Alan Fernando Bravo Pimentel - A01645691 / Miguel Angel Rivas Torres - A0164462 /
Edgar Andrey Balvaneda - A01644770

PSO-PINN: PHYSICS-INFORMED NEURAL NETWORKS TRAINED WITH PARTICLE SWARM OPTIMIZATION

Introducción

El artículo discute un problema que es de gran importancia en el entrenamiento de Redes Neuronales Informadas por la Física. Las PINNs, usualmente se entrenan con técnicas basadas en el descenso de gradiente con el propósito de disminuir una función de perdida que engloba datos y también las ecuaciones diferenciales parciales conocidas como PDEs que rigen el sistema, No obstante, este proceso, el de entreno mediante el descenso de gradiente, a menudo muestra "patologías" y algunas complicaciones con la convergencia, más que nada si las soluciones de las PDEs presentan características irregulares o de cambio rápido, la dinámica del flujo del gradiente, podría llegar a ser "rígida" dificultando al optimizador la tarea de hallar la solución correcta.

Los Investigadores del artículo proponen PSO-PINN, la cual es una metodología innovadora para entrenar PINNs, utilizando la Optimización por Enjambre de Partículas, PSO. En lugar de una simple red neuronal, afinada por gradiente, PSO-PINN utiliza un enjambre de redes y partículas, explorando de manera conjunta el espacio de soluciones, los pesos de la red.

La clave radica en una versión híbrida llamada PSO-BP-CD, PSO con Retropropagación y Decaimiento de Coeficientes. Este enfoque combina la búsqueda global de PSO con la optimización local eficiente del descenso por gradiente. Durante el entrenamiento, el algoritmo reduce la influencia del enjambre, el comportamiento social, y aumenta el peso del descenso por gradiente, permitiendo al enjambre "llegar" a un óptimo local muy prometedor.

Esta propuesta es de gran importancia por dos argumentos principales:

- **Solidez**: Debido a que llega a mitigar los problemas de convergencia del descenso del gradiente en las PINNs, accediendo a resolver problemas de PDE de alta complejidad donde los métodos tradicionales suelen fallar.
- **Cuantificación de Incertidumbre**: A causa de ser un método basado en un ensamble. PSO-PINN brinda de forma natural una aproximación de la incertidumbre. La variación entre las predicciones de las diferentes partículas del enjambre indica la seguridad del modelo sobre su propia solución.

El artículo fue seleccionado porque aborda un problema práctico y relevante que enfrentamos en el campo del “Machine Learning”. Aunque las PINNs son una herramienta poderosa, sus limitaciones en el entrenamiento son bien conocidas. Este artículo ofrece una solución que no solo mejora la precisión a un problema de

optimización, sino que añade la característica crucial de la cuantificación de la incertidumbre al combinar sensiblemente el aprendizaje profundo con una parábola de optimización clásica.

En este proyecto se pretende reproducir y validar los resultados del artículo original en el nuevo repositorio, así como ampliar su alcance incorporando un nuevo modelo adicional. Para esto se realizarán modificaciones significativas en los hiperparámetros como lo son el tamaño de la población, el número de iteraciones y la arquitectura de la red, para así analizar la sensibilidad del algoritmo e investigar si dichas variaciones producen un desempeño más eficiente, estable y preciso que el reportado originalmente.

Marco Teórico

La optimización por Enjambre de Partículas, es un algoritmo muy utilizado en el mundo de la optimización estocástica. Este algoritmo basado en poblaciones está inspirado en el comportamiento social de bandadas de aves o bancos de peces.

El algoritmo funciona de la siguiente manera:

- **Inicialización**: Se crea una "población" de partículas, siendo cada partícula una solución del problema de optimización. En el contexto de PSO-PINN, cada partícula es un vector donde cambian todos los pesos y sesgos de la red neuronal completa.
- **Evaluación**: Se llega a evaluar el “fitness” de cada partícula usando una función objetivo que, en este caso, sería la función de pérdida de la PINN.
- **Actualización de Velocidad**: Cada una de las partículas ajusta la dirección y magnitud de su próximo movimiento dentro del espacio de búsqueda. Esto se basa en tres componentes principales:
    - La tendencia a seguir su dirección actual.
    - Atracción hacia el mejor posicionamiento encontrado por la propia partícula.
    - Atracción hacia el mejor posicionamiento encontrado por cualquier partícula.
- **Posición**: La partícula se mueve a una nueva posición sumando su velocidad actualizada a su posición actual.
- **Iteración**: Este procedimiento se llega a repetir y con el paso del tiempo, el enjambre converge hacia un óptimo global.

El artículo usa una versión híbrida clave, que añade un cuarto componente en la actualización de la velocidad, que es el vector de gradiente, obtenido por retropropagación. Esto ayuda a guiar al enjambre de manera más eficiente,

combinando la exploración global que proporciona PSO con la explotación local del descenso de gradiente.

El problema es resolver Ecuaciones Diferenciales Parciales. Con una PINN el cual son una red neuronal que aproxima la solución desconocida de una PDE. La mayor característica de una PINN es su función de pérdida. En lugar de solo ajustarse a puntos dados, la red también se ve forzada a obedecer las leyes de la física descritas por la PDE. Esto causa que se logre minimizar la pérdida total que es la suma de varios componentes. En dónde la pérdida total es la suma de la Pérdida Residual, Pérdida de Frontera y Pérdida de datos. El objetivo es encontrar los pesos que ayuden a minimizar esta pérdida combinada. Se utilizaron ciertas métricas para poder evaluar el rendimiento del modelo, como lo son el error relativo, mismo que compara la solución predicha por la PINN con la solución analítica exacta, además como fue mencionada anteriormente, el valor final de la pérdida también fue utilizada como un factor importante y por último se tomaron en cuenta la media del enjambre en dónde se grafica la predicción media de todo el enjambre contra la solución exacta y la varianza/desviación estándar del modelo.

Implementación

El proyecto se desarrolló en Python con TensorFlow para la construcción de las PINNs, NumPy para cualquier tipo de cálculo, Matplotlib para la visualización y pyDOE para generar puntos de colocación. Para la organización del código se utilizó de base él repositorio compartido en el repositorio en dónde se tomaron en cuenta el módulo src/swarm instalable y scripts de examples para cada PDE.

Para las decisiones claves nuevamente nos basamos mucho en la base del artículo en dónde en el script teníamos @tf.function para acelerar la ejecución en grafo y tf.vectorized_map para paralelizar el enjambre. La mayor dificultad fue el ajuste del modelo Sine-Gordon en dónde se realizaron +8 intentos a prueba y error, uno de más de una hora por el alto coste computacional. Fue resuelto con la implementación del decaimiento de coeficientes y ajustando la inercia, equilibrando mejor la exploración global de PSO.

Supuestos y Adaptaciones

A pesar de que el artículo estaba bastante completo y contenía información suficiente en cuanto a la arquitectura de la red neuronal así como los hiperparámetros del PSO, el artículo no especifica las semillas aleatorias utilizadas para la inicialización de los pesos de la red neuronal ni para el muestreo de los puntos de colocación, lo que hace imposible una réplica exacta de los resultados. Se asumió que se obtendrían resultados similares mediante cualquier semilla. Asimismo, se asumió que el archivo llamado

AC.pckl1 contiene los datos de alta fidelidad de la solución a la ecuación Allen-Cahn de los cuales hablan los autores y se empleó este archivo para analizar la precisión del algoritmo metaheurístico.

Con el objetivo de replicar los resultados del artículo, se ajustaron los hiperparámetros del código para que coincidieran con los descritos en el artículo en cada uno de los experimentos (Allen-Cahn, Burgers, Poisson, Diffusion y Advenction). En el caso de la ecuación de Allen-Cahn, por ejemplo, se modificaron los parámetros pop_size de 10 a 100, el número de iteraciones de 600 a 2000, el coeficiente de inercia de 0.999 a 0.99, el coeficiente cognitivo c_1 de 0.0008 a 0.08, el coeficiente social de 0.05 a 0.5, la tasa de aprendizaje de Adam de 0.0001 a 0.005 y se añadió el parámetro c_decrease = True para el decaimiento de coeficientes. Además, se modificó la arquitectura de la red neuronal de tener 5 capas de 15 neuronas a tener 5 capas de 8 neuronas. Asimismo, se aumentaron los puntos de frontera N_b de 200 a 400 y N_f (Puntos Residuales/Física): aumentó de 500 a 2000. Modificaciones similares se llevaron a cabo con cada experimento, siempre apegándose a la información provista por el artículo.

Variantes y Mejoras

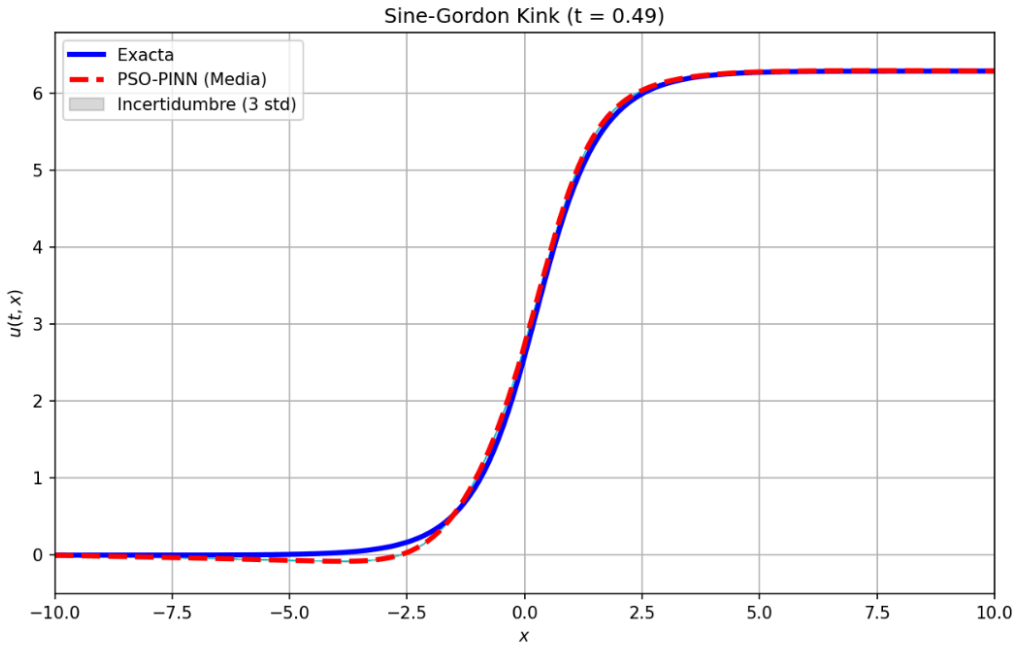
Algunas de las modificaciones que se realizaron al método original, además de los cambios a todos los hiperparámetros que fueron mencionados anteriormente, también se llegó a modificar, la estrategia de ponderación de la función de pérdida en el ejemplo de la ecuación de Burgers. En el método original, se aplicaba una ponderación específica al error de la condición inicial, utilizando el término $1.5 \times \text{mse}_0$. Dicha ponderación fue eliminada y se optó por una suma simple de todos los componentes de error, esta modificación busca otorgar la misma importancia a cada condición.

Para la optimización se implementó el uso de decoradores para simplificar la gestión del cálculo de gradientes. En el repositorio original se había que definir manualmente una función interna “loss_grad()” que tomaba la función de pérdida en cada script. Con los cambios realizados ese proceso se eliminó al implementar el decorador “@grad_decorator” que abstrae automáticamente la lógica de cálculo de gradientes de TensorFlow. Gracias a este cambio, la función de perdida queda mucho más limpia y declarativa,

Además, se hizo la implementación de un nuevo problema de PINN para la ecuación de Sine-Gordon. Sin-Gordon es una ecuación diferencial parcial hiperbólica no lineal que modela fenómenos como lo es la propagación de solitones. Para el experimento se implementó la solución conocida como “Kink”, que es la representación de una transición de fase estable que se propaga a una velocidad constante. Consideramos que este problema es de más complejidad y no lineal que los demás ejemplos vistos en el artículo original.

Se obtuvieron los siguientes resultados:

```
Pérdida Final (Last Loss):  0.000861365
Error L2: 2.202133e-02
Guardando animación...
MovieWriter ffmpeg unavailable; using Pillow instead.
Animación guardada como 'sine_gordon_demo.gif'
```



Los resultados para la ecuación muestran un éxito claro. El Error L2 de 0.022 indica que la solución predicha por la red está muy cerca de la analítica, con solo un 2.2% de desviación, confirmando alta precisión del modelo. La pérdida final de 0.00086 refleja que el optimizador PSO-PINN logró converger satisfaciendo tanto la PDE como las condiciones de frontera. La visualización llega para confirmar todo esto en dónde se puede observar, las líneas de la solución exacta y la predicha son prácticamente idénticas, y la banda de varianza es muy estrecha, mostrando consenso entre las redes y alta confianza en las predicciones.

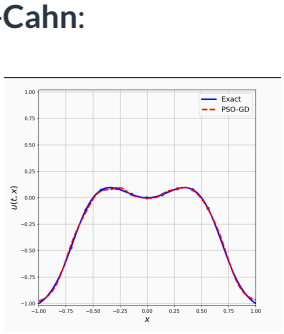
Haciendo la comparación con el repositorio original, podemos concluir que hay dos factores importantes que se han implementado, una de ellas es él usó de decoradores que gracias a esto hay una mejora en el código para abstraer la lógica del gradiente, haciendo la función de pérdida más declarativa y limpia que el método original. La segunda que consideramos la más importante fue la implementación de la ecuación Sine-Gordon, un problema de PDE no lineal más complejos que los del repositorio original.

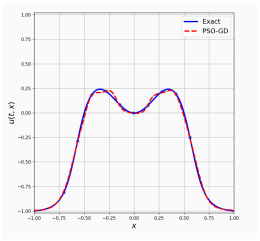
Alan Fernando Bravo Pimentel - A01645691 / Miguel Angel Rivas Torres - A0164462 /
Edgar Andrey Balvaneda - A01644770

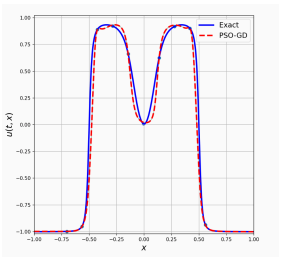
Resultados Experimentales

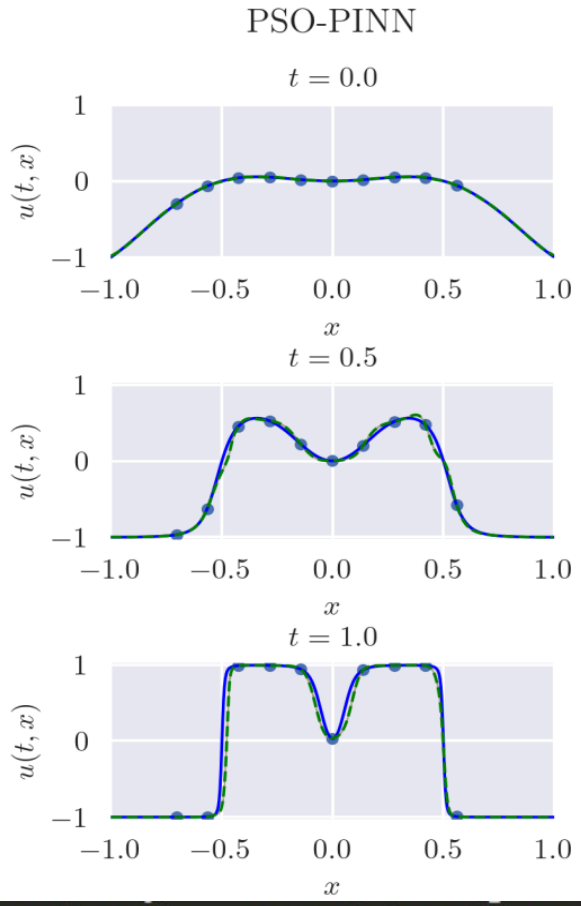
Reproducción izquierda, original derecha.:

Allen-Cahn:









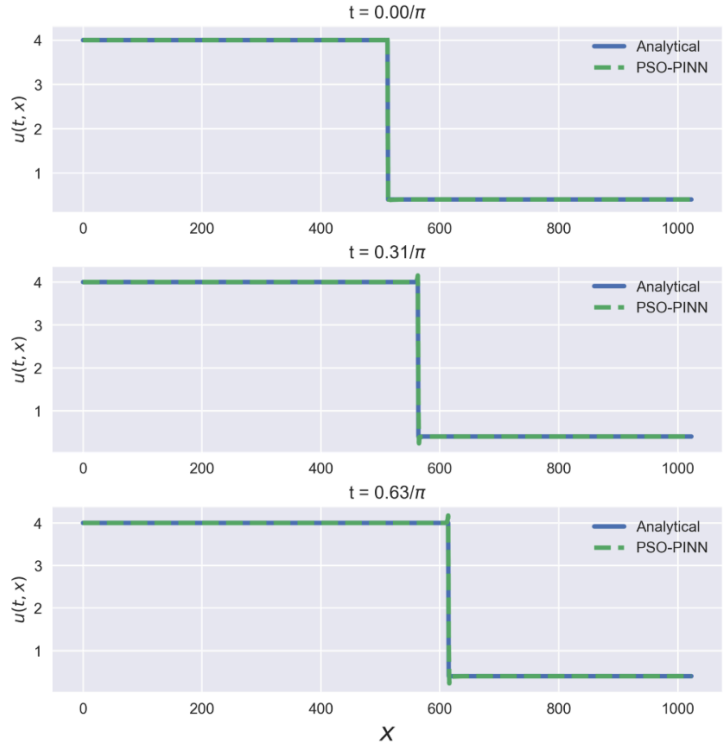

```
[################################] 100%  --  urrent loss: 0.009972968
Time elapsed :  0:12:27
Last Loss:  0.008594073
Error u: 1.269083e-01
```

El error está a aproximadamente una desviación estándar de lo reportado en el artículo (L2 error of 0.093 ± 0.032), lo cual es coherente dadas la situación de aleatoriedad y la falta de semilla.

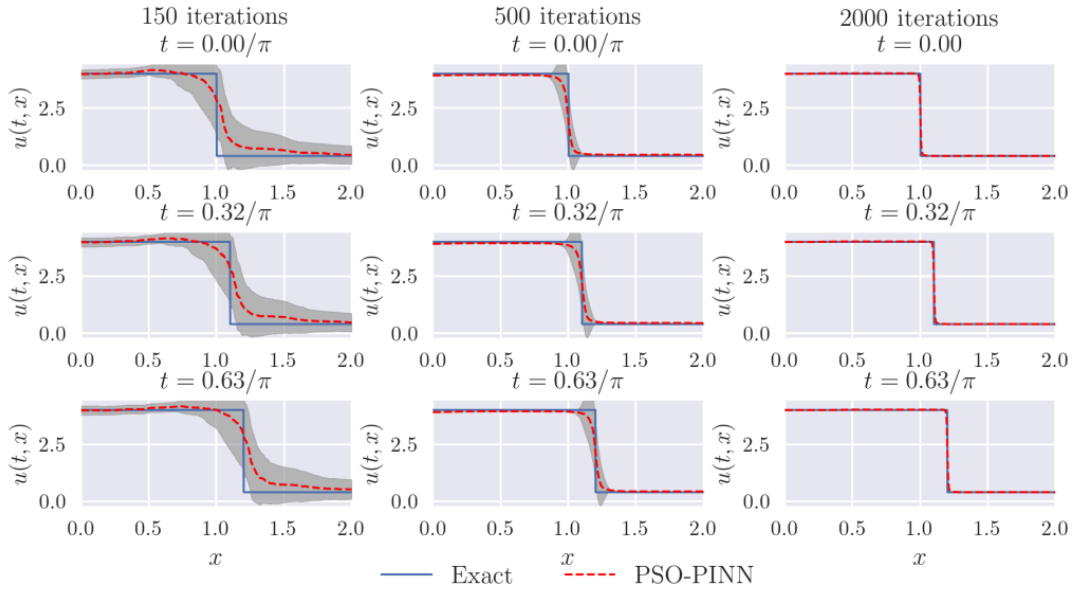
Alan Fernando Bravo Pimentel - A01645691 / Miguel Angel Rivas Torres - A0164462 /
Edgar Andrey Balvaneda - A01644770

Advection:

Réplica con 2000 iteraciones:



Artículo:



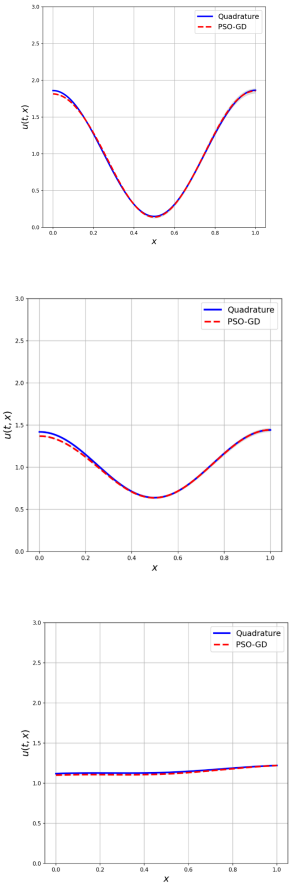

```
[##############################] 100%  --  urrent loss: 3.1905346
L2 error PSO-GD: 2.17e-02
```

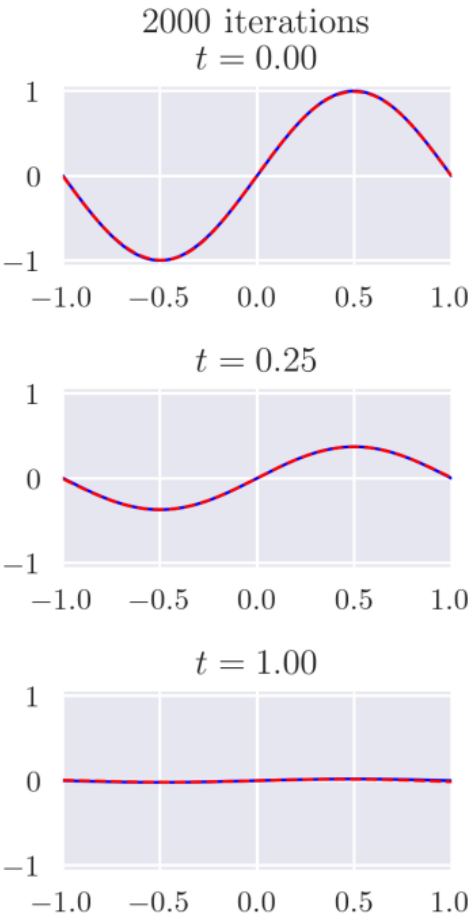
Alan Fernando Bravo Pimentel - A01645691 / Miguel Angel Rivas Torres - A0164462 /
Edgar Andrey Balvaneda - A01644770

Coincide con lo reportado en el artículo para PSO-BP-CD

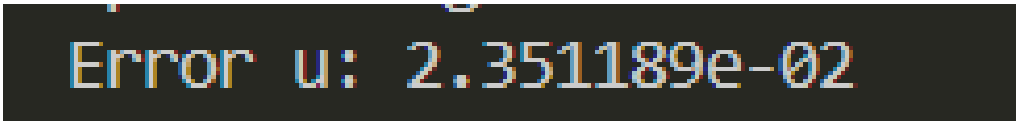
Advection	PSO	0.1177 ± 0.0714	(0.0391)
	PSO-BP	0.0234 ± 0.0017	(**0.0212**)
	PSO-BP-CD	$\mathbf{0.0232} \pm 0.0016$	(0.0213)

Diffusion (Heat equation)





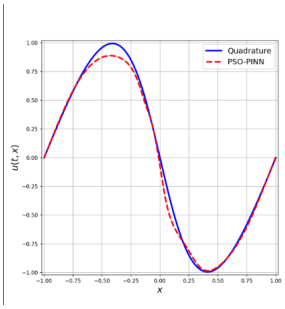
Coincide con lo reportado en el artículo (PSO-BP-CD).

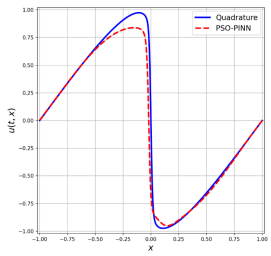


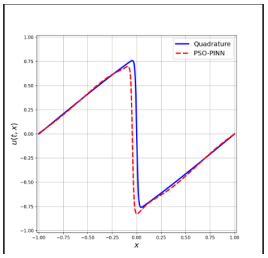
Heat	PSO	0.3974 ± 0.0834	(0.2617)
	PSO-BP	0.0068 ± 0.0020	(0.0027)
	PSO-BP-CD	$\mathbf{0.0051} \pm 0.0026$	(**0.0023**)

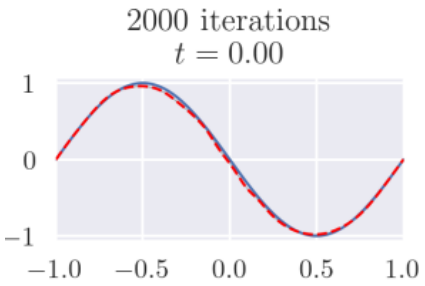
Alan Fernando Bravo Pimentel - A01645691 / Miguel Angel Rivas Torres - A0164462 /
Edgar Andrey Balvaneda - A01644770

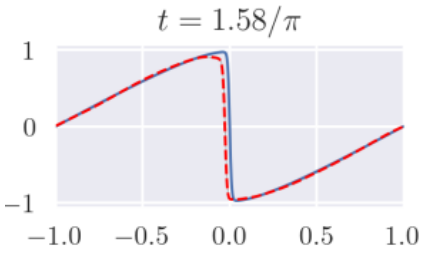
Burgers:

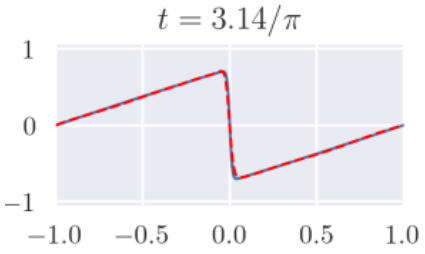













```
nu = 0.02/pi
L2 Error = 1.294207e-01
Last Loss:  0.0059695398
```

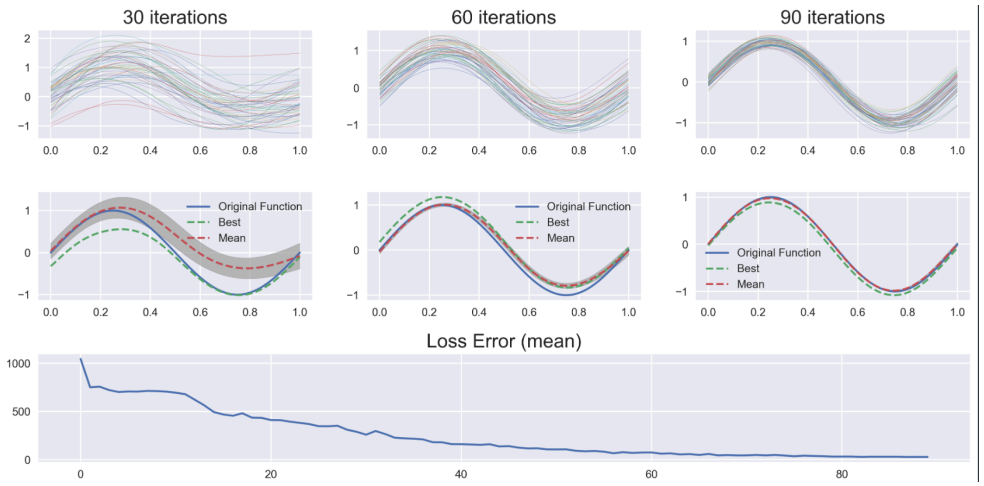
Burgers	PSO	0.4197 ± 0.0711	(0.2819)
	PSO-BP	0.1470 ± 0.0826	(0.0579)
	PSO-BP-CD	$\mathbf{0.0125} \pm 0.0502$	(**0.0057**)

Existe una diferencia de una orden de magnitud en el error a pesar de utilizar los mismos parámetros. Se tratará esta cuestión en la discusión.

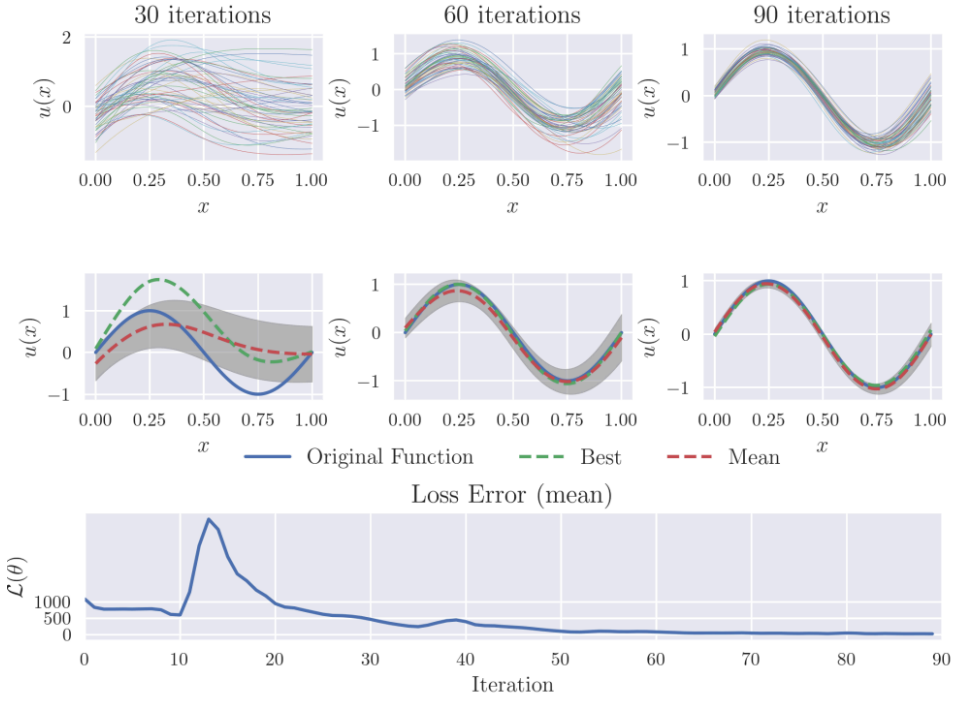
Alan Fernando Bravo Pimentel - A01645691 / Miguel Angel Rivas Torres - A0164462 /
Edgar Andrey Balvaneda - A01644770

Poisson

Réplica:



Artículo:



Discusión

Los resultados se lograron reproducir de manera satisfactoria y fueron muy similares. Estos experimentos, que fueron compartidos en el artículo, las soluciones convergieron de manera notable y robusta, llegando a demostrar la viabilidad del método PSO-PINN. Más importante aún, todos estos buenos resultados nos ayudó de aprendizaje para la implementación de un experimento adicional como es la ecuación de Sine-Gordon. En dónde los resultados obtenidos fueron excelentes y podemos ver que el método PSO-PINN funciona de gran manera tanto con las ecuaciones del artículo como la propuesta por nosotros.

Durante el análisis del artículo y el repositorio, se encontraron algunas discrepancias, pero no en la validez de los resultados, sino en la configuración del código. El error u de la ecuación de Burgers es una orden de magnitud superior al reportado en el artículo, probablemente por la falta de semilla o algún error en el artículo. Nos dimos cuenta de que el repositorio original algunas veces difería de las especificaciones que se mencionaban en su propio artículo. Dada la experiencia que tenemos podemos validar por completo los resultados. El artículo afirma que el método híbrido PSO-BP-CD llega a resolver PDEs de alta complejidad donde el descenso de gradiente llega a fallar, además el resolver la ecuación de Sine-Gordon, un problema no lineal, es una validación de la robustez que tiene este método. Entonces podemos decir que el enjambre mitiga las patologías de convergencia y llega a hacer soluciones muy precisas.

En el transcurso del proyecto hemos llegado a aprender una gran cantidad de lecciones clave, como lo es el costo de la robustez, debido a que entrenar un enjambre de redes tiene gran costo computacional, como se llegó a comprobar en el experimento de Sine-Gordon que tardó más de una hora. No obstante, este costo es el precio de la robustez, que se necesita para escapar de los óptimos locales dónde los métodos simples llegan a fallar. Una segunda lección es el aprender que el balance en un algoritmo híbrido lo es todo. El desarrollo de prueba y error nos ha enseñado a entender cómo la modificación de la lógica del algoritmo llega a afectar de manera directa la estabilidad de las soluciones.

Conclusiones

Los hallazgos principales de este proyecto se centran en tres áreas clave: la exitosa extensión del método al incluir la ecuación no lineal de Sine-Gordon, un experimento no incluido en el repositorio original que se resolvió con alta precisión , como segundo sería la optimización crucial del rendimiento mediante el uso de decoradores (@tf.function), lo que aceleró drásticamente el entrenamiento y, finalmente, el entendimiento profundo del funcionamiento y sensibilidad del método híbrido PSO-BP-CD que proveyó todo el proceso de prueba y error requerido para lograr la convergencia.

Podemos concluir que la reproducibilidad no es instantánea. Se publica el resultado final, que oculta el gran trabajo de prueba y error. Nuestra experiencia al resolver la ecuación de Sine-Gordon, en el que hubo que hacer más de 8 intentos, uno de ellos con una carga computacional de más de 1 hora, es prueba de que el ajuste de estos métodos no es solo un problema computacional, sino que es un problema en sí mismo, es por eso que sin las optimizaciones de rendimiento que se implementaron, el proceso de prueba y error sería algo inviable.

Para un trabajo a futuro hemos considerado el poder arreglar un problema que se mencionó anteriormente, que es el cuello de botella que causa un coste computacional. Siguiendo con el artículo original, un paso crucial es implementar estrategias de entrenamiento distribuido para escalar el método a enjambres mucho más grandes y problemas en 3D.

Bibliografía

Davi, C., & Braga-Neto, U. (2022). PSO-PINN: Physics-Informed Neural Networks Trained with Particle Swarm Optimization. arXiv. https://arxiv.org/abs/2202.01943

(S/f). Sciencedirect.com. Recuperado el 27 de octubre de 2025, de https://www.sciencedirect.com/topics/mathematics/sine-gordon-equation