Multi-level Neural Networks for Accurate Solutions of Boundary-Value Problems

Ziad Aldirany

Département de Mathématiques et de Génie Industriel
Polytechnique Montréal

with Régis Cottereau (CNRS, Marseille), Marc Laforest (Poly Montréal) and Serge Prudhomme (Poly Montréal)

Deep Neural Networks

Let us consider a feedforward neural network consisting of n hidden layers, each layer being of width N_i. The DNN with input $\boldsymbol{z}_0$ and output $\boldsymbol{z}_{n+1}$ is defined as

$$\boxed{\begin{array}{ll} \text{Input layer:} & \boldsymbol{z}_0, \\ \text{Hidden layers:} & \boldsymbol{z}_i = \sigma(W_i \boldsymbol{z}_{i-1} + \boldsymbol{b}_i), \quad i = 1, \cdots, n, \\ \text{Output layer:} & \boldsymbol{z}_{n+1} = W_{n+1} \boldsymbol{z}_n + \boldsymbol{b}_{n+1}, \end{array}}$$

where

$$\begin{aligned} \sigma &= \text{given activation function}, \\ \boldsymbol{W}_i &= \text{weights matrix with size } N_i \times N_{i-1}, \\ \boldsymbol{b}_i &= \text{biases vector with size } N_i. \end{aligned}$$

PINNs [Raissi et al.(2019)]

Consider the linear PDE in its residual form:

$$R\big(\boldsymbol{x}, u(\boldsymbol{x})\big) := f(\boldsymbol{x}) - Au(\boldsymbol{x}) = 0, \quad \forall \boldsymbol{x} \in \Omega,$$

with the zero Dirichlet boundary conditions

$$B\big(\boldsymbol{x}, u(\boldsymbol{x})\big) := u(\boldsymbol{x}) = 0, \quad \forall \boldsymbol{x} \in \partial\Omega.$$

Defining a function $g(\boldsymbol{x})$ that vanishes on the boundary, we approximate u with

$$\tilde{u} = \tilde{u}_\theta = g(\boldsymbol{x}) z_{n+1},$$

by minimizing the **loss function**

$$\boxed{\mathcal{L}(\theta) := \int_\Omega R\big(\boldsymbol{x}, \tilde{u}(\boldsymbol{x})\big)^2 dx.}$$

Model Problem

We will use the one dimensional Poisson problem to back our findings. Our goal is to find $u(x)$ that satisfies

$$\boxed{\begin{aligned} -\partial_{xx} u(x) &= f(x), \qquad \forall x \in (0,1), \\ u(0) &= 0, \\ u(1) &= 0. \end{aligned}}$$

The source term f is chosen such that the exact solution is

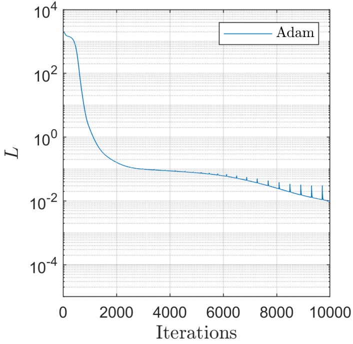
$$u(x) = e^{\sin(k\pi x)} + x^3 - x - 1,$$

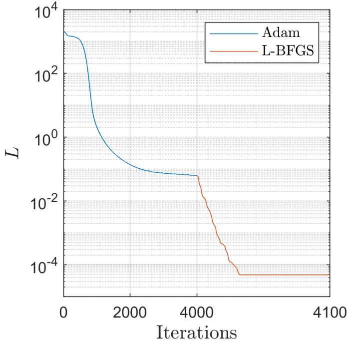
where k is a given integer.

Optimization Algorithm

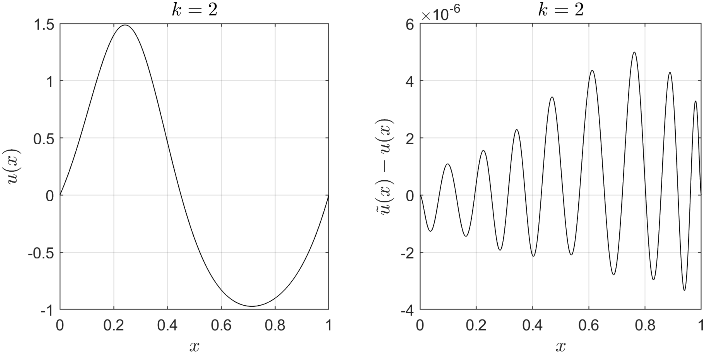
Solving Poisson problem for $k = 2$ using:

- ▷ Adam
- ▷ Adam followed by L-BFGS





Error Analysis



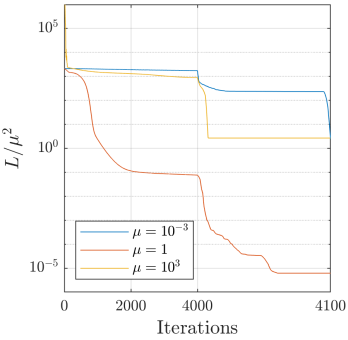
If we want to approximate the remaining error with a new neural network, two issues arise:

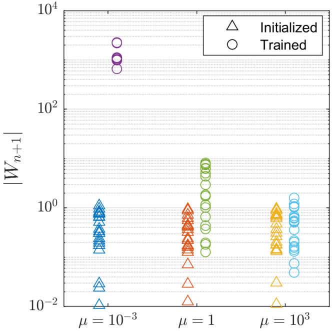
- the error is **not normalized**
- the error exhibits **higher frequencies**

Normalization

Poisson problem with the exact solution:

$$u(x) = \frac{1}{\mu}\left(e^{\sin(2\pi x)} + x^3 - x - 1\right)$$





High Frequencies

We consider again the Poisson problem on $\Omega = (0,1)$, with $k = 10$.

To approximate high frequencies we present the Fourier feature mapping:

$$\gamma(x) = [\cos(\boldsymbol{\omega}_M x), \sin(\boldsymbol{\omega}_M x)], \qquad \gamma_g(x) = [\sin(\boldsymbol{\omega}_M x)].$$

with

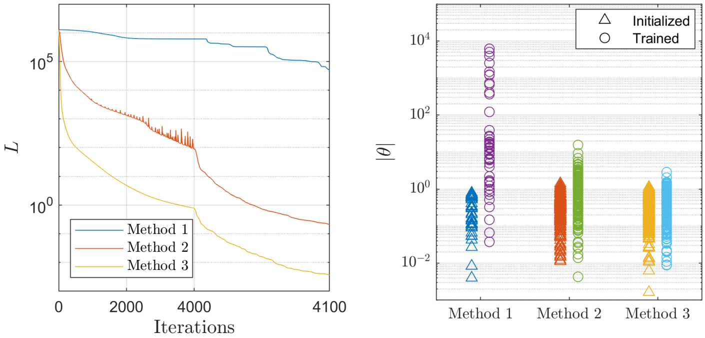
$$\boldsymbol{\omega}_M = [2^0\pi, \dots, 2^{M-1}\pi]$$

	Method 1	**Method 2**	**Method 3**
$\boldsymbol{z}_0$	x	$\gamma(x)$	$\gamma(x)$
$\tilde{u}$	$z_{n+1}x(1-x)$	$z_{n+1}x(1-x)$	$M^{-1}\gamma_g(x) \cdot \boldsymbol{z}_{n+1}$

High Frequencies

Poisson problem with the exact solution:

$$u(x) = e^{\sin(10\pi x)} + x^3 - x - 1$$



Correction Neural Network

Consider an initial solution $\tilde{u}_0$ to the boundary value problem.

We define the error in $\tilde{u}_0$ as $e(\boldsymbol{x}) = u(\boldsymbol{x}) - \tilde{u}_0(\boldsymbol{x})$ and satisfies:

$$\begin{aligned} R(\boldsymbol{x}, \tilde{u}_0(\boldsymbol{x})) - Ae(\boldsymbol{x}) &= 0, \quad \forall \boldsymbol{x} \in \Omega, \\ B(\boldsymbol{x}, e(\boldsymbol{x})) &= 0, \quad \forall \boldsymbol{x} \in \partial\Omega. \end{aligned}$$

In order to normalize the solution we will modify the problem to:

$$\begin{aligned} \tilde{R}(\boldsymbol{x}, \tilde{e}(\boldsymbol{x})) = \mu R(\boldsymbol{x}, \tilde{u}_0(\boldsymbol{x})) - A\tilde{e}(\boldsymbol{x}) &= 0, \quad \forall \boldsymbol{x} \in \Omega, \\ B(\boldsymbol{x}, \tilde{e}(\boldsymbol{x})) &= 0, \quad \forall \boldsymbol{x} \in \partial\Omega. \end{aligned}$$

The corrected solution is given as:

$$\boxed{\tilde{u}(\boldsymbol{x}) = \tilde{u}_0(\boldsymbol{x}) + \frac{1}{\mu}\tilde{e}(\boldsymbol{x}).}$$

Multi-level Neural Networks

Considering the first normalized approximation verifying:

$$\begin{aligned} R_0(\boldsymbol{x}, u_0(\boldsymbol{x})) = \mu_0 f(\boldsymbol{x}) - A u_0(\boldsymbol{x}) &= 0, \quad \forall \boldsymbol{x} \in \Omega, \\ B(\boldsymbol{x}, u_0(\boldsymbol{x})) &= 0, \quad \forall \boldsymbol{x} \in \partial\Omega. \end{aligned}$$

Each new correction u_i then satisfies the boundary-value problem:

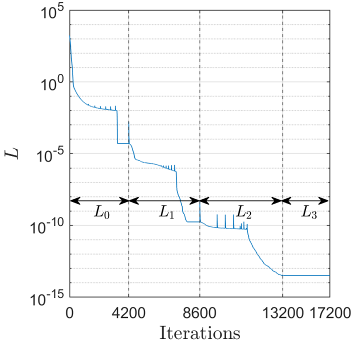
$$\begin{aligned} R_i(\boldsymbol{x}, u_i(\boldsymbol{x})) = \mu_i R_{i-1}(\boldsymbol{x}, \tilde{u}_{i-1}(\boldsymbol{x})) - A u_i(\boldsymbol{x}) &= 0, \quad \forall \boldsymbol{x} \in \Omega, \\ B(\boldsymbol{x}, u_i(\boldsymbol{x})) &= 0, \quad \forall \boldsymbol{x} \in \partial\Omega. \end{aligned}$$

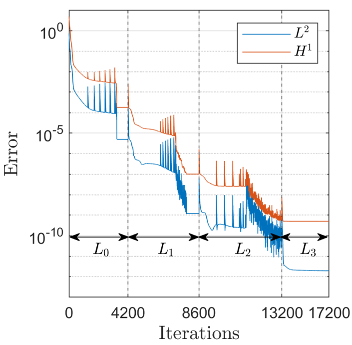
The final approximation $\tilde{u}$ of u is given as:

$$\boxed{\tilde{u}(\boldsymbol{x}) = \frac{1}{\mu_0}\tilde{u}_0(\boldsymbol{x}) + \frac{1}{\mu_0\mu_1}\tilde{u}_1(\boldsymbol{x}) + \ldots + \frac{1}{\mu_0\mu_1\ldots\mu_L}\tilde{u}_L(\boldsymbol{x}).}$$

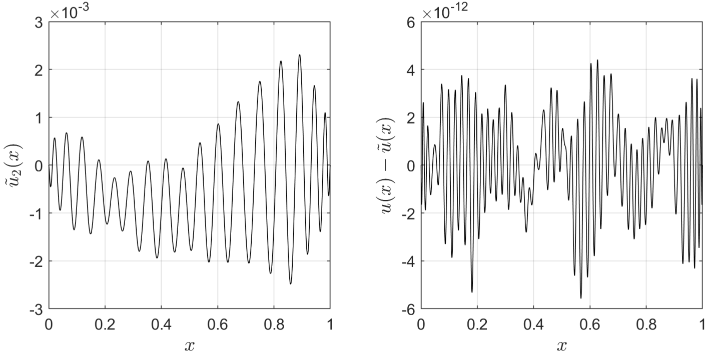
MLNNs Example

	$\tilde{u}_0$	$\tilde{u}_1$	$\tilde{u}_2$	$\tilde{u}_3$
Wave numbers M	1	3	5	1
Normalization μ_i	1	10^3	10^3	10^2





MLNNs Example



- The error is reduced to the order of 10^{-12} after three corrections
- Even after normalizing the amplitude of $\tilde{u}_2(x)$ is small

Extreme Learning Method

To suitably choose the normalizing factors μ_i we will:

- Calculate a coarse prediction of the remaining error using the **Extreme Learning Method**
- Choose μ_i using the amplitude of the estimated error

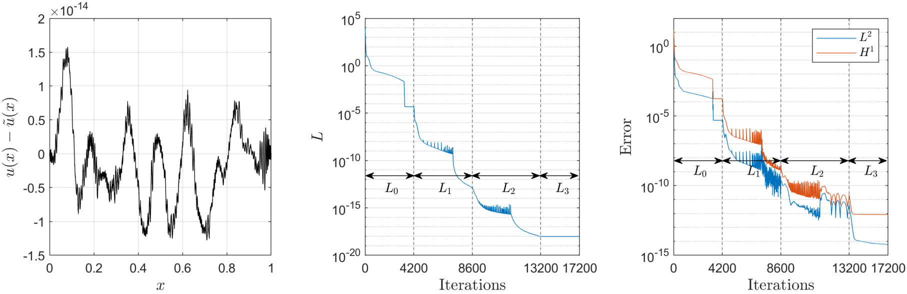
In the **Extreme Learning Method** the solution is approximated with a neural network by:

- Fixing the parameters of the hidden layers
- Minimizing the loss function with respect to the output layer parameters using a least square method

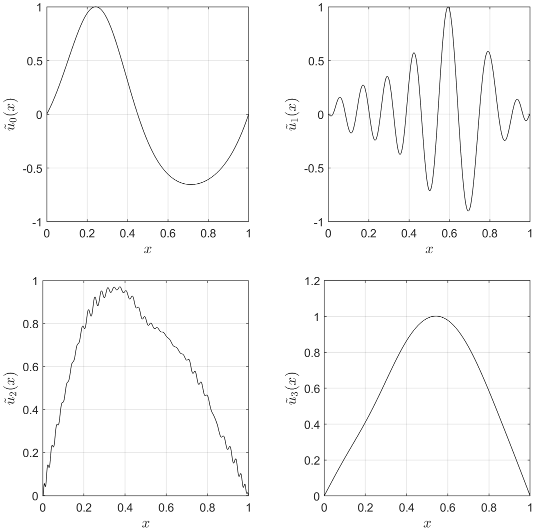
We choose the **Extreme Learning Method** because it is **fast** and **scale independent.**

MLNNs Example with ELM

	$\tilde{u}_0$	$\tilde{u}_1$	$\tilde{u}_2$	$\tilde{u}_3$
Wave numbers M	1	3	5	1



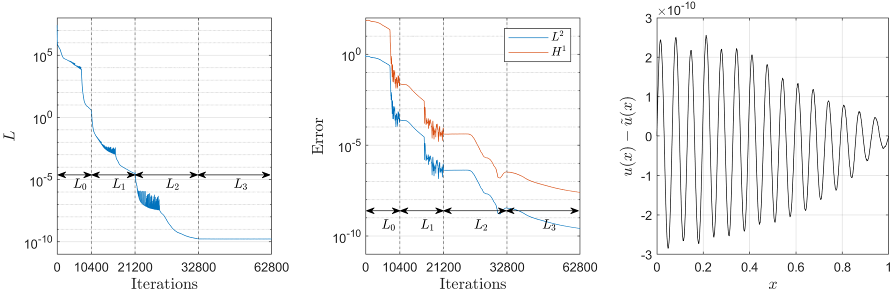
MLNNs Example with ELM



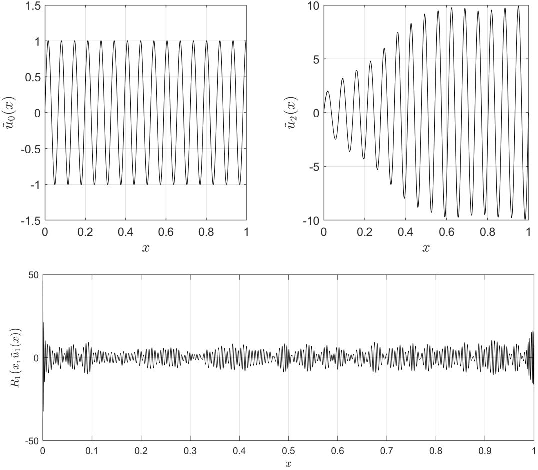
Helmholtz Equation

One dimensional Helmholtz equation

$$-\partial_{xx}u(x) - 9200u(x) = 0, \qquad \forall x \in (0,1),$$
$$u(0) = 0, \qquad u(1) = 1.$$



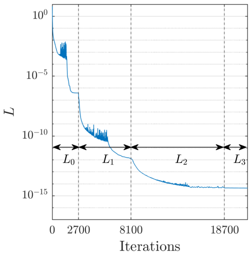
Helmholtz Equation

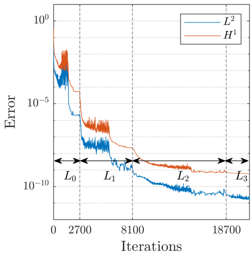


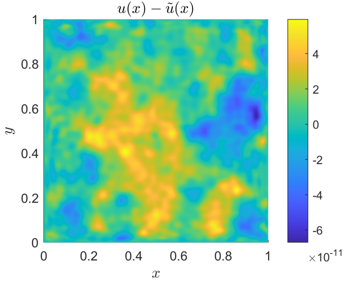
2D Poisson Equation

Two dimensional Poisson problem

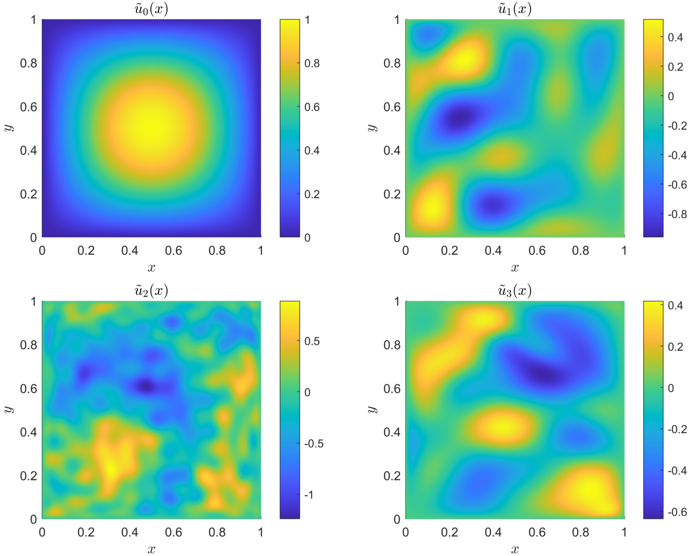
$$\begin{aligned} -\nabla^2 u(x,y) &= f(x,y), && \forall x \in \Omega, \\ u(x,y) &= 0, && \forall x \in \partial\Omega. \end{aligned}$$







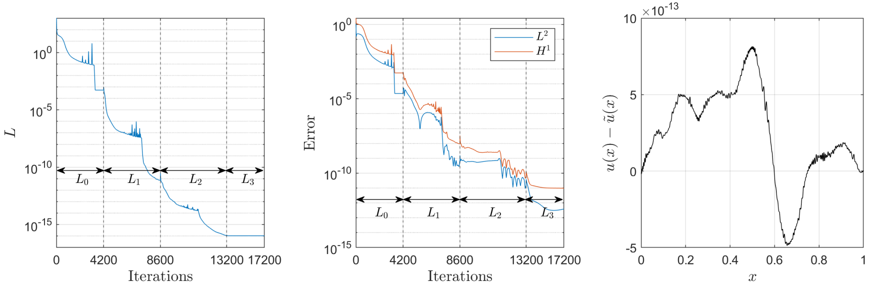
2D Poisson Equation



Non-linear problem

Non-linear boundary value problem

$$
\begin{aligned}
&-\partial_{xx}u(x) - 8u(x)\partial_x u(x) = 0, \qquad \forall x \in (0,1),\\
&u(0) = -1, \qquad u(1) = -1/5.
\end{aligned}
$$



Summary

- Introduced the multi-level neural networks to control and reduce the errors for linear BVPs when using deep learning approaches
- Addressed the issues of low amplitudes and high frequencies to correct the resulting errors
- Exhibited the potential of MLNNs to significantly reduce L^2 and H^1 errors for various BVPs, achieving machine precision in some cases
- For future work, we aim to extend MLNNs to other deep learning approaches and propose automated hyper-parameter selection for different levels

Thank you!

For further details:
Aldirany, Z., Cottereau, R., Laforest, M., Prudhomme, S. (2024). Multi-level Neural Networks for Accurate Solutions of Boundary-Value Problems. *Comp. Meth. Appl. Mech. Eng.* 419(116666).
arXiv:2308.11503.