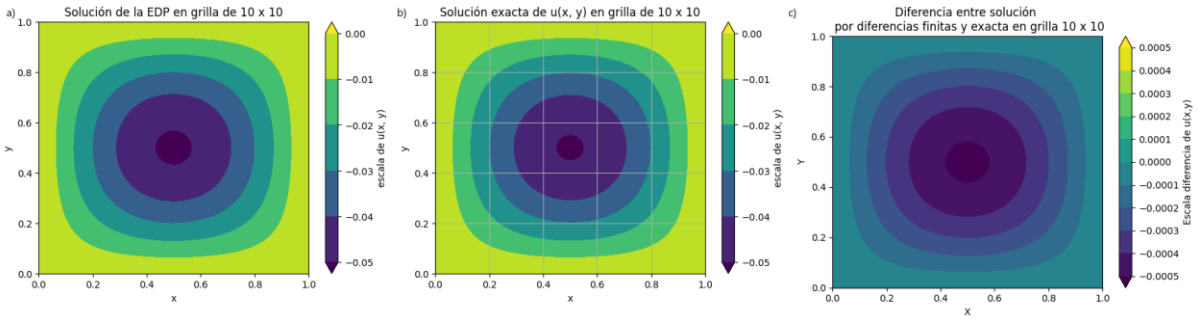


***Fig. 4**: Campos de de solución de u en grilla de 10x10. a) diferencias finitas, b) solución exacta, c) diferencia entre soluciones.*

Al igual que en el caso de 5x5, los campos mostraron la misma morfología. No obstante, la forma del mínimo absoluto se ajustó mejor a la forma suave sinusoidal. En cuanto a las diferencias, se vio que la intensidad de las mismas disminuyó. Esto puso en evidencia que al aumentar la resolución espacial el método por diferencias finitas se ajustó mejor a la solución exacta.

Luego, al trabajar con la red PINN en las configuraciones de capas ocultas propuestas, se llegó a los resultados de la figura 5.

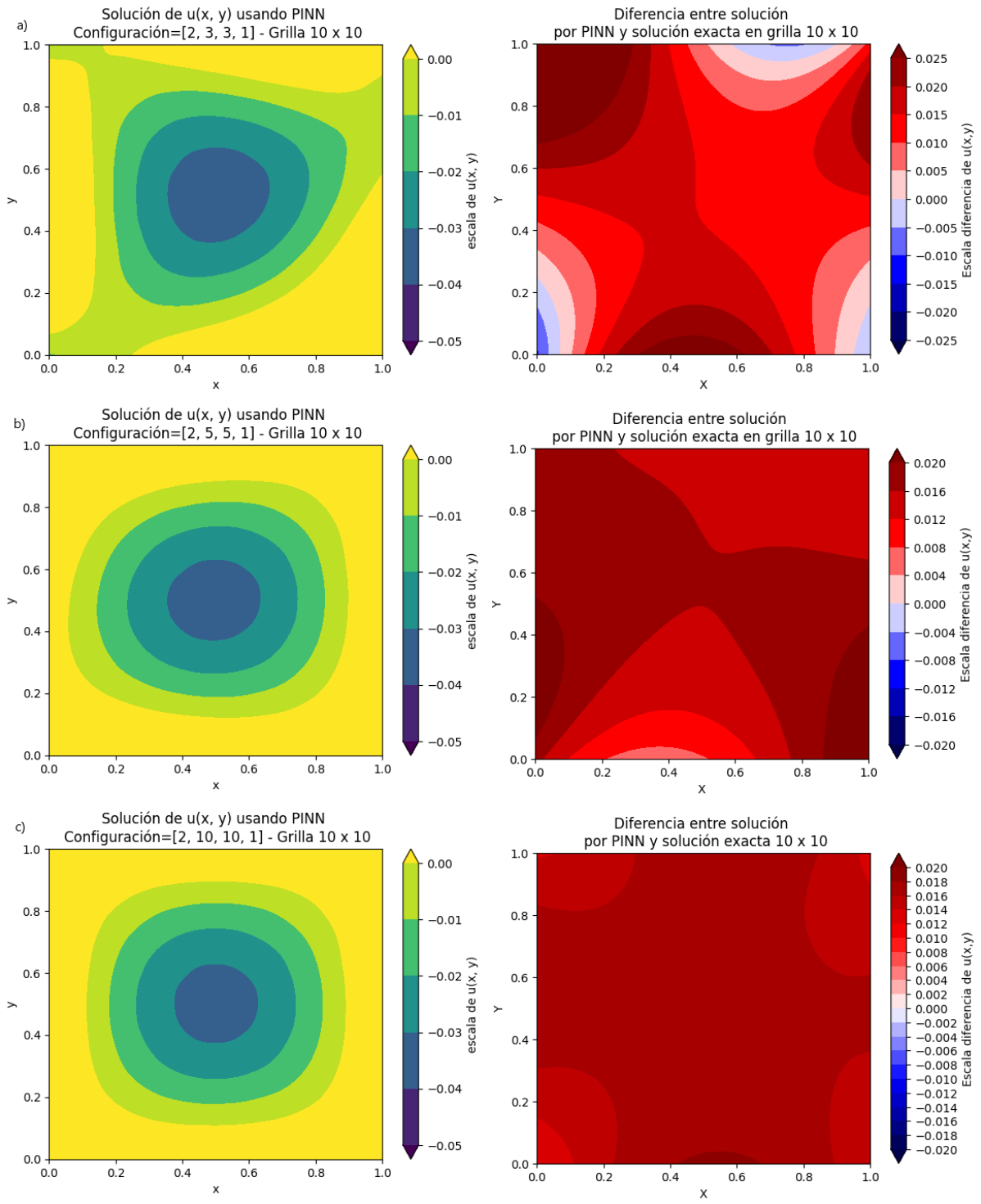


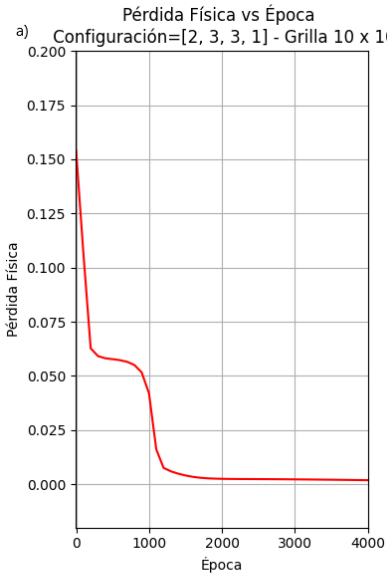
***Fig. 5**: Campos de solución de u(x,y) por PINN, de solución exacta y de diferencia entre estos en grilla de 10x10 para configuración PINN de: a) [2, 3, 3, 1], b) [2, 5, 5, 1] y c) [2, 10, 10, 1]*

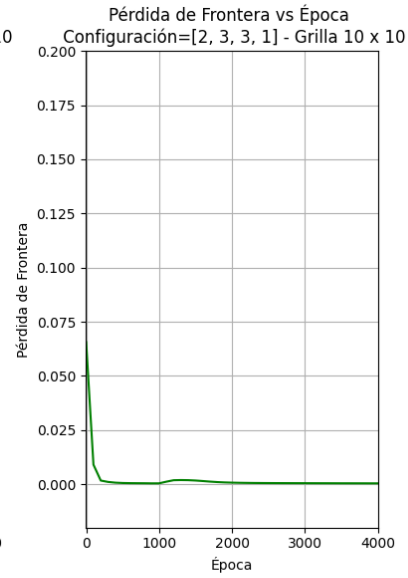
Al igual que en el caso anterior, al utilizar una malla de 10x10, la solución obtenida mediante PINN mostró una estructura similar a la de la solución exacta. No obstante, persistieron las dificultades para ajustar correctamente las condiciones de borde cuando se emplean tres neuronas por capa oculta. Además, las diferencias entre las soluciones obtenidas con las diversas configuraciones de la

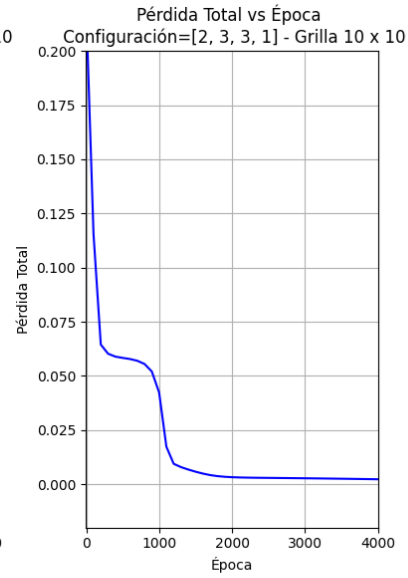
red se mantuvieron principalmente positivas, mostrando una reducción en intensidad y una homogeneización al aumentar el ancho de la red.

Nuevamente, se generaron las curvas de pérdida vs las épocas para cada componente de pérdida y para cada configuración de capas.

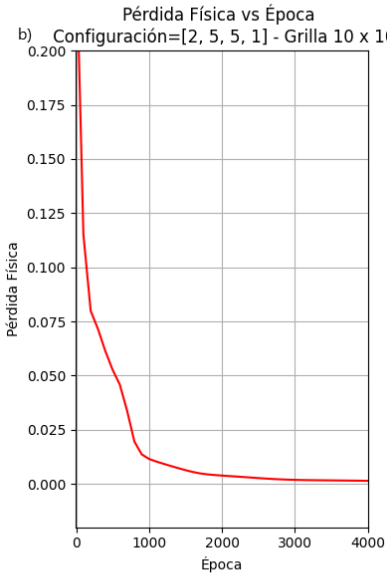
a)

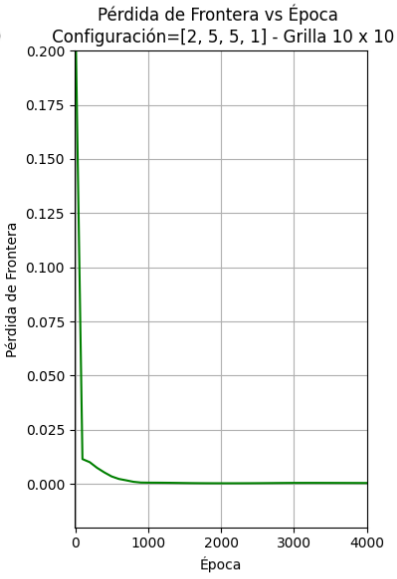

Pérdida Física vs Época
Configuración=[2, 3, 3, 1] - Grilla 10 x 10
Pérdida Física
Época

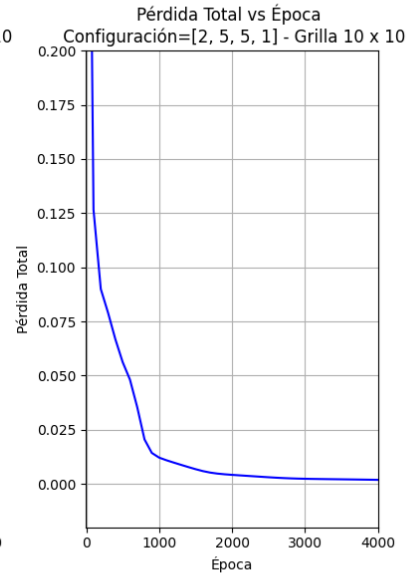

Pérdida de Frontera vs Época
Configuración=[2, 3, 3, 1] - Grilla 10 x 10
Pérdida de Frontera
Época


Pérdida Total vs Época
Configuración=[2, 3, 3, 1] - Grilla 10 x 10
Pérdida Total
Época

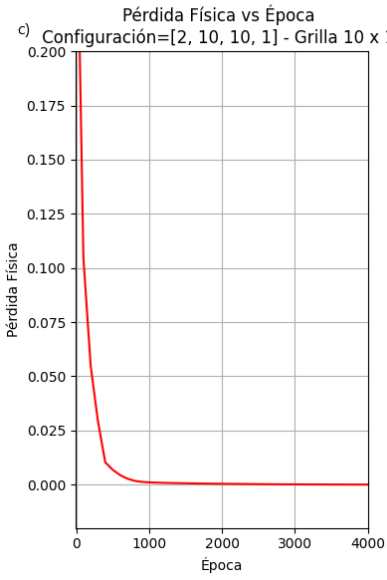
b)

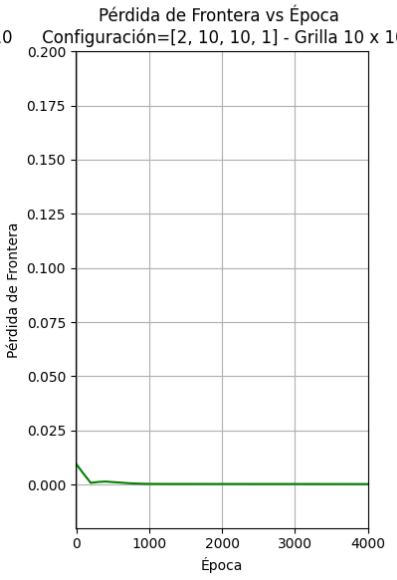

Pérdida Física vs Época
Configuración=[2, 5, 5, 1] - Grilla 10 x 10
Pérdida Física
Época


Pérdida de Frontera vs Época
Configuración=[2, 5, 5, 1] - Grilla 10 x 10
Pérdida de Frontera
Época


Pérdida Total vs Época
Configuración=[2, 5, 5, 1] - Grilla 10 x 10
Pérdida Total
Época

c)


Pérdida Física vs Época
Configuración=[2, 10, 10, 1] - Grilla 10 x 10
Pérdida Física
Época


Pérdida de Frontera vs Época
Configuración=[2, 10, 10, 1] - Grilla 10 x 10
Pérdida de Frontera
Época

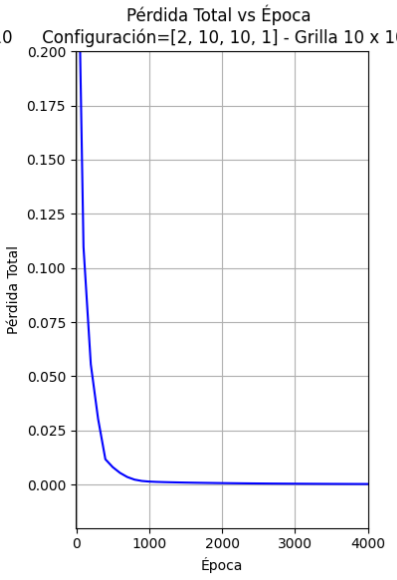

Pérdida Total vs Época
Configuración=[2, 10, 10, 1] - Grilla 10 x 10
Pérdida Total
Época

***Fig 6.**: Curvas de pérdida total, ecuación de gobierno y de condiciones de borde en grilla de 10x10 para diferentes configuraciones de PINN: a) [2, 3, 3, 1], b) [2, 5, 5, 1] y a) [2, 10, 10, 1]*

De manera similar al caso con la malla de 5x5, al incrementar el número de neuronas en las capas ocultas, se observó una reducción en la cantidad de épocas necesarias para alcanzar el mínimo de la función de pérdida en la componente asociada a la ecuación de gobierno. En cuanto a la pérdida por condiciones de borde, no se observaron diferencias significativas al modificar la configuración de la red.

4.1.3 Utilizando una grilla de 20x20

Se repitió el experimento de búsqueda de la solución mediante diferencias finitas y por solución exacta analítica de *u*, en la Figura 7 se muestran los resultados.

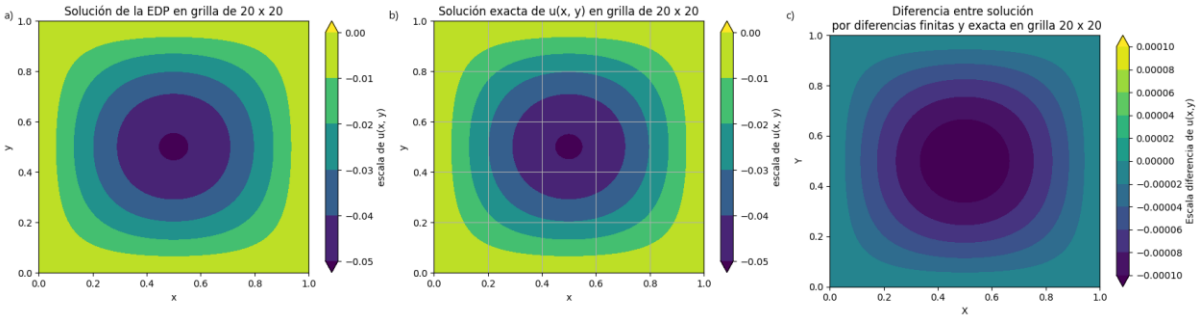


***Fig. 7**: Campos de de solución de u en grilla de 20x20 para: a) utilizando diferencias finitas, b) utilizando la solución exacta propuesta, c) diferencia entre soluciones.*

Al aumentar la resolución espacial (aumentar la cantidad de puntos de grilla del dominio) se logró representar mejor la solución *u(x,y)* al punto tal que se pudo representar de mejor forma la suavidad característica de la sinusoide. Nuevamente, al aumentar la resolución espacial de la grilla se logró disminuir la sobrestimación (en intensidad) de u por diferencias finitas que se vió en casos previos.

También se generaron los campos de solución con PINN en las tres configuraciones propuestas, y se las comparó con la solución exacta.

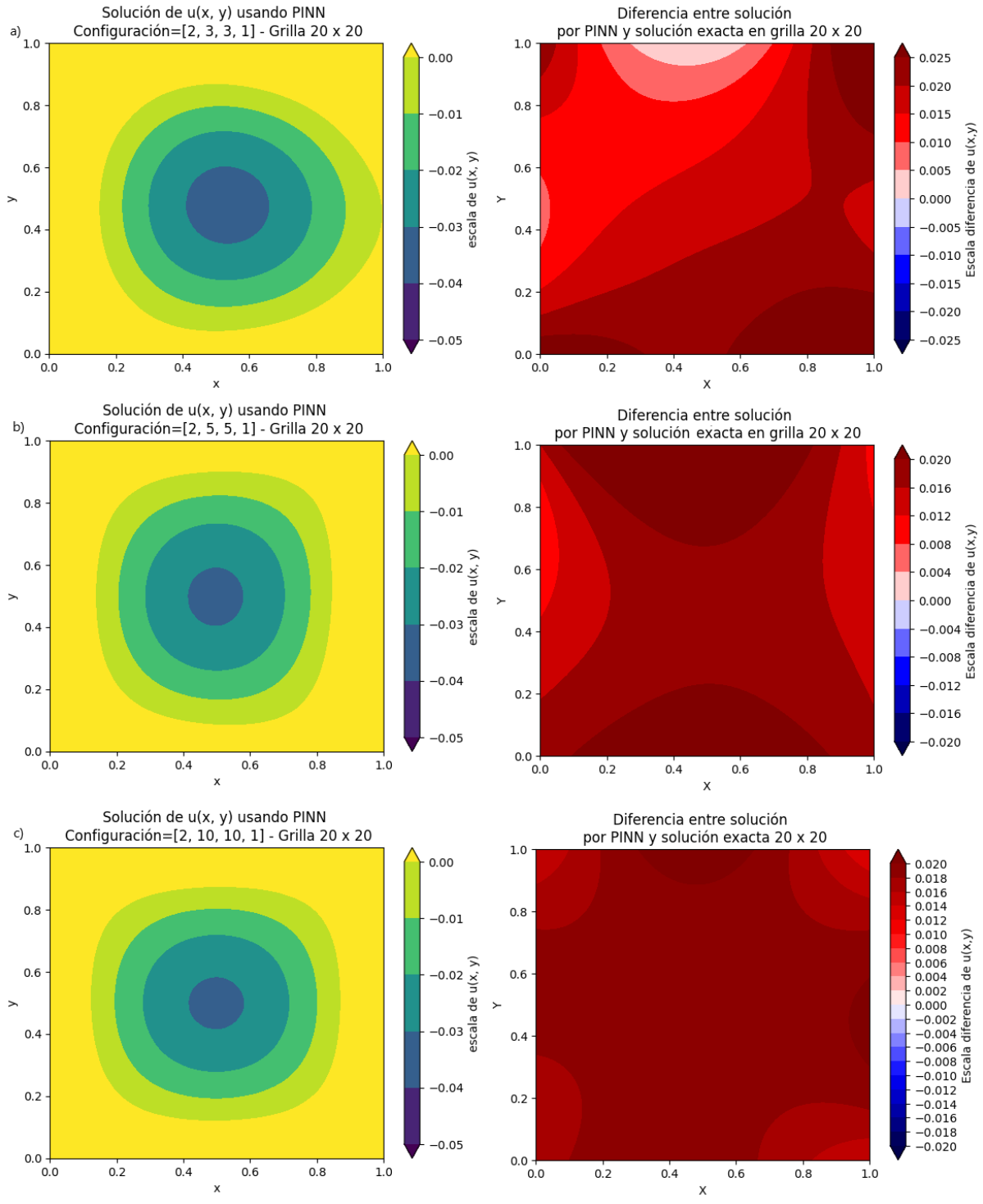


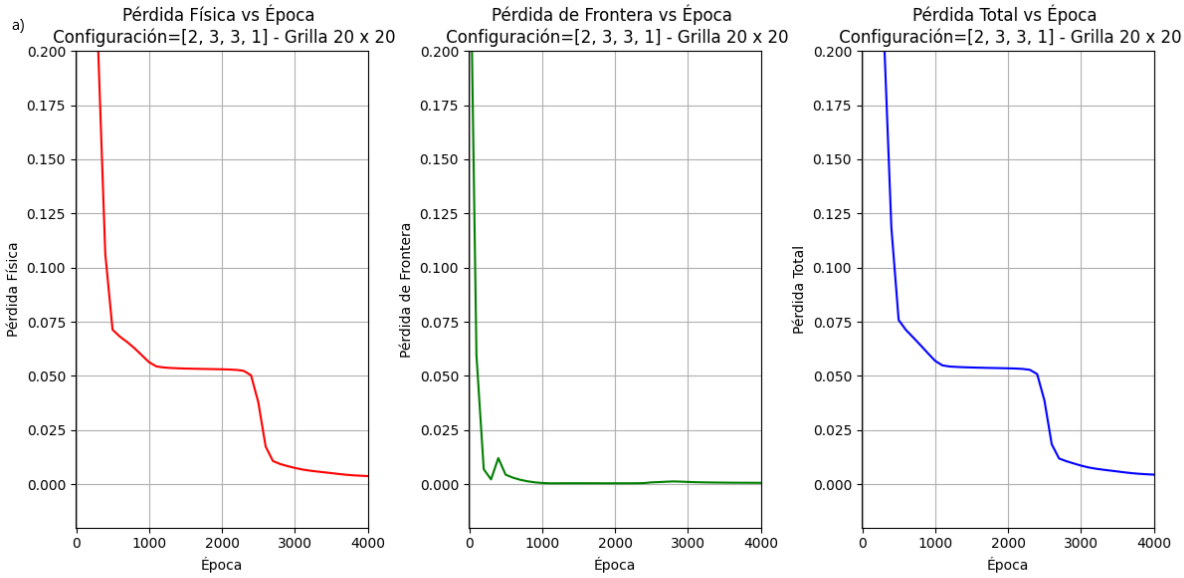
***Fig. 8**: Campos de solución de u(x,y) por PINN, de solución exacta y de diferencia entre estos en grilla de 20x20 para configuración PINN de: a) [2, 3, 3, 1], b) [2, 5, 5, 1] y c) [2, 10, 10, 1].*

Al igual que en los experimentos anteriores, la morfología de la solución obtenida se mantuvo similar a la de la solución exacta, logrando en este caso una mejor adaptación de la red a las condiciones de

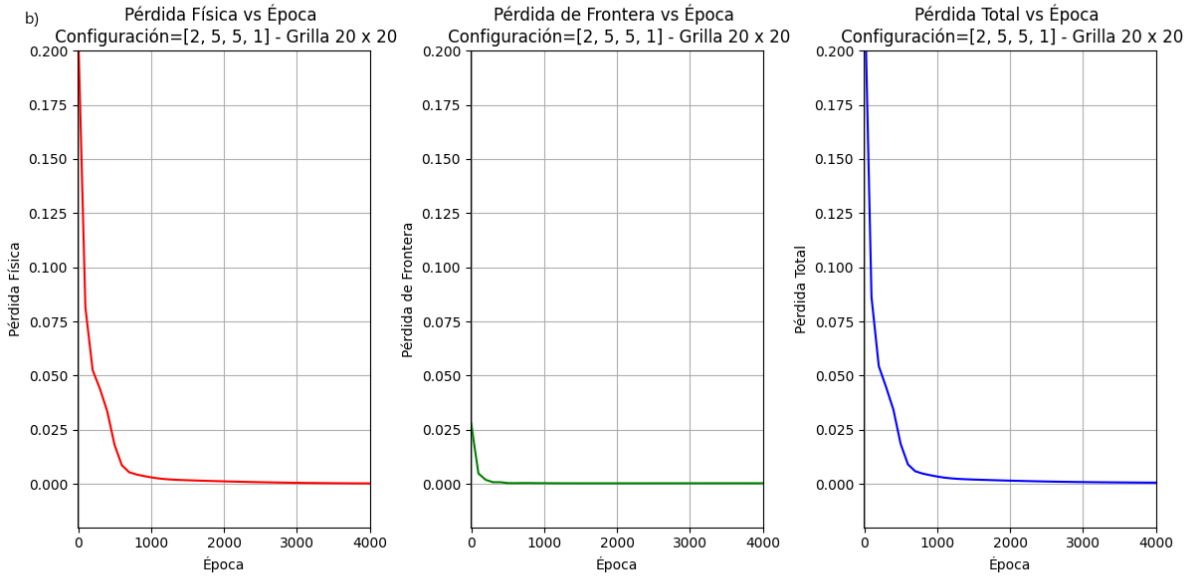
borde, incluso con tres neuronas por capa oculta. Además, la subestimación en la intensidad de la solución permaneció homogénea en los casos con mayor cantidad de conexiones.

Por último, las funciones de pérdida por componente y configuración mostraron los siguientes resultados.

a)


Pérdida Física vs Época
Configuración=[2, 3, 3, 1] - Grilla 20 x 20
Pérdida Física
Época
Pérdida de Frontera vs Época
Configuración=[2, 3, 3, 1] - Grilla 20 x 20
Pérdida de Frontera
Época
Pérdida Total vs Época
Configuración=[2, 3, 3, 1] - Grilla 20 x 20
Pérdida Total
Época

b)


Pérdida Física vs Época
Configuración=[2, 5, 5, 1] - Grilla 20 x 20
Pérdida Física
Época
Pérdida de Frontera vs Época
Configuración=[2, 5, 5, 1] - Grilla 20 x 20
Pérdida de Frontera
Época
Pérdida Total vs Época
Configuración=[2, 5, 5, 1] - Grilla 20 x 20
Pérdida Total
Época

c)

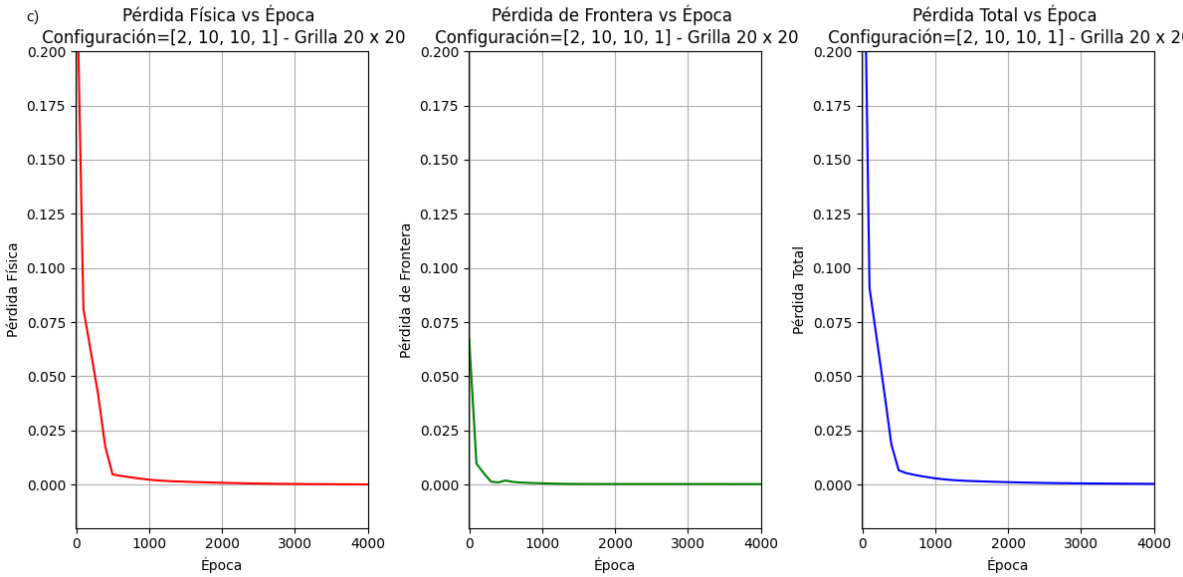

Pérdida Física vs Época
Configuración=[2, 10, 10, 1] - Grilla 20 x 20
Pérdida Física
Época
Pérdida de Frontera vs Época
Configuración=[2, 10, 10, 1] - Grilla 20 x 20
Pérdida de Frontera
Época
Pérdida Total vs Época
Configuración=[2, 10, 10, 1] - Grilla 20 x 20
Pérdida Total
Época

***Fig. 9.**: Curvas de pérdida total, ecuación de gobierno y de condiciones de borde en grilla de 20x20 para diferentes configuraciones de PINN: a) [2, 3, 3, 1], b) [2, 5, 5, 1] y c) [2, 10, 10, 1].*

Al igual que en los casos anteriores, se observó una disminución en la cantidad de épocas necesarias para minimizar la pérdida asociada a la ecuación de gobierno al aumentar el número de neuronas en las capas ocultas, aunque este patrón no fue tan claro al comparar las configuraciones de cinco y diez neuronas por capa oculta. Además, en el caso de tres neuronas por capa oculta, se detectó un cambio de tendencia, lo que podría haberse debido a un mínimo local del cual el optimizador logró escapar. Por otro lado, la pérdida asociada a las condiciones de borde no presentó variaciones significativas entre las diferentes configuraciones.

4.1.4 Utilizando arquiteturas alternativas.

Con intención de resolver la sobrestimación dada por PINN con respecto a la solución exacta vista en la sección anterior, se decidió implementar una arquitectura con una configuración más profunda dada por [2, 10, 10, 10, 10, 1]. Además, a esta arquitectura, se le aplicaron técnicas de mejora como la implementación estricta de la condición de borde y adimensionalización de la EDP. Los resultados se muestran a continuación.

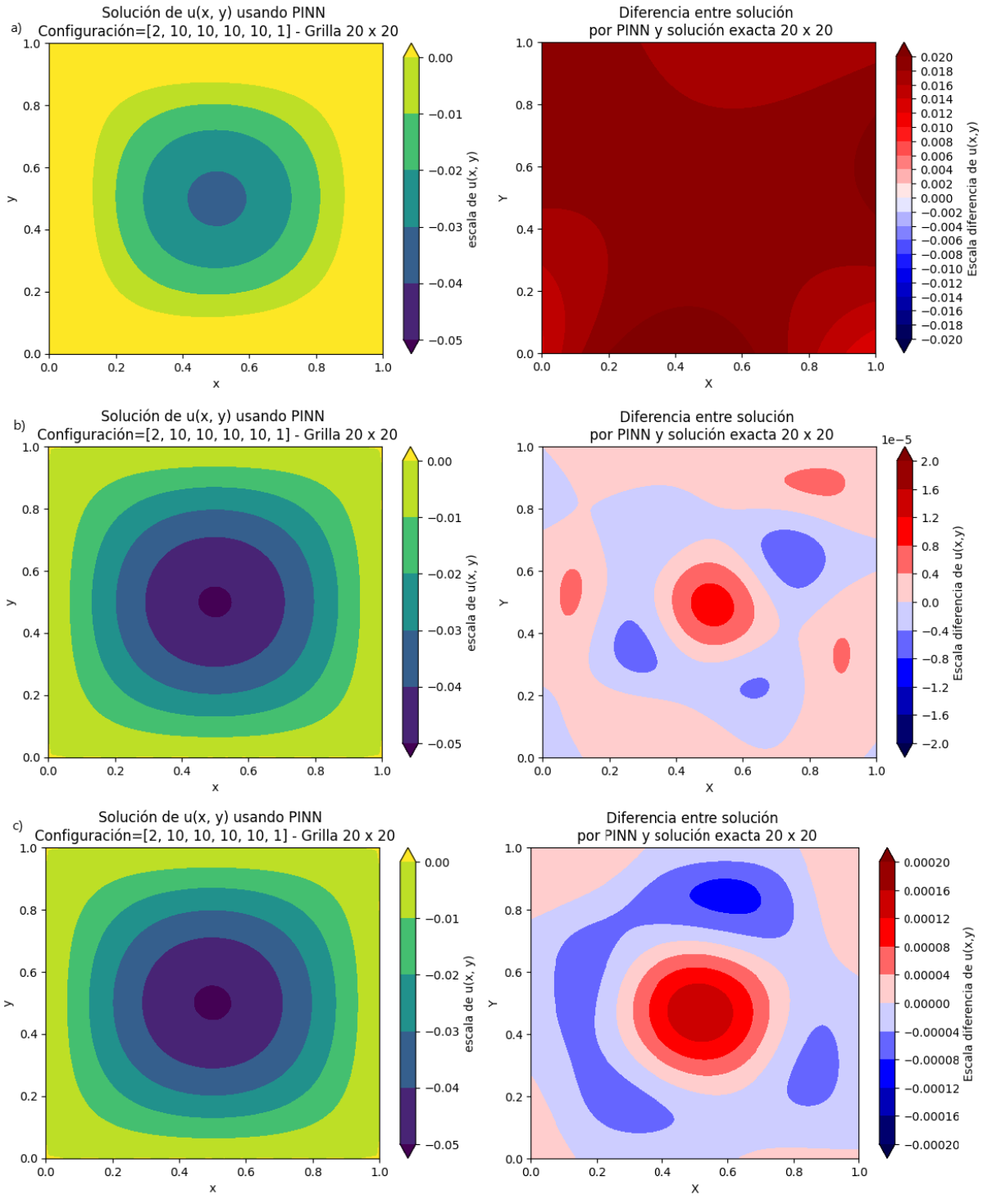


***Fig. 10**: Campos de solución de u(x,y) por PINN, y de diferencia con solución exacta en grilla de 20x20 para configuración PINN de [2, 10, 10, 10, 10, 10, 1]: a) vanilla, b) implementación estricta de condición de borde, y c) implementación estricta de condición de borde y adimensionalización de la EDP.*

Se observó que el simple aumento en la profundidad de la red no resolvió el problema. Sin embargo, al incorporar la implementación estricta de la condición de borde, se logró aumentar la intensidad de la solución *u* lo suficiente como para disminuir la sobrestimación en tres órdenes de magnitud. En

este escenario, el campo de diferencias mostró un comportamiento particular, destacando una sobrestimación en la intensidad de *u* en el centro del dominio. Por otro lado, al introducir la adimensionalización de la ecuación en derivadas parciales (EDP), se observó un aumento de un órden de magnitud en la sobrestimación en el centro del dominio con respecto al caso solo con la implementación estricta. No obstante, es importante tener en cuenta que la solución obtenida mediante PINN presenta una naturaleza aleatoria, debido al optimizador utilizado en su implementación.

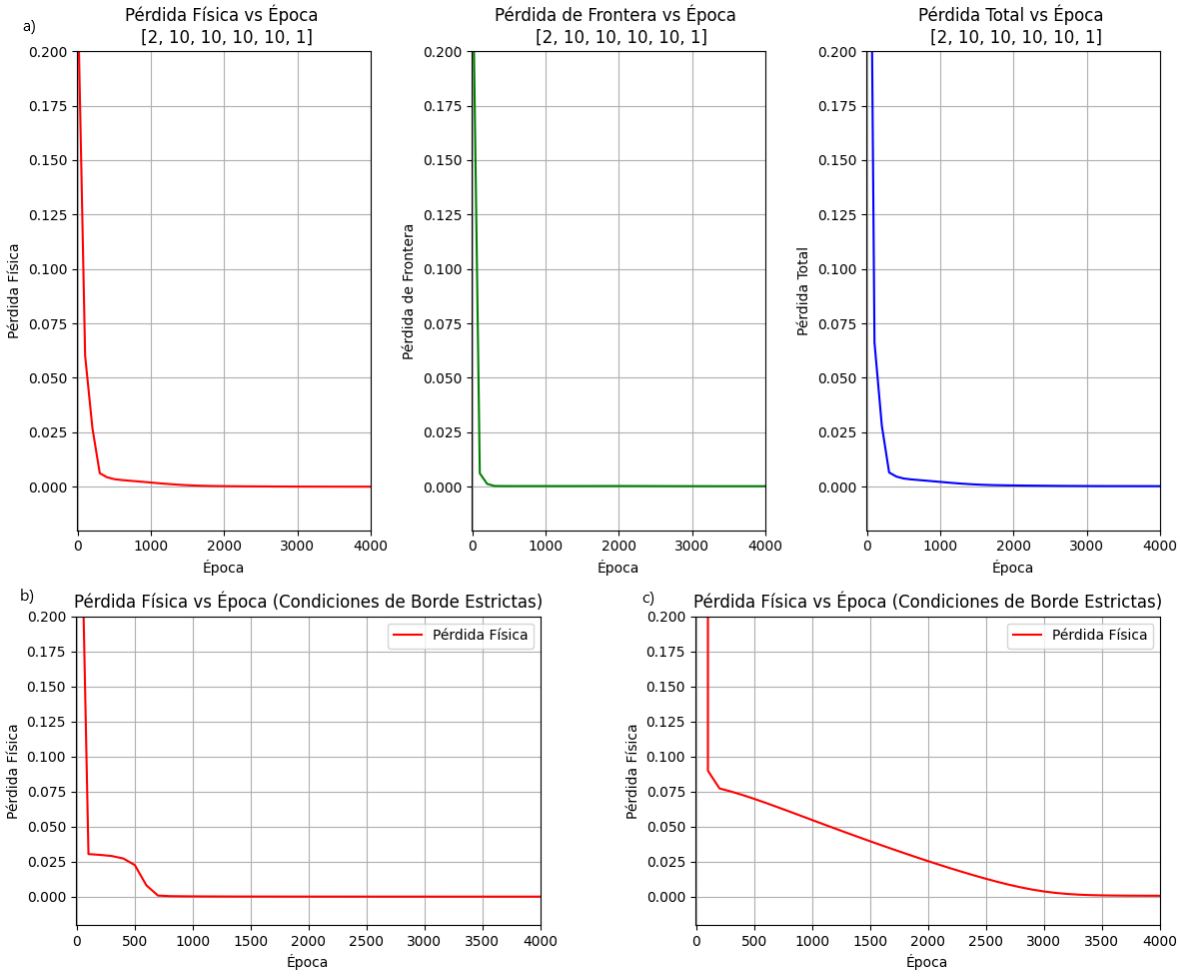


***Fig. 11**: Curvas de pérdida para configuración PINN de [2, 10, 10, 10, 10, 10, 1] en grilla de 20x20: a) vainilla, b) implementación estricta de condición de borde, y c) implementación estricta de condición de borde y adimensionalización de la EDP.*

En el caso sin la implementación estricta ni la adimensionalización, la disminución de la pérdida fue rápida al principio (aproximadamente primeras 400 épocas), luego más lento hasta la época 1200, luego de estos comportamientos se estabilizó. Por otro lado, la cantidad de épocas asociada a las condiciones de borde para converger a un mínimo de pérdida fue significativamente menor, lo que sugiere que las condiciones de frontera fueron fácilmente satisfechas. En el caso en el que se implementaron estrictamente las condiciones de borde, la convergencia fue más rápida en comparación con el caso "vanilla", lo que indica que el cumplimiento estricto de estas condiciones mejora la eficiencia del entrenamiento. Al incorporar la EDP adimensionalizada, la pérdida disminuyó

a una tasa constante pero demandó muchas épocas en alcanzar el mínimo de pérdida (entre 3000 y 3500) en comparación con las técnicas previas.

4.2 Soluciones de la ecuación de conducción de calor con término fuente no lineal

4.2.1 Utilizando una grilla de 5x5

Utilizando la metodología definida en la sección 3, se procedió a generar la solución mediante PINN con la configuración con cinco neuronas por capa oculta y por diferencias finitas en grilla de 5 x5 en ambos casos.

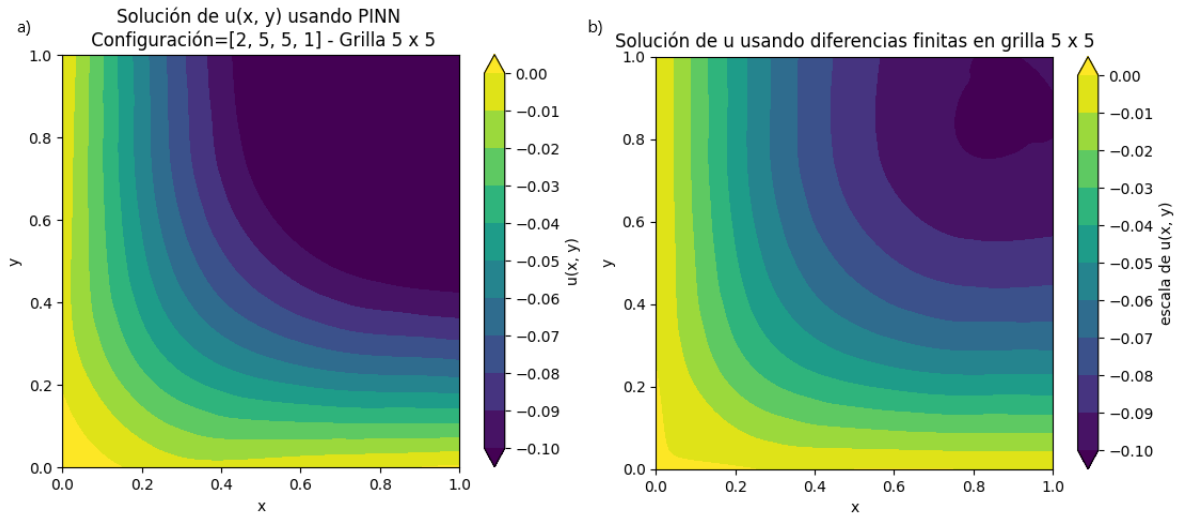


***Fig. 12**: Campos de solución de u(x,y) en una grilla de 5x5 con: a) PINN, y b) diferencias finitas.*

Ante todo se pudo observar la misma estructura de la solución *u* con ambos métodos. Esto es, un mínimo absoluto en $x = y = 1$. Además, en ambos casos se respetan las condiciones de borde. A partir de la configuración de las paletas, la magnitud de la solución en las gráficas resulta comparable. Con esto, se vio que en el caso de PINN, la extensión espacial del mínimo es mayor, resolviendo con una intensidad mayor el gradiente en el resto del dominio.

También se generaron las curvas de aprendizaje para cada componente.

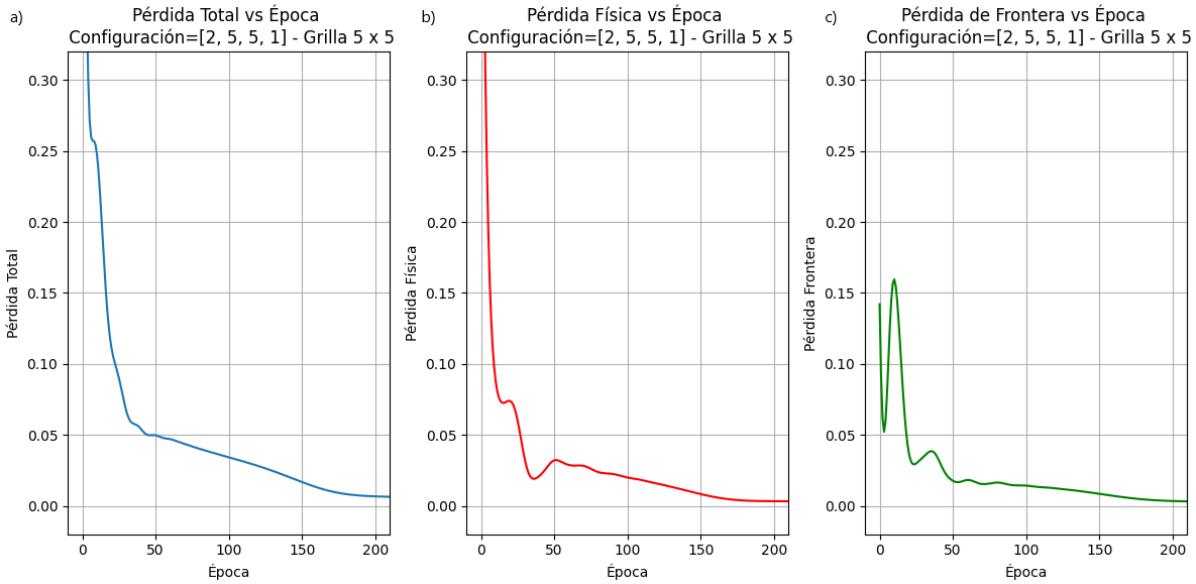


***Fig. 13.**: Curvas de pérdida total (a), de función de gobierno (b) y de condiciones de borde (c) en grilla de 5x5 para la configuración de PINN [2, 5, 5, 1].*

El mínimo de la pérdida total se alcanzó en las primeras 200 (de 2000) épocas, con la mayor tasa de disminución observada durante las primeras cincuenta. El incremento en la pérdida asociada a las condiciones de borde en las primeras épocas es posible que se debiera a que el optimizador escapó de un mínimo local antes de converger al mínimo alcanzado en épocas posteriores.

4.2.2 Utilizando una grilla de 10x10

Al analizar los resultados obtenidos mediante PINN y el método de diferencias finitas, se observó nuevamente que la estructura de la solución se conserva en ambos casos. Aunque, al igual que en el caso anterior, la extensión del mínimo es mayor en la solución obtenida con PINN, la diferencia entre ambos métodos resultó ser menor.

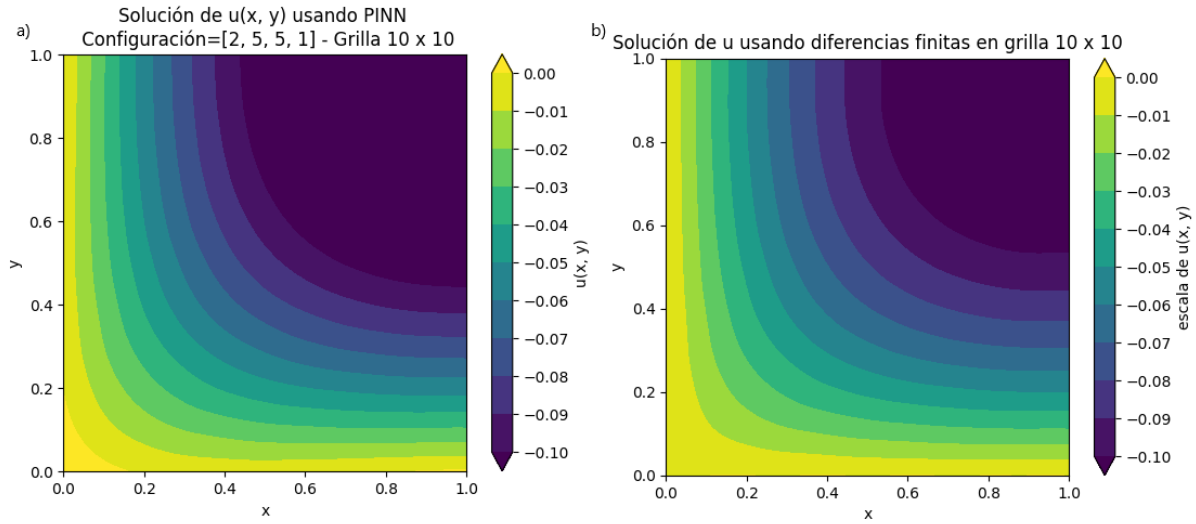


***Fig. 14**: Campos de solución de u(x,y) en una grilla de 10x10 con: a) PINN, y b) diferencias finitas.*

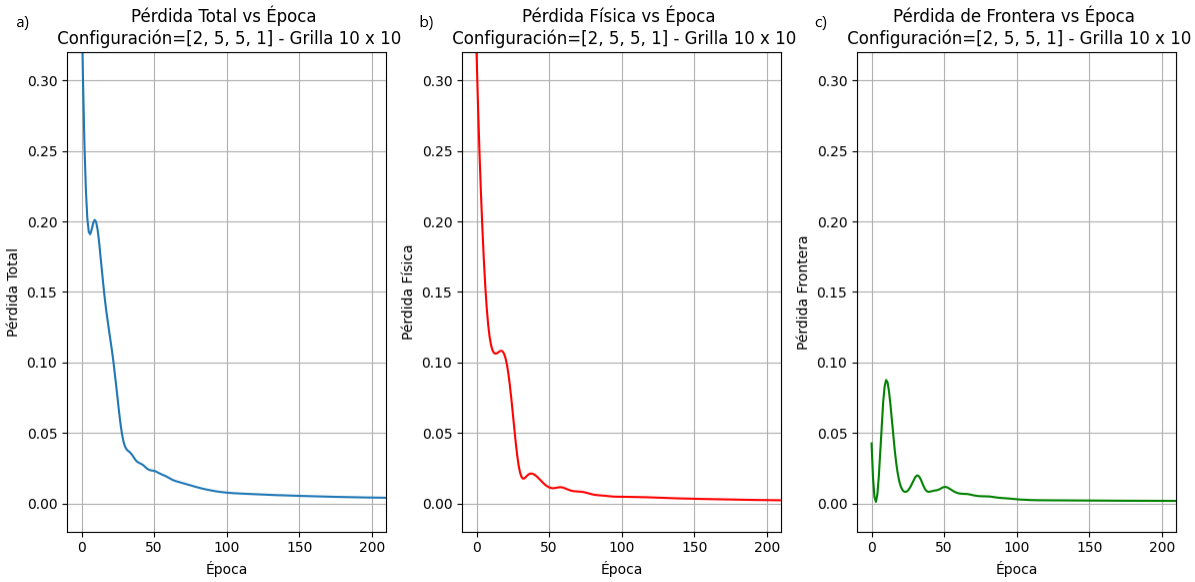


***Fig. 15.**: Curvas de pérdida total (a), ecuación de gobierno (b) y de condiciones de borde (c) en grilla de 10x10 para la configuración de PINN [2, 5, 5, 1].*

El comportamiento de las curvas de aprendizaje no mostró variaciones con respecto al caso de grilla de 5x5, si bien la pérdida total converge al mínimo en las primeras 100 (de 2000) épocas . Nuevamente se apreció el aumento inicial de la pérdida en las condiciones de borde, para luego converger a un mínimo.

4.2.3 Utilizando una grilla de 20x20

Una vez más las soluciones por ambos métodos presentaron la misma estructura, si bien se conservó que el mínimo por PINN fuese un poco más extenso que el caso por diferencias finitas.

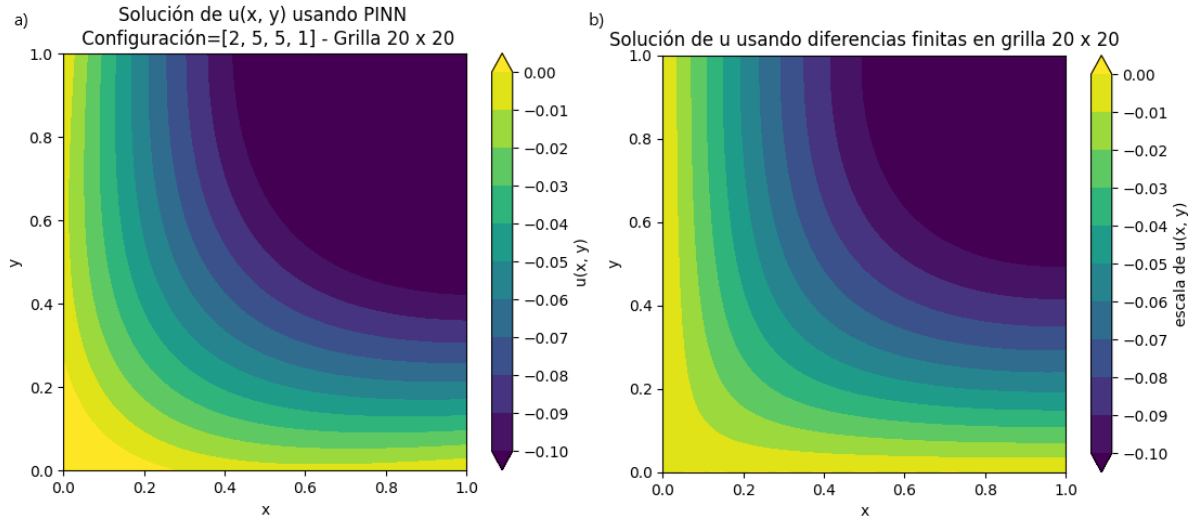


***Fig. 16**: Campos de solución de u(x,y) en una grilla de 20x20 con: a) PINN, y b) diferencias finitas.*

Sin mayores cambios, las curvas de pérdida mostraron el mismo comportamiento que en casos previos, alcanzando el mínimo en pocas épocas.

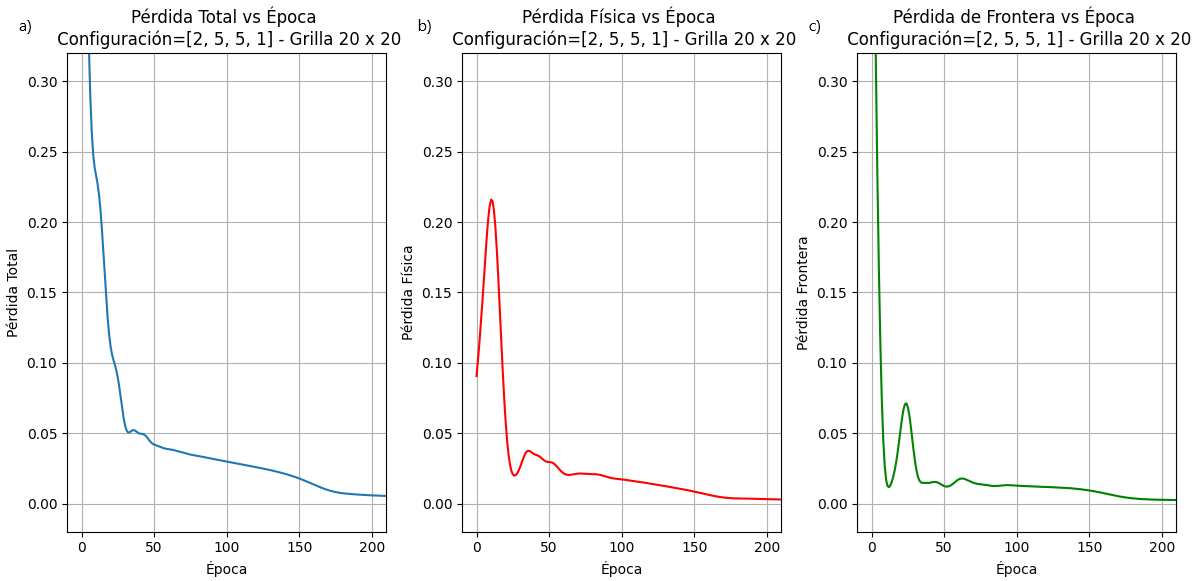
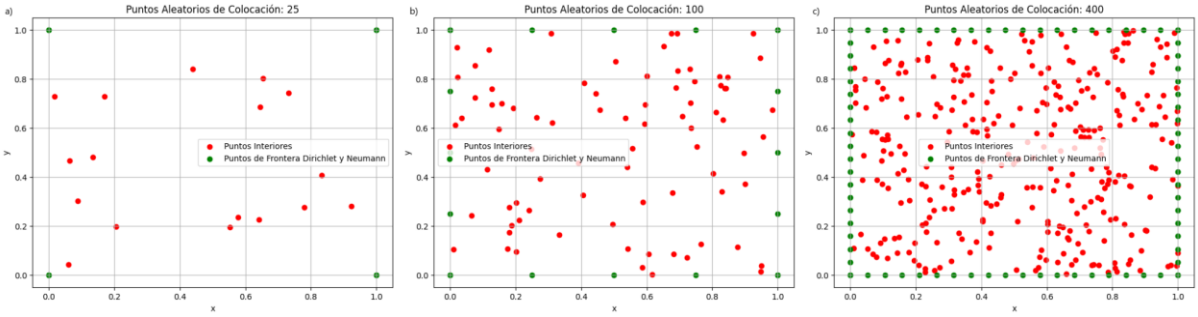


***Fig. 17.**: Curvas de pérdida total (a), ecuación de gobierno (b) y de condiciones de borde (c) en grilla de 20x20 para la configuración de PINN [2, 5, 5, 1].*

4.2.4 Utilizando una muestra aleatoria de puntos de colocación

Para este experimento se utilizó nuevamente una configuración [2, 5, 5, 1] ubicando la misma cantidad de puntos de colocación que las grillas de 5x5, 10x10 y 20x20 pero de manera uniforme y aleatoria. Destinando un 20% de los datos a la frontera en los casos de 400 y puntos de colocación, y

un 32% en el caso de 25 puntos. En la siguiente figura se muestran la distribución de puntos en cada caso.



***Fi. 18**: Distribución de puntos de colocación para: a) 25 puntos, b) 100 puntos y c) 400 puntos. En rojo se marcan los puntos internos del dominio, mientras que en verde los puntos de frontera.*

En el caso de 25 puntos aleatorios, se destinó 8 puntos (32%) a la frontera, con dos puntos por borde. Luego, en el caso de 100 y 400 puntos aleatorios, se destinó el 20% de los mismos a la frontera (20 y 80 respectivamente). En el primer caso (Figura 18a), fue evidente la gran cantidad de espacios sin representación, lo que en principio no pareció ser una buena resolución para hallar la solución. En el segundo caso (Figura 18b), si bien hubo una mejora en la resolución espacial, aún se pudieron identificar regiones con ausencia de puntos. Sin ir más lejos, el último caso (Figura 18c), fue la que mejor distribución de puntos presentó, tanto en el interior del dominio como en la frontera.

Definidos estos dominios, se calculó la solución mediante PINN con la configuración [2, 5, 5, 1], los resultados se aprecian en la siguiente figura.

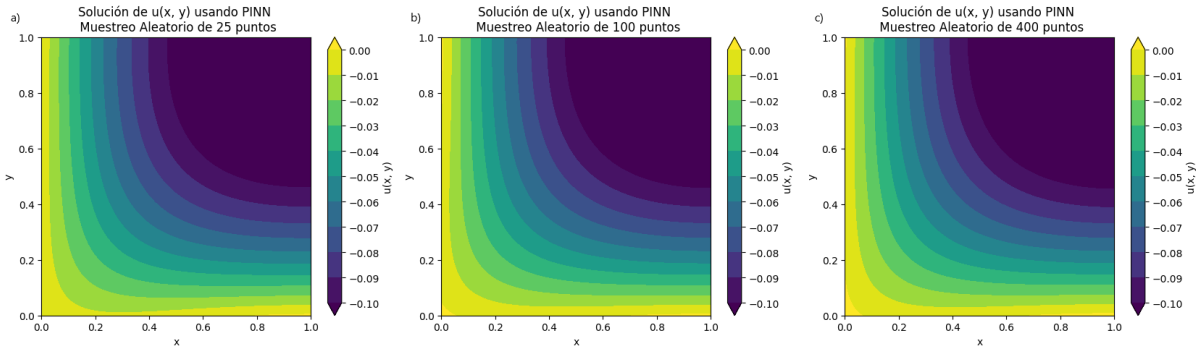


***Fig. 19**: Soluciones de campo u mediante PINN para los casos: a) 25 puntos, b) 100 puntos, y c) 400 puntos.*

Al emplear un esquema *vanilla*, los resultados mostraron una morfología consistente con la solución obtenida mediante diferencias finitas y utilizando PINNs con mallas de puntos equiespaciados. Las PINNs propuestas lograron aprender correctamente tanto las condiciones de borde como los valores en los puntos internos, sin presentar diferencias significativas al variar la cantidad de puntos aleatorios de colocación utilizados.

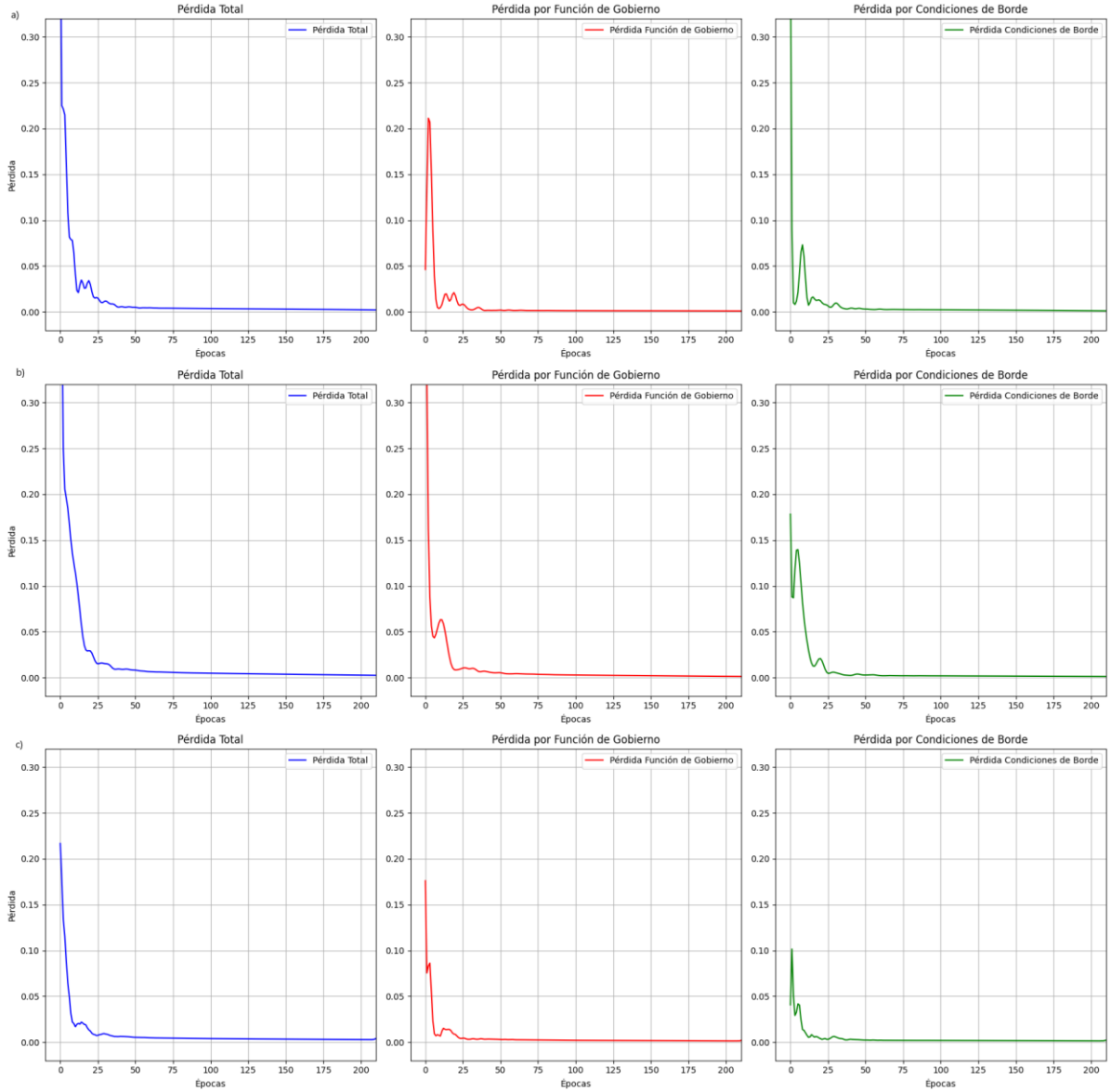
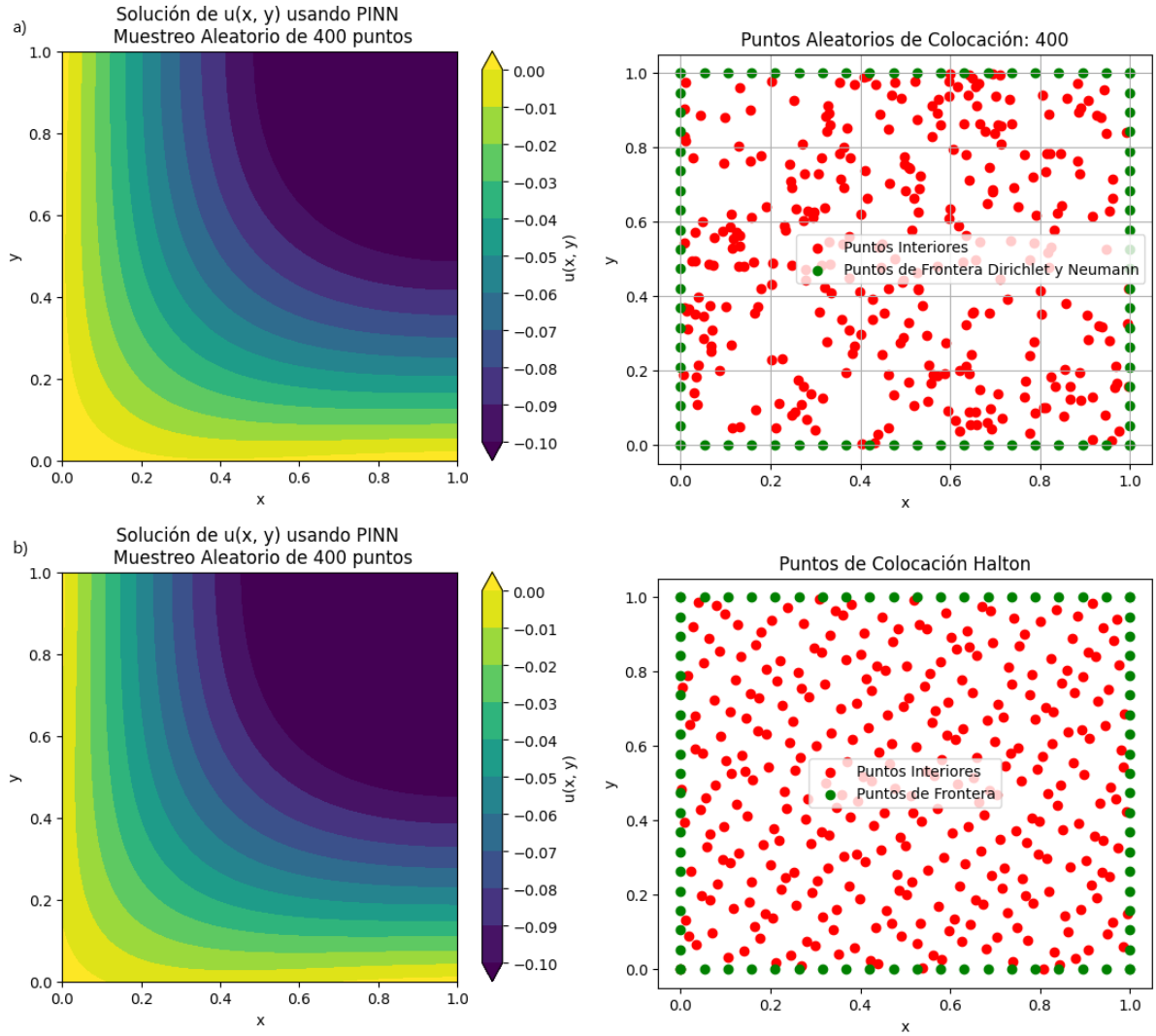


***Fig. 20**: Curvas de pérdida total (azul), función de gobierno (rojo) y de condiciones de borde (verde) para la configuración de PINN [2, 5, 5, 1] en diferentes muestras de puntos aleatorios de colocación: a) 25, b) 100 y c) 400.*

Las PINNs definidas emplean el optimizador estocástico Adam, lo que puede generar variaciones en los resultados entre distintas ejecuciones del modelo. Sin embargo, como se observa en los resultados presentados en la Figura 20, los modelos tienden a aprender de manera consistente durante las primeras 20 a 30 épocas. Además, en todos los casos analizados, la pérdida asociada a la función de gobierno requirió menos épocas para converger en comparación con la pérdida correspondiente a las condiciones de borde.

4.2.5 Utilizando técnicas de mejoras sobre arquitecturas adicionales.

Para evaluar otras arquitecturas, se propuso una red más profunda con la configuración [2, 10, 10, 10, 10, 1]. Por un lado con el esquema de puntos de colocación aleatorios sobre el dominio interior al igual que casos previos, y por otro lado se utilizó la distribución de Halton. Los resultados se muestran a continuación.



***Fg. 21**: Solución de EDP utilizando PINN con configuración [2,10,10,10,10,1] y su distribución de puntos de colocación interiores para: a) distribución aleatoria, b) distribución de Halton.*

No se observaron diferencias significativas en la solución al utilizar colocación de puntos de manera aleatoria o mediante el método de Halton, aunque las distribuciones de puntos en el dominio interior fueron diferentes. Esto sugiere que esta técnica no produjo una mejora considerable para el problema en cuestión.

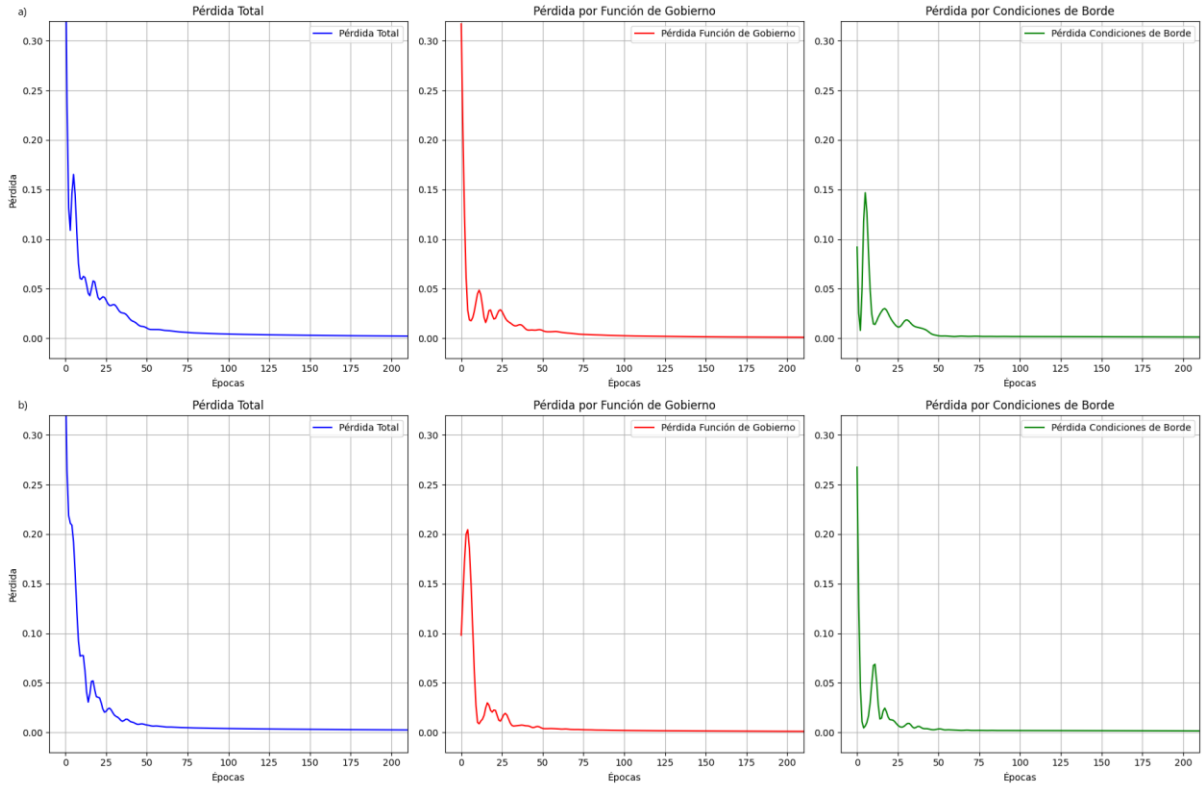


***Fig. 22**: Curvas de pérdida total (azul), función de gobierno (rojo) y de condiciones de frontera (verde) utilizando PINN con configuración [2,10,10,10,10,1] considerando distribución de puntos de colocación interiores para: a) distribución aleatoria, b) distribución de Halton.*

Al igual que el análisis de la figura anterior, no se identificaron diferencias significativas en las curvas de pérdida. En ambos casos, se logró la convergencia al mínimo entre las épocas 25 y 50. No obstante, se apreció cierta oscilación en la pérdida de la condición de borde hasta que logró alcanzar el mínimo en ambas distribuciones de puntos.

5. CONCLUSIONES

Relativo a la ecuación sinusoidal

El método de diferencias finitas ha mostrado buenos resultados, proporcionando una morfología similar a la de la solución exacta, con un mínimo en el centro del dominio. Sin embargo, se observó que la solución obtenida mediante diferencias finitas tiende a sobrestimar la intensidad en comparación con la solución exacta, especialmente en el centro del dominio. Esta sobrestimación disminuye y se homogeniza espacialmente a medida que aumenta el tamaño de la grilla.

Las soluciones obtenidas mediante PINN también presentaron una morfología similar a la de la solución exacta. A diferencia de lo observado en el método de diferencias finitas, la solución obtenida con PINN tiende a subestimar la intensidad con respecto a la solución exacta. Esta subestimación parece ser independiente de la resolución espacial, mostrando un campo de diferencias homogéneo a lo largo del dominio.

Es importante resaltar que los resultados obtenidos mediante PINN en este trabajo incluyen una componente aleatoria debido al uso de optimizadores estocásticos en su arquitectura, lo que puede influir en la precisión de las soluciones.

Finalmente, al probar arquitecturas alternativas, se observó que el aumento de la profundidad de la red no generó mejoras significativas por sí solo. No obstante, los cambios positivos fueron evidentes cuando se incorporaron técnicas de mejora, como la implementación estricta de las condiciones de frontera y la adimensionalización de la ecuación diferencial parcial (EDP). En estos casos, se logró reducir la sobrestimación (en magnitud) de la solución u obtenida por diferencias finitas en comparación con la solución exacta.

Relativo a la ecuación exponencial

Independientemente del tamaño de la grilla empleada, las soluciones obtenidas mediante diferencias finitas y PINN presentaron una morfología similar, con un mínimo absoluto en la esquina superior derecha del dominio (x=y=1). En todos los casos, las condiciones de frontera de Dirichlet y Neumann se ajustaron adecuadamente. Sin embargo, en las soluciones obtenidas mediante PINN, la extensión del mínimo fue mayor. Esta diferencia, no obstante, tiende a disminuir a medida que se incrementa el tamaño de la grilla.

Al emplear puntos de colocación con distribuciones diferentes a la grilla uniforme, no se observaron diferencias significativas al aumentar la profundidad y el ancho de la red neuronal (PINN), ni al incrementar la cantidad de puntos de colocación o modificar su distribución dentro del dominio.

6. REFERENCIAS

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[2] Aniruddha Bora and Weizhong Dai. *Gradient preserved method for solving heat conduction equation with variable coefficients in double layers*. *Applied Mathematics and Computation*, 386:125516, 2020.

[3] MWM Gamini Dissanayake and Nhan Phan-Thien. *Neural-network-based approximations for solving partial differential equations*. *Communications in Numerical Methods in Engineering*, 10(3):195–201, 1994.

7. ANEXO

Abreviaturas

PINN: Physics-Informed Neural Networks (Redes Neuronales Informadas por Física).

EDP: Ecuaciones Diferenciales Parciales.