Computational Methods for Accurate Approximations of Peridynamic Models

Serge Prudhomme

Département de Mathématiques et de génie industriel
Polytechnique Montréal

with P. Diehl (LSU), C. Bilodeau, R. Flachaire, Z. Aldirany,
M. Laforest (Poly Montréal), and R. Cottereau (CNRS, France)

Workshop on Experimental and Computational Fracture Mechanics
Baton Rouge, LA, USA

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

March 3-6, 2024

NSERC CRSNG

Outline

- High-order integration methods for peridynamic simulations
- Coupling methods on matching and non-matching grids
- Multi-level neural networks for PINNS
- Numerical examples
- Concluding remarks

Acknowledgement:

- NSERC Discovery Grant

High-order quadrature rules for peridynamics in 2D

Linearized microelastic model:

$$\boxed{-\int_{H_\delta(\boldsymbol{x})} \kappa \frac{(\boldsymbol{y}-\boldsymbol{x}) \otimes (\boldsymbol{y}-\boldsymbol{x})}{\|\boldsymbol{y}-\boldsymbol{x}\|^3} \big(\boldsymbol{u}(\boldsymbol{y}) - \boldsymbol{u}(\boldsymbol{x})\big)\, dy = \boldsymbol{f}_b(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \Omega_\delta}$$

By change of variable $\boldsymbol{\xi} = \boldsymbol{y} - \boldsymbol{x}$, above integral becomes:

$$\int_{H_\delta(\mathbf{0})} \kappa \frac{\boldsymbol{\xi} \otimes \boldsymbol{\xi}}{\|\boldsymbol{\xi}\|^3} \big(\boldsymbol{u}(\boldsymbol{x}+\boldsymbol{\xi}) - \boldsymbol{u}(\boldsymbol{x})\big)\, d\xi$$

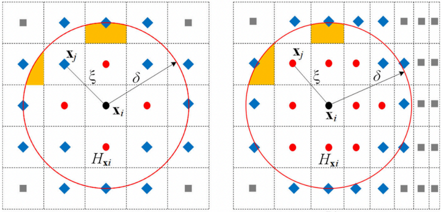
In other words, one needs to evaluate integrals of the form:

$$\boxed{\mathcal{I}(f_k) = \int_{H_\delta(\mathbf{0})} f_k(\boldsymbol{\xi})\, d\xi}$$

where $f_k(\boldsymbol{\xi})$, $k = 1, 2, 3$, are the components of vector-valued integrand.

Volume correction methods

- ▷ FA (Silling and Askari, Comput. Struct., 2005)
- ▷ LAMMPS (Parks et al., Comput. Phys. Comm., 2008)
- ▷ IPA-3 (Bobaru et al., Technical Report, 2010)
- ▷ QWJ (Le et al., IJNME, 2014)
- ▷ PA-AC (Seleson, CMAME, 2014)
- ▷ RHL (Ren et al., CMAME, 2017)
- ▷ NT (Ni et al., Eng Fract Mech, 2018)
- ▷ Zheng et al., Int J Fract (2021)
- ▷ Etc.



From Zheng et al., 2021

Integration in 2D

Integrand in 2D reads:

$$\boldsymbol{f}(\boldsymbol{\xi}) = \begin{bmatrix} f_1(\boldsymbol{\xi}) \\ f_2(\boldsymbol{\xi}) \end{bmatrix} = \frac{\kappa}{\|\boldsymbol{\xi}\|^3} \begin{bmatrix} \xi^2 \left[u(\boldsymbol{x}+\boldsymbol{\xi}) - u(\boldsymbol{x})\right] + \xi\eta \left[v(\boldsymbol{x}+\boldsymbol{\xi}) - v(\boldsymbol{x})\right] \\ \xi\eta \left[u(\boldsymbol{x}+\boldsymbol{\xi}) - u(\boldsymbol{x})\right] + \eta^2 \left[v(\boldsymbol{x}+\boldsymbol{\xi}) - v(\boldsymbol{x})\right] \end{bmatrix}$$

Using Taylor expansion and polar coordinates, i.e. $\boldsymbol{\xi} = (r\cos\theta, r\sin\theta)$, the first component of $\boldsymbol{f}$ reads:

$$\begin{aligned} f_1(\xi,\eta) = \kappa\Bigg(& \cos^3(\theta)\frac{\partial u}{\partial x} + \cos^2(\theta)\sin(\theta)\left[\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right] + \cos(\theta)\sin^2(\theta)\frac{\partial v}{\partial y} \\ & + \frac{\xi\cos^3(\theta)}{2}\frac{\partial^2 u}{\partial x^2} + \frac{\xi\cos^2(\theta)\sin(\theta)}{2}\left[\frac{\partial^2 u}{\partial x\partial y} + \frac{\partial^2 v}{\partial x^2}\right] \\ & + \frac{\xi\cos(\theta)\sin^2(\theta)}{2}\left[\frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 v}{\partial x\partial y}\right] + \frac{\xi\sin^3(\theta)}{2}\frac{\partial^2 u}{\partial y^2} + \dots\Bigg) \end{aligned}$$

Integration in 2D

To integrate f_1 and f_2, we want to exactly integrate the functions of the form:

$$\boxed{f_{\nu,p,q}(\xi,\eta) = \cos^{3-\nu}(\theta)\sin^{\nu}(\theta)\,\xi^p\eta^q, \quad \nu = 0,1,2,3, \quad p,q = 0,1,\ldots}$$

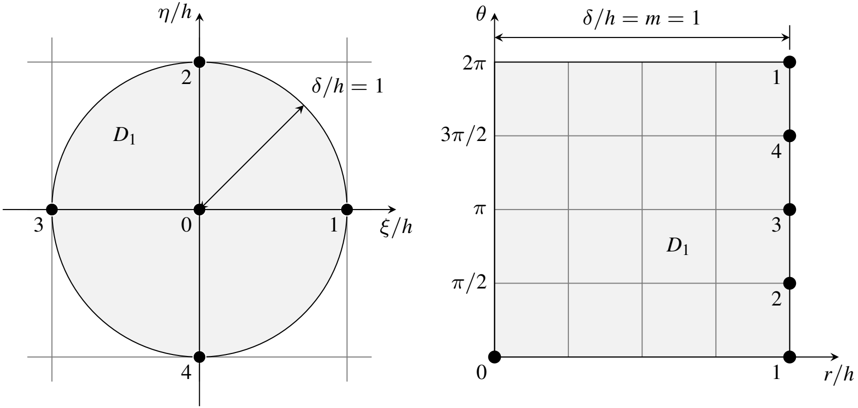
We also develop integration rules for polynomial functions:

$$\boxed{f_{p,q}(\xi,\eta) = \xi^p\eta^q, \qquad p,q = 0,1,\ldots}$$

Goal:

$$\boxed{\mathcal{I}(f) = \int_{H_\delta(\mathbf{0})} f(\xi,\eta)\,d\xi d\eta}$$

Quadrature rules for $m = \delta/h = 1$



Domain D_1 in Cartesian coordinates (ξ, η) (left) and in polar coordinates (r, θ) (right).

Quadrature rules for $m = \delta/h = 1$

$$\boxed{\mathcal{I}(f) = \int_{D_1} f(\boldsymbol{\xi})\, d\xi \approx \sum_{i=0}^{4} \omega_i f(\boldsymbol{\xi}_i)}$$

Case with $f_{p,q}(\xi,\eta) = \xi^p \eta^q$ with $p, q = 0, 1, \ldots$: equality for $p + q = 0, 1, 2$.

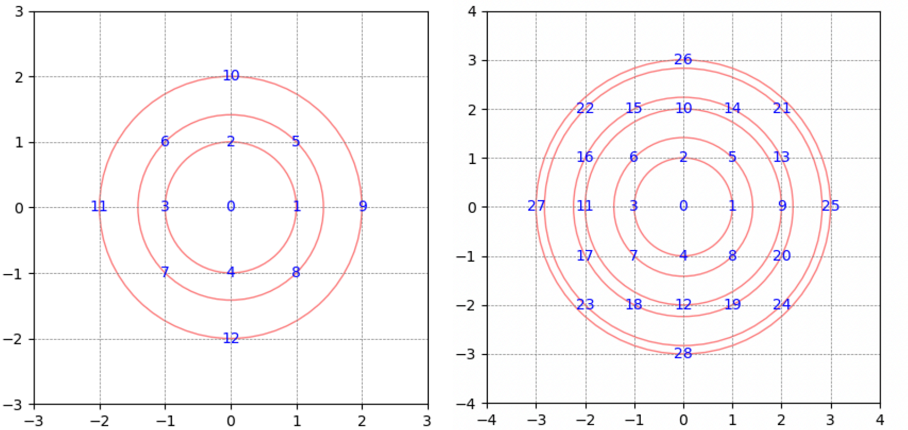
$$\boxed{\int_{D_1} f(\boldsymbol{\xi})\, d\xi = \frac{\pi h^2}{2} f(\mathbf{0}) + \frac{\pi h^2}{2}\left[\frac{f(\boldsymbol{\xi}_1) + f(\boldsymbol{\xi}_2) + f(\boldsymbol{\xi}_3) + f(\boldsymbol{\xi}_4)}{4}\right]}$$

Integration rule is exact for $f(\xi,\eta) = \xi^3$, $\xi^2\eta$, $\xi\eta^2$, and η^3 (odd functions). Therefore, the order of precision is $p + q = 3$.

Case with $f_{\nu,p,q}(\xi,\eta) = \cos^{3-\nu}(\theta)\sin^{\nu}(\theta)\xi^p\eta^q$ with $\nu = 0, 1, 2, 3$ and $p, q = 0, 1, \ldots$: with equality for $p + q = 0, 1$ and $\nu = 0, 1, 2, 3$.

$$\boxed{\int_{D_1} f(\boldsymbol{\xi})\, d\xi = 0\, f(\mathbf{0}) + \frac{\pi h^2}{2}\left[\frac{f(\boldsymbol{\xi}_1) + f(\boldsymbol{\xi}_2) + f(\boldsymbol{\xi}_3) + f(\boldsymbol{\xi}_4)}{4}\right]}$$

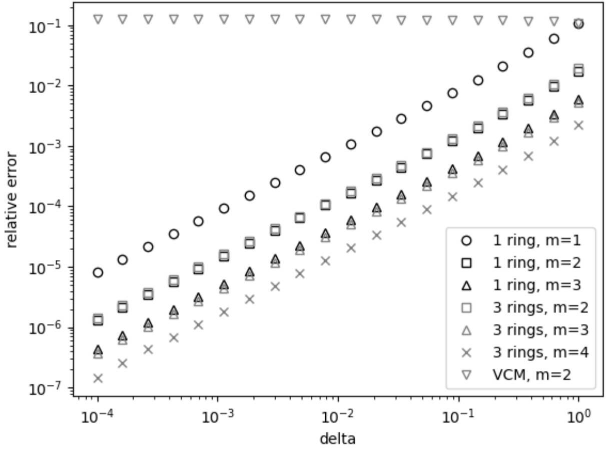
Quadrature rules for $m = \delta/h = 2$ and $m = \delta/h = 3$



Quadrature rules

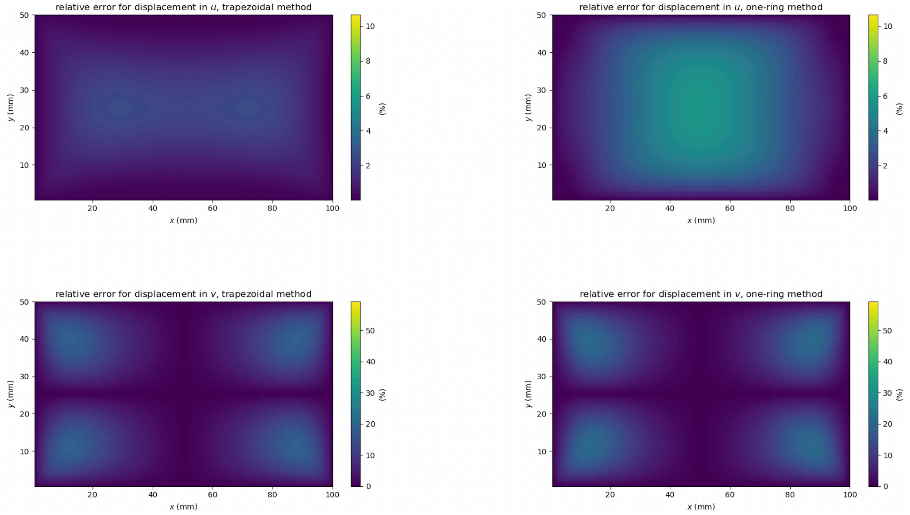
	Polynomial-based integration rules			
n_r \ δ	h $(m=1)$	$2h$ $(m=2)$	$3h$ $(m=3)$	degree
1	$\frac{\pi\delta^2}{2}\{1,1\}$	$\frac{\pi\delta^2}{8}\{1,2,3,2\}$	$\frac{\pi\delta^2}{18}\{1,2,3,3,4,4,1\}$	3
3	–	$\frac{\pi\delta^2}{6}\{1,0,4,1\}$	$\frac{\pi\delta^2}{864}\{64,256,-63,445,55,107\}$	5
6	–	–	$\frac{\pi\delta^2}{40960}\{13685,-29340,28800,22482,-6660,8505,3488\}$	7

Numerical examples

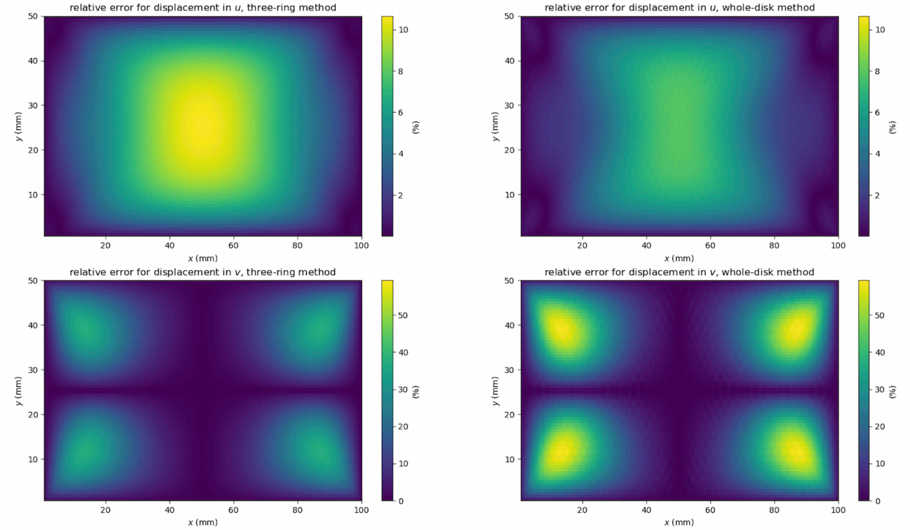


$$f(\xi, \eta) = (1 - \exp(-r))/r$$

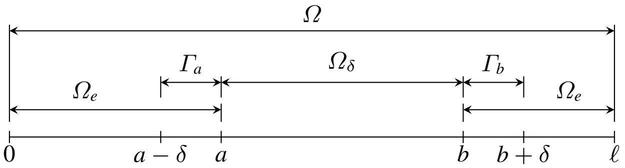
Numerical examples, $m = 3$



Numerical examples, $m = 3$



Coupling Methods



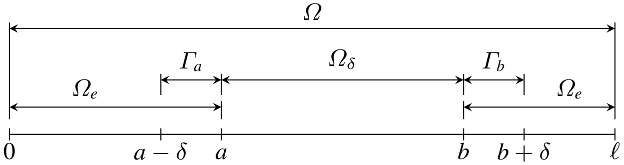
We consider three different approaches:

MDCM = Coupling method with matching displacements
[Zaccariotto and Galvanetto, et al.], [Kilic and Madenci, 2018], [Sun and Fish, 2019], [D'Elia and Bochev, 2021], etc.

MSCM = Coupling method with matching stresses
[Silling, Sandia Report, 2020]

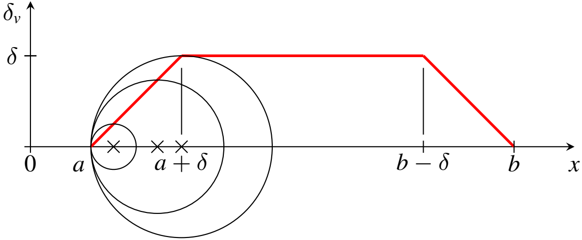
VHCM = Coupling method with variable horizon
[S. Silling et al., 2015], [Nikpayam and Kouchakzadeh, 2019]

MDCM formulation



$$
\boxed{
\begin{aligned}
-E\underline{u}''(x) &= f_b(x), \quad && \forall x \in \Omega_e \\
-\int_{x-\delta}^{x+\delta} \kappa \frac{u(y)-u(x)}{|y-x|} dy &= f_b(x), \quad && \forall x \in \overline{\Omega}_\delta \\
\underline{u}(x) &= 0, \quad && \text{at } x = 0 \\
E\underline{u}'(x) &= g, \quad && \text{at } x = \ell \\
u(x) - \underline{u}(x) &= 0, \quad && \forall x \in \overline{\Gamma_a \cup \Gamma_b}
\end{aligned}
}
$$

VHCM formulation



Variable horizon function:

$$\delta_v(x) = \begin{cases} x-a, & a < x \leq a+\delta \\ \delta, & a+\delta < x \leq b-\delta \\ b-x, & b-\delta < x < b \end{cases}$$

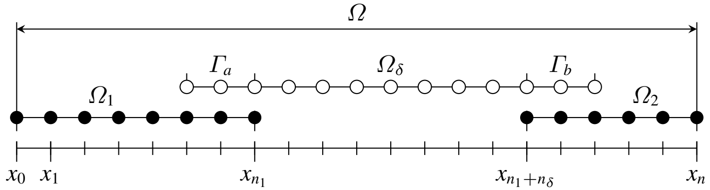
$$\boxed{\bar{\kappa}(x)\delta_v^2(x) = \kappa\delta^2, \quad \forall x \in \Omega_\delta}$$

VHCM formulation

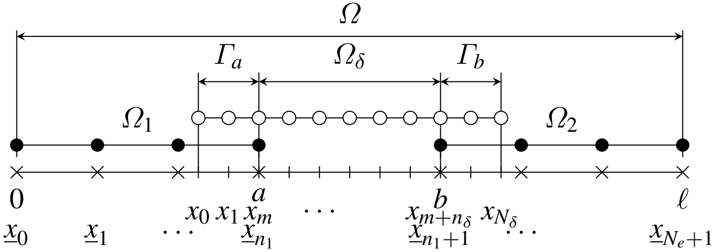
$$
\boxed{
\begin{aligned}
-E\underline{u}''(x) &= f_b(x), \quad && \forall x \in \Omega_e \\
-\int_{x-\delta_v(x)}^{x+\delta_v(x)} \bar{\kappa}(x) \frac{u(y)-u(x)}{|y-x|} dy &= f_b(x), \quad && \forall x \in \Omega_\delta \\
\underline{u}(x) &= 0, \quad && \text{at } x = 0 \\
E\underline{u}'(x) &= g, \quad && \text{at } x = \ell \\
u(x) - \underline{u}(x) &= 0, \quad && \text{at } x = a, b \\
\sigma^+(u)(x) - E\underline{u}'(x) &= 0, \quad && \text{at } x = a \\
\sigma^-(u)(x) - E\underline{u}'(x) &= 0, \quad && \text{at } x = b
\end{aligned}
}
$$

Discretization

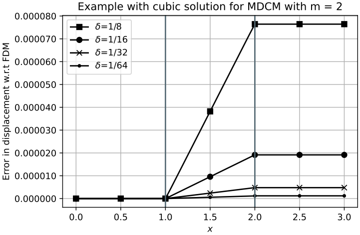
Matching grids

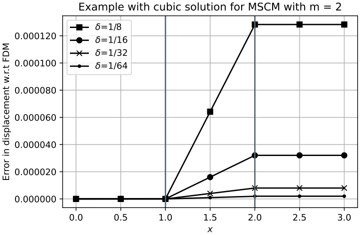


Non-matching grids

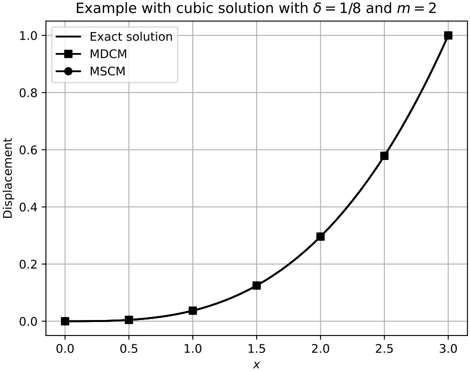


Example with cubic solution





Interpolant of degree 2



Interpolant of degree 3

$\delta = 1/8$, $m = 2$, $h_\delta = \delta/m = 1/16$, and $h_e = \delta = 1/8$.

Neural Networks

A feedforward neural network (FNN), consisting of n hidden layers, each layer being of width N_i, with input z_0 and output z_{n+1} is defined as

$$\boxed{\begin{array}{ll} \text{Input layer:} & z_0, \\ \text{Hidden layers:} & z_i = \sigma(W_i\, z_{i-1} + \boldsymbol{b}_i), \quad i = 1, \cdots, n, \\ \text{Output layer:} & z_{n+1} = W_{n+1}\, z_n + \boldsymbol{b}_{n+1}, \end{array}}$$

where

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

We shall denote by θ the parameters $(\boldsymbol{W}_i, \boldsymbol{b}_i)_{i=1,\ldots,N+1}$ of the NN.

PINNs [Raissi et al. (2019)]

Consider a linear PDE in its residual form with homogeneous Dirichlet BCs:

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

The BCs can be strongly prescribed by considering a function $g(\boldsymbol{x})$ that vanishes on the boundary, such that the solution u of the problem is approximated by a NN as:

$$u(\boldsymbol{x}) \approx \tilde{u}(\boldsymbol{x}) = \tilde{u}_\theta(\boldsymbol{x}) = g(\boldsymbol{x}) z_{n+1}(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \Omega$$

by minimizing the **loss function**

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

What is the error $\boxed{e = u - \tilde{u}}$? This is a solution verification issue.

Model Problem

We use the simple 1D Poisson problem to illustrate the main observations.

The problem is to find $u(x)$ that satisfies

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

Manufactured solution:

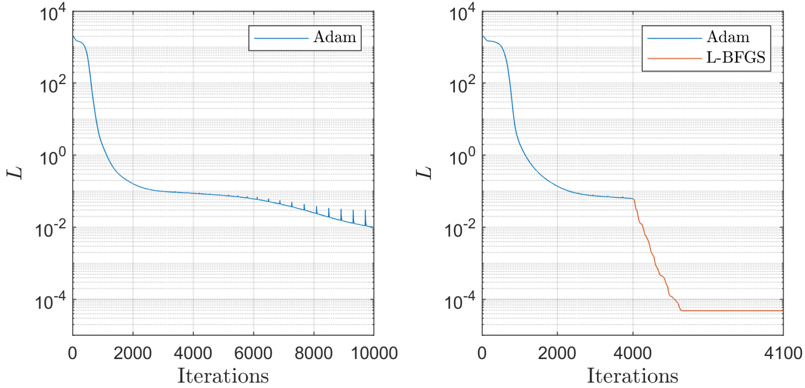
$$\boxed{u(x) = e^{\sin(k\pi x)} + x^3 - x - 1}$$

where k is a given integer.

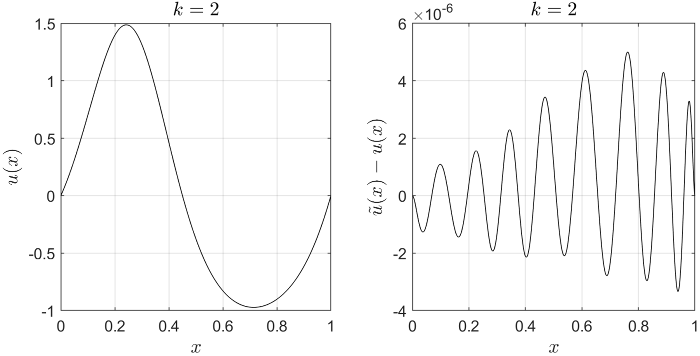
Optimization Algorithm

Solving the Poisson problem for $k = 2$ using:

- ▷ Adam (stochastic gradient descent method with variable learning rate)
- ▷ Adam followed by L-BFGS (quasi-Newton using estimate of Hessian)



Error Analysis



If one wants to estimate the approximation error $e = u - \tilde{u}$ with a new neural network, two issues arise:

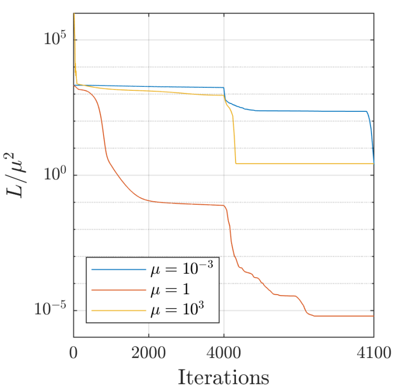
- the error is **very small**
- the error exhibits **high frequencies**

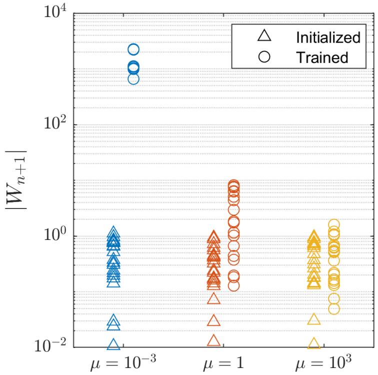
Normalization

Poisson problem with manufactured solution ($k = 2$):

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

One hidden layer with $N_1 = 20$.





Solutions with High Frequencies

To approximate high frequencies, we use the Fourier feature mapping:

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

with

$$\boldsymbol{\omega}_M = (2^0\pi, \ldots, 2^{M-1}\pi)$$

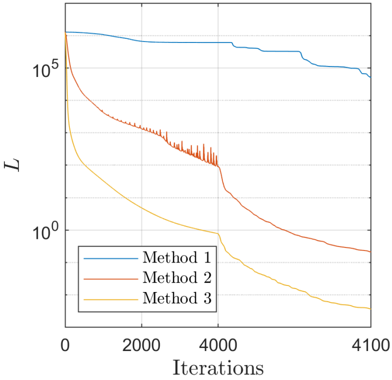
	Method 1	**Method 2**	**Method 3**
z_0	x	$\gamma(x)$	$\gamma(x)$
$\tilde{u}$	$z_{n+1}x(1-x)$	$z_{n+1}x(1-x)$	$M^{-1}\gamma_g(x) \cdot z_{n+1}$

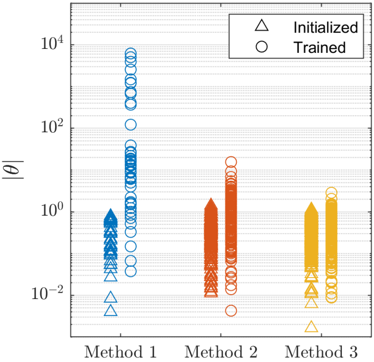
Solutions with High Frequencies

Poisson problem with manufactured solution ($k = 10$):

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

One hidden layer with $N_1 = 10$, $M = 4$, $\omega_M = [\pi, 2\pi, 4\pi, 8\pi]$.





Correction Approximation

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

The error $e = u - \tilde{u}_0$ in approximation $\tilde{u}_0$ satisfies:

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

For normalization, we need to modify the problem to:

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

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:

$$\begin{aligned} \mathcal{R}_0(\boldsymbol{x}, u_0(\boldsymbol{x})) = \mu_0 f(\boldsymbol{x}) - A u_0(\boldsymbol{x}) &= 0, \quad \forall \boldsymbol{x} \in \Omega \\ \mathcal{B}(\boldsymbol{x}, u_0(\boldsymbol{x})) &= 0, \quad \forall \boldsymbol{x} \in \partial\Omega \end{aligned}$$

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

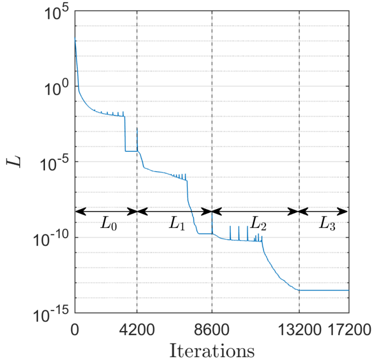
$$\begin{aligned} \mathcal{R}_i(\boldsymbol{x}, u_i(\boldsymbol{x})) = \mu_i \mathcal{R}_{i-1}(\boldsymbol{x}, \tilde{u}_{i-1}(\boldsymbol{x})) - A u_i(\boldsymbol{x}) &= 0, \quad \forall \boldsymbol{x} \in \Omega \\ \mathcal{B}(\boldsymbol{x}, u_i(\boldsymbol{x})) &= 0, \quad \forall \boldsymbol{x}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

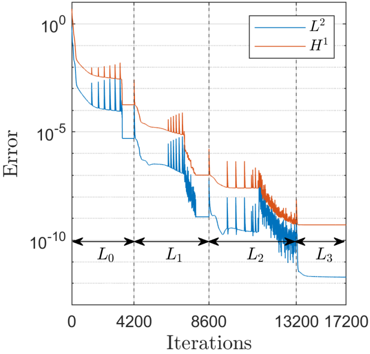
The final approximation $\tilde{u}$ of u is given as:

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

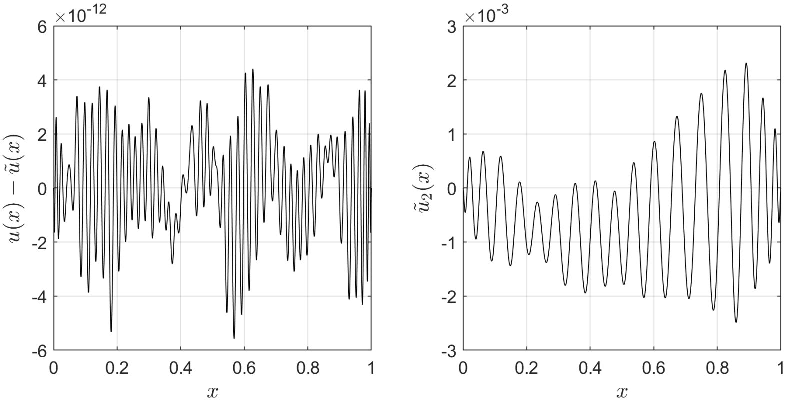
MLNNs Example

	$\tilde{u}_0$	$\tilde{u}_1$	$\tilde{u}_2$	$\tilde{u}_3$
Layer width N_1	10	20	40	20
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
- ▷ The amplitude of $\tilde{u}_2(x)$ is small using a priori normalization

Estimation of normalization factors

To suitably choose the normalizing factors μ_i:

- ▷ Calculate a coarse prediction of the error using the **Extreme Learning Method** (ELM)
- ▷ Choose μ_i using the amplitude of the estimated error

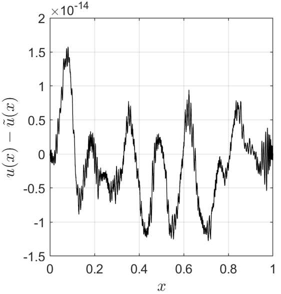
In the **Extreme Learning Method**, the solution is approximated with a neural network by:

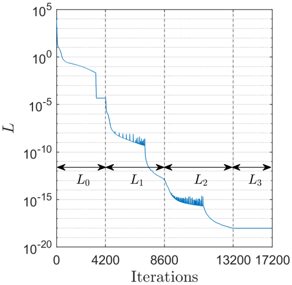
- ▷ Fixing the parameters of the hidden layers
- ▷ Minimizing the loss function with respect to the output layer parameters using a least square method

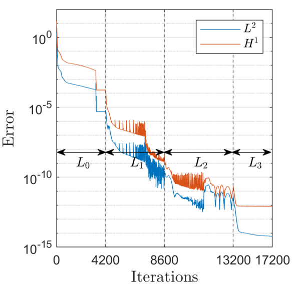
We choose the **Extreme Learning Method** because it is **fast** and **scale independent.**

MLNNs Example with ELM

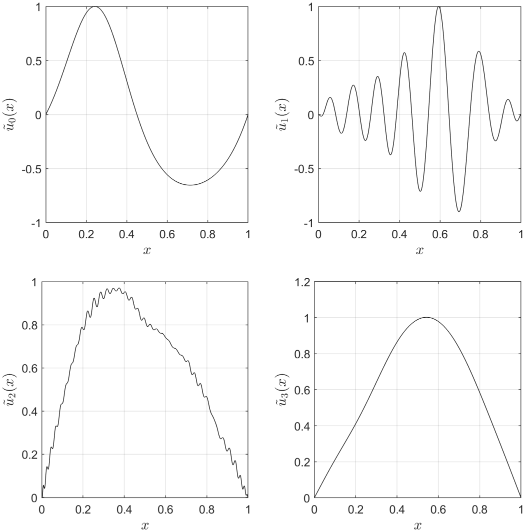
	$\tilde{u}_0$	$\tilde{u}_1$	$\tilde{u}_2$	$\tilde{u}_3$
Layer width N_1	10	20	40	20
Wave numbers M	1	3	5	1



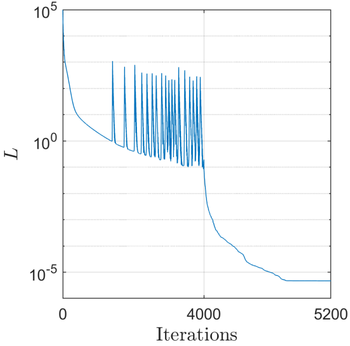


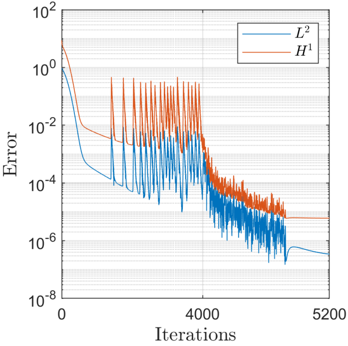


MLNNs Example with ELM



Neural network with a single hidden layer of width 60



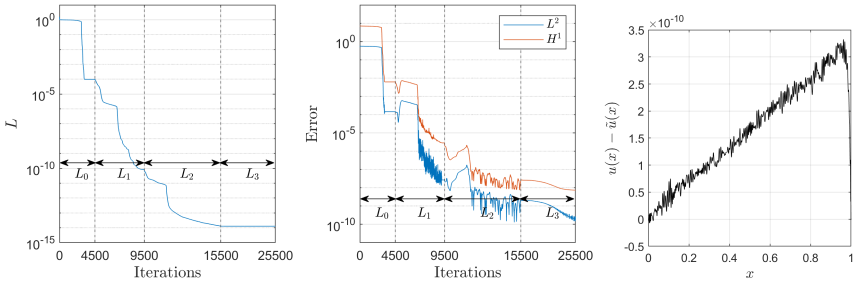


The number of parameters of the network is 961.

Convection-diffusion Equation

$$\boxed{\begin{aligned} -0.01u''(x) + u'(x) &= 1, \quad \forall x \in (0,1) \\ u(0) &= 0, \\ u(1) &= 0, \end{aligned}}$$

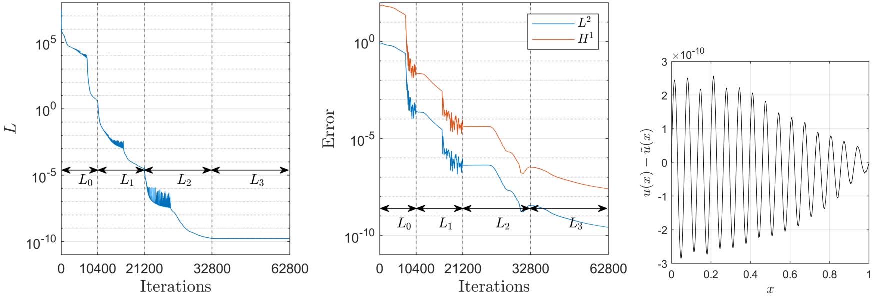
One hidden layer with $N_1 = 5, 10, 20, 20$ and $M = 3, 5, 7, 3$.



1D Helmholtz Equation

$$\boxed{\begin{aligned} -u''(x) - 9200u(x) &= 0, \quad \forall x \in (0,1) \\ u(0) &= 0, \\ u(1) &= 1, \end{aligned}}$$

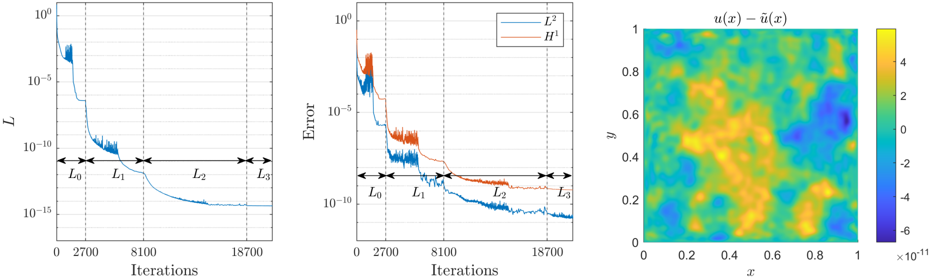
One hidden layer with $N_1 = 10, 20, 40, 10$ and $M = 5, 7, 9, 5$.



2D Poisson Equation

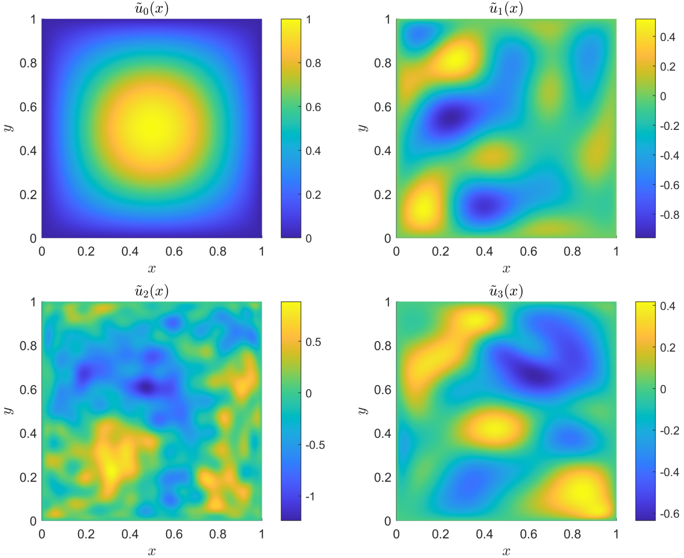
$$\boxed{\begin{aligned} -\Delta u(x,y) &= f(x,y), \quad \forall x \in \Omega \\ u(x,y) &= 0, \quad \forall x \in \partial\Omega \end{aligned}}$$

Two hidden layers with $N_1 = N_2 = 10, 20, 40, 40$ and $M = 1, 3, 5, 1$.



Pointwise errors are reduced to values of the order of 10^{-11}

2D Poisson Equation



Nonlinear Example

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

Loss function:

$$\boxed{\mathcal{L}(\theta) = w_r \int_\Omega \mathcal{N}\big(\boldsymbol{x}, \tilde{u}(\boldsymbol{x})\big)^2 dx + w_{bc} \int_{\partial\Omega} \mathcal{B}\big(\boldsymbol{x}, \tilde{u}(\boldsymbol{x})\big)^2 dx}$$

Normalization and high frequency issues arise as well for nonlinear problems. We seek the solution using MLNN in the form:

$$\tilde{u}(\boldsymbol{x}) = \sum_{k=0}^{i} \frac{1}{\Pi_{j=0}^{k} \mu_j} \tilde{u}_k(\boldsymbol{x})$$

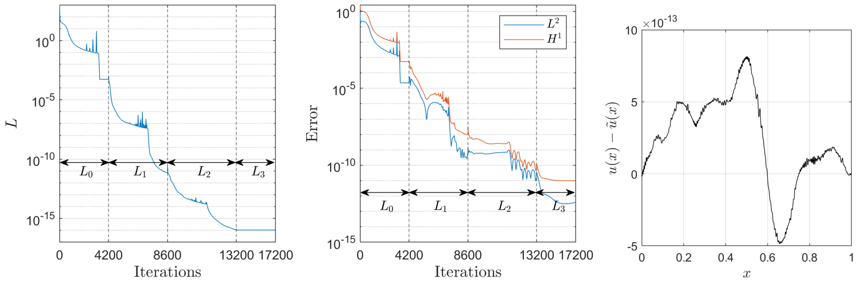
Extreme Learning method applied to the linearized equation:

$$\delta_u \mathcal{N}(\boldsymbol{x}, \tilde{u})(\tilde{e}) = -\mathcal{N}(\boldsymbol{x}, \tilde{u})$$

Burgers' Equation

$$\boxed{\begin{aligned} \mathcal{N}(x,u(x)) = u''(x) - 8u(x)u'(x) = 0, &\quad \forall x \in (0,1) \\ u(0) = -1, & \\ u(1) = -0.2, & \end{aligned}}$$

One hidden layer with $N_1 = 10, 20, 40, 20$ and $M = 1, 3, 5, 1$.



Conclusions

- High-order integration rules for peridynamic simulations
- Interpolation operators for coupling on non-marching grids.
- Numerical examples showed that MLNNs can reduce the 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 approches and to peridynamic modeling.

References:

- P. Diehl, SP. "Coupling Approaches for Classical Linear Elasticity and Bond- Based Peridynamic Models." Journal of Peridynamics and Nonlocal Modeling, 2022.
- Z. Aldirany, R. Cottereau, M. Laforest, SP. "Multi-level Neural Networks for Accurate Solutions of Boundary-Value Problems." CMAME, 2024.
- C. Bilodeau, P. Diehl, R. Flachaire, SP. "High-order integration rules for peridynamic modeling in one and two dimensions." In preparation, 2024.
- P. Diehl, E. Downing, A. Edwards, SP. "Coupling approaches with non- matching grids for classical linear elasticity and bond-based peridynamic models in 1D." In preparation, 2024.