University of Stavanger

ANNMD - Artificial Neural Network Model Developer

Jure Smrekar

June 2010

University of Stavanger
N-4036 Stavanger
NORWAY
www.uis.no

© 2010 Jure Smrekar

ISBN: 978-82-7644-416-2

Abstract

This booklet presents a tool for Artificial Neural Network (ANN) modelling called the Artificial Neural Network Model Developer (ANNMD). The ANNMD is a generic tool for system modelling with ANNs and its development is based on initiatives from previous experiences in ANN modelling of energy systems. The strength of the tool is:

- Automated modelling procedure with respect to initial weights, data randomization, number of neurons in the hidden layer, sensitivity analysis and generalization aspect which are thoroughly cross-checked and will eventually contribute to optimal ANN-model solutions.
- Besides the basic parameters, which are important in every ANN modelling, other options such as different network types, learning algorithms, transfer functions, etc. are available as well, enhancing the utility of the ANNMD.
- User-friendly interface which enables easy ANN modelling with graphically presented results making decisions on final model selection easier.
- Detailed record of the tested ANNs in sensitivity analysis that include ANN models for implementation in real life, important information about accuracies, training records, etc.
- Employing full computational power of multiple core computers enabling fast modelling times.

The development of the ANNMD is a part of a research project which takes place at Risavika Gas Centre (RGC) in Norway. The main activities related to this project are theoretical and experimental investigation of biogas fuelled technologies using advanced and intelligent modelling and monitoring tools. The project anticipates tests on three experimental systems, i.e. gas turbine, gas engine and fuel cell. Hence, the ANNMD provides a necessary generic tool for the modelling purposes of the energy systems at RGC.

Contents

1. Introduction

Artificial Neural Network (ANN) modelling has been proven to be a powerful tool in energy system applications [1-8]. Its role is increasing with improving ANN algorithms that make ANN applications on real systems even more accurate and reliable. Since ANN modelling can be used for different purposes, its improvements reflect in different areas such as system simulation, monitoring, sensor validation, staff training, etc. Due to its increasing applicability, the need for a powerful tool [9] for automated system modelling has emerged.

Today there are many programs for ANN modelling, e.g. NeuroSolutions [10], Emergent [11], JavaNNS [12], Neural Lab [13], Matlab ANN toolbox [14], etc., that are available on the market. However, none of them provide automated modelling procedure taking into account general parameters such as initial weights, data randomization, number of neurons in the hidden layer, etc. that are important in every ANN modelling and need to be cross-checked in order to find the optimal solution. Finding the best solution according to the operator's requirements, demands that many different aspects, such as sensitivity analysis (SA), ANN structures, learning algorithms, model validation, etc., be examined and evaluated during modelling of a system. This requires expert knowledge and a lot of human effort in modelling which is time-consuming and increases the possibility for human error-making. Therefore, solutions that enable a user to model a system in a user-friendly, easy, fast and reliable way have to be provided [9].

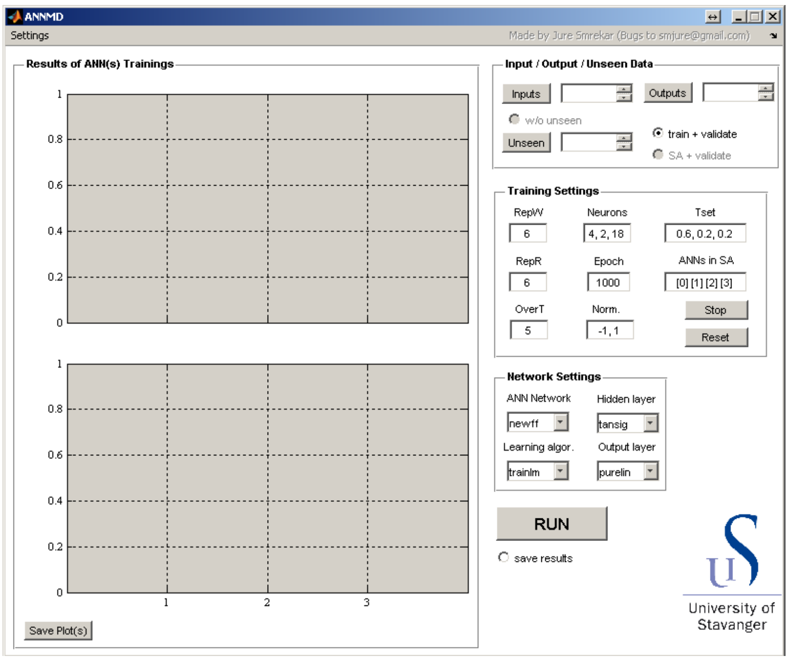
One such solution is described in this booklet which presents a generic tool for ANN modelling called the Artificial Neural Network Model Developer (ANNMD). ANNs are being used in many different fields (energy science, mechatronics, biology, etc.), and the number of these is growing. Besides, ANNs are applied in many different tasks such as prediction, classification, time series projection, etc. that require different structures and features to be examined. Despite this diversity, most of the tasks involve some basic parameters that are common in every type of ANN modelling. Starting from this fact, the ANNMD is designed as a generic tool that can automatically cross-check the basic parameters such as initial weights, data randomization, number of neurons in the hidden layer and ANN input parameters (sensitivity analysis). Furthermore, it also comprehends specific features, for instance different network types, learning algorithms, normalization rate, etc., that can be specified and easily examined with a user-friendly interface. An additional feature that takes into account a generalization aspect as one of the most important factors in ANN modelling is also implemented in the ANNMD.

The ANNMD is based on static ANN architecture with one hidden layer including different network types such as Multi-Layer Perceptron (MLP) [15], cascade-forward network, pattern recognition network, etc. that are generally used in the modelling of energy systems and many others. By considering different network types modelling, sensor validation and fault diagnosis tasks can be handled by the use of the ANNMD.

The idea for the ANNMD emerged from the previous experiences in ANN modelling of energy systems. The realization of the idea is connected with a research project at Risavika Gas Centre (RGC) in Norway. With a focus on energy systems, the project involves theoretical and experimental investigation of biogas fuelled technologies using advanced and intelligent modelling and monitoring tools. Hence, the ANNMD provides the base tool for modelling three experimental systems, i.e. gas turbine, gas engine and fuel cell, at RGC.

2. Default layout of the ANNMD

One of the main goals of creating the ANNMD was to make an independent application for ANN modelling. This makes ANN modelling also possible for users that are not skilled in programming. Thus, such software must be as simple as possible without having any program-special features which would require additional expert knowledge. In this respect the simple Graphic User Interface (GUI) for the ANNMD was created, as shown in Fig. 1. Despite the simplicity, all the main features for modelling with static ANN networks with one hidden layer are available.



w/o unseen
[✓ train + validate]
SA + validate

Training Settings

RepW	Neurons	Tset
6	4, 2, 18	0.6, 0.2, 0.2
RepR	**Epoch**	**ANNs in SA**
6	1000	[0] [1] [2] [3]
OverT	**Norm.**	
5	-1, 1	

Network Settings

ANN Network	Hidden layer
newff	tansig
Learning algor.	**Output layer**
trainlm	purelin

save results

Figure 1: Default layout of the ANNMD tool.

Due to Matlab limitations on compiling a standalone application with Matlab ANN toolbox, Matlab installation is required to run the ANNMD. By simply writing ‘ANNMD’ in the

Command Window, the default layout of the ANNMD will appear, as presented in Fig. 1. From here on the ANNMD can be considered as if it is a standalone application. Fig. 1 shows the basic construction of the ANNMD consisting of four sections, where each of them is represented by its own panel as follows:

- *Input / Output / Unseen Data*
- *Training Settings*
- *Network Settings*
- *Results of ANN(s) Trainings*

The titles of corresponding sections representatively describe their intention and each section that follows is presented in more detail. The sections in turn describe the real general usage of the ANNMD. It should be noted that an explanation of ANN basics is out of the scope of this booklet, so the specified variables are presented rather briefly. For further explanation on particular variables refer to literature on ANN basics. Some can be found here [1 - 20].

Input / Output / Unseen Data

The first section *Input / Output / Unseen Data* refers to data with which ANNs are going to be trained. At this stage the training data should be already preprocessed, i.e. elimination of outliers and careful selection of training data with respect to operational conditions of a system have to be done (an example showing the importance of careful selection of training data can be found here [21]). As the title of the panel indicates, there are three sets of data that can be provided, i.e. input data (button 'Inputs'), output data (button 'Outputs') and unseen data (button 'Unseen'). Needless to say, input and output data are mandatory for

ANN modelling, but, on the other hand, the unseen data can be used optionally if available. These data are used for examination of the generalization capability of ANNs as explained later on. All the data in the ANNMD have to be provided in either **.txt* or **.dat* format. The data in files should be arranged as column matrixes meaning that each column represents one parameter. It should be noted that the index column (which is commonly required in many ANN programs) has to be omitted. Only the training data are required! A file containing unseen data must include both input and output parameters stored in the same file, i.e. first columns of unseen input data, after which follow unseen output data. With respect to the availability of unseen data, the ANNMD is able to conduct trainings in three different ways denoted by three different options in the ANNMD. These are 'train + validate' and 'SA + validate' if unseen data are available, and 'w/o unseen' if not. How each option works can be found in Chapter 3, and recommendations on when each option should be applied are written in Chapter 6.

Training Settings

The second section refers to *Training Settings*. In this section there are eight variables that can be specified which are important in finding an optimal size and settings of a selected network. In turn variable names and features are:

- *RepW* refers to number of randomizations of initial weights. Every ANN network will be trained RepW-times in order to find the best weights contributing to optimal ANN accuracy.
- *Neurons* refers to tested range of neurons in the hidden layer. The variable must be specified as follows: *Neurons* = [*initial*, *step-size*, *end*]. The variable must be specified correctly, beginning with the *initial* number of neurons

which meets the *end* number of neurons with the correctly specified *step-size*, e.g. *Neurons* = [4, 2, 16] (an example of a wrong definition would be *Neurons* = [4, 3, 16]). If you want to train an ANN with only a certain number of neurons in the hidden layer then the *initial* and *end* values must be equal, e.g. *Neurons* = [**14**, 2, **14**].

- *Tset* refers to distribution of training data among training, cross-validation and test sets respectively. The variable is specified in ratios meaning that the sum of specified ratios must be 1.
- *RepR* refers to number of randomizations of training data. In other words, this also means that SA is performed *RepR*-times.
- *Epoch* refers to the number of epochs in ANN training.
- *ANNs in SA* stands for Artificial Neural Networks in Sensitivity Analysis. In general, an ANN in sensitivity analysis is defined by included inputs. Since it is much easier to specify eliminated input(s) than included ones in SA, the 'elimination' option has been applied in the ANNMD. So an ANN in sensitivity analysis has to be defined by the column number of eliminated input. Fig. 2 shows an example of how the variable *ANN in SA* works. The figure shows the following definition of *ANNs in SA* = [0] [1] [3] [1 3].

  Beginning with [0], it means that no parameters are eliminated (the ANN includes all specified input parameters). Such an ANN model with all included inputs is usually used to as the reference case for later comparison. There follows [1], meaning that the first parameter is eliminated, [3] refers to the third and [1 3] means that the first and the third parameters are eliminated at the same time. Note that multiple parameters can be eliminated in all combinations. It should also be emphasized that each ANN in SA must be specified within square brackets and that the specified value is not less than zero or higher than the number of input parameters.

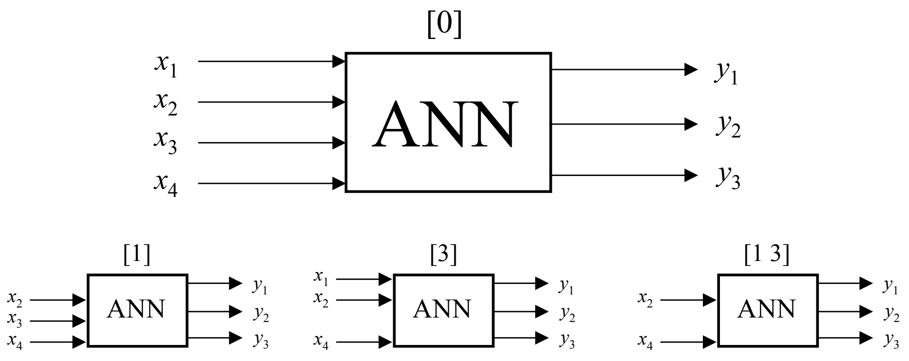


Figure 2: ANN input parameters under examination in SA.

- *OverT* refers to the prevention of overtraining (over-fitting) in ANN training procedure.
- *Norm.* implies normalization rates of input and output data before and after training. This value must be in the following limits [-1 1].

Furthermore, in the *Training Settings* section, two buttons can be noted, i.e. *Reset* and *Stop*. After training is completed, the *Reset* button can be used to bring the ANNMD program to its default settings. The *Stop* button can be used for termination of the training procedure (e.g. if training appears to be too time-consuming, new settings can be applied). When training is ongoing, this is the only button that you can press (all others are shaded, meaning that they are inactive).

Network Settings

The third section contains *Network Settings*. Four pop-up menus are provided. Menu *ANN Network* contains six ANN architectures, i.e. *newff, newcf, newfit*, *newelm*, *newlrn* and *newpr* that can be used for the training of an ANN. The first five networks are used for regression modelling while the last one, *newpr*, represents a classification (pattern recognition) option. For more information on each ANN network, refer to Matlab help on Neural Network toolbox [22].

The second pop-up menu, *Learning algor.*, in some literature also named as network *training function*, contains the back-propagation learning algorithms. The number of learning algorithms depends on the selected ANN Network; each contains several options. The third and fourth pop-up menus are related to transfer functions used in the hidden and output layers respectively. For both there are three options, i.e. *tansig*, *logsig* and *purelin*. More on each one can be found in [22].

Results of ANN(s) Trainings

The fourth section, *Results of ANN(s) Trainings*, provides the results of SA/trainings. The results are presented in graphical form. Two plots can be noted in Fig. 1; the bottom plot refers to training data and the upper plot to unseen data (the latter is optional, only used when unseen data are available). The plots of final ANNs in SA are also recorded and can be saved in **.fig*, **.emf* or **.bmp* format. Similarly detailed information on final *ANNs in SA* can also be saved. Both options are described in more detail in Chapter 5.

3. How does the ANNMD program work?

ANN modelling of energy systems is more or less a routine; it can thus be programmed and automated to some extent. Standard steps involve randomization of data, sensitivity analysis (SA), determination of optimal number of neurons in the hidden layer, randomization of weights, evaluation and presentation of results. These steps have been implemented in the ANNMD as independent parameters that can be adjusted as needed. These parameters can also be considered as basic parameters as they have to be examined in every ANN modelling. Since the ANN modelling is a so-called data-driven approach, the optimal set of basic parameters is literally found through trial-and-error procedure. Generally, it demands that a high number of parameter combinations are examined; so far this has not been in common practice as it is a very time-consuming procedure.

Some of those issues were taken into consideration when developing the ANNMD. Hence, the main task of the ANNMD is to automatically conduct the ANN training procedure in which the basic parameters are cross-checked in all combinations. Since parameters such as network type, transfer functions, etc. depend on the type of application, it is not reasonable to include them in the cross-checking procedure as well. However, the specific parameters in the ANNMD can be simply examined by the user considering the type of application. Different options in the ANNMD in conjunction with previous experience in ANN modelling can thus help to find an optimal ANN model of a system easily. The core of the ANNMD program can be represented by a basic scheme, as depicted in the following figure.

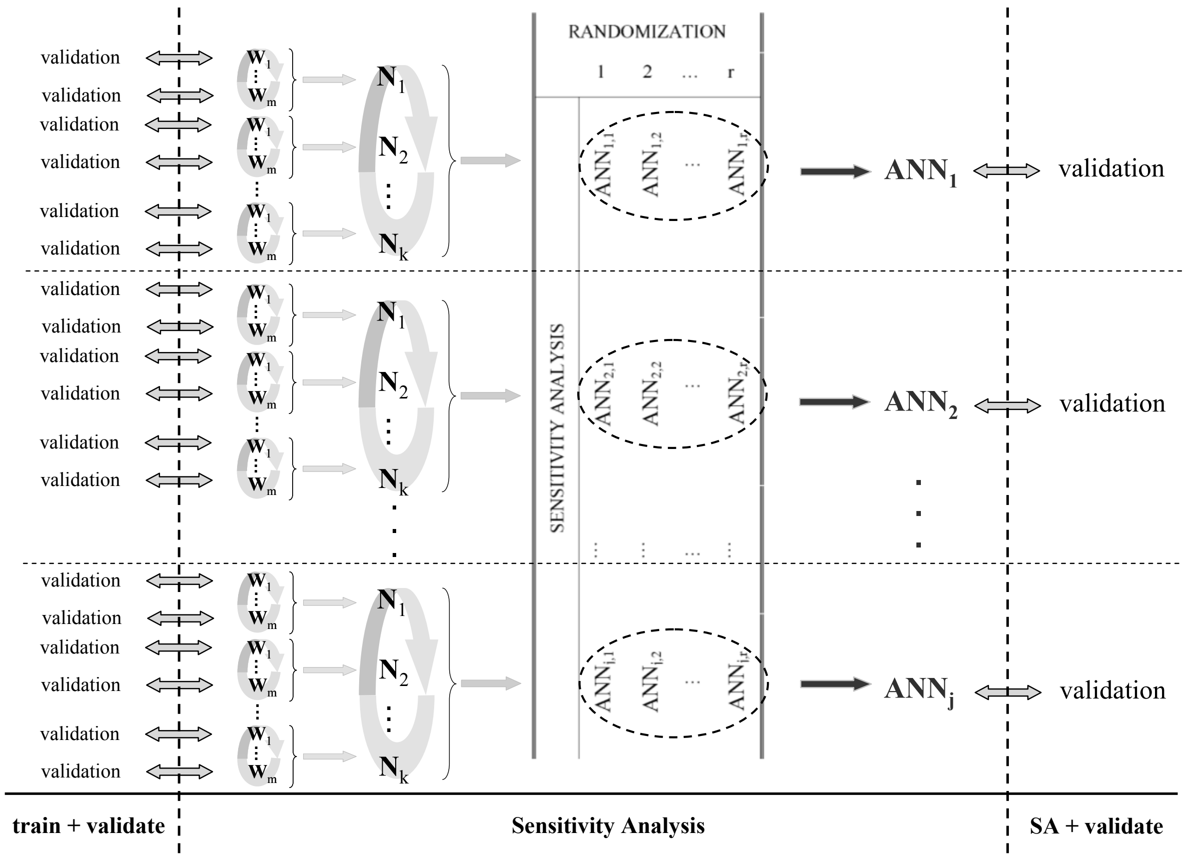


Figure 3: Basic scheme of the ANNMD program.

The ANNMD program involves examination of four parameters which generally yield an optimal ANN network structure. These parameters are:

- Initial weights (w-loops)
- Neuron number in the hidden layer (N-loops)
- ANN input parameters (sensitivity analysis)
- Randomization of data
- Generalization factor, which cannot be included as a parameter but is one of the most important issues in ANN modelling, has also been considered, as explained in the following.

How the ANNMD works will be explained from the root-level of the program, presented in Fig. 3. It can be seen that the ANNMD contains three options with respect to availability of unseen data. The program core, which is common to all options and is represented within the dashed lines, is explained on the basic option, i.e., when only training data are available and not also unseen data. Looking at the basic option, one can notice that the diagram is oriented from the left to right side of Fig. 3. The functionality of the ANNMD can be explained by beginning on the elementary level which is presented by the W-loops comprehending randomizations of initial weights. It follows the N-loops comprehending the tested range of neurons in the hidden layer.

We can begin by looking at the first ANN in SA (ANN_1) which, in the figure, is represented above the first dashed horizontal line. We shall assume that training data are randomized and the first number of neurons in the hidden layer is set. In the first step, the tested ANN model is trained *m*-times with respect to initial weights, i.e. $w_1 \ldots w_m$ (first W-loop), and the solution with the lowest mean relative error (MRE) is recorded for the first set of neurons in the hidden layer, i.e. N_1 (first N-loop). In the second step, the number of neurons in the hidden layer is increased to N_2 (still the first N-loop), the trainings with respect to weights are repeated (second W-loop: $w_1 \ldots w_m$) and the lowest MRE from the second step is recorded. This is done *k*-times, i.e. $N_1, \ldots, N_k$, covering the tested range of neurons in the hidden layer (entire first N-loop) in the procedure of testing the ANN_1 model. Since all MREs for each number of neurons are recorded, the ANNMD selects the optimal number of neurons in the hidden layer yielding the $ANN_{1,1}$ model (the first index refers to the number of ANN models in SA and the second to the data randomization number).

The same procedure is repeated with same randomized data for the second model in SA which yields ANN$_{2,1}$. This is the optimal second ANN model with respect to weights and neuron number in the hidden layer for the first randomized data in SA. This continues in the same way until the optimal weights and neuron number in the hidden layer are found for the last ANN$_{j,1}$ in SA. The second data randomization follows and all the previous steps in finding the optimal weight and neuron number in the hidden layer are repeated for all ANNs in SA. The result is the second column of ANNs in Fig. 3, i.e. ANN$_{1,2}$, ANN$_{2,2}$,…, ANN$_{j,2}$.

Repetition of data randomizations is continued until the specified number of data randomizations *r* is reached. At this stage all the trainings have been finished and there is ($r * j$)-number of ANNs. In the last step of the automated procedure, the selection of best ANNs in SA with respect to data randomization is conducted. This selection is also made on the basis of lowest MRE. The result is ANN$_1$, ANN$_2$,…, ANN$_j$ which represents the optimal solution with respect to weights, neuron number in the hidden layer and data randomization. The selection of the final ANN model with respect to different number of inputs is left to the user. In practice the final model is usually selected as a compromise between ANNs' accuracies and the number of inputs to the ANN; it can also depend on operators' demands for inclusion of a certain parameter. The final selection is rather easy; a schematic example of results from the ANNMD is shown in Fig. 4. There are three ANNs which predict three outputs, and the MREs of each output can be determined from the left axis. For each ANN the mean MRE is also presented, as shown in the figure.

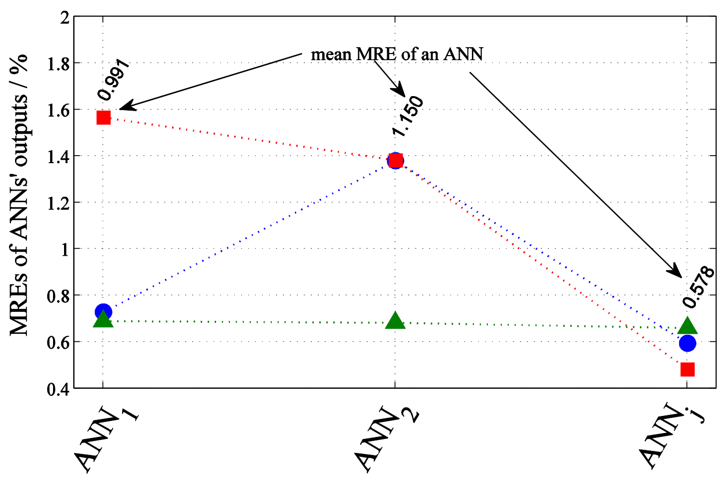


Figure 4: A schematic example of the presented results by the ANNMD. The example shows three ANNs with three predicted outputs.

As described above, the core of the ANNMD program involves four parameters that are automatically cross-checked and thus thoroughly examined. The program's structure can also be represented program-wise, as shown in the following figure.

```
For  $l_1 = 1:r$                      (randomization loop)
     For  $l_2 = 1:j$                            (SA loop)
          For  $l_3 = 1:k$                  (neurons loop)
               For  $l_4 = 1:m$             (weights loop)
```

Figure 5: Schematic presentation of the ANNMD program core.

The figure shows four loops, the structure of which enables cross-checking of weight space, neurons in the hidden layer, input parameters (SA) and data randomization. Such a program structure is also suitable for parallel computing employing multiple processors which is also implemented in the ANNMD.

On the other hand, if unseen data are available, the generalization aspect, as one of the most important issues in the ANN modelling, can be tested. Since the examination of the generalization capability of an ANN depends on the availability of unseen data and thus the generalization is not a parameter that can be set, it is explained in the next chapter covering the main options of the ANNMD.

4. Main options in the ANNMD

Input / Output / Unseen Data Section

In the preceding chapter, the functionality of the ANNMD is explained on the basic option, i.e. ‘w/o unseen’ data. However, in total there are three options for how the ANNMD can select the ANNs in SA. This is related to the availability of unseen data which can be specified in the Input / Output / Unseen Data section (see the basic layout of the ANNMD in Fig. 1). If there is no unseen data available the option ‘w/o unseen’ must be selected. In this case the basic procedure of ANN trainings is conducted as explained in the preceding chapter.

If the unseen data are available, it is possible to choose from two options on how to conduct the selection of ANNs in SA. If option ‘SA + validate’ is checked, the validation of final ANNs is conducted subsequently to the basic SA. This option includes the part denoted by the right vertical dashed line in Fig. 3. The second option with respect to availability of unseen data is the default one and it is conducted when ‘train + validate’ is selected. In this option the trained ANN is validated with unseen data after every single training. In this case the selection of ANNs on all levels, i.e. W-loops, N-loops and randomization of data, is conducted on the basis of lowest average MREs of unseen and training data. This option is denoted by the left vertical dashed line in Fig. 3. It can be noticed that the ‘left’ option includes as many validations as there are trainings of ANNs and the ‘right’ option as many validations as there are ANNs in SA. Since the 'left' option involves validation after every training and since the selection of the ANNs considers both training and unseen data, it can be expected that the generalization performance of the ANNs is increased. As previously mentioned, the generalization is one of the most important factors that needs to be examined before the application of an ANN in practice. Hence, the 'left' option is chosen as the default one. It should be mentioned that the decision on which option to select generally depends on the quality of available data and on knowledge/experience about the influential inputs, as explained in Chapter 6 where the recommendations on ANN training procedure with the ANNMD are given.

Training Settings section

The parameters of this section in the ANNMD were already explained briefly in Chapter 3. These are eight basic parameters which need to be set by ANN modelling and hence it is presumed that a reader is familiar with them. The default values of each parameter are set according to rather general experience. The ANNMD user should therefore reconsider modification of the basic parameters on the basis of, e.g. quality of data, number of ANN inputs and outputs, sampling frequency of data, etc. Some general tips on how to set these parameters in the ANNNMD can be found in Chapter 6.

Network Settings section

In this section the network structure and learning algorithm can be defined. There are six ANN networks available for training in the ANNMD, i.e. newff – feed-forward network, newcf – cascade-forward network, newfit – fitting network, newelm – Elman network, newlrn – layered-recurrent network and newpr – pattern recognition network. The first five networks are for regression applications while the latter one is for classification applications. Each of these networks can be trained with many different learning algorithms which can be selected accordingly. When choosing an ANN network, a default learning algorithm according to Matlab selection is set. For all ANN networks the available transfer functions in the hidden and/or output layers are tansig – tan-sigmoid, logsig – log-sigmoid and purelin - linear. For more details on each specific option in this section, i.e. network, learning algorithm, transfer function, refer to Matlab ANN toolbox help and to ANN literature [17, 23].

Results of ANN(s) Trainings section

This section contains presentation of results. There are two plots showing the Mean Relative Error (MRE) of ANNs' outputs. The MRE for each output is calculated as follows:

$$\mathrm{MRE} = \frac{1}{N}\sum_{i}\left|\frac{a_i - p_i}{a_i}\cdot 100\right| \quad / \quad \%$$

where a refers to measured (or reference) value, p to ANN prediction and N to number of (data) rows. Basically, the two plots can be explained in conjunction with the three options related to the availability of unseen data. The upper plot is used for presenting results with respect to unseen data and the bottom plot to training data. If the unseen data are not available, then only the bottom plot is used, showing the MREs of outputs calculated on the basis of training data. If the unseen data are available, then the bottom plot shows MREs based on training data (similarly as in the case without unseen data) and the upper plot shows the MREs of predictions of unseen data (this holds for both options with unseen data, i.e. ‘train + validate’ and ‘SA + validate’). In this way, the best generalization performance can be considered.

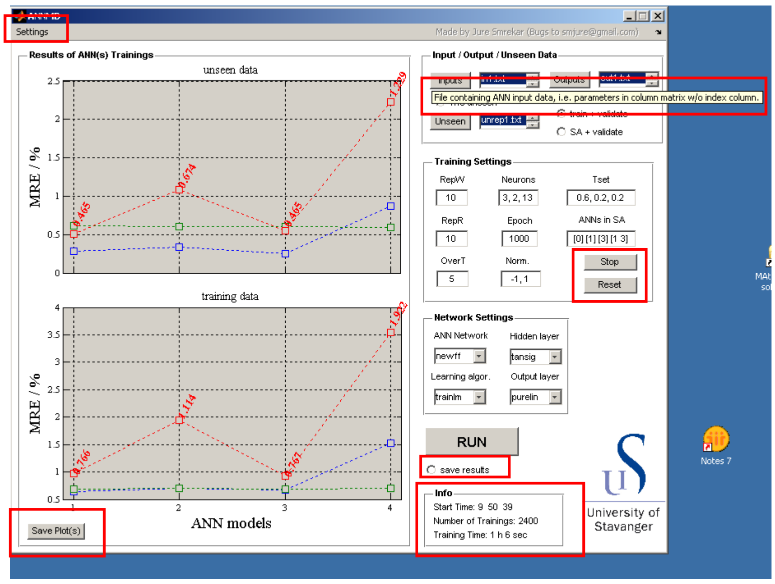
5. Presentation of results and other useful features in the ANNMD

Presentation of results

One of the main ideas behind development of the ANNMD was to make it as user-friendly as possible. Therefore, the Graphical User Interface (GUI) of the ANNMD is designed to be very simple and can be used for ANN modelling without any special knowledge when employing it. To make it easier to use, many practical features have been implemented and they are presented in the following.

We can start with interpretation of the results. Fig. 6 is an example of how the ANNMD looks, after the ANN trainings are completed. The selection of the final ANN model can be easily made by analyzing the training results represented by the two plots. Since the unseen data were available, both plots are occupied. Apparently, four ANN models with three outputs were under examination. Four ANN models are denoted with consecutive numbers, i.e. 1, 2, 3 and 4 (see *x*-axis of the bottom plot) corresponding to the ANN models, as defined in the field ‘ANNs in SA’, i.e. [0] [1] [3] [1 3]. The interpretation of the ANNs under examination in sensitivity analysis is as follows:

- [0] no input parameter was eliminated
- [1] first input parameter was eliminated
- [3] third input parameter was eliminated
- [1 3] first and third input parameters were eliminated at once



File containing ANN input data, i.e. parameters in column matrix w/o index column.

Unseen: unrep1.txt

RepW: 10
Neurons: 3, 2, 13
Tset: 0.6, 0.2, 0.2
RepR: 10
Epoch: 1000
ANNs in SA: [0] [1] [3] [1 3]
OverT: 5
Norm.: -1, 1

ANN Network: newff
Hidden layer: tansig
Learning algor.: trainlm
Output layer: purelin

Info
Start Time: 9 50 39
Number of Trainings: 2400
Training Time: 1 h 6 sec

Figure 6: An example of the ANNMD layout after a finished SA.

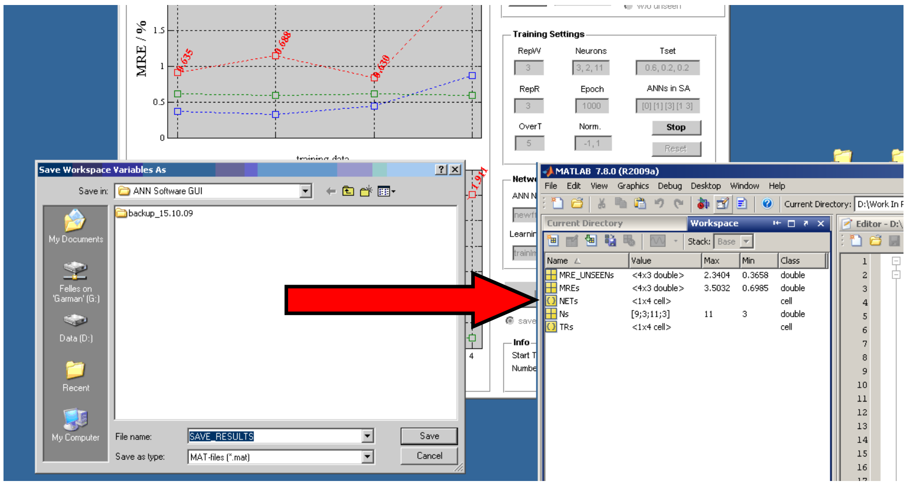
The outputs are evaluated by Mean Relative Error (MRE) and are plotted in different colours. The dashed lines connect the MREs of the same predicted outputs. This makes the comparison of single predicted outputs easy. The rotated red value above each ANN is the mean MRE value of the outputs, which is actually a mean MRE of an ANN model.

Useful features

The ANNMD comprehends many useful features that should be mentioned. Firstly, all the fields and buttons in the ANNMD are equipped with tips (drop-down boxes) explaining their meaning or how they should be defined. The tips appear when the mouse cursor is placed above the desired button, field, etc. An example is denoted by the red square in the upper right corner in Fig. 6. It represents a tip for button 'Inputs', and it tells a user what she/he should provide when pressing the 'Inputs' button: 'File containing ANN input data, i.e. parameters in column matrix w/o index column'.

The next useful feature of the ANNMD is the 'Settings' menu in the upper left corner. When pressing the menu, two options, i.e. 'Save Settings' and 'Load Settings', are available. A user can save the ANNMD settings that were set before or after the run and load them when necessary in the future. For instance, if we save the settings of the ANNMD in Fig. 6, the next time we load it exactly the same settings (also with resulting plots) will appear again. However, it should be mentioned here that after recalling the settings, the results of the trainings cannot be recalled as well. They have to be saved separately after a training, as explained in the following. The 'Settings' feature can be very useful after performing an extensive ANN modelling. Namely, by saving the final settings that gave optimal results, later retraining can be easily conducted by only specifying new training (and unseen) data and by pressing the 'Run' button.

The feature 'Save Results' in the red square under the 'Run' button is used for saving the ANNs and detailed ANNs information. Since in ANN modelling there are no certain rules, a user at first usually tries certain combinations of parameters in order to get some rough estimations. There can be quite a lot of preliminary tests before an extensive and/or final 'Run', therefore the 'Save Results' option is unchecked by default. It has to be emphasized, that if the 'Save Results' option is unchecked when pressing the 'Run' button, the results will not be saved after the trainings are completed! If it is checked, then a dialogue box asking to save the results in *.*mat* format will appear, as shown in Fig. 7.



Name	Value	Max	Min	Class
MRE_UNSEENs	<4x3 double>	2.3404	0.3658	double
MREs	<4x3 double>	3.5032	0.6985	double
NETs	<1x4 cell>			cell
Ns	[9;3;11;3]	11	3	double
TRs	<1x4 cell>			cell

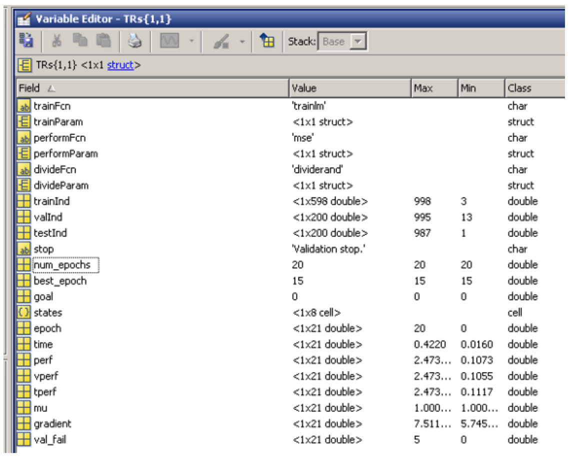
Figure 7: ‘Save Results’ option saves the results of modelling with the ANNMD as the Matlab work-space variables.

So what does the ‘Save Results’ option do? It saves five parameters of the best ANNs in the training procedure corresponding to the ANN MREs presented in the two plots. These results contain ANN networks and detailed training information that can be recalled later with Matlab and used for analysis, comparison, ANN network implementation, degradation analysis, etc. As presented in Fig. 7, the five parameters are:

- MRE_UNSEENs – MREs of outputs validated with unseen data. The values are presented in the upper plot. This variable is not present when the unseen data are not available.
- MREs – MREs of the outputs of the training data.
- NETs – In this variable all the networks of the final ANNs in SA are stored. These can be used for real application later on.

- Ns – Optimal number of neurons in the hidden layer of the best ANNs. Looking at the example in Fig. 7, one can observe that the values can vary dramatically.
- TRs – Contains all information related to ANN trainings of the best ANNs in SA.

Fig. 8 shows an example of the training record of an ANN model. Information such as error of cross-validation, error gradients, learning algorithm, performance function, etc. are stored and can be used for later detailed analysis. Further details on each piece of information can be found in the 'Help' section of the Matlab ANN toolbox [22].



TRs{1,1} <1x1 struct>

Field	Value	Max	Min	Class
trainFcn	'trainlm'			char
trainParam	<1x1 struct>			struct
performFcn	'mse'			char
performParam	<1x1 struct>			struct
divideFcn	'dividerand'			char
divideParam	<1x1 struct>			struct
trainInd	<1x598 double>	998	3	double
valInd	<1x200 double>	995	13	double
testInd	<1x200 double>	987	1	double
stop	'Validation stop.'			char
num_epochs	20	20	20	double
best_epoch	15	15	15	double
goal	0	0	0	double
states	<1x8 cell>			cell
epoch	<1x21 double>	20	0	double
time	<1x21 double>	0.4220	0.0160	double
perf	<1x21 double>	2.473...	0.1073	double
vperf	<1x21 double>	2.473...	0.1055	double
tperf	<1x21 double>	2.473...	0.1117	double
mu	<1x21 double>	1.000...	1.000...	double
gradient	<1x21 double>	7.511...	5.745...	double
val_fail	<1x21 double>	5	0	double

Figure 8: An example of the training record of an ANN model.

Another useful feature that can be seen at the bottom left of Fig. 6 is the 'Save Plot(s)' button. After the training procedure, the plotted results can be saved in *.*fig*, *.*emf* or *.*bmp* format. The *.*fig* format enables the plots to be adapted in Matlab into a suitable format required by journals, reports, etc. If unseen data are available, two dialogue boxes, one after the other, appear in order to save both plots; otherwise it appears as only one dialogue box.

In the section 'Training Settings' in Fig. 6, there are the 'Stop' and 'Reset' buttons, as already mentioned. The 'Stop' button is for termination of ANN trainings and it is the only button that can be pressed during 'Run' time. The 'Reset' button resets the ANNMD to its default setting. This can be useful when starting a new modelling session.

When the 'Run' button is pressed, an additional panel called 'Info' appears at the bottom of the ANNMD (see Fig. 6). This panel includes the starting time of the ANN trainings, the total number of trainings and the training time. From the example in Fig. 6, it can be seen that the training started at 9 hours, 50 minutes and 39 seconds; the number of ANN trainings were 2400 and the training time was one hour and six seconds. This information can be very useful for the reports as it gives the impression of how extensive the ANN modelling was. In this example, four ANNs were trained, meaning that 600 trainings per ANN model were conducted. It is quite a high number, providing information that each ANN was thoroughly examined with respect to initial weights, data randomization, neurons in the hidden layer and generalization aspect. Extensive examination enables us to draw reliable decisions and conclusions. (A word of caution: the number 600 trainings per model should not be taken for granted; it depends on many parameters. Also with today's computers the training times are getting shorter and even more thorough examination can be carried out without being very time-consuming.)

To make the ANNMD even more user-friendly, the warning and error signs were incorporated as, for example, presented in Fig. 9. The title of the message box, in this case: 'Argument Tset', provides information about the argument to which the error is referring, and the message tells about the error. In this case the definition of shares among the training data sets was done incorrectly, as can be also seen from the figure.

If an error appears, the ANN training is stopped and the corresponding field must be corrected. If a warning sign appears (sign with an exclamation mark), the ANN training is continued, but the message makes the user aware of possible 'danger'.

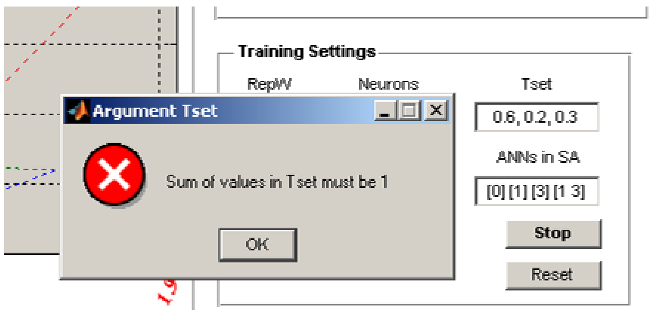

Training Settings
RepW Neurons Tset
0.6, 0.2, 0.3
ANNs in SA
[0] [1] [3] [1 3]
Argument Tset
Sum of values in Tset must be 1

Figure 9: An example of an error sign in the ANNMD.

Another feature of the ANNMD worth mentioning is the progress bar (Fig. 10) which provides information about the remaining time of ANN trainings. The progress bar presents two values, i.e. the remaining time and percentage of already completed trainings. It should be mentioned that the remaining time is an estimated time based on previous trainings. As the number of neurons in the hidden layer increases, the complexity of the ANN increases meaning that training time will be longer. Therefore, at the beginning of a 'Run' the estimated time is too optimistic, but it gradually reaches a reasonable estimation. Also the overtraining parameter in combination with initial weights influences the training time. On the other hand, the percentage in the upper left corner, to which the red bar corresponds, is an exact value representing already finished trainings.

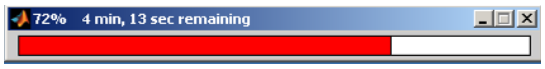

72% 4 min, 13 sec remaining

Figure 10: Progress bar in the ANNMD.

6. Recommendations on training procedure with the ANNMD

ANNs are so-called data-driven methods meaning that relations between inputs and outputs are found on the basis of statistical methods. There is no defined relationship as in mathematical models. This is one of the main reasons that ANN modelling has no certain rules which one can follow. Instead, experience is the best guide for acquiring a reliable ANN. However, with a good understanding of a system (or phenomena), an adequate number of parameters describing the system and quality (measured) data, good guidelines for ANN modelling of a specific system can be established. In this chapter some of the general experiences are given as recommendations on how to get representative results with the ANNMD. This will enable to get more reliable modelling results giving either reliable ANN models or, on the other hand, showing that a system cannot be modelled by applying ANN methods.

General aspects that are worth considering when conducting ANN modelling

Before starting to model with the ANNMD, a word on its outer frame has to be given in order to understand better how the ANNMD works. There are certain parameters that need to be thoroughly tested when modelling any system with ANNs. A classification can be made with respect to parameters and tasks, as shown in Fig. 11.

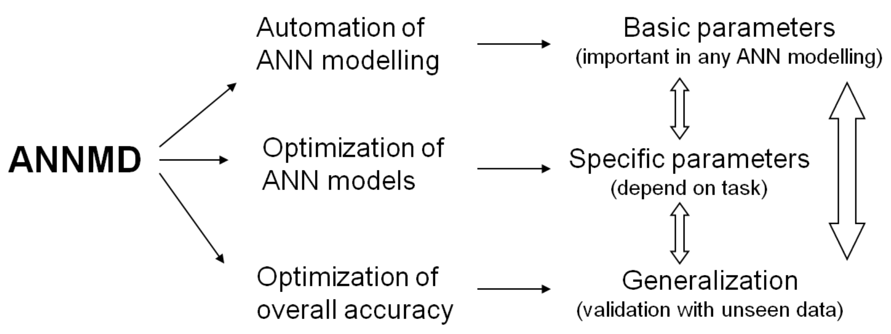

ANNMD
Automation of ANN modelling
Optimization of ANN models
Optimization of overall accuracy
Basic parameters
(important in any ANN modelling)
Specific parameters
(depend on task)
Generalization
(validation with unseen data)

Figure 11: Classification of tasks and parameters in the ANNMD.

The ANNMD was designed to cover three major tasks, i.e. *Automation of ANN modelling*, *Optimization of ANN models* and *Optimization of overall accuracy*. The first task is related to basic parameters, e.g. determination of the optimal number of neurons in the hidden layer, which is important in any ANN modelling and is thus automated (see Chapter 3). The second task, i.e. optimization of ANN models, is related to specific parameters, e.g. ANN network type, which depends on an application that needs to be modelled with ANNs. The intention of this optimization is that a user finds the best ANN network type, learning algorithm, normalization rates, transfer functions, etc. that are most suitable for modelling a specific application. The third task is related to optimization of overall accuracy when the ANN network type is already chosen, and it considers the generalization aspect. The generalization can be also influenced through parameters such as prevention of overtraining (*OverT*), but an additional feature was implemented into the ANNMD which takes into account both training and unseen data when selecting the best ANNs (see Chapter 4). As can be seen from Fig. 11, all the tasks are inter-related, meaning that the basic parameters are always cross-checked for any set of specific parameters and at the same time, if unseen data are available, the generalization aspect can be cross-checked as well.

So for the specific ANN network type, the ANNMD automatically examines the following parameters: (initial) weights, data randomization, ANN inputs (sensitivity analysis), number of neurons in the hidden layer and, if unseen data are available, it can also take into account the generalization aspect. In the following each of these factors, with its influence on the ANN's accuracy, is briefly presented:

- In the ANNMD, SA of input parameters can be performed (outputs are usually predetermined according to the operator's demands; besides, by switching inputs and outputs, an inverse influence can also be investigated). When deciding which inputs to include, the knowledge of a system's operation or physical phenomena is desired. In general, the higher the correlation between an input and output(s), the higher the need to include that parameter. One has to also consider that when a system has many operational conditions, the influence of an input on output(s) might vary. Therefore, it is recommended to include a parameter in SA, if it is considered that it may have some influence on the output(s). Proceeding on the safe side by including many inputs will take more computational time, but the goal of 'clearing things out' will be met. Needless to say, automated SA is now easy and fast with the ANNMD which is able to employ multiple processors.
- Initial weights are one of the most important boundary conditions in ANN trainings which may have a considerable influence on an ANN's overall accuracy. By thorough examination of weight space, one may prevent a training algorithm from getting 'stuck' in local instead of finding optimal minima of a multi-dimensional space presented by an ANN's inputs and outputs. The number of randomization of initial weights depends mainly on the dynamics of a process and the dimensionality of the ANN network (determined

by the number of inputs and outputs and the number of neurons in the hidden layer). The higher the dynamics and dimensionality, the higher the value should be set in the ANNMD.

- Data randomization is a very important parameter which has a strong influence on the generalization capability of an ANN. It influences correlations between inputs and outputs of training data, meaning that one data randomization may be better suited to one ANN model in SA than to another. Data randomization also means representation of a process. If a highly dynamic process has a low sampling frequency, then a certain randomized data set may lead to underrepresented data in training and cross-validation test sets. So generally, the higher the dynamics of a process and the lower the sampling frequency, the higher the value should be for data randomization in the ANNMD.
- The number of neurons in the hidden layer affects the order of the ANNs. The higher the number, the more complex a neural network will be. In literature one may find different suggestions on the initial value of neurons in the hidden layer. Proceeding on the safe side, the lowest starting number of the neurons in the hidden layer should be equal to the number of outputs.
- Generalization of an ANN is also one of the most important factors which is connected with many parameters such as data randomization, overfitting, etc. Of what use would an ANN model with a very good training accuracy be, if its application fails in real-life performance? Such ANNs are useless! The idea of generalization is nicely presented with an overfitted ANN, as shown in Fig. 12.

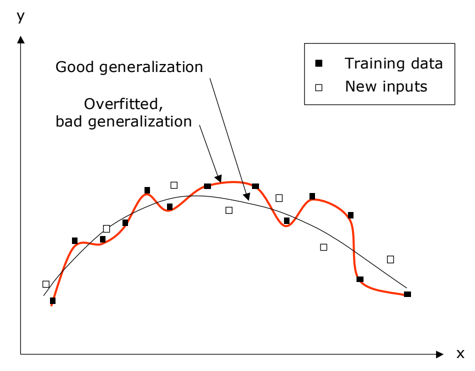


Figure 12: Overfitting of training data [24].

The figure shows that instead of having a good generalization capability, the ANN will simply memorize the training data. In this respect the ANNMD has one additional feature that may increase generalization of an ANN. In the ANNMD this is the option 'train + validate' with available unseen data (see Chapters 3 and 4). Firstly, the network is trained to its best accuracy (with respect to boundary conditions) and then validated with unseen data. This holds for each training in SA. This also means that both training data and unseen data are taken into account when selecting the best ANNs in SA, i.e. ANNs with maximum overall accuracy.

How to start modelling with the ANNMD

At this stage it is presumed that a user is already familiar with ANN basics and that the training data are preprocessed. The understanding of parameters in the ANNMD will help a user to get the optimal ANN models and to correctly interpret the results. By preprocessed data, a user ensures that outliers are eliminated; otherwise an ANN will try to fit them as well, resulting in deteriorated ANN accuracy. So here are some thoughts on how to use the ANNMD, starting from the layout presented in Fig. 1.

Principally, if a user plans to do a regression ANN modelling and has provided input, output and unseen data, she/he can already press the RUN button and SA will begin. It is that simple! However, in order to reduce overall modelling time and to get the best ANN accuracy, it is good to consider how to proceed from the beginning. Let's start with the case when unseen data are not available and we will return later to when they are. In the case of the option 'w/o unseen' a user can focus on the definition of parameters only in the *Training Settings* and *Network Settings* sections. In the ANNMD the default settings are already the most common ones. For instance, if you do regression modelling, then the *Network Settings* section can at first be left as it is, i.e. ANN network *newff* and learning algorithm *trainlm* are the most common ones with proven good accuracy in many applications. So, at first we can only check if the default parameters in *Training Settings* 'suit' our system (or phenomena) that we are modelling. These parameters are set by default as follows:

- The number of data randomizations (*RepR*) is set to six. This value is already quite high meaning that if the system is well represented with its data and if the system dynamics are not high, the value can be left as it is. In the opposite case, when a system is not well represented by its data and/or the system's dynamics are high, then the value *RepR* can be increased. It should be emphasized that a modelled system should be represented by (measured) data as much as possible. In general, today's systems are usually equipped with rather good acquisition systems; so, having this in mind, the value *RepR* can be left as it is at the beginning.

- The default number of randomization of initial weights (*RepW*) is set also to six. Similar logic as with *RepR* applies here for *RepW*. At the beginning, the value *RepW* can also be left as it is.
- The *OverT* parameter is set to five, meaning that when the error of cross-validation set will start to increase and will appear five times higher with respect to the error of training set, the ANN training will be terminated. As mentioned above, this parameter influences the generalization capability. If the *OverT* is set too high, it may result in an ANN that is overtrained. Therefore, at the beginning of modelling this parameter can be left and a user can return to it when the final optimization of the ANN model is conducted.
- The range of tested neurons in the hidden layer is defined under parameter *Neurons* with default range 4, 2, 18. It means that the ANN will be tested from four to 18 neurons in the hidden layer with step size two. As stated above, the starting number can be equal to the number of outputs. How high we should set the upper limit depends on a system and the quality of data. For the beginning, the initial number of neurons should be checked and set equal to the number of outputs.
- The number of epochs is set to 1000. This number is rather high and it is usually not reached because of termination due to the prevention of overtraining (parameter *OverT*). One can check with parameter *num_epochs* (for example see Fig. 8) at

which a training is terminated. If the set value of epochs has been reached, then it should be increased.

- The *Tset* parameter represents the distribution of training data among training, cross-validation and test set. Its default values are [0.6, 0.2, 0.2]. These values are set as default values in many programs. Changing them is not common as it has been shown many times that the ratios between sets are close to optimal. However, there may be exceptions which are out of the scope of this document.
- The normalization rate is set to [-1, 1]. Changing these values to a narrower interval is usually the case when we want an ANN with extrapolation capabilities. It is desired that data for ANN modelling covers the whole operational range of a system, so the extrapolation is not necessary. In such cases the values can be left as they are.
- The field *ANNs in SA* includes the input parameters in sensitivity analysis. A user has to specify them accordingly to see which parameters are influential. Starting on the safe side, a user can include all available parameters, which influence the outputs to a reasonable extent, and then eliminate one parameter at a time. Influential parameters can be found through principal component analysis or by calculating the matrix of correlation coefficients between inputs and outputs. However, such methods do not provide deeper insight into dependences between inputs and outputs throughout the system's operation as the correlation may vary with respect to particular operational condition significantly. In such cases more advanced techniques of training data selections must be applied.

From the above description we can see that at the beginning only two parameters, i.e. *Neurons* and *ANNs in SA*, need to be adjusted before the first RUN. When we get the results of the first RUN, the next thing we can do is to eliminate the parameters that do not influence the ANN predictions and/or affect the predictions negatively. If necessary, we can perform a second RUN similarly as the first one, but with a reduced number of inputs. When we find the most influential input parameters we can test other training parameters that we left untouched before. Up to this point all ANNs in SA are tested under the same conditions with the intention that the best ANN(s) with the most influential parameters has been found. Notation ANN(s) is used since more ANNs can be included in the following stages of SA if we are still not completely sure - due to, e.g. similar error level - which is the optimal ANN. In the next stage, we try to optimize the ANN network in order to get as high accuracy as possible. For instance, if the measuring frequency was considered low, we can increase parameter *RepR* to see whether more representative data sets can be found. We can also vary parameter *OverT* to see if the models' accuracy can further be improved, etc. At this point the parameters in the *Network Settings* section can also be varied in order to examine their influence. We can play in this way until we have found the best setting that gives the best ANN model of a system. To demonstrate the simplicity of using the ANNMD, a comparison study with commercial software for ANN modelling was conducted and can be found here [9].

Modelling with available unseen data

Now we can look into the case when unseen data are available. Basically the procedure is rather similar to the one without unseen data, but the overall accuracy of ANNs can be significantly improved. Validation of models with unseen data has usually been the last and the most important stage in ANN modelling as it confirms the reliability of a model for use in a real-life application. The usual approach was that firstly 'the best' ANN model had been found on the basis of training data only and then it was validated with unseen data.

This means that validation was performed only on a few final models. Besides this conventional option 'SA + validate', the ANNMD also contains the option 'train + validate' in which after every training of the ANN, its validation with unseen data follows and the best ANNs are selected on the basis of the MREs of training as well as unseen data sets. Hence, as the unseen data are not presented to the ANN network during the training procedure, a higher generalization capability can be expected.

So the question is which option to choose, i.e. 'train + validate' or 'SA + validate', at the beginning of ANN modelling when the unseen data are available. It mainly depends on the quality of data and the previous experiences in ANN modelling of a particular system. Again to proceed on the safe side, starting with the option 'SA + validate' can be considered as more representative. This is because at the beginning there are usually many input parameters involved, including those whose influence we are not sure of but which we include anyway. Many of these parameters do not improve the accuracy of the ANNs. But since they participate in training the ANNs (and are thus taken into account), they contribute to the 'inaccuracy' of other inputs that are influential. So many input parameters also mean many combinations between inputs and outputs which inevitably influence the generalization capability of such ANNs. Hence, the option 'train + validate' may result in an unrepresentative comparison of ANNs when looking at the MREs of training data and unseen data. So the first stage can be accomplished with the 'SA + validate' option. When the final ANN(s) are acquired, then it is recommended to also try the option 'train + validate'. In this way the generalization aspect is thoroughly examined.

Remark

The ANN network is selected according to the specific engineering problem. Since there is a pattern recognition type network available in the ANNMD, a word about it should be given also. With this option, tasks to classify or describe classes can be handled. The classification or description scheme is usually based on the availability of sets of patterns that have already been classified or have distinct features. One of the practical applications that has been tested to some extent is a sensor validation study with pattern recognition network [25]. However, it has to be mentioned that the ANNMD does not include an option with three hidden layers which is common in Auto-Associative Neural Networks (AANN) tasks for sensor validation.

7. Summary

This booklet presents a tool for ANN modelling called the Artificial Neural Network Model Developer (ANNMD), its features and recommendations on the modelling procedure. The ANNMD is a generic tool for ANN modelling of arbitrary systems. Its main features are a user-friendly interface and an automated ANN modelling procedure that cross-checks four basic parameters, i.e. (initial) weights, data randomization, number of neurons in the hidden layer and ANN inputs (sensitivity analysis) that are important in every ANN modelling. Cross-checking along the sensitivity analysis (SA) enables a user to easily find an optimal ANN network with respect to the basic parameters. Besides these, the ANNMD also contains specific parameters, e.g. ANN network type and learning algorithm, in order to further optimize the best ANNs in SA. Among the six ANN network types, five are for regression tasks and one for pattern recognition. Furthermore, the ANNMD can also examine the generalization aspect of the ANNs as one of the most important factors that needs to be examined before application of ANN models in real practice.

Last but not least, there are three options available with respect to the availability of unseen data. Two options refer to when unseen data are available. The first one covers the conventional way of conducting the SA, i.e. after SA, final ANNs are validated. The second option represents an innovation that considers the generalization aspect of ANNs, i.e. all of the ANNs in SA are validated with unseen data after every training. In this way the generalization capability of an ANN can be increased and thoroughly examined.

8. References

1 S. A. Kalogirou, ―Applications of artificial neural-networks for energy systems,‖ Appl. Energy., vol. 67, pp. 17–35, 2000.

2 U. Kesgin, H. Heperkan, ―Simulation of thermodynamic systems using soft computing techniques, Int. J. Energy. Res., vol. 29, pp.581–611, 2005.

3 C. Boccaletti, G. Cerri, B. Seyedan, ―A neural network simulator of a gas turbine with a waste heat recovery section, Trans. ASME: J. Eng. Gas Turbines Power, vol. 123, pp. 371–6, 2001.

4 J. Smrekar, M. Assadi, M. Fast, I. Kustrin, S.De, ―Development of artificial neural network model for a coal-fired boiler using real plant data, Energy, vol. 34, pp. 144–152, 2009.

5 J. Smrekar, D. Pandit, M. Fast, M. Assadi, S. De, Prediction of power output of a coal-fired power plant by artificial neural network, Neural Comput. & Applic., DOI 10.1007/s00521-009-0331-6, December 2009.

6 M. Fast, M. Assadi, J. Smrekar, ―Application of artificial neural network to the condition monitoring and diagnosis of a CHP plant, International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems, Cracow-Gliwice, Poland, June 2008.

7 J. Arriagada, M. Costantini, P. Olausson, M. Assadi, T. Torisson, ―Artificial neural network model for a biomass-fuelled boiler, Proc. of the ASME Turbo Expo, Georgia USA, 2003.

8 Wu, Xiao-Juan, Zhu, Xin-Jian, Cao, Guang-Yi, Tu, Heng-Yong, Nonlinear modelling of a SOFC stack by improved neural networks identification, Journal of Zhejiang University: Science A, 8(9), p 1505-09, August 2007.

9 J. Smrekar, M. Assadi, Comparison Study of an In-house Developed Tool and Commercial Software for ANN Modeling, IEEE The 17th International Conference on Industrial Engineering and Engineering Management, October 2010, Xiamen, China.

10 http://www.neurosolutions.com/

11 http://grey.colorado.edu/emergent/index.php/Main_Page

12 http://www.ra.cs.uni-tuebingen.de/software/JavaNNS/manual/JavaNNS-manual-4.html

13 http://www.dicis.ugto.mx/profesores/sledesma/documentos/index.htm

14 http://www.mathworks.com/products/neuralnet/

15 http://en.wikipedia.org/wiki/Multilayer_perceptron

16 S. Haykin, Neural Networks: A Comprehensive Foundation (2nd Ed), Prentice Hall, 1999.

17 C.M. Bishop, Neural Network for Pattern Recognition, Oxford University Press, 1995.

18 J.C Principe, N.R. Euliano, W.C. Lefebuer, Neural and Adaptive Systems, John Wiley and Sons Inc, 2000.

19 T. Palmé, Gas Turbine Modelling for Degradation Tracking and Monitoring with Artificial Neural Networks, Lund University, 2008.

20 M. Fast, Artificial Neural Networks for Gas Turbine Modeling and Sensor Validation, Lund University, 2005.

21 J. Smrekar, M. Assadi, M. Fast, I. Kustrin, S. De, Development of artificial neural network model for a coal-fired boiler using real plant data, Energy, 34, pp. 144–152, 2009.
22 http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/index.html?/access/helpdesk/help/toolbox/nnet/&http://www.mathworks.com/access/helpdesk/help/helpdesk.html
23 C.M. Bishop, Pattern Recognition and Machine Learning, Springer 2006.
24 J. Arriagada, Introduction of Intelligent Tools for Gas Turbine Based, Small-Scale Cogeneration. Licentiate Thesis, Department of Heat and Power Engineering, Lund Institute of Technology, Lund University, Sweden, 2001.
25 T. Palmé, J. Smrekar, M. Fast, M. Assadi, J. Oman, FPSV – Fult Pattern Sensor validation, application of different ANN model structures for sensor validation, The Future of Gas Turbine Technology 4th International Conference, Belgium, October 2008.