Training augmented interpolation

FONSECA HINCAPIE Diana Sol Angel

23 August 2023

Objectives

The main objective of this internship project is to solve simple transport equations using the Semi-Lagrangian scheme and the deep interpolation operator.

For this the specific objectives are:

- Implement the PINNs strategy.
- Implement the Semi-Lagrangian scheme.
- Study of the error.

Introduction

We are interested in the resolution of transport equations, the aim is to find a way to increase the accuracy of the method using deep learning

Advection Equation

$$\begin{cases} \frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0 \quad (x,t) \in \Omega \times (0,T) \\ u(x, t=0) = u_0(x) \quad x \in \delta\Omega \end{cases} \tag{1}$$

Has an analytical solution equal to:

$$u(t,x) = u_0(x, x - at)$$

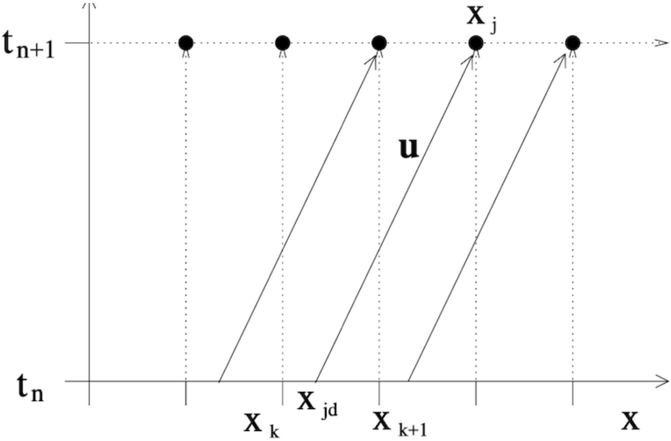
General context

This internship was supervised by the National Institute for Research in Digital Science and Technology (INRIA) and the University team Modeling and Control of the IRMA laboratory.

Semi-Lagrangian Scheme

The Semi-Lagrangian scheme is defined as it follows:

$$u\left(t_{n+1}, x_i\right) = \mathcal{I}_h^m\left(u\left(t_n, x\right)\right)\left(x_i - a\Delta t\right)$$



Lagrange Interpolation Operators

The Lagrange interpolation operator:

$$\mathcal{I}_h^m(f)(x) = \sum_{i=1}^{m} f\left(x_i\right) P_i(x)$$

with $P_i\left(x_j\right) = \delta_{ij}$
The Deep Lagrange Interpolation:

$$\mathcal{I}_d^m(f) = \sum_{i=1}^{n} \frac{f\left(x_i\right)}{u_\theta\left(x_i\right)} P_i(x) u_\theta(x) = \mathcal{I}^m\left(\frac{f}{u_\theta}\right) u_\theta(x)$$

With $P_i\left(x_j\right) = \delta_{ij}$ Using this choice, we obtain that $\mathcal{I}_d(f)\left(x_i\right) = f\left(x_i\right)$ as the classical interpolator.

u_θ Function:

How do we choose u_θ ?
We use a neural network which will approximate the $u_\theta(x)$ function.

$$u_{pred} = u_\theta(x, t, \mu, \sigma, a)$$

We train a neural network with a **Physics Informed Neural Network** strategy, and we use the previous interpolation to approximate the solution of the PDE.

Universal Approximation Theorem

Any continuous function $f : [0, 1]^n \to [0, 1]$ can be approximated arbitrarily well by a neural network with at least 1 hidden layer with a finite number of weights.

Even if neural networks can express very complex functions compactly, determining the precise parameters (weights and biases) required to solve a specific PDE can be difficult.

Physics informed neural network

- PINNs are a type of universal function approximators that can embed the knowledge of any physical laws that govern a given data-set in the learning process, and can be described by PDEs.
- They approximate PDE solutions by training a neural network to minimize a loss function; it includes terms reflecting the initial and boundary conditions along the space-time domain's boundary and PDE residual and data points.
- PINNs training can be thought of as a unsupervised and/or supervised learning approach.

Physics Informed Neural Network

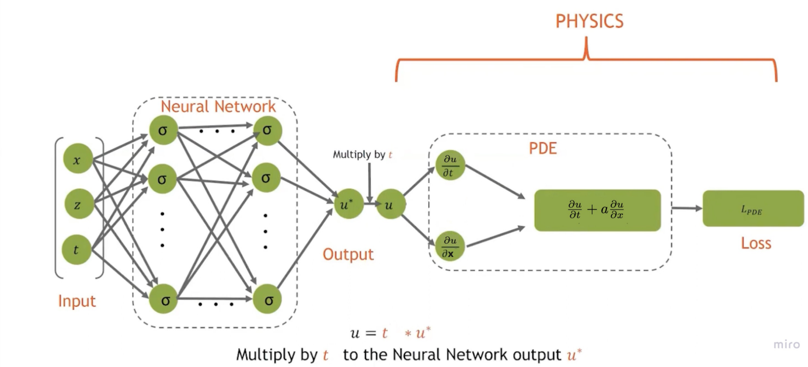

PHYSICS
Neural Network
σ
σ
σ
σ
σ
σ
x
z
t
Input
Multiply by t
u^*
u
Output
PDE
$\frac{\partial u}{\partial t}$
$\frac{\partial u}{\partial \mathbf{x}}$
$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x}$
L_{PDE}
Loss
$u = t \;\; * u^*$
Multiply by t to the Neural Network output u^*
miro

Figure: PINN strategy - Source: Youtube video "Introduction to PINNs"

Methodology

We have an initial condition u_0:

$$u_0 = \exp\left(-\frac{(x-\mu)^2}{\sigma}\right)$$

with μ the mean and σ the variance of the Gaussian distribution.

Using the **PINNs** approach we determine the parameters β of the NN, u_{pred}:

$$u_{pred}(x,t,\beta) = u_0(x) + b_c(x) u_{\beta}^{NN}(x,t)$$

We assume then that:

$$u_{pred}(x,t,\beta) \approx u(x,t)$$

Where the **Loss function** is defined as:

$$L(\beta) = L_{data}(\beta) + L_{physics}(\beta) + L_{SBC}(\beta) + L_{TBC}(\beta) + L_{IC}(\beta)$$

Implementation

We implemented a solver based on the SL scheme, and implemented the deep Lagrange interpolation operator with the u_{pred} approximated by a NN that uses the PINNs strategy for different initial conditions depending on the values of μ and σ.

- **Neural Network class:** Represents a neural network model.
- **Network class:** Represents a PINNs model implementing the previous NN class and the Adam optimizer algorithm.
  - **Train function:** Optimization step for training a NN by combining PDE constraints, data fitting and boundary conditions, implementing the MSE loss, and a backpropagation step.
- **Semi-Lagrangian solver:** Implements the SL scheme using the deep interpolation.

Results PINNs startegy

Finding u_{pred} using PINNs

For training the neural network and solving the transport equation we used the following parameters:

- $min = 0.$
- $xmax = 1.$
- $tmin = 0.$
- $tmax = tf$
- $a = 1.$
- $learning\ rate = 1e-3$
- $min\ mean = 0.45$
- $max\ mean = 0.55$
- $min\ variance = 0.01$
- $max\ variance = 0.05$

Unsupervised learning

We trained our NN with 40.000 epochs and 50.000 collocation points, the best loss we obtained was: 3.76e-04 With a Gaussian initial condition with mean $\mu = 0.49$ and variance $\sigma = 0.04$

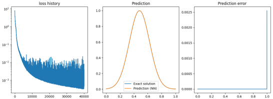


Figure: t=0

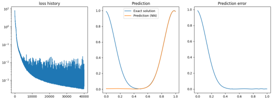


Figure: t=0.5

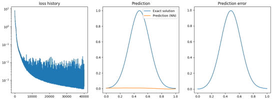


Figure: t=1.

Supervised learning

We trained our NN with 40.000 epochs and 15.000 data points, the best loss we obtained was: 3.21e-04 With a Gaussian initial condition with mean $\mu = 0.48$ and variance $\sigma = 0.042$

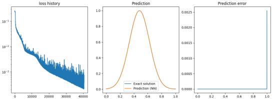


Figure: t=0

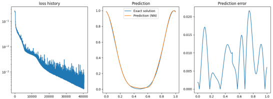


Figure: t=0.5

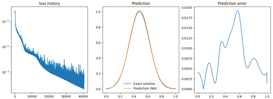


Figure: t=1.

Unsupervised and Supervised learning

We trained our NN with 40.000 epochs, 50.000 collocation points and 10.000 data points, the best loss we obtained was: 7.80e-04 With a Gaussian initial condition with mean $\mu = 0.5$ and variance $\sigma = 0.035$

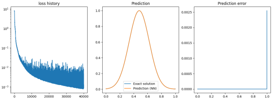


Figure: t=0

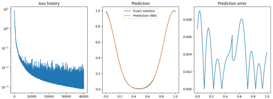


Figure: t=0.5

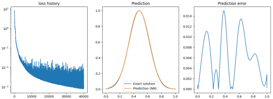
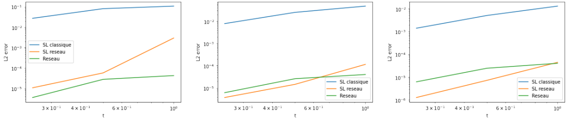


Figure: t=1.

SL deep Interpolation vs classical interpolation

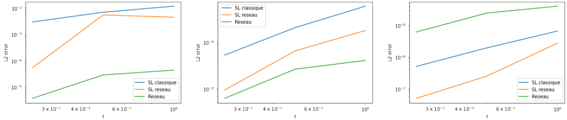
For $nt = 100$ and different values of nx we have:

1st Order Interpolation



(a) nx=10 (b) nx=20 (c) nx=40

3rd Order Interpolation



(a) nx=10 (b) nx=20 (c) nx=40

Average gain

By randomly drawing 20 parameters for the mean and the variance to have different initial conditions, we calculate for each the solution with the classic and deep SL method, we look at the gain = SL error/deep SL error and by doing the average and by making several draws of 20 parameters we obtain:

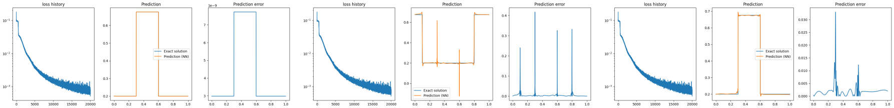
For a 1st order interpolation:
137.7119541688143, 97.15674818809492,185.7843283662372
For an interpolation of order 3:
7.608668520339034, 3.1395029764283313,4.496461973620286

Other work done

Discontinuous initial condition "Créneau"

$$u_0(x) = \begin{cases} A & \text{if } 0.40 \leq x \leq 0.50, \\ B & \text{otherwise}, \end{cases} \tag{2}$$

By training our NN with A varying beetween 0.6 and 0.7 and $B = 0.3$ fixed, epochs $= 20.000$, collocation $= 0$, data$= 10000$ and $A = 0.66$ we obtained:



(a) t=0 (b) t=0.5 (c) t=1

Conclusions

- By implementing the SL scheme with the deep Lagrange interpolation operator and the u_{pred} function found using the PINNs strategy a reduction in the error is found as compared to the Lagrange interpolation.
- The application of the deep operator with 1st-degree interpolation exhibits a more significant improvement when compared to the 3rd-degree interpolation.
- The PINNs strategy exhibit a remarkable capacity to assimilate a diverse range of solutions for the advection equation, with different initial conditions and parameters

Final remarks

Outlook:
There is still further research to be done implementing different initial conditions, other deep learning models and explore their full potential in solving more complex equations.

Personal feedback:
This internship provided me with a valuable opportunity to delve into a field where my prior knowledge was limited to the surface level.
The topic of PINNs really interested me and how we can apply deep learning to more scientific fields. I found really interesting projects done by others in the medical field or chemistry.
It challenged my patience, organization skills. In overall, I feel I learned a lot and even if there were things that didn't work out at the end I feel proud of my work.

Bibliography

1 Semi Lagrangian
`https://www.youtube.com/watch?app=desktop&v=egnsVOvJYIA`

2 Neural Networks and the Universal Approximation Theorem Milind Sahay Published in Towards Data Science

3 Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next, S. Cuomo, V. Schiano Di Cola, F. Giampaolo, G. Rozza, M. Raissi, F. Piccialli1