DSAIE– CEGM2003

BEAM project – Final presentation

Bart Slingerland

Daniel Agócs

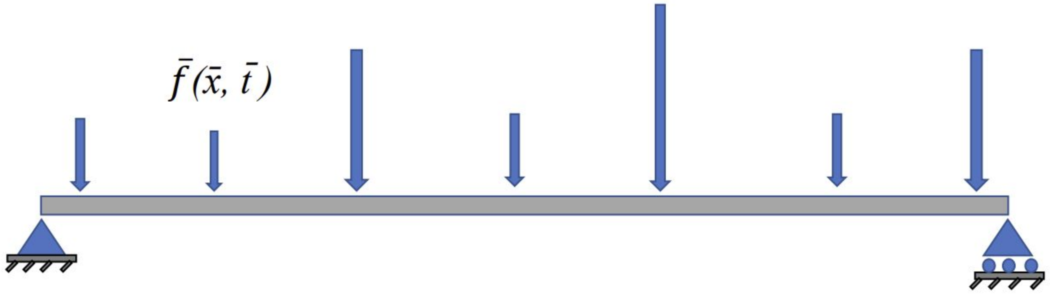
Evan Peeters

Koen de Rooij

Nicola Minikus

30/01/2026

Introduction



- **Project: AI for beams**
- **Regression problem**
- **Apply PINN :**

  Euler-Bernoulli beam theory

  Timonshenko beam theory

  Normalized equations□ EI = 1

Difference in loss functions (boundary conditions, governing equations)

Euler-Bernoulli

$$u(x,0) = \sin(x), \quad u_t(x,0) = 0,$$

$$u(0,t) = u(\pi,t) = u_{\mathrm{xx}}(0,t) = u_{\mathrm{xx}}(\pi,t) = 0.$$

$$f(x,t) = (1 - 16\pi^2)\sin(x)\cos(4\pi t)$$

$$u(x,t) = \sin(x)\cos(4\pi t)$$

Timoshenko

$$\theta(x,0) = \frac{\pi}{2}\cos(x) + \left(x - \frac{\pi}{2}\right), \quad \theta_t(x,0) = 0,$$

$$w(x,0) = \frac{\pi}{2}\sin(x), \quad w_t(x,0) = 0,$$

$$\theta(0,t) = \theta(\pi,t) = w(0,t) = w(\pi,t) = 0.$$

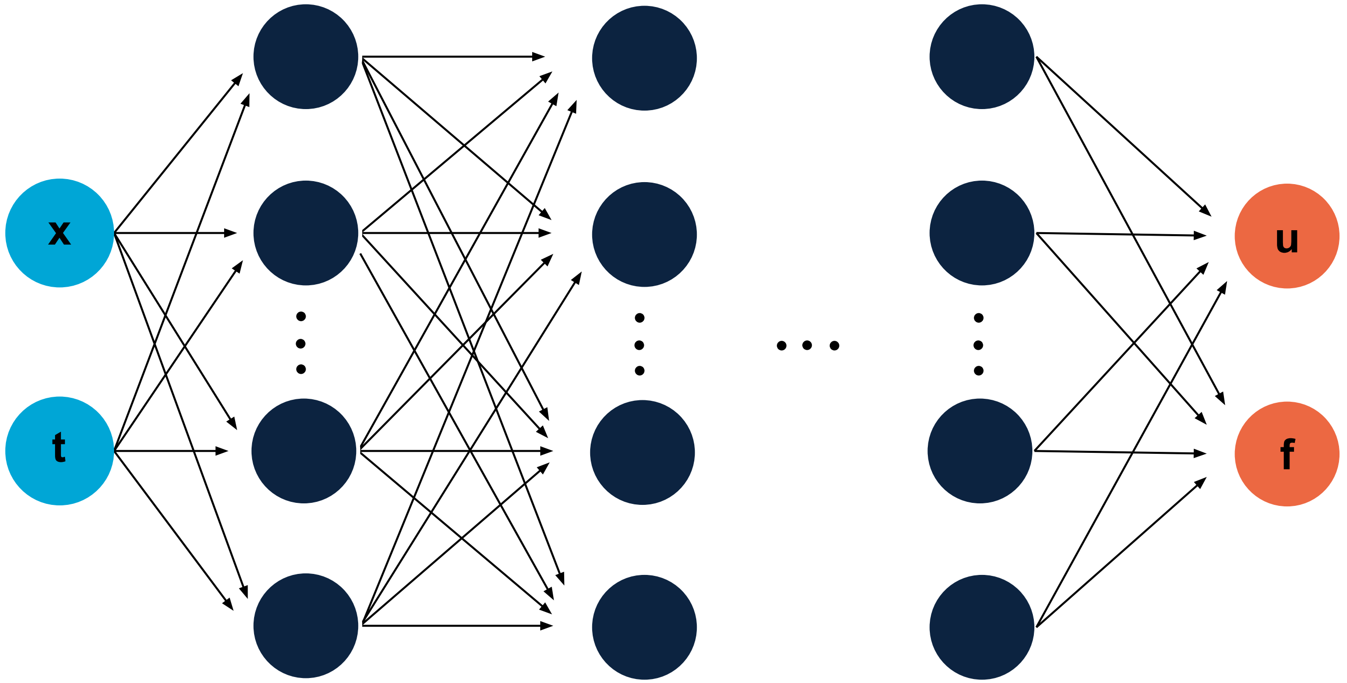
$$g(x,t) = \cos(t) - \frac{\pi}{2}\sin(x)\cos(t)$$

$$\theta(x,t) = \left(\frac{\pi}{2}\cos(x) + \left(x - \frac{\pi}{2}\right)\right)\cos(t),$$

$$w(x,t) = \frac{\pi}{2}\sin(x)\cos(t).$$

Forward problem

- Inputs : **x** (position along the beam) and **t** (time)
- Outputs : **u** (displacement profiles) and **f** (force function)



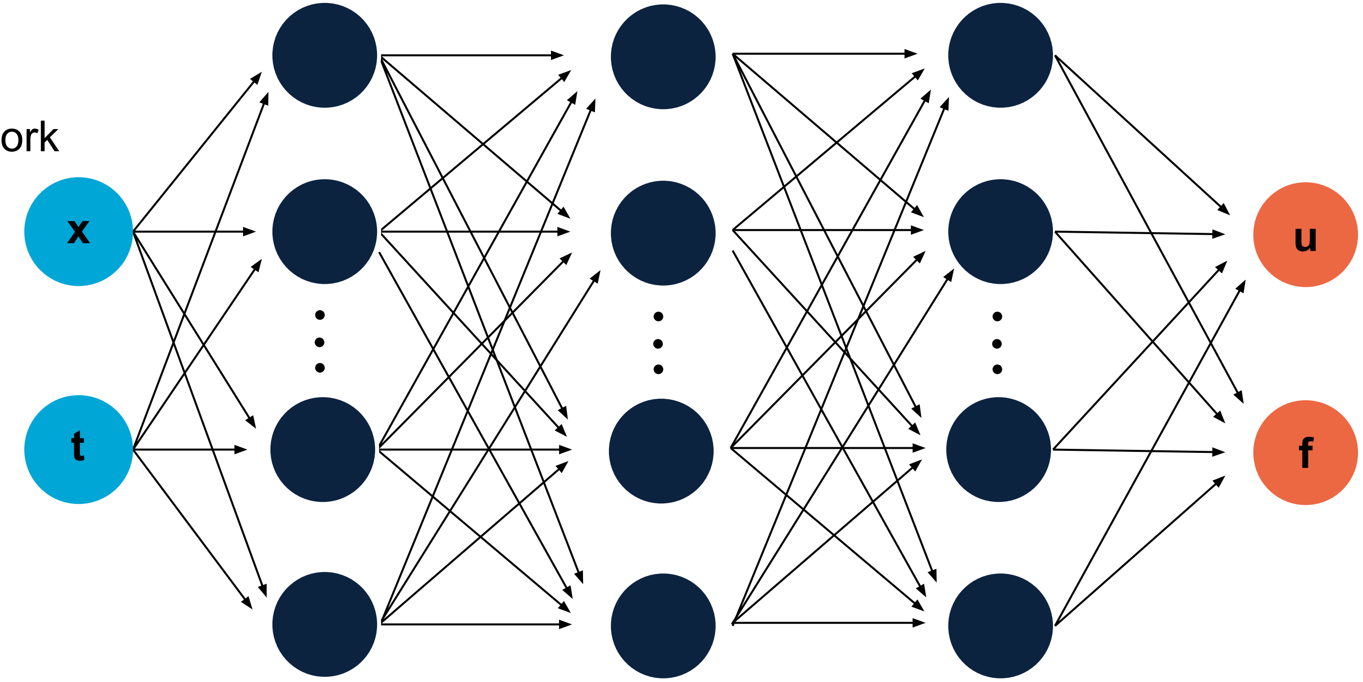
1 Deep learning implementation – Without PI

Architecture: Fully connected neural network

Optimal hyperparameter details:

- **Learning rate: 0.0001**
- **L1 Loss function**
- **AdamW optimizer**
- **ReLU activation function**
- **4 hidden layers**
- **128 neurons per layer**

```
Using params (MSE): {'hidden_size': 96, 'optimizer':
'AdamW', 'learning_rate': 0.0050691542944099315,
'activation': 'ReLU', 'loss': 'MSE'}
```

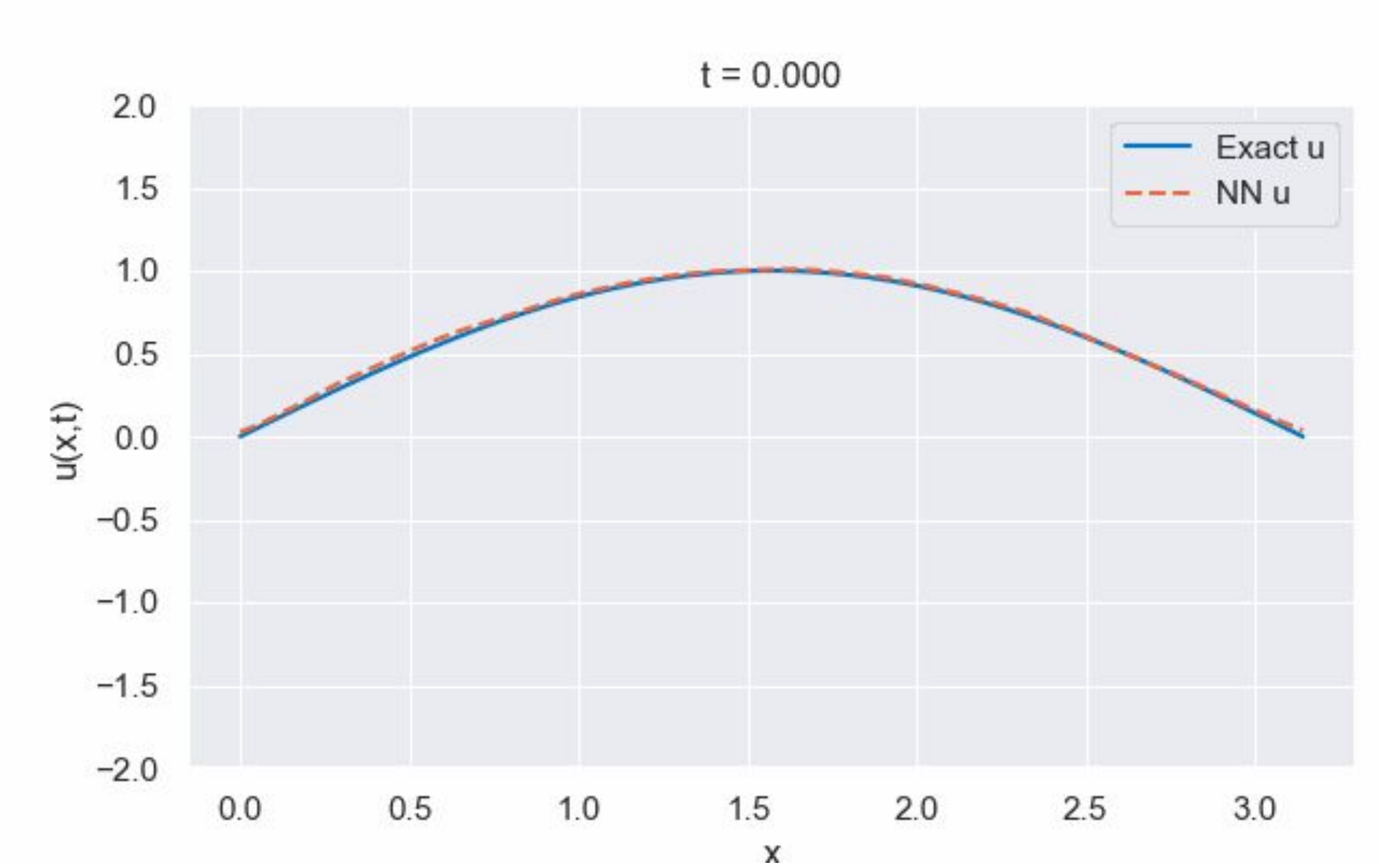


Recap

NN without PI □ Decent interpolation, poor extrapolation, **large training dataset**

```
Errors for L1 loss model:
error (f):   1.397341%
error (u):   1.389134%

Errors for L2 loss model:
error (f):   0.785919%
error (u):   0.787875%
```

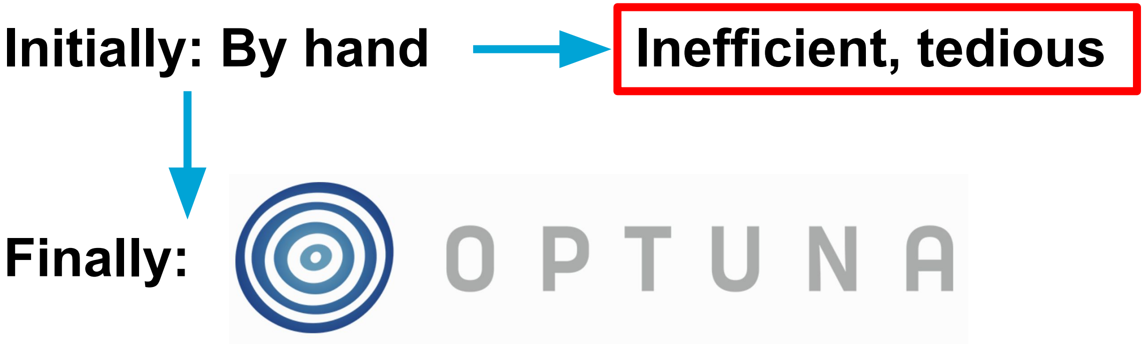


Hyperparameters

Hyperparameters to analyse:

- Learning rate
- Loss function
- Optimizer
- Number of layers
- Number of hidden features
- PINN loss weights

Hyperparameter tuning



Initially: By hand → Inefficient, tedious

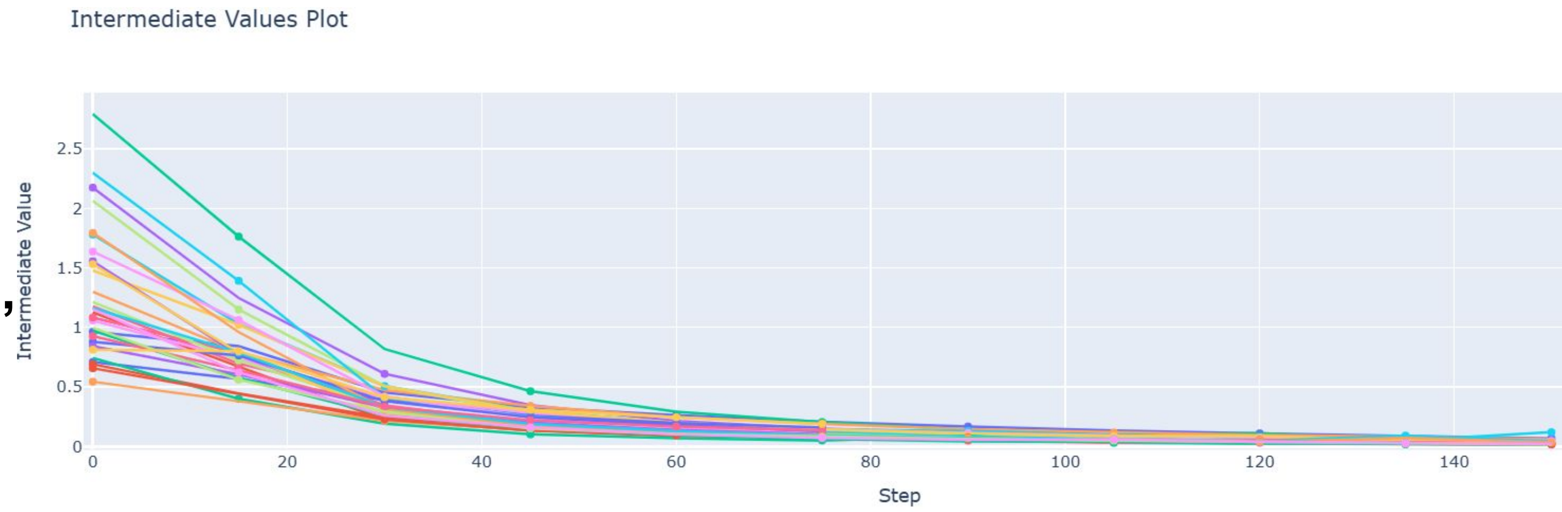
Finally:

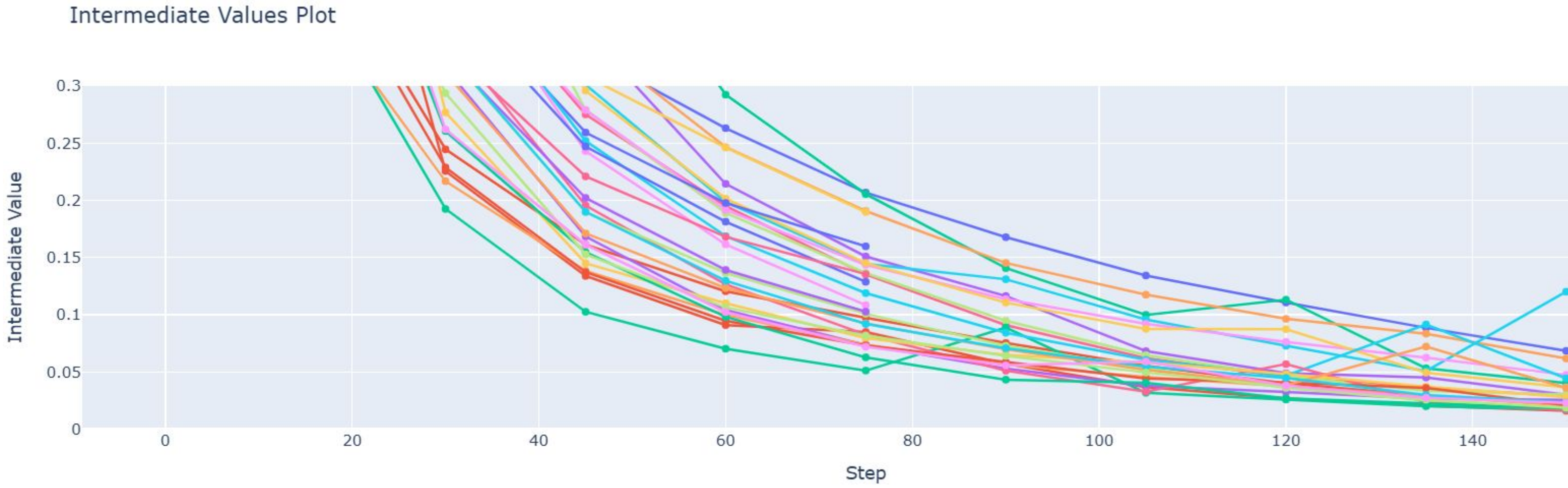
Hyperparameter optimizer framework

Optuna

Optuna workflow:

- Objective function: Minimise validation loss
- “Suggesting hyperparameters”
- Studies and trials
- Pruning
- Curse of dimensionality





2 PINN framework: Euler-Bernoulli Beam Model

Total loss: λ_{Data}* Data loss + λ_{IC}* IC loss + λ_{BC}* BC losses (left, right) + λ_{Ph}* Physics loss

Optuna hyperparameter optimisation in 2 steps:

1. Hidden features, number of layers, learning rate
2. Loss function weights

Best model:

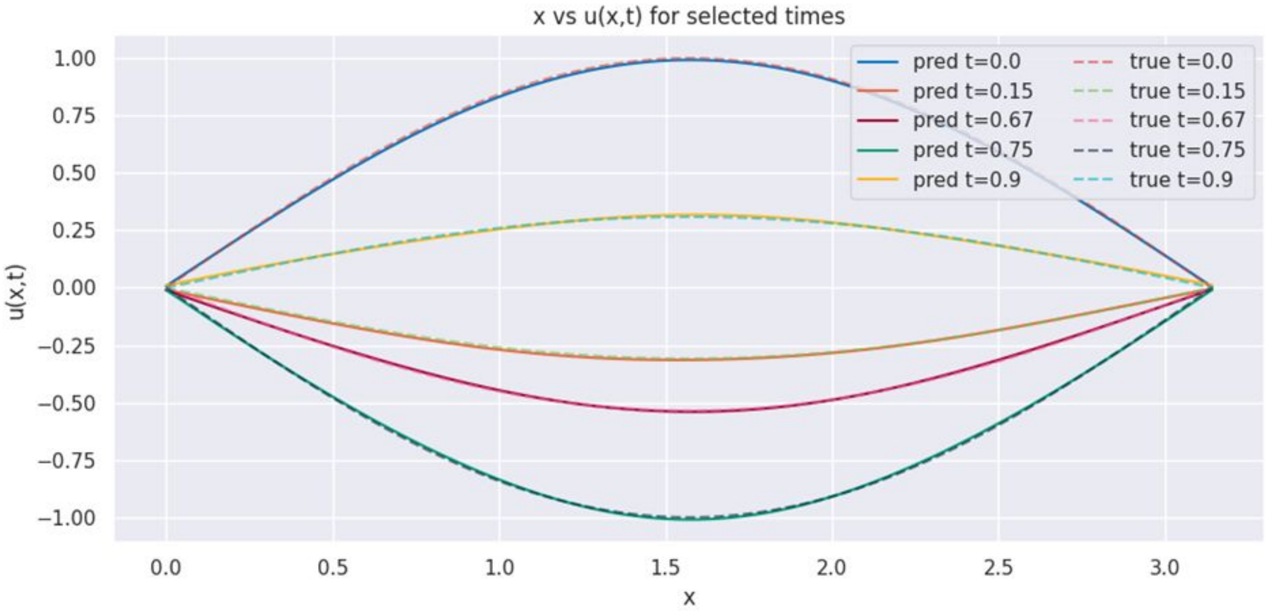
L_1 Loss, hidden features = 400, layers = 1, learning rate = 0.001156,
λ_{Data} = 0.982, λ_{IC} = 0.993, λ_{BC} = 1.694, λ_{Ph} = 0.998

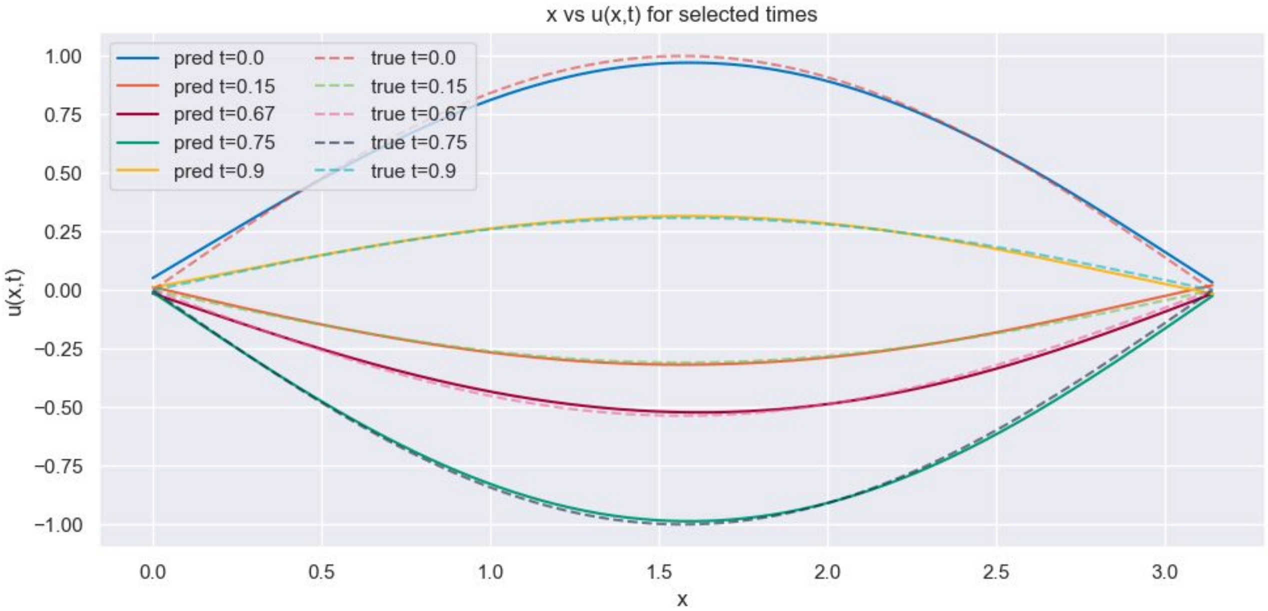
Relative errors against analytical solution $u(x,t) = \sin(x) \cdot \cos(4\pi t)$

- L_1: 1.12%
- L_2: 2.22%

2 PINN framework: Euler-Bernoulli Beam Model

Displacement results



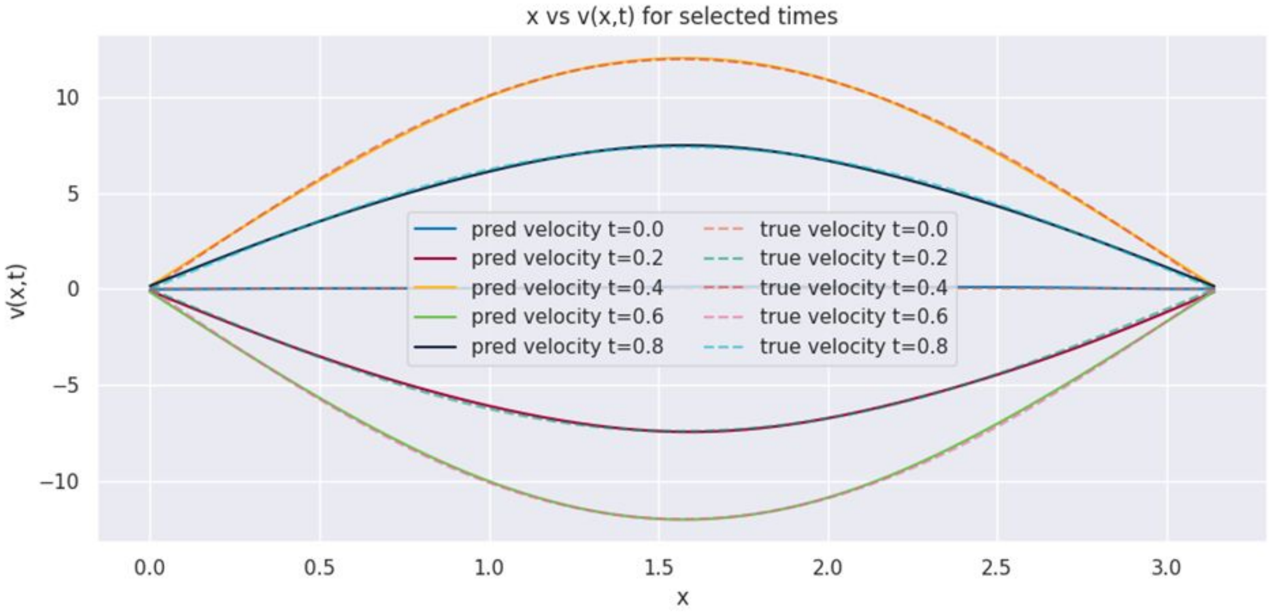


L_1 model: relative error 1.12%

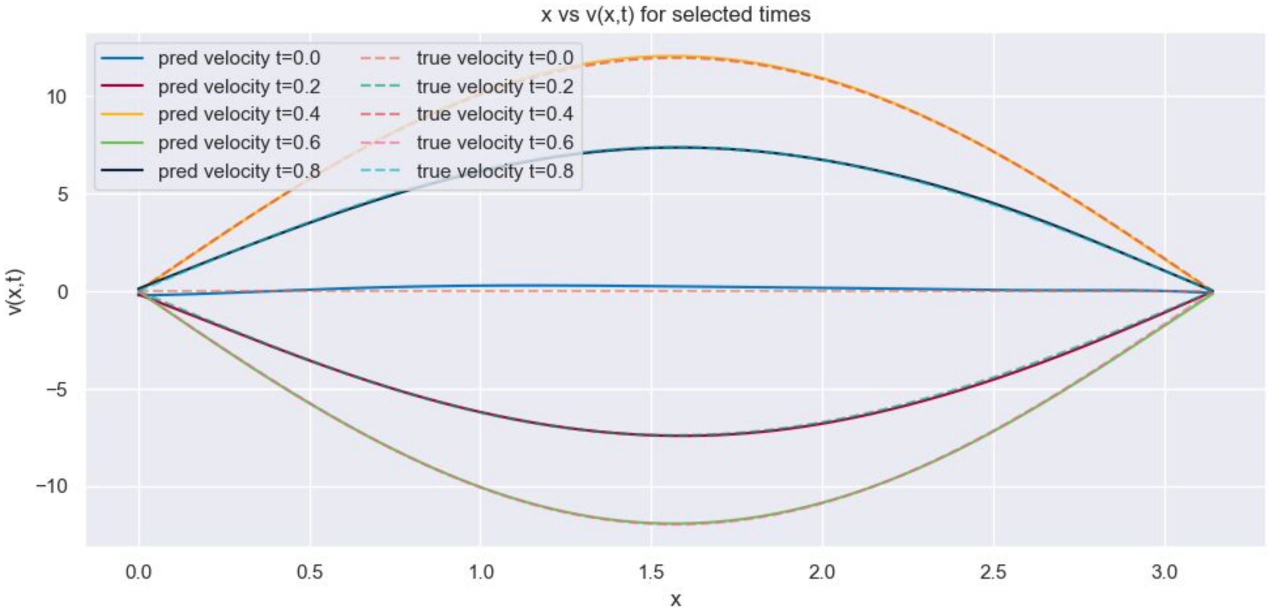
L_2 model: relative error 2.22%

2 PINN framework: Euler-Bernoulli Beam Model

Velocity results



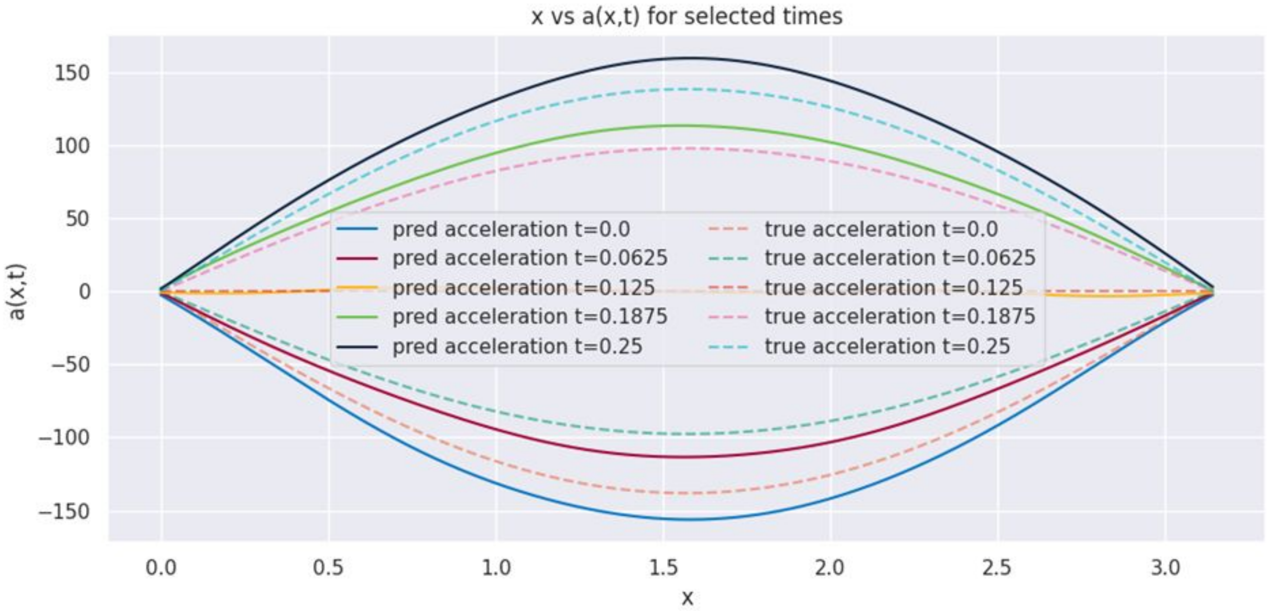
L_1 model: relative error 1.20%



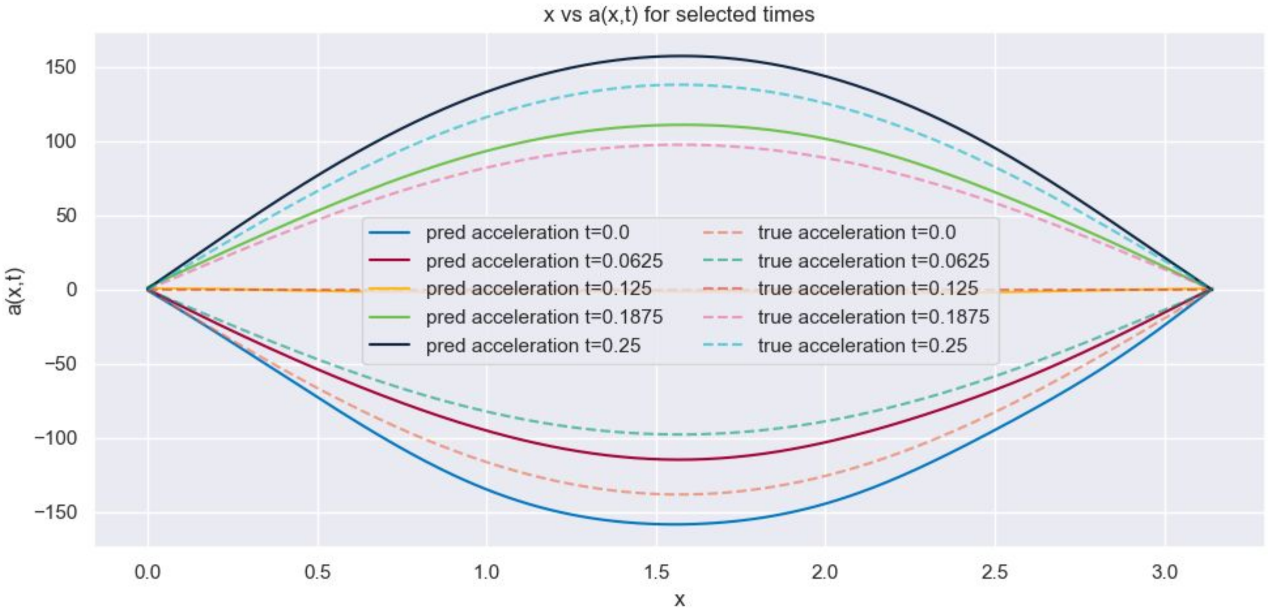
L_2 model: relative error 1.22%

2 PINN framework: Euler-Bernoulli Beam Model

Acceleration results



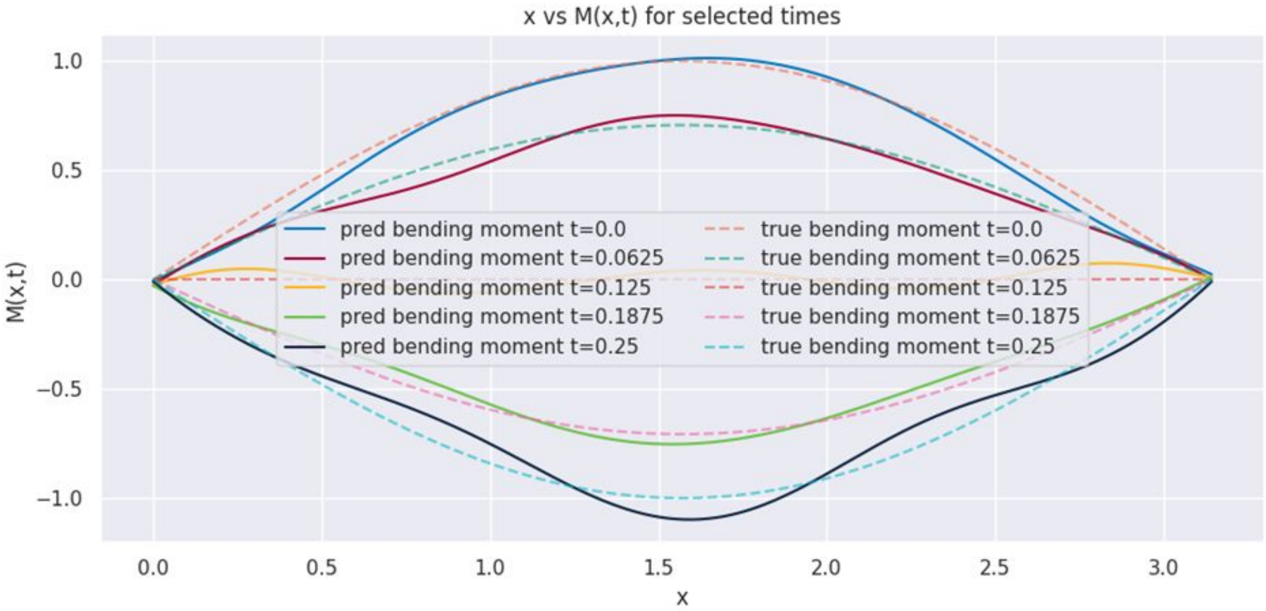
L_1 model: relative error 14.47%



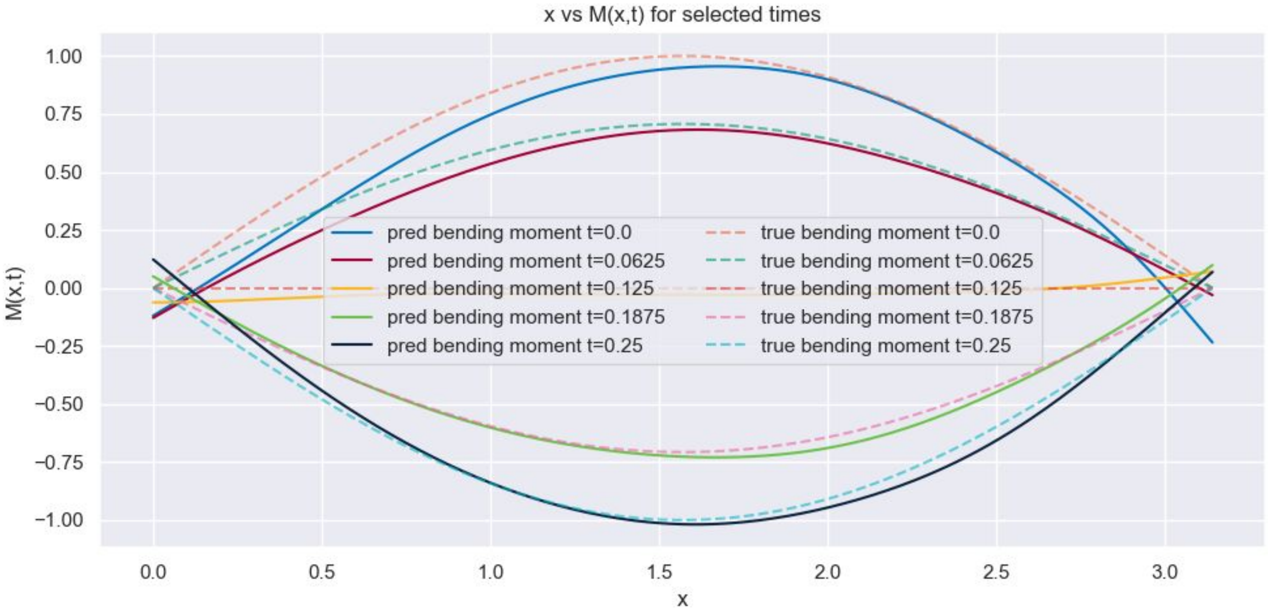
L_2 model: relative error 14.69%

2 PINN framework: Euler-Bernoulli Beam Model

Bending moment results



L_1 model: relative error 10.15%



L_2 model: relative error 11.56%

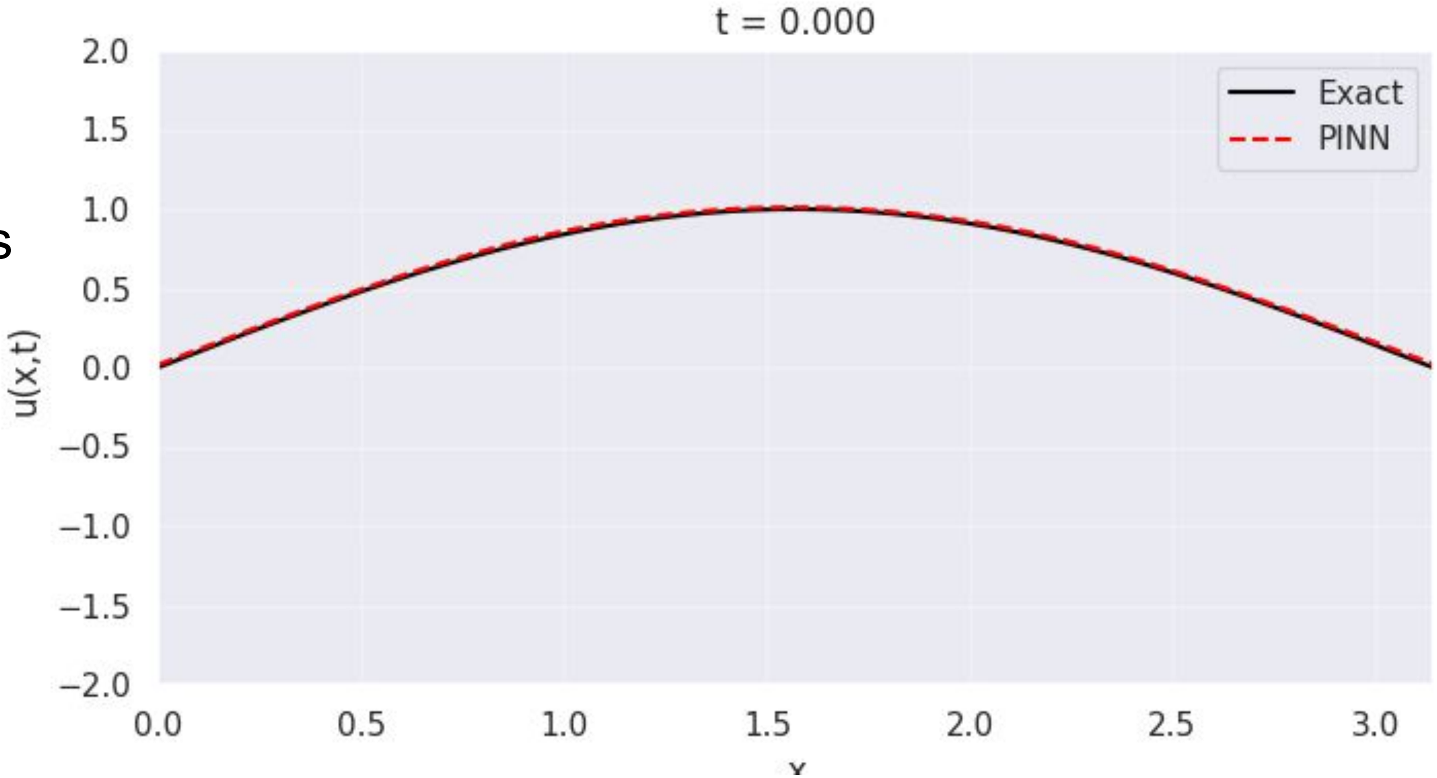
2 PINN framework: Euler-Bernoulli Beam Model

Training data: 1 second
Physics loss and boundary condition loss : 5 seconds

Further improvement:

- More complex architecture
- More epochs
- Adapt loss function weights

Already better than without PINN!



3 PINN framework: Timoshenko Beam Model

Also L_1 and L_2 norm

L_1 relative errors:

- u – 1,28%
- Θ – 2,05%

L_2 relative errors:

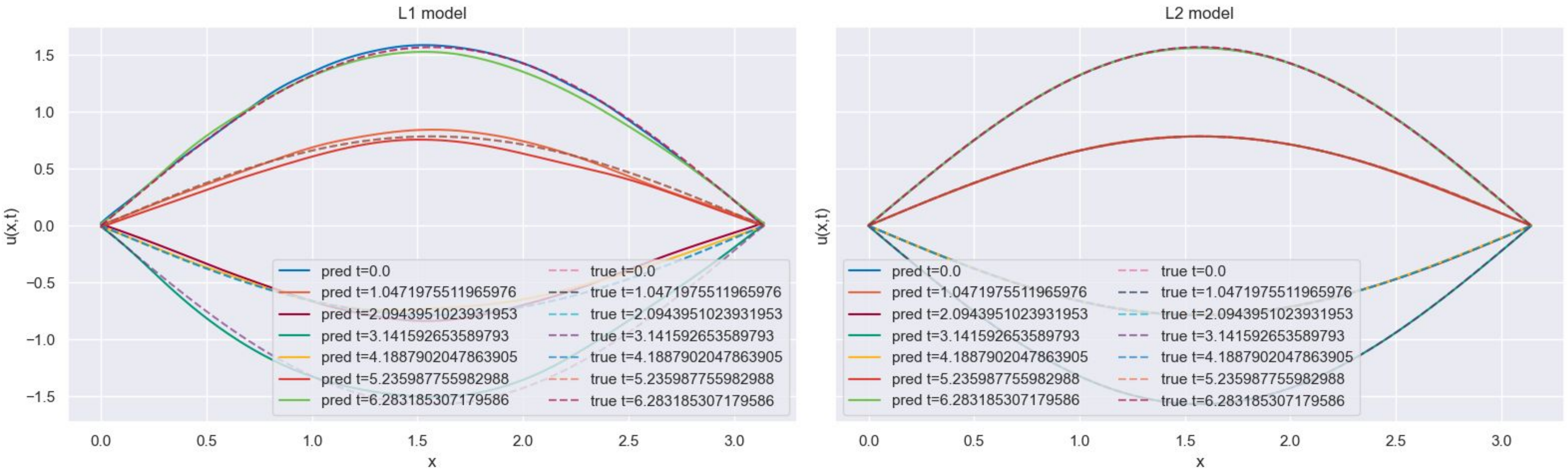
- u – 0,162%
- Θ – 0,276%

→ **Significant differences!**
(visually as well)

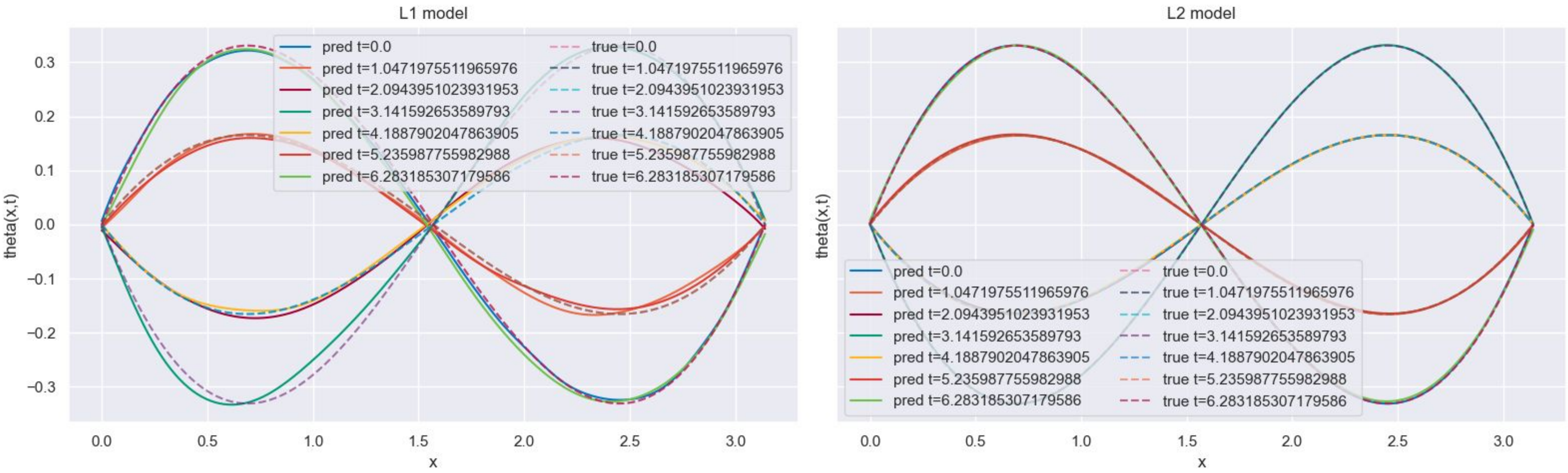
Optuna optimisation →

L_1 norm	L_2 norm
Adam	Adam
Hidden features = 250	Hidden features = 100
Hidden layers = 3	Hidden layers = 2
Learning rate ≈ 0.008	Learning rate ≈ 0.017
λ_{data} ≈ 0.480	λ_{data} ≈ 0.963
λ_{IC} ≈ 0.247	λ_{IC} ≈ 0.71
λ_{BC} ≈ 0.133	λ_{BC} ≈ 0.815
λ_{Ph} ≈ 0.490	λ_{Ph} ≈ 0,857

3 PINN framework: Timoshenko Beam Model

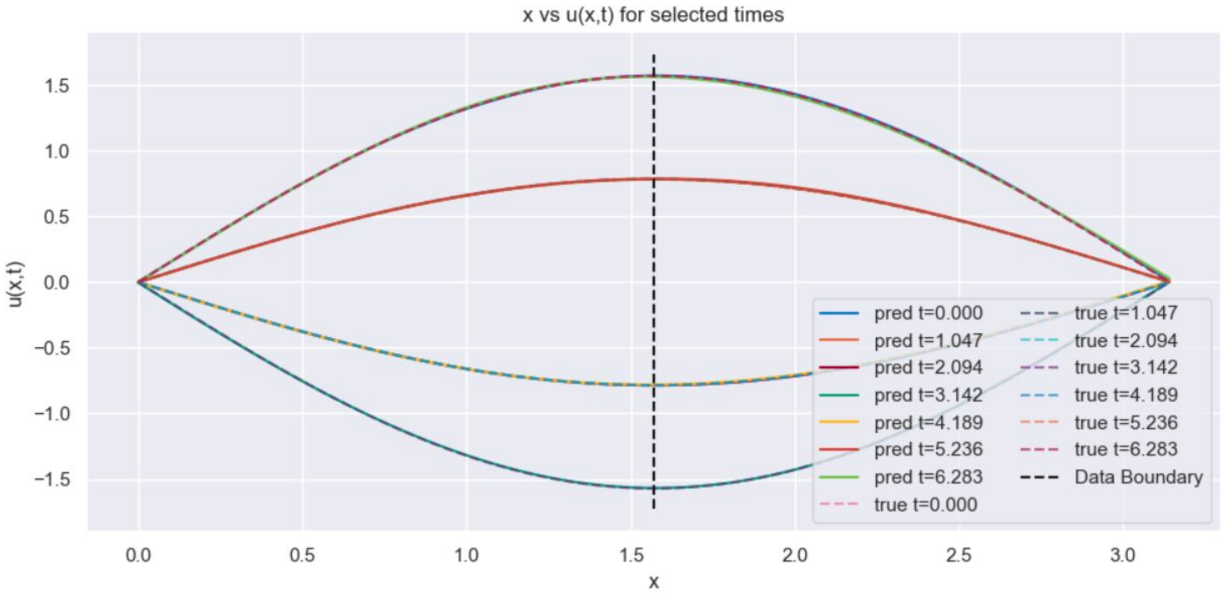


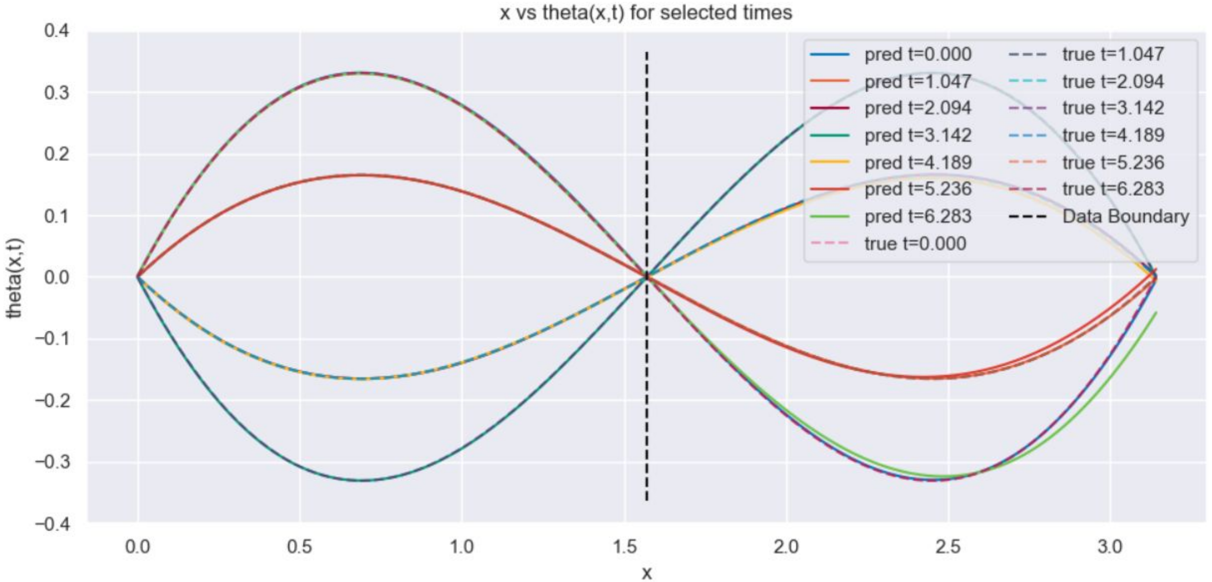
3 PINN framework: Timoshenko Beam Model



3 PINN framework: Timoshenko Beam Model extrapolation

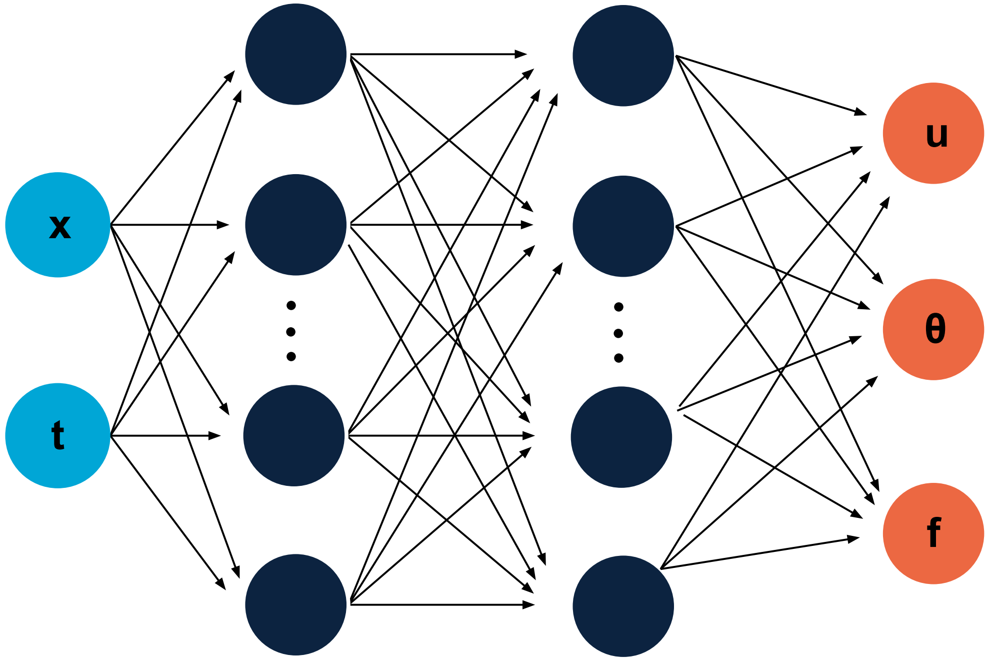
- Data for the left half
- Relative errors:
  - u = 0.53%
  - Theta = 2.29%





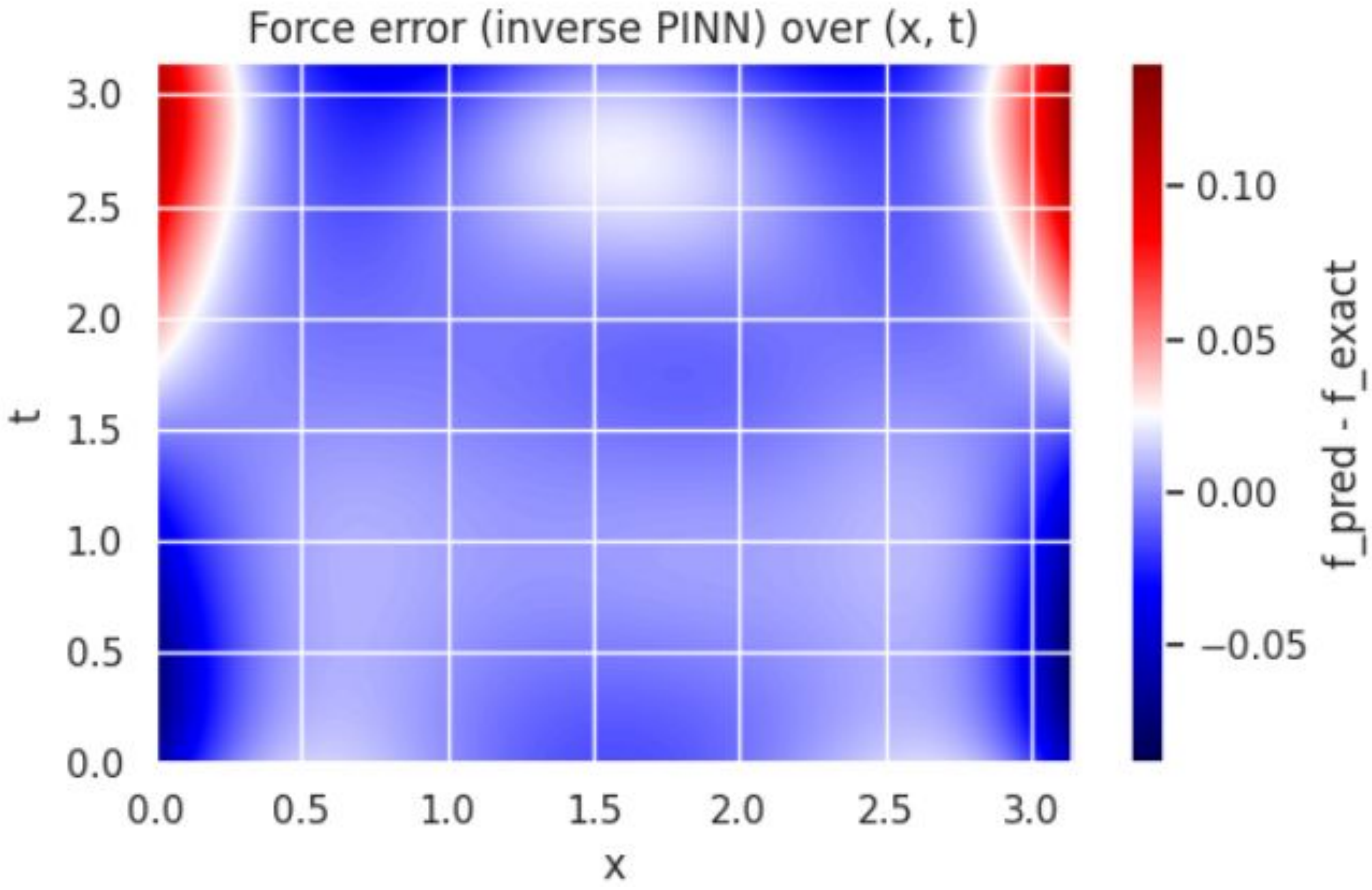
4 Inverse model

- **Force function** unknown, **u** and **θ** known at n amount of locations.
- Inputs : **x** (position along the beam) and **t** (time)
- Outputs : **u** (displacement), **θ** (rotation) and **f** (force function)
- f_pred instead of f_exact
- L-BFGS solver instead of ADAM
- L-BFGS better for little data

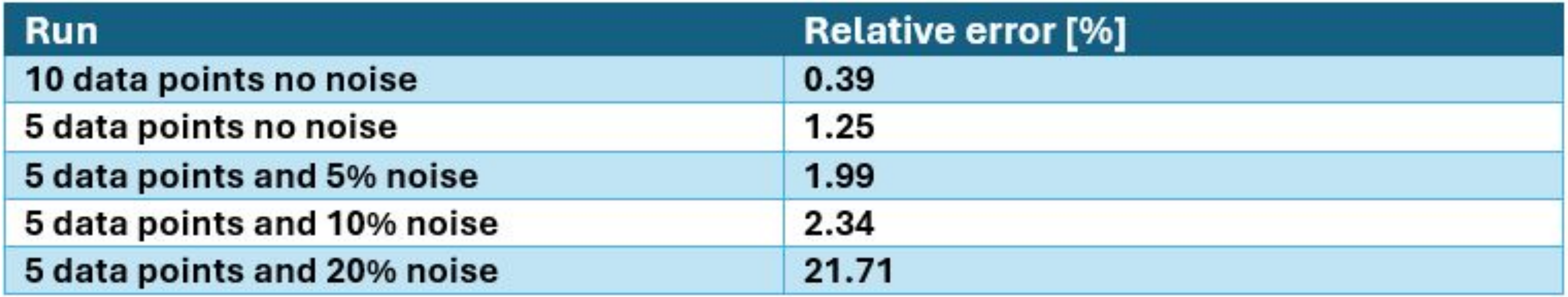


4 Inverse model

- 10 data points no noise
- Relative error = 0.39%



4 Inverse model



Run	Relative error [%]
10 data points no noise	0.39
5 data points no noise	1.25
5 data points and 5% noise	1.99
5 data points and 10% noise	2.34
5 data points and 20% noise	21.71

Discussion

Advantages of PINN framework compared to FNN:

- Less training data
  - FNN: ~10000
  - PINN timoshenko: ~300
- Better extrapolation due to physics and boundary conditions
- Solving inverse problems

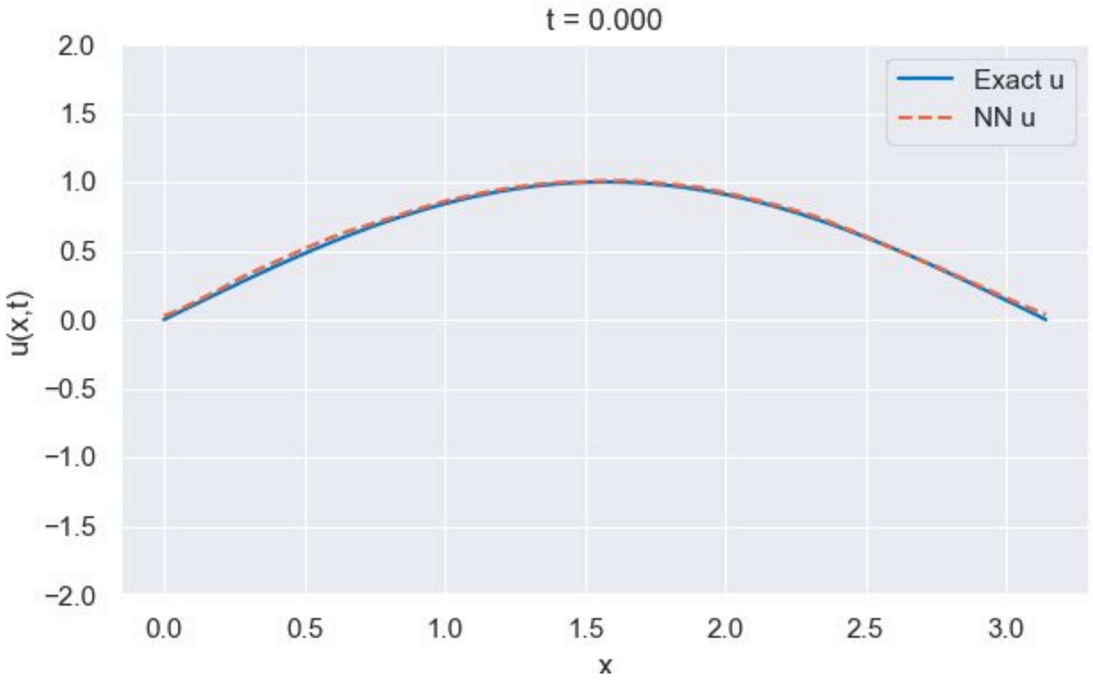
EB-beam vs Timoshenko-beam

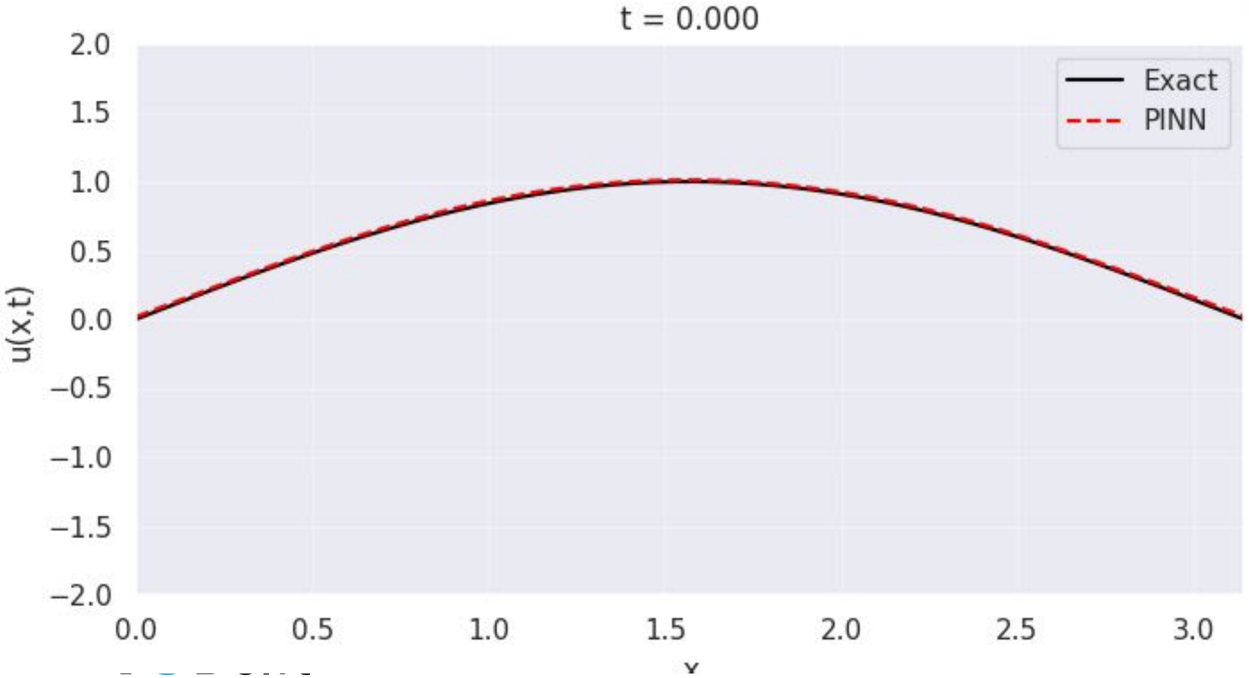
- Timoshenko achieves lower error percentages
- EB-beam performs better with L1 loss
- Timoshenko beam performs better with L2 loss
- Timoshenko performs better in general

Further exploration

- Finding the limits for simplicity of the model and training
- Making a working PINN model without data
- Use real world data for training a model
- More complex structural members
- Different boundary conditions

The End


t = 0.000
Exact u
NN u
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
u(x,t)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x


t = 0.000
Exact
PINN
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
u(x,t)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x