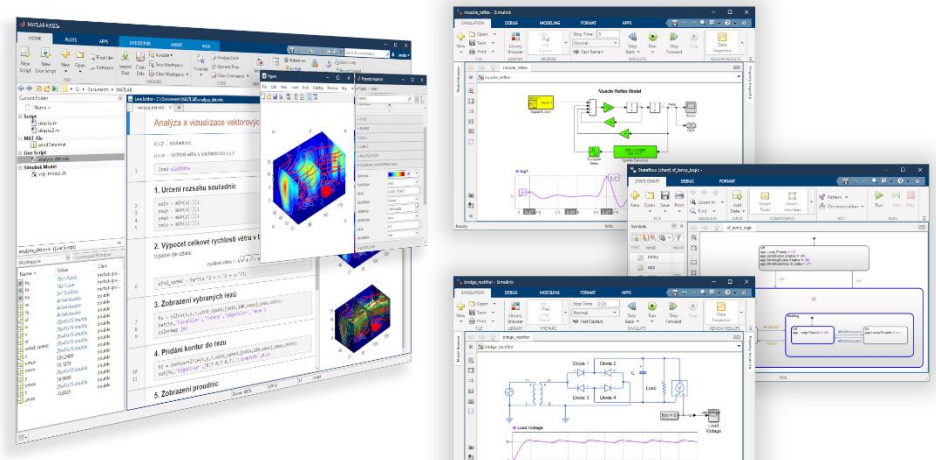
HUMUSOFT

Physics-Informed Neural Networks: COMSOL Multiphysics and MATLAB



MathWorks Authorized Reseller | MATLAB & SIMULINK

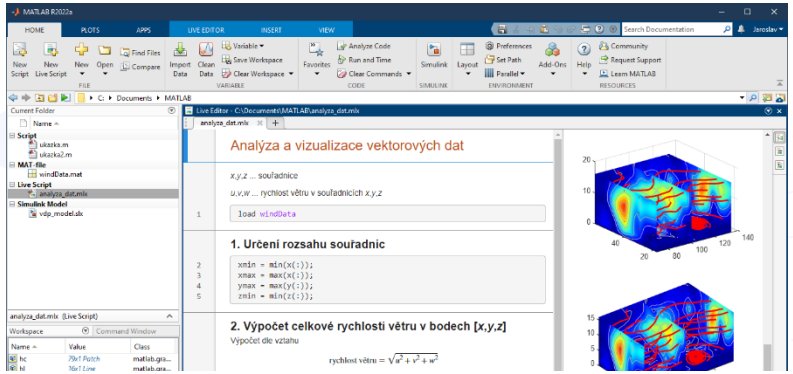
Jaroslav Jirkovský

jirkovsky@humusoft.cz

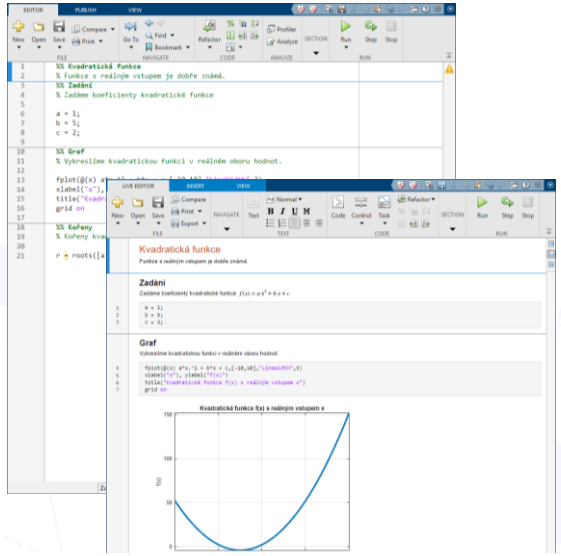
Martin Kožíšek

kozisek@humusoft.cz

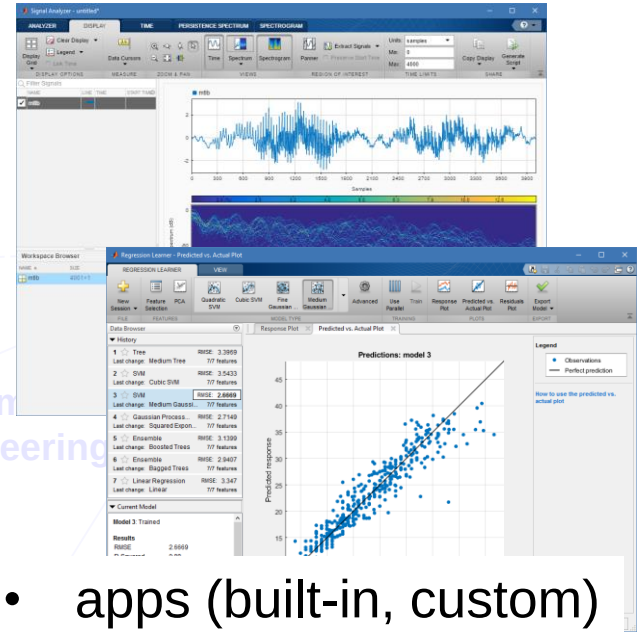
About MATLAB



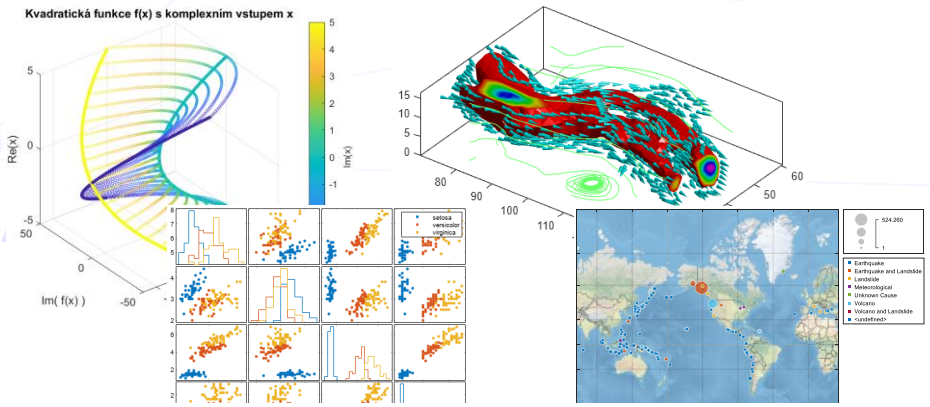
- engineering tool
- interactive environment
- technical and scientific computing



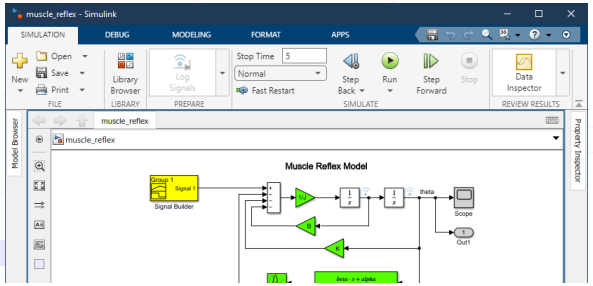
- set of programming tools
- open system



- apps (built-in, custom)



- graphics and visualization

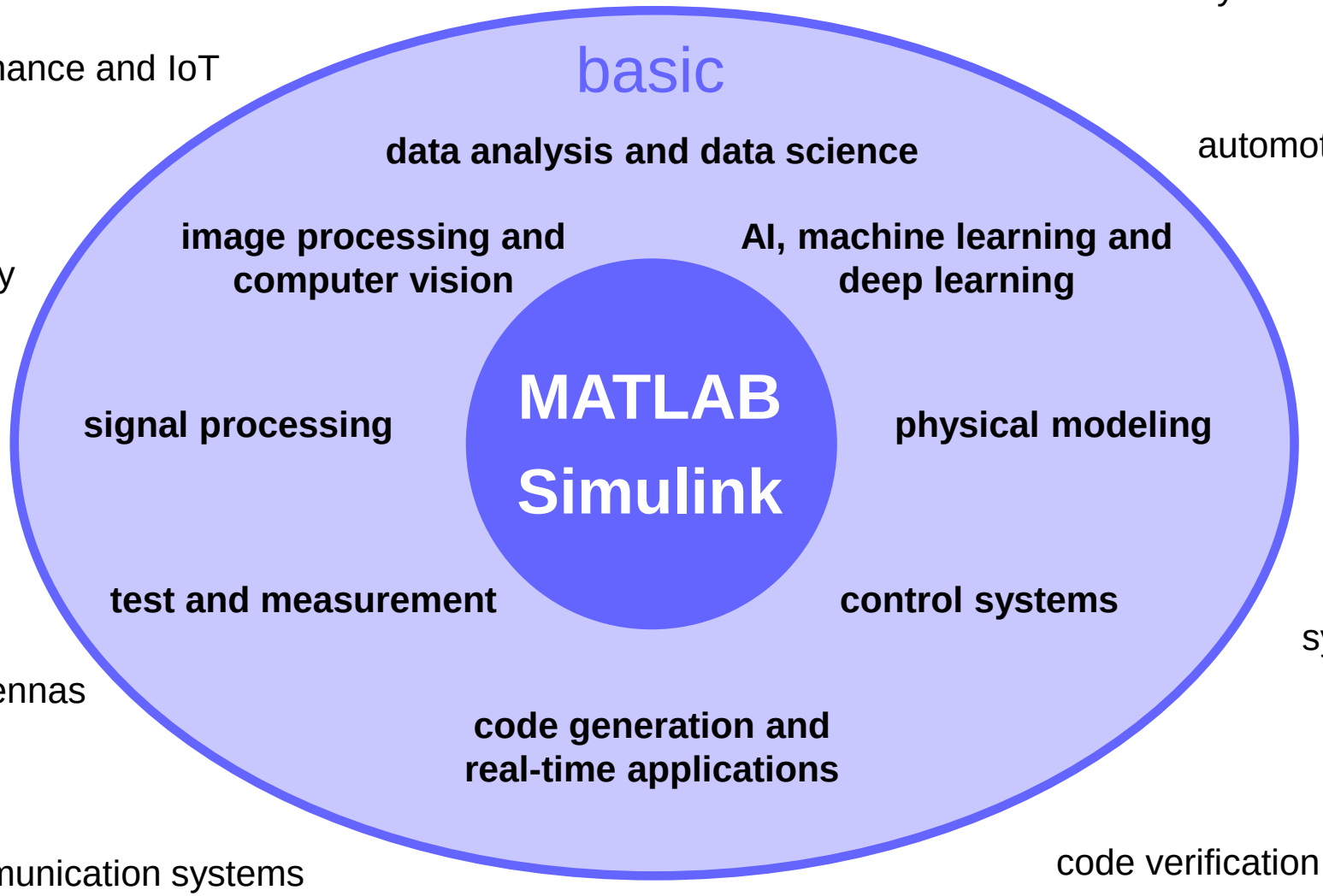


- systems modeling
- simulation and analysis
- Model-Based Design

- 100+ application libraries
- 10 000+ built-in functions
- unified documentation

- connection to external hw/sw
- application development

MATLAB Applications



autonomous systems and robotics

predictive maintenance and IoT

automotive systems and ADAS

computational biology

aerospace and UAV

financial analysis

systems engineering

RF systems and antennas

wireless communication systems

code verification and certification

MATLAB in the Industry

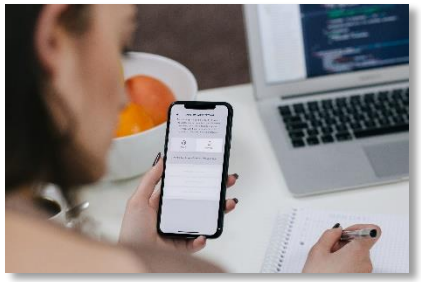
Aerospace and Defense

Automotive

Biological Sciences

Biotech and Pharmaceutical

Communications



Electronics

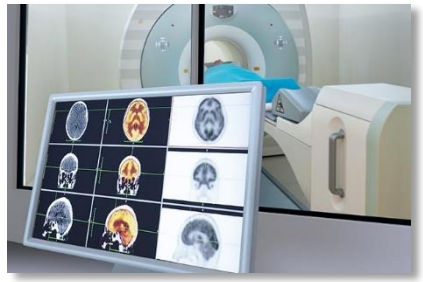
Energy Production

Quantitative Finance

Industrial Automation

Medical Devices

Metals and Mining



Neuroscience

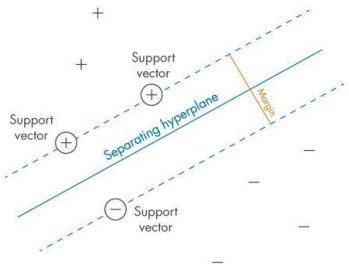
Railway Systems

Applied Physics

Software and Internet

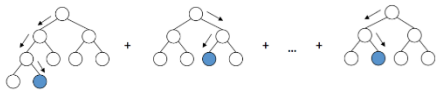
AI models in MATLAB

Machine Learning



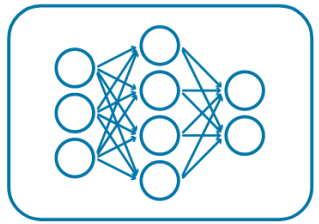
SVM

Clustering

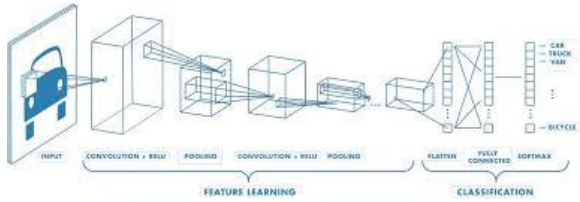


Decision trees

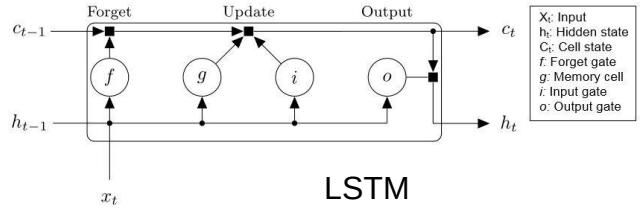
Deep Learning



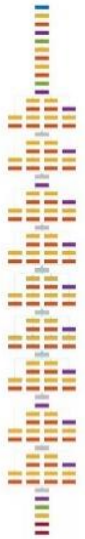
FC



CNN



LSTM

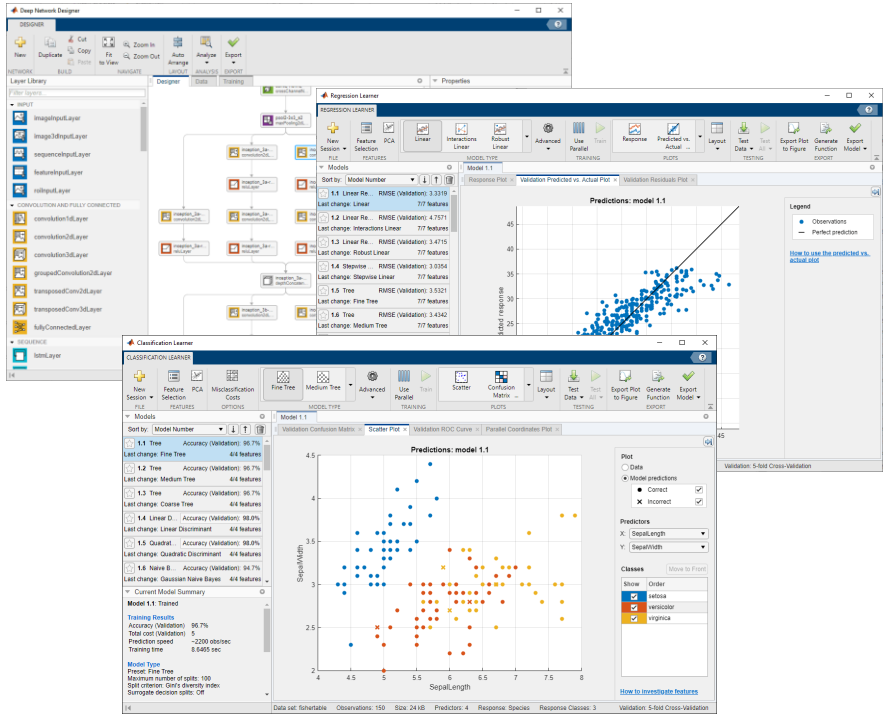


3 ways how to create AI model in MATLAB

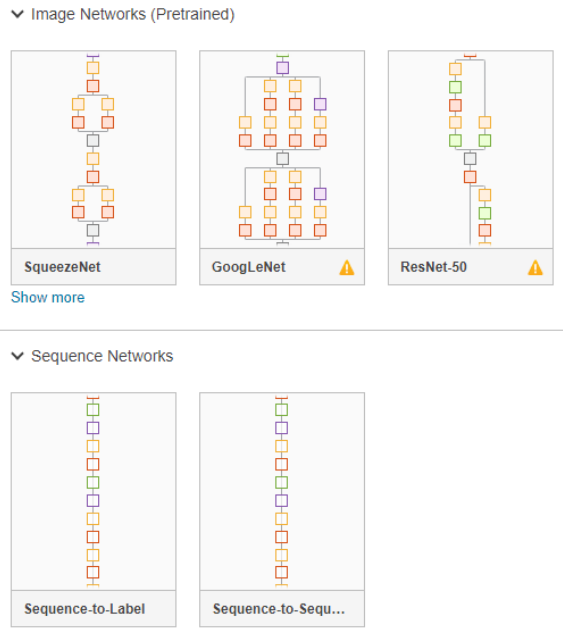
```
inputSize = 12;
numHiddenUnits = 100;
numClasses = 9;

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

`fitcauto / fitrauto`

Programmatically using scripts and functions



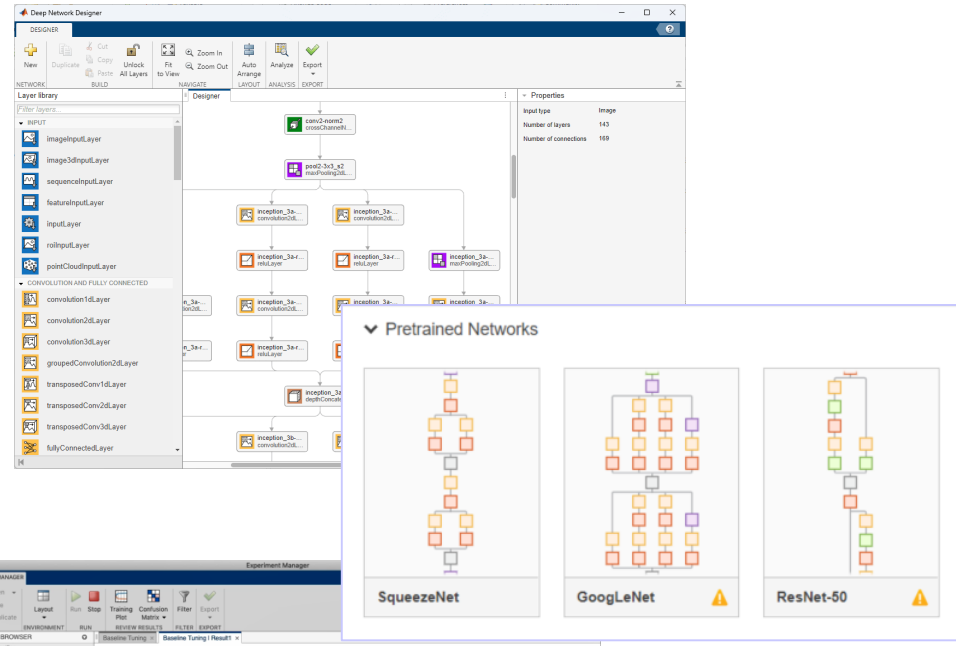
Interactive design using apps

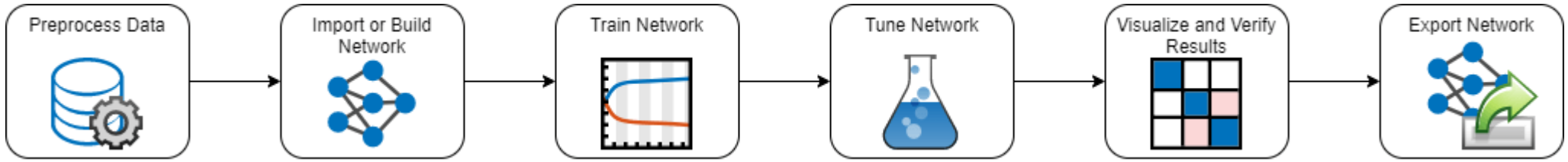


Leverage pre-defined networks and pretrained networks

Deep Learning in MATLAB

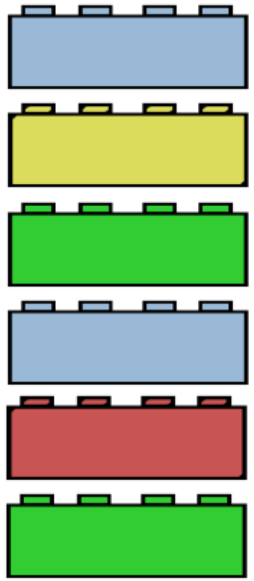
- Create, train and deploy neural networks
  - variety of applications
  - pre-built networks
- Create networks in the graphical designer
  - design network easier and faster
- Find the optimal network using experiments
- Explain and visualize how networks work
- Interoperability with other environments



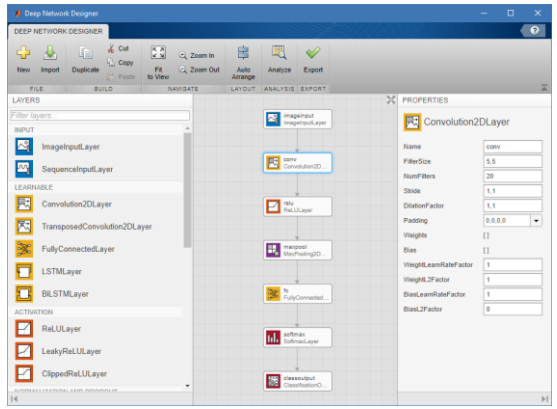


Create Deep Neural Networks in MATLAB

~100 layer types



Deep Network Designer



Prepared functions
(low code)

```
layers = [imageInputLayer([28 28 1])
          convolution2dLayer(5,20)
          reluLayer()
          maxPooling2dLayer(2,'Stride',2)
          fullyConnectedLayer(10)
          softmaxLayer()];
```

Customizations

training
using
APP*

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

```
scores = predict(net,newData);
```

for most deep learning tasks

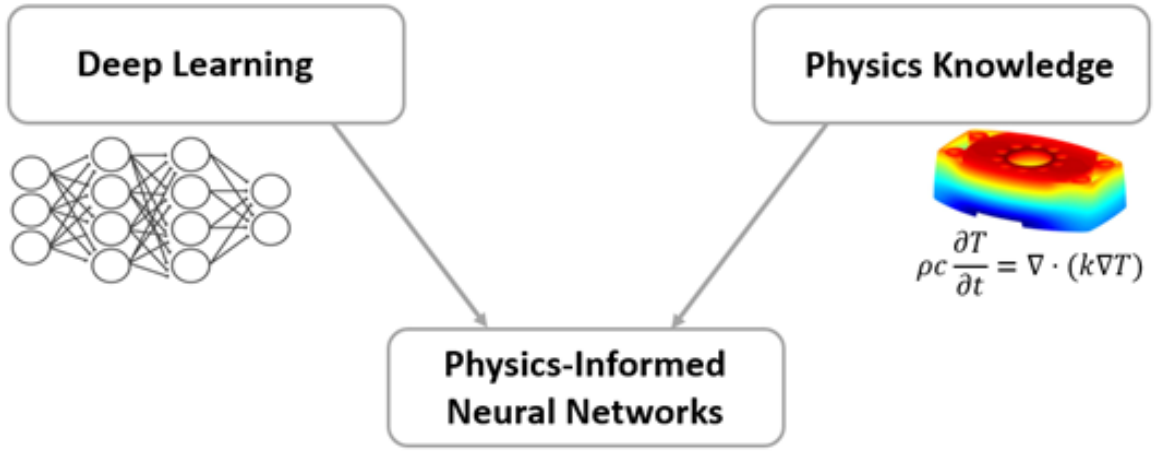
custom training loops
automatic differentiation
custom loss functions
add physical constraints
...

PINN networks, GANs, ...

* image classification tastks

Physics-Informed Neural Networks (PINNs)

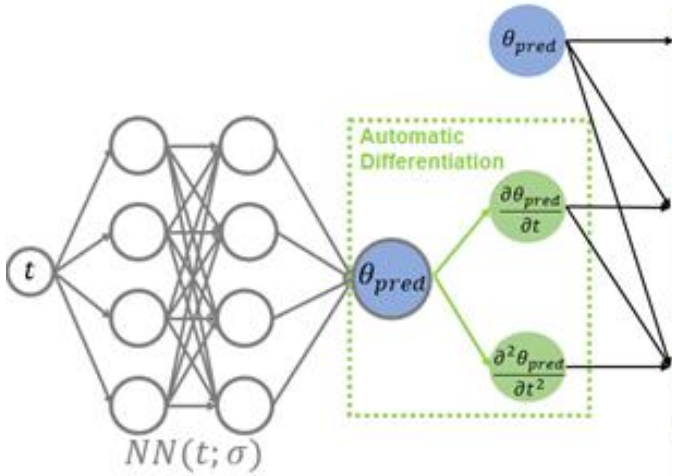
- Neural networks that incorporate physical laws
  - physical laws described by differential equations in their loss functions
- Main purpose
  - guide the learning process toward solutions that are more consistent with the underlying physics
  - use the trained network as the solution of the differential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

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

Boundary/Initial Conditions

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

Differential Equation

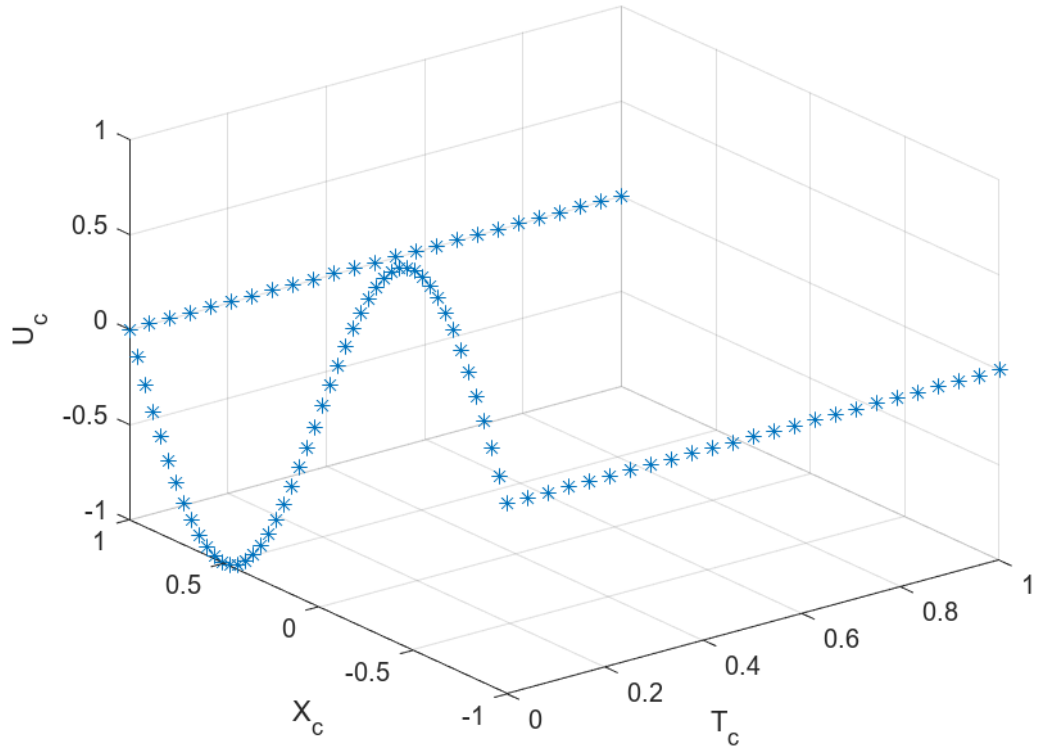
$$L_{\text{Physics}}(\sigma) = \frac{1}{N}\sum_{i=1}^{N} |f(\theta_{\text{pred}}(\sigma_i, t_i))|^2 \quad \Rightarrow \text{PDE}(\theta_{\text{predicted}}) \longrightarrow 0$$

Loss function

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

Example: Partial Differential Equation

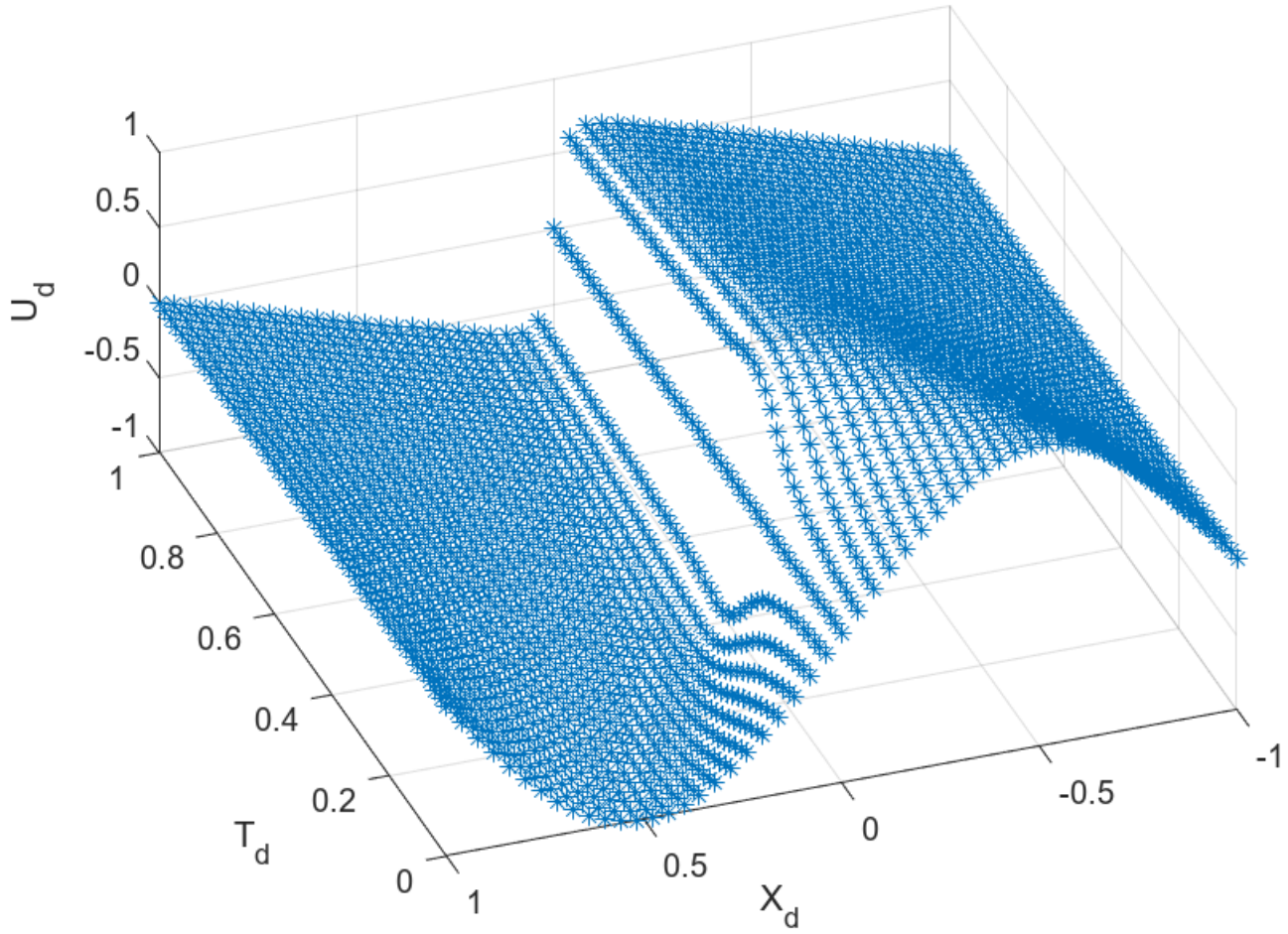
- Burger's equation: $\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \frac{0.01}{\pi}\frac{\partial^2 u}{\partial x^2} = 0$
- Initial conditions: $u(x,0) = -sin(\pi x)$
- Boundary conditions: $u(-1,t) = 0$
  $u(1,t) = 0$
- Solution space: $(t,x) \in (0,1) \times (-1,1)$



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

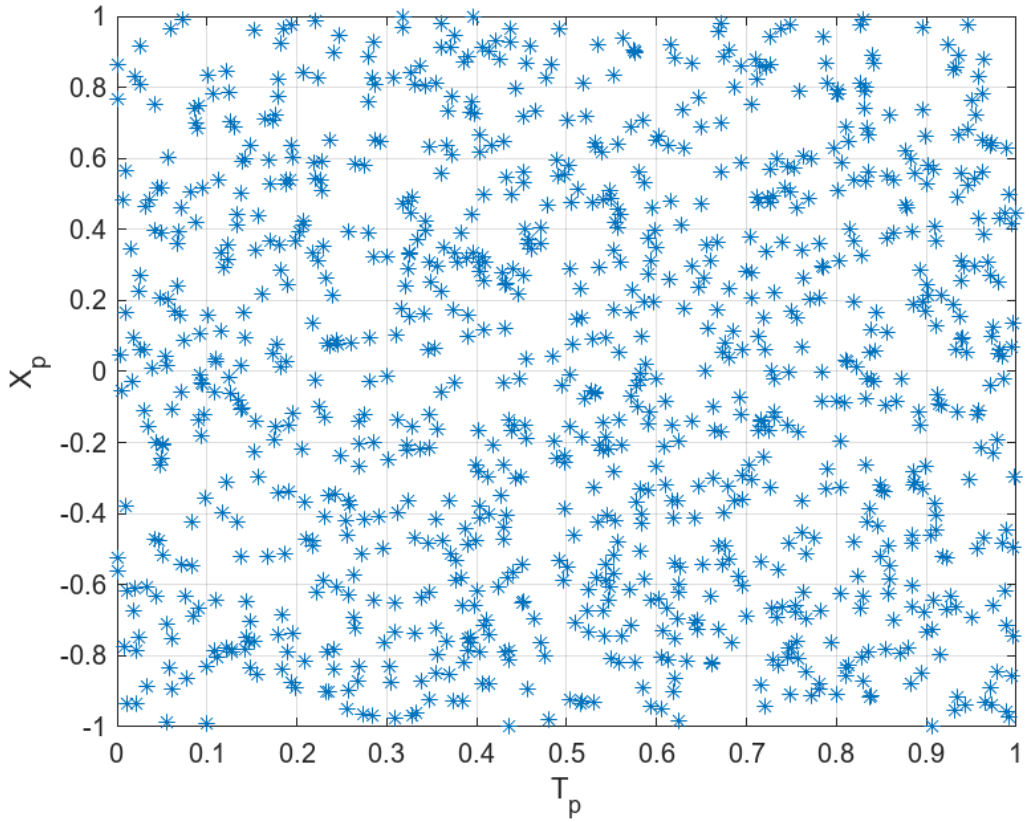
Example: FEM Results

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

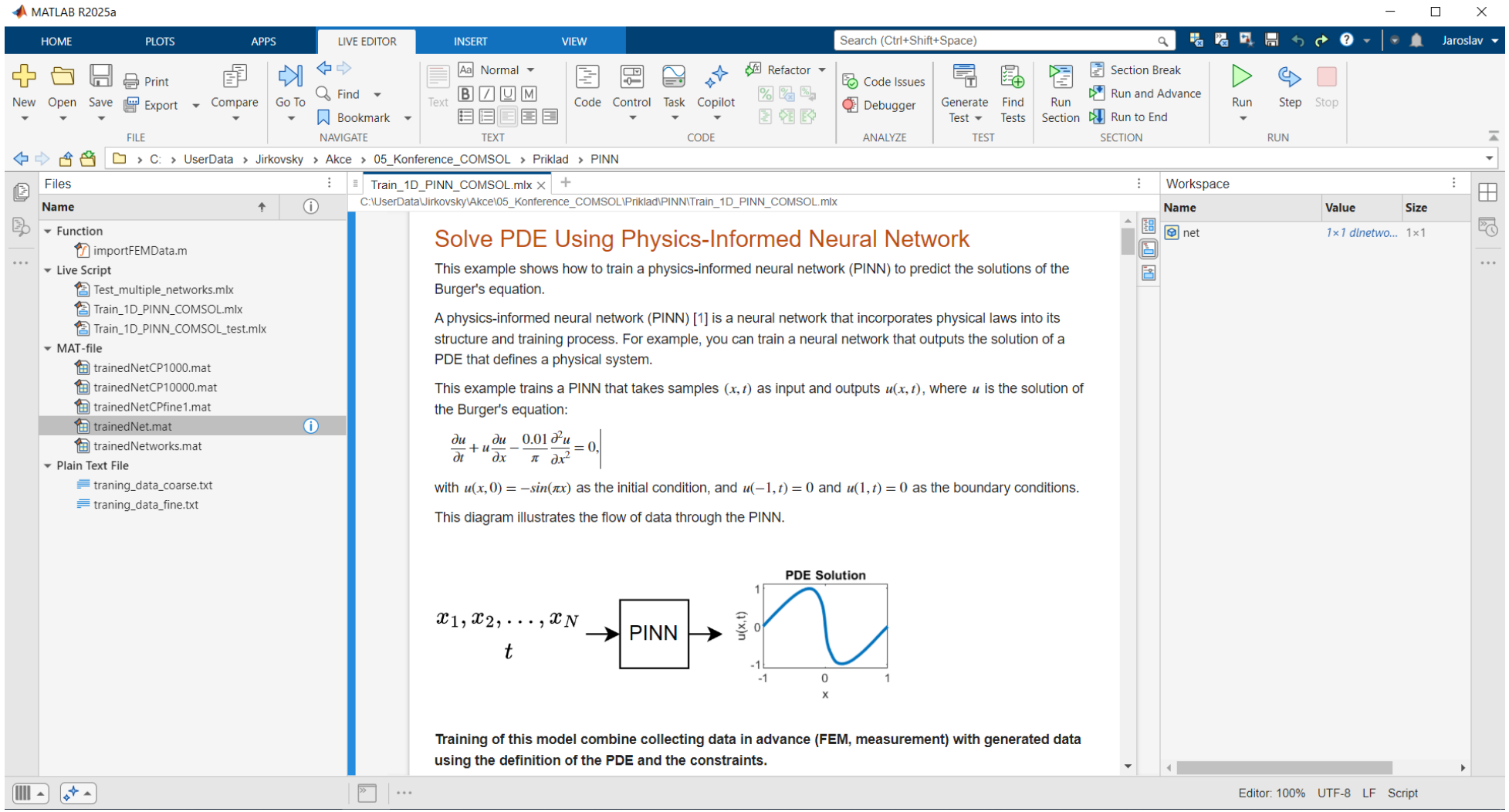


Example: Enforce the Physics

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

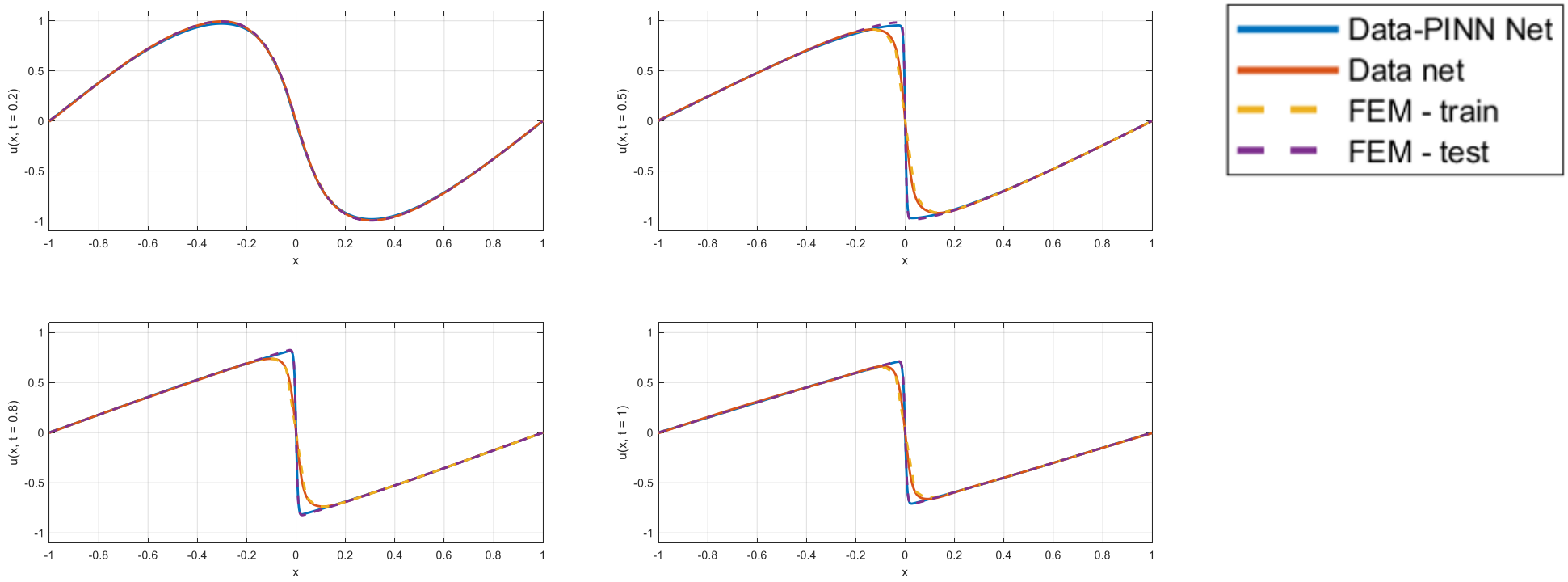


Example: Live Script in MATLAB



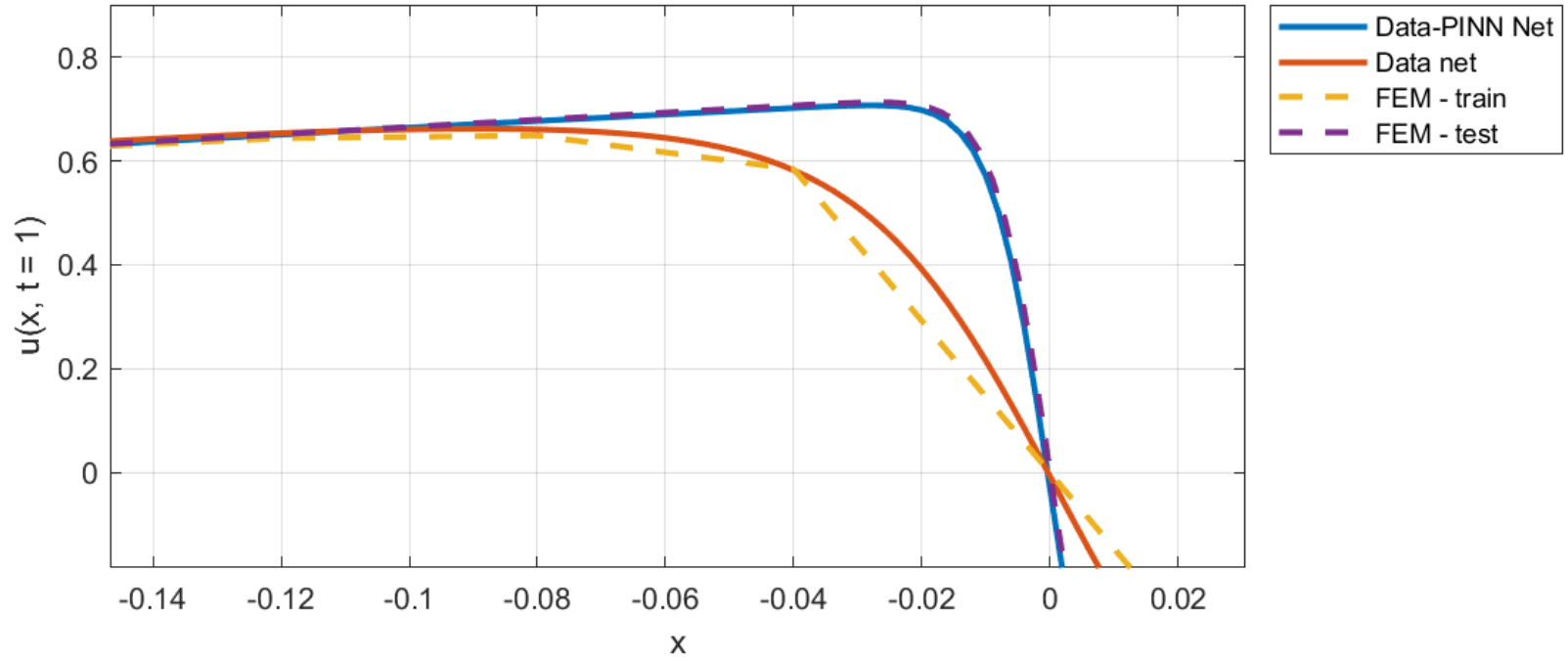
Example: Results

- 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)



- PINN trained on sparse data (FEM – train) provides better results (Data-PINN net) compared to the standard network without physical knowledge (Data net)
- The data (FEM – test) is assumed to represent the correct result

Thank you for your attention!