An Algorithm to Transform an Artificial Neural Network into its Open Equation Form and its Potential Applications

Wolfram C. Rinke

Abstract— During the last decades artificial neural networks have evolved to an accepted and proven technology for modelling and function approximation. Different kinds of network architectures exist to support certain domains and applications in an efficient way. This paper assumes the traditional multilayer feedforward artificial neural network (ANN) architecture with one input layer, one or more fully interconnected hidden layers and one output layer. Each layer consists of several classic perceptron nodes using a differentiable transfer function like the logistics function. Very often it is useful to have an ANN model in an open equation form available, that allows a deeper analysis of the model and to do more complex experiments and simulations. The following paper presents an algorithm that makes it possible to transform an ANN into its open form equivalent, called process model architecture network or PMA network. It has been used as an integral part in several industrial control projects. A PMA network can be used for system simulation, scenario analyses or inverse model based control. An example application is discussed.

Keywords—artificial neural networks, inversion, model based control, open equation transformation.

I. Introduction

OVER the last decades artificial neural networks have emerged as an established powerfool tool in a broad range of engineering and scientific applications especially for process modelling and control. ANNs are also well known and widely used for data mining tasks or used for generic nonlinear function mapping applications. To find a function to map data set A to data set B can be done with different mathematical algorithms, ANNs are proved to be the best algorithms for nonlinear unknown relationships between data set A and data set B.
One of the most popular algorithms to build artificial neural network models is the algorithm for building and training an artificial neural network based on feedforward multilayer perceptrons. It is an excellent technology and has been proven as universal approximators. The error backpropagation learning algorithm presented by Rumelhart [9] is typically used for network learning. Cybenko [1] and Hornik [3] proved that any continuous mapping over a compact domain could be approximated as accurately as necessary by a feedforward artificial neural network with one hidden layer and differentiable activation function. These findings make the ANN technology so powerfull and generic, because it makes it possible to find a proper mapping function, even for difficult or impossible to describe systems, because not all involved parameters or disturbances of the observed system are known or could be measured.The theory of artificial neural networks has developed over the last 3 decades and is now well established. They are a way to describe and simulate a simple biological model of a real nervous cell and its network. The electrical charge over the connected dendrite to the nerve cell body represents one input value to the perceptron. Excitatory dendrits have a positive energy load or positive input value and inhibitorical dendrits have a negative electrical load or a negative input value. The nerve cell itself calulates a new energy level based biochemical conditions. The mathematical model uses the summation function and a scale or connection weight for each input node. The electrical output of the nervous cell is then transported over the Axon to other nervous cells.

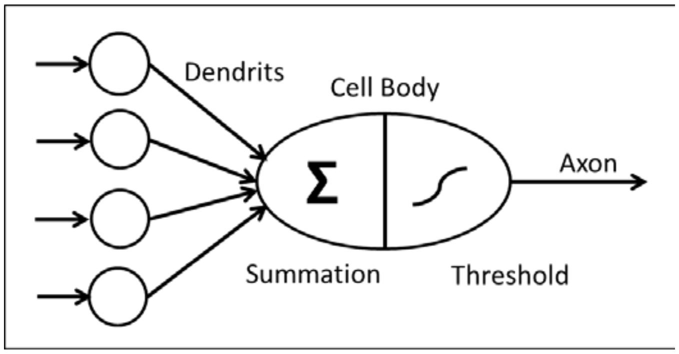


Fig.1 a typical perceptron.

The mathematical model describes the output value of each perceptron as a result of the threshold function applied to the weighted sum of input values to the perceptron according to equation (1) and (2) below.

$$net_j = \sum w_{ji} o_j \qquad \forall\, i, j. \qquad (1)$$

$$o_j = \frac{1}{1 + e^{-(net_j - \varphi_j)/\varphi_0}} \qquad \forall\, j. \qquad (2)$$

A feedforward artificial neural network is organized in typically 3 layers. One input layer, one output layer and in between at least one or more hidden layers. Each layer consists of several perceptrons or nodes. The number of nodes of the input layer depends on the number of independent variables of the model and the number of nodes of the output layer depends on the number of dependent variables. The hidden layers in between are responsible for the processing and mapping. Each node of the input layer is connected to each node of the following hidden layer and each node of the hidden layer is connected to each node of the output layer.

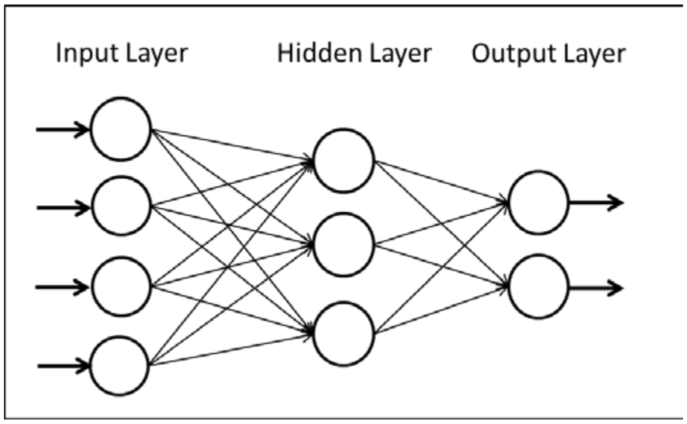


Fig.2 a typical feedforward 3 layer ANN built from many perceptrons

This kind of network, as shown in fig.2 is also called fully interconnected network. To build the proper functional mapping from the independent to the dependent variables a training process is used, which find the proper connection weights between the input and output nodes. Typically the trained ANN is interpreted as a black box. For the network training an adaptive backpropagation algorithm [9] with dynamic learning rate and momentum is used to speed up the network training process.

II. Problem Formulation

Model based control algorithms as part of advanced control strategies for industrial plants or chemical plants like gas and oil processing plants are current state of the art since the early ninety's with the invention of matrix controllers. The implementation and configuration of such controllers is quite complex and complicated. The basic idea behind such controlling algorithms is to transfer the closed form equations set into an open form equations set. [4][5] Open form equation sets are very common in chemical engineering for optimization applications because of their power and flexibility.
Apart from process control and optimization applications in industrial or other technical disciplines, there are also other application areas like in marketing or tourism research. Researchers have the need and are looking for technologies to build sufficient market and consumer models beyond basic statistics for market analyses, what-if scenario analyses or other simulation tasks. In many applications it is useful to have a simulation model of the observed systems. Experimenting with the real observed system is not always possible, because of concluded costs of production loss or other involved financial or technical risks. For example it is almost impossible to run tests on a processing plant or try certain strategies of control, when the unit is in full production mode. Testing on those plants or process units is very cost intensive, risky because of potential damage and needs also careful planning in advance. This is a typical situation you find in the process control industry all over the world.
You will also find a similar scenario in market research. The models that are for example built from customer behavior research resulting from quantified questionnaires and are typically available as descriptive statistics. These results cannot be used for simulation, unless they are approximated by some means of mathematics. Typically linear models are very popular, because they are easy to understand and simple to calculate. The drawback is a too simplified model of the observed system and complex interactions could not be modelled sufficiently. Artificial neural network technology is an approved way to build complex models for all kind of observable systems. They are a very powerful model building technology, which are described as a set of equations. But how do we get this equation set into its open form equivalent for analyses and optimization applications? A solution to this problem is presented and discussed in the next chapters of this paper.

III. Problem Solution

A. Basic Equations and ANN Equivalents

Initially we have an observable system, which consists of a set of *n* independent variables *x* and *m* dependent variables *y*. Where each dependent variable *y* can be described as a function *f* of *n* independent variables *x* as following:

$f_i(x_1, x_2, \ldots, x_n) = y_i$ for all *i* from 1, … ,m (3)
with
$net_i(x_1, x_2, \ldots, x_n) = y_i$
as the ANN equivalent to f_i

where all dependent variables y have to be independent to the other dependent variable. If this is not the case we define the system as a MISO (multiple input – multiple output) systems or the equivalent ANN

$f(x_1, x_2, \ldots, x_n) = (y_1, y_2, \ldots, y_m)$ (4)
with
$net(x_1, x_2, \ldots, x_n) = (y_1, y_2, \ldots, y_m)$
as the ANN equivalent to f

For a complex system or unknown system it is likely that it does not fulfill the requirement that all dependent and independent variables are independent to each other. Therefore we have to define a MIMO (multiple input – multiple output) system architecture as the proper equivalent model. We assume the open form equivalent of the equation (4) is as follows

$$G(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m) = 0 \qquad (5)$$

B. Algorithm to Build the Open Form Equivalent

The equation in (5) can be formed into a set of *n+m* functions with *n+m-1* variables, where each variable of (5) is represented as following:

$$
\begin{aligned}
&g_1(x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_{m-1}) = y_m \qquad (6)\\
&g_2(x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_m) = y_{m-1}\\
&\quad \dots\dots\\
&g_{m-1}(x_1, x_2, \dots, x_n, y_1, y_3, \dots, y_m) = y_2\\
&g_m(x_1, x_2, \dots, x_n, y_2, y_3, \dots, y_m) = y_1\\
&g_{m+1}(x_2, x_3, \dots, x_n, y_1, y_2, \dots, y_m) = x_1\\
&g_{m+2}(x_1, x_3, \dots, x_n, y_1, y_2, \dots, y_m) = x_2\\
&\quad \dots\dots\dots\\
&g_{m+n-1}(x_1, \dots, x_{n-2}, x_n, y_1, y_2, \dots, y_m) = x_{n-1}\\
&g_{m+n}(x_1, x_2, \dots, x_{n-1}, y_1, y_2, \dots, y_m) = x_n
\end{aligned}
$$

Each equation g_j is again modeled by a MISO type ANN $_g net_j$ with j from 1 to n+m .

$$
\begin{aligned}
&{}_gnet_1(x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_{m-1}) = y_m \qquad (7)\\
&{}_gnet_2(x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_m) = y_{m-1}\\
&\quad \dots\dots\dots\\
&{}_gnet_{m-1}(x_1, x_2, \dots, x_n, y_1, y_3, \dots, y_m) = y_2\\
&{}_gnet_m(x_1, x_2, \dots, x_n, y_2, y_3, \dots, y_m) = y_1\\
&{}_gnet_{m+1}(x_2, x_3, \dots, x_n, y_1, y_2, \dots, y_m) = x_1\\
&{}_gnet_{m+2}(x_1, x_3, \dots, x_n, y_1, y_2, \dots, y_m) = x_2\\
&\quad \dots\dots\dots\\
&{}_gnet_{m+n-1}(x_1, \dots, x_{n-2}, x_n, y_1, y_2, \dots, y_m) = x_{n-1}\\
&{}_gnet_{m+n}(x_1, x_2, \dots, x_{n-1}, y_1, y_2, \dots, y_m) = x_n
\end{aligned}
$$

In the next step the degree of freedom is defined for a model simulation and the input vector as a starting point for the iteration.

$$I_0 = ({}_0i_1, {}_0i_2, \dots, {}_0i_{n+m}) \qquad (8)$$

where I_0 is the initial input vector and where i_j is the j th input parameter of model G as defined in (5)

Next we define vector F to specify the dependent or independent variables used in the model simulation as following:

$$F = (f_1, f_2, \dots, f_{n+m}) \quad \forall f \in \{0, 1\} \qquad (9)$$

with f=0 dependent variables and f=1 for independent variables in G

Finally we introduce two termination conditions for the iteration algorithm. The first condition T limits the number of iteration steps *t* with

$$T>0 \text{ and } T \in N \qquad (10)$$

with N as the set of natural numbers.

And the second condition is a stability criteria ε with

$$\varepsilon > 0 \text{ and } \varepsilon \in R \qquad (11)$$

with R as the set of real numbers.

The input vector *I* for iteration step t is defined as

$$I_t = ({}_ti_1, {}_ti_2, \dots, {}_ti_{n+m}) \qquad (12)$$

where I_t is the input vector of the iteration *t* and $_ti_j$ is defined as

$${}_ti_j = (1 - f_j) * {}_gnet_j^{t-1} + f_j * {}_{t-1}i_j \qquad (13)$$

where $_gnet_j^t$ is the value of the j th ANN mode $_gnet$ at iteration t

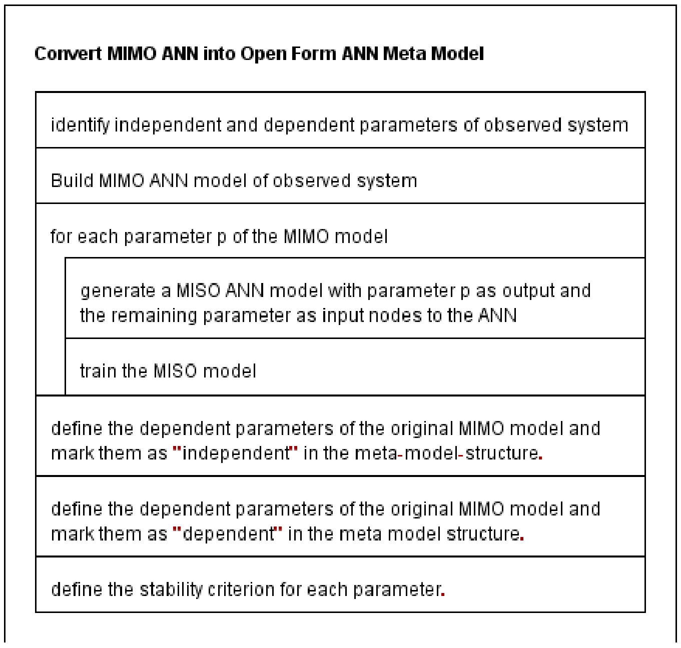

Convert MIMO ANN into Open Form ANN Meta Model
identify independent and dependent parameters of observed system
Build MIMO ANN model of observed system
for each parameter p of the MIMO model
generate a MISO ANN model with parameter p as output and the remaining parameter as input nodes to the ANN
train the MISO model
define the dependent parameters of the original MIMO model and mark them as "independent" in the meta-model-structure.
define the dependent parameters of the original MIMO model and mark them as "dependent" in the meta model structure.
define the stability criterion for each parameter.

Fig.3 NS-Diagram to describe the transformation

In Fig.3 the algorithm is shown as a Nassi-Shneiderman (NS) diagram.

C. Algorithm to Evaluate the Open Form Equivalent

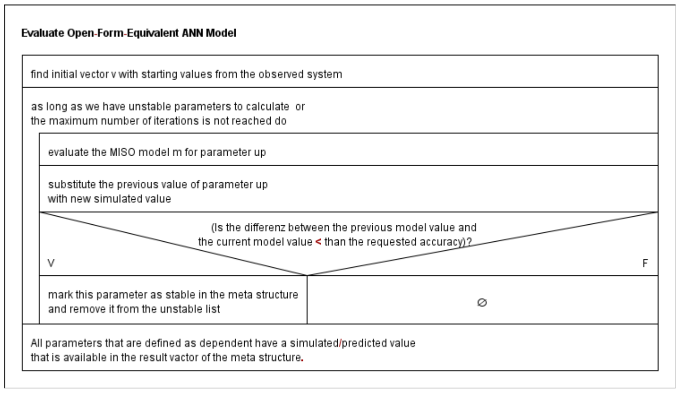
Based on the above transformation and conditions the iterative algorithm to calculate the values of the dependent variables is as following:

1) Find a proper ANN model that match the function according to (4)
2) Build all MISO ANN models for each parameter of the ANN model that was found in step 1 according to (7)
3) Train each ANN model until the requested accuracy is reached.
4) Define the vector F and decide which of the model parameters are treated as dependent or independent variables.
5) Define input vector I0 with xi and yj for all i and j. Appropriate initial values of y can be found by evaluating the ANN defined in (4).
6) For each gnetj calculate the network value, iff the value of fj is 0, which means the model parameter

with index j is a dependent variable or the value of parameter j is assumed to be stable.

7) Compare the network value of parameter j at iteration step t with its previous value of iteration t-1.
8) If following condition | gnetjt - gnetjt-1 | < ε is valid, the value of gnetjt is assumed to be stable.
9) Prepare the input vector It+1 for the next iteration step.
10) If all ANN models have stabilized or the number of maximum iterations has been reached, the iteration algorithm terminates otherwise follow step 10.
11) Repeat steps 6 to 9 for all ANN models, which have not stabilized yet during the iteration process.

After the algorithm has terminated, the input vector I holds the final values for the simulated dependent variable. All the ANN models representing each variable of the observed system $_{g}net_{j}$ is organized in one meta-model, named "process model". It contains all the required information as shown in Fig.6.



```
Evaluate Open-Form-Equivalent ANN Model

find initial vector v with starting values from the observed system
as long as we have unstable parameters to calculate  or
the maximum number of iterations is not reached do
    evaluate the MISO model m for parameter up
    substitute the previous value of parameter up
    with new simulated value
    (Is the differenz between the previous model value and
    the current model value < than the requested accuracy)?
    V
        mark this parameter as stable in the meta structure
        and remove it from the unstable list
    F
        ∅
All parameters that are defined as dependent have a simulated/predicted value
that is available in the result vactor of the meta structure.
```

Fig.4 NS-Diagram to describe the meta model evaluation

IV. Sample Application

A typical process control scenario shows a bypass controlled counter flow heat exchanger. A typical counter flow heat exchanger layout is shown in Fig5 below.

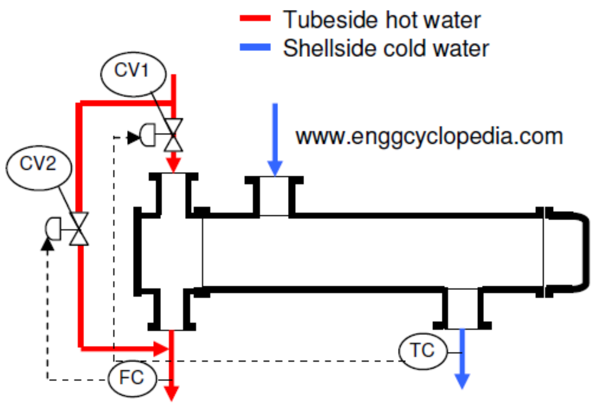
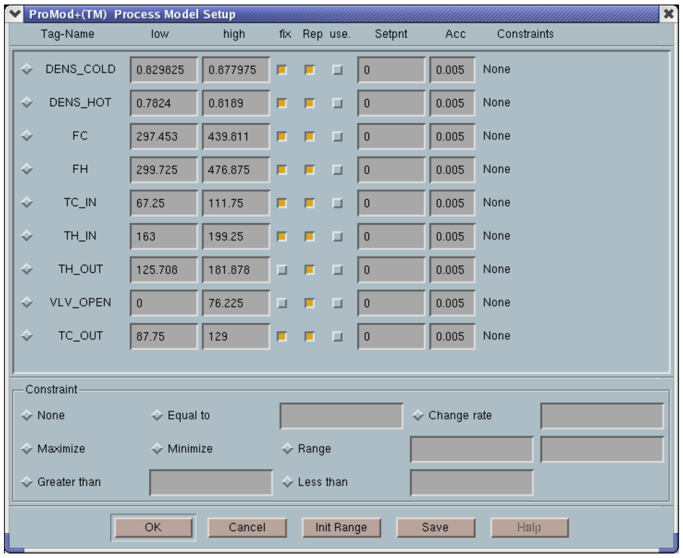


Fig.5 Schematic of a bypass controlled heat exchanger

This application demonstrates the usage and flexibility of the described algorithm. The system and data used in this example is part of a real and existing monitoring application and the heat exchanger is part of the preheater chain of a crude distillation unit of a crude oil refinery in Austria. The observed heat exchanger system is described by following variables:

FC	feed of the cold side in tons per hour
FH	feed of the hot side in tons per hour
DENS_COLD	online density of the cold flow one hour average
DENS_HOT	online density of the hot feed in one hour average values
TC_IN	input temperature as hour average (cold side) in degree Celsius
TC_OUT	output temperature as hour average (cold side) in degree Celsius
TH_IN	input temperature as hour average (hot side) in degree Celsius
TH_OUT	output temperature as hour average (hot side) in degree Celsius
VLV_OPEN	percentage of opening the valve located in the hot bypass stream

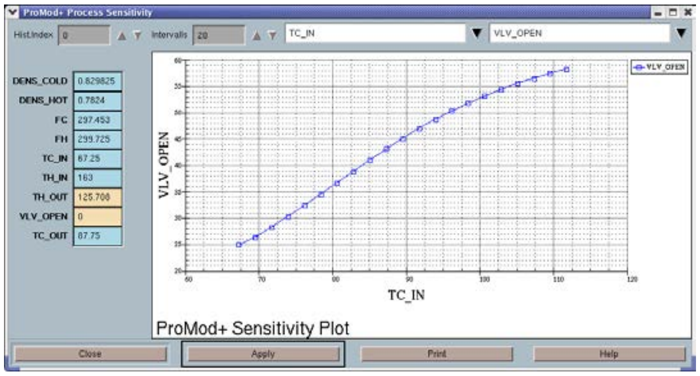
For a rigorous equation based model several additional variables are required to describe the process. Like the heat transfer coefficient or the size of the transfer zone of the heat exchanger. But for the ANN based observation model these parameters are not required, as they are implicitly modeled. First we build the MIMO model, which models and calculates or predicts the TC_OUT and TH_OUT variables. This is done with a three layer ANN, using one hidden layer. The data for training was directly collected from the process control system.



Tag-Name	low	high	fix	Rep	use.	Setpnt	Acc	Constraints
DENS_COLD	0.829825	0.877975	✓	✓		0	0.005	None
DENS_HOT	0.7824	0.8189	✓	✓		0	0.005	None
FC	297.453	439.811	✓	✓		0	0.005	None
FH	299.725	476.875	✓	✓		0	0.005	None
TC_IN	67.25	111.75	✓	✓		0	0.005	None
TH_IN	163	199.25	✓	✓		0	0.005	None
TH_OUT	125.708	181.878		✓		0	0.005	None
VLV_OPEN	0	76.225		✓		0	0.005	None
TC_OUT	87.75	129	✓	✓		0	0.005	None

Fig.6 Configuration form to configure meta data.

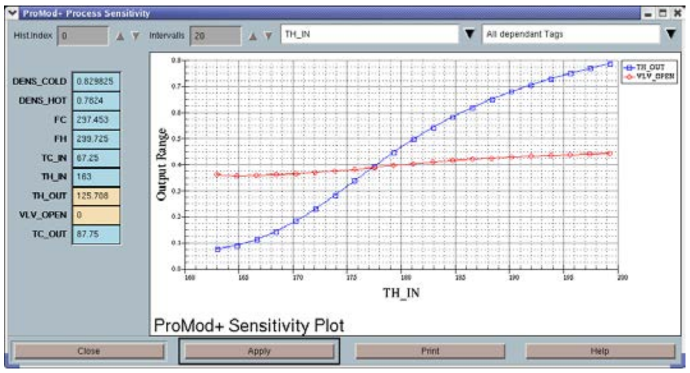
The open form equivalent data configuration form shown in Fig.6 has all meta information as currently implemented in the modelling toolkit, that was used for the simulation. There is some extra information shown in the bottom of the form that is used for a constraint based optimizer. This optimizer sits on top of the meta model, but it is not necessary for the open form equivalent model described in this article.



HistIndex	0
Intervalls	20
	TC_IN
	VLV_OPEN
DENS_COLD	0.829825
DENS_HOT	0.7824
FC	297.453
FH	299.725
TC_IN	67.25
TH_IN	163
TH_OUT	125.708
VLV_OPEN	0
TC_OUT	87.75

Fig.7 Simulated valve position dependent of TC_IN

In Fig.7 the setpoint of the bypass valve is shown, which calculated by the inverted model. In this case the input temperature of the cold stream is modified over its input range for the simulation.

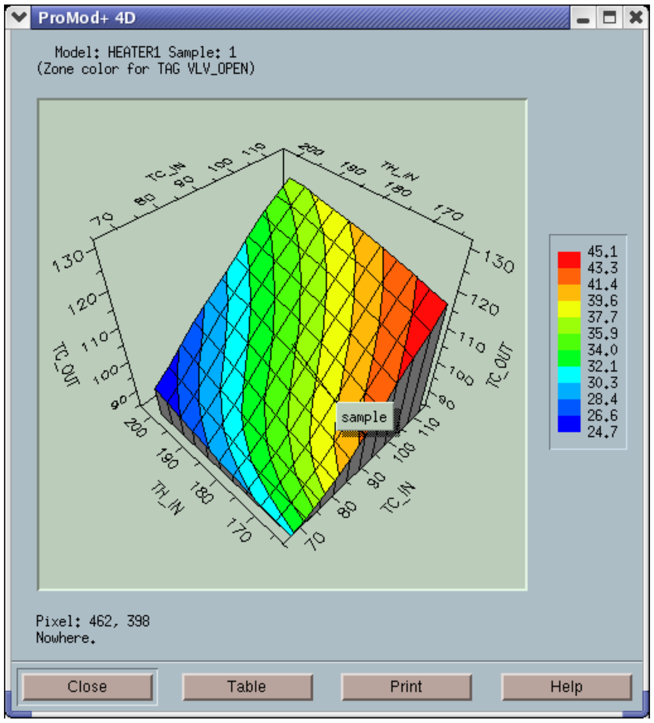


HistIndex	0
Intervalls	20
	TH_IN
	All dependant Tags
DENS_COLD	0.829825
DENS_HOT	0.7824
FC	297.453
FH	299.725
TC_IN	67.25
TH_IN	163
TH_OUT	125.708
VLV_OPEN	0
TC_OUT	87.75

Fig.8 Simulating the control curve for valve with fixed flows and target temperature

In Fig.8 the result from model simulation of the two dependent variables TH_OUT (output temperature of the hot stream) and VLV_OPEN (open position of the bypass valve) is shown. The open form model has also learned the control algorithm that is used for the bypass valve, which seems to be linear. The simulator tries to increase the bypass of the hot stream to reduce the amount of heat transfer to keep the TC_OUT setpoint at about 88°C.

In Fig.9 the model results of the simulated output temperature of the hot stream and the control surface for the bypass valve (overlaid color surface) is shown together in a 4D plot. For this simulation the independent variables for the input temperature on the cold side (TC_IN) and the hot side (TH_IN) of the heat exchanger are varied over their value range. The open form model predicts the bypass valve position to keep the outlet temperature on the cold side of the stream (TC_OUT) close to its setpoint.



```
Model: HEATER1 Sample: 1
(Zone color for TAG VLV_OPEN)
```

```
Pixel: 462, 398
Nowhere.
```

Fig.9 Control surface of the bypass valve

The sensitivity analysis [3] of the bypass valve position (VLV_OPEN) in relation to the outlet temperature of the cold stream (TC_OUT) of the heat exchanger is shown in Fig.10.. For this analysis all other independent variables remain constant.

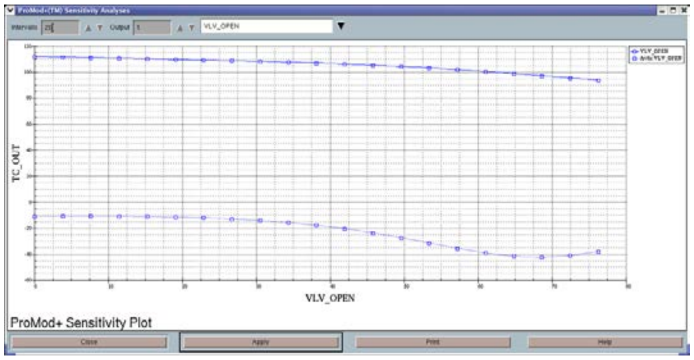


Fig.10 Sensitivity plot of the simulated output temperature of the cold stream

V. Conclusion

This paper present a novel algorithm to transform a multilayer feed forward artificial neural network into its open form equivalent meta model. The resulting architecture leads to one separate ANN model per variable that depends on the remaining variables of the observed system. The additional meta structure is used to keep track of the open form model configuration, like the iteration limits or the list of dependent and independent variables used for a simulation run. The presented architecture does not interfere with the internal structure of the ANN or its perceptron connections, because the resulting architecture from the algorithm is a network of networks configured in a recurrent way. The open form model can be easily combined with an optimization algorithm like simulated annealing. A non-trivial control application is presented and discussed. The results and the power of the open form ANN model and the transformation algorithm is shown.

Model inversion is a commonly used in optimization applications in the process control industry. The open form model has also a broad range of applications in other research disciplines like marketing or empirical modelling. It is a powerful tool to model the results of empirical studies. Any feedforward ANN models can be transformed into its open form equivalent and used for scenario analyses or what if analyses after the transformation.

References

[1] Cybenko, G., Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signal and Systems, 2:303-314, 1989

[2] Eftekhar B, Mohammad K, Ardebili HE, Ghodsi M, Ketabchi E, Comparison of artificial neural network and logistic regression models for prediction of mortality in head trauma based on initial clinical data. BMC Med Inform Decis Mak, 5:3. PubMed Abstract, 2005

[3] Hashem Sherif, Sensitivity Anlyses for Feedforward Artificial Neural Networks with Differentiable Activation Functions, Proceedings 1992 IJCNN vol1 pp.419-424, IEEE (1992), ISBN 0-7803-0559-0, 1992

[4] Hornig K., Stinchcombe M., and White H., Multilayer feedforward networks are universal approximators, Neural Networks, 2:359-366, 1989

[5] Iserman, Ralf, Adaptive Control Systems, ISBN 0-13-005414-3, BPCC Wheatons Ltd, 1991

[6] Omidvar O., Elliott D. L., Neural Systems for Control, Academic Press, ISBN 0-12-526430-5, 1997

[7] Pao Yoh-Han, Adaptive Pattern Recognition and Neural Networks, Addison-Wesley Publishing Company, ISBN 0-201-12584-6, 1989

[8] Patterson, Dan W., Artificial Neural Networks: Theory and Applications, Prentice Hall Simon & Schuster, ISBN 0-13-295353-6, 1996

[9] Rumelhart, D. E., Hinton, G. E. & Williams, R. J. in Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations (eds Rumelhart, D. E. & McClelland, J. L.) 318−362 MIT, Cambridge, 1986

[10] Runkler, Thomas A., Data Mining, Vieweg+Teubner Fachverlag, Wiesbaden 2010, ISBN 978-3-8348-0858-5, 2010

[11] Swingler, Kevin, Applying Neural Networks, Academic Press Harcourt Brace & Company Publishers, ISBN 0-12-679170-8, 1996

[12] Terrin N, Schmid CH, Griffith JL, D'Agostino RB, Selker HP, External validity of predictive models: A comparison of logistic regression, classification trees, and neural networks, Journal of clinical epidemiology, ISSN 0895-4356, 2003

Wolfram C. Rinke owns a master in computer science from the Technical University of Vienna, Austria, Europe in 1984 and a degree in data technology from the Technical University of Vienna, Austria, Europe in 1982. Next, the author’s educational background is listed. The degrees should be listed with type of degree in what field, which institution, city, state or country, and year degree was earned.

He has over 20 years of professional work experience in applied artificial intelligence, covering several industries like oil and gas and stock markets. He was CEO and CTO of two companies implementing industrial artificial intelligence applications for oil refineries like applications for furnace scheduling, process monitoring, process scheduling and process diagnoses. Since 2005 he is member of the regular teaching stuff at the University of Applied Science Burgenland at Eisenstadt, Austria, Europe. His publications are about quantification of empirical studies using artificial neural networks and about a method, the dependency matrix, he invented in the early 90’s as part of a ANN based process modelling toolkit. His current research interest is focusing on big data and data mining and its applications.

Dipl.-Ing Rinke is member of the Austrian Society of Artificial Intelligence for more than 30 years.