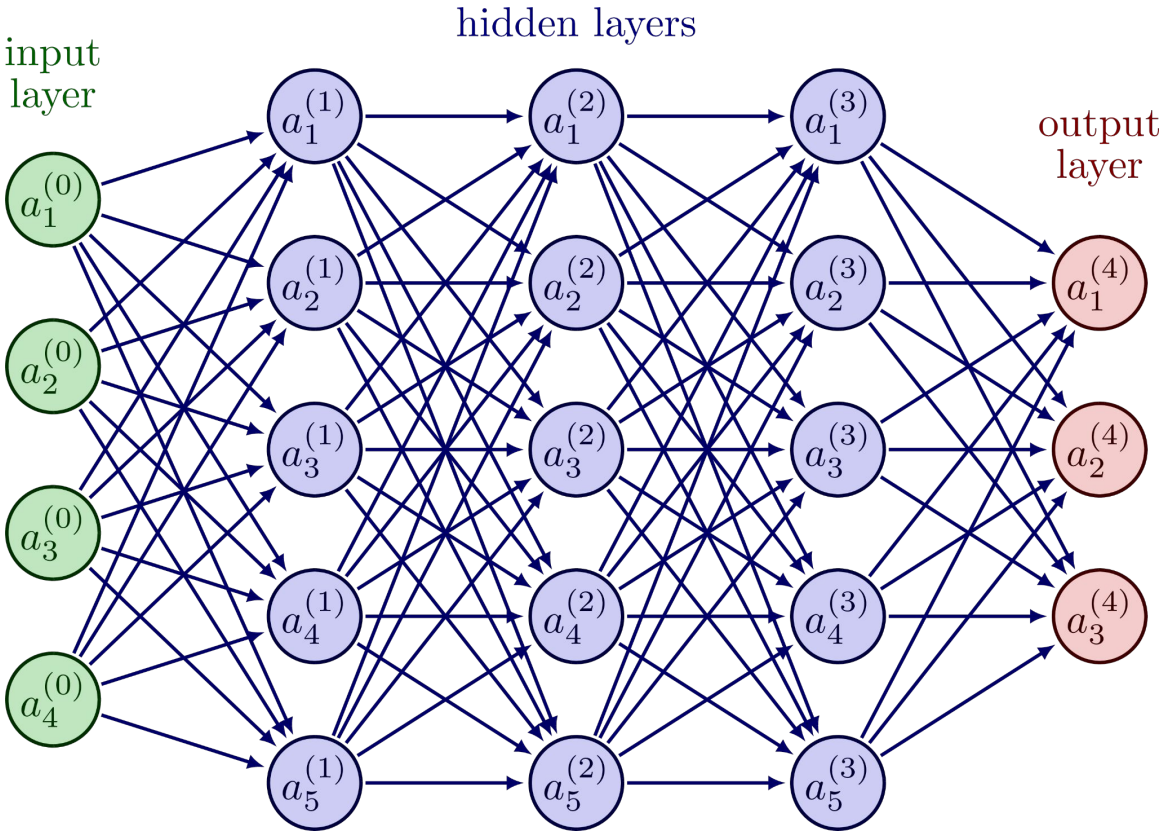
Automatic Differentiation && Physics Informed Neural Networks

Carlos Andrés del Valle

Automatic Differentiation

¿Cómo entrenar una red?



Gradient Descent

$$argmin_\theta \; J(\theta)$$

$$\theta_{i+i} = \theta_i - \lambda(\nabla_\theta J(\theta_i))^T$$

¿Cómo se calculan esas derivadas?

$$argmin_\theta \, J(\theta)$$

$$\theta_{i+i} = \theta_i - \lambda(\nabla_\theta J(\theta_i))^T$$

¿Cómo se calculan esas derivadas?

$f(\vec{x})$

- Analíticamente
- Simbólicamente
- Numéricamente
- Automáticamente

$$\frac{\partial}{\partial x_i} f(\vec{x})$$

Analíticamente

- Exacto
- Demorado

Simbólicamente

- Exacto
- Rápido
- Costoso

¿Qué pasa si?

$$f = f_1 \circ f_2 \circ \cdots \circ f_n$$

Numéricamente

$$\left.\frac{d}{dx} f(x)\right|_{x=a} = \frac{f(a+dx)-f(a)}{dx}$$

$$\left.\frac{d}{dx} f(x)\right|_{x=a} = \frac{f(a+dx)-f(a-dx)}{2dx}$$

Errores Numéricos Grandes

Diferenciación Automática

Soluciona todos los problemas anteriores. Hay 2 tipos:

- Forward Mode (Tangent)
- Reverse Mode (Adjoint)

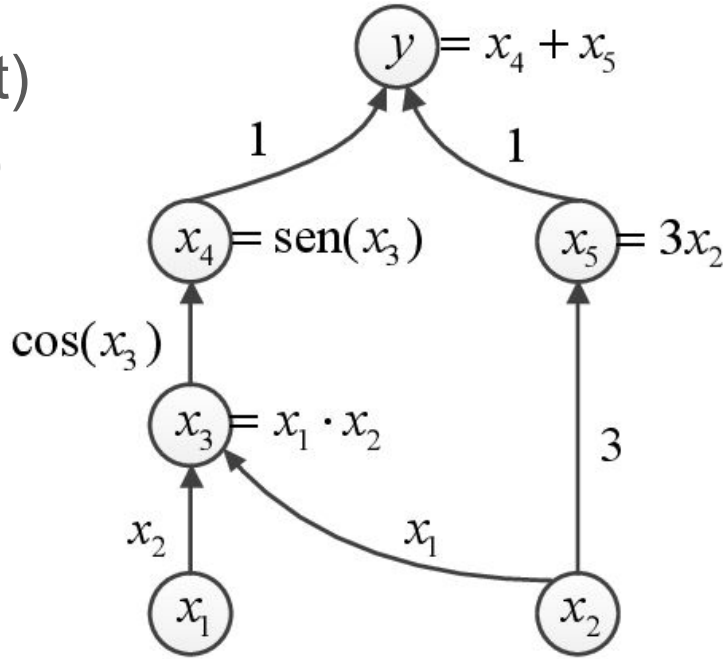


Fig. 1. Árbol computacional de $y = \text{sen}(x_1 \cdot x_2) + 3x$

Forward Mode

- Calcula Columnas del Jacobiano.
- Bueno cuando hay **pocas entradas** y **muchas salidas** (n<<m).
- En tiempo O(nf).
- En espacio O(1).
- No es adecuado para ML.

$$\mathbf{J} = \frac{d\mathbf{f}(\mathbf{x})}{d\mathbf{x}} = \left[\frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_1} \cdots \frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_u}\right] = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_u} \\ \vdots & & \vdots \\ \frac{\partial f_v(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_v(\mathbf{x})}{\partial x_u} \end{bmatrix}$$

Muy usado en Sensitivity Analysis.

Ejemplo

$$f(x) = x^2 + 3x \longrightarrow f'(x) = 2x + 3$$

```
function f(x)
    a = x*x
    b = 3*x
    return a+b
end
```

```
function AD_df(x,dx=1)
    a=x*x
    da=dx*x+x*dx
    b=3*x
    db=3*dx
    return da+db
end
```

Reverse Mode

- Calcula Filas del Jacobiano (gradientes).
- Bueno cuando hay **muchas entradas** y **pocas salidas** (n>>m).
- En tiempo O(mf).
- En espacio O(f).
- Es adecuado para ML.

$$\mathbf{J} = \frac{d\,\mathbf{f}(\mathbf{x})}{d\,\mathbf{x}} = \left[\frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_1} \cdots \frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_u}\right] = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_u} \\ \vdots & & \vdots \\ \frac{\partial f_v(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_v(\mathbf{x})}{\partial x_u} \end{bmatrix}$$

Derivadas de Orden Superior

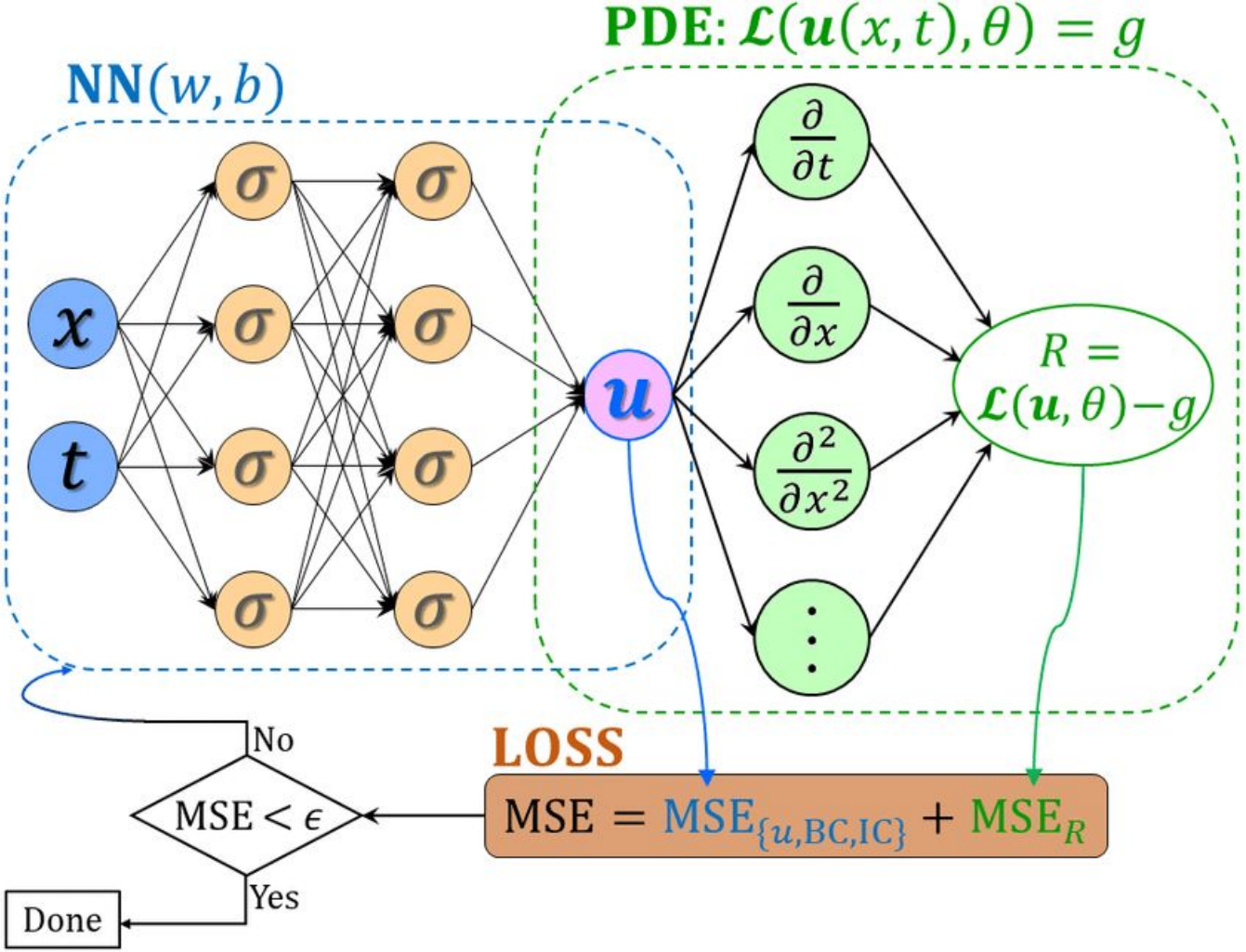
Forward Mode: $\nabla f \cdot v \rightarrow \dot{x} = v$

Reverse Mode: $\nabla^2 f \cdot v = H_f v$

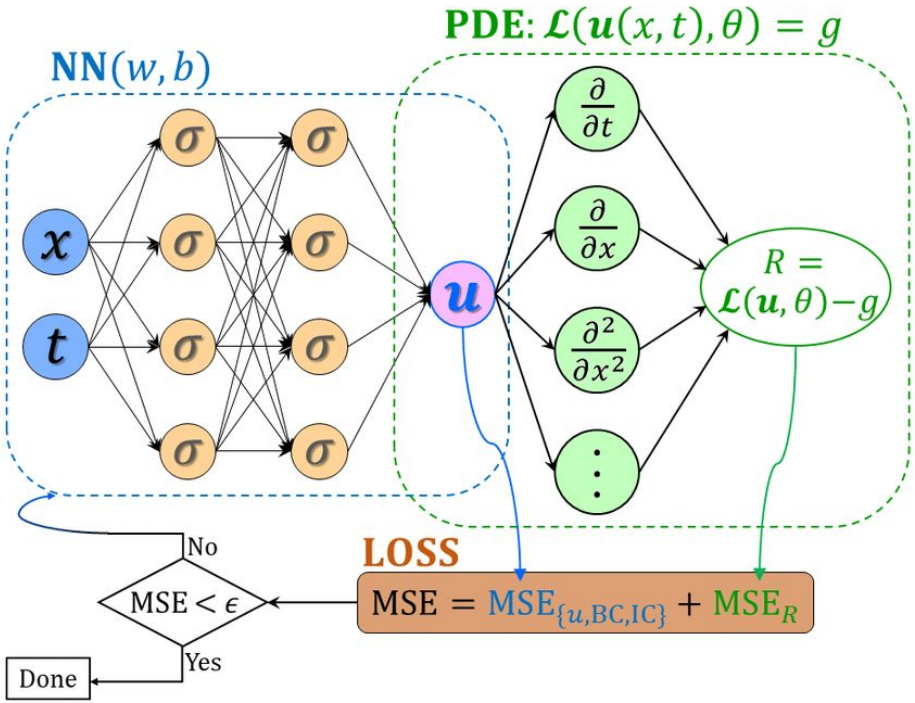
(No calculamos explícitamente J y H con AD)

$$H(f) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

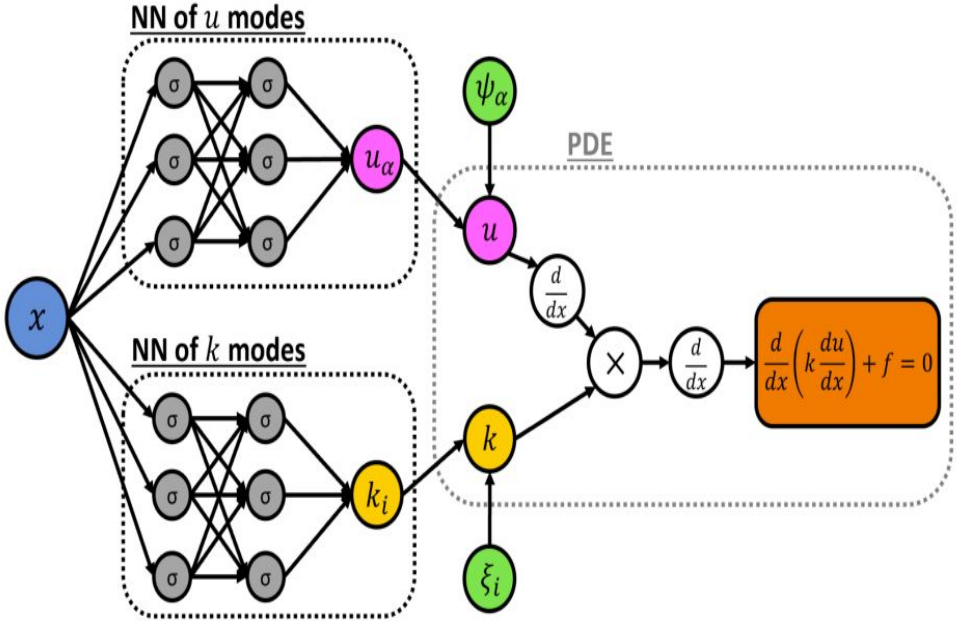
Physics Informed Neural Networks


PDE: $\mathcal{L}(u(x,t),\theta) = g$
NN(w,b)
x
t
σ
u
$\frac{\partial}{\partial t}$
$\frac{\partial}{\partial x}$
$\frac{\partial^2}{\partial x^2}$
$R = \mathcal{L}(u,\theta) - g$
LOSS
MSE $= \text{MSE}_{\{u,\text{BC,IC}\}} + \text{MSE}_R$
MSE $< \epsilon$
No
Yes
Done

Direct Problems



Inverse Problems



Comparación de una PINN con FEM

PINN

- Muy demorada de entrenar
- Evaluación muy rápida
- No es precisa: $10^{-3} - 10^{-7}$
- Geometrías complejas no son tan viables
- Descubrir parámetros
- IC y BC no son necesarias
- Se pueden añadir datos experimentales.
- Aceleración con GPU

FEM

- No hay que entrenar
- Evaluación lenta
- Muy preciso: $|u - \hat{u}| \leq ch \approx 10^{-8} - 10^{-16}$
- Geometrías complejas
- Lineal
- Toca conocer la forma débil
- IC y BC son necesarias
- Masivamente Paralelo

Cuándo Usar una PINN

- Ajustar Datos Experimentales.
- IC y/o BC Desconocidas.
- Parámetros de la Ecuación Desconocidos.
- PDE Muchas Dimensiones.
- PDE Integrodiferencial.
- PDE con Derivadas Fraccionarias.
- PDE Estocásticas.
- Comportamiento no Local.
- Vamos a Evaluar la Solución Muchas Veces.
- Cuando una Solución Discontinua da Problemas.

Cuándo NO Usar una PINN

- Métodos Clásicos Diseñados para esos Casos (Advección, Difusión, …).
- Geometrías Raras
- Comportamiento Caótico o Turbulento (Causal PINNs)
- Cuando Queremos Precisión
- Cuando Vamos a Evaluar la Solución Pocas Veces
- Cuándo no Tenemos una GPU

¿Cómo Mejorar la Solución?

$$f' = g \quad g' = w \longrightarrow f'' = w$$

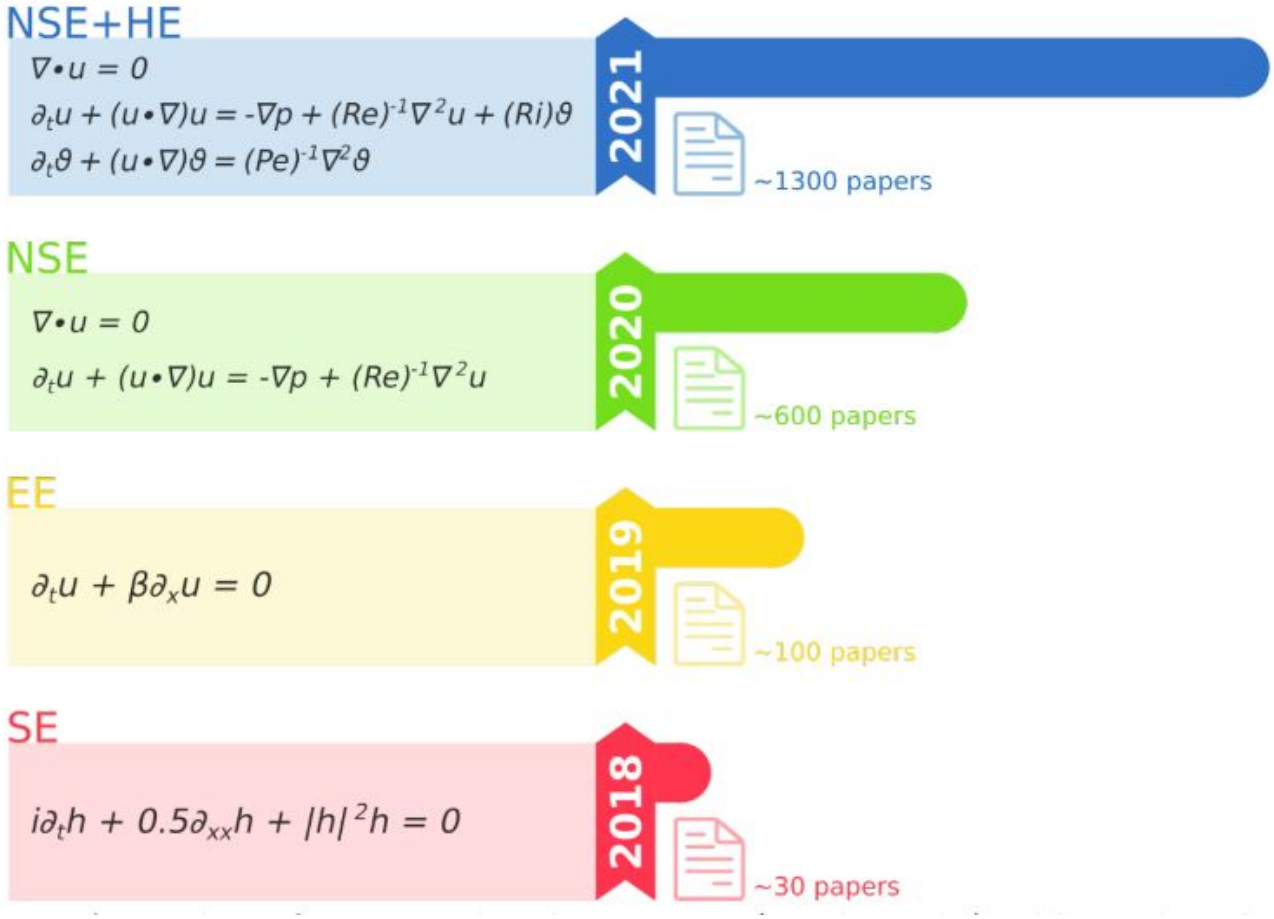
- No calcular derivadas de orden superior
- transformar la salida

$$\hat{u} = f(u)$$

- Entrenar en dominios [0,1]
- Mezclar optimizador Adam con BFGS optimizador O(2)
- Elegir los puntos problemáticos del dominio
- Funciones de activación continuas (teorema)
- Arquitectura de la red
- Imponer fuertemente condiciones de frontera $\hat{u} = b_0 + (u - a)u$

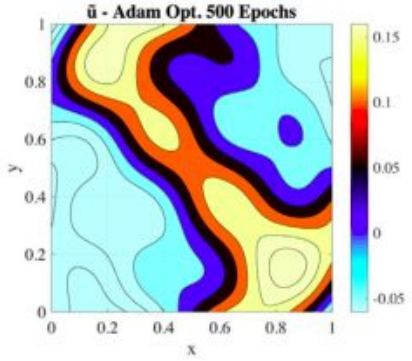
la frontera vale b0 en u=0 y u=a

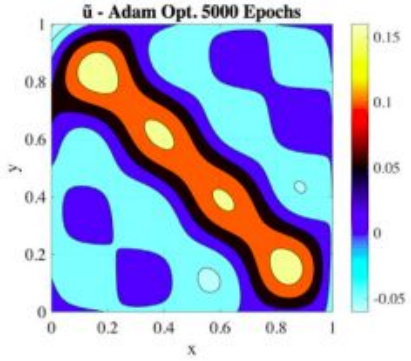
¿Qué es lo Más Trabajado?

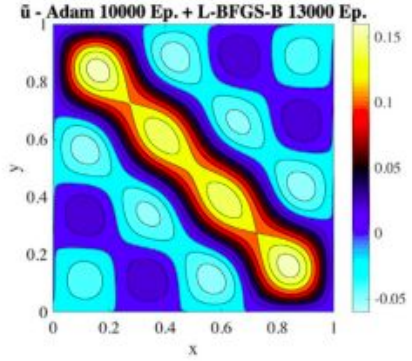


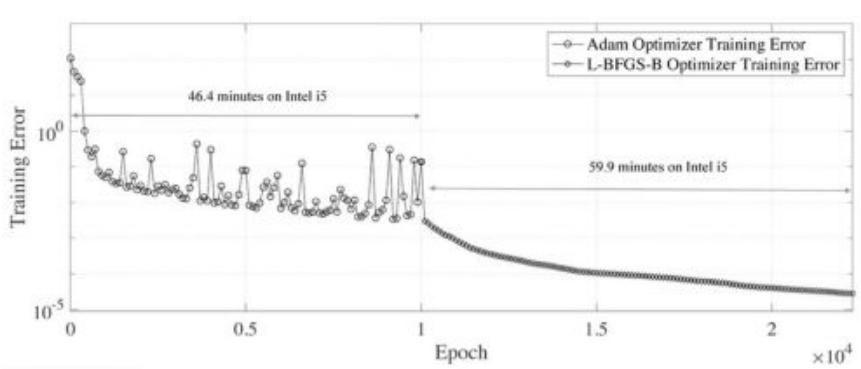
Una Aplicación Revolucionaria

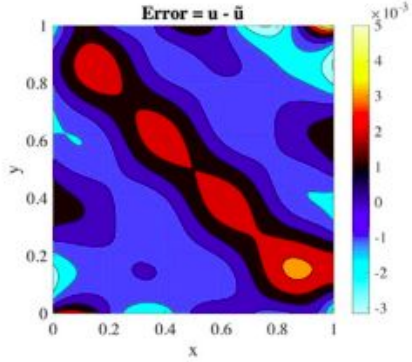
$$\nabla^2 u(x,y) = f(x,y) \qquad f(x,y) = \frac{1}{4}\sum_{k=1}^{4}(-1)^{k+1}2k\sin(k\pi x)\sin(k\pi y)$$



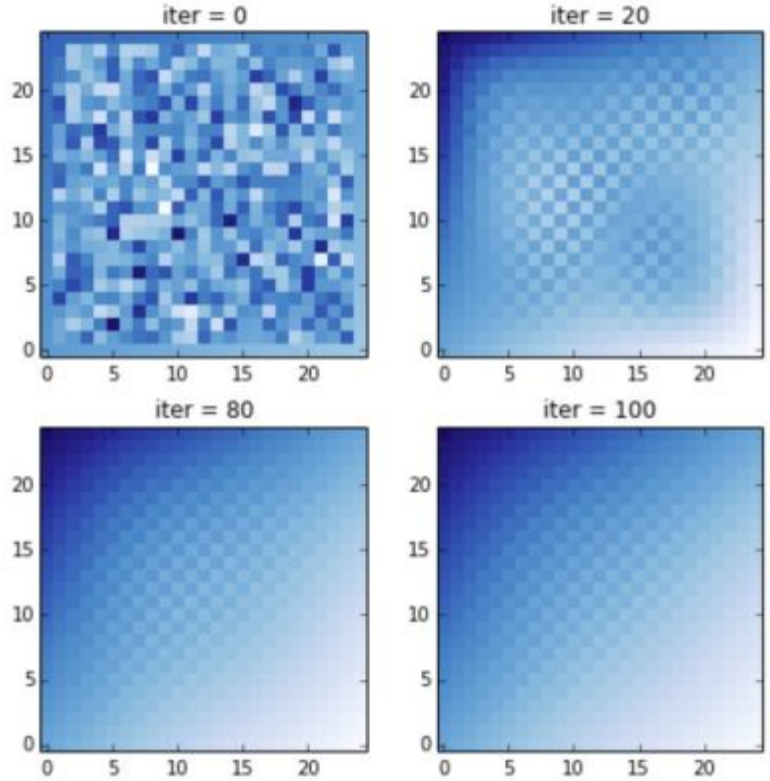


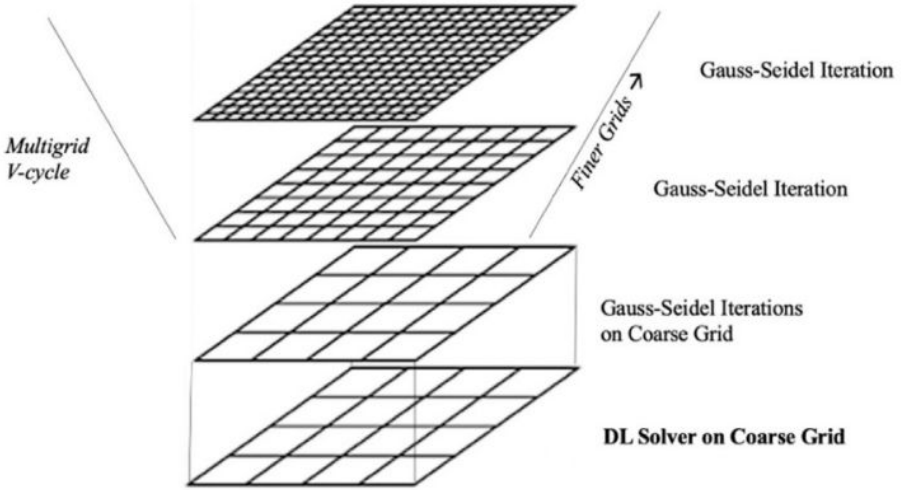


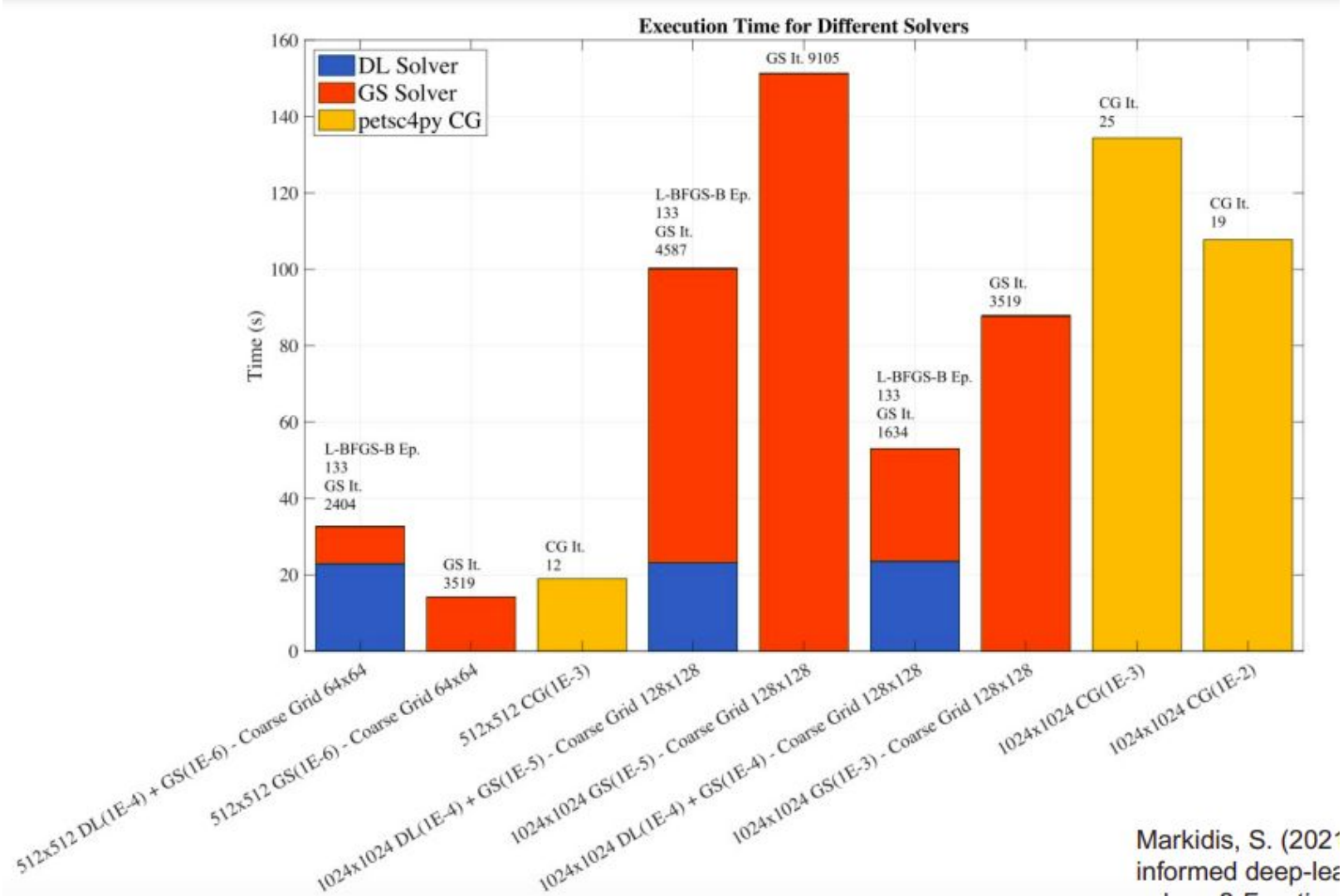




Jacobi Iteration







Markidis, S. (202
informed deep-lea

Tareas

- Ecuación de Onda
- Ecuación de laplace fraccionaria
- Ecuación Allen-Chan
- Ecuación de Helmholtz
- Ecuación de Klein-Gordon
- Ecuación de Schrodinger
- Ecuación de Fokker-Planck
- Ecuación de alta dimensionalidad:Hamilton-Jacobi-Bellman
- Ecuación de alta dimensionalidad: Black-Scholes-Barenblatt Equation
- Ecuación de Kolmogorov
- Atractor de Lorentz
- Atractor de Rössler

Bibliografía

Papers

- **Artificial neural network subgrid models of 2D compressible magnetohydrodynamic turbulence**
- **Neural Network Reconstruction of Plasma Space-Time**
- **Helicity-conservative Physics-informed Neural Network Model for Navier-Stokes Equations**
- **A deep learning framework for hydrogen-fueled turbulent combustion simulation**
- **Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations**
- **RESPECTING CAUSALITY IS ALL YOU NEED FOR TRAINING PHYSICS-INFORMED NEURAL NETWORKS**
- **Scientific Machine Learning through Physics-Informed Neural Networks: Where we are and What's next**
- **A Review of the Theory of Galactic Winds Driven by Stellar Feedback**
- **Can non-ideal magnetohydrodynamics solve the magnetic braking catastrophe?**

Links

- https://www.youtube.com/watch?v=DS3CZ_gVAMs&t=425s
- https://www.youtube.com/watch?v=vAp6nUMrKYg&t=15s
- https://www.youtube.com/watch?v=RJSMN-STTFk
- https://www.youtube.com/watch?v=jS-0aAamC64&t=619s
- https://www.youtube.com/watch?v=wG_nF1awSSY
- https://neuralpde.sciml.ai/stable/
- https://juliadiff.org/ForwardDiff.jl/stable/
- https://blog.rogerluo.dev/2018/10/23/write-an-ad-in-one-day/
- https://julia.quantecon.org/more_julia/optimization_solver_packages.html
- https://www.youtube.com/watch?v=rK-Bb6-0svs
- https://www.youtube.com/watch?v=KXb6KcetA10
- https://www.youtube.com/watch?v=BiB82F_fgUw
- https://www.youtube.com/watch?v=cFoe7SDcFPE&list=PLcK0exoS00ZTPdvhmh0IdyCIlVQ2lzjJ5&index=4
- https://www.youtube.com/watch?v=d8zc7fVEZHY&list=PLcK0exoS00ZTPdvhmh0IdyCIlVQ2lzjJ5&index=6
- https://www.youtube.com/watch?v=28cOPr1J5oI&list=PLcK0exoS00ZTPdvhmh0IdyCIlVQ2lzjJ5&index=7

- https://www.youtube.com/watch?v=IaS72aHrJKE
- https://github.com/zongyi-li/fourier_neural_operator/blob/master/fourier_3d.py
- https://www.youtube.com/watch?v=b1zojoTEmnI
- https://www.youtube.com/watch?v=xvOsV106kuA
- https://www.youtube.com/watch?v=OmySUTFwh2g
- https://www.youtube.com/watch?v=AXXnSzmpyoI
- https://www.youtube.com/watch?v=77jChHTcbv0
- https://www.youtube.com/watch?v=hKHI68Fdpq4
- https://www.youtube.com/watch?v=OaEu3UhCh1Y&list=PL1e3Jic2_DwwJQ528agJYMEpA0oMaDSA9&index=1
- https://www.youtube.com/watch?v=EO2lc4tXBHA&list=PLcK0exoS00ZTPdvhmh0IdyCIlVQ2lzjJ5&index=5
- https://www.youtube.com/watch?v=7n7xaviepKM
- https://docs.juliahub.com/NeuralPDE/uYXxU/3.11.0/pinn/fp/
- https://deepxde.readthedocs.io/en/latest/demos/pinn_forward.html
- https://neuralpde.sciml.ai/stable/
-