# PINN White Paper

Ilia Kheirkhah

## Abstract

Conventional Methods of solving heat transfer problems can be time consuming and expensive, especially for complex systems. Machine Learning methods can remove these barriers. Once trained, Machine Learning methods can be a fast alternative to conventional simulations, but have yet to be widely adopted. Here we explore how Physics Informed Neural Networks (PINNs) can be used in Heat Transfer and analyze how Neural Network error compares to conventional errors. Data is obtained from a Finite Difference Method solution to the Heat Equation with simple, uniform heat generation. Various heat generation values are used to train the Physics Informed Neural Network which is then evaluated at various spatial locations and heat generations rates to estimate an error profile for the Neural Network. We find that the average error of PINNs over the spatial domain is constant and roughly zero for low heat generation terms. Analysis of the error trend with respect to the heat generation terms show us that at some critical heat generation value, $q_{crit}$ the error randomly increases. The exact value follows no predictable trend but is many orders of magnitude larger than conventional methods, even on course grids. We also see that the evaluation time scaled linearly with the number of field points, compared to the number of field points cubed for conventional methods. Given this, PINNs have great potential in the preliminary design phase. However, due to the increasing accuracy outside of the training range, conventional methods should still be used where maximum precision is necessary. In addition, conventional simulations methods are necessary for training PINNs in the way we are today, maintaining their necessity for new systems.

## Background and Problem

Many problems in engineering cannot be solved analytically. As a result, we resort to numerical methods to help in the design process. In heat transfer applications, these simulations help tell us temperature and heat flux values over our entire field of interest. This information helps us ensure that no melting, structural weakening, or dangerous conditions exist in the finalized part. However, simulations can also be used in structural dynamics to predict stresses and failure modes or in fluid dynamics to predict flow properties such as lift and drag.

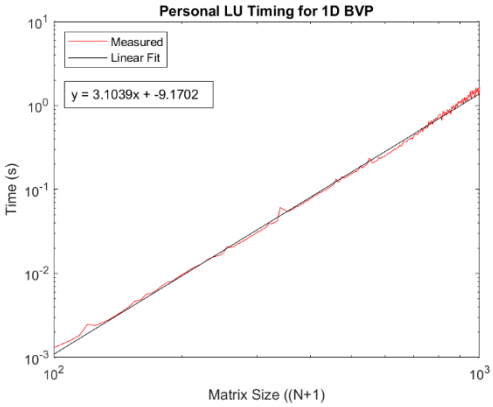


*Figure 1: Conventional Method Time Scaling*

There are 3 common methods for Finite Difference Methods, Finite Element Methods, and Finite Volume Methods. Each has their own niches and limitations, but they all fundamentally work by dividing our domain into small but finite subdomains. The properties of your solution are then assumed to have some profile in these subdomains. Most of the time, a linear variation is assumed but higher order methods can also be used. Once this assumption is made, one then solves your equation with these simplified subdomains. This results in almost all problems being eventually simplified to solving a linear system of equations. This is

something that can be written in the form $Ax = b$, where $A$ is a matrix and $x, b$ are both vectors. The problem with these methods is that solving linear systems is, in general, very computationally expensive. If you want to solve this system with $N$ subdomains, the time to solve this linear system scales as $N^3$. For large systems, which are necessary for high fidelity, this can result in simulations lasting up to multiple days for simulations that are run. In the most ideal cases, simulations might need to last weeks, which exceeds the computational limits most people have. This means that many simulations that people would like to run are out of the range of computational possibility.

There exists several methods of speeding up simulations, such as multithreading, using GPUs, or taking advantage of linear system structure to speed up the evaluation of a linear system. However, these methods generally do not change the scaling of the problem significantly, so new methods are always being looked for.

## Solution

Neural Networks have shown the ability to be very quick once trained. We propose using this fact to greatly speed up simulation time while still capturing physics by taking advantage of Physics Informed Neural Networks (PINNs). Like traditional Neural Networks, PINNs have a data loss, which makes sure that the system fits to provided data well. However, we introduce a new PDE loss. To understand this, let us first introduce the problem we are solving.

In this analysis, we will be solving a simple steady heat equation with a heat generation term with fixed zero temperature boundary conditions. This gives us:

$$\frac{d^2T}{dx^2} + q = 0$$

$$T(0) = T(1) = 0$$

In this case, our PDE is already in the form $F(T) = 0$, which allows us to simply make this our loss. To do this quickly, we take advantage of the ability of neural networks to find derivatives automatically and quickly. This allows us to make a full loss:

$$L = L_{data} + L_{PDE}$$

$$L_{data} = \frac{1}{N}\sum\left(T_{pred} - T_{true}\right)^2$$

$$L_{PDE} = \frac{d^2T}{dx^2} + q$$

This allows us to train our system to fit not only the data, but the governing equation which we are trying to model also. To get data, we used a finite difference solution to gather basic data. We used the DeepXDE framework in Python, a library built for PINNs with the capability of finding higher derivatives.

# Conclusion

Upon implementing our PINN in python and training, we saw significant time improvements. Instead of scaling as size cubed, we saw a linear scaling as seen in Figure 2. As a result of the linear scaling, the low number of points actually ends up being slower. However, simulations are not meant to be used to gather a few hundred data points but several hundred thousand data points. At these scales, which we could not test here due to time constraints, a PINN would be significantly faster due to the more generous scaling.

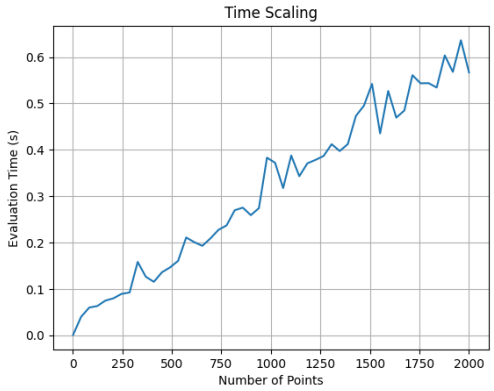


*Figure 2: PINN Time Scaling*

Although PINN are faster, they have the limitation of training ability. In conventional methods, the error grows linearly with the heat generation term (more generally, linearly with the second derivative). However, PINN do not have this connection with the physics and instead are nearly perfect up until a critical heat generation term. At this point, the error grows unpredictably. This behavior can be seen in Figure 3. This means that PINN have a risk of not giving realistic results if not verified on certain benchmark cases beforehand. However, once this $q_{crit}$ if established, results below this value can be trusted and have close to zero error.

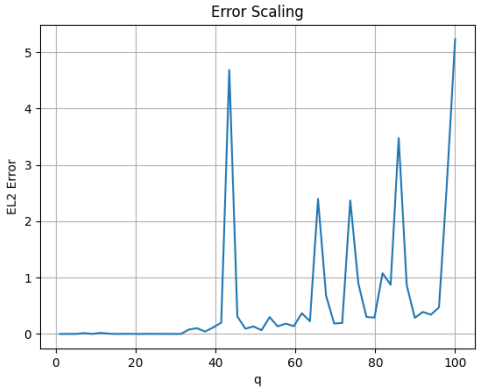


*Figure 3: PINN Error*

This exploration of PINN has demonstrated that it is possible to use PINN to speed up simulations. This is within the error thresholds that conventional methods have and allows for significantly more data points to be found, with no limitation on the domain. This analysis used PINN for a full simulation replacement, but we can also use these methods for dimensionality reduction in simulations to not fully replace simulations but reduce the computational size to more acceptable levels. This is becoming more and more common in research circles, especially in the field of fluid dynamics. This meshing of PINN and simulations could be a great future project.