Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Dr. Andreas Adelmann

Physics informed neural networks (PINN) (with a mathematically informed architecture) for the nuclear decay equation

Semester Project for Miha Pompe

Introduction

Isotopic decay and transmutation is governed by the nuclear decay equation

$$\frac{\vec{N}(t)}{dt} = A\vec{N}(t)\,, \quad \text{with} \quad \vec{N}(t=0) = \vec{N}_0\,, \tag{1}$$

where $\vec{N} \in \mathbb{R}^n$ is the isotopic composition, $A \in \mathbb{R}^{n\times n}$ is the decay/transmutation matrix, and $n \in \mathbb{N}$ is the number of isotopes. In common problems the number of isotopes is $n = 2000$ up to $n = 4000$, and A is stiff, with up to $||A|| \sim 10^{20}$. Furthermore A is sparse and close to upper-triangular. Accurately solving the equation is notoriously hard due to the stiffness of the problem [1].

The decay equation is used in virtually all nuclear codes, both for simulation of reactor cores, and for radioactive decay in spent fuel. Thus, accurate and fast methods for the solution of the equation are very important for modelling reactors and for criticality safety of spent fuel repositories.

Physics Informed Neural Networks

Physics Informed Neural Networks (PINNs) [2] offer a new paradigm for solving differential equations. The student Guclielmo Pacifico is exploring the viability of general-purpose PINNs for solving the nuclear decay equation. The neural network has input t and output $\vec{N}(t)$ and, in the one-layer case, can be written as

$$\vec{N}(t) = M\sigma\left(\vec{w}t + \vec{b}\right)\,, \quad \text{with} \quad M \in \mathbb{R}^{n\times n},\, \vec{w}, \vec{b} \in \mathbb{R}^n\,,$$

where $\sigma(x)$ is a common activation function such as $\tanh$ or $\texttt{GELU}$. In this one-layer case the optimiser has to learn $2n + n^2$ parameters, and even more if the number of layers is increased. Furthermore, à priori one does not know how many neurons or layers will be required.

However, upon closer examination of the problem (1), it seems that there might exist networks of a specific architecture that are more adequate for the problem at hand. This project aims to explore these type of networks.

Mathematically informed architecture

It is known [3] that if A has n distinct real eigenvalues, the solution to (1) is the linear combination

$$\vec{N}(t) = \sum_{i=1}^{n} \vec{a}_i e^{\lambda_i t} \,,$$

where $\vec{a}_i \in \mathbb{R}^n$ and $\{\lambda_i\}$ are the eigenvalues of A. This solution can be reformulated as a neural network with a hidden layer of n neurons, an activation function $\sigma(x) = e^x$, and known weights of the first layer $\vec{w} = (\lambda_1, \lambda_2, ..., \lambda_n)$. I.e. the solution can be written as

$$\vec{N}(t) = M\vec{v}\,, \quad \text{with} \quad \vec{v} = (e^{\lambda_1 t}, e^{\lambda_2 t}, ..., e^{\lambda_n t})\,.$$

This network architecture has the advantage that only n^2 parameters need to be learned, and the exact required numbers of neurons and layers are known. Thus many hyperparameters are known in advance (whereas with regular neural networks tuning the hyperparameters is nontrivial).

More complex matrices

Having n distinct eigenvalues is ensured with probability 1, but having only real eigenvalues is not. If some eigenvalues are complex, the solution is

$$\vec{N}(t) = \sum_{i=1}^{n} e^{\mathrm{Re}(\lambda_i)t} \cos\left(\mathrm{Im}(\lambda_i)t\right) \vec{a}_i + e^{\mathrm{Re}(\lambda_i)t} \sin\left(\mathrm{Im}(\lambda_i)t\right) \vec{b}_i \,,$$

with $\vec{a}_i, \vec{b}_i \in \mathbb{R}^n$. This solution can also be represented as a neural network, with more neurons and a slightly different activation function.

Several open questions need to be explored regarding this network architecture:

- Is it more efficient than a general-purpose neural network? Especially when A is stiff.
- How expensive is it to obtain the eigenvalues of A, given its sparse nature? Especially in the case of complex eigenvalues this seems nontrivial. Could one also "learn" the eigenvalues if they are not known?
- Will the imaginary case converge?

1 Project

Goal: explore the viability of mathematically-informed PINNs for solving the decay equation (1).

In the first step the PINN will be applied to matrices with real known eigenvalues, of varying stiffness. Toy problem with $n = 3$ at first, and secondly $n = 4000$.

This method shall be compared to state-of-the-art methods such as CRAM [3] or the analytical chains, and to the general-purpose PINNs (the PINNs that Guglielmo is working on).

If successful, matrices with complex eigenvalues may be considered.

This project will be done with a custom PyTorch neural network (in which case the student can start from Guglielmo's code and collaborate with him).

Deliverables

- presentation at the end of the project
- well-documented code and a short to the point documentation written in LaTeX. The documentation must be written such that it is intelligible to a fellow-student. All presentations, the report and the code must be pushed to a GitLab repository.

Contacts

- Andreas Adelmann, andreas.adelmann@psi.ch, Tel: 056 310 42 33
- Arnau Albà, arnau.albajacas@psi.ch
- Romana Boiger, romana.boiger@psi.ch

Literature

[1] J. M. Hykes and R. M. Ferrer. Solving the Bateman equations in CASMO5 using implicit ode numerical methods for stiff systems. Technical report, American Nuclear Society - ANS; La Grange Park (United States), July 2013.

[2] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, February 2019.

[3] M. Pusa and J. Leppänen. Computing the Matrix Exponential in Burnup Calculations. *Nuclear Science and Engineering*, 164(2):140–150, February 2010.