Comparative Study of Neural Networks for Learning the Parameters of a Differential Equation

Arthur Barjot

École Normale Supérieure de Lyon

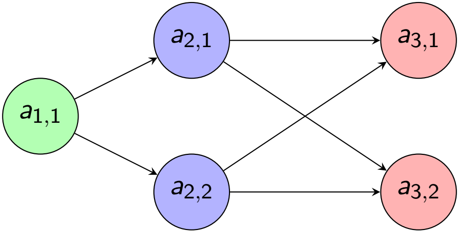
May 2024

Introduction

- Unusable solution:

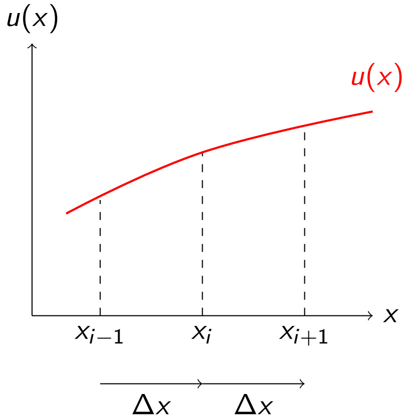
$$f(x) = \sum_{k \in \mathbb{Z}} \hat{f}(k) e^{2ik\pi}$$

- Approximate solution to PDEs



Finite Difference Method

- Finite Difference Scheme :



$$u'(x_i) \approx \frac{u(x_{i+1}) - u(x_i)}{\Delta x}$$

Second derivative approximation

- We can often use $u'(x_i) \approx \frac{u(x_{i+1})-u(x_i)}{\Delta x}$ or $u'(x_i) \approx \frac{u(x_i)-u(x_{i-1})}{\Delta x}$, for the second derivative we use both :

$$
\begin{aligned}
u''(x_i) &\approx \frac{u'(x_i) - u'(x_{i-1})}{\Delta x} \\
&\approx \frac{\frac{u(x_{i+1})-u(x_i)}{\Delta x} - \frac{u(x_i)-u(x_{i-1})}{\Delta x}}{\Delta x} \\
&= \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{\Delta x^2}
\end{aligned}
$$

Example : Heat Equation

$$\begin{cases} \forall x \in]0,1[, \forall t \in]0, t_f[, & \frac{\partial T}{\partial t}(t,x) = \kappa \frac{\partial^2 T}{\partial x^2}(t,x), \\ \forall t \in [0, t_f], & T(t,0) = T(t,1) = 0, \\ \forall x \in [0,1], & T(0,x) = \sin(2\pi x). \end{cases}$$

- $[0,1] \to \{x_0, x_1, ..., x_M\}$, $x_0 = 0, x_M = 1$ and $\Delta_x = \frac{1}{M}$
- $[0, t_f] \to \{t_0, t_1, ..., t_N\}, t_0 = 0, t_N = t_f$ and $\Delta_t = \frac{t_f}{N}$
- $T^{(i)} = \begin{pmatrix} T(t_i, x_1) \\ T(t_i, x_2) \\ T(t_i, x_3) \\ \vdots \\ T(t_i, x_{M-1}) \end{pmatrix}$ the vector of temperatures at time i

Example : Heat Equation

- $\forall k \in [\![1, M-1]\!], \forall i \in [\![0, N-1]\!]$,

$$\frac{T(t_{i+1}, x_k) - T(t_i, x_k)}{\Delta t} = \kappa \frac{T(t_i, x_{k+1}) - 2T(t_i, x_k) + T(t_i, x_{k-1})}{\Delta x^2}$$

$$\Rightarrow T(t_{i+1}, x_k) = (1-2\lambda)T(t_i, x_k) + \lambda T(t_i, x_{k+1}) + \lambda T(t_i, x_{k-1})$$

Where $\lambda = \frac{\kappa \Delta t}{\Delta x^2}$, so in matrix form : $T^{(i+1)} = AT^{(i)}$, where :

$$A = \begin{pmatrix} (1-2\lambda) & \lambda & 0 & \cdots & 0 \\ \lambda & (1-2\lambda) & \lambda & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \lambda & (1-2\lambda) & \lambda \\ 0 & \cdots & 0 & \lambda & (1-2\lambda) \end{pmatrix}$$

Example : Heat Equation

$$T^{(i+1)} = AT^{(i)} \Rightarrow T^{(i)} = A^i T^{(0)} = A^i \begin{pmatrix} T(0, x_1) \\ T(0, x_2) \\ T(0, x_3) \\ \vdots \\ T(0, x_M) \end{pmatrix} = A^i \begin{pmatrix} \sin(2\pi x_1) \\ \sin(2\pi x_2) \\ \sin(2\pi x_3) \\ \vdots \\ \sin(2\pi x_M) \end{pmatrix}$$

- Compute A^i, for all i
- A is tri-diagonal, there exists efficient algorithm : Thomas algorithm for tri-diagonal system

Such a good option?

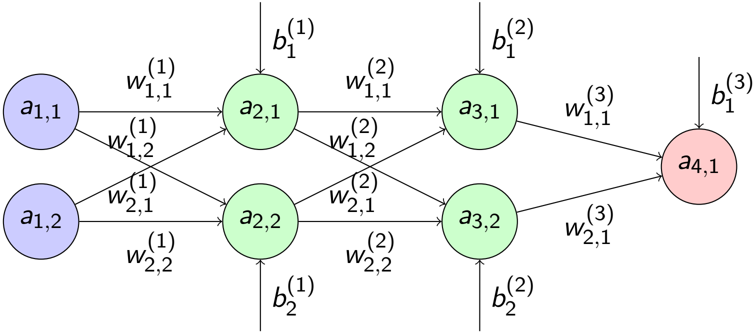
Advantages

- Adaptable to a wide range of linear differential equations
- Very simple to implement

Disadvantages

- Only linear equation
- Convergence depends on the discretisation.
- Requires a fine mesh to capture details.

Neural Networks



$$\forall i \in [\![2, N]\!],\ \forall j \in [\![1, n_i]\!], \quad a_{i,j} = \sigma\left(b_j^{(i-1)} + \sum_{k=1}^{n_{i-1}} w_{k,j}^{(i-1)} a_{i-1,k}\right).$$

Training Steps of a Neural Network

1. Initialization of the network weights and biases.
2. Forward Propagation:
   - Computing the predicted output.
3. Calculation of the Loss Function:
   - Comparison between the predicted output and the current labels (MSE): $\mathcal{L}_{DATA}(u) = \sum_{k=1}^{n} \left\| u(t_i^{data}, x_i^{data}) - u_i^{*,data} \right\|^2$
4. Backpropagation:
   - Computing the gradients of the loss function with respect to the weights and biases. (Chain rule)
   - Updating the weights and biases: $w_{j,l}^{(i)} \leftarrow w_{j,l}^{(i)} - \alpha \frac{\partial \mathcal{L}}{\partial w_{j,l}^{(i)}} (w_{j,l}^{(i)})$
5. Repeat steps 2 to 4 for multiple epochs.

Physics-Informed Neural Network (PINN)

$$ELU : x \mapsto \begin{cases} x & \text{if } x > 0, \\ \exp(x) - 1 & \text{if } x \leq 0. \end{cases}$$

- Database

$$\mathcal{L}_{DATA}(u) = \sum_{k=1}^{n} \left\| u(t_i^{data}, x_i^{data}) - u_i^{*,data} \right\|^2$$

Physics-Informed Neural Network (PINN)

$$\begin{cases} \forall x \in \Omega, \forall t \in]0, T[, & \frac{\partial u}{\partial t}(t,x) = F(u(t,x), \frac{\partial u}{\partial x}(t,x), ..., \frac{\partial^k u}{\partial x^k}(t,x)), \\ \forall t \in [0, T], \forall x \in \Gamma & u(t,x) = f(t,x), \\ \forall x \in \Omega, & u(0,x) = u_0(x), \end{cases}$$

$$\mathcal{L}_{PDE}(u) = \sum_{i=1}^{n} \left\| \frac{\partial u}{\partial t}(t_i^{PDE}, x_i^{PDE}) - F(u, t_i^{PDE}, x_i^{PDE}) \right\|^2,$$

$$\mathcal{L}_{BC}(u) = \sum_{i=1}^{n} \| \ u(t_i^{BC}, x_i^{BC}) - 0 \ \|^2,$$

$$\mathcal{L}_{IC}(u) = \sum_{i=1}^{n} \| \ u(0, x_i^{IC}) - u_0(x_i^{IC}) \ \|^2$$

Example : the Heat Equation

$$\begin{cases} \forall x \in]0,1[, \forall t \in]0,t_f[, & \frac{\partial T}{\partial t}(t,x) = \kappa \frac{\partial^2 T}{\partial x^2}(t,x), \\ \forall t \in [0,t_f], & T(t,0) = T(t,1) = 0, \\ \forall x \in [0,1], & T(0,x) = \sin(2\pi x). \end{cases}$$

$$\mathcal{L}_{PDE}(T) = \sum_{i=1}^{n} \left\| \frac{\partial T}{\partial t}(t_i^{PDE}, x_i^{PDE}) - \kappa \frac{\partial^2 T}{\partial x^2}(t_i^{PDE}, x_i^{PDE}) \right\|^2,$$

$$\mathcal{L}_{BC}(u) = \sum_{i=1}^{n} \| T(t_i^{BC}, 1) \|^2 + \sum_{i=1}^{n} \| T(t_{i+n}^{BC}, 0) \|^2,$$

$$\mathcal{L}_{IC}(u) = \sum_{i=1}^{n} \| T(0, x_i^{IC}) - \sin(2\pi x_i^{IC}) \|^2$$

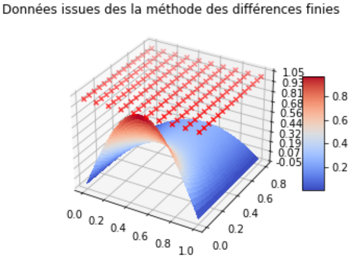
Case of study : Heat equation, the direct problem

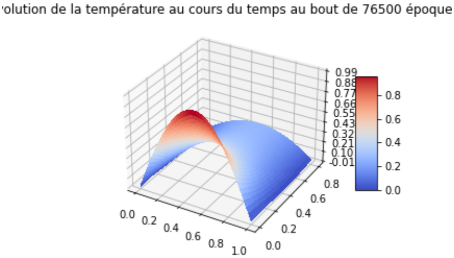
$$
\begin{cases}
\forall x \in]0,1[, \forall t \in \mathbb{R}_+^*, & \frac{\partial T}{\partial t}(t,x) = \kappa \frac{\partial^2 T}{\partial x^2}(t,x) \\
\forall t > 0, & T(t,0) = T(t,1) = 0 \\
\forall x \in [0,1], & T(0,x) = u_0(x)
\end{cases}
$$

- All the parameters are known
- Analytical Solution
- Python Program
- 2D Case

Some graphics

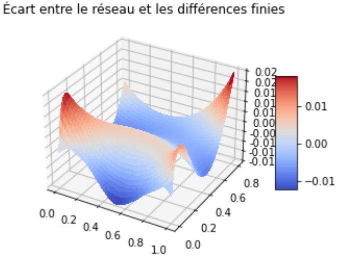
On the left, "real" values, on the right those obtain with our neural network :

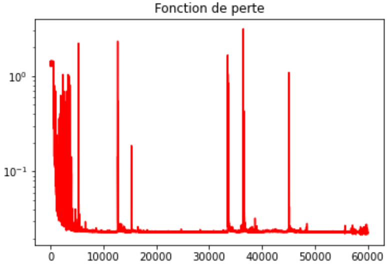




Some graphics

On the left the error curve, on the right the loss function :





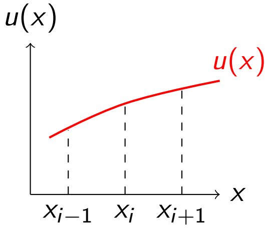
Numerical Method or Neural Network?

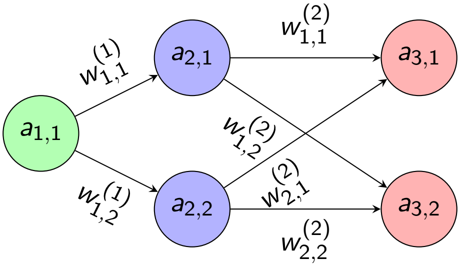
Finite Difference

- Do not need data base
- Very fast
- Discrete solution

Neural network

- Non linear problem
- Inverse problem
- Continuous solution





Heat equation, the inverse problem

Stationary Problem :

$$\begin{cases} \forall x \in]0,1[, & \Delta T(x) = \frac{-1}{\kappa} S(x) \\ & T(0) = T_0, \text{ et } T(1) = T_1 \\ \forall (x,u) \in \mathcal{D}, & T(x) = u \end{cases}$$

- We search T, without knowing S ; T_0, T_1 and κ are known
- Mathematical Form of S and Network Architecture

The inverse problem : Linear Source Term

$$S(x) = \alpha x + \beta$$

$$\begin{cases} \forall x \in]0,1[, & \Delta T(x) = \frac{-1}{\kappa}(\alpha x + \beta) \\ & T(0) = T_0, \text{ et } T(1) = T_1 \\ \forall (x,u) \in \mathcal{D}, & T(x) = u \end{cases}$$

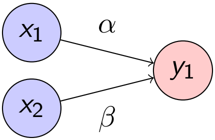
$$\Rightarrow T(x) = -\frac{\alpha}{6\kappa}x^3 - \frac{\beta}{2\kappa}x^2 + C_1 x + C_2$$

- As $T(0) = C_2 = T_0$, $T(1) = -\frac{\alpha}{6\kappa} - \frac{\beta}{2\kappa} + C_1 + T_0 = T_1$, we find :

$$T(x) = -\frac{\alpha}{6\kappa}x^3 - \frac{\beta}{2\kappa}x^2 + \left(\frac{\alpha}{6\kappa} + \frac{\beta}{2\kappa} + T_1 - T_0\right)x + T_0$$

The inverse problem : Linear Source Term

$$\Rightarrow T(x) = \alpha \times \underbrace{\left(\frac{1}{6\kappa}(x - x^3)\right)}_{x_1(x)} + \beta \times \underbrace{\left(\frac{1}{2\kappa}(x - x^2)\right)}_{x_2(x)} + \underbrace{((T_1 - T_0)x + T_0)}_{C(x)}$$

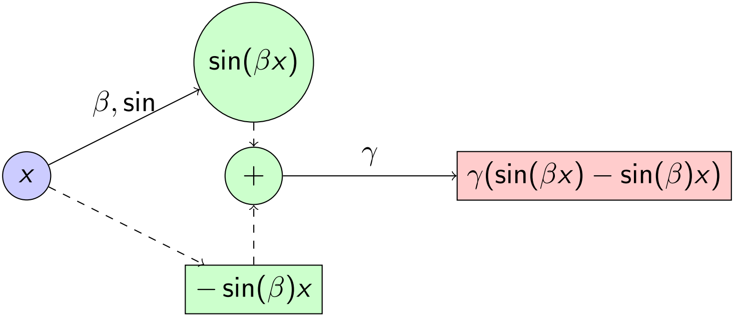


$$x \to (x_1(x), x_2(x)) \qquad T(x) \to T(x) - C(x)$$

$$NN(x_1(x), x_2(x)) = T(x) - C(x)$$

The inverse problem : Sinusoidal Source Term

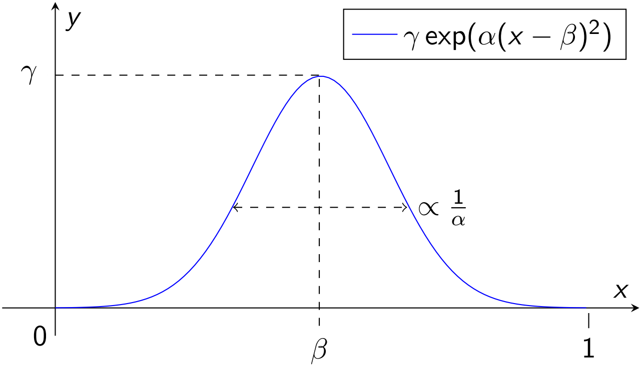
$$S(x) = \alpha \sin(\beta x)$$



The inverse problem : Gaussian Source Term

$$S = \gamma \exp(\alpha(x-\beta)^2) \text{ où } \alpha < 0,\ \beta \in]0,1[\text{ et } \gamma > 0$$

- Unsolvable equation



The inverse problem : Gaussian Source Term

- Taylor expansions :

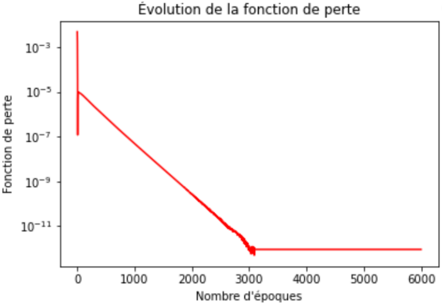
$$T(x) \approx T(\beta) + (x-\beta)T'(\beta) + \ldots + (x-\beta)^k \underbrace{T^{(k)}(\beta)}_{\frac{-1}{\kappa}S^{(k-2)}(\beta)},$$

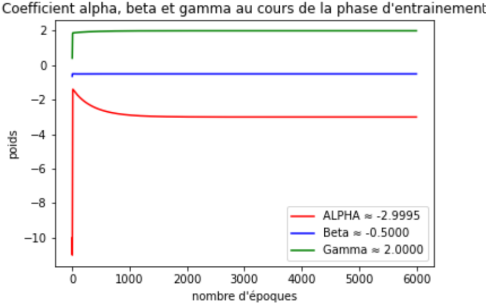
- We make the equation 'solvable'

$$T(x) \approx \gamma \left(C_1 x + C_2 + \underbrace{\sum_{i=0}^{12} (\alpha(x-\beta)^2)^i (x-\beta)^2 \frac{1}{i!(2i+1)(2i+2)}}_{=:\Sigma(x)} \right), \tag{1}$$

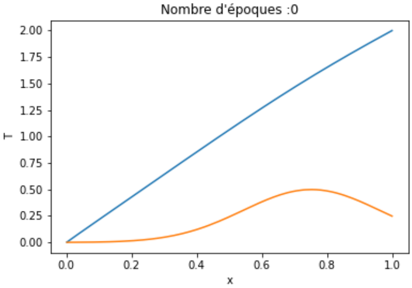
The inverse problem : Gaussian Source Term

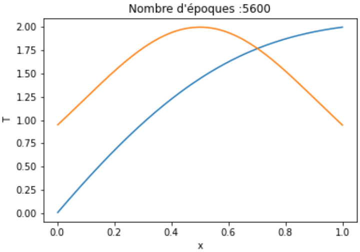
On the left the loss function, on the right the values of the coefficients α, β, γ :

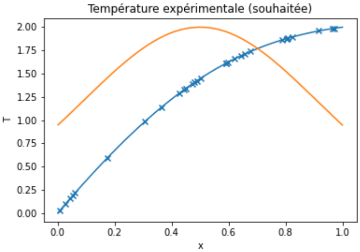




The inverse problem : Gaussian Source Term

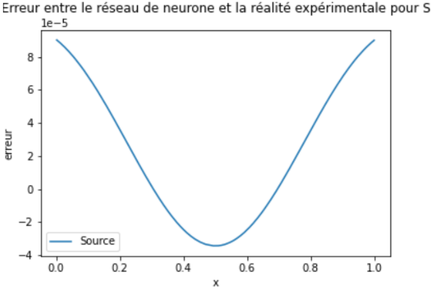


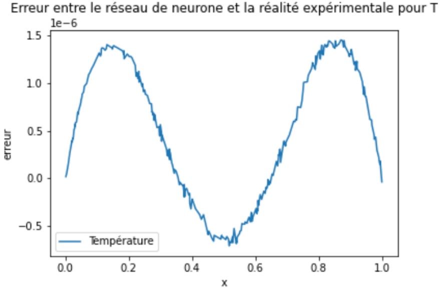




The inverse problem : Gaussian Source Term

On the left : the absolute error of the source term, on the right : absolute error of the temperature.



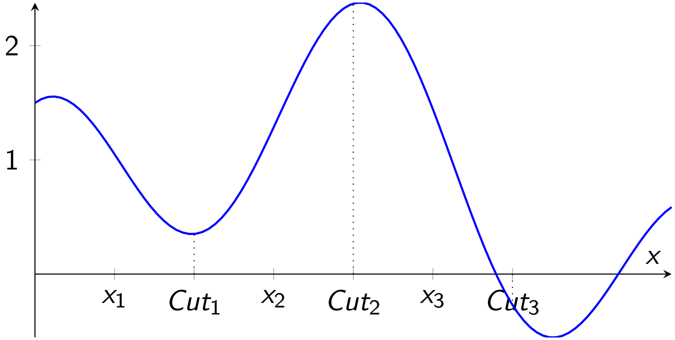


For further

- Generalisation ?
- Locally / Globally
- Can we check the error ?

For further

- Cut the space



- Make Taylor expansion in x_1, x_2, and x_3

Conclusion