Using PINN’s to simulate beam behaviour

By Amco, Bastiaan, Daan, Hidde and Jur
BEAM2

Why we use PINNs?

Main goal: Prediction of complex engineering problems

Key advantages:

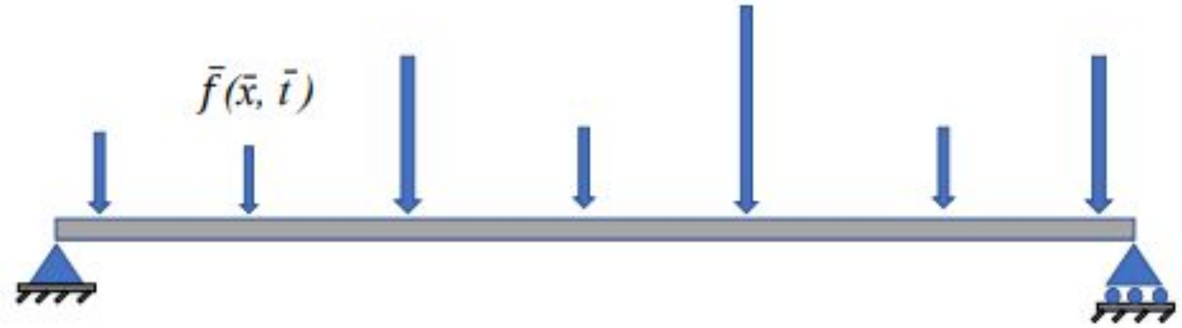
- Data-driven learning
- Reduced computational costs

Challenges:

- Incorporating physics
- Loss function design
- Training stability

Problems

- Euler-Bernoulli forward problem
  -> Build PINN to accurately predict deflection
  -> Check accuracy of velocity,
  acceleration and bending moment
- Timoshenko forward problem
  -> Build and train PINN to predict θ and w
- Timoshenko inverse problem
  -> Build and train PINN to predict f alongside
  θ and w using observations



Euler–Bernoulli beam

- Non-dimensionalized
- Partial Differential Equation (PDE)
- 4 boundary and 2 initial conditions
- Dynamic forcing term
- Goals:
  - Part 1.1 and 1.2: Predict deflection and obtain high accuracy
  - Part 1.3: Bending moment, velocity, acceleration

$$\rho A \frac{\partial^2 u}{\partial t} + EI \frac{\partial^4 u}{\partial x^4} = f(x,t)$$

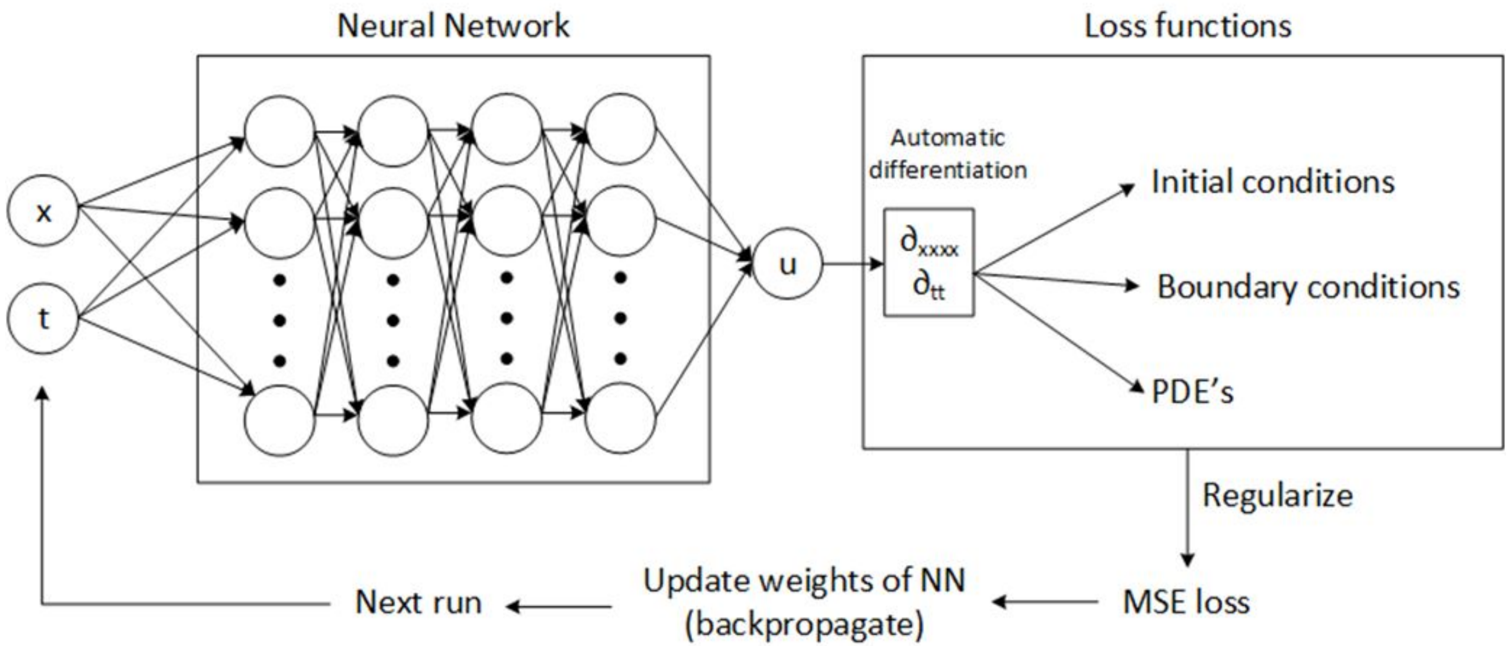
$$u(x,0) = \sin(x), \quad u_t(x,0) = 0$$

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

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

Method

- MLP
- tanh activation function
- 4 hidden layers
- 25 neurons per layer
- Xavier initialisation
- LBFGS optimiser
- Loss = λ_1* initial + λ_2* boundary + λ_3*physics

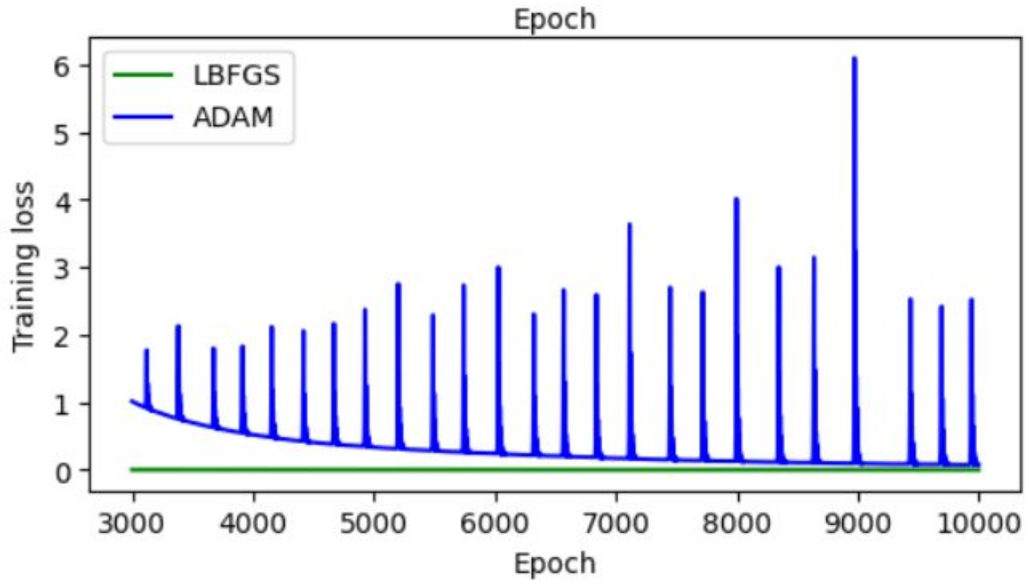


Training input

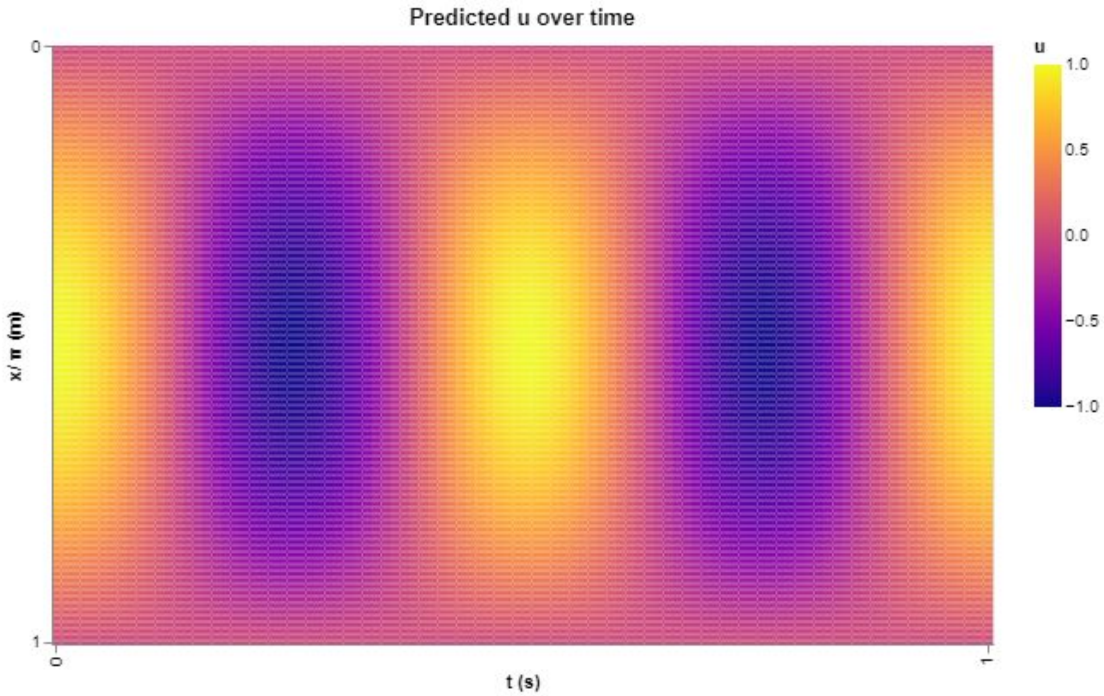
- Initial points: 500
- Left boundary points: 500
- Right boundary points: 500
- Residual points: 2000

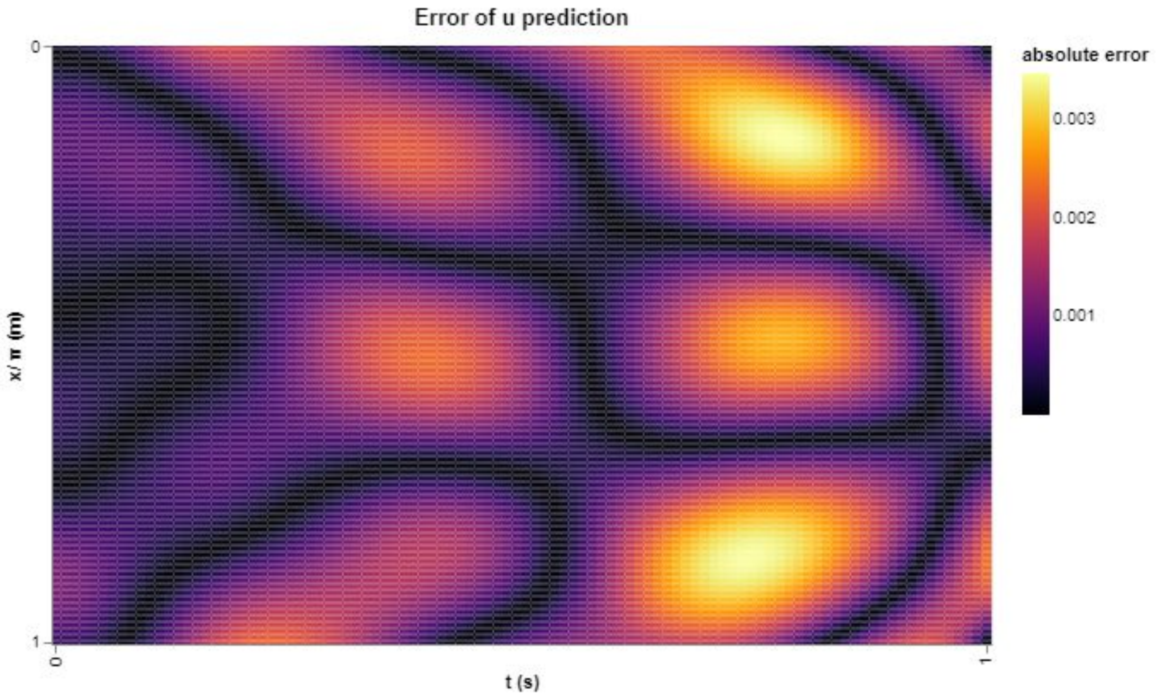
Hyperparameters

- Activation function
- Optimiser
- Epochs 14000
- Learning rate 0.1
- $\lambda_1 = 1, \quad \lambda_2 = 1, \quad \lambda_3 = 0.1$

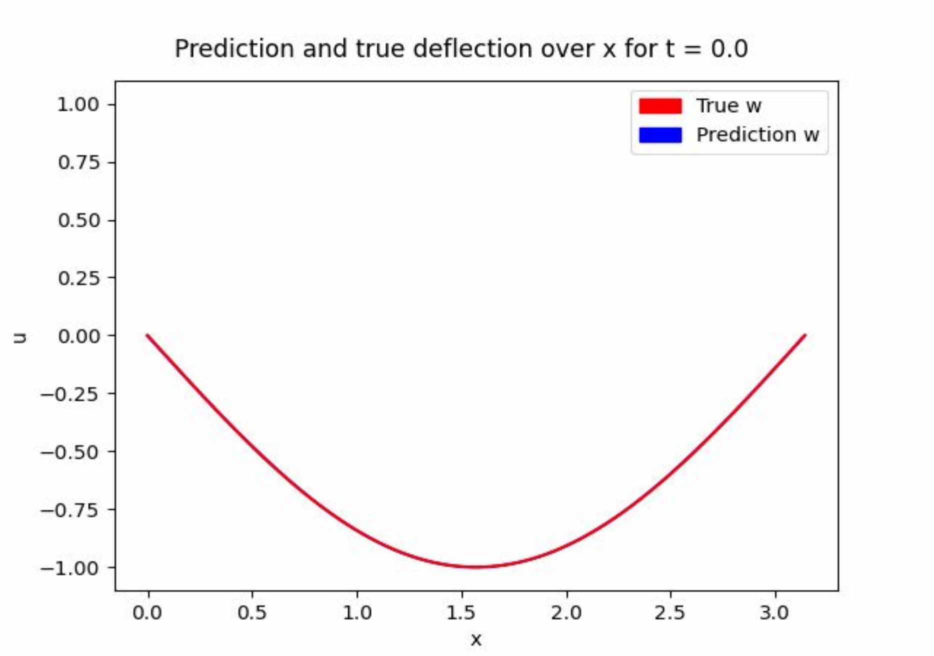


Results


Predicted u over time
x/ π (m)
t (s)
0
1
0
1
u
1.0
0.5
0.0
−0.5
−1.0


Error of u prediction
x/ π (m)
t (s)
0
1
0
1
absolute error
0.003
0.002
0.001

Results



Prediction and true deflection over x for t = 0.0
True w
Prediction w
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
u
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x

Differentiated values

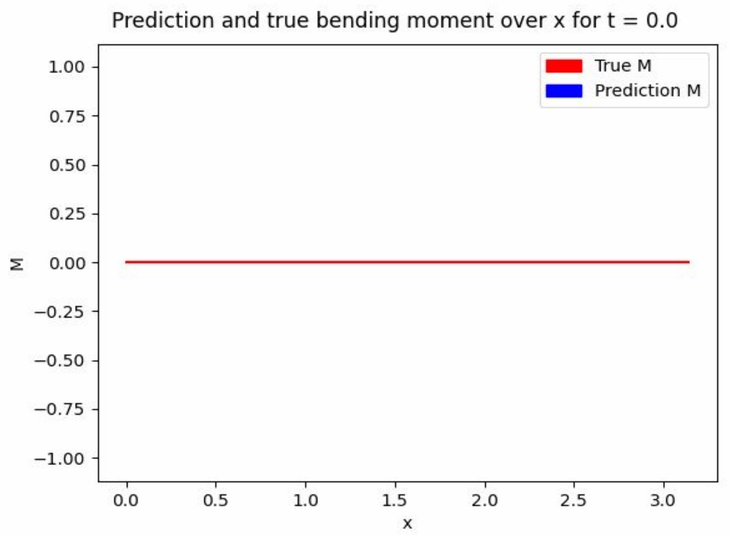
- Bending moment (u_{xx})
- Velocity (u_t)
- Acceleration (u_{tt})
- Relative error

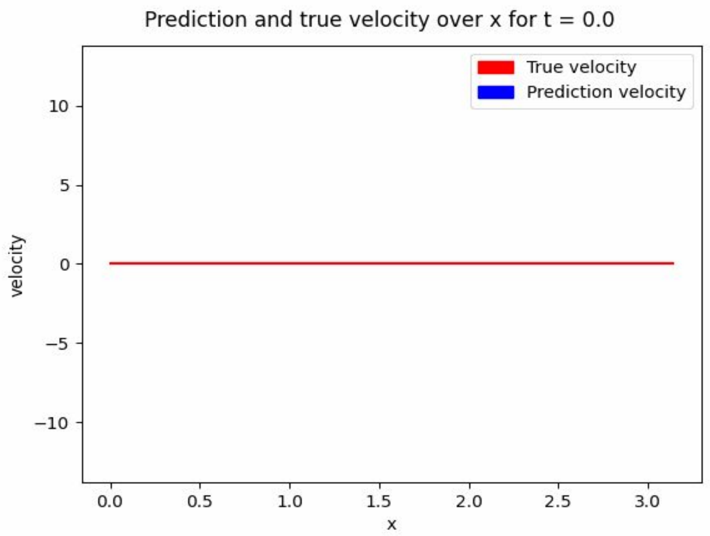
$$\epsilon_{\mathrm{re}}[\%] = \frac{\frac{1}{N}\sum_{i=0}^{n}(u_{\mathrm{true}} - u_{\mathrm{pred}})^2}{\frac{1}{N}\sum_{i=0}^{n} u_{\mathrm{true}}^2} \times 100\%$$

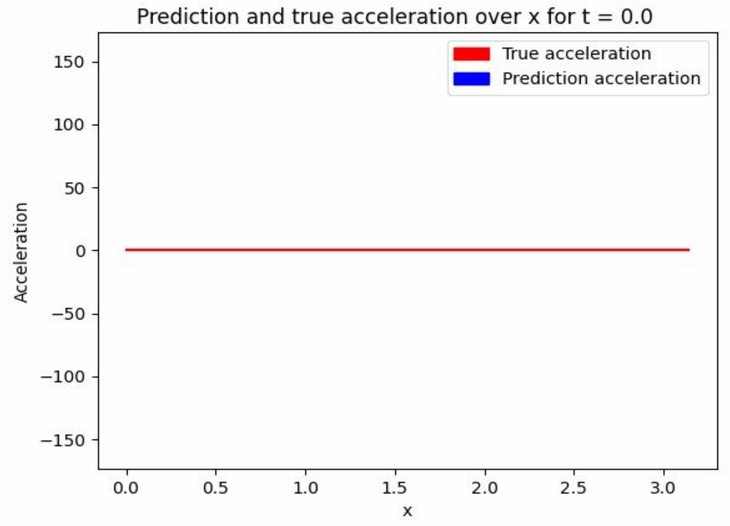
$$u_{xx}(x,t) = -\sin(x)\cos(4\pi t)$$

	Relative error
Deflection	0.00015%
Bending moment	0.02649%
Velocity	0.00011%
Acceleration	0.00009%

Results


Prediction and true bending moment over x for t = 0.0
True M
Prediction M
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
M
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x


Prediction and true velocity over x for t = 0.0
True velocity
Prediction velocity
10
5
0
−5
−10
velocity
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x


Prediction and true acceleration over x for t = 0.0
True acceleration
Prediction acceleration
150
100
50
0
−50
−100
−150
Acceleration
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x

Limitation of PINNs

- Spectral bias of NN
- Problems with high frequencies

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

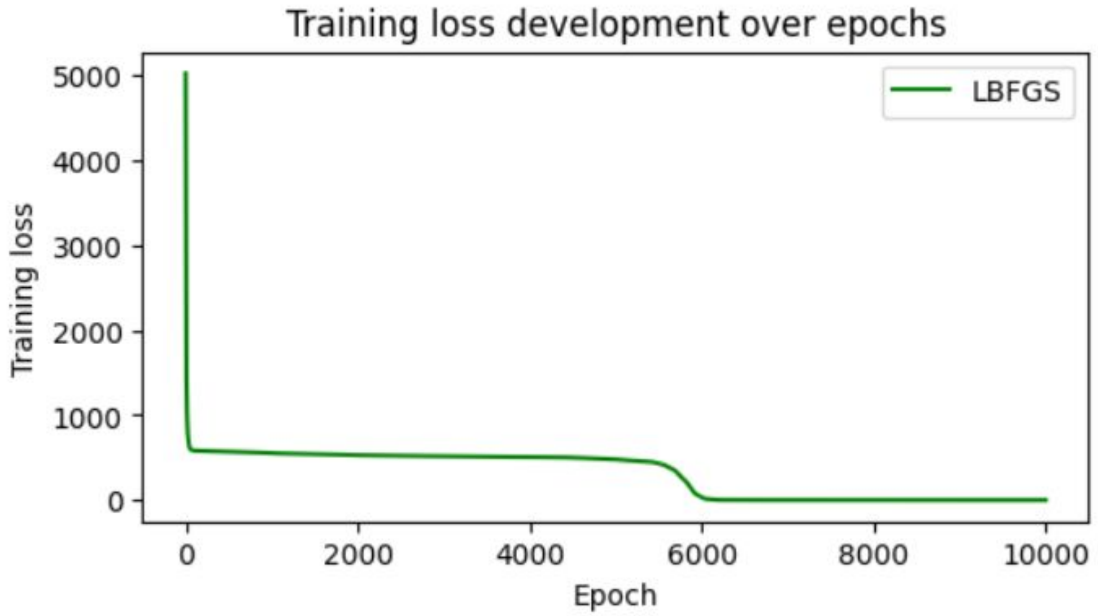
$$\downarrow$$

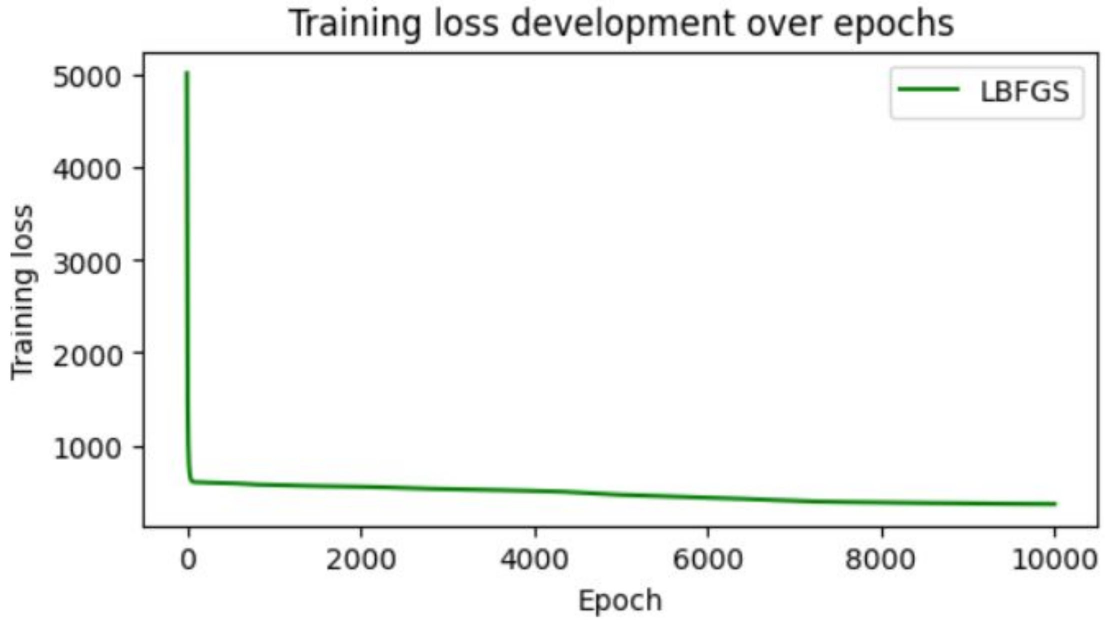
$$f(x,t) = (1 - 16\pi^2)\sin(x)\cos(10\pi t)$$

$$\downarrow$$

- Loss after 10000 epochs: 365

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

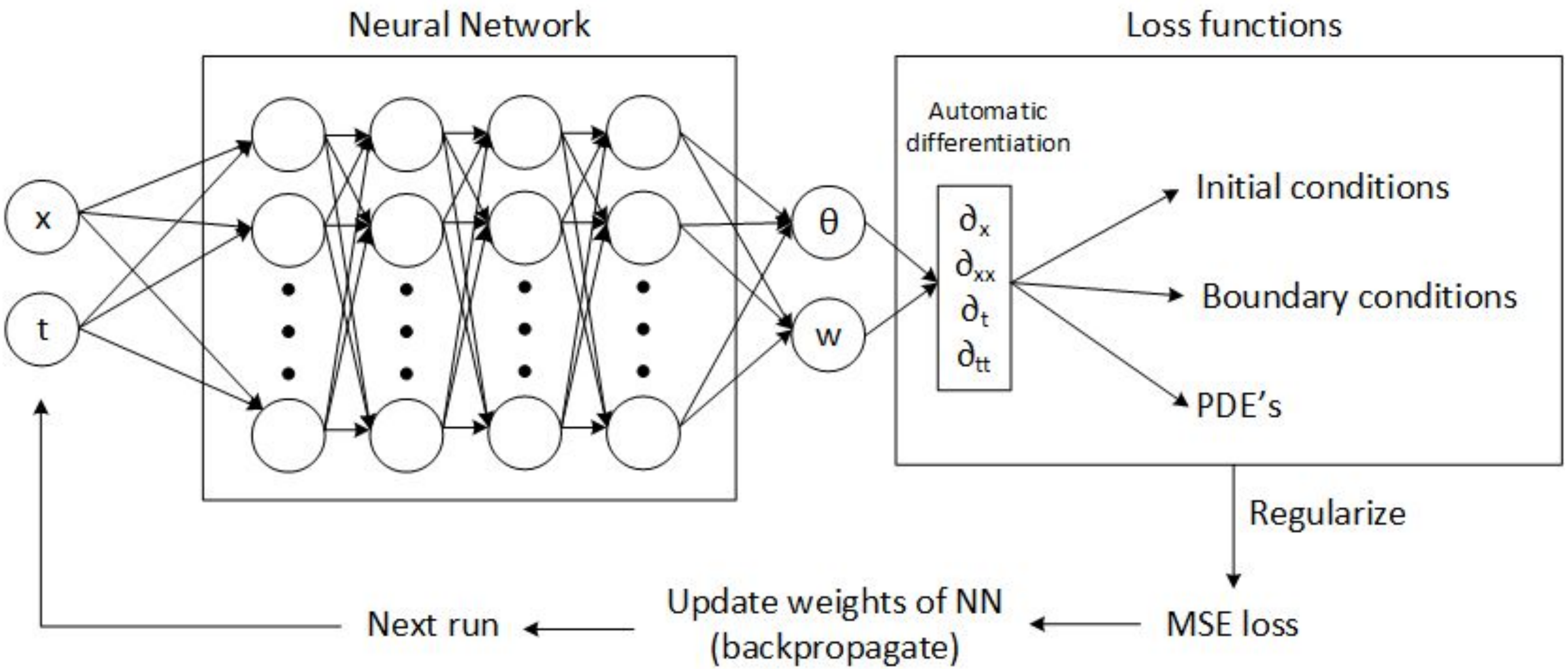




Timoshenko beam

- Part 1 (forward problem):
    - Predict exact solutions for rotation and deflection
- Part 2 (inverse problem):
    - Use previously trained PINN to generate “sensor observations” for rotation and deflection
    - Inversely predict forcing on the beam, using these observations
    - Optimise the amount of sensors and their locations

Method (forward problem)



Method (forward problem)

- Two coupled Partial Differential Equations (PDE's)
- Non-dimensionalized
- Eight boundary and initial conditions
- Dynamic forcing term

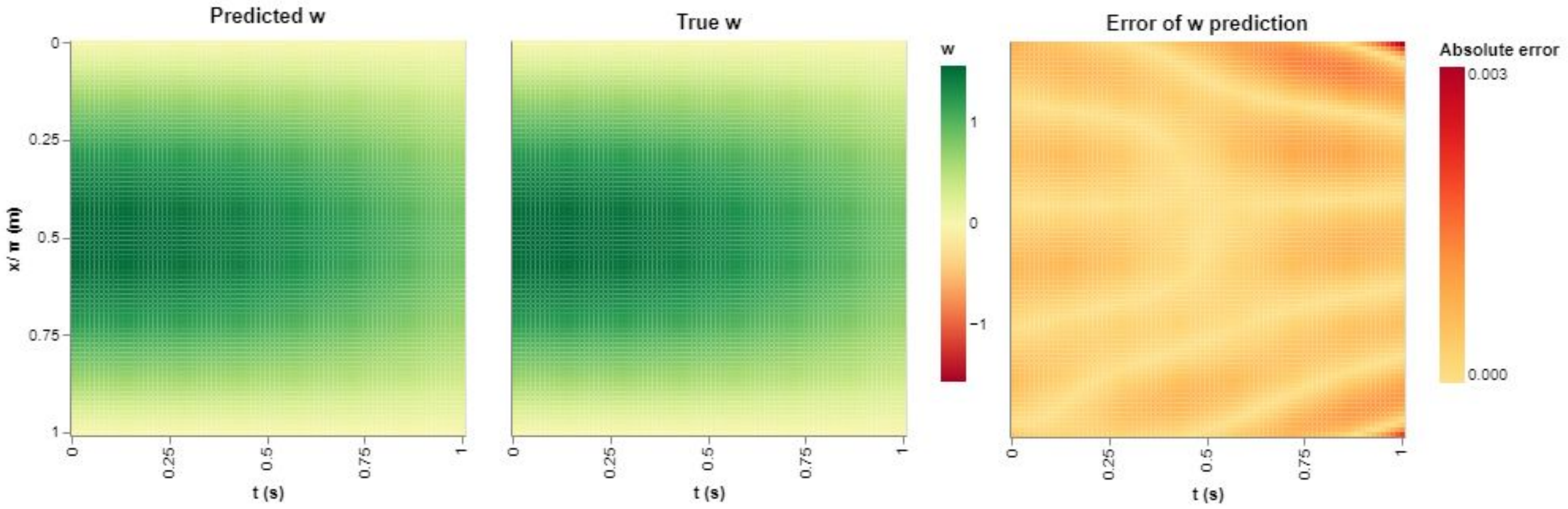
$$\rho I\theta_{\mathrm{tt}} - EI\theta_{\mathrm{xx}} - kAG(w_x - \theta) = 0$$
$$\rho A w_{\mathrm{tt}} - kAG(w_{\mathrm{xx}} - \theta_{\mathrm{x}}) = g(x,t),$$

Hyperparameters

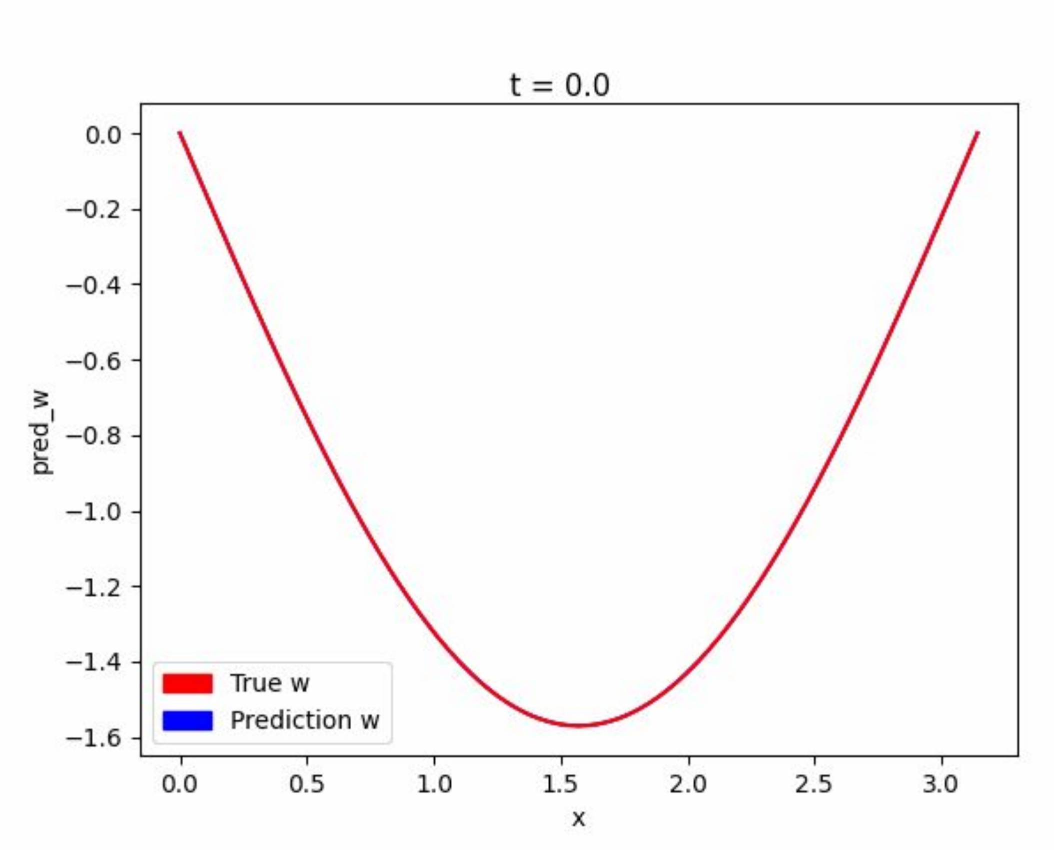
- MLP (4 layers, 20 features)
- Activation function: Hyperbolic tangent
- Optimiser: LBFGS
- Epochs 10000
- Learning rate 0.1
- $\lambda_1 = 1$, $\lambda_2 = 1$, $\lambda_3 = 0.1$

$$Loss = \lambda_1 * Loss_{BC} + \lambda_2 * Loss_{IC} + \lambda_3 * Loss_{physics}$$

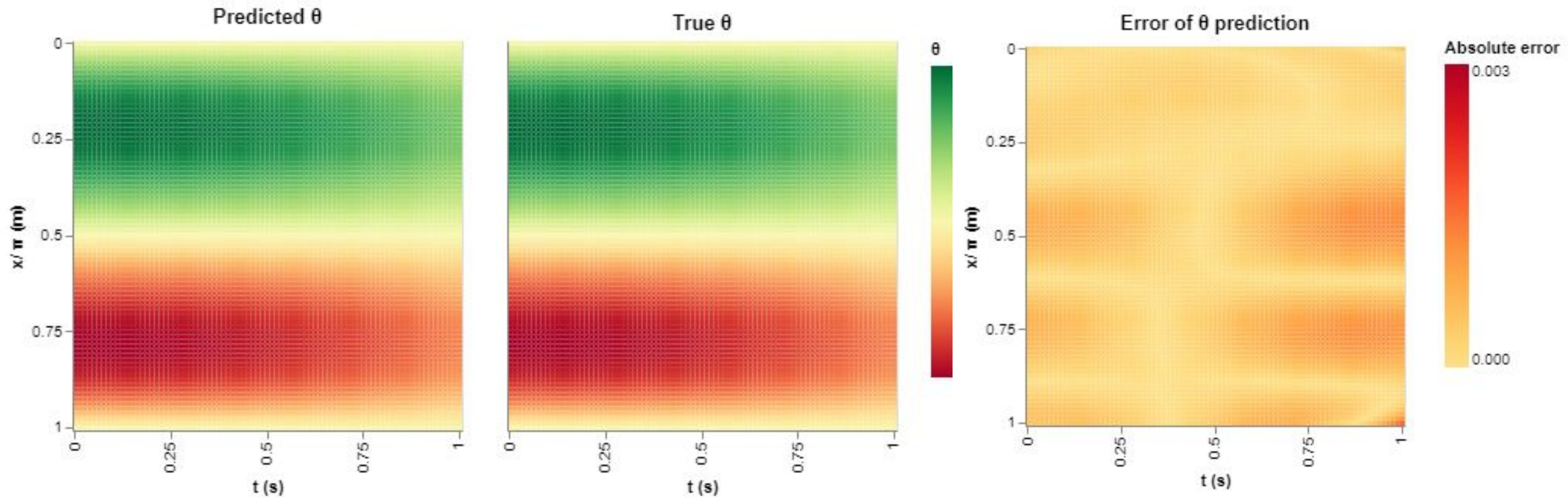
Results Timoshenko forward



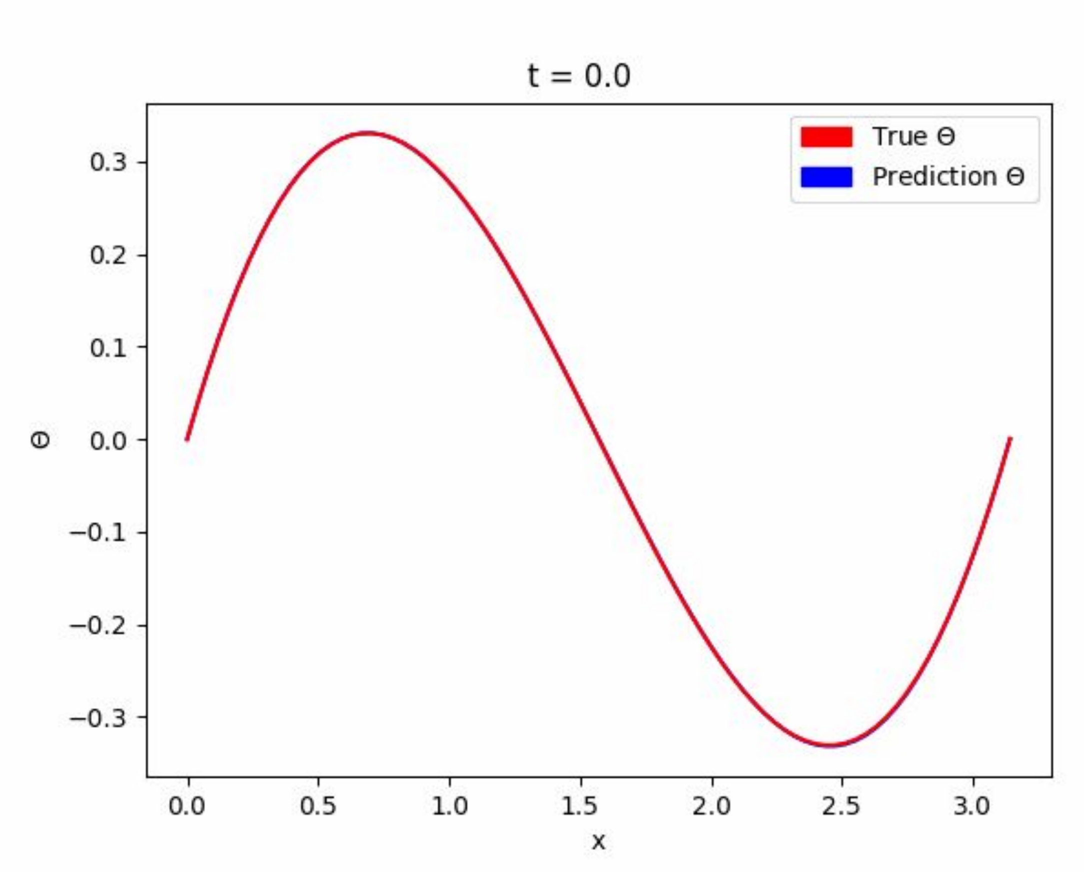
Results Timoshenko forward



Results Timoshenko forward

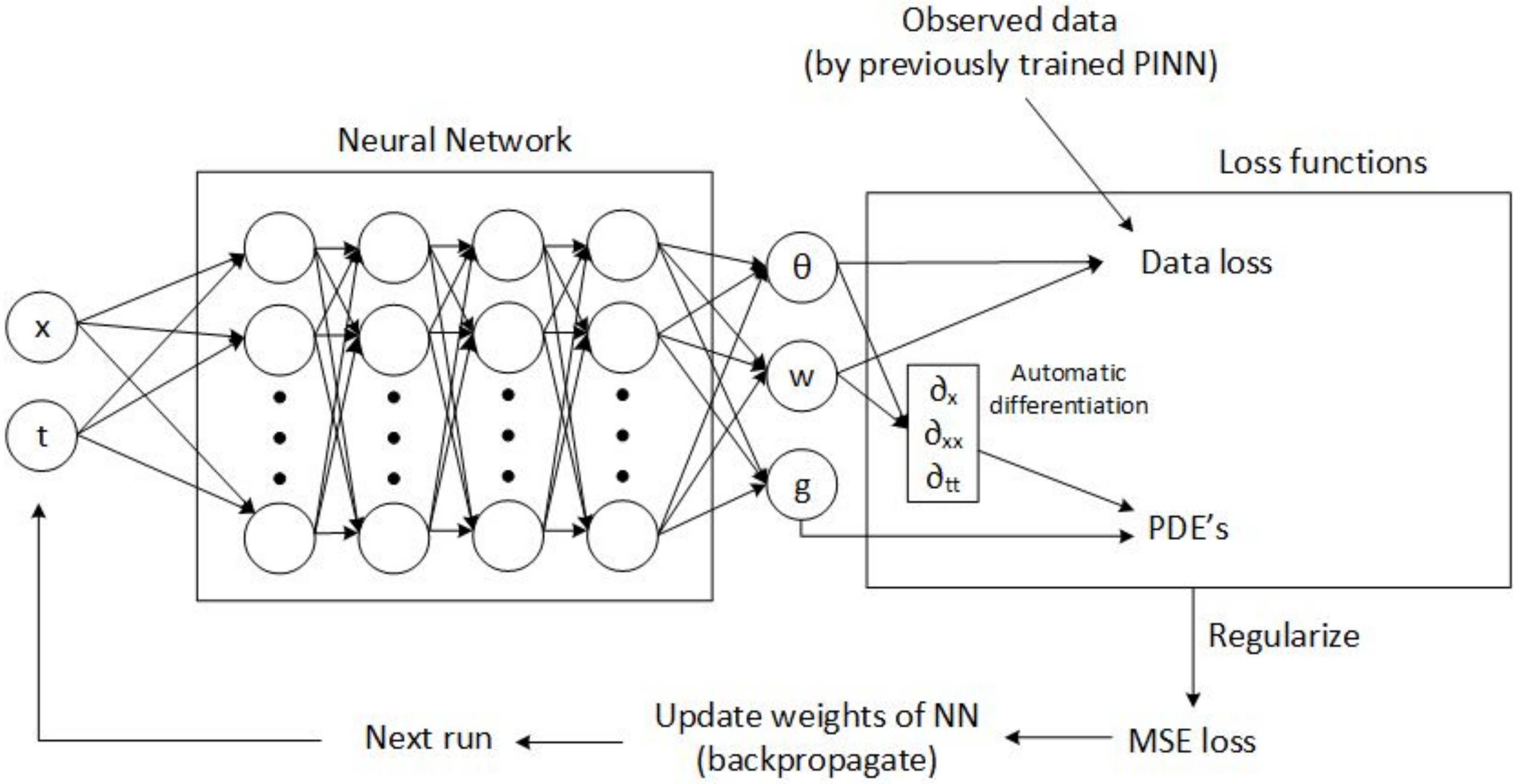


Results



Method (inverse problem)

- Two coupled Partial Differential Equations (PDE's)
- Predict dynamic forcing term alongside theta and w
- Loss is a function of observations and physics -> $Loss = \lambda_1 * Loss_{Data} + \lambda_2 * Loss_{Physics}$



Hyperparameters

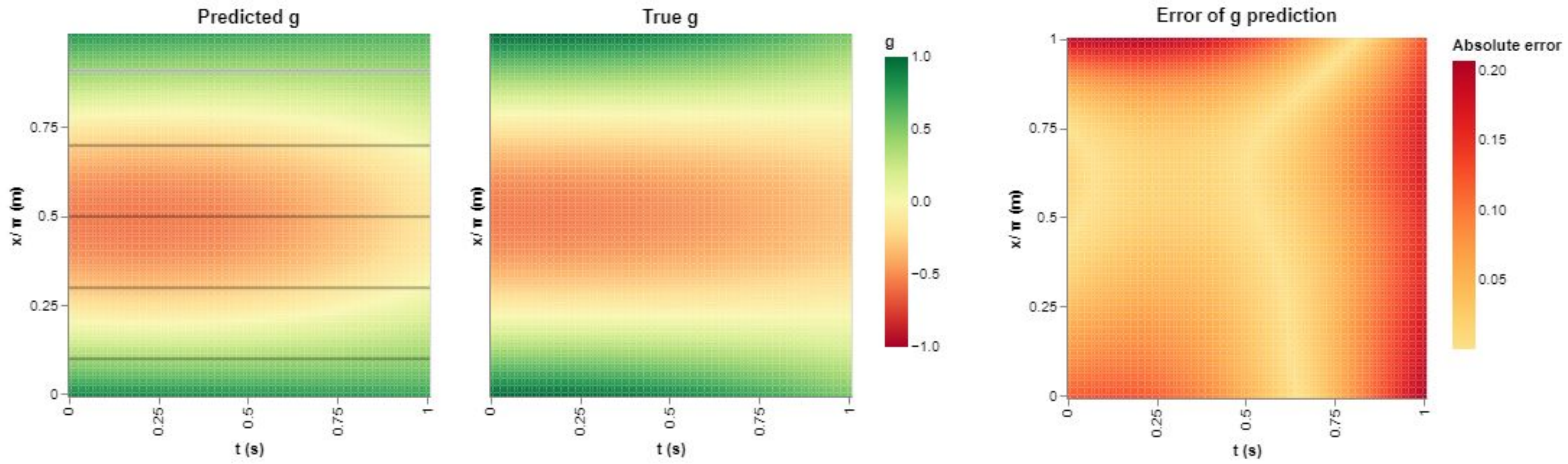
Obtained from empirical testing and literature

- MLP (4 layers, 20 features)
- Activation function: Hyperbolic tangent
- Optimiser: LBFGS
- Epochs 15000
- Learning rate 0.1
- $\lambda_1 = 1, \quad \lambda_2 = 0.05$

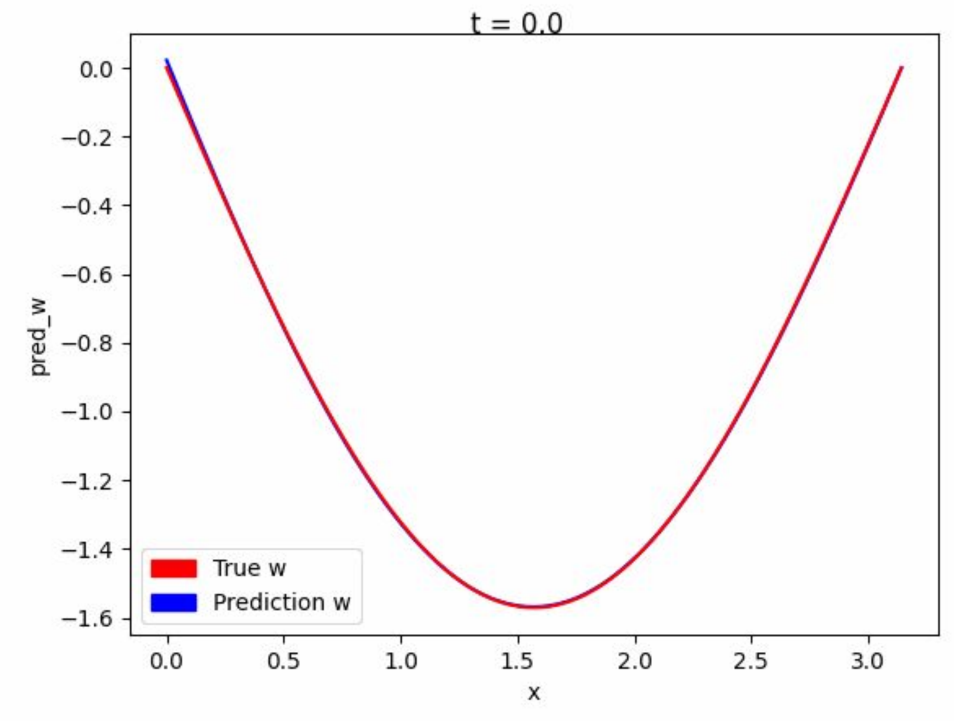
Assumptions for optimisation

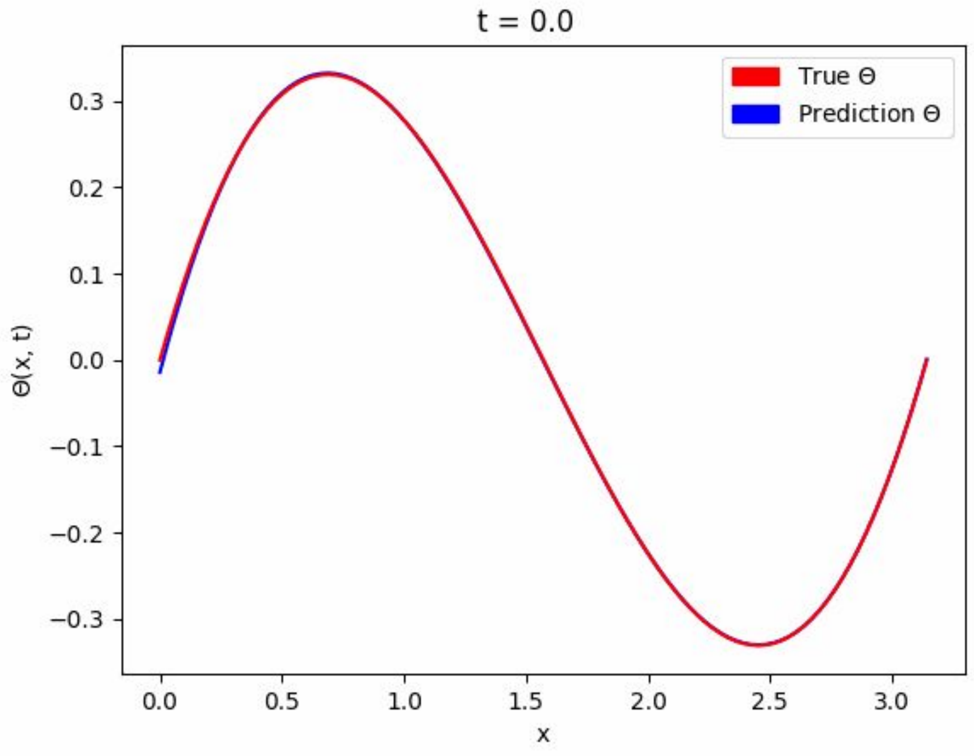
- Assumed sensor sampling rate of 1000 /s
- Sensors fixed at their locations -> to be optimised

Results before sensor optimisation



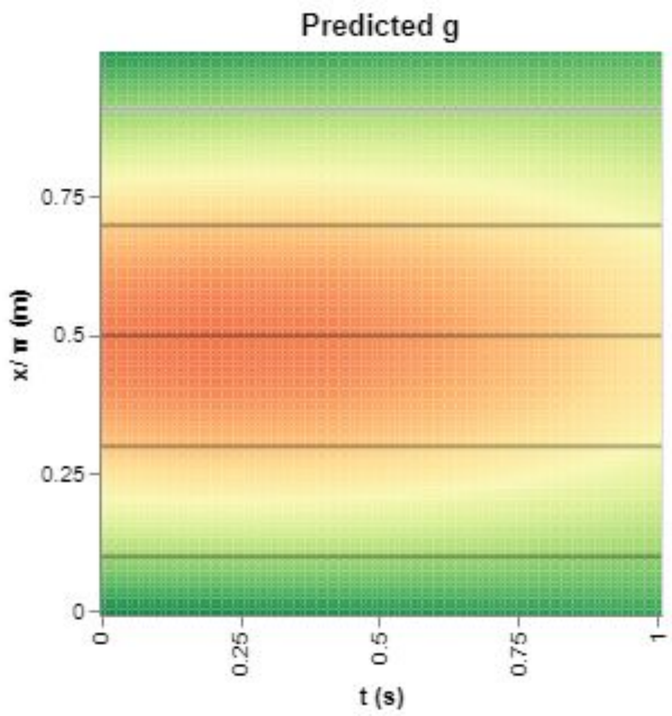
Results before sensor optimisation

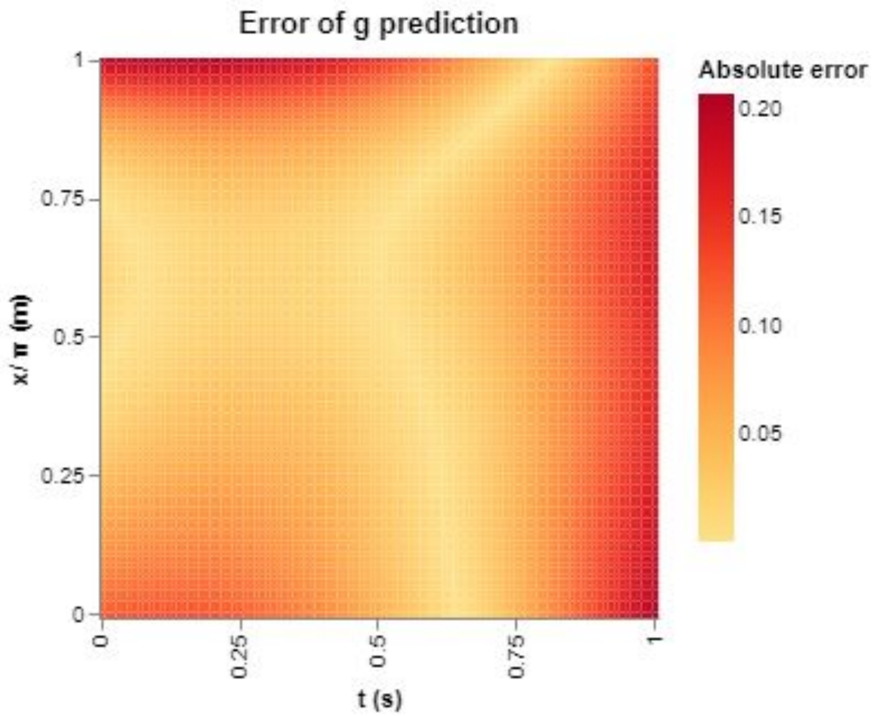




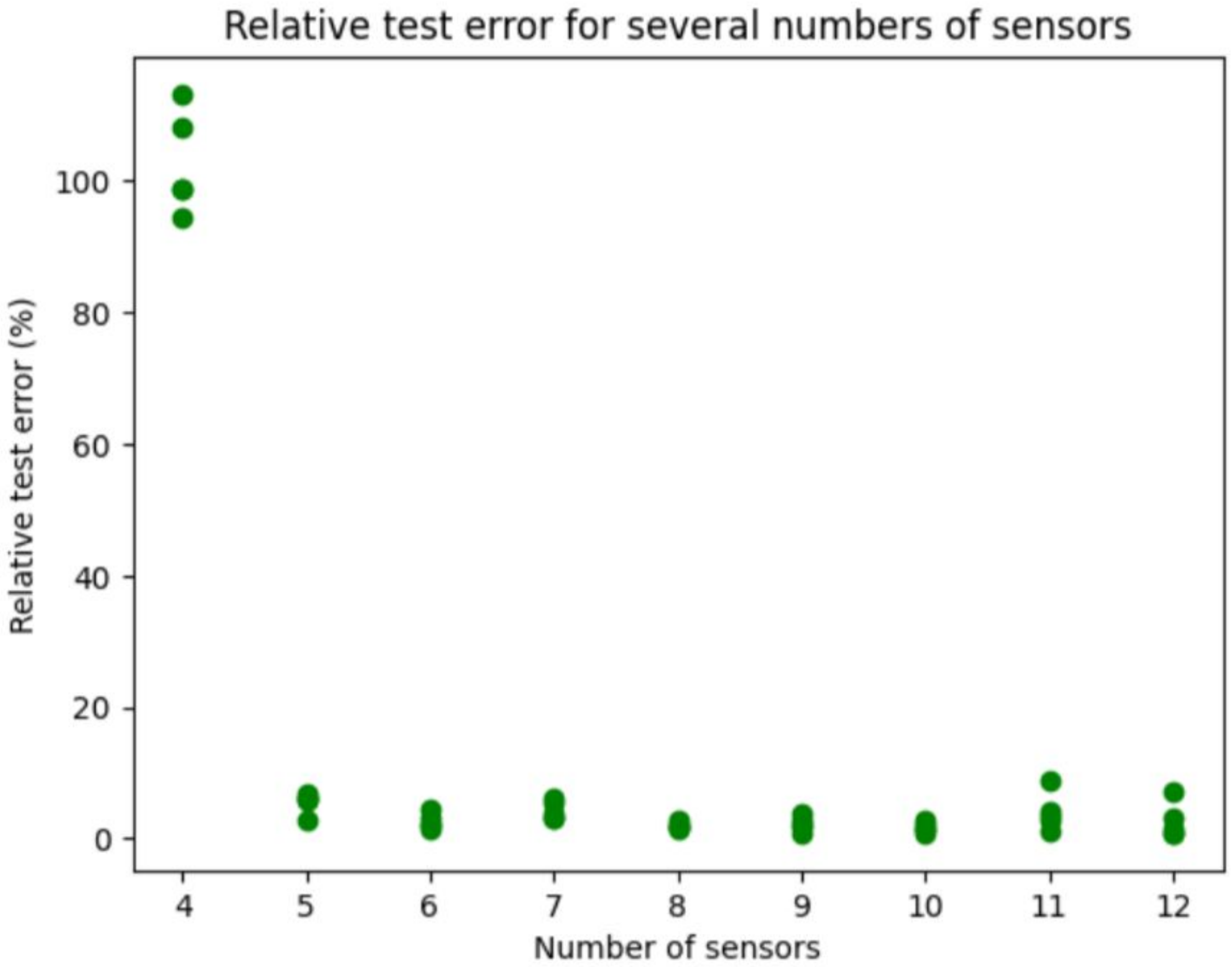
Optimisation sensor placement algorithm

- Run model with linear spaced multiple times with N = 2 to N=15
- Calculate relative error for every model
- Get the model with least amount of sensors for which error < 10%
- Optimise location of this model empirically by looking at error



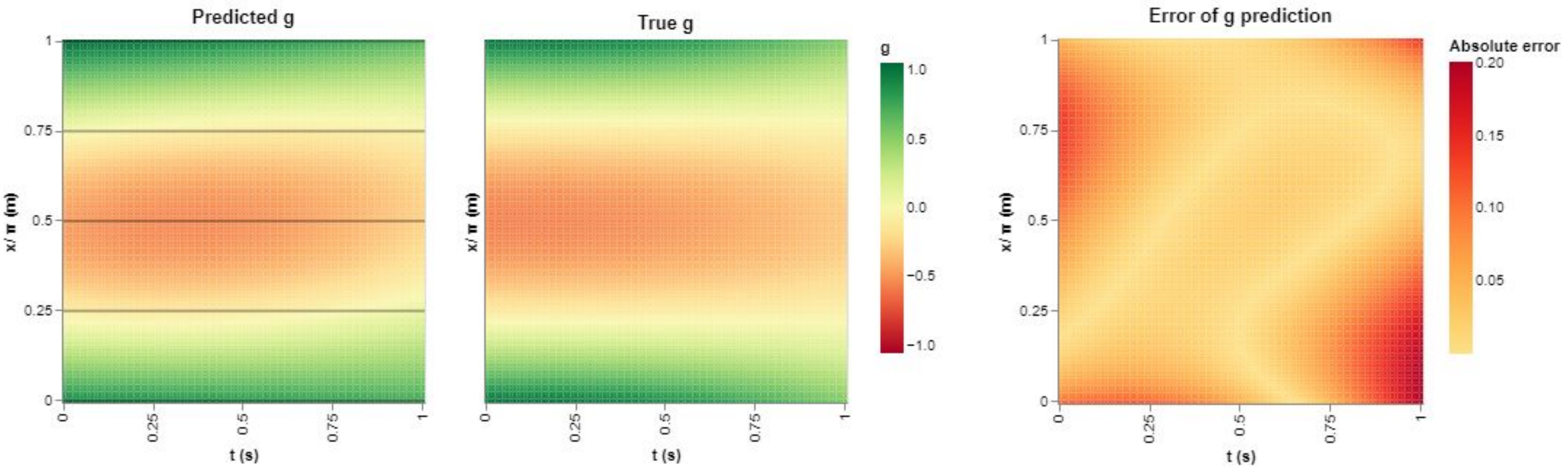


Optimisation amount of sensors



Results optimisation

- Amount of sensors = 5
- Placed on boundaries and linearly spaced in between
- Other placings of sensors are tried but no benefit is found



Relative error of the optimised model = 1.75683181732893 %

Results final estimate of g(x,t)

- Not as accurate as w and θ
- G is less represented in the loss function as θ and w

$$\rho I\theta_{tt} - EI\theta_{xx} - kAG(w_x - \theta) = 0$$
$$\rho A w_{tt} - kAG(w_{xx} - \theta_x) = g(x,t),$$