5 Advanced Physics-Informed Neural Networks

Leon Herrmann

Stefan Kollmannsberger

Chair of Data Engineering in Construction

Bauhaus-Universität Weimar

Deep Learning in Computational Mechanics – an introductory course, Herrmann et al. 2025

website

book

Contents

4 Introduction to Physics-Informed Neural Networks

Physics-informed neural networks make use of parametrized differential equations of the form

$$\mathcal{N}[u(x,t);\lambda(x,t)] = 0, x \in \Omega, t \in \mathcal{T}$$

In data-driven inference, a neural network approximates the solution (to solve the forward problem)

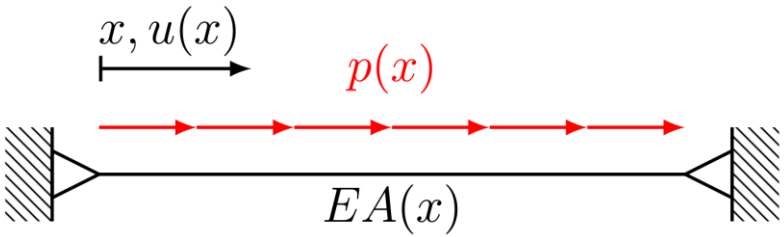
$$\hat{u}(x,t) = u_{NN}(x,t)$$

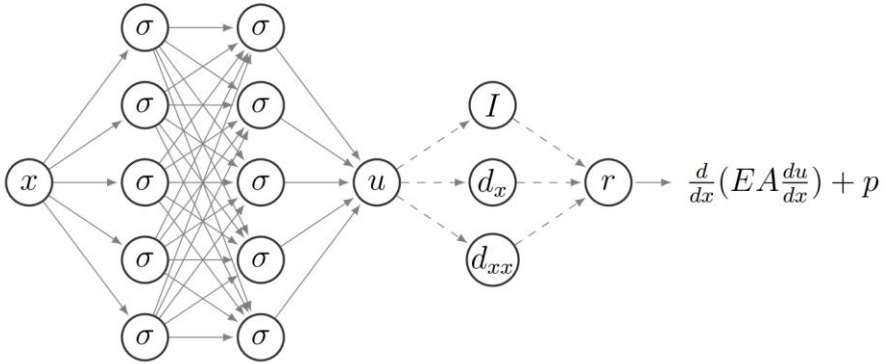
The physics are encoded via the cost function (composed of the residual $\mathcal{L}_R$ and boundary loss $\mathcal{L}_B$)

$$C = \mathcal{L}_R + \mathcal{L}_B$$

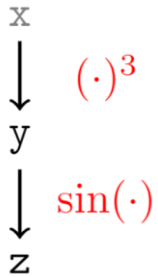
The derivatives in the loss terms $\mathcal{L}_R, \mathcal{L}_B$ are computed with automatic differentiation with respect to the inputs x, t

One-dimensional static problem as example:





5.1 Variations



Computing **higher order derivatives** for the residual loss $\mathcal{L}_R$ is expensive

- Consider the example of

$$y = x^3$$
$$z = \sin(y)$$

- Automatic differentiation relies on the chain rule
- Computation of the first derivative

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}$$

- Computation of the second derivative

$$\frac{\partial^2 z}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\partial z}{\partial x}\right) = \frac{\partial}{\partial x}\left(\frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}\right) = \frac{\partial}{\partial x}\left(\frac{\partial z}{\partial y}\right) \cdot \frac{\partial y}{\partial x} + \frac{\partial z}{\partial y} \cdot \frac{\partial}{\partial x}\left(\frac{\partial y}{\partial x}\right)$$

- Computation of the third derivative

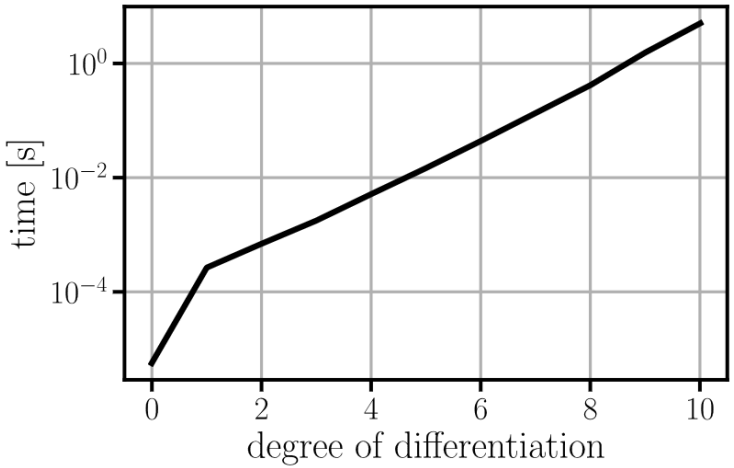
$$\frac{\partial^3 z}{\partial x^3} = \frac{\partial}{\partial x}\left(\frac{\partial^2 z}{\partial x^2}\right) = \frac{\partial}{\partial x}\left(\frac{\partial}{\partial x}\left(\frac{\partial z}{\partial y}\right)\right) \cdot \frac{\partial y}{\partial x} + \frac{\partial}{\partial x}\left(\frac{\partial z}{\partial y}\right) \cdot \frac{\partial}{\partial x}\left(\frac{\partial y}{\partial x}\right) + \frac{\partial}{\partial x}\left(\frac{\partial z}{\partial y}\right) \cdot \frac{\partial}{\partial x}\left(\frac{\partial y}{\partial x}\right) + \frac{\partial z}{\partial y} \cdot \frac{\partial}{\partial x}\left(\frac{\partial}{\partial x}\left(\frac{\partial y}{\partial x}\right)\right)$$

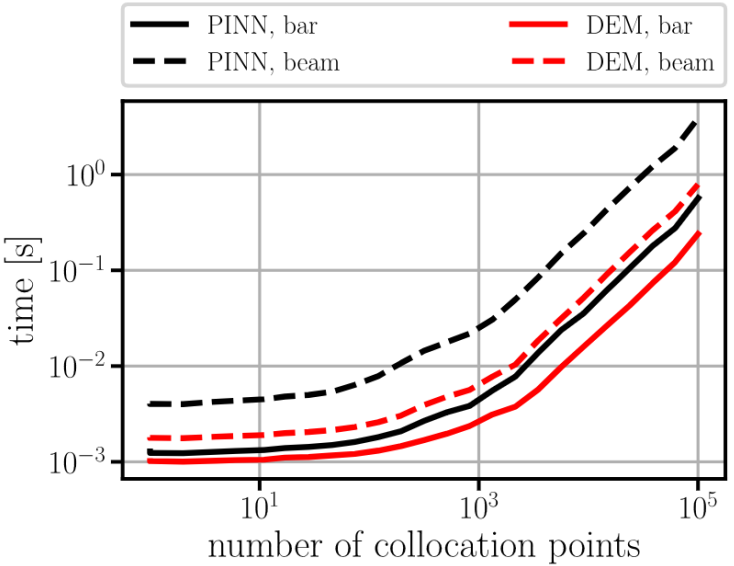
5.1 Variations

Computation times of higher-order derivatives with in autograd PyTorch

Deep energy method relies on weak form of the differential equation, i.e., requires lower order derivatives (1 and 2 for correspondingly the bar and beam equations)

Exponential growth





5.1 Variations

The runtime of PINNs can be improved by reducing the order of the differential operators

- Variational Principles
  - \+ Integration by parts of the partial differential equation
  - \+ Lowering of the order of the differential operator
  - \- Instead of differentiation an integration is needed (but that is cheaper than a differentiation)

Equivalent to the summation over collocation points in standard PINNs

- Variational Physics-informed Neural Networks (VPINNs)
  - Instead of minimizing the residual of the differential equation (the strong form), a variational formulation (the weak form) is minimized
  - Weak Adversarial Networks use neural networks as test functions (instead of manually selecting them)

- A special form of VPINNs is the Deep Energy Method
  - Minimization of the potential energy Π_{tot} of a physical system

$$\Pi_{\text{tot}} = \Pi_i + \Pi_e$$

  - Π_i is the internal energy; Π_e is the external energy

5.1.1 Deep Energy Method

A Deep Learning-Based Numerical Algorithm for Solving Variational Problems, Weinan et al. 2018

Basic methodology of the deep energy method (also called Deep Ritz Method)

- A neural network $u_{NN}(x)$ approximates the solution u to a partial differential equation

$$\hat{u} = u_{NN}(x)$$

- Computation of relevant derivatives of $\hat{u}$ with respect to inputs x
- Computation of the potential energy Π_{tot}

$$\Pi_{\text{tot}} = \Pi_i + \Pi_e$$

- Computation of the error between the data $\tilde{u}$ and the prediction for the observed datapoints (e.g., boundary conditions)

$$\mathcal{L}_B = \frac{1}{m_B} \sum_{i=1}^{m_B} \left(\tilde{u}_B^i - \hat{u}_B^i \right)^2$$

- Computation of the cost function

$$C = \Pi_{\text{tot}} + \mathcal{L}_B$$

- Minimization of the cost function C via gradient-based optimization (to improve the prediction $\hat{u}$)

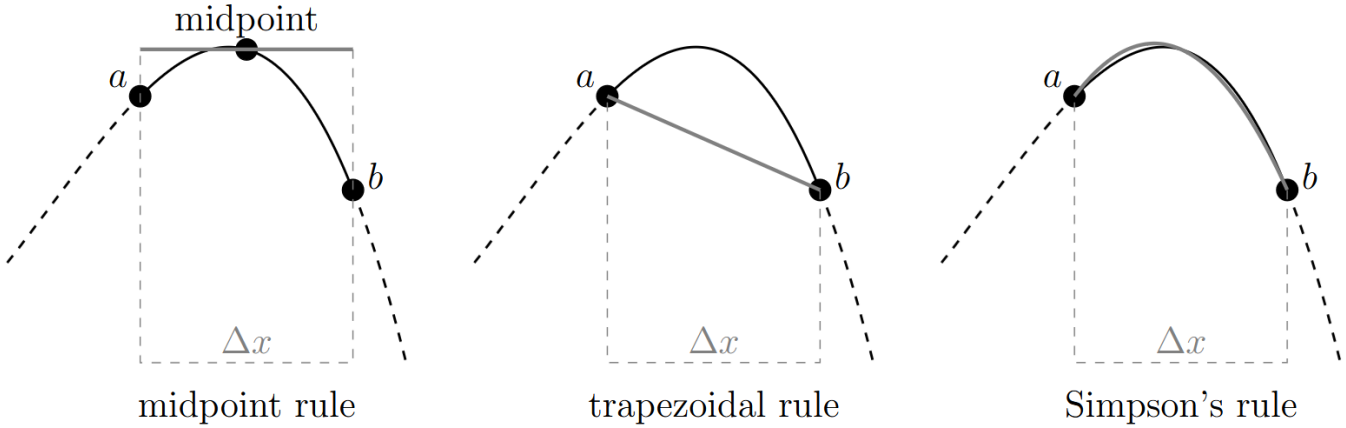
5.1.1 Deep Energy Method

The evaluation of the potential energy Π_{tot} requires an integration

$$\Pi_{\text{tot}} = \Pi_i + \Pi_e$$

Solution can be evaluated at specific collocation points and integrated numerically by, e.g.,

- Midpoint rule
- Trapezoidal rule
- Simpson's rule
- Gaussian quadrature

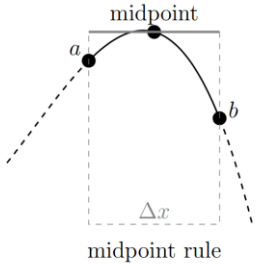


5.1.1 Deep Energy Method

Domain is divided into n subdomains with size $\Delta x = \frac{b-a}{n}$ in which the function $f(x)$ is approximated up to the order of integration and integrated

- Midpoint rule, order of integration 0

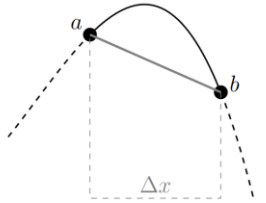
$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} f\left(\frac{x_{i+1} - x_i}{2}\right) \Delta x$$



midpoint rule

- Trapezoidal rule, order of integration 1

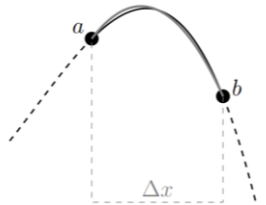
$$\int_a^b f(x)dx \approx \frac{\Delta x}{2}[f(x_0) + 2f(x_1) + \cdots + 2f(x_{n-1}) + f(x_n)]$$



trapezoidal rule

- Simpson's rule, order of integration 2

$$\int_a^b f(x)dx \approx \frac{\Delta x}{3}[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots + 4f(x_{n-1}) + f(x_n)]$$



Simpson's rule

The integration points are defined as $x_i = a + i\Delta x$

5.1.2 One-Dimensional Static Model

The strong form of the linear elastic bar

$$\frac{d}{dx}\left(EA\frac{du}{dx}\right)+p=0, \qquad x\in\Omega$$
$$EA\frac{du}{dx}=F, \qquad x\in\Gamma_N$$
$$u=g, \qquad x\in\Gamma_D$$

A weak form of the linear elastic bar (expressed in terms of potentials)

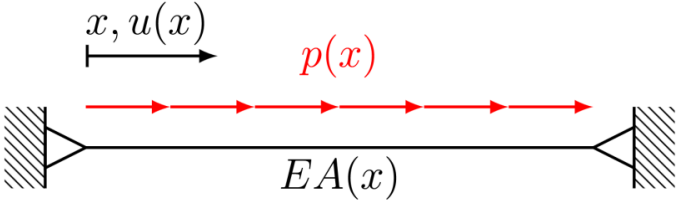
$$\Pi_{\text{tot}}=\Pi_i+\Pi_e$$

The internal energy

$$\Pi_i=\frac{1}{2}\int_\Omega EA\left(\frac{du}{dx}\right)^2 dx$$

The external energy

$$\Pi_e=-\int_\Omega p(x)u(x)dx-\sum_i F_i u(x_i)$$



5.1.2 One-Dimensional Static Model

Strong enforcement of Dirichlet boundary conditions

- Prediction of a quantity $z(x) = u_{NN}(x)$ for the displacement from a neural network
- Modification of $z(x)$, such that the boundary conditions are automatically fulfilled

$$u(x) = a(x)z(x) + b(x)$$

- Where $a(x), b(x)$ are selected for this purpose

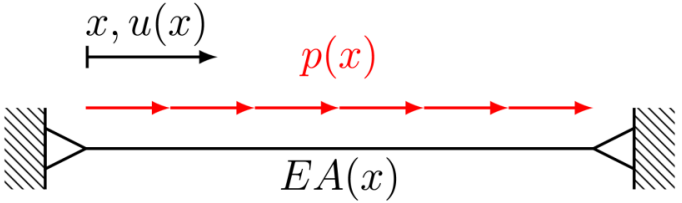
Example

- Possible solution

$$u(0) = u(L) = 0$$

$$a(x) = (x - L)x$$
$$b(x) = 0$$

$$u(x) = (x - L)xz(x)$$

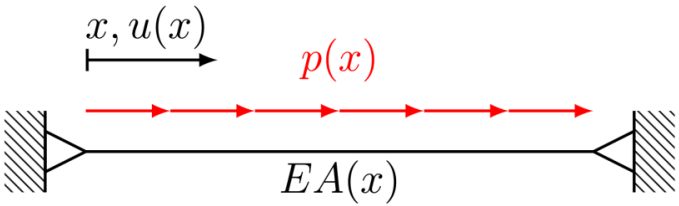


Note: Neumann boundary conditions are already enforced via the weak form

5.1.2 One-Dimensional Static Model

Example with a manufactured solution defined in the domain $x \in [0,1]$

$$u(x) = \sin(2\pi x)$$



- After insertion of $u(x)$ into the differential equation with $EA(x) = 1$

$$p(x) = 4\pi^2 \sin(2\pi x)$$

- With two Dirichlet boundary conditions

$$u(0) = u(1) = 0$$

- Prediction of the displacement and with strong enforcement of the Dirichlet boundary conditions

$$\hat{u} = (1 - x)x u_{NN}(x)$$

- Optimization with the cost function C (note the missing boundary term)

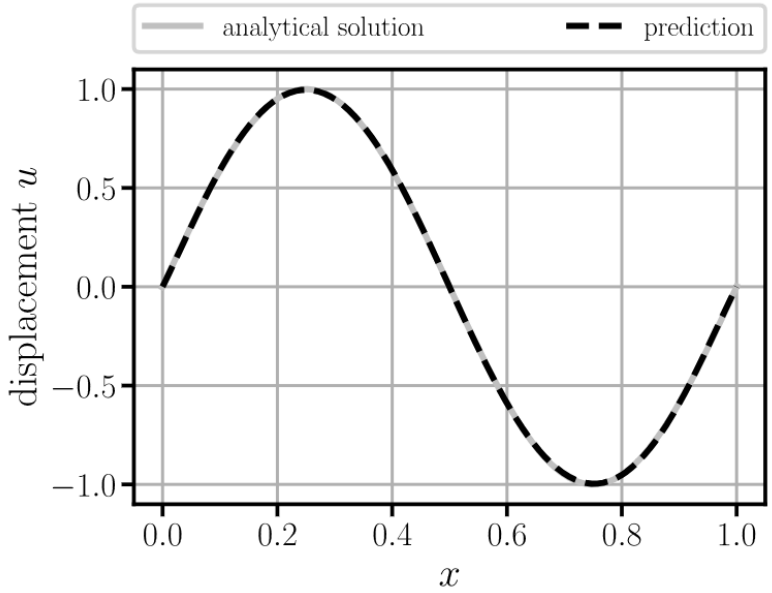
$$C = \Pi_{\text{tot}}$$

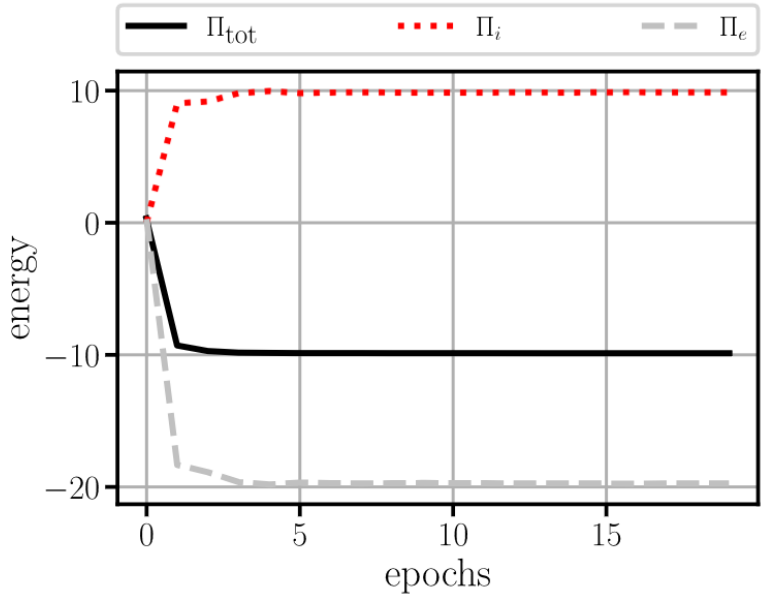
- The potential energy

$$\Pi_{\text{tot}} = \int_\Omega \frac{1}{2} EA \left(\frac{du}{dx}\right)^2 dx - \int_\Omega p(x)u(x)dx - \sum_i F_i u(x_i)$$

5.1.2 One-Dimensional Static Model

Results after 20 epochs

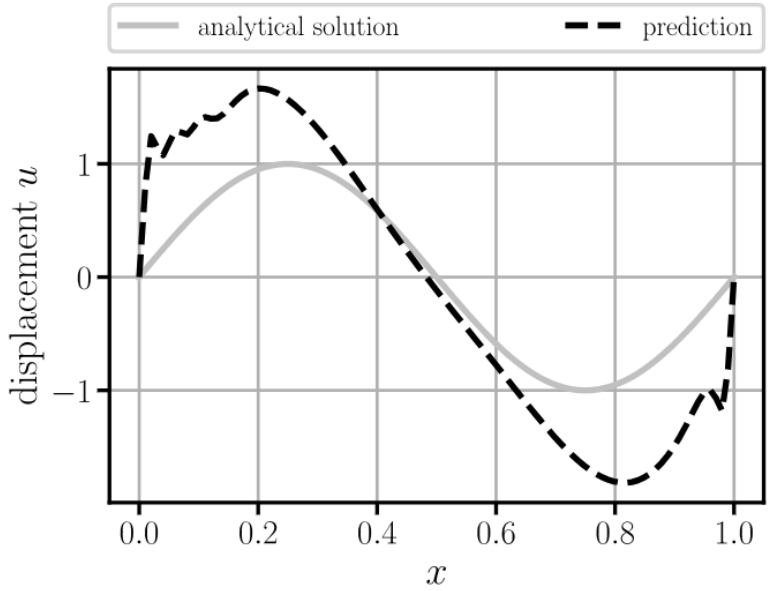


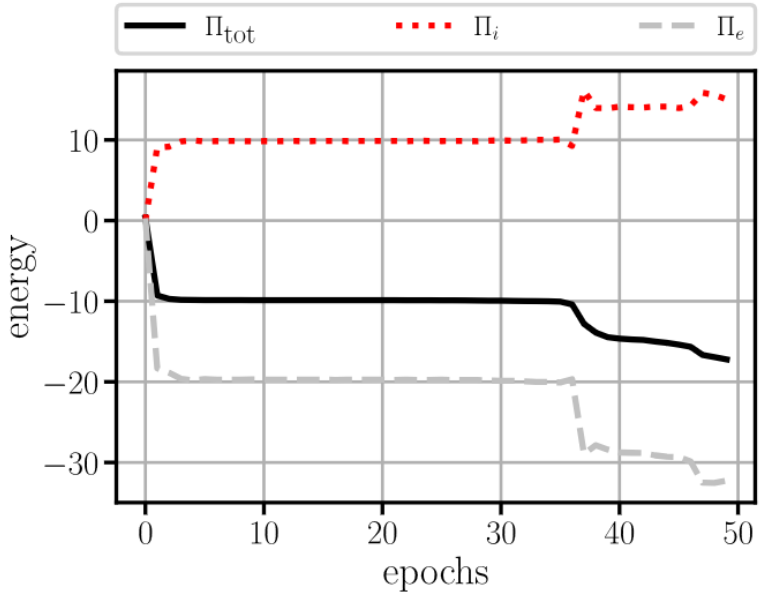


- Faster predictions than with standard PINNs (fewer epochs & cheaper epochs)
- Note, that the final potential energy Π_{tot} is nonzero

5.1.2 One-Dimensional Static Model

But: Overfitting after 35 iterations?





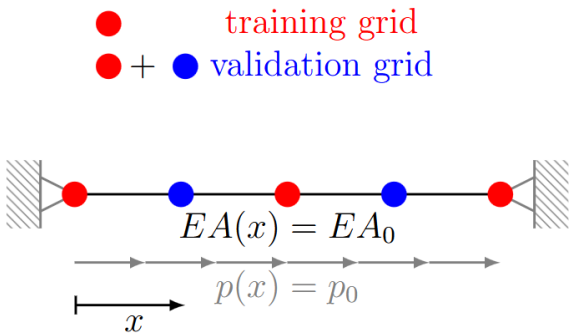
Can be avoided through regularization, such as early stopping (monitor the energies on a validation grid)

5.1.2 One-Dimensional Static Model

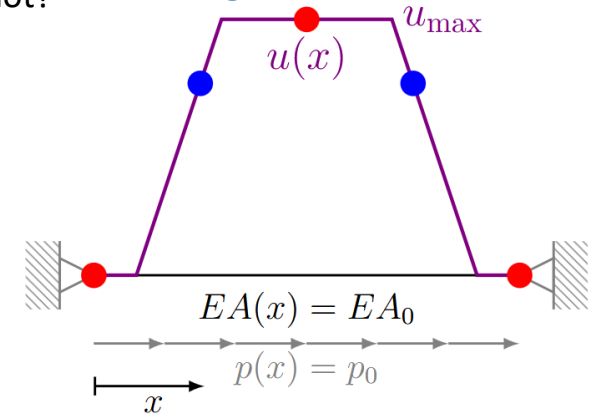
On quadrature rules for solving Partial Differential Equations using Neural Networks, Riviera et al. 2022

Why does the deep energy method overfit and PINNs do not?

- Consider the following toy example



Toy example



One possibility of an "optimal" deformation

- The proposed deformation leads to an external energy of $-\infty$ for $u_{max} \to \infty$

$$\Pi_e = -\int_\Omega p(x)u(x)dx \to -\infty$$

- The internal energy of the system is zero on the training grid

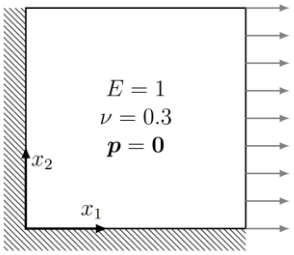
$$\Pi_i = \frac{1}{2}\int_\Omega EA\left(\frac{du}{dx}\right)^2 dx = 0$$

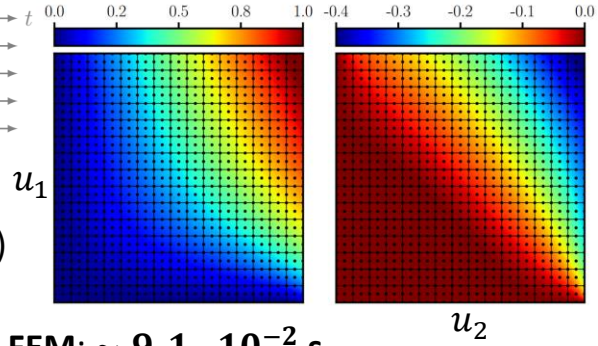
Unphysical minimization of Π_{tot} towards $-\infty$ is possible (on the training grid)

→ use the validation grid!

5.1.3 Two-Dimensional Static Model

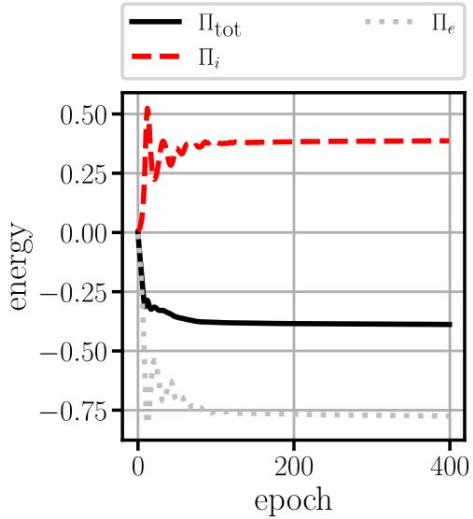
Reconsider the plate in membrane action
(with a stress singularity)

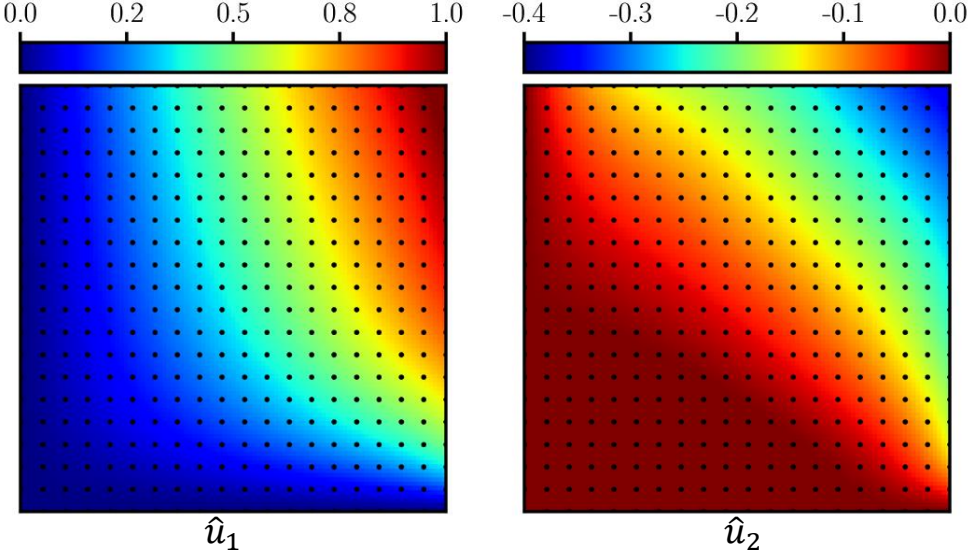




Prediction with deep energy method after 400 epochs with Adam (PINN took 5000)

- Using 20×20 collocation points
- **DEM**: $400 \cdot\sim 4 \cdot 10^{-3}$ s $=\sim \mathbf{1.6}$ **s**; **PINN**: $5000 \cdot\sim 2.3 \cdot 10^{-2}$s $=\sim \mathbf{1.2 \cdot 10^{2}}$ **s**: **FEM**: $\sim \mathbf{9.1 \cdot 10^{-2}}$ **s**



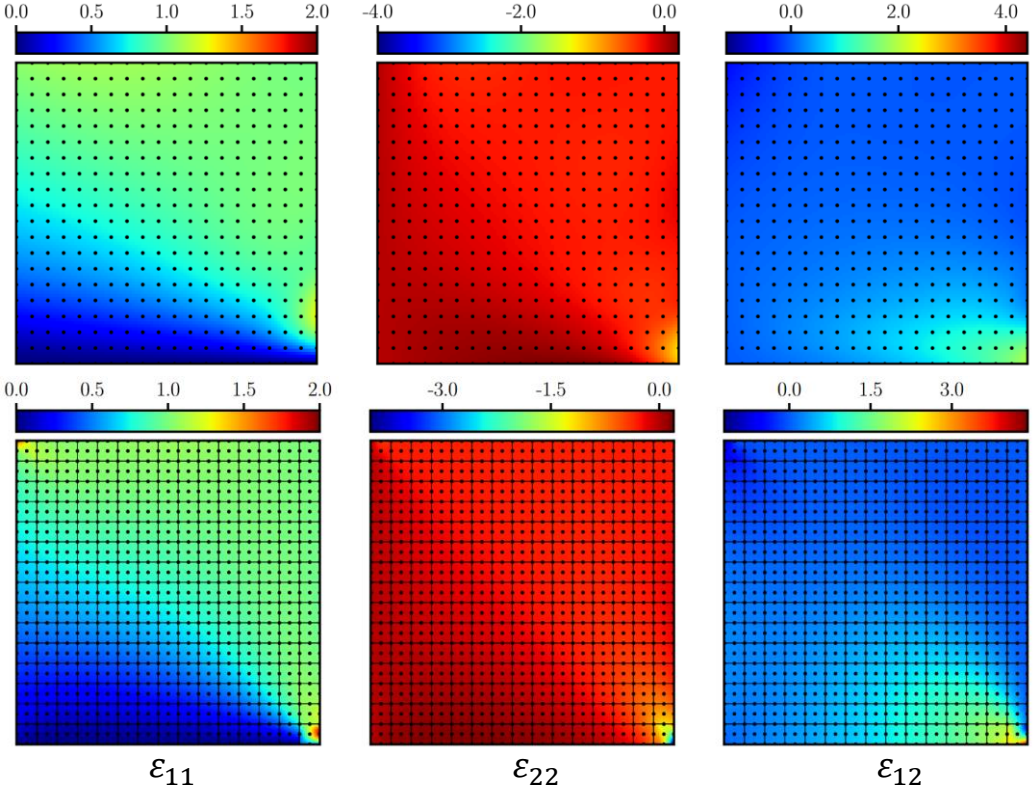


5.1.3 Two-Dimensional Static Model

Although displacement field $\boldsymbol{u}$ has improved through deep energy method, the gradients remain inaccurate

Deep energy method

Finite element reference solution



5.1.1 Deep Energy Method

Advantages

- Lower order differential operators
- (Possibility to work with singularities)

Disadvantages

- Introduction of numerical errors through numerical integration
- (More complex optimization task, if no strong enforcement of the boundary conditions is chosen)
- No inbuilt regularization as empirically observed in PINNs
- No direct inversion

Remedies

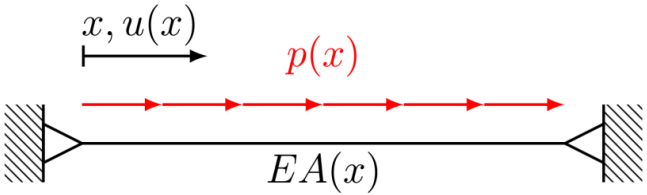
- Finer grids or better integration schemes
- Strong enforcement of the boundary conditions
- Usage of a regularization technique

5.1.4 Variational Physics-Informed Neural Networks

Variational Physics-Informed Neural Networks For Solving Partial Differential Equations, Kharazmi et al. 2019

The strong form of the linear elastic bar

$$\frac{d}{dx}\left(EA\frac{du}{dx}\right)+p=0, \qquad x\in\Omega$$
$$EA\frac{du}{dx}=F, \qquad x\in\Gamma_N$$
$$u=g, \qquad x\in\Gamma_D$$



The general weak form of the linear elastic bar

- Multiplication with test functions v_i

$$\int_\Omega \frac{d}{dx}\left(EA\frac{du}{dx}\right)v_i d\Omega+\int_\Omega pv_i d\Omega=0, \forall v_i$$

- Integration by parts

$$-\int_\Omega EA\frac{du}{dx}\frac{dv_i}{dx}d\Omega+\int_\Gamma EA\frac{du}{dx}v_i d\Gamma+\int_\Omega pv_i d\Omega=0, \forall v_i$$

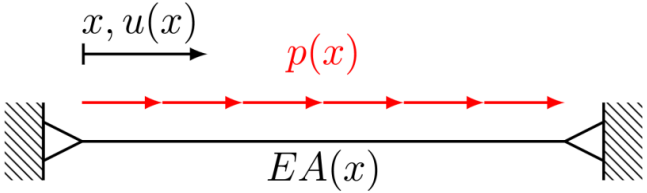
- Neumann (natural) boundary conditions are incorporated in the weak form as

$$\int_\Gamma EA\frac{du}{dx}v_i d\Gamma=\int_{\Gamma_N} Fv_i d\Gamma_N$$

5.1.4 Variational Physics-Informed Neural Networks

Prediction of solution u via neural network $u_{NN}(x)$

$$\hat{u} = u_{NN}(x)$$



Residual loss via the weak form with test functions v_i

$$-\int_\Omega EA\frac{du}{dx}\frac{dv_i}{dx}d\Omega + \int_\Gamma EA\frac{du}{dx}v_i d\Gamma + \int_\Omega pv_i d\Omega = 0, \forall v_i$$

The residual loss over test function i

$$\mathcal{L}_{V_i} = \left(-\int_\Omega EA\frac{d\hat{u}}{dx}\frac{dv_i}{dx}d\Omega + \int_{\Gamma_N} Fv_i d\Gamma_N + \int_\Omega pv_i d\Omega\right)^2$$

The residual loss over all test functions

$$\mathcal{L}_V = \frac{1}{m_V}\sum_{i=1}^{m_V}\mathcal{L}_{V_i}$$

Test functions can be chosen arbitrarily with the condition that they are zero at inhomogeneous Dirichlet boundaries

$$\text{e.g., } v_i(x) = \sin\left(\frac{i\pi}{L}x\right)$$

Minimization of $C = \mathcal{L}_V + \mathcal{L}_B$ via gradient-based optimization

5.1.5 Weak Adversarial Networks

Weak adversarial networks for high-dimensional partial differential equations, Zang et al. 2020

Residual loss based on the weak form using test functions v_i

$$\mathcal{L}_{V_i} = \left(- \int_\Omega EA \frac{d\hat{u}}{dx} \frac{dv_i}{dx} d\Omega + \int_{\Gamma_N} F v_i d\Gamma_N + \int_\Omega p v_i d\Omega \right)^2$$

Weak adversarial networks rely on a neural network for the test functions v_i

$$\hat{v}_i = v_{i_{NN}}(x; \boldsymbol{\Theta}_v)$$

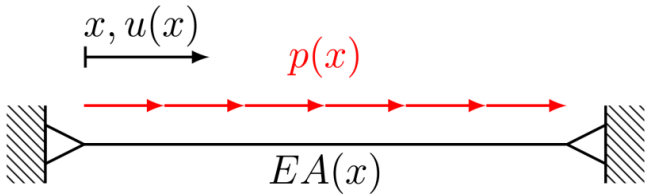
In addition to the neural network of the solution u

$$\hat{u} = u_{NN}(x; \boldsymbol{\Theta}_u)$$

To continuously challenge the solution $\hat{u}$, the test functions aim at maximizing the cost function $C = \mathcal{L}_V + \mathcal{L}_B$

This is achieved through a minimax optimization

$$\min_{\boldsymbol{\Theta}_u} \max_{\boldsymbol{\Theta}_v} C$$



Exercises

E.18 Variations of Physics-Informed Neural Networks (C)

- Try the main variations of physics-informed neural networks on the one-dimensional static bar problem. Familiarize yourself with the implementational differences.

E.19 Deep Energy Method for a Plate in Membrane Action (C)

- Use the deep energy method to obtain the solution of a plate in membrane action and compare the results with reference solutions obtained with the finite element method. Tune the neural network to improve convergence. Lastly compare the results with a standard physics-informed neural network using E.17

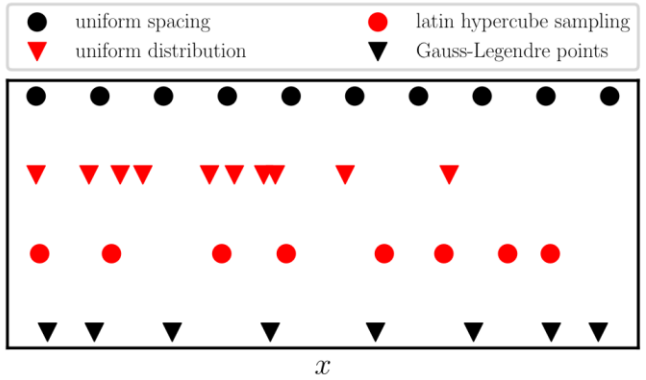
5.2 Extensions

5.2.1 Optimizer

- Adam, L-BFGS, etc. (see Chapter 2): Typically Adam and L-BFGS are combined (start with Adam)

5.2.2 Sampling

- Uniform spacing, uniform distribution, latin hypercube sampling, Gauss-Legendre points



5.2.3 Boundary & initial conditions

- Strong enforcement
- Weak enforcement
  - With manual weighting terms

$$C = \kappa_R \mathcal{L}_R + \kappa_B \mathcal{L}_B$$

  - With automatic (learnable) weighting terms

$$\min_{\boldsymbol{\Theta}} \max_{\boldsymbol{\kappa}} C$$

5.2.4 Differentiation

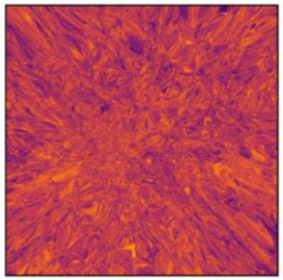
- Automatic differentiation
- Numerical differentiation: e.g., central difference $\frac{dy(x_i)}{dx} \approx \frac{y(x_{i+1}) - y(x_{i-1})}{x_{i+1} - x_{i-1}}$
- Analytical differentiation through prior interpolations, e.g., through splines as in spline-PINNs

Exercises

E.20 Extensions for Physics-Informed Neural Networks (C)

- Try the different extensions for physics-informed neural networks (optimizer, sampling, loss term weighting, strong enforcement, weighting per collocation point, learning rate scheduling, activation functions, numerical differentiation, convolutional neural networks, feature encoding layers). Understand when which extension is beneficial and familiarize yourself with the implementations.

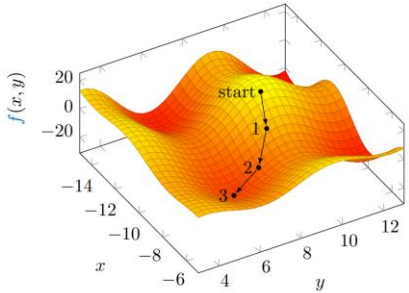
5.2.5 Ansatz



Neural network ansatz

$$f_{NN} = f_3\left(f_2\left(f_1(x)\right)\right)$$

- High representational capacity
- Nested nonlinearities requires iterative solution procedure



Finite element ansatz

$$f_{FEM} = \sum_{i=1}^{n} N_i(x) u_i$$

- Lower representational capacity
- Can be solved directly (one matrix inversion)

$$\boldsymbol{KU} = \boldsymbol{F}$$

$$\boldsymbol{U} = \boldsymbol{K}^{-1}\boldsymbol{F}$$

Can the advantages of both Ansatz spaces be combined?

- Finite element interpolated neural networks
- Hierarchical deep learning neural networks (HiDeNN)

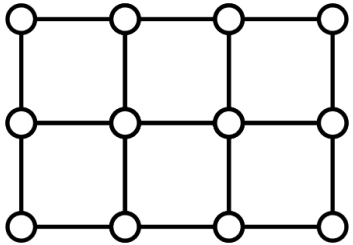
5.2.5.3 Finite Element Interpolated Neural Networks

Finite element interpolated neural networks for solving forward and inverse problems, Badia et al. 2024

Finite element interpolated neural networks

- From coordinates x_1, x_2 a neural network predicts the solution field, e.g., displacements

$$\hat{u}(x_1, x_2) = u_{NN}(x_1, x_2)$$



- The neural network is evaluated only at the nodal positions of a finite element mesh

$$\left\{\left(x_1^{(1)}, x_2^{(1)}\right), \left(x_1^{(2)}, x_2^{(2)}\right), \dots, \left(x_1^{(n)}, x_2^{(n)}\right)\right\}$$

- The predictions at the nodal positions serve as the nodal values of the mesh

$$\left\{\hat{u}^{(1)}, \hat{u}^{(2)}, \dots, \hat{u}^{(n)}\right\}$$

- A field interpolation is recovered through the finite element ansatz

$$\hat{u}_{FEM} = \sum_{i=1}^{n} N_i(x)\hat{u}^{(i)}$$

- Thus: the neural network is projected onto a finite element space
- From $\hat{u}_{FEM}$ the potential energy $\Pi_{\text{tot}} = \Pi_i + \Pi_e$ can be computed (as in the deep energy method) and minimized with respect to the neural network parameters $\boldsymbol{\Theta}$

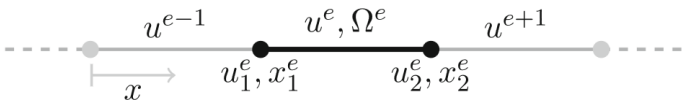
- **Note**: the representational space is effectively the finite element space, which is explored iteratively

5.2.5.4 Hierarchical Deep Learning Neural Networks

Hierarchical deep-learning neural networks: finite elements and beyond, Zhang et al. 2021

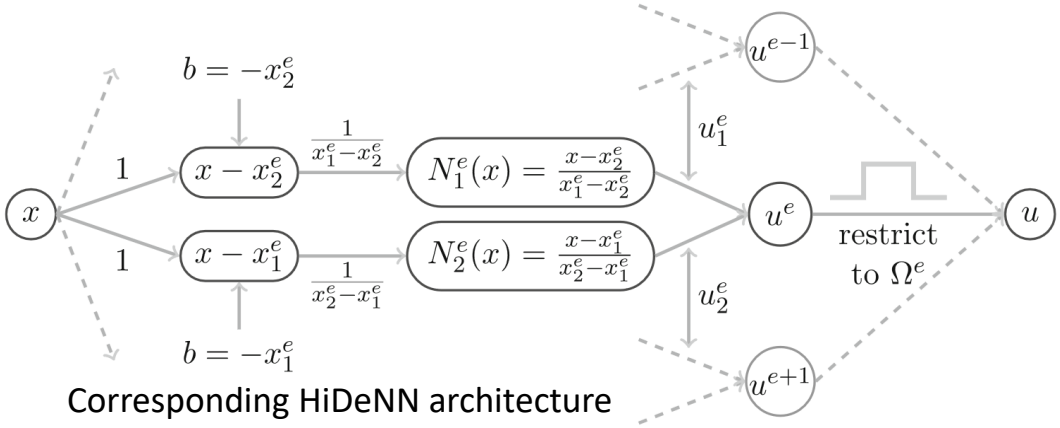
HiDeNN also reduces the space to the finite element space

- Finite elements and neural networks are both composed of linear transformations & nonlinearities
- Assemble a neural network from finite elements (i.e., a neural network constrained to the finite element space)



One-dimensional finite element

Hardcoding the shape functions is faster than treating them as a neural network.



Corresponding HiDeNN architecture

- From the predicted $\hat{u}$ the potential energy $\Pi_{\text{tot}} = \Pi_i + \Pi_e$ can be computed and minimized with respect to the standard nodal degrees of freedom of a finite element ansatz
  - The gradients are obtained with automatic differentiation (although they can be obtained analytically)

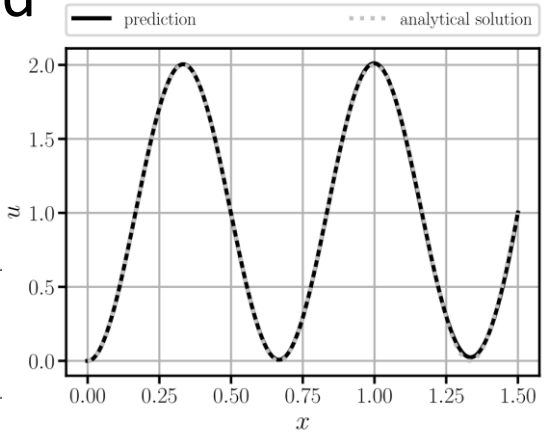
Using the analytical gradients is faster!

Deep learning in computational mechanics: a review, Herrmann et al. 2023

5.2.5.5 Comparison to the Finite Element Method

Manufactured solution for a one-dimensional linear static bar

$$u(x) = 1 - \cos(3\pi x)$$
$$p(x) = -6x\pi\sin(3\pi x) - 9\pi^2(x^2+1)\cos(3\pi x)$$



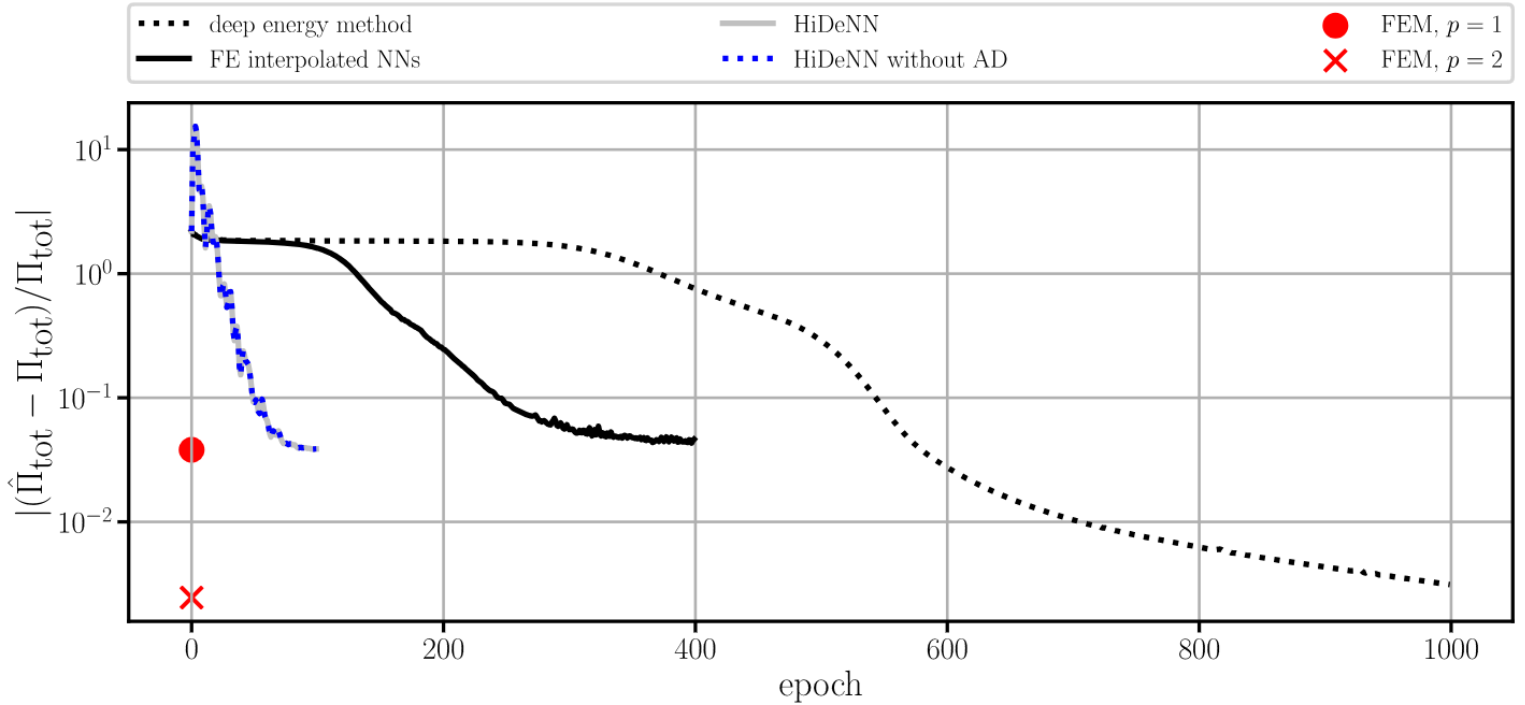
method	relative error in potential energy	elapsed time per epoch	number of epochs	total elapsed time
deep energy method	0.31%	$\sim 8.1 \cdot 10^{-4}$ s	~ 1000	~ 0.81 s
FE-interpolated NNs, $p = 1$	4.6%	$\sim 1.3 \cdot 10^{-2}$ s	~ 400	~ 5.3 s
HiDeNN, $p = 1$	3.9%	$\sim 1.3 \cdot 10^{-2}$ s	~ 100	~ 1.3 s
HiDeNN without AD, $p = 1$	3.9%	$\sim \mathbf{1.7 \cdot 10^{-3}}$ s	~ 100	~ 0.17 s
FEM, $p = 1$	**3.8%**	$\sim 4.6 \cdot 10^{-2}$ s	**1**	$\sim$ **0.046** s
FEM, $p = 2$	0.24%	$\sim 5.2 \cdot 10^{-2}$ s	1	~ 0.052 s

- Neither finite element-interpolated neural networks, nor HiDeNN are able to combine the advantages of NNs and FE
  - Are limited to the finite element space (lower representational capacity)
  - Require a gradient-based optimization (higher computation times)

- Finite element-interpolated neural networks retain the advantage of being able to learn multiple solutions

5.2.5.5 Comparison to the Finite Element Method

Visualization of convergence over the epochs



Exercises

E.21 Variations of the Ansatz Space (C)

- Compare the different Ansatz spaces (i.e., neural network and finite element) on the one-dimensional static bar problem using the different solvers (gradient-based optimization and direct solving when possible). The considered methods are the deep energy method, FE-interpolated neural networks, HiDeNN (with and without automatic differentiation), and the finite element method.

Contents

5 Advanced Physics-Informed Neural Networks

Leon Herrmann

Stefan Kollmannsberger

Chair of Data Engineering in Construction

Bauhaus-Universität Weimar

Deep Learning in Computational Mechanics – an introductory course, Herrmann et al. 2025

website

book