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

Miha Pompe
Supervised by: Dr. A. Adelmann, A. Albà, Dr. R. Boiger

Outline

- Nuclear decay equation
- Physics-informed neural networks
- Implementation
- Results
- Conclusion

- We are interested in the time evolution of isotope concentrations in spent nuclear fuel and in nuclear cores.
- The decay equation:

$$\frac{d\mathbf{N}(t)}{dt} = A\mathbf{N}(t)\,, \qquad \text{with} \quad \mathbf{N}(t=0) = \mathbf{N}_0\,.$$

- $A \in \mathbb{R}^{n\times n}$ can be very large ($n = 4000$) and stiff $|A| = \frac{|\lambda_{max}|}{|\lambda_{min}|} > 10^{20}$.
- Formal solution of the equation is $\mathbf{N}(t) = e^{At}\mathbf{N}(0)$[1].
- CRAM and Padé are rational approximation methods that estimate the exponential function[2].
    - CRAM: accurate, less reliable for burnup matrices.
    - Padé: fast, but inaccurate in stiff cases.
    - Runge-Kutta: accurate, but slow.

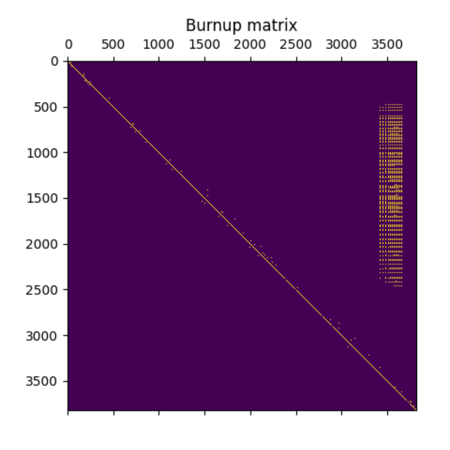
[1] Centar (2013)
[2] Pusa (2010)

- A matrix can be a burnup or decay matrix.
- A general solution can be written in form of [2]:

$$N(t) = \sum_{i=1}^{n} a_i \cos(\mathrm{Im}(\lambda_i)t) e^{\mathrm{Re}(\lambda_i)t} + b_i \sin(\mathrm{Im}(\lambda_i)t) e^{\mathrm{Re}(\lambda_i)t}.$$

- Solution for the decay problem [2]:

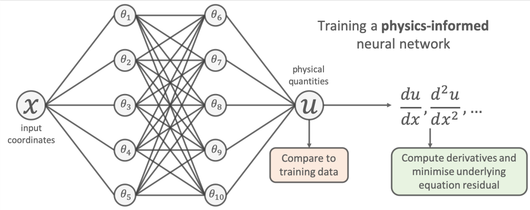
$$N(t) = \sum_{i=1}^{n} a_i e^{\lambda_i t}.$$



[2] Pusa (2010)

Physics-informed neural networks (PINN)

- PINNs are comprised of two parts, neural network and physics-based loss function. They are used to solve PDEs.[3]



- Physics-based loss function:

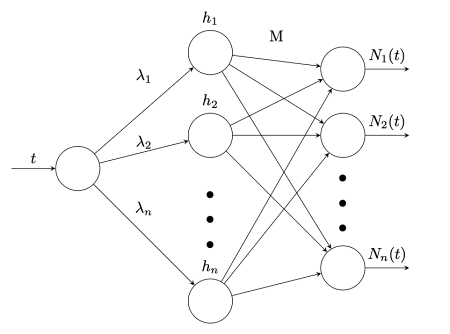
$$\mathcal{L}_{ODE} = \left| \frac{d\mathbf{N}(t)}{dt} - A\mathbf{N}(t) \right|^2, \quad \mathcal{L}_{IC} = |\mathbf{N}(0) - \mathbf{N}_0|^2$$

- The weights of the network are adjusted to minimize both the prediction error and the physics-based loss.
- Potential use case of PINNs is much faster uncertainty analysis compared to other methods.

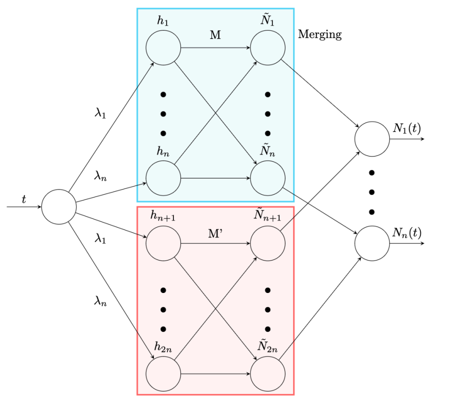
[3] Raissi (2019)

Implementation - Decay problem

- Solution for decay problem is in form $\mathsf{N}(t) = \sum_{i=0}^{n} \mathsf{a}_i e^{\lambda_i t}$ $\Leftrightarrow \mathsf{N}(t) = M\mathbf{h}$, with $\mathbf{h} = (e^{\lambda_1 t}, ..., e^{\lambda_n t})$.
- Input and output layer with one hidden layer.
- Change the activation function to $\sigma(z) = e^z$.
- Constraining the weights of the network: fix weights in input layer, M triangular.
- Trainable weights $\frac{n(n+1)}{2}$.
- Reduction in number of trainable weights and hyperparameters.



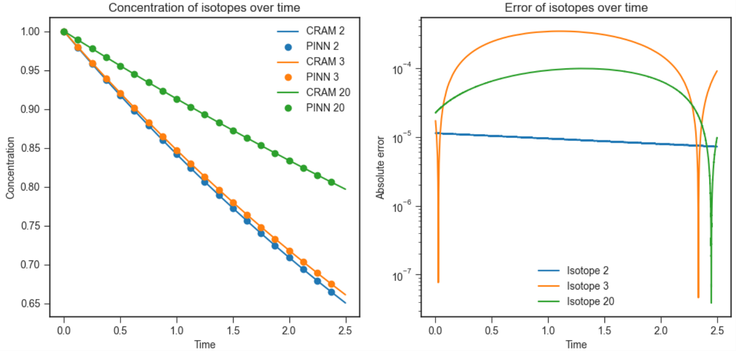
Implementation - Burnup problem



- Solution of the burnup problem is $N_j(t) = M_{ij}h_i + M'_{ij}h_{n+i}$, where $h_i = \cos(\text{Im}(\lambda_i)t)e^{\text{Re}(\lambda_i)t}$, $h_{n+i} = \sin(\text{Im}(\lambda_i)t)e^{\text{Re}(\lambda_i)t}$.
- Two neural networks and a merging layer.
- Two activation functions.
- Fix weights in input layer.
- Trainable weights $2n^2$.

Results - Solution

- Analysis done on a decay matrix with initial conditions $N_i(0) = 1$ for all i.
- The method also achieves similar performance with burnup matrices.



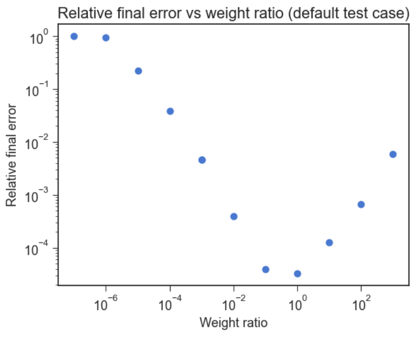
- Decay matrix of size 50×50, with stiffness 10^{16} (default test case).
- Final error defined as the sum of absolute errors for each isotope at final time, $\Delta N(t_{max}) = |\mathbf{N}_{CRAM}(t_{max}) - \mathbf{N}(t_{max})|$.

Results - Weight ratio

The total loss is a weighted sum of both losses:

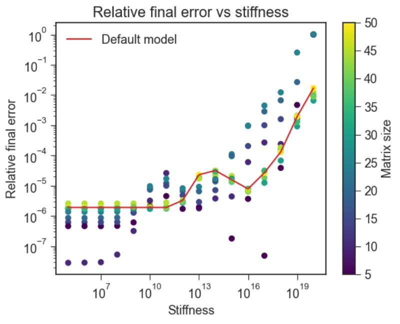
$$\mathcal{L} = \frac{w_{ODE}\mathcal{L}_{ODE} + w_{IC}\mathcal{L}_{IC}}{w_{ODE} + w_{IC}}.$$

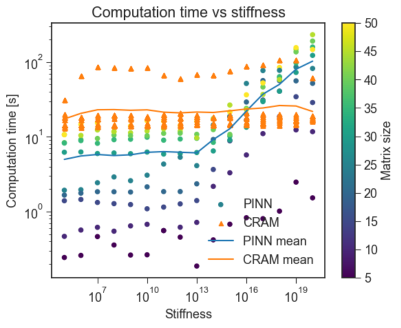
Define weight ratio as $w = w_{IC}/w_{ODE}$. It turns out that it is a very important hyperparameter that should be tuned for each test case.



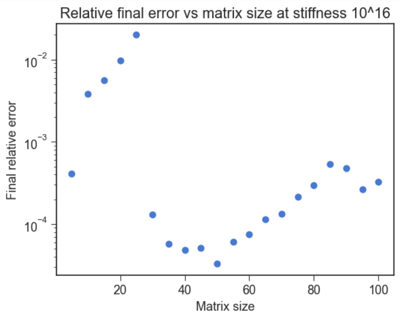
Results - Dependence on stiffness

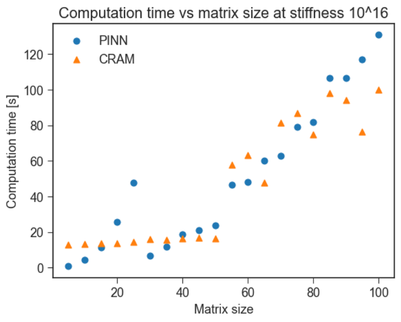
- Final relative error slightly rises up to a stiffness cutoff at round 10^{19}.
- Computation time for PINN is roughly constant up to the stiffness cutoff. Computation time for CRAM is independent of stiffness.
- All the simulations were done on CPU, but code also runs on GPU, where the computation time is lower.





- The relative final error slightly dependent on matrix size.
- Computation time scales linearly with the size of the matrix, as does the CRAM method for $n > 45$.
- All matrices derived from ENDF/B VII.1 nuclear data library.





Conclusions

The method works:

- for both decay and burnup matrices,
- for any matrix size (tested on up to 442×442 burnup matrix, limited by memory and time),
- for any stiffness up to about $|A| = 10^{19}$,
- faster than CRAM for matrices with stiffness under 10^{16}, with relative final error of around 10^{-5},
- faster and can handle bigger and more stiff problem than general purpose PINNs.

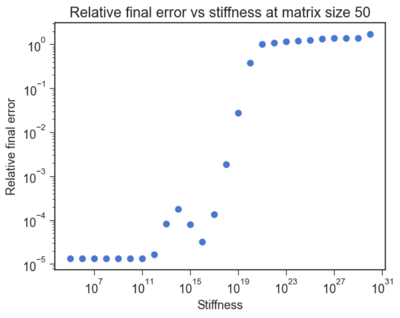
Future work:

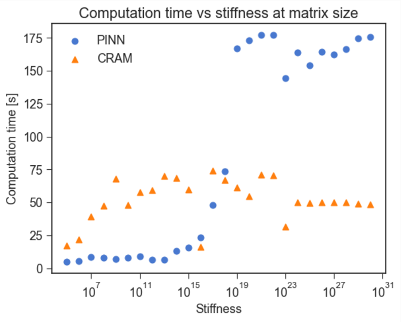
- improve the performance at bigger stiffnesses,
- find an optimal way of setting the weight ratio.

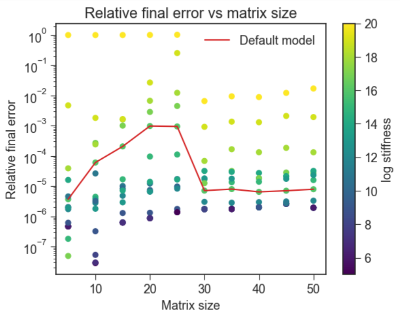
References

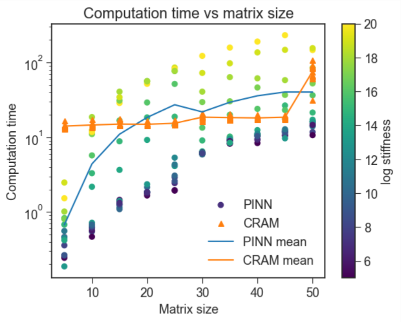
1. J. Cetnar *General solution of Bateman equations for nuclear transmutations.* Annals of Nuclear Energy (2013) https://www.sciencedirect.com/science/article/pii/S0306454906000284: :text=The%20Bateman%20equations%20for%20radioactive, decay%20constant%20of%20ith%20nuclide
2. M. Pusa and J. Leppänen. *Computing the Matrix Exponential in Burnup Calculations.* Nuclear Science and Engineering: 164, 140–150, (2010) https://www.tandfonline.com/doi/abs/10.13182/NSE09-14
3. 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, (2019) https://www.sciencedirect.com/science/article/pii/S0021999118307125
4. H. Baty. *Solving stiff ordinary differential equations using physics informed neural networks (PINNs): simple recipes to improve training of vanilla-PINNs.* (2023)
5. G. Pacifico and A. Adelmann, *Solving the Decay Equation using a Physics Informed Neural Network (PINN).* (2023)

Backup slides - More data on stiffness dependence

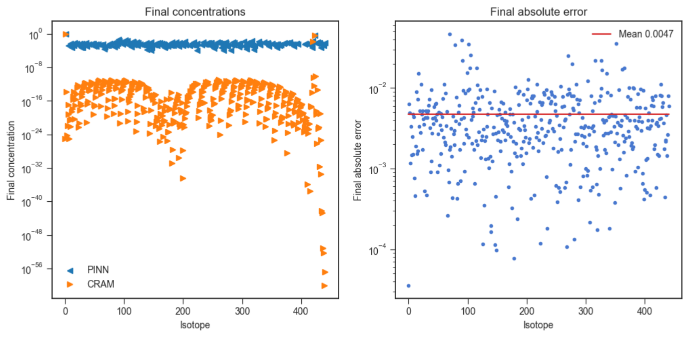


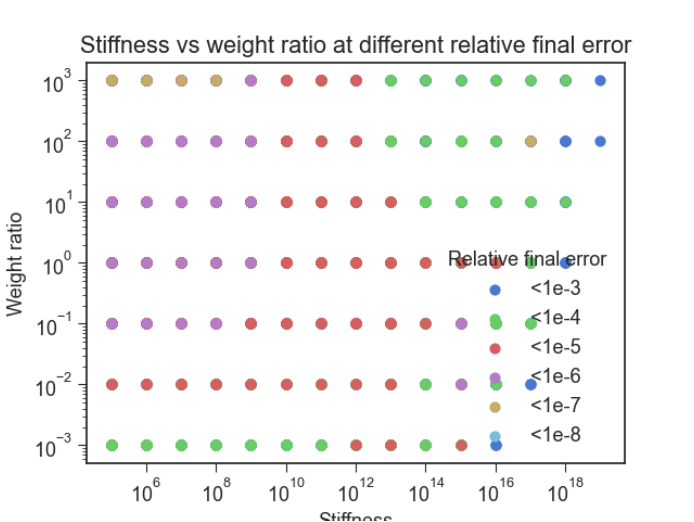



Relative final error vs matrix size
Default model
Relative final error
Matrix size
log stiffness
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10
20
30
40
50
20
18
16
14
12
10
8
6


Computation time vs matrix size
Computation time
Matrix size
log stiffness
PINN
CRAM
PINN mean
CRAM mean
10^{2}
10^{1}
10^{0}
10
20
30
40
50
20
18
16
14
12
10
8
6

Backup slides - Burnup problem




Stiffness vs weight ratio at different relative final error
Weight ratio
Stiffness
Relative final error
<1e-3
<1e-4
<1e-5
<1e-6
<1e-7
<1e-8
10^{3}
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{6}
10^{8}
10^{10}
10^{12}
10^{14}
10^{16}
10^{18}