HPC tool based on optimization algorithms for efficient selection of the best architecture of Artificial Neural Network

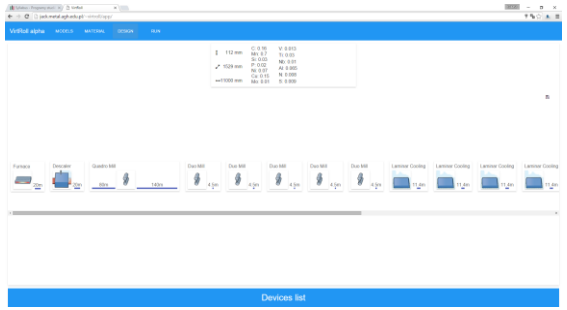
Grzegorz Górecki, Łukasz Rauch, Jakub Płaziak, Maciej Pietrzyk

Akademia Górniczo-Hutnicza, Kraków, Poland

ACK CYFGRONET AGH

System architecture

Browser



Mill design description
JSON file

High Performance Computing

Middleware Scalarm

- FEM based software
- Computing models (shared libraries)
- Additional apps: **optimization, sensitivity analysis, metamodelling**

WWW server

- Material definition
- Model selection and upload
- Design of the mill
- Configuration of calculations
- Submission of computing tasks

- Authentication
- Authorization
- Preparation of computing tasks
- Design of experiment
- Management of computing tasks
- Download of results

Objectives of the work

Creation of the numerical tool supporting determination of optimal Artificial Neural Network architecture for metamodels by using High Performance Computing infrastructures.

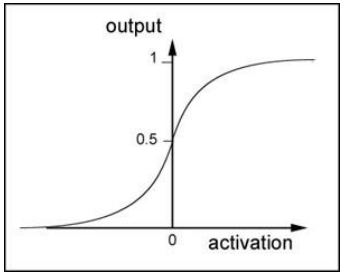
Metamodelling

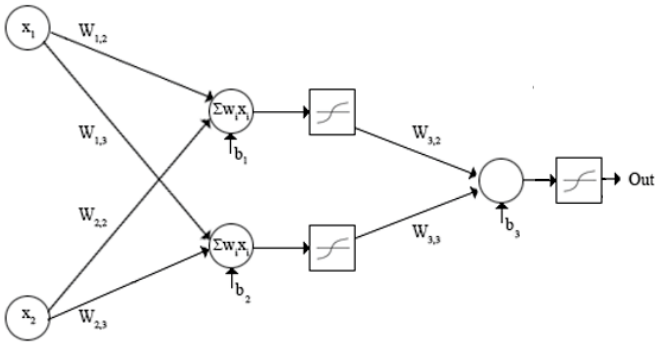
Metamodeling is a technique to build and use derivatives of models.

The main goal is to accelerate computations by replacing computationally intensive model by a black box approach

Metamodel is built and validated by using simulation data.

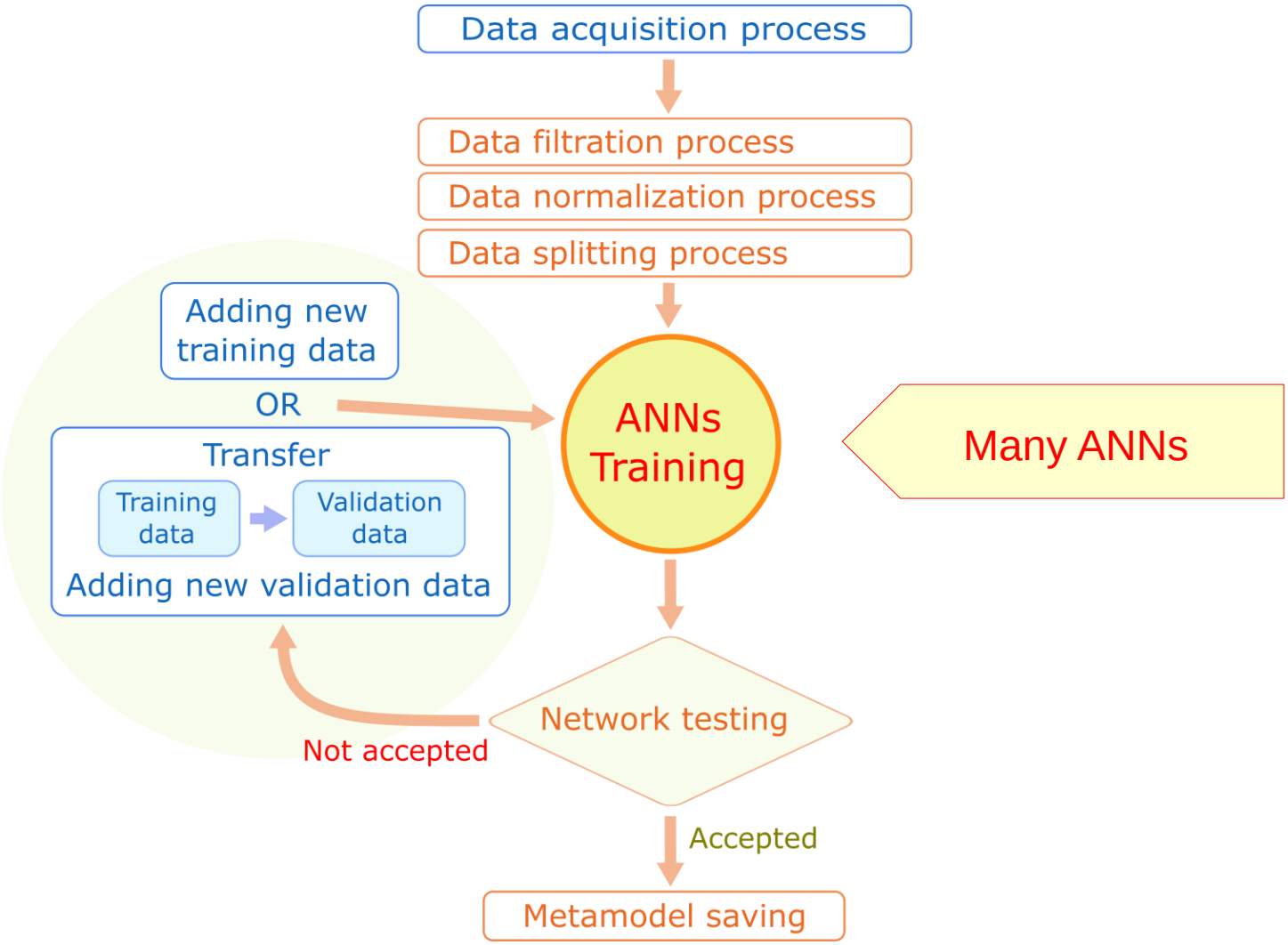
We use Kriging and Response Surface Method, but the most commonly applied approach is Artificial Neural Network. Most popular architecture is Multi-Layered Perceptron.





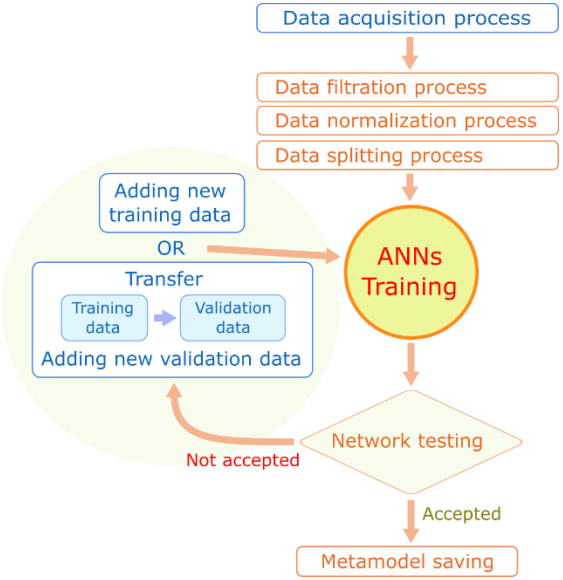
Metamodeling

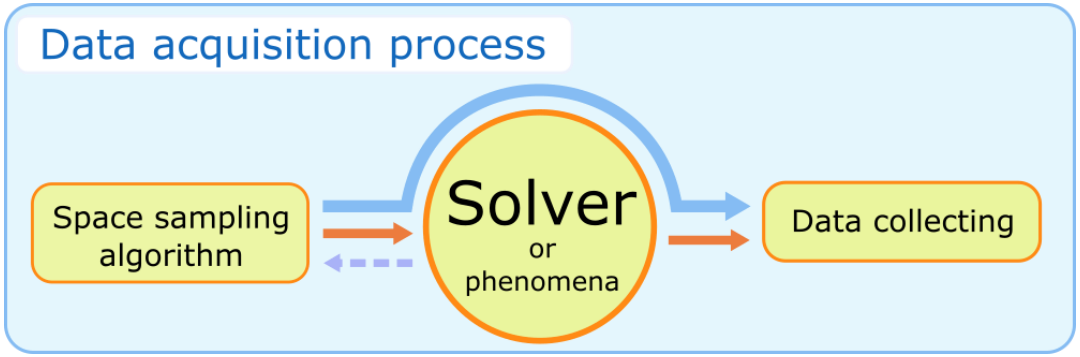
Developed software

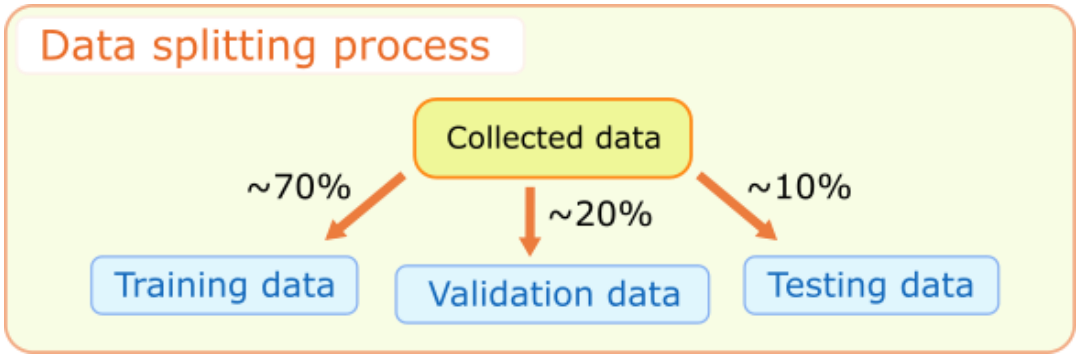


Metamodeling

Developed software



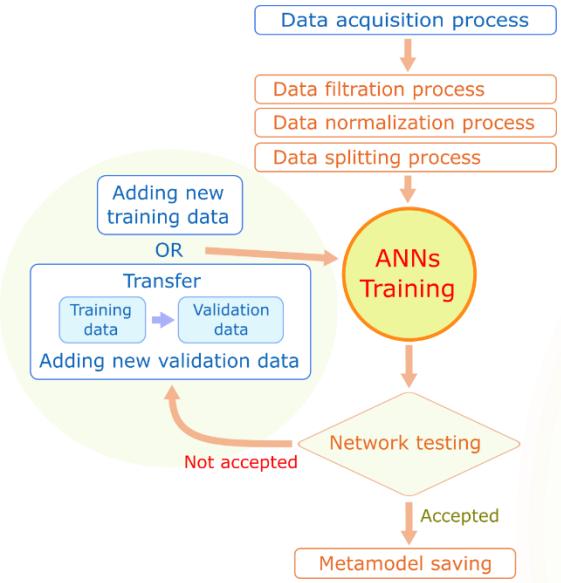


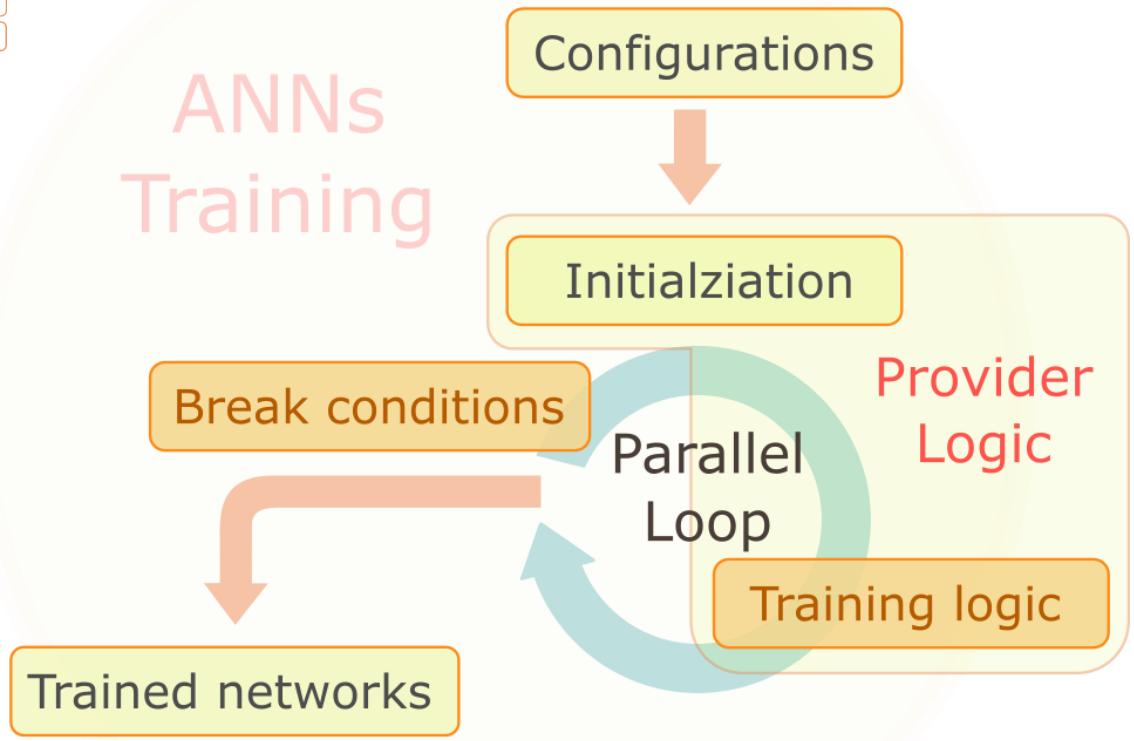


Metamodeling
Developed software

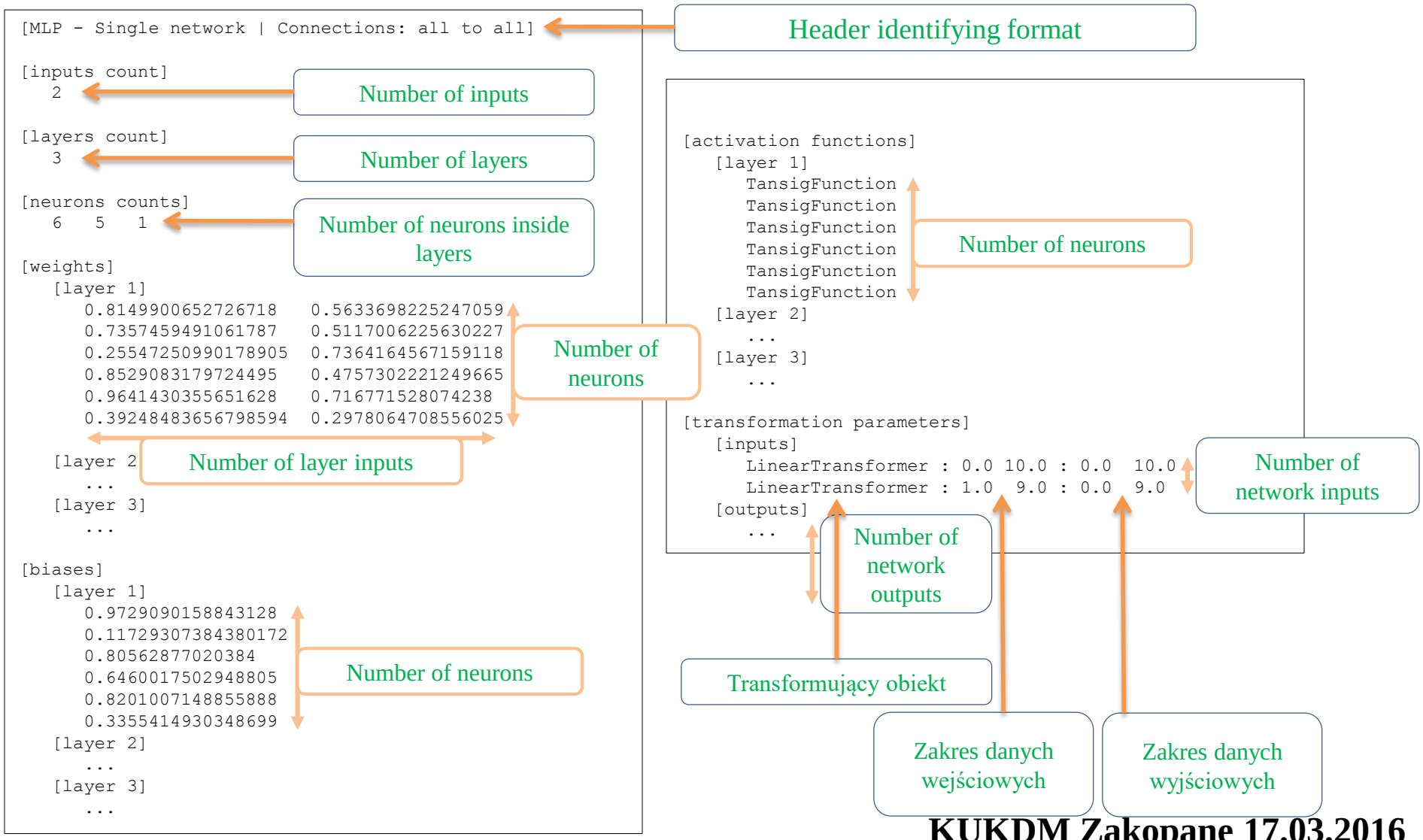
Variants:

- ANN Design
  - Number of hidden layers
  - Number of neurons in h. layers
  - Different activation functions
  - Different parameters of training
  - …
- ANN Re-training



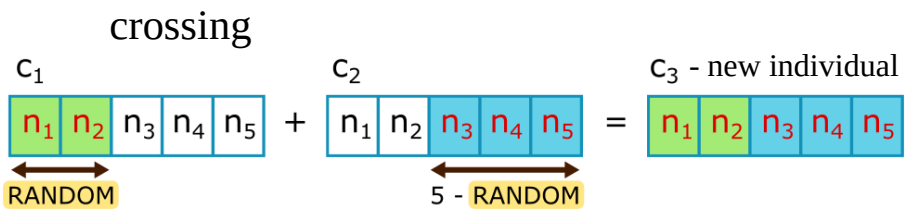


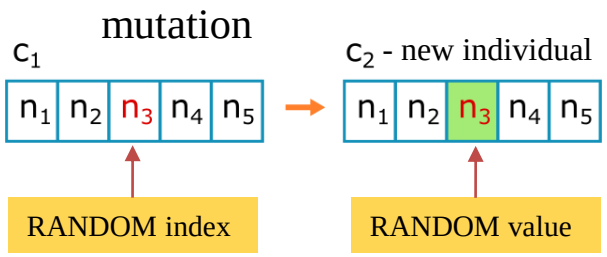
Simple ANN based metamodel format



Genetic Algorithm application idea

- Each ANN is a represented as a specimen keeping information on layers, neurons, activation functions and parameters
- The algorithm is started with random initial population between 20-40, which is distributed and trained
- The results of training are gathered on master node, where new population is generated by:





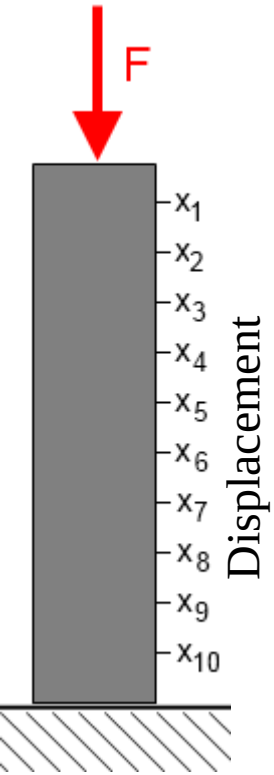
where:
c_i – individual (ANN)
n_j – number of neurons in j-th layer

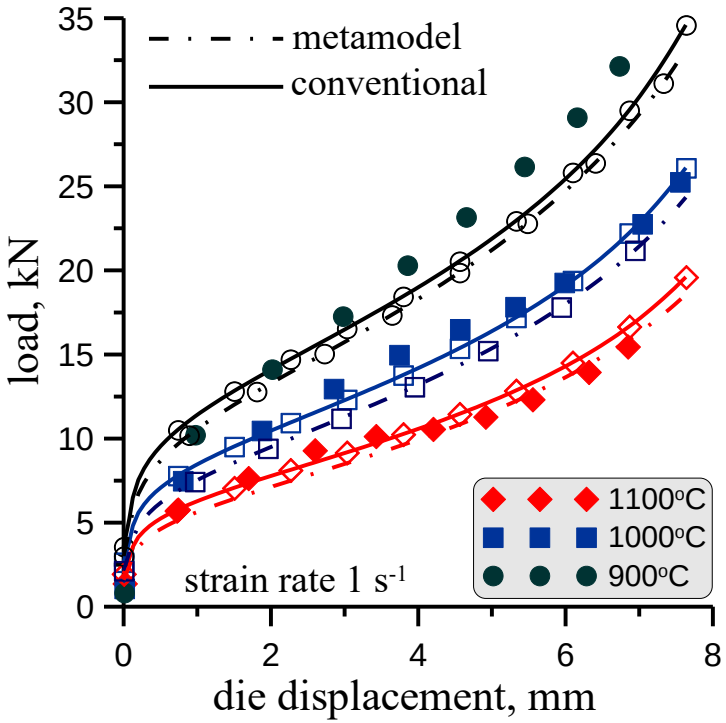
- Applied limits and parameters:
  - ✓ Population: 50
  - ✓ New individuals: 50% of current population
  - ✓ After each epoch only best individuals survives after population limit exceeded
  - ✓ Crossover: 50% probability
  - ✓ Mutation: 50% probability

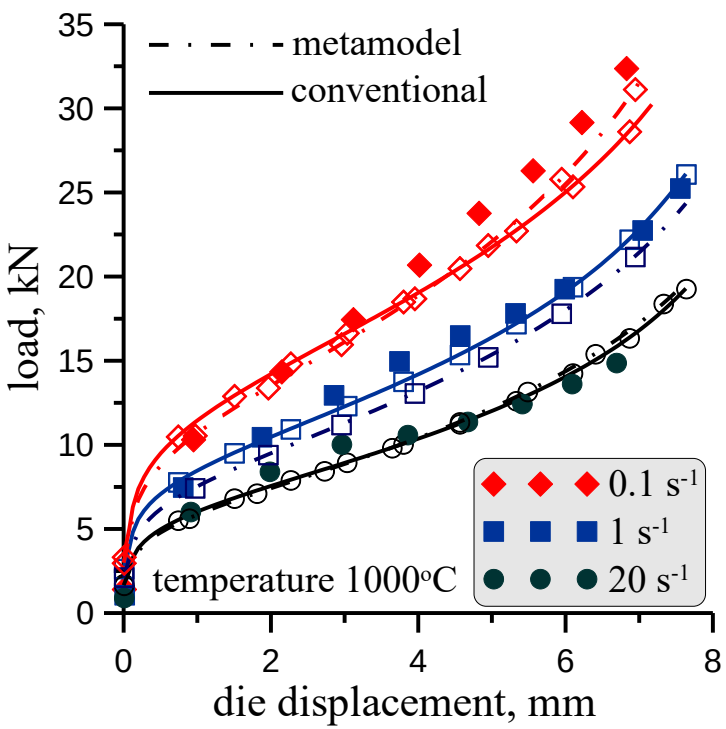
Results comparison between FEM and ANN

$$\sigma_p = \sqrt{3}\left[W p_1 \varepsilon^{p_3} \exp\left(\frac{p_2}{RT}\right) + (1-W) p_5 \exp\left(\frac{p_6}{RT}\right)\right]\left(\sqrt{3}\dot{\varepsilon}\right)^{p_4}$$

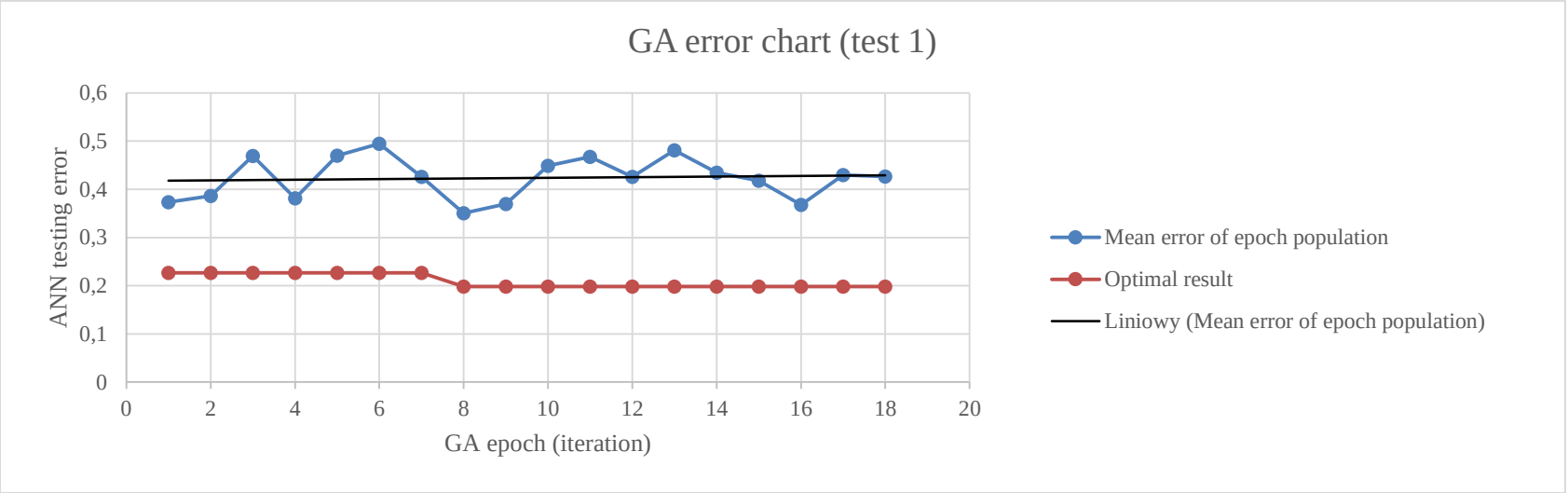
$$W = \exp(-p_7 \varepsilon)$$

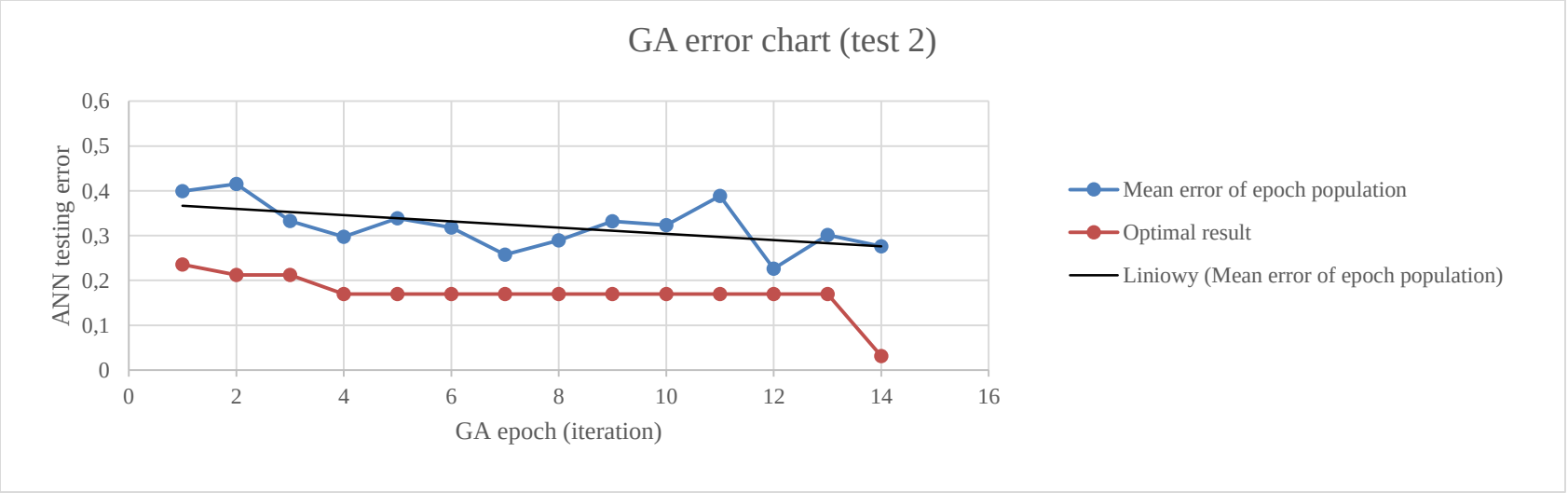






ANN errors


GA error chart (test 1)
ANN testing error
0,6
0,5
0,4
0,3
0,2
0,1
0
0
2
4
6
8
10
12
14
16
18
20
GA epoch (iteration)
Mean error of epoch population
Optimal result
Liniowy (Mean error of epoch population)


GA error chart (test 2)
ANN testing error
0,6
0,5
0,4
0,3
0,2
0,1
0
0
2
4
6
8
10
12
14
16
GA epoch (iteration)
Mean error of epoch population
Optimal result
Liniowy (Mean error of epoch population)

Conclusions

Issues realized:

- Implementation of the module for ANN design and parameterization,
- Submission of computing tasks on HPC infrastructure
- Application of genetic algorithm to obtain optimal configuration of ANN for metamodelling purposes
- Test and validation was performed

Future issues:

- Integration of ANN library with Scalarm – similar approach as in the case of optimization procedures and sensitivity analysis
- Application of created approach to industrial cases