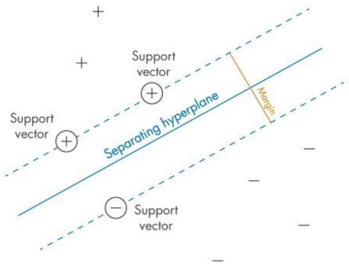
AI models in MATLAB

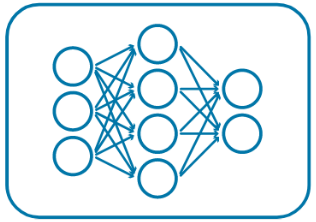
Machine Learning

Deep Learning

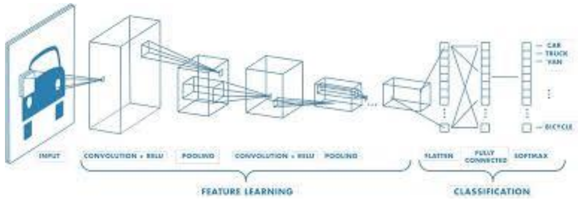


SVM

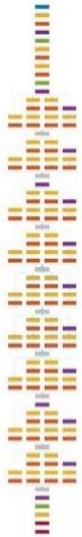
Clustering

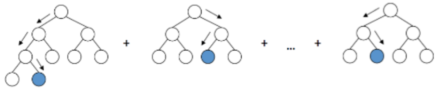


FC

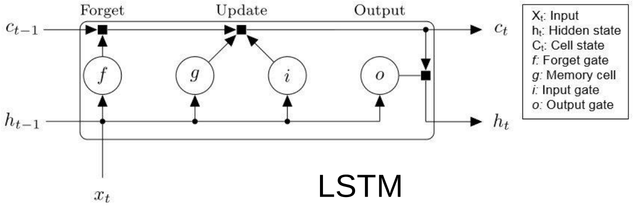


CNN





Decision trees



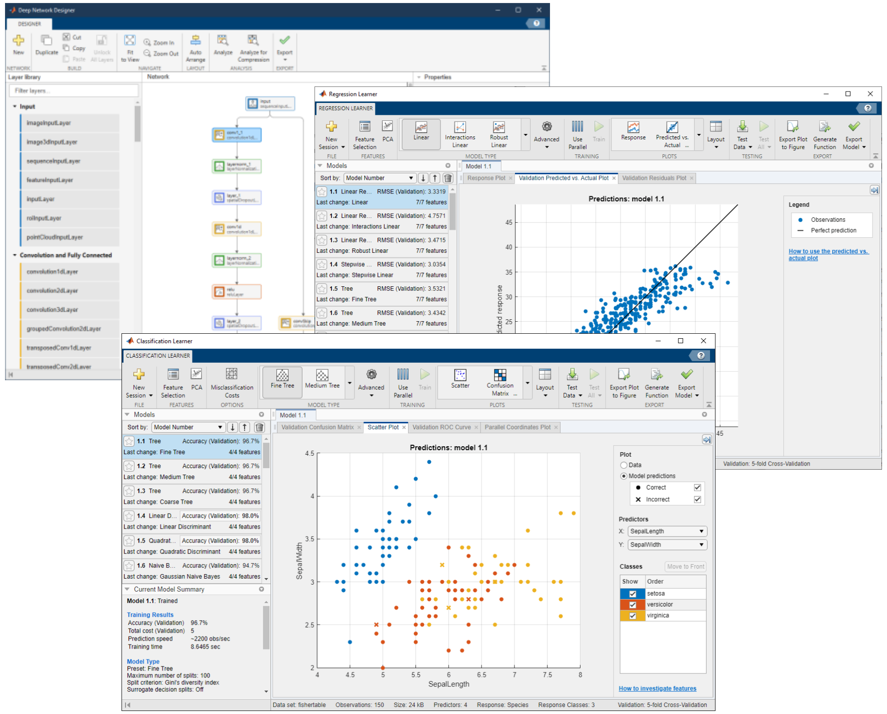
LSTM

3 ways how to create AI model in MATLAB

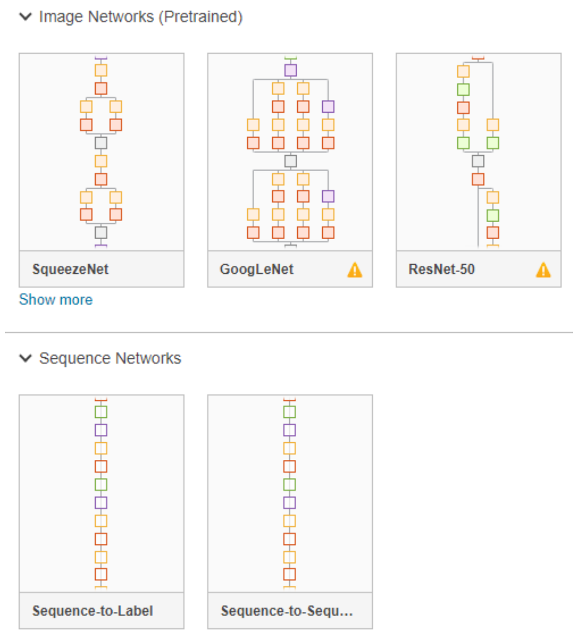
```
inputSize = 12;
numHiddenUnits = 100;
numClasses = 9;

layers = [ ...
    sequenceInputLayer(inputSize)
    bilstmLayer(numHiddenUnits,'OutputMode','last')
    fullyConnectedLayer(numClasses)
    softmaxLayer
    classificationLayer]
```

`fitcauto / fitrauto`

Programmatically
using scripts
and functions



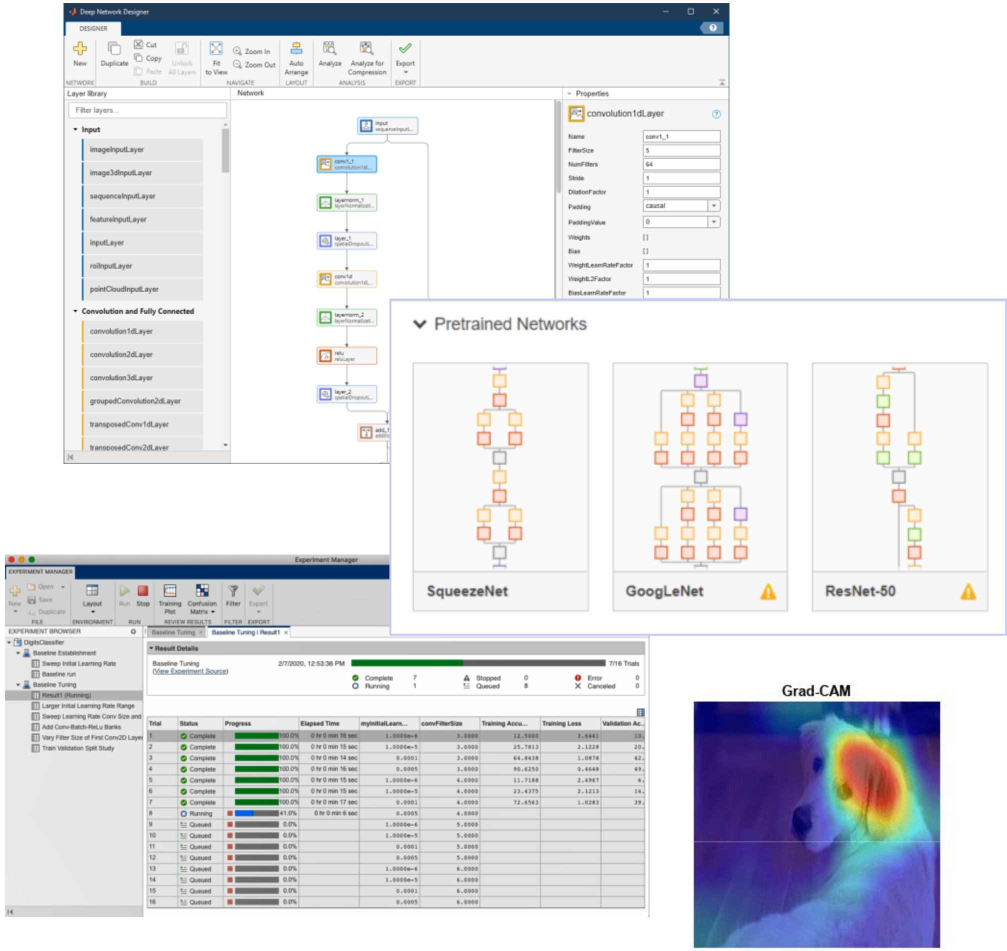
Interactive design
using apps

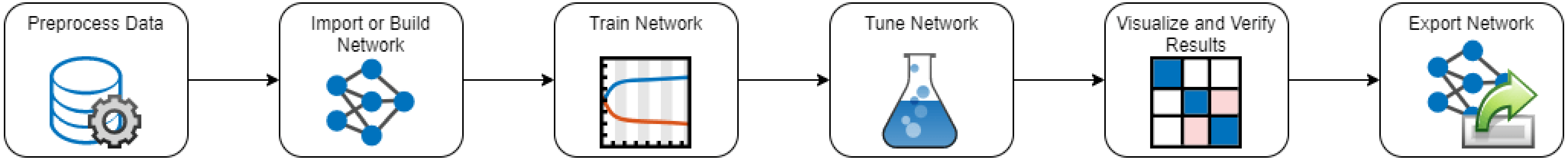


Leverage
pre-defined networks
and pretrained networks

Deep Learning in MATLAB

- Create, train and deploy neural networks
  - variety of applications
  - pre-built networks
- Create networks in the graphical designer
  - design network easier and faster
- Find the optimal network using experiments
- Explain and visualize how networks work
- Interoperability with other environments

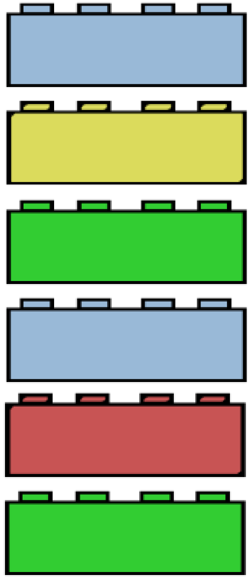




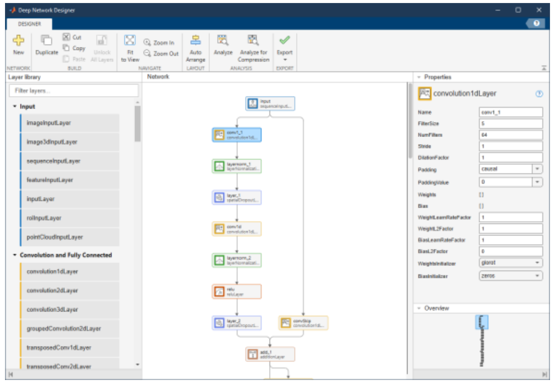
HUMUSOFT

Create Deep Neural Networks in MATLAB

~100 layer types



Deep Network Designer



training
using
APP*

Prepared functions (low code)

```
layers = [imageInputLayer(inputSize)
          convolution2dLayer(filterSize,numFilters)
          reluLayer()
          maxPooling2dLayer(poolSize)
          fullyConnectedLayer(outputSize)
          softmaxLayer()];
```

```
opts = trainingOptions('solver');
net = trainnet(data,layers,lossfcn,opts);
```

```
scores = minibatchpredict(net,newData);
label = scores2label(scores,classNames);
```

for most deep learning tasks

Customizations

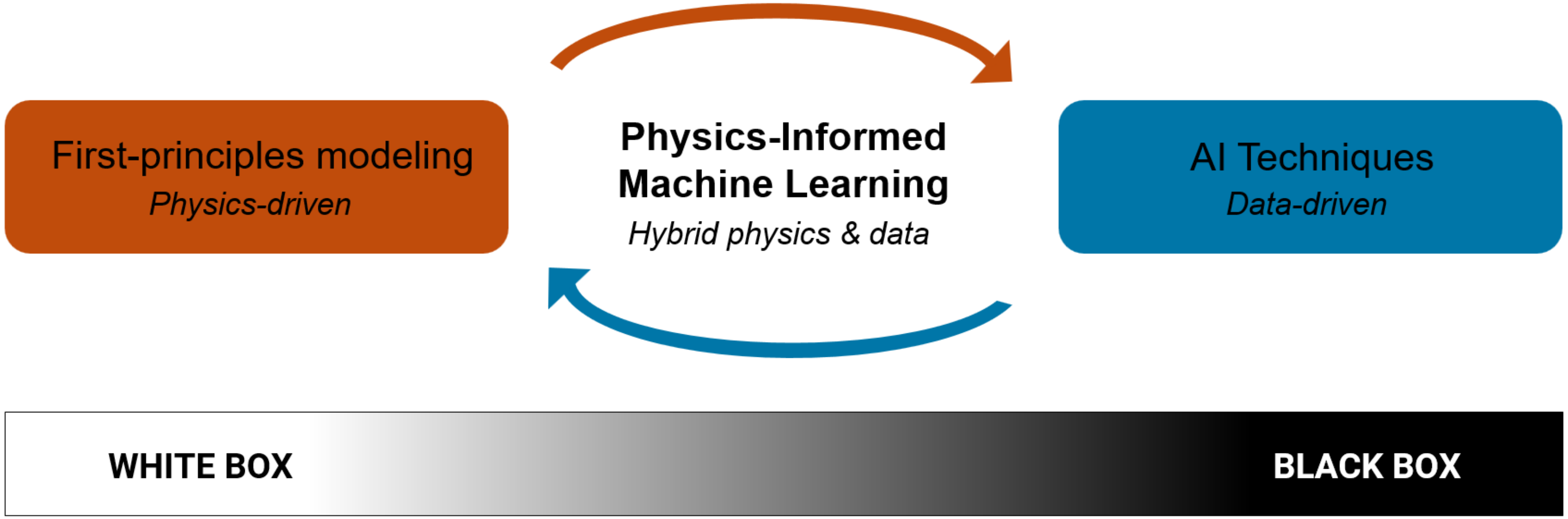
custom training loops
automatic differentiation
custom loss functions
add physical constraints
...

PINN networks, GANs, ...

* image classification tasks

Physics-Informed Machine Learning

- What is it used for?
  - modeling unknown dynamics
  - discovering equations
  - solving known equations



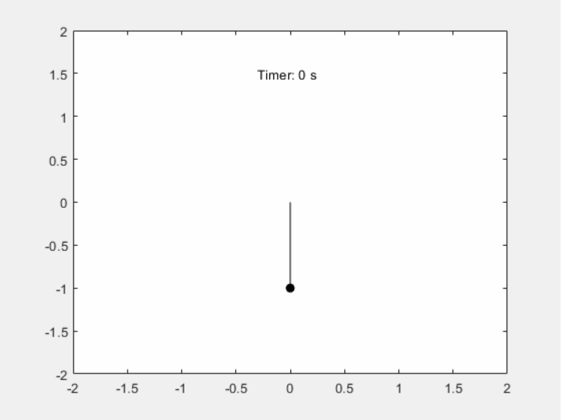
https://blogs.mathworks.com/deep-learning/2025/06/23/what-is-physics-informed-machine-learning/

Physics-Informed Machine Learning

- A key strategy is to impose constraints based on physics principles to AI model
- How is physics knowledge represented?
  - – governing equations
  - – conservation laws and symmetries
  - – boundary and initial conditions
  - – domain-specific knowledge

example: simple pendulum

Type	Given Information	Mathematical Formulation
Governing equations	Pendulum equation	$\ddot{\theta} = -\omega_0^2 \sin\theta$
Conservation Laws	Conservation of energy	$E = \frac{1}{2} m\ell^2 \dot{\theta}^2 + mg\ell(1 - \cos\theta)$
Boundary / Initial Conditions	Initial angular position, velocity	$\theta(0) = \theta_0,\ \dot{\theta}(0) = \dot{\theta}_0$
Domain knowledge	Physical limits (e.g. maximum swing angle)	$\lvert\theta\rvert \leq \theta_{max}$



Physics-Informed Machine Learning

- How is physics knowledge integrated with machine learning?

Stage	Description
1. Defining Objective	Specify what needs to be modeled, including input-output relationships and any known physics.
2. Curating Training Data	Gather training data through experiments, measurements, or simulations. Preprocess raw data into a format suitable for analysis and modeling.
3. Building Model	Choose a machine learning algorithm or a deep learning architecture that best suits your data and task.
4. Defining Loss Function	Create a loss function that quantifies the model's performance in meeting its objectives, such as matching observed data or adhering to physical laws, during training.
5. Optimizing Model	Adjust the model parameters to minimize loss and increase predictive accuracy.
6. Making Predictions	Use the trained model to make predictions or simulate system behavior.

Physics-Informed Machine Learning

- Physics can be incorporated at various stages, but often informs:
  - model’s structure (stage 3)
  - evaluation (stage 4)
- Soft enforcement:
  - add physics-based constraints in the loss function (stage 4)
  - during training, the model is penalized for violating constraints
  - once trained, its predictions may not strictly satisfy them
  - e.g. PINN’s
- Hard enforcement:
  - design the model architecture (stage 3) so that physical constraints are always satisfied
  - e.g. constrained deep learning

Examples of PIML methods

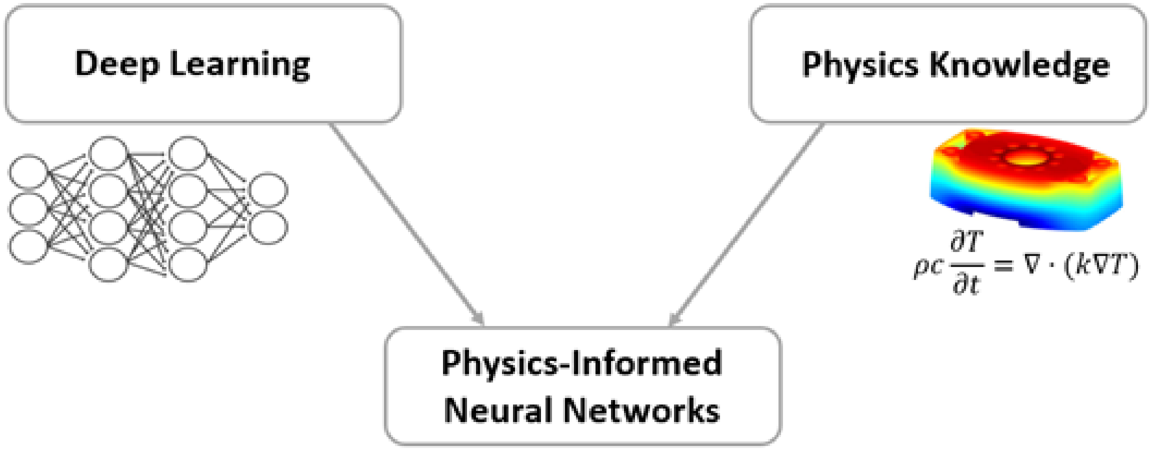
* weak embedding of physical knowledge, e.g. physics-inspired architecture

Approach	Description
Neural Ordinary Differential Equation *	$\dot{x} = f(x, u)$, use a neural network to learn f directly from data
Neural State-Space *	$\dot{x} = f(x, u)$, $y = g(x, u)$, use neural networks to learn f and g directly from data
Universal Differential Equation	combine known physics with machine-learned components
Hamiltonian Neural Networks	account for energy conservation
SINDy (Sparse Identification of Nonlinear Dynamics)	reveal the underlying mathematical relationships from data
Physics-Informed Neural Networks	combine governing equations (ODEs, PDEs) with data to find solutions that match both observed data and physical laws
Fourier Neural Operator *	learn a mapping from the space of input functions directly to the space of solution functions, enabling fast prediction for new scenarios
Graph Neural Networks *	operate directly on mesh or graph-based representations

https://blogs.mathworks.com/deep-learning/2025/07/14/physics-informed-machine-learning-methods-and-implementation/

Physics-Informed Neural Networks (PINNs)

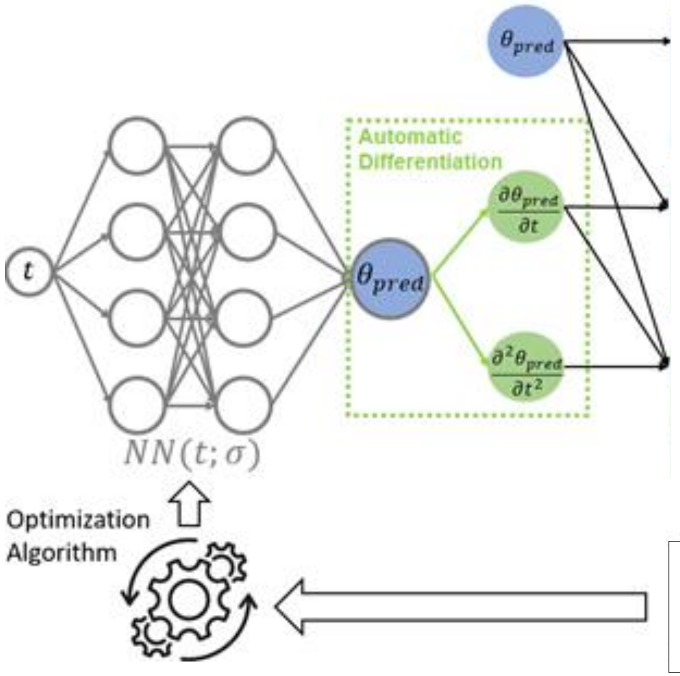
- Neural networks that incorporate physical laws
  - physical laws described by differential equations in their loss functions
- Main purpose
  - guide the learning process toward solutions that are more consistent with the underlying physics
  - use the trained network as the solution of the differential equation



https://www.mathworks.com/discovery/physics-informed-neural-networks.html

Physics-Informed Neural Networks: Loss Function

- Compute loss function L(σ) from three terms
  - $L_{Data}(\sigma)$: known input-output data point from FEM solution
  - $L_{Conds}(\sigma)$: input-output data points from initial and boundary conditions
  - $L_{Physics}(\sigma)$: random input data with physical equation to force the physical constraints



Pre-computed FEM data or measurement data

$$L_{\text{Data}}(\sigma) = \frac{1}{N}\sum_{i=1}^{N} |\theta_{\text{pred}}(\sigma_i, t_i) - \theta_i|^2 \Rightarrow \text{MSE}(\theta_{\text{predicted}} - \theta_{\text{known}})$$

Boundary/Initial Conditions

$$L_{\text{Conds}}(\sigma) = \frac{1}{N}\sum_{i=1}^{N} |\theta_{\text{pred}}(\sigma_i, t_i) - \theta_i|^2 \Rightarrow \text{MSE}(\theta_{\text{predicted}} - \theta_{\text{known}})$$

Differential Equation

$$L_{\text{Physics}}(\sigma) = \frac{1}{N}\sum_{i=1}^{N} |f(\theta_{\text{pred}}(\sigma_i, t_i))|^2 \Rightarrow \text{PDE}(\theta_{\text{predicted}}) \longrightarrow 0$$

Loss function

$$L(\sigma) = L_{\text{Data}}(\sigma) + L_{\text{Conds}}(\sigma) + L_{\text{Physics}}(\sigma)$$

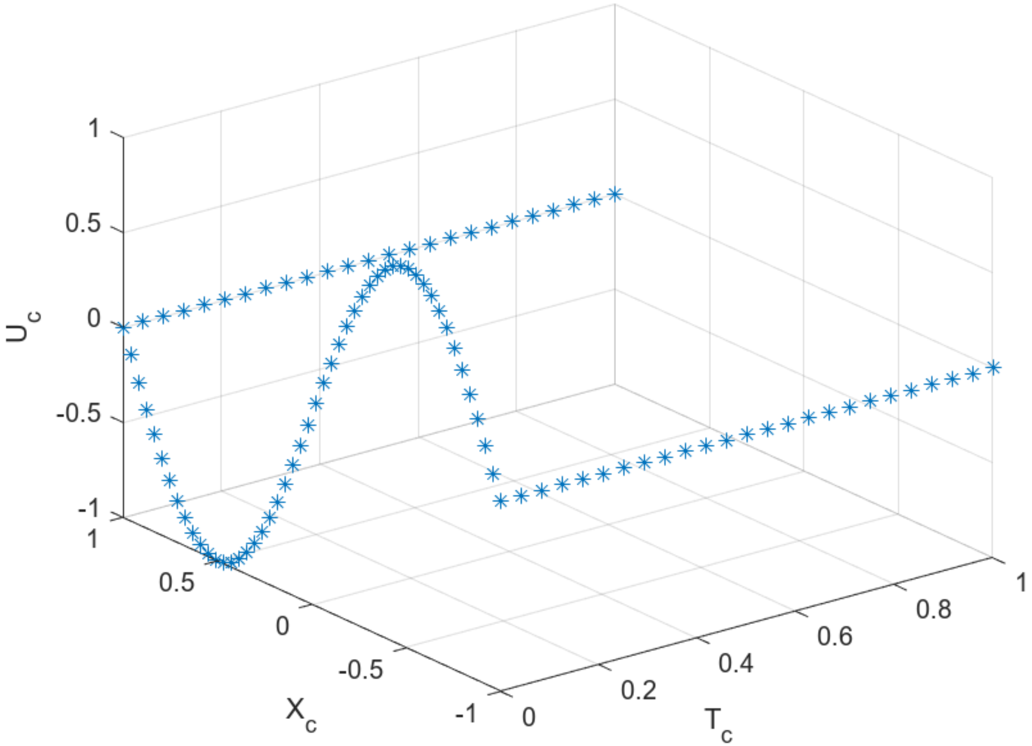
Example: Partial Differential Equation

- Burger's equation: $$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \frac{0.01}{\pi}\frac{\partial^2 u}{\partial x^2} = 0$$

- Initial conditions: $$u(x,0) = -sin(\pi x)$$

- Boundary conditions: $$u(-1,t) = 0$$ $$u(1,t) = 0$$

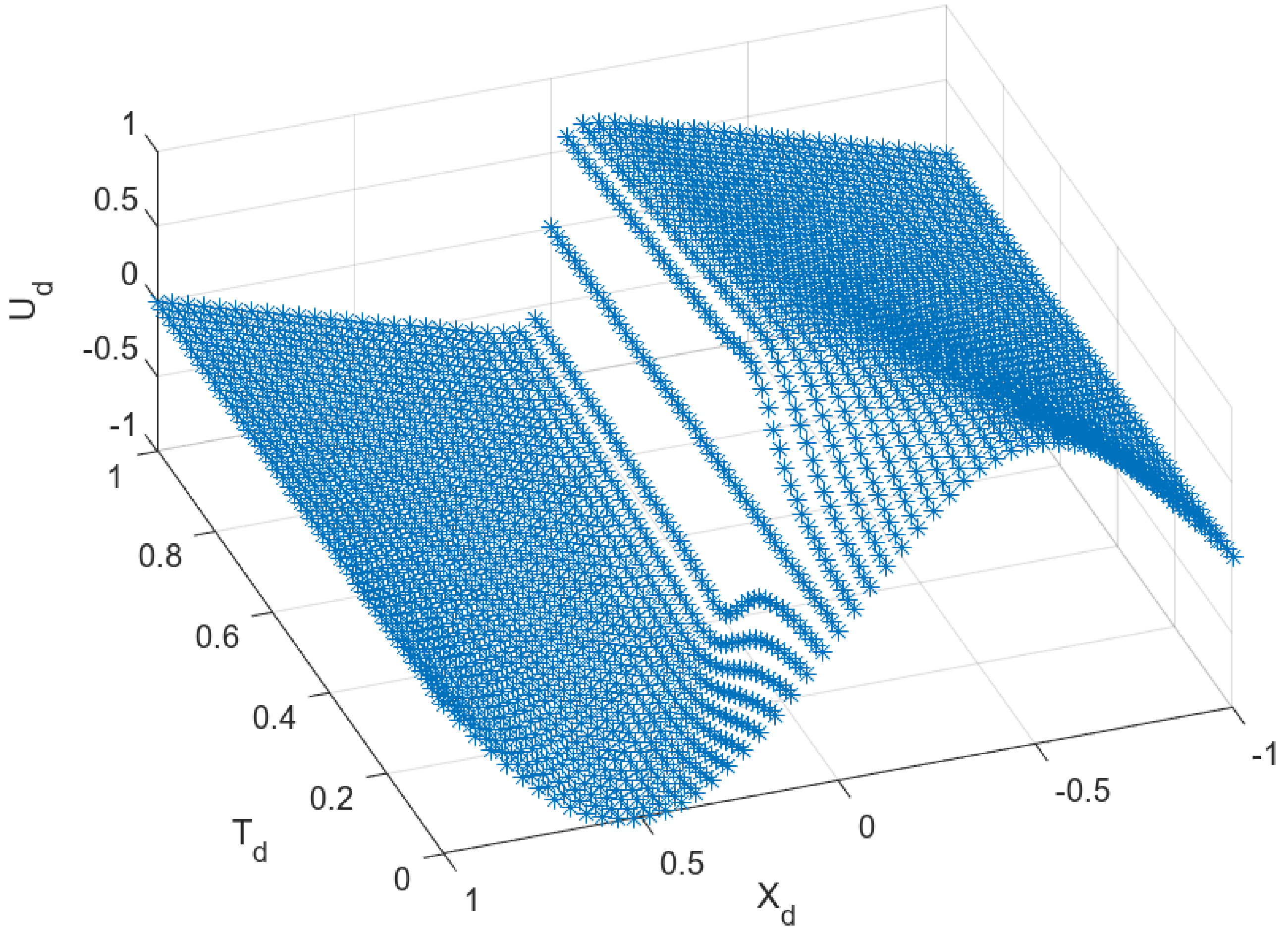
- Solution space: $$(t,x) \in (0,1) \times (-1,1)$$



- Data points from initial and boundary conditions are used for L_{Cond} evaluation

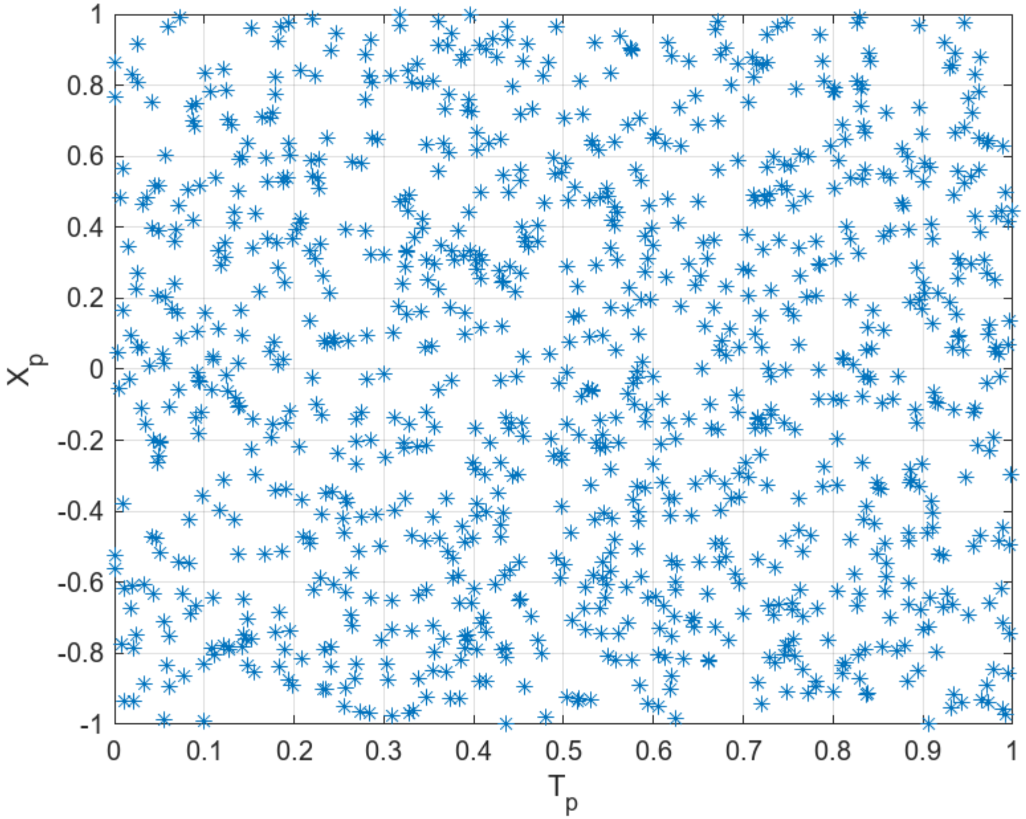
Example: FEM Results

- Data points computed by COMSOL Multiphysics used for L_{Data} evaluation

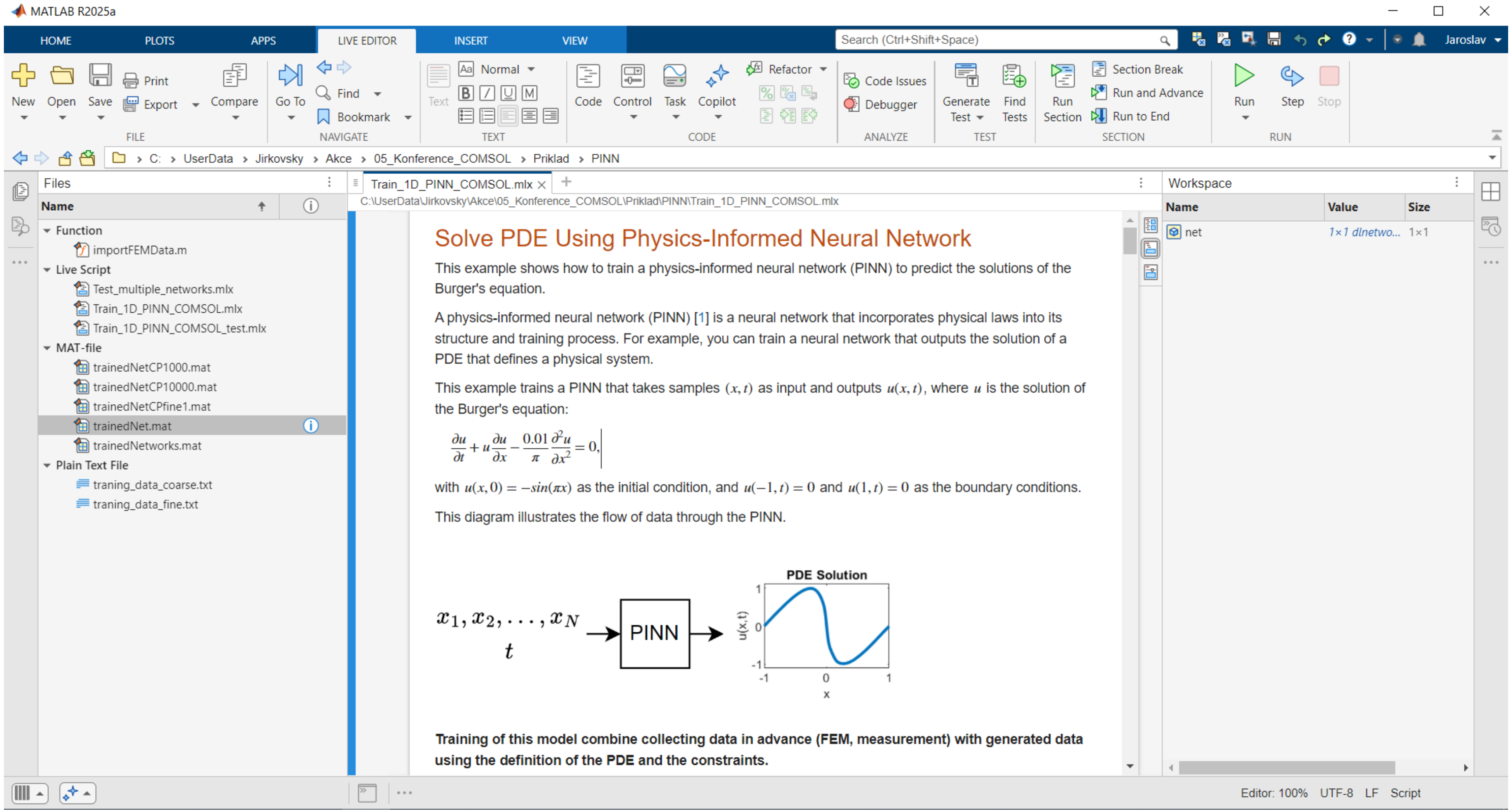


Example: Enforce the Physics

- Random data samples used for $L_{Physiscs}$ evaluation
- Used to enforce the output of the network to fulfill the Burger's equation

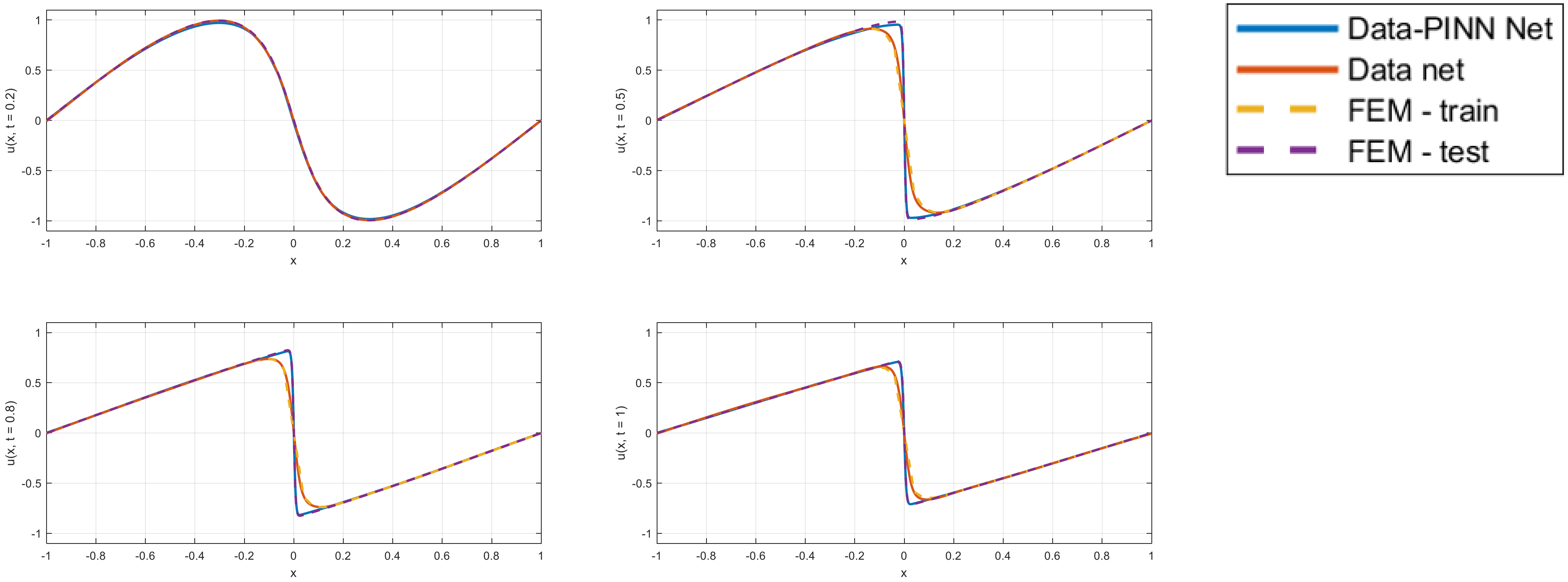


Example: Live Script in MATLAB



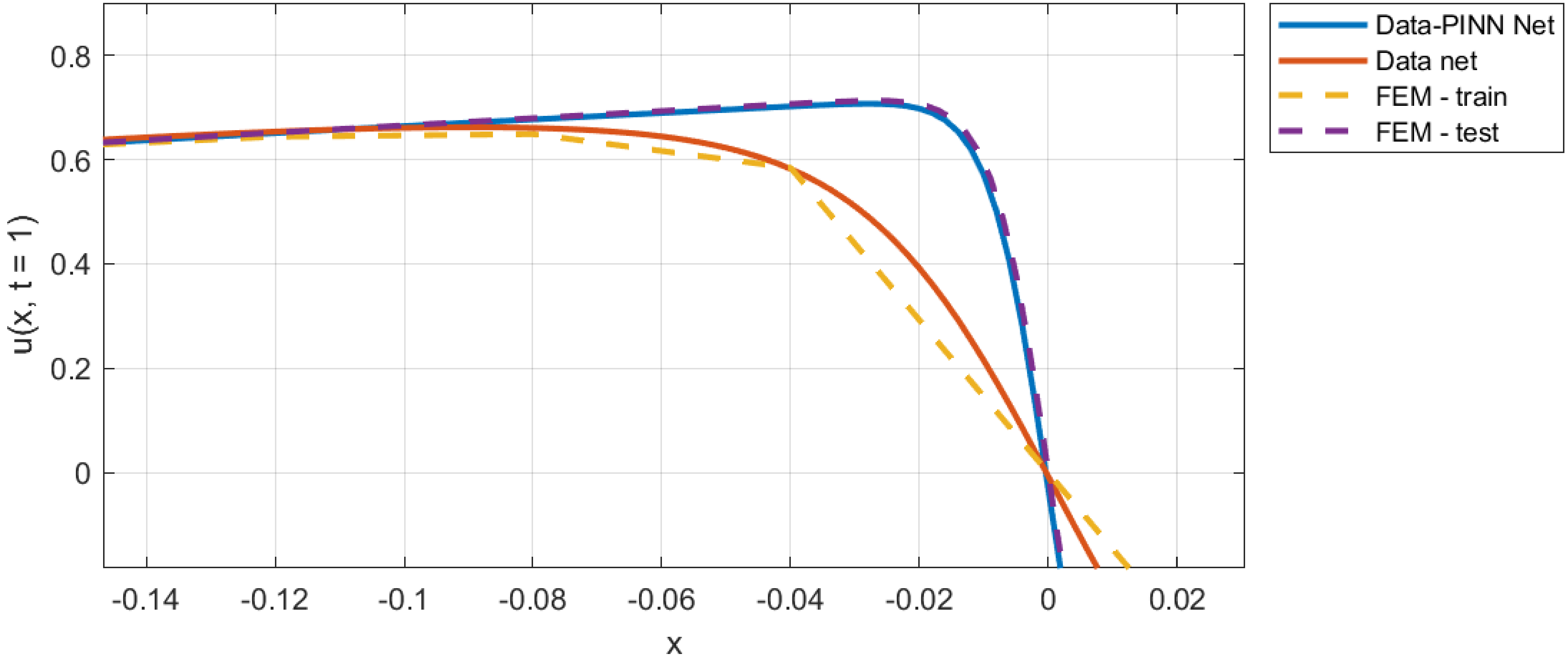
Example: Results

- Solution computed by the trained network at the timestamps 0.2, 0.5, 0.8, 1 sec
- Comparison with the standard (non-PINN) network trained only on the FEM data



Example: Results

- Zoom-in the result at t = 1 sec



Thank you for your attention!