PRESENTATION 31.01.2022

Implementation of a Modelica library for modeling and simulation with neural networks in OpenModelica

Sören Möller (FH Bielefeld) | Prof. Dr. Bernhard Bachmann (FH Bielefeld) | Dr. Rüdiger Franke (ABB AG)

Structure of the presentation

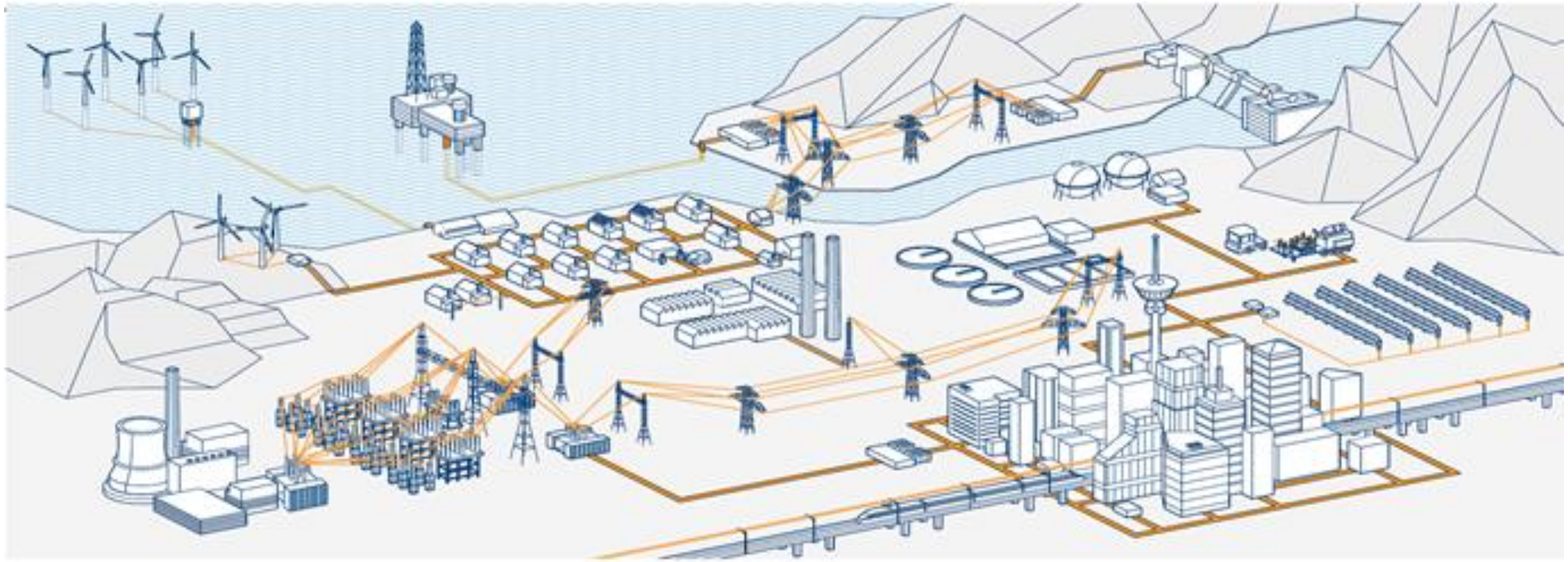
1. Introduction
2. State of the art
3. Neural Network Library
4. Results
5. Conclusions and further work

Introduction

Multi-energy systems

Challenges

- Energy supply facing massive changes and major challenges
- Decentralized energy systems
- Cross-sectoral coupling

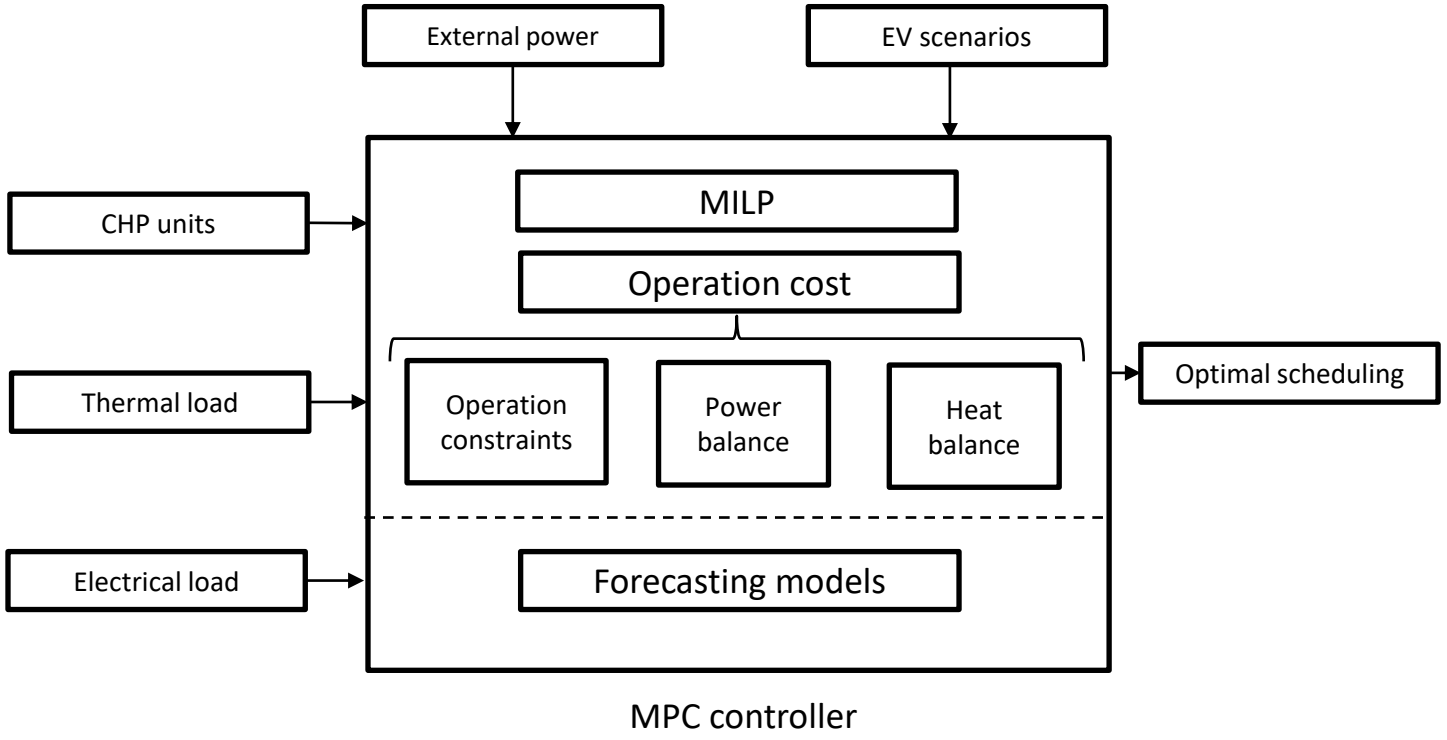


Introduction

Multi -energy systems

MPC for multi-energy system

- Finding optimal setpoints
- Essential task: Transform dynamic behavior into an accurate mathematical model
- White-box-models: precise theoretical analysis
- Black-box-models: no or little prior knowledge
- Use artifical neural networks to model complex systems



Structure of the presentation

1. Introduction
2. State of the art
3. Neural Network Library
4. Results
5. Conclusions and further work

State of the art

Data Transformation

Hypercube Standardisation

- Useful for neural networks
- Normalize inputs to [0,1] or [-1,1]

$$\hat{x}^{(i)} = \frac{x^{(i)} - x_{min}^{(i)} * (1,1,\dots)^T}{x_{max}^{(i)} - x_{min}^{(i)}}$$

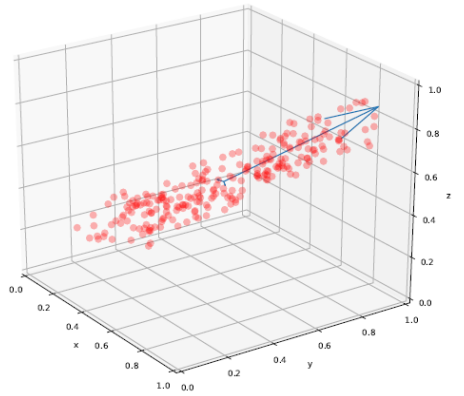
Mu-Sigma Standardisation

- Another approach for data preprocessing
- Arithmetic mean zero and empirical variance one

$$\tilde{x}^{(i)} = \frac{x^{(i)} - \bar{x}^{(i)} * (1,1,\dots)^T}{\sigma_{x^{(i)}}}$$

Principle component analysis (PCA)

- Transforming attributes of a dataset into a new set of uncorrelated attributes
- Reduce dimensionality of a dataset
- A linear combination of features is searched for, so that the variance is maximized



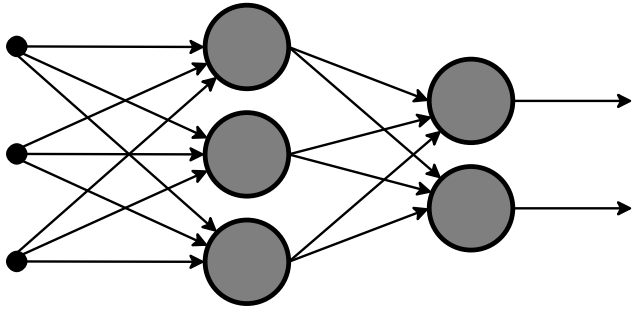
State of the art

Feed-forward neural networks

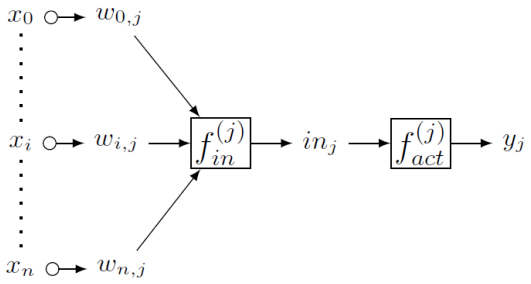
Characteristics

- Most commonly used neural networks
- Good results in many application areas
- Universal approximation theorem
- No internal state

Structure



Neuron



Actication functions

- $sig(a * x) = \frac{1}{1+\exp(-a*x)}$
- $\tanh(a * x) = \frac{\exp(2a*x)-1}{\exp(2a*x)+1}$
- $Identity(a * x) = a * x$
- $ReLU(x) = \begin{cases} x, x \geq 0 \\ 0, x < 0 \end{cases}$

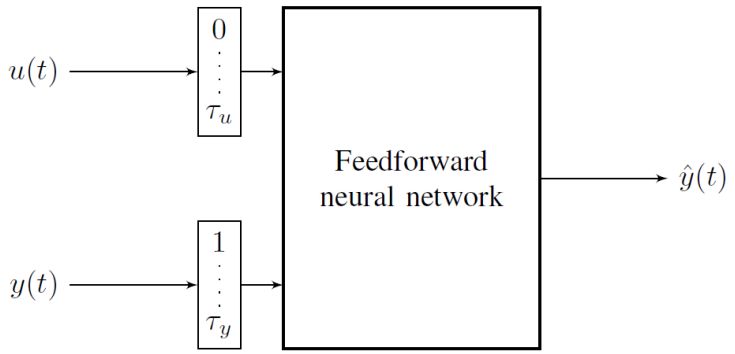
State of the art

Dynamic neural networks

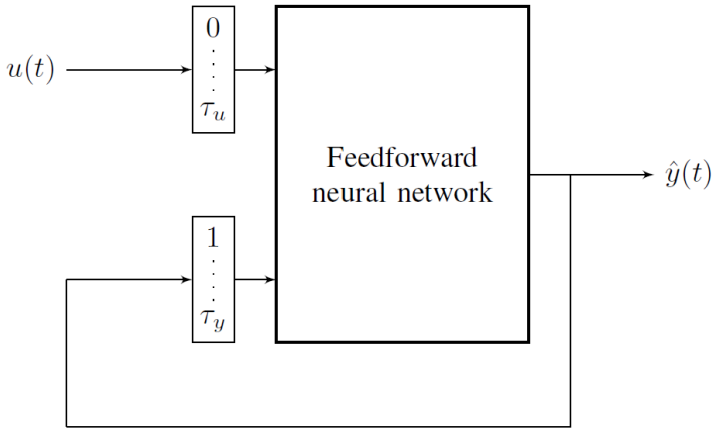
Nonlinear Autoregressive Networks with Exogenous Inputs (NARX)

- Model nonlinear dynamical systems
- Based on recurrent dynamic neural networks
- Storing past values
- Two different architectures
- Good convergence and generalization properties

Serial parallel architecture



Parallel architecture



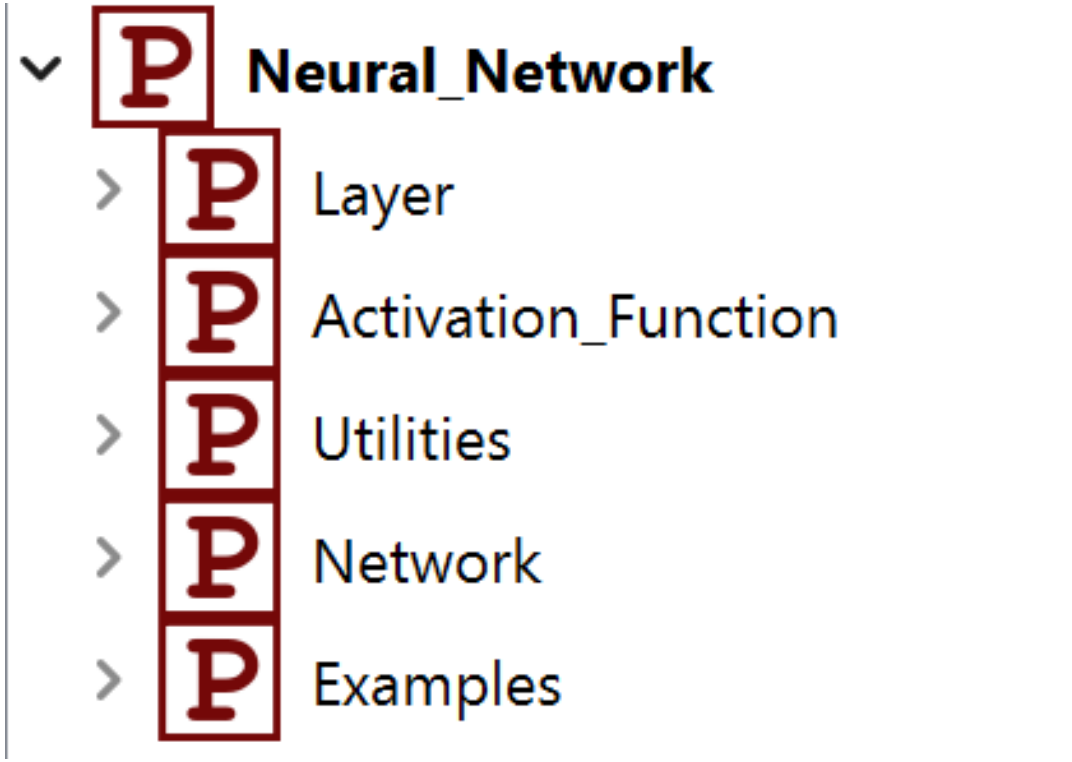
Neural Network Library

Structure

General

- Neural networks for black-box-modeling in OpenModelica
- Reactivation of *Neural Network Libaray* (Casella et. al)
- Included various data transformation elements
- Five sub-packages:
  1. Layer
  2. Activation_Function
  3. Utilities
  4. Network
  5. Examples

Overview

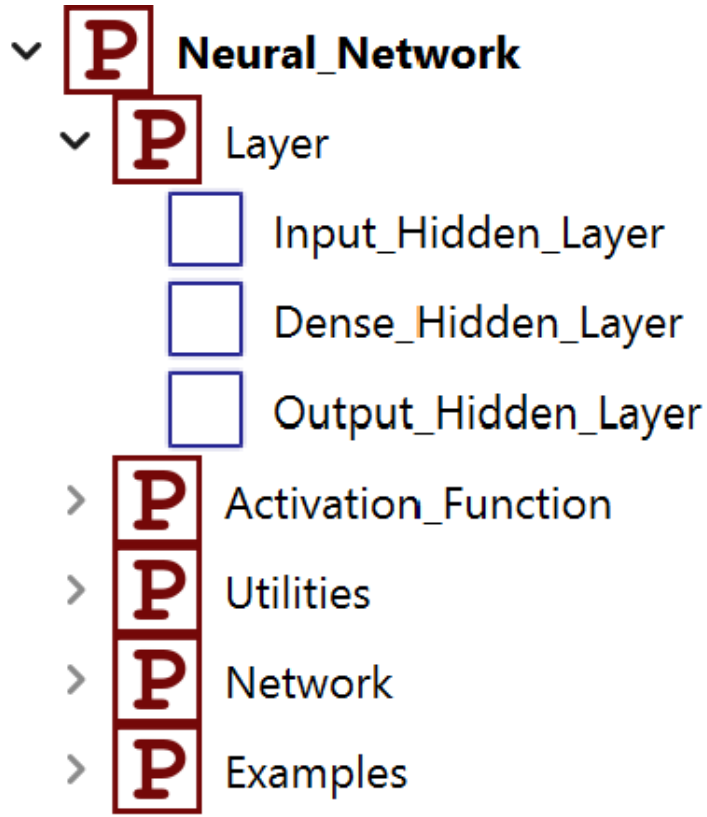


Neural Network Library

Layer

Description

- Set of neurons with weights and a given activation function
- Implemented as multiple-input-multiple-output interfaces
  - bias
  - weights
  - NeuronActivation_Function
  - numInputs
  - numNeurons
- Additional methods for data transformation of inputs and outputs
  - PCA in input layer
  - Apply hypercube and/or mu-sigma standardization
  - Rescaling of the targets

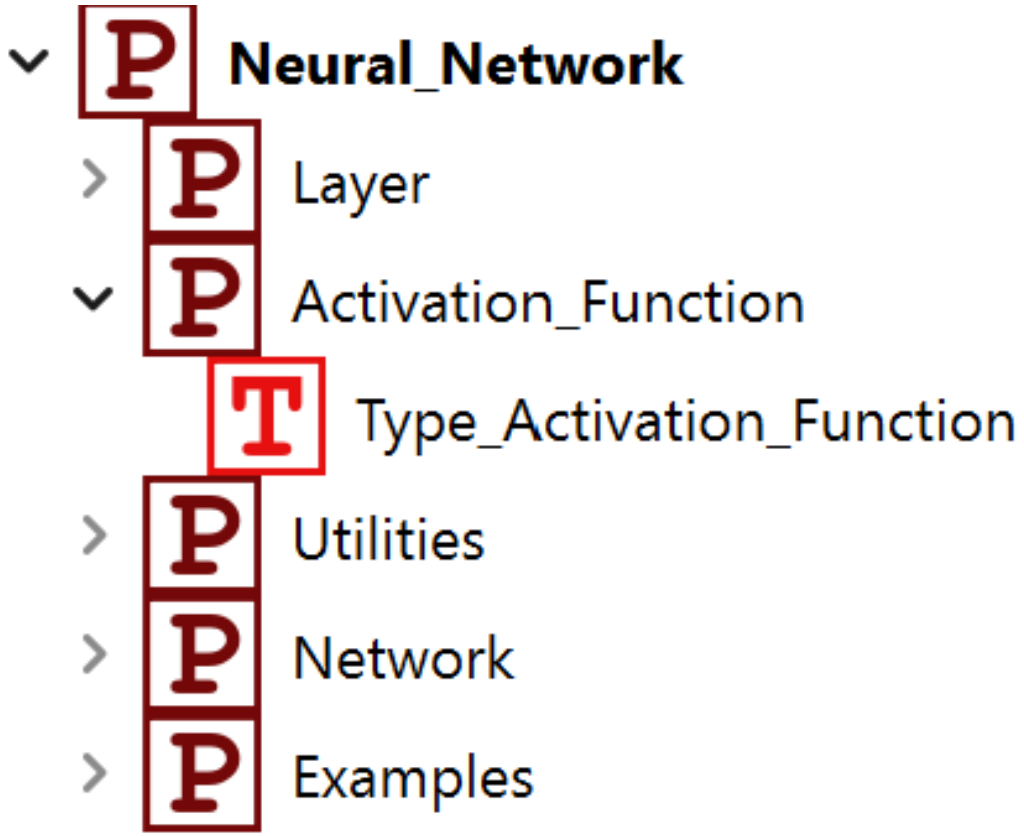


Neural Network Library

Activation_Function

Description

- Activation function describes behavior of neurons in the corresponding layer
- Selected activation function is set for whole layer
- Available activation functions:
  - ReLU
  - Sigmoid function
  - Tangent hyperbolic
  - Identity

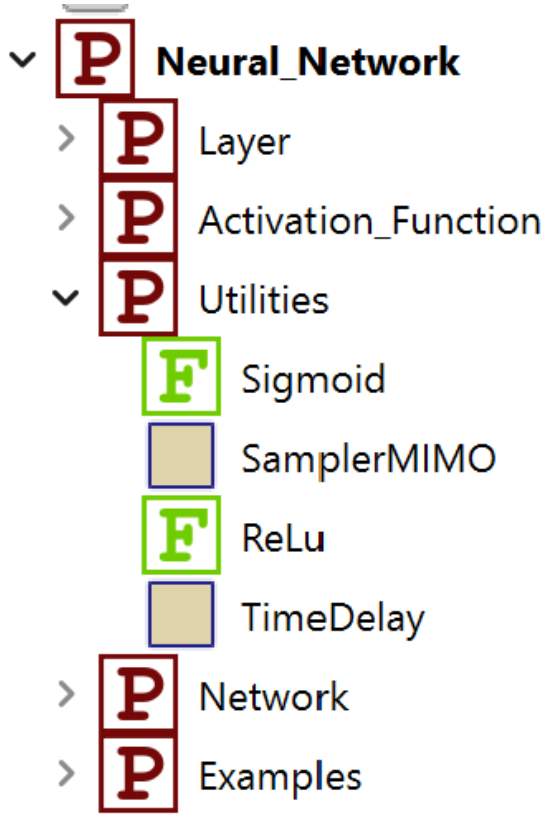


Neural Network Library

Utilities

Description

- Definition of the activation functions
- SamplerMIMO discretizes its input signals according to a selected sampling rate
- TimeDelay serves as an external shift register



Neural Network Library

Model generation

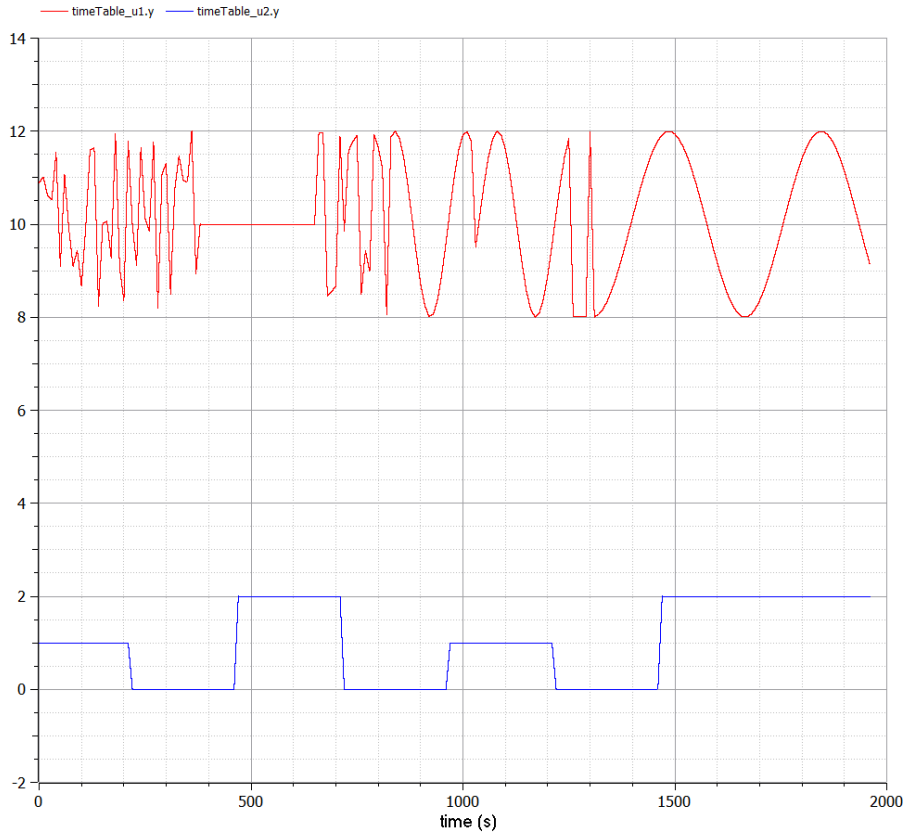
- Usefull to use already implemented frameworks for the training
- Offer a variety of learning methods and functionalities to efficiently train neural networks (early stopping, regularization, optimization methods,…)
- Use Keras for the implementation of the training
- Developed a Python module to generate a Modelica model of the neural network

Results

Case study pH neutralization process

Process

- Multivariable pH neutralization process
- Constant volume stirring tank (1100 liters)
- Two input flows
  - Acid solution (0.0032 mol/l)
  - Base solution (0.05 mol/l)



Results

Case study pH neutralization process

NARX Model

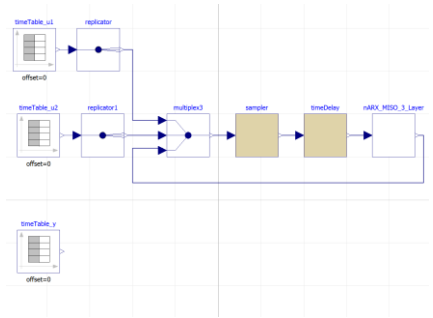
- Model the nonlinear, dynamic behavior of the system with a 2001 samples with a sampling rate of 10 seconds
  - First 201 as test set
  - Following 1800 for training
- Delay of $\tau = 4$
- No PCA

Layer	# Inputs	# Outputs	Activation
input	14	32	ReLU
hidden	32	32	ReLU
output	32	1	Identity

Pure Modelica Model

```
block NARX_MISO_3_Layer
  parameter Neural_Network.Activation_Function.Type_Activation_Function
        ActivationFunction=Neural_Network.Activation_Function.ReLu;
  Neural_Network.Layer.Input_Hidden_Layer Layer_1(bias={...},weights=[...],
        NeuronActivation_Function=ActivationFunction,numInputs=14,numNeurons
        =32,scale=true,max={...},min={...},standardization=false,mean={},std
        ={});
  Neural_Network.Layer.Dense_Hidden_Layer Layer_2(bias={...},weights=[...],
        NeuronActivation_Function=ActivationFunction,numInputs=32,numNeurons
        =32);
  Neural_Network.Layer.Output_Hidden_Layer Layer_3(bias={...},weights=[...],
        numInputs=32,numNeurons=1,rescale=false,max={},min={},destandardization
        =false,mean={},std={});
 extends Modelica.Blocks.Interfaces.MIMO(final nin=14,final nout=1);
equation
  connect(u,Layer_1.u);
  connect(Layer_1.y,Layer_2.u);
  connect(Layer_2.y,Layer_3.u);
  connect(Layer_3.y,y);
end NARX_MISO_3_Layer;
```

Graphic Modelica Model

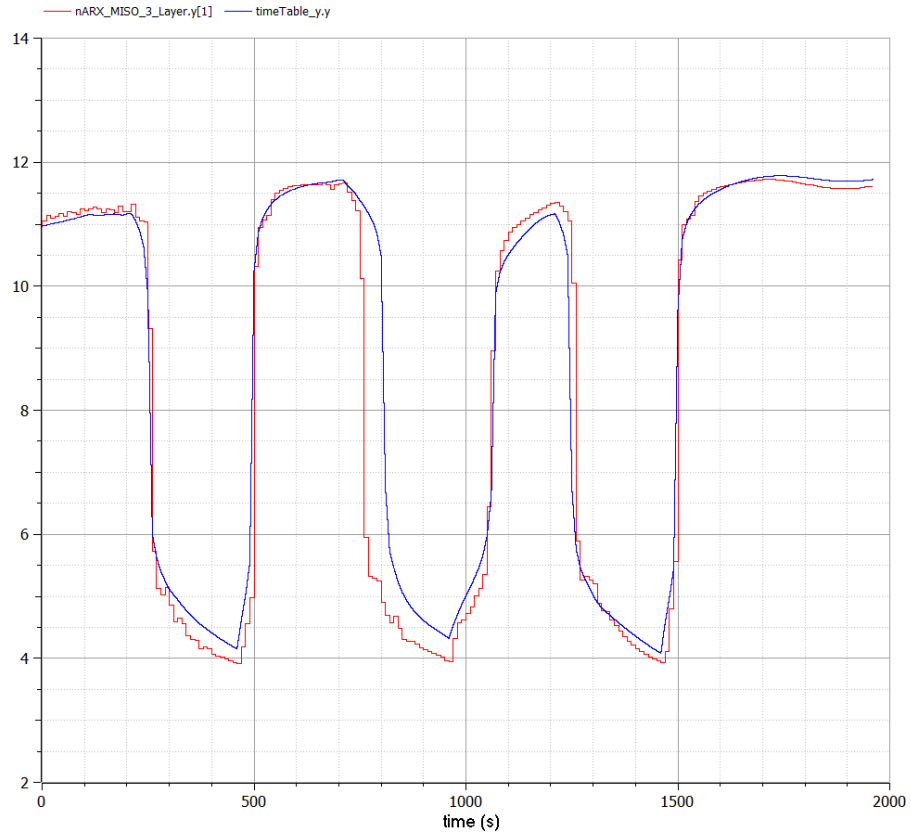


Results

Case study pH neutralization process

Results of the estimation

- NARX estimates the pH concentration of the tank
- Network can learn behavior of systems with multiple inputs
- Not always very well



Conclusion and further work

- Presented basic structure of the newly implemented *Neural Network Library*
- Explained the essential methods and procedures by which the library was extended
- Training takes place in Keras
- Interface between Keras and OpenModelica
- Presented a method to generate a Modelica model for a feedforward network
- Good foundation for making OpenModelica usable for black-box modeling
- Extend the library by further, more complex structures


ABB