5 Advanced Physics-Informed Neural Networks

Leon Herrmann

Stefan Kollmannsberger

Chair of Data Engineering in Construction

Bauhaus-Universität Weimar

Paris, February 2025

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)

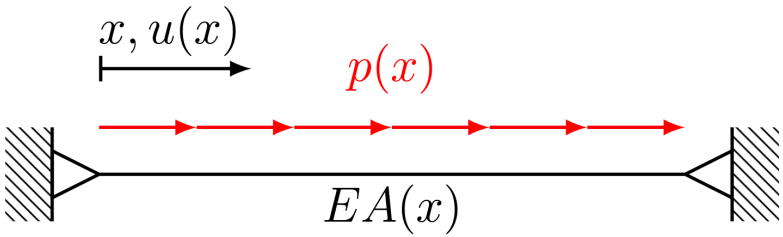
$$\hat{u}(x,t) = u_{NN}(x,t)$$

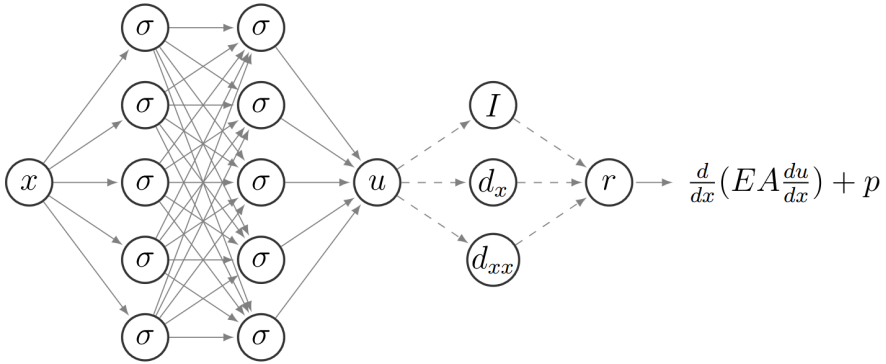
The physics are encoded via the cost function (composed of the residual $\mathcal{L}_R$ and boundary loss $\mathcal{L}_B$)

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

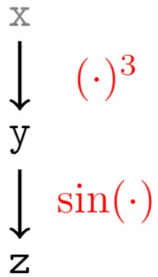
The derivatives in the loss terms $\mathcal{L}_R, \mathcal{L}_B$ are computed with automatic differentiation with respect to the inputs x, t

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

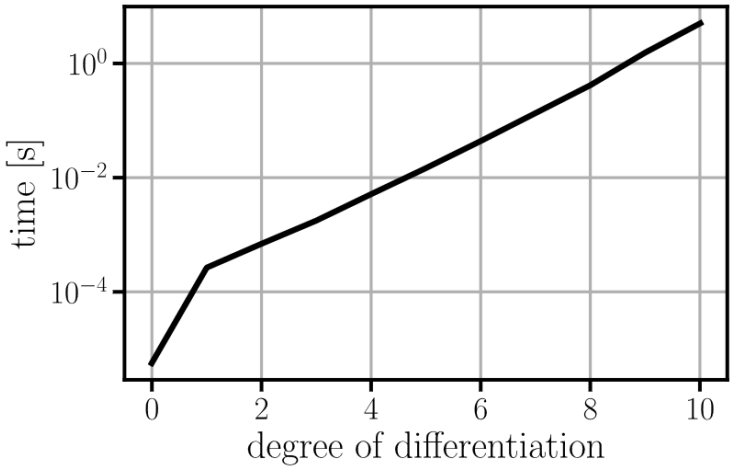
$$\frac{\partial^3 z}{\partial x^3} = \frac{\partial}{\partial x}\left(\frac{\partial^2 z}{\partial x^2}\right) = \frac{\partial}{\partial x}\left(\frac{\partial}{\partial x}\left(\frac{\partial z}{\partial y}\right)\right) \cdot \frac{\partial y}{\partial x} + \frac{\partial}{\partial x}\left(\frac{\partial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

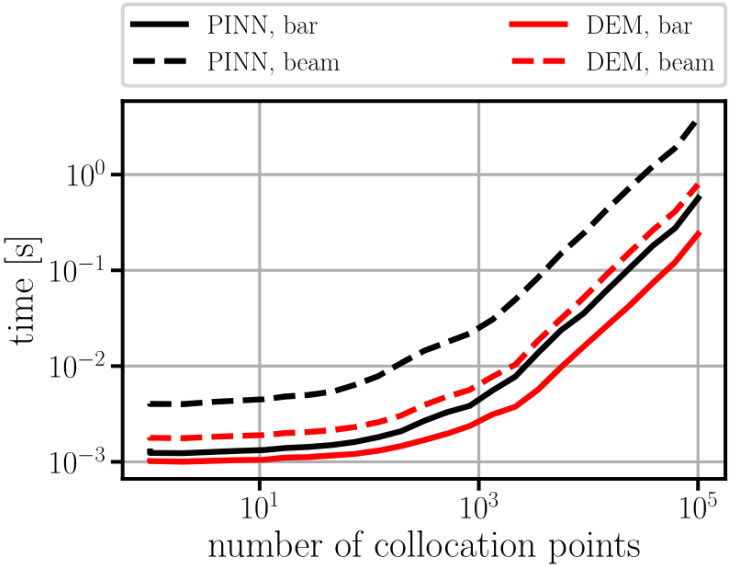
5.1 Variations

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





DEM is better, but still scales exponential in time

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 $\hat{u}$)

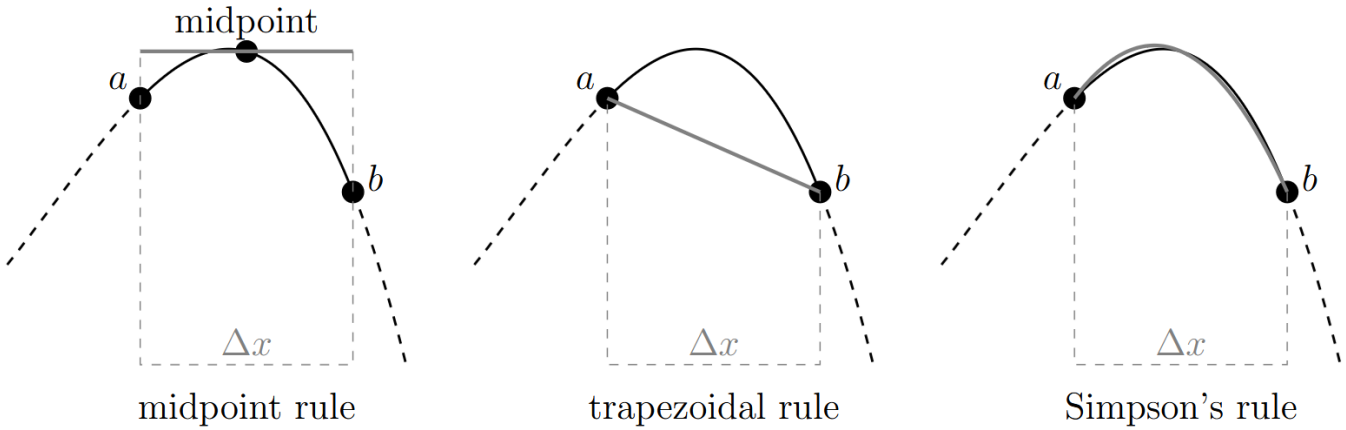
5.1.1 Deep Energy Method

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

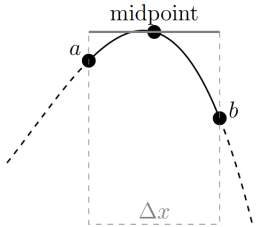


5.1.1 Deep Energy Method

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

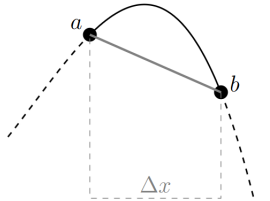
$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} f\left(\frac{x_{i+1} - x_i}{2}\right)\Delta x$$



midpoint rule

- Trapezoidal rule, order of integration 1

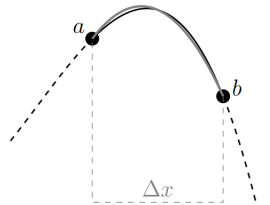
$$\int_a^b f(x)dx \approx \frac{\Delta x}{2}[f(x_0) + 2f(x_1) + \cdots + 2f(x_{n-1}) + f(x_n)]$$



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 $x_i = a + i\Delta x$

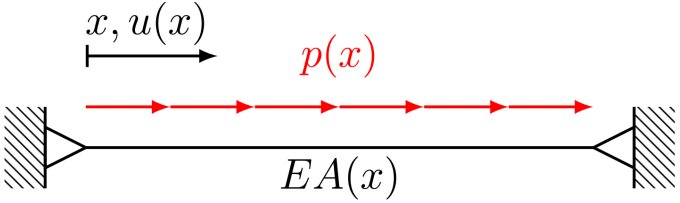
5.1.2 One-Dimensional Static Model

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

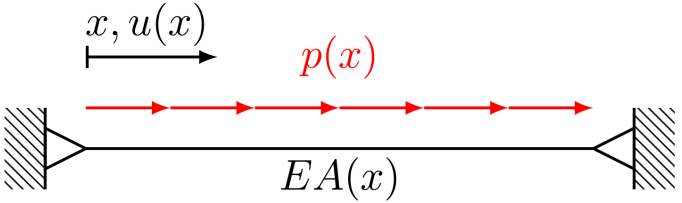
Example

- Possible solution

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

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

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

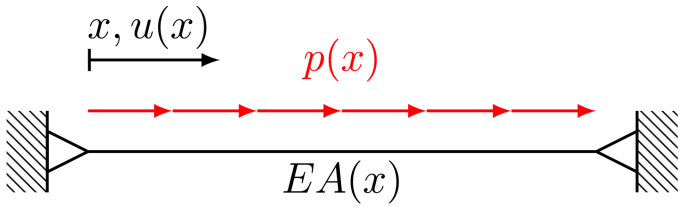


Note: Neumann boundary conditions are already enforced via the weak form

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)

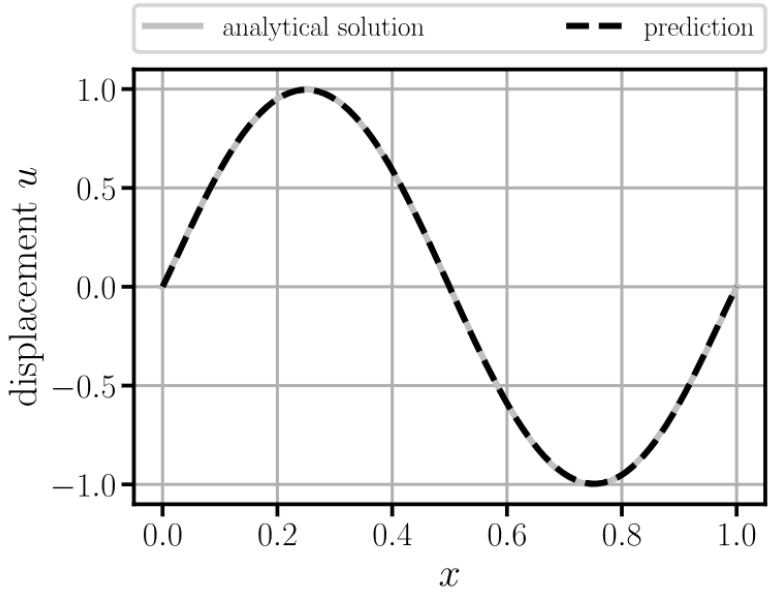
$$C = \Pi_{\text{tot}}$$

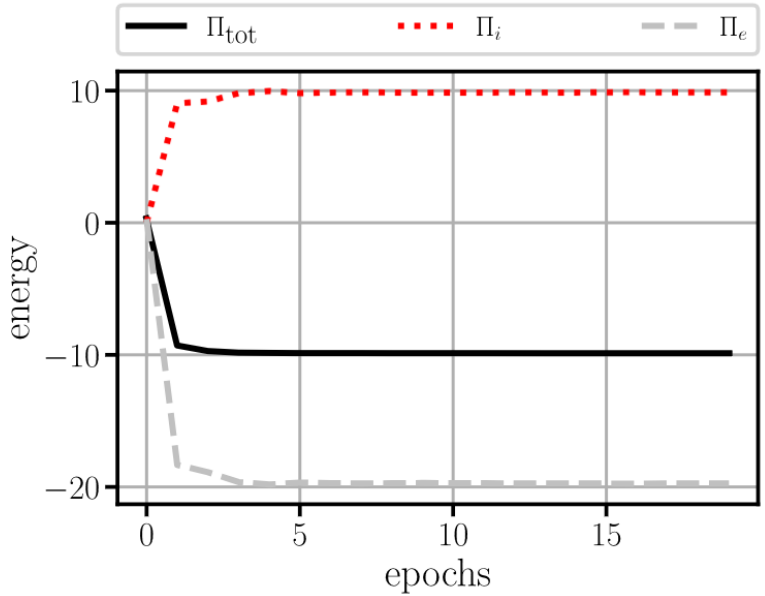
- The potential energy

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

5.1.2 One-Dimensional Static Model

Results after 20 epochs

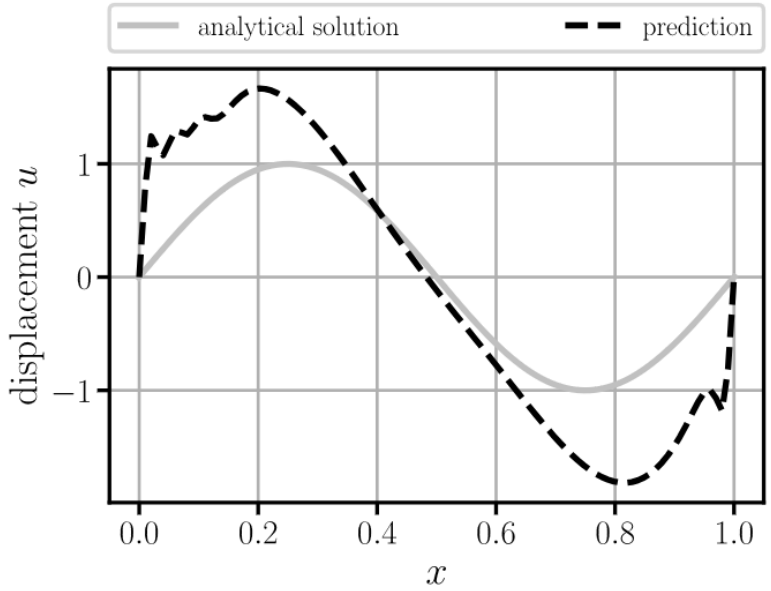


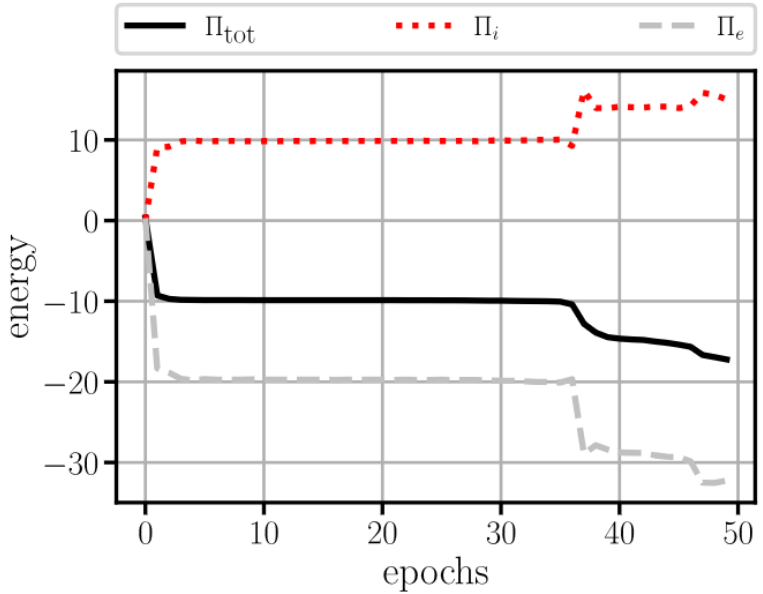


- Faster predictions than with standard PINNs (fewer epochs & cheaper epochs)
- Note, that the final potential energy Π_{tot} is nonzero

5.1.2 One-Dimensional Static Model

But: Overfitting after 35 iterations?





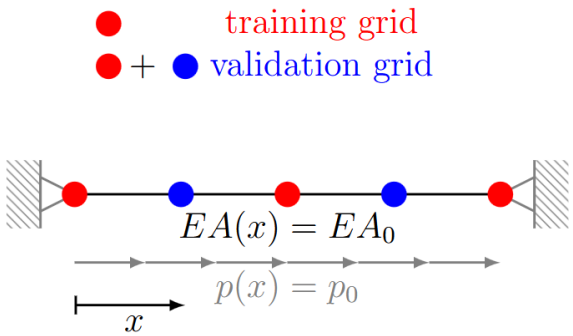
Can be avoided through regularization, such as early stopping (monitor the energies on a validation grid)

5.1.2 One-Dimensional Static Model

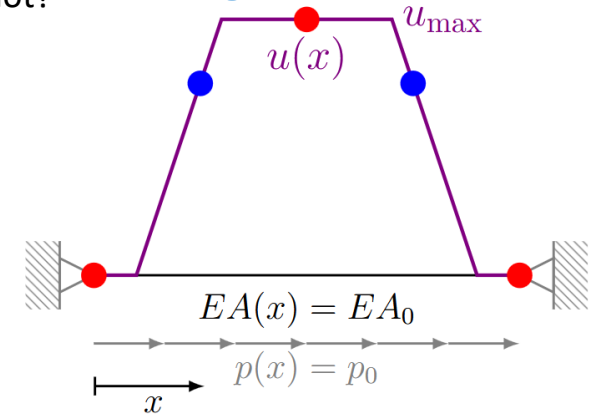
On quadrature rules for solving Partial Differential Equations using Neural Networks, Riviera et al. 2022

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 $u_{max} \to \infty$

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

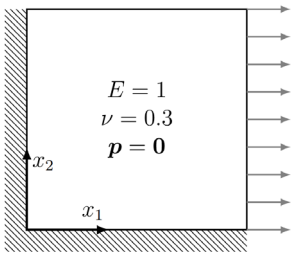
- The internal energy of the system is zero on the training grid

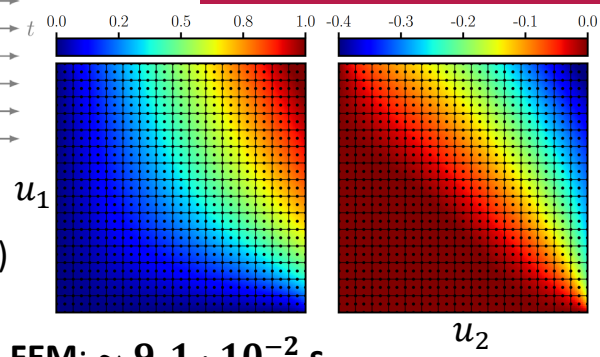
$$\Pi_i = \frac{1}{2}\int_\Omega EA\left(\frac{du}{dx}\right)^2 dx = 0$$

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

→ use the validation grid!

5.1.3 Two-Dimensional Static Model

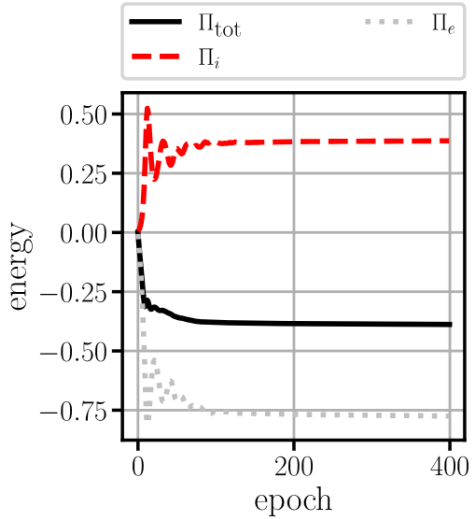


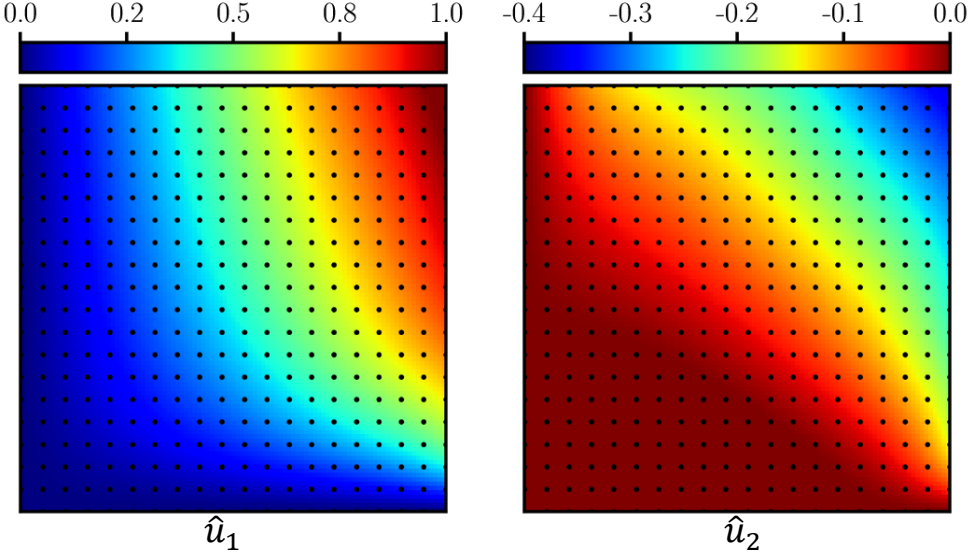


Reconsider the plate in membrane action
(with a stress singularity)

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**



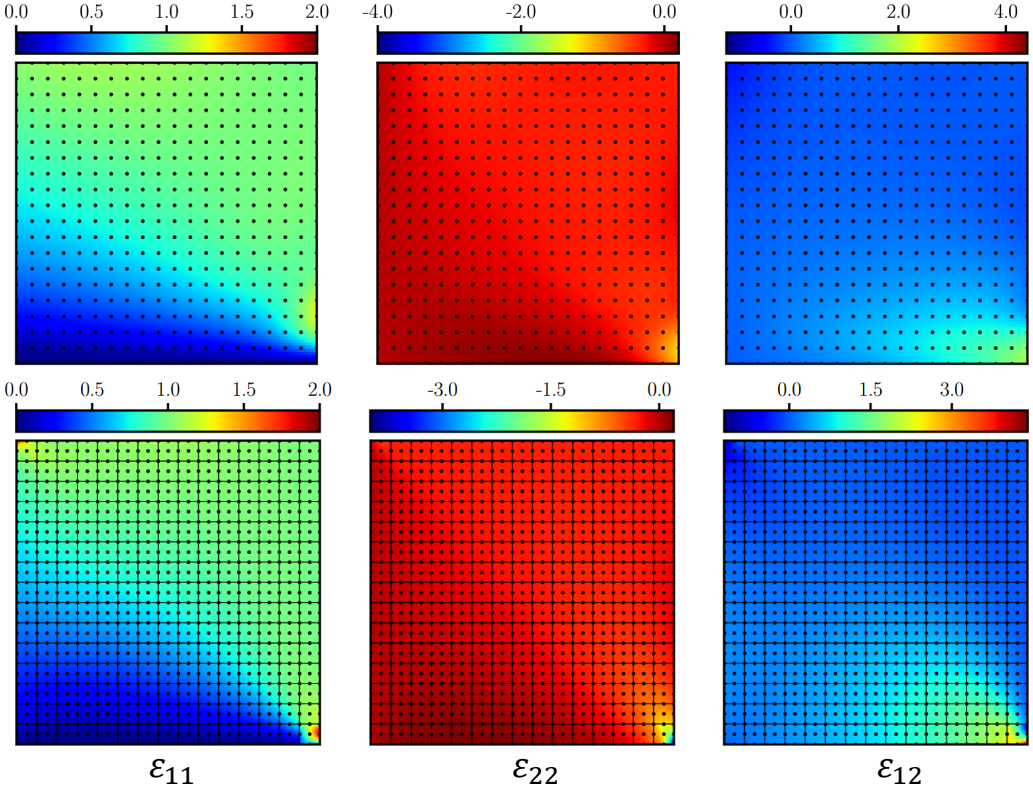


5.1.3 Two-Dimensional Static Model

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

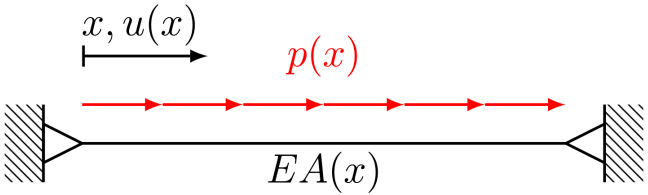
- Finer grids or better integration schemes
- Strong enforcement of the boundary conditions
- Usage of a regularization technique

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

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

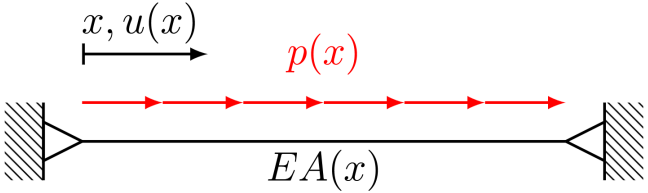
- Neumann (natural) boundary conditions are incorporated in the weak form as

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

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

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

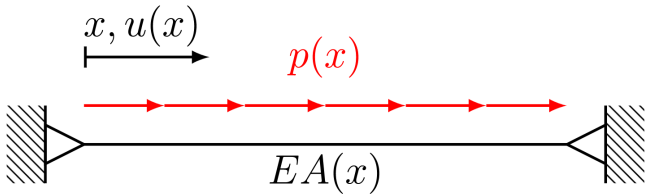
In addition to the neural network of the solution u

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

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

Contents

5 Advanced Physics-Informed Neural Networks

Leon Herrmann

Stefan Kollmannsberger

Chair of Data Engineering in Construction

Bauhaus-Universität Weimar

Paris, February 2025

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

website

book