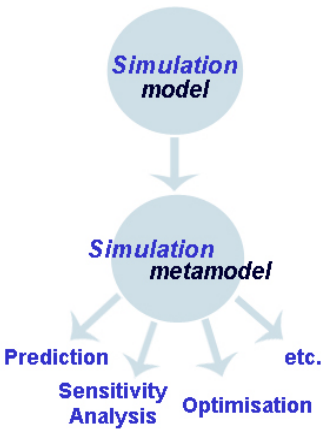
Applications of NeuroFuzzy training algorithms to simulation metamodelling

G. Merkuryeva, L. Napalkova
Department of Modelling and Simulation
Riga Technical University
1 Kalku Street, LV - 1658, Riga, Latvia
E-mail: gm@itl.rtu.lv, liana@factm.com

Simulation-based metamodelling



General aim: to support performing of simulation projects in order:

- to analyse the behaviour of complex simulation models itself,
- to improve their validation process, and
- to make easier an analysis of the simulation output data.

Metamodelling for Simulation Input-Output Approximations

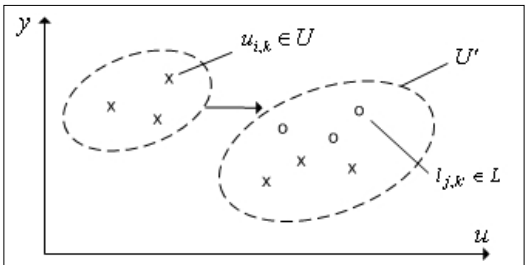
$$Us = \{u_{i,k}\}_{\substack{i=\overline{1,Ns} \\ k=\overline{1,K}}} \rightarrow \boxed{\text{Simulation model}} \rightarrow y_k$$

$$U = \{u_{i,k}\}_{\substack{i=\overline{1,N} \\ k=\overline{1,K}}} \rightarrow \boxed{\text{Simulation metamodel } \hat{y}_k = f(u_{i,k}, \delta)} \rightarrow \hat{y}_k$$

where ***Ns*** is a number of simulation model input variables, ***N*** is a number of input variables investigated, ***K*** is a set of experimental points, and δ represents an approximation error defined by an average linear error as:

$$\delta = \frac{\sum_{k=1}^{K} |y_k - \hat{y}_k|}{K}$$

ANN-based approximations



Transformation of the metamodel training set structure

ANN-based simulation metamodel, $\delta' < \delta$:

$$\hat{y}_{k'} = f'(u_{i',k'}, \delta'), i' = \overline{1,N'}, k' = \overline{1,K'}$$

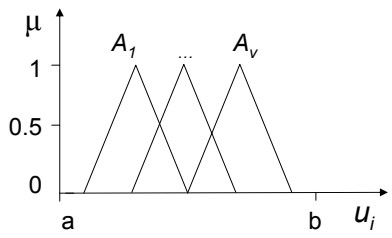
where $u_{i',k'} \in U'$, $U' \subseteq U \cup L$ is the modified training set structure, $L = \{l_{j,k'}\}$ is a set of derived variables, $j = 1,..,M$, and M is a number of derived variables.

Adaptive–Network–based Fuzzy Inference System (ANFIS)

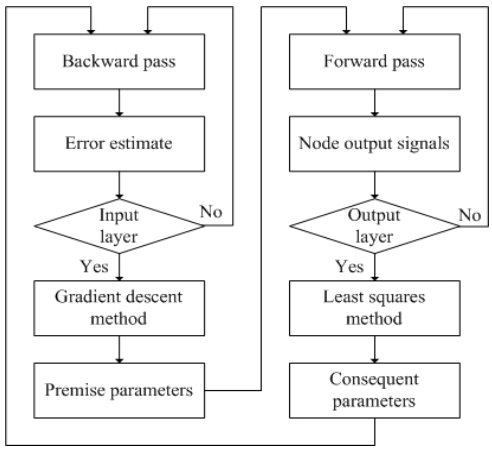
ANFIS (ANN-based metamodel) represents knowledge in the form of production rules as:

IF $g(u_i$ is $A_v)$ THEN $y = f(u_i)$

where A_v are membership functions.



Triangular membership functions for input variable u_i



ANFIS training algorithm for adjusting production rules parameters

Transformation of the Training Set structure according to ADL model

Autoregressive Distributed Lag (ADL) model

$$y_t = \alpha_0 + \sum_{j=1}^{P} \alpha_j y_{t-j} + \sum_{i=1}^{N} \sum_{j=0}^{R} \beta_{ij} u_{i,t-j} + \varepsilon_t$$

Metamodel Training Set structure

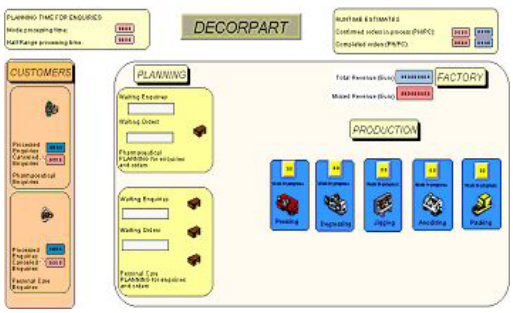
$$\{< y_t, .., y_{t-P}, u_{1,t-1}, ..., u_{1,t-R}, ..., u_{N,t-1}, ..., u_{N,t-R} >\}$$

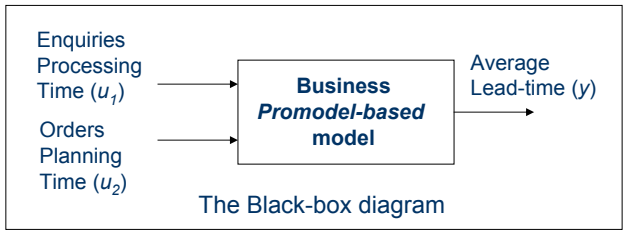
y – output variable, u – input variable,
α, β – regression coefficients, *t-j* – observation times,
N – number of input variables, P, R – lag lengths,
ε – error term.

NeuroFuzzy Training Algorithm (NFT)

- ✓ **Step 1.** ADL model identification.
- ✓ **Step 2.** Generating the training set structure according to ADL model.
- ✓ **Step 3.** Generating ANN-based metamodel structure by fuzzy production rules set.
- ✓ **Step 4.** Training simulation metamodel.
- ✓ **Step 5.** Simulation metamodel validation.

Case: Understanding relationships in the SME's business simulation model





The Black-box diagram

Testing the Proposed Algorithm (I)

Metamodel built without NFT:

Training set structure:

$$\{< y_k, u_{1,k}, u_{2,k} >\},\ K = 17$$

Sample production rule (16 rules):

IF $(u_1$ is $A_2) \wedge (u_2$ is $B_3)$

THEN $y_4 = -1549u_1 + 1037u_2 + 98.38$

where *A*, *B*, *C*, *D* represent membership functions and coefficients of linear equations are consequent parameters

Metamodel built with NFT:

Regression equation:

$$y = 9277.03 - 21.05u_1 + 4.83u_2 + 0.62(u_1)^2 + \\ + 0.41u_1u_2$$

Training set structure:

$$\{< y_{k'}, u_{1',k'}, u_{2',k'}, u_{3',k'}, u_{4',k'} >\},\ K' = 17$$

$$u_{1'} = u_1,\ u_{2'} = u_2,\ u_{3'} = (u_1)^2,\ u_{4'} = u_1u_2$$

Sample production rule (256 rules):

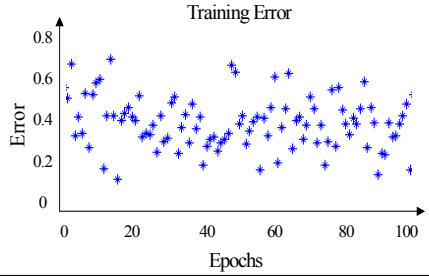
IF $(u_{1'}$ is $A_2) \wedge (u_{2'}$ is $B_2) \wedge (u_{3'}$ is $C_2) \wedge$

$\wedge (u_{4'}$ is $D_1)$ THEN $\hat{y}_{4'} = 0.1004u_{1'} +$

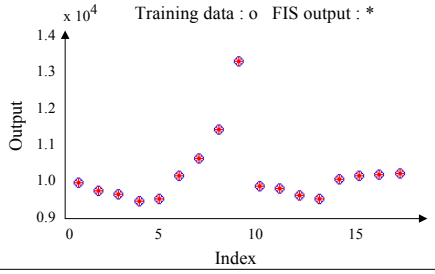
$+ 0.257u_{2'} + 3.247u_{3'} + 8.527u_{4'} + 0.00312$

Testing the Proposed Algorithm (II)

Metamodel training process



Metamodel testing process



Training

ANFIS training process includes 100 epochs. At each epoch standard error of estimate is calculated.

Testing

Simulation output data is compared with metamodel (ANFIS) output.

	Metamodel built without NFT algorithm	*Metamodel built with NFT algorithm*
Approximation Error ***(Test 1)***	**1.388**	**0.449**
Approximation Error ***(Test 2)***	**12.038**	**2.644**

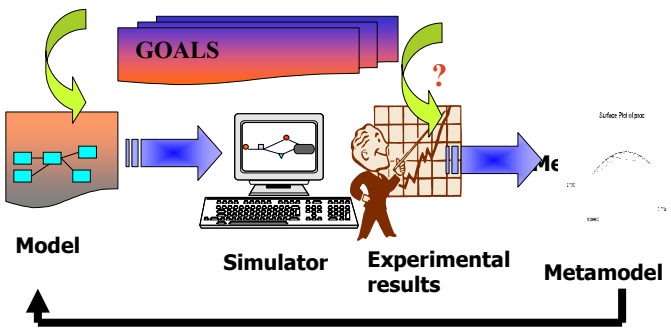
Conclusions

1. The developed NeuroFuzzy Training (NFT) algorithm is based on application of regression models to modify the training set structure of ANN-based simulation metamodel.

2. Modified training set structure is used in metamodelling knowledge base of production rules to approximate the target functional dependency with a piece-wise linear function.

3. Application of the algorithm leads to decreasing the approximation error of simulation output data.

"Metamodels are easier to manage and provide more insight than simulation alone".



Thank you for your attention !