ISRN LUTMDN/TMHP—06/5085—SE
ISSN 0282-1990

Artificial Neural Networks for Gas Turbine Modeling and Sensor Validation

Magnus Fast

Thesis for the Degree of Master of Science

Division of Thermal Power Engineering
Department of Energy Sciences
LUND UNIVERSITY
Faculty of Engineering LTH
P.O. Box 118, S – 221 00 Lund
Sweden

Artificial Neural Networks for Gas Turbine Modeling and Sensor Validation

Magnus Fast

December 2005
Master Thesis
Division of Thermal Power Engineering
Department of Energy Sciences
Lund University, Sweden
http://www.vok.lth.se

ISSN 0282-1990
ISRN LUTMDN/TMHP—06/5085—SE
Printed in Sweden
Lund 2006

Preface

This master thesis has been conducted at the division of Thermal Power Engineering, department of Energy Sciences, Lund University, Sweden. The work has come about in cooperation with Lunds Energi AB, during the fall of 2005.

This master thesis has been very educational in terms of implementing my knowledge in the field of gas turbine theory and combined heat- and power plants, as well as providing me with new knowledge about artificial neural networks and software accompanied with it.

I want to acknowledge my supervisors Mohsen Assadi and Jaime Arriagada for their support and profound knowledge in the fields described above. At Lunds Energi AB I would like to acknowledge Stefan Nilsson and Vinko Culjak for answering all my questions and providing me with necessary information to complete this master thesis.

I would also like to thank Mehrzad Kaiadi for our long discussions about artificial neural networks and the many conclusions we have drawn from them.

Finally I would like to thank my better half Alisa Selimovic for always supporting me no matter what I am doing.

Lund 2005-12-23

Magnus Fast

Abstract

The aim of this collaboration, between the division of Thermal Power Engineering and Lunds Energi AB, is to investigate the possibilities of training artificial neural networks (ANNs) with power plant operational data. For this purpose operational data from Lunds Energi's gas turbine GT10B with heat recovery unit (HRU) will be used. Furthermore a model with user interface is created to demonstrate the possibilities of using ANN. The results are evaluated through feedback from Lunds Energi and many different areas of implementation are considered.

ANN differs from conventional mathematical models in the sense that they are trained rather than programmed. During training, data is presented in an iterative manner in order to find the relation between selected inputs and outputs. After the network is trained the weights, i.e. the parameters containing the network information, are locked and if the network is presented with new, before unseen, data it is able to predict new outputs.

The software used for modeling ANNs is called NeuroSolutions. The resulting networks have been processed in Visual Basic for final use in Excel.

Thru these studies several ANN models have been produced, both models of the gas turbine (GT) and models for sensor validation (SV). The results have been promising, e.g. with networks demonstrating high performance predictability.

Contents

List of Figures

List of Tables

Nomenclature

d	Desired output or target	-
e	Error	-
E	Error function	-
F	Transfer function	-
H	Number of hidden neurons	-
m	Mass flow rate	kg/s
M	Number of input nodes	-
n	Iteration number	-
N	Number of output nodes	-
p	Pressure	bar
P	Power	W
Q	Heat	W
s	Effective input to transfer function	-
T	Temperature	°C
w	Weight	-
w	Weight vector	-
x	Input signal to neural network	-
x	Input vector	-
y	Output signal from neural network	-

GREEK SYMBOLS

α	Alpha value	-
δ	Delta term	-
η	Efficiency	%
η	Learning rate	-
Σ	Summation function	-

SUBSCRIPTS

amb	Ambient
dh	District heating
cout	Compressor outlet
e	Electrical
el	Electrical
i	Arbitrary node in the input layer
j	Arbitrary neuron in the hidden layer
k	Arbitrary neuron in the output layer
3	Compressor outlet

7	Turbine outlet
0	Bias

SUPERSCRIPTS

T	Transposed (vector)

ABBREVIATIONS

ABB	Asea Brown Boveri
ADALINE	ADAptive LINear Element
ANN	Artificial Neural Network
BP	Back Propagation
CL	Confidence Level
CV	Cross-Validation
GT	Gas Turbine
HRU	Heat Recovery Unit
LMS	Least Mean Square
MLP	Multi Layer Perceptron
NG	Natural Gas
NS	NeuroSolutions
RH	Relative Humidity
SV	Sensor Validation
TIT	Turbine Inlet Temperature
UHC	Unburned HydroCarbons
VB	Visual Basic

1 Introduction

1.1 Background

The department of Energy Sciences at Lund University is the first in Sweden to evaluate ANNs potential within power plant applications.

This master thesis is based on previous work presented in a licentiate thesis and a doctoral thesis by PhD Jaime Arriagada [2,3]. Several models, e.g. fuel cell models, have been studied before and modeling a gas turbine with power plant operational data is a step in this development.

By creating a useful product, or a base for a product, for Lunds Energi AB, power companies might realize the potential of using ANN models and thereby make future investments in such projects.

ANN is used when modeling the gas turbine, which could be modeled with e.g. the heat- and mass balance program IPSEPro. The reason for using ANN instead of other programs is because once the software (NeuroSolutions) is mastered the models are produced very fast. The requirement is that operational data is available. The final product can be utilized by anyone, in contrary to IPSEPro models.

Lunds Energi are the providers of operational data, generated by their GT10B gas turbine and heat recovery unit. As they entered the project they had no knowledge about ANN, so the goal was ultimately to make them realize the potential. This was accomplished by producing a model of the gas turbine complete with user interface. This model is very easy to use and has in return opened their eyes for the possibilities of using ANN models in their daily work.

1.2 Objectives

- Understanding ANN and implementing that knowledge to create ANN models in NeuroSolutions.
- Investigate the possibilities of creating ANN models with operational data from the power plant.
- Recognize potential fields where Lunds Energi can benefit from using ANN models.
- Produce an interesting, useful and easy to use product.
- Presenting a base for future work

1.3 Limitations

When describing ANN multi-layer perceptrons (MLPs) are in focus and some networks, e.g. self-organizing, are completely left out of the discussion.

This master thesis will primarily focus on the gas turbine and models based on gas turbine operational data.

No effort has been done making the models universally functional, i.e. they might not be applicable on other gas turbine configurations.

1.4 Methodology

Literature studies and ANN simulations based on operational data is the foundation of this master thesis.

The simulation tool used is the commercial software program NeuroSolutions.

Constant contact with my supervisors and personnel at Lunds Energi has guided me through the work.

1.5 Outline of the thesis

Chapter 2 explains ANN fundamentals. The first section is recommended to everyone, while sections 2.1 – 2.6 offer a more detailed description of ANN.

Chapter 3 briefly describes the software, NeuroSolutions, used for modeling the ANNs.

Chapter 4 gives a description of the gas turbine and heat recovery unit at Lunds Energi AB.

Chapter 5 contains the case study with results from the ANN models studied. Section 5.2 contains results from the GT models and section 5.3 results from the SV models.

Chapter 6 presents conclusions and future work.

2 Artificial Neural Networks

Artificial Neural Networks consist of a number of processing units, or artificial neurons, communicating with each other by sending signals thru weighted connections. In contrary to traditional mathematical models, which are programmed, ANNs learn the relations between selected inputs and outputs.

An ANN does not care what the actual relations between selected input and output parameters are but will instead find its own connections. The easiest way to look at an ANN is as a box that, when presented with input signals, generates output signals (see figure 2.1). An effort to explain what happens inside of this box is made in this chapter.

ANN is a sophisticated tool to use when the relations between input and output are complicated or even unknown.

For demonstrative purpose, a simple arbitrary equation (see equation 2.0) will act as the relation between input signals and output signals, where x represents the input and y the output. If input signals are presented to the equation an equal amount of output signals will be generated (see table 2.0).

$$y = \frac{x^2}{3} \tag{2.0}$$

TABLE 2.0 INPUT AND OUTPUT SIGNALS.

x (input)	y (output)
3	3
6	12
9	27
12	48
15	75
18	108
21	147

In this case the equation is known and it is very easy to produce an output signal with a specific input signal. Lets imagine that the equation does not exist but the input and output signals do and it is still desirable to be able to produce output signals for new input signals. This is where ANN comes in handy with its ability to learn the relations between input and output signals, no matter how complicated they are. This is accomplished trough a process called learning, where the signals are presented to the ANN in an iterative manner. After the learning process is completed the network is locked and the information about the relationship between output and input is thereby stored. When a new input signal is presented to the network, lets say 10, the network will produce the output signal 33.33, or at least very close to (it depends on the success of the learning process). The accuracy of the ANN produced

outputs will be best in the interval 3-21, i.e. the interval used in the learning process. It is possible to train ANNs in a way that enables some extrapolation capability.

ANNs are not restricted to having only one input parameter and one output parameter but are also appropriate for modeling non-linear multi-dimensional systems, i.e. multiple inputs and outputs with non-linear relations (see figure 2.1).

The overall advantage, compared to traditional mathematical models, is that very complex systems can be modeled relatively fast, with the requirement that system data is available.

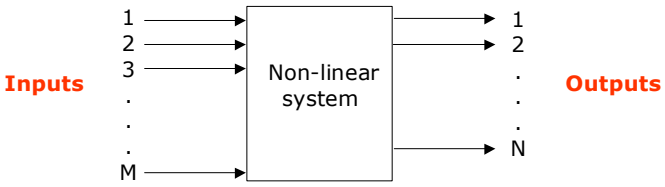

FIGURE 2.1 NON-LINEAR MULTI-DIMENSIONAL SYSTEM [2].

In this chapter, sections 2.1 – 2.4 provide the necessary background information for understanding section 2.5. This section describes the multi-layer feed-forward network, which is the network type used in chapter 5, case study, when modeling the gas turbine.

2.1 Network topology

ANNs are divided into two main groups depending on the topology of the connections, feed-forward and recurrent (feed-back) networks (see figure 2.2).

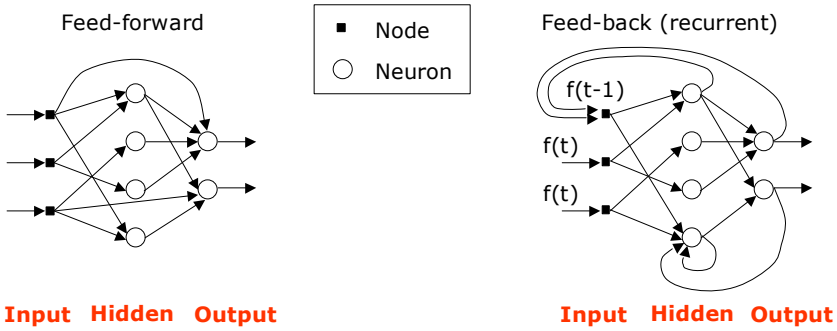

FIGURE 2.2 MAIN ANN ARCHITECTURES [2].

In this thesis, focus will be on feed-forward networks and as the name suggests signals are only allowed to pass in one direction, i.e. from input to output. Furthermore this ANN consists of different layers, one input layer, one or more hidden layers and one output layer. The communication between the environment and ANN consists of the input layer receiving data and the output layer delivering the computed results. The hidden layer communicates only within the network. If all units in one layer are connected to all units in the next layer it is called fully connected. No processing is done in the input layer and its components are called input nodes. Both the hidden layer and output layer are made up from artificial neurons, i.e. processing units.

The main objective for an artificial neuron is to receive weighted inputs from other neurons or nodes, compute an output and propagate this to other neurons. Another task is participating in the updating process of the connection weights during the learning process.

2.2 Basics of the learning process

Learning involves modifying the strength of connections between the elements. The learning process is called training and is performed according to learning rules, which are categorized in the two following groups:

- Supervised learning in which both inputs and desired outputs are known, this means the network can measure its predictive performance for given inputs.

- Unsupervised learning where only the inputs are known and the neurons must find a way to organize themselves without help from the outside.

After the training process is completed the weights are fixed. If new, previously unknown, inputs are presented to the network it will predict matching outputs. This ability is called generalization.

2.3 The artificial neuron

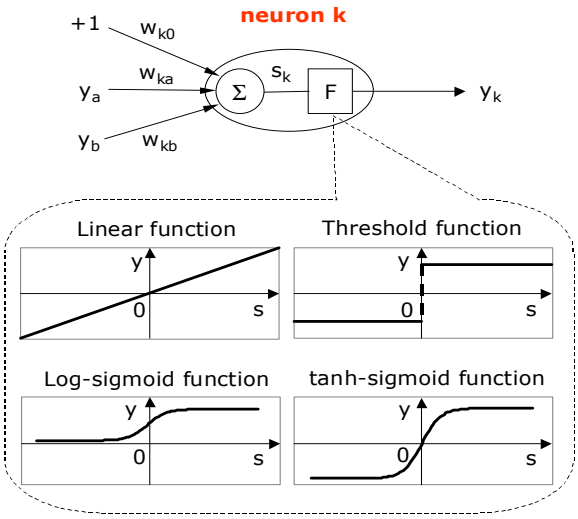

FIGURE 2.3 SCHEMATIC REPRESENTATION OF AN ARTIFICIAL NEURON AND COMMON TRANSFER FUNCTIONS [2].

A specific neuron, k, joins the incoming signals, from other nodes or neurons, by a propagation function, normally standard summation. The resulting signal is called effective input, s_k, and is passed thru a transfer function, F (different transfer functions are described in section 2.3.1). The output signal from the neuron has the designation y_k. The incoming signals have all been weighted, which means that they are multiplied with respective weights (w_{ka}, w_{kb}) equivalent to their significance for neuron k. This means that the signals are either inhibited or exhibited. In addition to the input from nodes and neurons there is an extra input to every neuron, called the bias. The bias has the value +1 and the corresponding weight is w_{k0}. The purpose is to present an offset to the transfer function, which allows the neuron to have an output even if the input is equal to zero (see figure 2.5). The first letter in the indexes represents the destination neuron and the second letter represents the origin neuron. Equation 2.1 describes the process mathematically.

$$\begin{cases} s_k = \sum_{j=a}^{b} (y_j w_{kj}) + w_{k0} \\ y_k = F(s_k) \end{cases} \tag{2.1}$$

2.3.1 Transfer functions

Different transfer functions are suited for different networks and applications, described below are four of the most commonly used:

The linear function (c = constant):

$$y = c \cdot s \tag{2.2}$$

The threshold function:

$$y = \begin{cases} +1, \text{if } s \geq 0 \\ -1, \text{otherwise} \end{cases} \tag{2.3}$$

The logistic sigmoid function:

$$y = \frac{1}{1+e^{-s}} \quad [0:1] \tag{2.4}$$

The tanh sigmoid function:

$$y = \frac{e^{s} - e^{-s}}{e^{s} + e^{-s}} \quad [-1:1] \tag{2.5}$$

2.4 Single-layer feed-forward networks

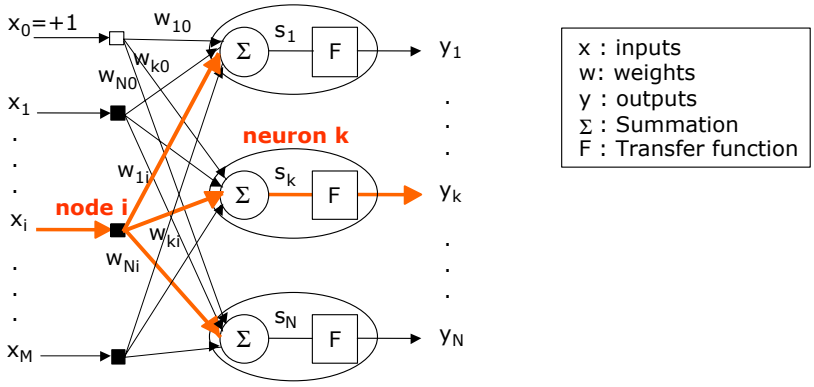

FIGURE 2.4 GENERAL STRUCTURE OF A SINGLE-LAYER FEED-FORWARD NETWORK. [2].

The definition of single-layer network is that there is only one layer, i.e. output layer, with artificial neurons. They are used for classification tasks and for function approximation tasks [7]. Single-layer networks are accompanied by some disadvantages such as only being able of creating linear classifiers, demonstrated in figure 2.7 (page 13), or representing linear functions. One advantage towards multi-layer networks is that, because of their linearity, they always converge to one optimal solution.

2.4.1 Networks with threshold activation function

The simplest network has only one output neuron and two input nodes. The output from the network is either +1, if the effective input is positive, or -1 if the effective input is negative (see equation 2.3). The network can be used for classification tasks,

i.e. deciding whether an input pattern belongs to one of two classes. The output is given by following equation:

$$y = F\left(\sum_{i=1}^{2} w_i x_i + w_0\right) \tag{2.6}$$

The single layer network represents a linear discriminant function (equation 2.7) that acts as a decision border, separating the two classes.

$$\begin{aligned} & w_1 x_1 + w_2 x_2 + w_0 = 0 \\ \Leftrightarrow\ & x_2 = -\frac{w_1}{w_2} x_1 - \frac{w_0}{w_2} \end{aligned} \tag{2.7}$$

Figure 2.5 illustrates this function, and how the weights determine the slope of the line and the bias determines how far the line is from its origin, i.e. offset.

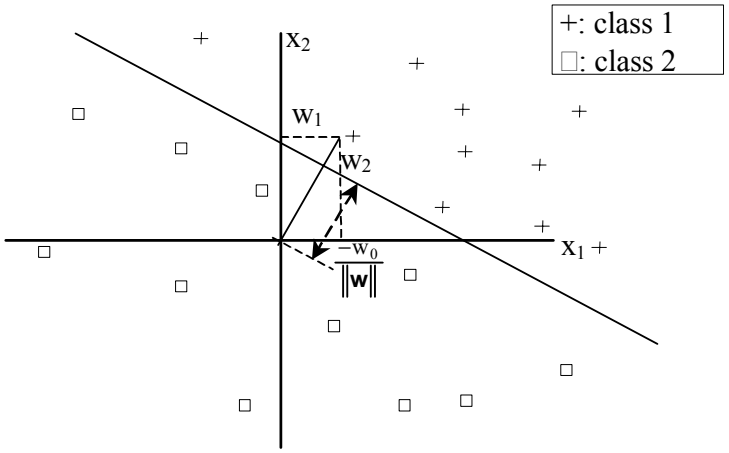

FIGURE 2.5 GEOMETRIC REPRESENTATION OF THE DISCRIMINANT FUNCTION AND THE WEIGHTS.

The remaining issue is how to handle the learning of the weights. There are a variety of learning rules associated with the single-layer feed-forward network. Two of the most famous are the perceptron rule and the delta or least mean square (LMS) rule, which will be discussed in following sections. They both use supervised training (see section 2.2) and work iteratively to update their weights. This procedure can be described with these equations:

$$\begin{aligned} w_i(n+1) &= w_i(n) + \Delta w_i(n) \\ w_0(n+1) &= w_0(n) + \Delta w_0(n) \end{aligned} \tag{2.8}$$

The learning rules are actually different ways to compute $\Delta w_i(n)$ and $\Delta w_0(n)$, where n is the iteration number.

2.4.2 The perceptron learning rule

Presuming we have a data set, with both inputs and desired outputs, an error can be defined with the difference between the desired output, also known as target, and the actual output:

$$e(n) = d(n) - y(n) \tag{2.9}$$

First of all the weighs are given random starting values and then an input vector is presented to the network, which will generate an error. The perceptron algorithm uses this error to calculate the next set of weights, i.e. the new weight vector (equation 2.10). After the weighs are updated a new input vector will be presented to the network and the procedure is repeated until something tells it to stop.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta \cdot e(n) \cdot \mathbf{x}(n) \tag{2.10}$$

The learning rate factor, η, varies between 0 and +1. Contemplating the feed-forward threshold function network in section 2.4.1 the possible error outcomes are:

$$\begin{cases} e = 0 \\ e = +2 \\ e = -2 \end{cases} \tag{2.11}$$

Corresponding equation for the bias:

$$w_0(n+1) = w_0(n) + \eta \cdot e(n) \cdot x_0 \tag{2.12}$$

The following example demonstrates the iterative procedure of updating the weights. For every modification of the weights the discriminant function, illustrated in figure 2.5, is moved. Every modification makes the separation of the two classes more accurate. Two iterations are performed and the desired output can be either +1 or –1, representing the two classes.

2.4.2.1 Example of the perceptron learning rule

The network is initialized with following data:

$$\mathbf{w}(0) = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, w_0(0) = -2, \eta = 0.5$$

Iteration #1

Input vector $\mathbf{x}(0) = \begin{bmatrix} 0.5 \\ 1.5 \end{bmatrix}$ with the desired output d(0) = +1 is presented to the network.

The output is calculated according to equation 2.6,

$$y(0) = F(1 \cdot 0.5 + 2 \cdot 1.5 - 2) = F(1.5) = +1$$

The error is calculated with equation 2.9 and since the output matches the desired output the error, e(0), adopts the value 0.
According to equation 2.10 the weight vector keeps its original values.

Iteration #2

Input vector $\mathbf{x}(1) = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$ with the desired output d(1) = +1 is presented to the network.

The output is calculated according to equation 2.6,

$$y(1) = F(1 \cdot 0.5 + 2 \cdot 0.5 - 2) = F(-0.5) = -1$$

The error is calculated with equation 2.9, e(1) = 2
The weights are altered according to equation 2.10 and 2.12:

$$\mathbf{w}(2) = \begin{bmatrix} 1 \\ 2 \end{bmatrix} + 0.5 \cdot 2 \cdot \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 1.5 \\ 2.5 \end{bmatrix}$$

$$w_0(2) = -2 + 0.5 \cdot 2 \cdot 1 = -1$$

2.4.3 The adaptive linear element

The adaptive linear element, ADALINE, was developed subsequent to the perceptron and uses the least mean square learning procedure, also known as the delta rule. The threshold transfer function has been replaced with a linear transfer function (see equation 2.2). An error function, E, is introduced and in order to find the optimal solution it has to be minimized.

$$E(\mathbf{w}) = \frac{1}{2} e^2(n) \tag{2.13}$$

The error, defined in equation 2.9, can be expressed as:

$$e(n) = d(n) - \mathbf{x}^T(n) \cdot \mathbf{w}(n) \tag{2.14}$$

The main assumption is that the optimal weights are to be found in the direction of the descent gradient of the error function with respect to the weights (steepest descent method) [2], i.e.:

$$\Delta\mathbf{w}(n) = -\eta \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}(n)} \tag{2.15}$$

Differentiating equation 2.13 gives:

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}(n)} = e(n) \frac{\partial e(n)}{\partial \mathbf{w}(n)} \tag{2.16}$$

Equation 2.14 and 2.16 gives:

$$\frac{\partial e(n)}{\partial \mathbf{w}(n)} = \frac{\partial (d - \mathbf{x}^T \cdot \mathbf{w})}{\partial \mathbf{w}(n)} = -\mathbf{x}(n) \tag{2.17}$$

The final result, equation 2.18, becomes seemingly identical to equation 2.10 but in this case e is not limited to adopting the discrete values -2, 0 and +2.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \Delta\mathbf{w}(n) = \mathbf{w}(n) + \eta \cdot e(n) \cdot \mathbf{x}(n) \tag{2.18}$$

Figure 2.6 illustrates the gradient based learning method.

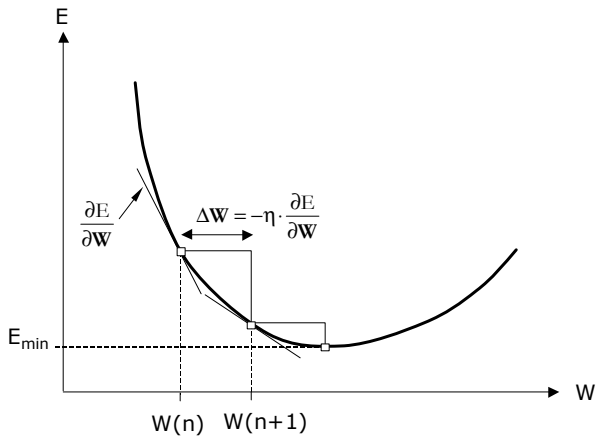

FIGURE 2.6 GRADIENT-BASED UPDATING OF THE WEIGHTS [2].

2.5 Multi-layer feed-forward networks

2.5.1 Multi-layer perceptron

Multi-layer feed-forward networks are often called multi-layer perceptrons, because of their similarity to the perceptron [2]. The big advantages towards single-layer networks are that the MLPs are able to represent non-linear functions and classifying non-linearly separable classes. The difference between linearly separable and non-linearly separable classes is shown in figure 2.7.

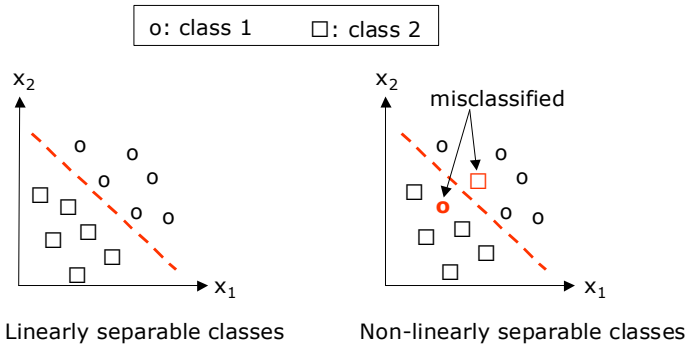

FIGURE 2.7 DIFFERENCE BETWEEN LINEARLY SEPARABLE AND NON-LINEARLY SEPARABLE CLASSES [2].

MLPs have at least one hidden layer but it has been proven that one layer with hidden neurons is enough to approximate any continuous function if it only has a sufficient number of units, provided the transfer functions of the hidden units are non-linear (see figure 2.8) [2]. Finding the sufficient number of hidden neurons, H, is a trial-and-error process.

Since the multi-layer perceptron has proven suitable for power plant applications other networks, such as self-organizing, will be left out of the discussion [2]. For more information on these networks Kröse's *An Introduction to Neural Networks* [7] is recommended.

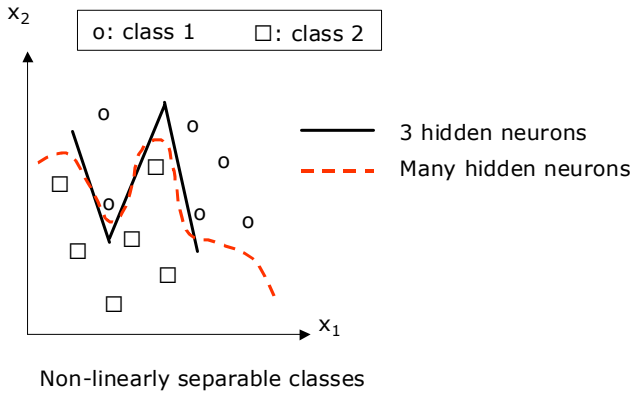

FIGURE 2.8 COMPLEX DECISION BORDERS WITH HIDDEN UNITS [2].

The general structure of a fully connected MLP, with M input nodes, H hidden neurons and N output neurons, is shown in figure 2.9.

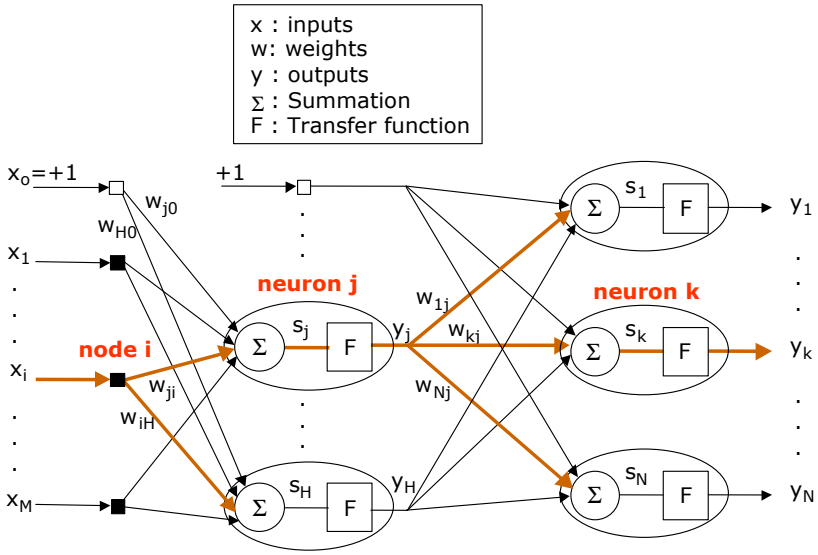

FIGURE 2.9 GENERAL STRUCTURE OF A TWO-LAYERED FEED-FORWARD MLP [2].

MLPs are trained in a supervised manner and they are most frequently trained with the back-propagation (BP) algorithm (described in section 2.5.2), originally known as the generalized delta rule. The advantages of MLPs have been known for a long time but it was not until in 1986, when Rumelhart, Hinton and Williams [11] presented the generalized delta rule, a satisfying way to update the weights was found. The original delta rule was only applicable on single-layer networks with linear transfer functions.

There are three different modes when training MLPs:

- Batch training: All available input patterns[1] are presented to the network at the same time and an average error is calculated and then back propagated. The weights are updated and the process is repeated, i.e. the weights are updated epoch-by-epoch[2]

- Sequential training: Every input pattern is presented individually to the network and the weights are updated pattern-by-pattern.

- Block training: This is a combination of the techniques mentioned above. The input patterns are divided into blocks and the weights are updated block-by-block.

The batch-training mode is considered to be the most stable but it is also associated with slow convergence rate.

[1] One input pattern is a group of inputs presented to the network at the same time and related to specific outputs.
[2] One epoch is a cycle finished when all available training input patterns have been presented to the network once during training.

2.5.2 Back-Propagation

The back-propagation method is also called the generalized delta rule, which is a generalized form of the LMS rule. What makes this more complicated is that the error has to affect two sets of weights, i.e. the weights between hidden and output layer, and the weights between input and hidden layer. The back-propagation method requires differentiable transfer functions, e.g. sigmoid or hyperbolic, in order to modify the weights. The training can be divided into two phases:

- An error is calculated and the weights between the hidden and output layer are updated, comparable with the perceptron and the ADALINE.

- Since no information about the desired output from the hidden layer is available the error from the ouput layer is back propagated and used to update the weights between the input layer and the hidden layer.

BP is valid for networks with more than one hidden layer but, as stated before, one hidden layer with sufficient number of hidden units can approximate any continuous function. For a detailed description of the BP method, complete with equations, Arriagada's *Introduction of Intelligent Tools for Gas Turbine Based, Small-Scale Cogeneration* [2] is recommended.

2.5.3 Data handling

An important aspect, not to be overlooked, in order to gain optimal results training MLPs, is the handling of available data. Following is to consider before, during and after training an ANN:

- Before starting any kind of training it must be realized that an ANN cannot become any better than the accuracy of the available data.

- A thorough review of the data to avoid and remove signals that are obviously distorted, i.e. outliers. If they are included in the training they might have a negative effect on the overall predictive performance of the network.

- The entire data set should be randomized and divided into a training set, a validation set and a test set. Different recommendations about the sizes of these sets are available but in general one third is used for testing while the rest is shared between the training and validation set. Approximately two thirds of this part is used for the training set and one third for the validation set. The training set is used to generate errors, which are used to update the weights, while the validation set is used to check the performance during training, to avoid over training (discussed in section 2.5.4). After the training is complete the test set is used to check the performance of the network with before unseen data, i.e. the nets generalization capability. This methodology is called cross-validation.

- Before introducing the data to the network it should be normalized to avoid that a signal with a value greater than the others does not become dominant. Normally data is linearly normalized within the boundaries for the transfer function output, e.g. [0,1] or [-1,1]. In order to understand the output signals, from the MLPs, they have to be de-normalized within the same boundaries. Figure 2.10 demonstrates this method.

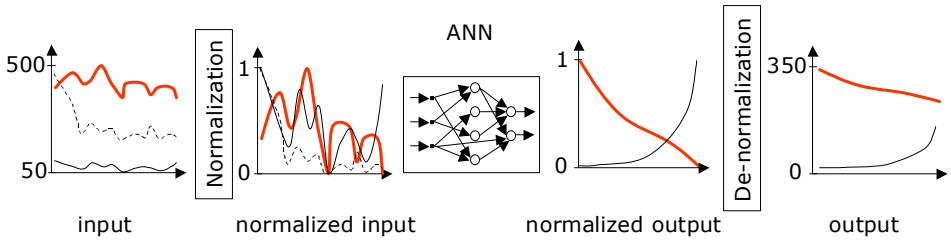

FIGURE 2.10 PRE-PROCESSING AND POST-PROCESSING OF THE DATA [2].

2.5.4 Overfitting

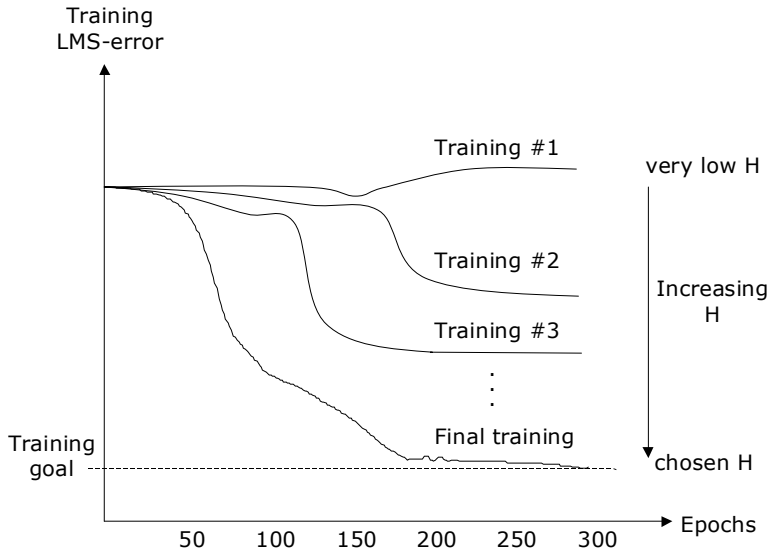

FIGURE 2.11 TUNING THE SIZE OF THE HIDDEN LAYER BY TRIAL AND ERROR [2].

As mentioned before, finding the right number of hidden units is a trial-and error process, well illustrated in figure 2.11. This however can cause some problems with overfitting the network. This means that instead of having a good generalization capability, the network will simply memorize the training data. When choosing H, it is important to consider that the predictability for the training data will improve with increasing H but at some point the predictability for the validation and test data will worsen.

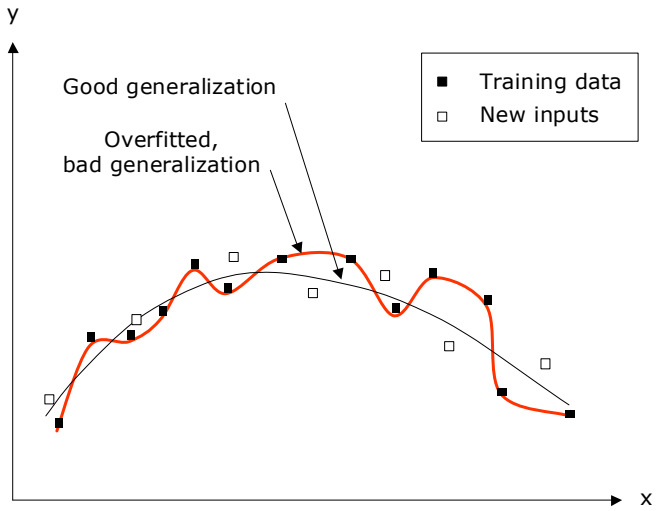

FIGURE 2.12 OVERFITTING OF TRAINING DATA [2].

Overfitting can also occur if training is performed with too many epochs, i.e. learning iterations. Fortunately, this can easily be overcome by using the cross-validation method (described in section 2.5.3), which monitors the network during training and can perform an early stop if overfitting is detected. The overfitting is detected when the LMS error for the validation set starts to increase but the LMS error for the training set continues to decrease. Figure 2.12 shows an overfitted network.

2.6 Concluding remarks

The aim of this chapter is to provide knowledge about ANNs and specifically MLPs. Different networks, and the learning rules accompanied with them, are addressed. The basics of data handling before, during and after training are described. Furthermore overfitting, a long with measures avoiding it, is explained.

3 NeuroSolutions

All ANNs have been modeled with the commercial software program NeuroSolutions (NS).

There are six levels of NeuroSolutions, all of which allow you to implement your own neural models. The one used is the educator, the entry-level version, intended for those who want to learn about neural networks and work with MLPs.
The users version extends the educator with a variety of neural models for static pattern recognition applications. The consultants version offers enhanced models that support dynamic pattern recognition, time-series prediction and process control problems. Furthermore, the consultant level offers the possibility to implement ANNs as DLL-files, which allows a smooth integration of finished ANNs into almost any software [10].

NeuroSolutions for Excel
NeuroSolutions for Excel is an Excel Add-in that integrates with any of the six levels of NeuroSolutions to provide a very powerful environment for manipulating your data, generating reports, and running batches of experiments [10].

3.1 Methodology

The gathering and pre-processing of data is performed in Excel with the assistance of additional, NeuroSolutions specific, commands. Input and output parameters are chosen, randomized and divided into sets (training, validation and testing).

The network is constructed in NS by means of selecting e.g. transfer functions, normalization boundaries and number of hidden units and epochs.

Once these steps are complete the training of the network can commence. NS offers the possibility of performing training variations, which simplifies the trial-and-error process of finding a satisfying number of hidden neurons (the best converging configuration will be saved). NS also offers the option of training with the cross-validation method and thereby avoiding overfitting of the network. A training report is created in Excel where the results for all the training variations can be studied.

Running all input patterns through the network and comparing the predicted outputs with the desired will test the predictive performance of the network.

The remaining issue is to create a user interface. This is accomplished by using the NS application, Formula Creator, which extracts the network to Visual Basic (VB) code, which in return can be employed as a function in Excel (after some adaptation of the VB code).

4 Lunds Energi: Gas Turbine and Heat Recovery Unit

The GT and HRU with district heating system are illustrated schematically in figure 4.1 below. The GT produces electricity and large amounts of hot exhaust gases. The exhaust gases passes thru the HRU and heat is extracted to the district heating water, which is distributed to the consumers along with the produced electricity.

The positioning of the sensors, used to create the models in chapter 5, can also be seen in the figure.

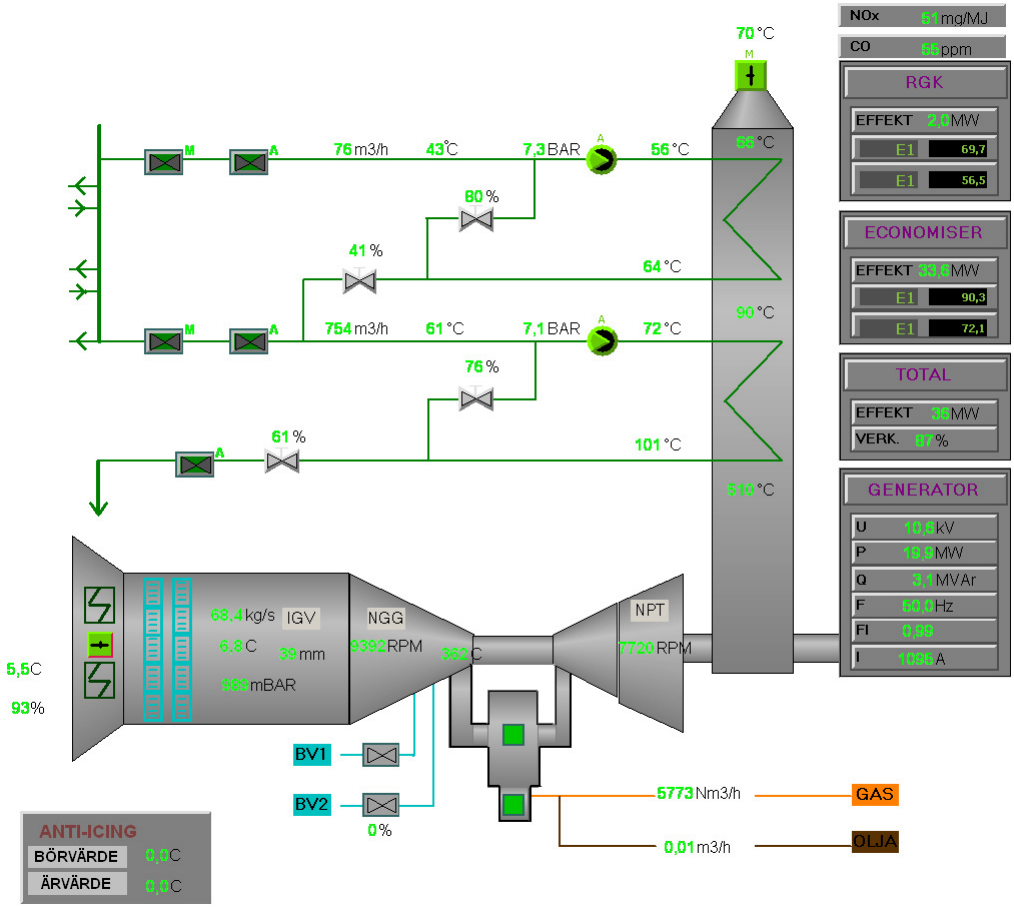

FIGURE 4.1 SCREEN DUMP FROM LUNDS ENERGI CONTROL PANEL [16].

4.1 The gas turbine

The gas turbine, GT10A, was installed by ABB STAL, now Siemens, in 1991 and rebuilt to a GT10B that same year. The GT10 is a lightweight industrial gas turbine (GT). The axial compressor has 10 stages and a pressure ratio of 13.6:1. The combustion chamber is annular and built for dual fuels, i.e. gas and liquid (at the moment Lunds Energi uses natural gas (NG)). The turbine has 2 stages where the first one is powering the compressor and the second one is connected to the generator. The turbine inlet temperature (TIT) can reach a maximum of 1062 °C and the GT has the capacity of producing 22 MW electricity [16]. The gas turbine is illustrated schematically in figure 4.2 below.

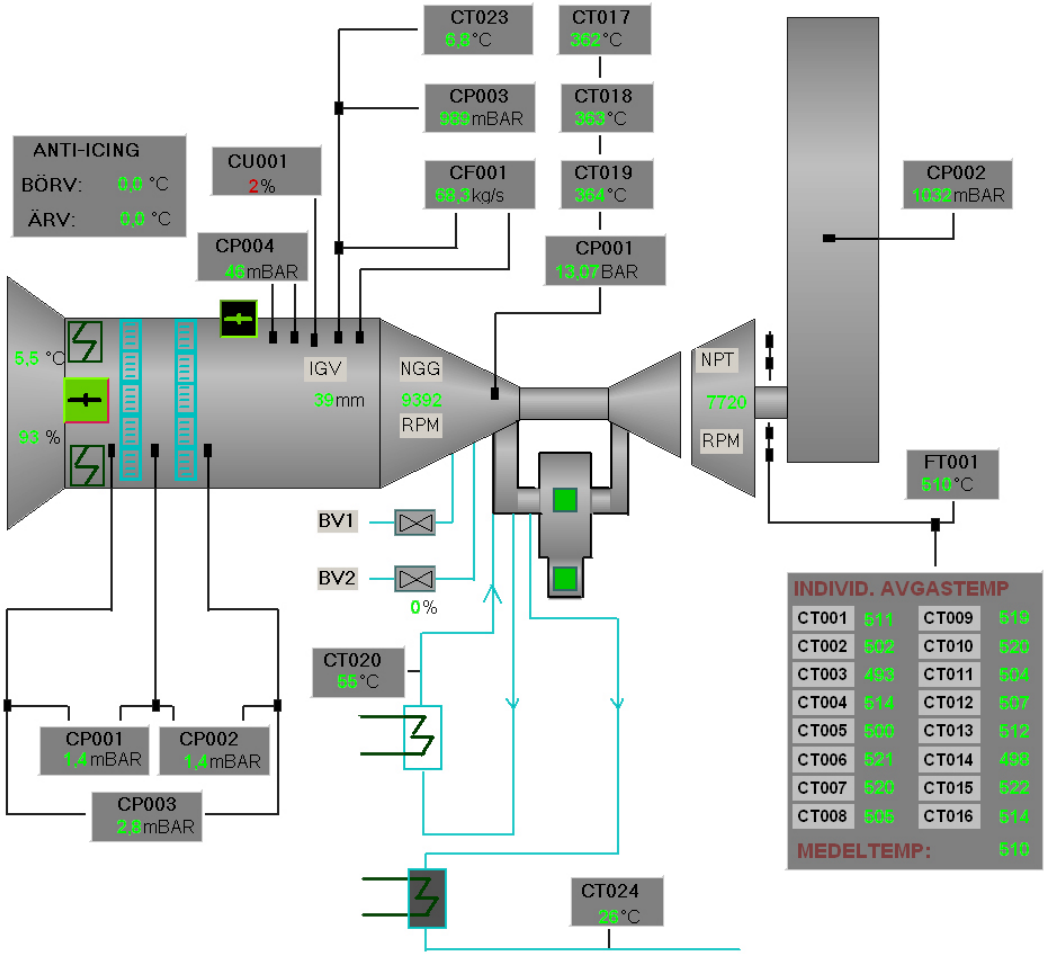

FIGURE 4.2 SCREEN DUMP FROM LUNDS ENERGI CONTROL PANEL [16].

4.1.1 Anti icing

If required anti icing air for the inlet is taken from the compressor delivery via a bleed-valve[3]. The anti icing system prevents ice from forming in the intakes during

[3] Certain amount of the main flow is bleed off in a bleed-valve.

high humidity/cold ambient air conditions. The danger with ice reaching the compressor is the potential harm it can inflict on the airfoils.

The mass flow thru the gas turbine's expander section will decrease and since the power output is directly dependant on this it will also decrease (see figure 5.2, page 29).

4.1.2 Performance degradation/deterioration

Even under normal engine operating conditions, with a good inlet filtration system, and using a clean fuel, the engine flow path components will become fouled, eroded, corroded, covered with rust scale, damaged, etc. The result will be deterioration in engine performance, which will get progressively worse with increased operating time. Three types of deterioration are recognized [5]:

- Performance deterioration recoverable with cleaning/washing.
- Performance deterioration non-recoverable with cleaning/washing.
- Permanent performance deterioration, which is not recoverable after an overhaul.

Recoverable deterioration is often caused by air-born particles, e.g. dirt, dust and pollen. The particles accumulate on the compressor airfoils and gas path surfaces (fouling the surfaces). Together with soot particles (NG is soot-free), produced in the combustor, they can also accumulate on the flow path surfaces in the turbine. Oil leaks and the presence of oily substances in the air can exacerbate the deterioration by acting as glue to the particles. Compressor fouling results in decreased mass flow and compressor efficiency. Hot-end fouling results in a reduction in the overall turbine efficiency. Three different methods of cleaning/washing the compressor are described below [5]:

- On-line[4] dry cleaning of compressor with e.g. rice husks or pecan shells.
- On-line wash of compressor with water, water mixed with detergent, or some other suitable fluid, which is preferably nontoxic, nonflammable, and biodegradable.
- Off-line soak wash with suitable fluid

The last method described is the only one that requires a shut down of the gas turbine but is also the most efficient method, i.e. the one that gives a higher grade of recovery in GT performance.

[4] When the GT is in operation.

4.1.3 Emissions

Gas turbine combustion is a steady flow process in which a hydrocarbon fuel is burned with a large amount of excess air, to keep the turbine inlet temperature at an appropriate value [12]. The exhaust gases are primarily a mixture of O_2, N_2, H_2O and CO_2 but also include small proportions of e.g. NO_x, CO and unburned hydrocarbons (UHC). Although the proportions of the later mentioned emissions are very small, the large flow of exhaust gases produces a considerable amount of pollutants. The only way to minimize CO_2 emissions, which is believed to contribute to global warming, is to increase the efficiency of the gas turbine or by using non-fossil fuels.

4.1.3.1 NO_x, CO and UHC emissions

There are many different parameters affecting the amount of pollution created, e.g ambient conditions, load, flame temperature and residence time[5]. The most important factor, regarding the formation of NO_x, is the flame temperature, which has a maximum at stoichiometric conditions. The relationship between NO_x formation and flame temperature is seen in figure 4.3 below.

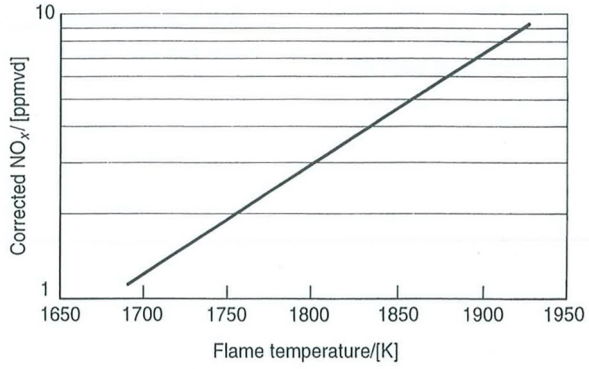

FIGURE 4.3 EFFECT OF FLAME TEMPERATURE ON NO_x EMISSIONS [12].

Operating far away from stoichiometric conditions (which gas turbines do) reduces NO_x but unfortunately CO and UHC increases. The relationship can be seen in figure 4.4.

[5] Time spent in the combustor.

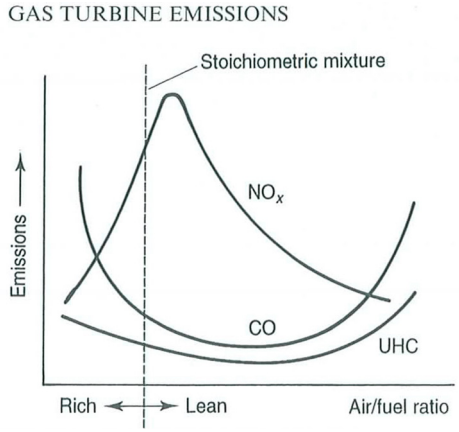

FIGURE 4.4 DEPENDENCE OF EMISSIONS ON FUEL/AIR RATIO [12].

4.2 The heat recovery unit

The heat recovery unit produces hot water for the district heating system by extracting heat from the exhaust gases in an economizer and a gas condenser (see figure 4.1). By using the excess heat produced by the gas turbine the overall efficiency of the plant increases dramatically. Approximately 35 MW heat is extracted. A regulator that continuously adjusts the valves situated on the district heating pipes automatically controls the HRU. Thru this a constant gas temperature after the gas condenser, i.e. the temperature of the gas exiting the chimney, and a desirable district heating delivery temperature is maintained.

4.2.1 Degradation

Degradation of the HRU can occur, thru the formation of coatings on the heat exchanger surfaces, when burning e.g. coal or oil but when using natural gas this problem disappears. A more probable cause of degradation would be coatings forming on the inside of the heat exchangers because of substances present in the district heating water.

5 Case Study

In this chapter the possibilities of training ANN models, with actual operational data from the power plant (described in chapter 4), are examined. The available operational data consists of approximately 100 different parameters saved as minute averages, and they can be divided into the three following groups:

- Gas Turbine
- Heat Recovery Unit and District Heating
- Emissions

One of the first steps is to select the parameters that are of significance to the current model. This is primarily done with the author's knowledge in the field of gas turbines and combined heat- and power plants. Different models will be studied and for each case unique combinations of parameters will be used.

Section 5.2 discusses GT models with performance predictions and section 5.3 considers the use of ANN sensor validation models. All ANNs are modeled in NeuroSolutions using MLP networks.

5.1 Data treatment

As mentioned in section 2.5.3 all data must be examined in order to discover distorted or unwanted data. For instance, this specific GT is always operated at full load but when the gas turbine is turned on there is a short period of time before the GT has reached its operating point. This data, corresponding to the start sequence, could be confusing for an ANN and simply worsen the predictive performance of the model. In a similar way, the data produced when the GT is turned off is meaningless for modeling purposes. Another thing to take into consideration is that a sensor can fail to deliver a correct value, e.g. the reading can be zero. If this remains undetected it can be devastating for the ANN model. Finally data with very large errors, i.e. outliers, has to be removed for the same purpose.

5.2 Gas turbine

Since there is a considerable amount of parameters available, two models will be trained separately (GT and emission models). This is done to simplify and hopefully improve the performance of the models.

Ambient conditions, i.e. relative humidity, atmospheric temperature and pressure, will be used as input and gas turbine parameters, e.g. power output, compressor discharge temperature and emissions, as outputs.

One initial problem encountered is that when the anti icing system is in operation the GT performance will differ significantly (for more information about anti icing, see section 4.1.1). Training an ANN with data both from when the anti icing system is in operation and when it is turned off can be compared with training an ANN with data from two different GTs. The obvious solution to this problem is to separate these data sets and train two different ANNs. A better and more sophisticated way to cope with the problem is to add an extra input parameter with the value 1 if the anti icing system is in operation and the value 0 otherwise. This will assist ANN to differentiate between the two cases (a comparison between an ANN with anti icing as input and one without is made in paragraph 5.2.2.2.1). The remaining issue is how to access this parameter. Since the ambient conditions control whether the anti icing system is in operation or not, it is possible to train an ANN with these conditions as input and anti icing as output. The system will then be modeled with both these ANNs (see figure 5.0). The output from the first (anti icing model) will act as input to the second (GT model). The results from the ANN anti icing model can be studied in this next section.

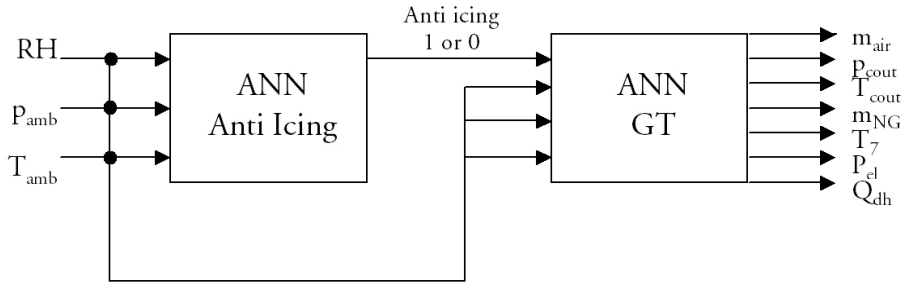

FIGURE 5.0 GT SYSTEM MODELED WITH TWO ANNS.

5.2.1 ANN anti icing model

Since data for the anti icing system does not exist it has to be produced manually, i.e. typed in by hand in Excel. This is accomplished by studying the compressor inlet temperature and when large leaps, within a small timeframe, are found it indicates that the anti icing system either is turned on or off. Table 5.1 demonstrates an arbitrary leap.

There is the possibility of recording the anti icing activity during GT operation, and thereby eliminating this procedure, but unfortunately there was no access to this kind of data during this master thesis.

TABLE 5.1 PARAMETERS USED FOR ANTI ICING MODEL.

#	Date/Time	Relative humidity [%]	Ambient pressure [mbar]	Ambient temp. [°C]	Comp. inlet temp. [°C]	Anti icing [1-on:0-off]
695	2005-02-11 11:34	78.5	983.4	4.0	10.0	1
696	2005-02-11 11:35	78.2	983.3	4.0	9.8	1
697	2005-02-11 11:36	78.3	983.3	4.0	10.0	1
698	2005-02-11 11:37	78.4	983.5	4.0	10.0	1
699	2005-02-11 11:38	78.2	983.4	4.1	9.8	1
700	2005-02-11 11:39	78.0	983.4	4.1	10.1	1
701	2005-02-11 12:12	77.7	984.1	3.3	3.8	0
702	2005-02-11 12:13	77.8	984.2	3.3	3.8	0
703	2005-02-11 12:14	78.0	984.3	3.3	3.8	0
704	2005-02-11 12:15	78.1	984.1	3.3	3.7	0
695	2005-02-11 11:34	78.2	984.2	3.3	3.8	0

The transition is actually more gradual than the table shows but still very detectable (a closer look at the date/time column will reveal this). The reason for this is that data representing the transition period, i.e. the period from when the anti icing system is turned off until the GT has stabilized in its new operation point, has been removed because if included in training it can worsen the predictive performance of the ANN.

For the ANN anti icing model a data set with approximately 11.000 rows, which covers a week, will be used. During this period the anti icing system is turned on/off several times, enough for ANN to learn the relations between ambient conditions and anti icing. Relative humidity, ambient pressure and ambient temperature will be used as inputs. The compressor inlet temperature will not be used in training because it is not accessible before the GT is in operation. As mentioned before, anti icing will be used as the sole output. A training variation, 1-10 hidden neurons and a maximum of 5000 epochs, shows that the best results are attained with 8 hidden neurons and 5000 epochs (for more information about ANN specifics turn to chapter 2). The predictive performance for this ANN is shown in table 5.2.

TABLE 5.2 PREDICTIVE PERFORMANCE FOR ANTI ICING MODEL.

	Correct	Misclassified
Quantity	10887	148
Share [%]	98.70	1.30

The results are considered acceptable and can be used as input in other ANNs. One possible origin of the missclassified data can be large gradients in the input parameters, e.g. ambient temperature, which could lead to a delayed start/stop of the anti icing system.

5.2.2 ANN GT model

Since the GT is operated at full load, it is assumed that only the ambient conditions and anti icing affects the performance. The input and output parameters used are:

TABLE 5.3 INPUT PARAMETERS FOR GT MODEL.

Input parameters	Entity
Anti icing	1 or 0
Relative Humidity	%
Ambient pressure	mbar
Ambient temperature	°C

TABLE 5.4 OUTPUT PARAMETERS FOR GT MODEL.

Output parameters	Entity
Airflow	kg/s
Compressor outlet pressure	bar
Compressor outlet temperature	°C
Gas flow	MJ/s
Turbine outlet temperature	°C
Power output	MW
Generated heat	MW

Although the generated heat is associated with the HRU it will be included because the whole system is in focus. The gas flow entity is [MJ/s] but still called a flow because it is merely multiplied with a constant calorific value [MJ/kg]. This model will not use the predicted values for anti icing but the manually produced (observe table 5.1), because it is desirable to separate potential errors origin. A training variation, with 1-10 hidden neurons and 10000 epochs, shows that the best network has 7 hidden neurons and is trained with 10000 epochs. Training with additional neurons, e.g. 11-20, will only improve the results marginally and is thereby considered unnecessary. Approximately 7500 rows of data are used for this training. An advantage of using large amount of data is that the contribution of eventual erroneous points will be very small and probably not affect the overall performance of the trained network. Table 5.5 demonstrates an arbitrary input pattern with its predicted outputs. Table 5.6 and 5.7 illustrates the error distribution, average and maximum error.

TABLE 5.5 ARBITRARY INPUT PATTERN AND PREDICTED OUTPUTS.

Input	Signal	Output	Signal	Predicted	Error [%]
Anti Icing	1	m_{air}	66.03	66.26	0.36
RH	84.29	p_{cout}	12.68	12.71	0.26
p_{amb}	980.20	T_{cout}	365.87	365.86	0.00
T_{amb}	2.31	m_{NG}	61.66	61.75	0.14
		T_7	513.41	513.36	0.01
		P_{el}	19.06	19.16	0.54
		Q_{dh}	34.25	34.33	0.25

TABLE 5.6 ERROR DISTRIBUTION FOR THE PREDICTIONS FROM THE GT MODEL.

Error [%]	<1	1-2	2-4	>4
Airflow	7498	0	0	0
Compressor outlet pressure	7498	0	0	0
Compressor outlet temperature	7498	0	0	0
Gas flow	6950	548	0	0
Turbine outlet temperature	7498	0	0	0
Power output	7494	4	0	0
Generated heat	6328	1051	118	1

TABLE 5.7 PREDICTION ERRORS FOR THE GT MODEL.

	Airflow	Compressor outlet pressure	Compressor outlet temperature	Gas flow	Turbine outlet temperature	Power output	Generated heat
Average Error [%]	0.13	0.12	0.09	0.50	0.04	0.21	0.56
Maximum Error [%]	0.84	0.75	0.52	1.96	0.20	1.23	5.09

Good results are attained for most of the parameters and fairly good for the remaining. The gas flow and generated heat are the two parameters that have higher prediction errors. The generated heat is dependent on other parameters than the ambient conditions such as the district heat return temperature and flow. The reason for the slightly higher errors in gas flow predictions can be referred to the fact that no consideration to variations in the calorific value of the fuel (Danish natural gas) is taken. To increase the predictive performance for the gas flow, the calorific value should be added as an input to the network. Unfortunately this kind of data is not available for modeling so the results have to be accepted as they are. The variation in higher calorific value for the natural gas used in the GT is illustrated in figure 5.1. Another reason for the slightly higher error can be that gas flows are generally more difficult to measure with high accuracy.

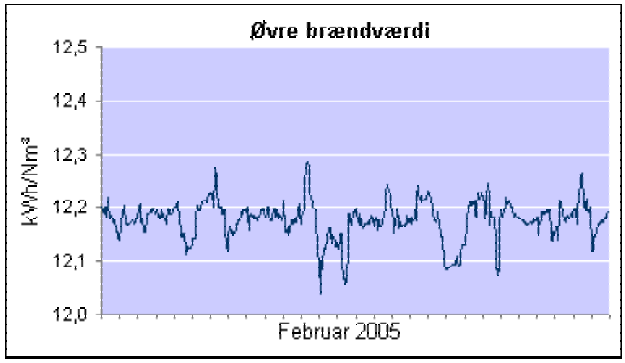

FIGURE 5.1 HIGHER CALORIFIC VALUE VARIATIONS FOR NG IN FEBRUARY 2005 [6].

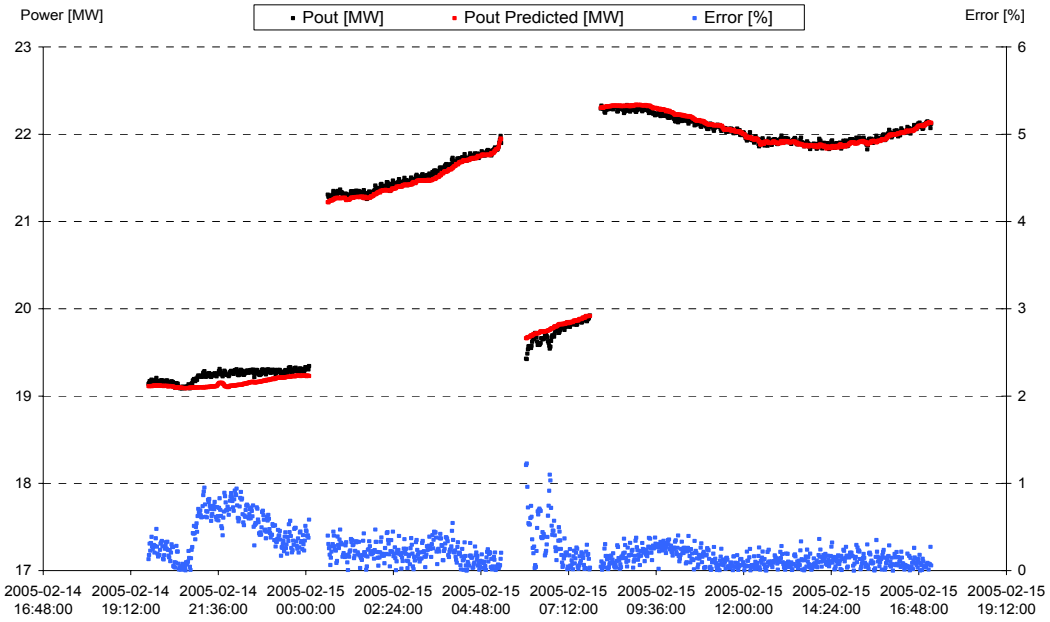

FIGURE 5.2 POWER OUTPUT PREDICTION FOR GT MODEL.

An arbitrary section of the data set including prediction values, for the power output, is presented in figure 5.2. The figure clearly shows when the anti icing system is in operation or not (high power output = anti icing off). It also shows ANNs capability of predicting the power output regardless of the anti icing systems influence.

The ramifications of using the predicted anti icing values, instead of the manually produced, would be that a certain amount (1.3%) of the input patterns would be misclassified (see table 5.2) and the predicted outputs would be totally wrong.

5.2.2.1 Extrapolation capabilities

The optimal network is achieved when training with data covering the whole operating range, while this specific network only uses data gathered during one week. The reason for not training with more extensive data is that data is usually not saved for a long period of time and waiting several months to get fresh data might not be an option (at least not during this master thesis). This leads us to the question of how the model will respond if presented with data outside of the training interval, i.e. the network's extrapolation capability. For investigative purpose a random input parameter will be chosen, in this case ambient temperature. The interval, in which the temperature varies during training, is seen in table 5.8

TABLE 5.8 INPUT PARAMETER TRAINING INTERVALS.

	RH	p_{amb}	T_{amb}
Max	91.99	996.66	4.06
Min	65.24	935.74	-4.50

An appropriate test data set with ambient temperatures outside of the training interval is used, in this data set the temperature varies from 4.5 °C to 13.5°C. The data is chosen so that the other input parameters remain within the same interval used when training the network. This means that the extrapolation capacity is tested with respect to the ambient temperature. Unfortunately the combustion chamber has been replaced in between the recording of these data sets and can cause the GT to behave differently. Anyhow, the following results are attained:

TABLE 5.9 EXTRAPOLATION: ORIGINAL SIGNALS.

INPUT				OUTPUT						
Anti Icing	RH	p_{amb}	T_{amb}	m_{air}	p_{cout}	T_{cout}	m_{NG}	T_7	P_{el}	Q_{dh}
0	79.8	986.2	4.5	69.4	13.4	363.8	65.9	506.1	20.7	35.6
0	76.4	986.0	5.5	69.2	13.3	363.8	65.6	506.6	20.6	35.6
0	77.9	984.6	6.5	68.6	13.2	363.8	65.1	507.7	20.3	35.2
0	71.8	982.4	7.5	68.0	13.1	363.8	64.4	508.7	20.0	35.6
0	74.2	980.2	8.5	67.3	13.0	365.6	63.8	509.4	19.8	34.8
0	88.9	976.0	9.5	66.5	12.8	365.5	62.6	510.9	19.4	34.7
0	72.2	976.0	10.5	66.0	12.6	365.6	63.1	512.2	19.1	33.5
0	89.5	973.7	11.5	65.4	12.6	365.5	61.4	512.7	19.0	34.6
0	90.0	973.6	12.5	64.9	12.6	367.3	61.1	513.7	18.8	33.7
0	90.3	973.7	13.5	64.6	12.5	367.3	60.7	514.0	18.7	34.1

TABLE 5.10 EXTRAPOLATION: PREDICTIONS.

INPUT				PREDICTED						
Anti Icing	RH	p_{amb}	T_{amb}	m_{air}	p_{cout}	T_{cout}	m_{NG}	T_7	P_{el}	Q_{dh}
0	79.8	986.2	4.5	68.7	13.1	361.6	64.2	507.4	20.2	35.4
0	76.4	986.0	5.5	68.1	13.0	361.4	63.3	508.2	19.9	35.2
0	77.9	984.6	6.5	67.3	12.9	361.3	62.3	509.1	19.6	34.9
0	71.8	982.4	7.5	66.8	12.8	360.9	61.7	509.7	19.4	34.7
0	74.2	980.2	8.5	66.2	12.7	360.7	61.0	510.4	19.2	34.5
0	88.9	976.0	9.5	65.2	12.5	362.4	60.2	511.2	18.8	34.1
0	72.2	976.0	10.5	65.4	12.5	360.4	60.3	511.3	18.9	34.1
0	89.5	973.7	11.5	64.7	12.4	362.4	59.7	511.9	18.6	33.8
0	90.0	973.6	12.5	64.5	12.4	362.5	59.6	512.2	18.6	33.7
0	90.3	973.7	13.5	64.3	12.3	362.6	59.5	512.3	18.5	33.6

TABLE 5.11 EXTRAPOLATION: ERRORS.

INPUT				ERROR [%]						
Anti Icing	RH	p_{amb}	T_{amb}	m_{air}	p_{cout}	T_{cout}	m_{NG}	T_7	P_{el}	Q_{dh}
0	79.8	986.2	4.5	1.1	1.7	0.6	2.7	0.3	2.3	0.6
0	76.4	986.0	5.5	1.6	2.2	0.7	3.6	0.3	3.1	1.2
0	77.9	984.6	6.5	1.9	2.4	0.7	4.2	0.3	3.4	0.8
0	71.8	982.4	7.5	1.7	2.3	0.8	4.1	0.2	2.9	2.5
0	74.2	980.2	8.5	1.7	2.4	1.3	4.3	0.2	3.3	0.9
0	88.9	976.0	9.5	2.0	2.5	0.8	3.8	0.1	3.2	1.5
0	72.2	976.0	10.5	0.9	0.9	1.4	4.5	0.2	1.1	1.7
0	89.5	973.7	11.5	1.2	1.8	0.8	2.8	0.1	1.9	2.2
0	90.0	973.6	12.5	0.7	1.6	1.3	2.5	0.3	1.3	0.2
0	90.3	973.7	13.5	0.5	1.4	1.3	2.1	0.3	0.8	1.4

As seen above, the predictions have larger errors but if this is caused by the new combustion chamber, bad extrapolation capability or both is hard to determine in

this study. One important conclusion drawn from the result is that the errors do not increase with increased ambient temperature. The trends of e.g. decreased power output and increased T_7 are predicted well. All over the result is satisfying, giving a hint of the actual values although training with a bigger interval, to avoid extrapolation, would be preferred.

5.2.2.2 Sensitivity analysis

The purpose of this paragraph is to understand the significance different inputs have on the network predictions. Both the influence of currently existing and supplementary inputs will be investigated.

5.2.2.2.1 ANN GT model without anti icing as input

Given the same prerequisites, as the original ANN GT model, an optimal network was found using 8 hidden neurons and 10000 epochs. The result is mediocre and it strengthens the statement that anti icing is a required input, made in section 5.2.

TABLE 5.12 PREDICTION ERRORS FOR THE GT MODEL WITHOUT ANTI ICING AS INPUT.

	Airflow	Compressor outlet pressure	Compressor outlet temperature	Gas flow	Turbine outlet temperature	Power output	Generated heat
Average Error [%]	0.50	0.47	0.14	0.79	0.14	0.79	0.71
Maximum Error [%]	7.31	6.89	2.21	9.2	2.07	12.18	6.98

TABLE 5.13 ERROR DISTRIBUTION FOR THE PREDICTIONS FROM THE GT MODEL WITHOUT ANTI ICING AS INPUT.

Error [%]	<1	1-2	2-4	>4
Airflow	6562	657	279	0
Compressor outlet pressure	6660	559	250	29
Compressor outlet temperature	7457	22	19	0
Gas flow	5747	1245	437	69
Turbine outlet temperature	7444	44	10	0
Power output	5858	987	495	158
Generated heat	5618	1518	338	24

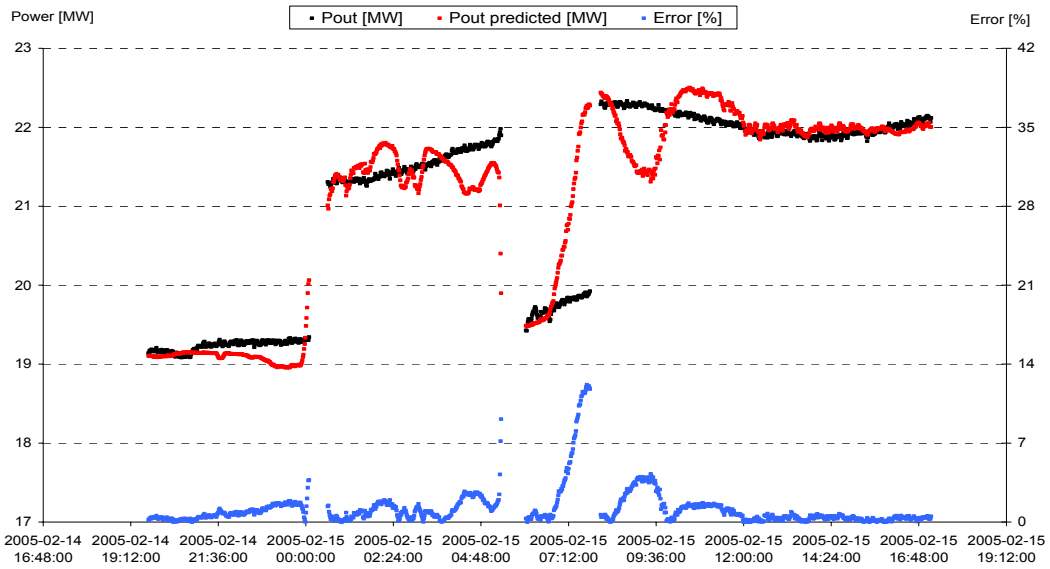

FIGURE 5.3 POWER OUTPUT PREDICTION FOR GT MODEL WITHOUT ANTI ICING AS INPUT.

For justification purpose the power output, with corresponding predictions, is plotted for the same time period as in figure 5.2. The predictive performance of the network has decreased dramatically after the anti icing input was removed. The maximum error for the power prediction is now over 12%.

5.2.2.2.2 ANN GT model with fuel temperature as input

The NG is preheated in order to maintain a specific temperature when it enters the combustion chamber. Nevertheless small variations occur and this changes the natural gas's latent energy, which in return can create small variations in GT performance, e.g. changed gas flow. The purpose of this paragraph is to investigate the possibilities of enhancing the network's predictive performance by including the NG temperature as an input. The best network, given the same prerequisites as the other models, was found using 10 hidden neurons and 10000 epochs. The mean and maximum errors are shown in table 5.14

TABLE 5.14 PREDICTION ERRORS FOR THE GT MODEL WITH T_{FUEL} AS INPUT.

	Airflow	Compressor outlet pressure	Compressor outlet temperature	Gas flow	Turbine outlet temperature	Power output	Generated heat
Average Error [%]	0.14	0.12	0.09	0.48	0.03	0.2	0.55
Maximum Error [%]	0.76	0.70	0.56	1.96	0.22	1.22	5.25

The results show virtually no improvement in the predictive performance, so in conclusion the variations of the natural gas's latent energy can be disregarded from and the first assumption, to only use ambient conditions (and anti icing) as inputs, is confirmed.

5.2.2.2.3 ANN GT model input variation

In this paragraph the three ambient conditions will be removed as inputs one at the time, to help understand their connection and importance to different output parameters. The same training variation as before is used and the errors can be studied in the tables below.

TABLE 5.15 PREDICTION ERRORS FOR THE GT MODEL WITHOUT P_{AMB} AS INPUT.

	Airflow	Compressor outlet pressure	Compressor outlet temperature	Gas flow	Turbine outlet temperature	Power output	Generated heat
Average Error [%]	0.73	0.73	0.09	0.91	0.04	0.77	0.92
Maximum Error [%]	2.97	2.94	0.50	3.71	0.25	3.1	5.95

TABLE 5.16 PREDICTION ERRORS FOR THE GT MODEL WITHOUT T_{AMB} AS INPUT.

	Airflow	Compressor outlet pressure	Compressor outlet temperature	Gas flow	Turbine outlet temperature	Power output	Generated heat
Average Error [%]	0.38	0.35	0.12	0.75	0.08	0.59	0.67
Maximum Error [%]	2.85	2.80	0.57	4.54	0.69	4.88	5.08

TABLE 5.17 PREDICTION ERRORS FOR THE GT MODEL WITHOUT RH AS INPUT.

	Airflow	Compressor outlet pressure	Compressor outlet temperature	Gas flow	Turbine outlet temperature	Power output	Generated heat
Average Error [%]	0.13	0.11	0.09	0.5	0.04	0.2	0.56
Maximum Error [%]	0.74	0.65	0.45	1.98	0.21	1.12	4.95

In the first scenario, i.e. without ambient pressure as input, all output parameters are affected except compressor outlet temperature and turbine outlet temperature. In the second scenario all of the output parameters are affected more or less. The last scenario shows that the relative humidity has no effect on neither of the output parameters, which means that the ambient pressure and temperature (and anti icing which is not a parameter per se) are the only two parameters needed for modeling this GT (of course, the relative humidity is still needed for the anti icing model). This makes the result of the two first scenarios even more interesting, showing that with using only one parameter (relative humidity has no effect) it is possible to predict the outputs with fairly good accuracy, some even with the same precision.

5.2.3 ANN GT emission model

The same input parameters will be used for modeling the GT emissions but the probability of predicting all of them is low because they often depend on combustion chamber specific parameters (see paragraph 4.1.3.1) that are not accessible. For this task a section of data without the anti icing system turned on will be used (the influence of this parameter has already been demonstrated sufficiently).

The output parameters are:

TABLE 5.18 OUTPUT PARAMETERS FOR GT EMISSION MODEL.

Output parameters	Entity
NO_x	ppm
CO	%
CO_2	ppm
NO	ppm
NO_2	mg/MJ

TABLE 5.19 PREDICTION ERRORS FOR THE GT EMISSION MODEL.

	NO_x	CO	CO_2	NO	NO_2
Average Error [%]	3.48	3.60	0.43	6.04	1.96
Maximum Error [%]	22.25	26.51	4.75	37.07	14.05

The results in table 5.19 show that the presumption of difficult predictability is correct. The CO_2 emissions are however predictable with reasonable small errors and demonstrated in figure 5.4. Also seen in the figure is that the measured signal is not updated every minute, hence the constant values, and can contribute to the prediction errors.

Tests with alternative input parameters, e.g. turbine outlet temperature and gas flow, have been carried out but the results have all been alike. The probable cause is that these parameters are predictable with the ambient conditions and are thereby not contributing with new information to the network.

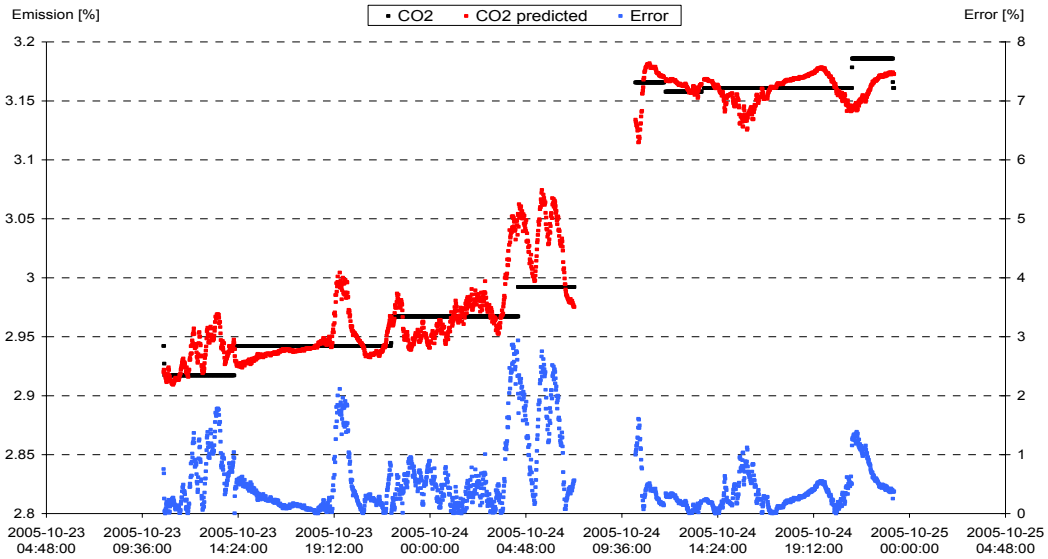

FIGURE 5.4 CO_2 PREDICTION FOR GT EMISSION MODEL.

5.2.4 Possible areas of implementation

In previous sections the performance for different ANNs are examined and evaluated. In this section, different possibilities regarding areas of application for the ANN models are discussed. As seen, some parameters are predicted with small errors and some are not, focus will be on the first category.

- Off-line modeling, i.e. when the GT is turned off. Considering that the ANN GT model only uses input parameters that are available at all time (i.e. ambient conditions), it is possible to predict e.g. power output, generated heat, gas flow and carbon dioxide emissions before start. This can prove helpful when evaluating if it is profitable or not starting the GT (with HRU). The power output and generated heat have direct economic dependencies since the power company is compensated per megawatt hour produced. The gas flow and CO_2 emissions also have economic dependencies since the power company disburses for consumed gas and produced CO_2. All compensation levels change continuously and deciding whether to start or not is rather complicated, so anything that facilitates this might prove useful. There is however lots of other considerations to take into account in the decision-making.

- On-line modeling, i.e. when the GT is in operation and automatically feeds the model with real-time data. Assuming the ANN model is trained with healthy GT data it is possible to compare the predicted values with the actual values and draw conclusions from that. For instance, if the power output decreases compared to the prediction it might intimate GT degradation and that the compressor needs washing (see 4.1.2). Comparably, decreased heat extraction could be a sign that the HRU is degraded (see 4.2.1). The model could also be used for fault diagnosis or an early warning system that indicates e.g. if a component is damaged. For more information about such a system Arriagada's *On the Analysis and Fault-Diagnosis Tools for Small-Scale Heat and Power Plants* [3] is recommended.

5.2.5 User interface

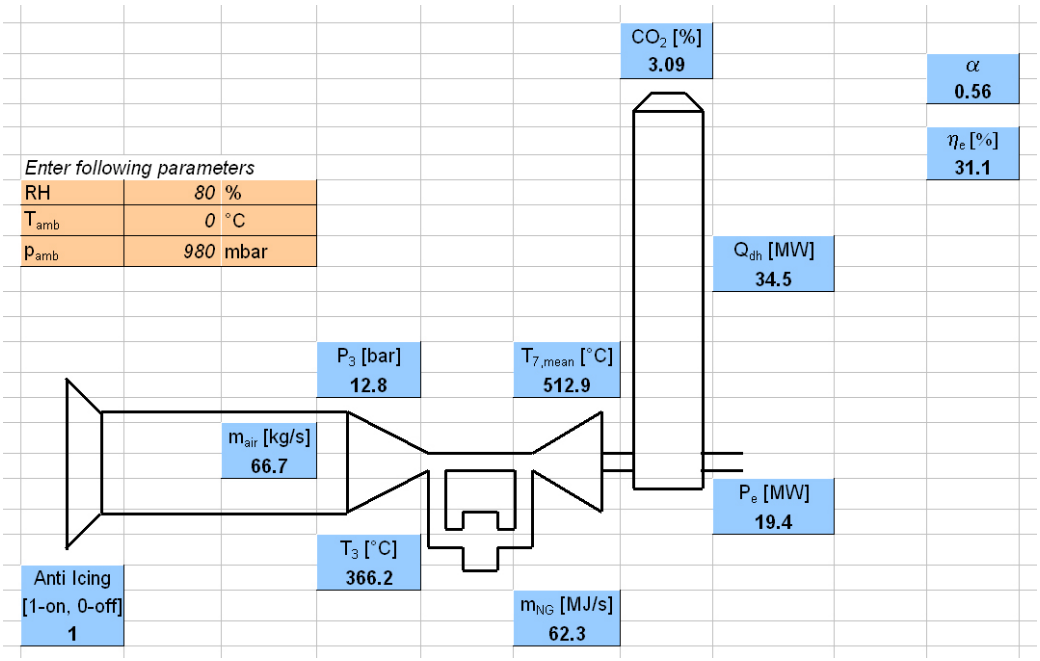

FIGURE 5.5 USER INTERFACE: GT MODEL.

A user interface has been created in an Excel environment. This is accomplished by extracting the ANN models into VB code and, after some modifications, implement them as functions in Excel (see appendix C). This creates an immediate advantage compared to other heat- and mass balance programs, e.g. IPSEPro, that requires special software and lots of knowledge to utilize. To make this product useful it is designed to be foreseeable and undemanding as seen figure 5.5. The entire Excel document is protected with a password except 3 orange cells, where the ambient conditions are to be typed in. The user will never see any calculations or formulas, only the results in the blue cells. The Excel sheet combines 2 ANN models, one that predicts if the anti icing system is in operation and another that predicts GT performance. Both ANNs use ambient conditions as input but the second one also uses the output from the first one, i.e. 0 or 1 (as described in section 5.2).

By presenting this simple, easy to use, model a lot of positive feedback was received from Lunds Energi. A first reaction was that now the personnel do not have to memorize the power output for a specific ambient temperature in order to know if the compressor needs washing. Furthermore the possibility of using such models for educational purposes was acknowledged.

5.2.6 Conclusions

ANN's potential has been acknowledged and there are numerous possible applications. As seen in previous sections the results for the GT model was impressive with high predictive performance. ANN could replace other expensive and complicated simulation programs. Once the basics of ANN is understood and the software is mastered ANNs are very easy to model and most importantly, anyone can utilize the final product.
Implementing ANN models could increase the reliability of the gas turbine and thereby the availability.

5.2.7 Future work

Three fields of future studies have been identified and are listed below.

- As a next step the user interface in section 5.2.5 could be integrated with the systems available at Lunds Energi and feed with real-time power plant data. By comparing and plotting the predictions and actual readings it would be possible to see e.g. GT degradation. Through further development of the model economic factors could be incorporated and determine whether e.g. a compressor wash (off-line) is profitable or not. As a whole the model would work as a monitoring system issuing warnings if the predicted value(s) separates from the actual reading(s).

- Furthermore it is desirable investigating whether it is possible to foresee the emissions by means of adding new sensors in the combustion chamber.

- There is also the possibility of creating an ANN model of the HRU using valve positions and flows and temperatures of the exhaust gas and the district heating system.

5.3 Sensor validation

Having reliable sensor readings is important for many reasons, e.g. to ensure that the control system receives correct information. If a sensor transmits a faulty signal to the control system it has no other option than to act accordingly, which can cause the GT to operate in an adverse way or even cause damage.

When working with data based systems such as ANNs, reliable sensor readings are of great importance. If corrupt input signals are presented to the ANN model the outputs from it will be worthless. Using sensor validation (SV) before the ANN model will ensure that the input signals are correct or at least reasonable. The main objective of the SV model is to filter out the faulty signal and replace it with a recovered value.

An ANN SV model has the same inputs as outputs and is called auto-associative network. The parameters chosen have to be correlated to each other, i.e. every output signal is dependent on every input signal and not only on itself or a limited group of signals. If a faulty input signal is presented to the network it has the capability of predicting a more accurate output value (i.e. recovered value), given that the remaining input signals are correct. If all input signals are accurate they will simply be passed thru the network without changing. Choosing appropriate number of parameters in a sensor validation model is a balance between higher predictability and earlier detection but also an increased hazard of encountering multiple sensor faults.

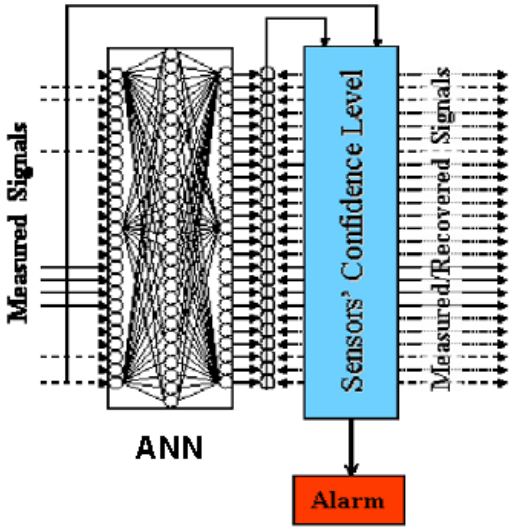

FIGURE 5.6 ANN FOR SENSOR VALIDATION [9].

The confidence level (CL) will be calculated for each reading:

$$CL_i = 1 - \frac{\left|\hat{S}_i - S_i^r\right|}{\hat{S}_i} \tag{5.1}$$

where $\hat{S}$ is the predicted value by ANN, S^r is the sensor reading and i is the sensor number. A CL close to 1 indicates a highly reliable sensor and a value close to zero describes a failed sensor [9]. The CL is used to detect which sensor is failing. Theoretically, looking at the sensors independently, it is possible to determine a specific CL and if the reading drops below this level the sensor is failing. Looking at all the sensors together will reveal that when one sensor is failing it will, because of their inter-dependencies, affect all the other outputs and the CL for all sensors will change. For small sensor errors it might seem like another sensor is failing. This is why the sensor that is most difficult to detect sets a common, lower CL (established through testing) and if any sensor drops below this level it can most definitely be determined as the failing one. If the sensor error is large several readings might drop below this level, in this case the sensor with the lowest CL is the failing sensor.

5.3.1 GT sensor validation

For this GT SV model 8 parameters, very similar to the output parameters in section 5.2.2, are chosen and can be seen in the CL plot in figure 5.7.

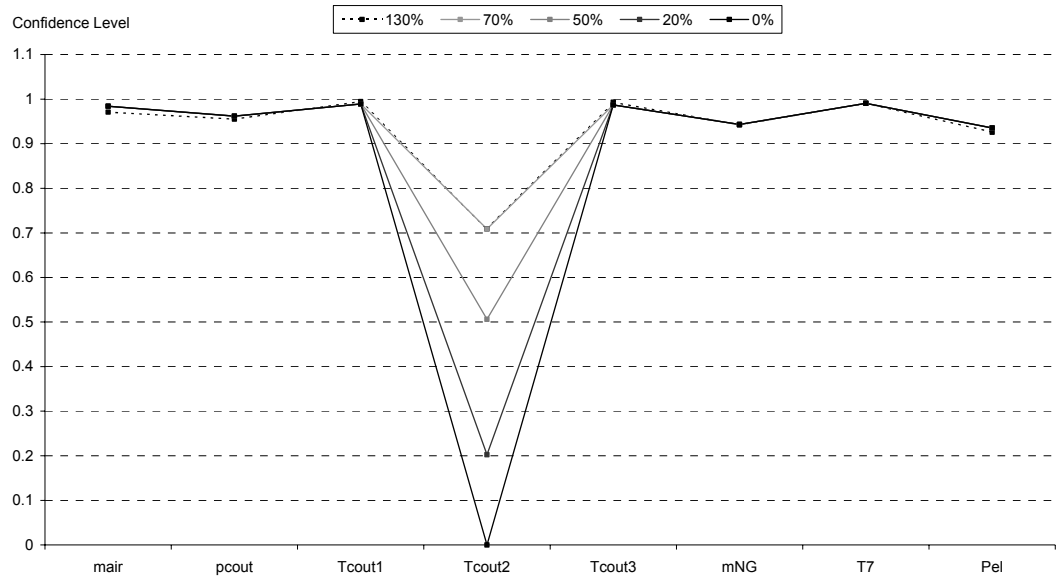

FIGURE 5.7 CONFIDENCE LEVEL PLOT WITH SENSOR $T_{COUT,2}$ FAILING.

TABLE 5.20 COMPARISON BETWEEN RECOVERED VALUE AND CORRECT READING WITH A FAULTY SENSOR.

	Recovered value when actual measurement is					
Sensor failed	Correct reading	0% reading	20% reading	50% reading	70% reading	130% reading
Airflow	67.00	63.61	63.61	63.63	63.68	70.03
Compressor outlet pressure	12.92	12.38	12.37	12.35	12.35	13.28
Compressor outlet temp. 1	365.64	361.04	361.03	361.03	361.03	368.03
Compressor outlet temp. 2	366.17	361.67	361.67	361.67	361.67	368.74
Compressor outlet temp. 3	366.87	361.68	361.68	361.68	361.68	369.56
Gas flow	63.38	59.69	59.69	59.68	59.67	67.40
Turbine outlet temperature	510.06	504.83	504.83	504.83	504.84	515.66
Power output	19.65	18.28	18.28	18.28	18.29	20.91

The figure above illustrates the confidence levels for the sensors when the reading from one specific sensor is 0% - 130% of its original signal. With errors of this size the CL clearly points out the faulty sensor and the SV model can generate an alarm. Before the sensor can be repaired the recovered value can be used temporarily. Table 5.20 shows the recovered values and it can be appreciated that they are fairly close to the correct values. An interesting remark is that the recovered values are roughly the same for 0%, 20%, 50% and 70%. This can be derived from the fact that the operating range, in which the parameters vary, is relatively narrow and the range, wherein data is normalized (see chapter 2.5.3), is set to [-0.8:0.8] so the network is not able to extrapolate any further (than seen in table 5.20). If the operating range had been broader the results would have looked differently. When normalizing the data it is of importance to investigate whether the data set used (for training) fully or partially covers the operating range. If the operating range is fully covered the normalization can be done close to [-1:1] (i.e. the transfer function boundaries). On the contrary, when the data partially covers the operating range, there is a need for extrapolation capability and the data is normalized within a smaller range to provide room for extrapolation. This can be summarized in the following conclusion: It is always preferable to have data covering the whole operating range because this facilitates the normalization of data and will improve the nets generalization capability since no extrapolation have to take place.

Given that the majority of prediction variation occurs before the sensor reaches a 70% fault (see table 5.20) it is of interest to have a closer look at this region.

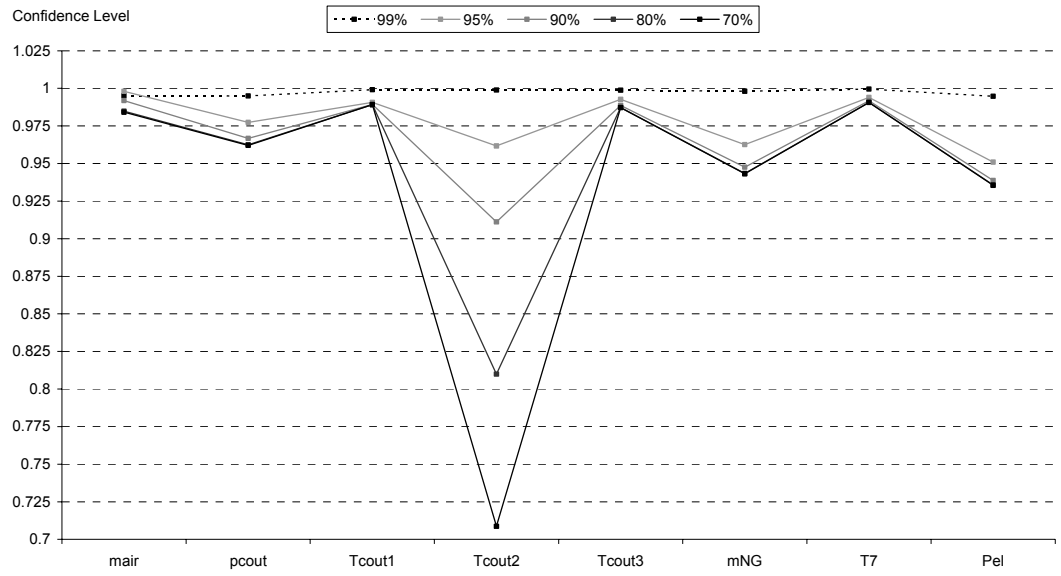

FIGURE 5.8 CONFIDENCE LEVEL PLOT WITH SENSOR $T_{COUT,2}$ FAILING.

The impact of having all parameters correlated to each other is that if one sensor fails all of the outputs are affected more or less. How much one specific output is affected depends solely on the significance the faulty input has on this output. A lower CL is established and if any sensor passes below this it can be determined as the faulty one. Figure 5.8 show that with small sensor errors it is hard to determine which sensor is failing (since all CLs are affected), but as the error increases it becomes clear which one it is. The recovered value for respective sensors can be seen in table 5.21.

TABLE 5.21 COMPARISON BETWEEN RECOVERED VALUE AND CORRECT READING WITH A FAULTY SENSOR.

Sensor failed	Recovered value when actual measurement is					
	Correct Reading	99% reading	95% reading	90% reading	80% reading	70% reading
Airflow	67.00	66.39	64.54	63.93	63.73	63.68
Compressor outlet pressure	12.92	12.88	12.66	12.47	12.36	12.35
Compressor outlet temp. 1	365.64	362.27	361.06	361.03	361.03	361.03
Compressor outlet temp. 2	366.17	362.90	361.70	361.68	361.67	361.67
Compressor outlet temp. 3	366.87	363.23	361.72	361.69	361.68	361.68
Gas flow	63.38	62.77	60.61	59.77	59.67	59.67
Turbine outlet temperature	510.06	505.89	504.87	504.86	504.86	504.84
Power output	19.65	19.49	18.88	18.49	18.32	18.29

5.3.1.1 Multiple sensor faults

For this specific case, with very small parameter variations, the prediction error will not increase after a certain value. This implies that even if another sensor fails the recovered value for the first sensor will remain at the same level and both faulty sensors will be detectable.

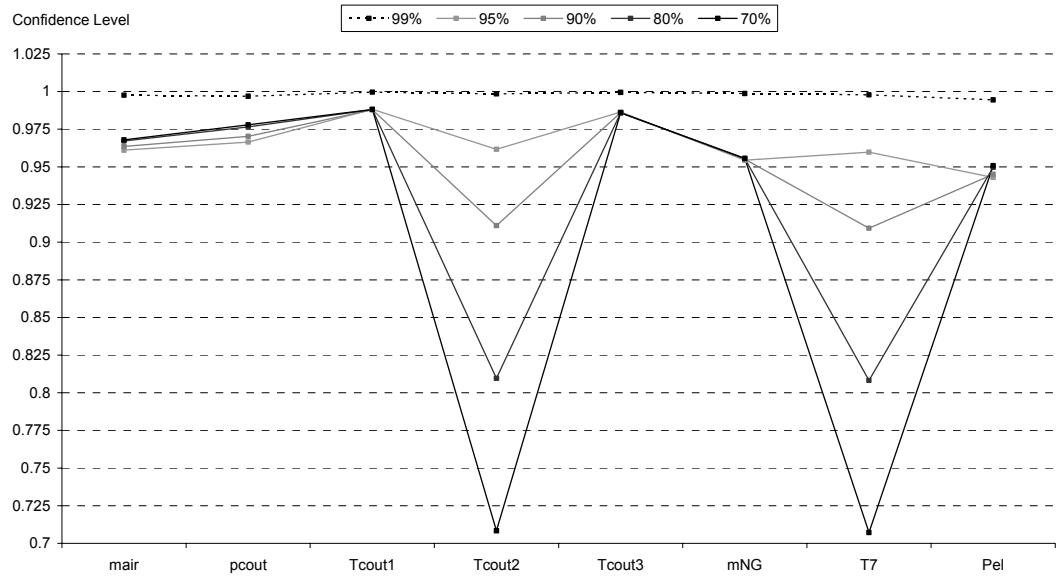

FIGURE 5.9 CONFIDENCE LEVEL PLOT WITH SENSORS $T_{COUT,2}$ AND T_7 FAILING.

Figure 5.9 illustrates that the CLs for the parameters with healthy sensors have changed compared with the single sensor fault scenario. This is the cause of two faulty input signals affecting all the outputs. The detection of faulty sensors is thereby delayed with multiple sensor faults and the recovered values are not as reliable as for single faults. Never the less, no matter how many sensors are failing this SV model will still produce acceptable outputs within (or close to) the operating range.

5.3.2 Turbine outlet temperature sensor validation

The turbine outlet temperature, T_7, is measured in 16 different positions (see figure 4.2) and from these an average value is calculated. In order to detect if one of these 16 sensors are failing a SV model is created using them as input/output parameters. Since the number of parameters has been doubled (compared to the model in section 5.3.1) it is expected to detect a faulty sensor at an earlier stage. This assumption can be made because every output depends on 16 inputs and thereby the contribution of one faulty sensor diminishes. Each one of the 16 signals are examined separately in order to find out at what level (% of original signal) a faulty sensor is detectable. Through these studies and by using an arbitrary input pattern (the results can vary depending on chosen input pattern) it was found that if the confidence level of any sensor drops below 0.99 it can be distinguished as failing. In this case it means that a sensor fault is discovered when the input signal reaches between 96%-99% (depending on which sensor it is) of the original signal. Figure 5.10 displays the confidence levels when sensor $T_{7,7}$ fails and table 5.22 shows the recovered values for a selected number of sensors.

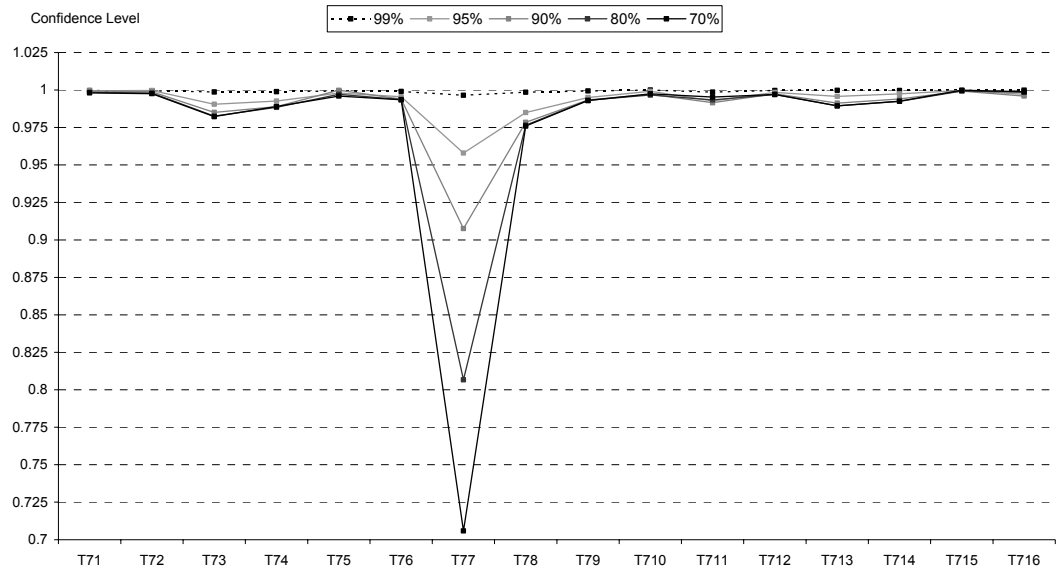

FIGURE 5.10 CONFIDENCE LEVEL PLOT WITH SENSOR $T_{7,7}$ FAILING.

TABLE 5.22 COMPARISON BETWEEN RECOVERED VALUE AND CORRECT READING WITH A FAULTY SENSOR.

	Recovered value when actual measurement is					
Sensor failed	Correct Reading	99% reading	95% reading	90% reading	80% reading	70% reading
Turbine outlet temperature 1	511.92	510.07	509.77	509.82	509.85	509.81
Turbine outlet temperature 2	498.21	494.74	493.85	493.82	493.80	493.80
Turbine outlet temperature 3	497.02	493.54	487.24	486.60	486.51	486.52
Turbine outlet temperature 4	514.36	509.76	505.10	504.97	504.94	504.93
Turbine outlet temperature 5	505.50	501.48	494.34	493.12	492.85	492.81

The results may look convincing and accurate but it is important to understand that since this specific GT only operates at full load the parameters, especially T_7, do not vary significantly. For instance, if T_7 only varies 2% during operation a detection of a

faulty sensor with e.g. 97% of its original value is suddenly not that impressive, On the other hand, no matter how corrupt an input signal may be, the SV model will still deliver a value within the operating range (or close to) and thereby allowing the GT to run without major disturbance.

5.3.3 User interface

No final product has been created but the general idea is to have a model, feed with real time data from the GT, that continuously predicts outputs and plots the CL for the sensors on a computer terminal visualized by a graphical interface to the operators in the control room. If a sensor is failing the CLs will digress from 1 and advise the operator through an alarm that something is happening, even if the failing sensor can not be discerned. One can even imagine that different levels of alarms are pre-established:

1st level: a non-distinguished sensor is starting to drive
2nd level: the driving sensor is detected and isolated
3rd level: the sensor drives too much and needs to be replaced (this can be adjusted to the planned maintenance intervals to avoid stop of the plant).

5.3.4 Conclusions

The results of these sensor validation studies can be questioned because of the fact that the parameter values do not vary significantly. This causes the recovered sensor readings (even for small errors) to quickly reach the outer boundaries wherein the GT operates. But even if it might not be possible to point out a faulty sensor, before this boundary is reached, the network will always provide a reasonable recovered value, within or close to these boundaries, and at the same time let the user know that something is wrong. Since the variations are so small it is not obvious that the user will notice a failing sensor without this kind of SV model.

The importance of proper data normalization is addressed and the effects it has on the network's extrapolation capability. This theory is not only applicable on SV models but on all ANN models. If data is normalized within the same boundaries as the transfer function uses (e.g. [-1:1]) the trained network will lack the capability of extrapolating outside of the training range.

5.3.5 Future work

A lot of time has been dedicated trying to improve the SV models and to detect a failing sensor at an earlier stage, listed below are some alternatives.

- One possible method could be training ANNs to recognize the specific CL patterns a failing sensor generates. The ANN SV model could be improved through re-training with data saved during the last incident of faulty sensor, which possibly has been saved in the operational computer of the plant.

- A method tried, but not fully evaluated, is instead of having all input parameters predict all output parameters in one ANN, as described in section 5.3, different ANNs are modeled for each output using all the other parameters (except itself) as input. Using the same parameters as in section 5.3.1 means that 8 separate ANNs have to be modeled. By doing so the recovered value for the failing sensor will be correct (results have proven this) whilst the other parameters are affected slightly more since neither of them depend on themselves anymore. An alternative is combining these 2 methods, using the second model to provide a correct value once the first model has established the failing sensor.

- Ensuring that the inputs to the ANN GT model (section 5.2.2) are reliable the model could serve as a SV model. Enhancing the reliability of the inputs is achieved by adding multiple sensors for each reading and using some kind of average value as input.

6 Summary of Results, Conclusions and Future Work

6.1 Gas turbine and emission model

The GT model has shown impressive results with very low prediction errors for all GT parameters. Since the GT always operates at full load only ambient conditions, which are always accessible, have been used as inputs to the ANN models. Power output, compressor outlet temperature and pressure are some of the parameters predicted with very good results.

From the beginning it was believed that the GT had to be modeled with two different ANNs (one with the anti icing system in operation and one without) but results have shown that by adding one extra input parameter, 1 or 0, for the anti icing system it allows the GT to be modeled with only one ANN. This was a big step and once again ANNs proves its versatility.

The GT model was trained with the ambient temperature varying from –4.5°C to 4°C and when it was presented with data outside of this interval it demonstrated good extrapolation capability.

Less success was achieved when predicting the emissions, with only CO_2 showing reasonable results. This however was expected since the airflow and gas flow are predictable with ambient conditions but combustion chamber specifics are not. CO_2 is dependent on the fuel/air ratio while e.g. NO_x is dependent on flame temperature. The fuel/air ratio determines the flame temperature in general but variations can still occur in different locations within the combustion chamber.

This work has contributed with a working model of the GT10B gas turbine at Lunds Energi complete with user interface applicable in Excel. The model has been validated and a sense of acceptance with the personnel has been won.

The objectives of this master thesis have been fulfilled, e.g. many different areas, where ANN models can be implemented, have been identified. An example is using the predicted power output, generated heat, gas flow and CO_2 emissions to facilitate the decision whether it is profitable or not to start the GT (with HRU).

The whole process has been a big learning experience, from receiving data from Lunds Energi, processing the data, identifying possibilities, training ANNs, creating a user interface and obtaining feedback.

The ANN GT model may well serve the same purpose as heat- and mass balance programs based on physical description of the components (e.g. IPSEPro among others commercial software). Advantages towards other programs are that ANNs are modeled relatively fast, for the experienced user, and the final product can be utilized by anyone.

One other immediate possibility for the GT model, recognized thru feedback from Lunds Energi, is to assess if the compressor needs washing (i.e. compressor degradation). This is accomplished by comparing the predicted power output with the actual reading. Another remark was that the model could serve well in educational purposes.

6.2 Sensor validation

Regarding the sensor validation part it has also shown interesting results and many conclusions have been drawn from them.

The sensor validation tests conducted with GT data have shown that it is possible to detect a failing sensor and at the same time produce a recovered value based on the healthy sensors.

The importance of choosing appropriate normalization boundaries for the data and the effects it has on the network's extrapolation capability has been addressed. For instance, if data used for training is normalized within the same boundaries as the transfer function uses, e.g. [-1,1], the network will lack extrapolation capability.

Furthermore the significance of establishing a common lower confidence level for all readings, in order to understand which one of the sensors is failing, is acknowledged. This means that no sensor can be determined as faulty before it reaches this lower confidence level, set by the sensor that, when faulty, is most difficult to detect.

6.3 Future work

Several fields of future studies have been identified and some of them are listed below. For more information, turn to sections 5.2.7 and 5.3.5.

6.3.1 Gas turbine model

- As a next step the GT model with user interface could be integrated with the systems available at Lunds Energi and feed with real-time power plant data. By comparing and plotting the predictions and actual readings it would be possible to see e.g. GT degradation. Through further development of the model economic factors could be incorporated and determine whether e.g. a compressor wash (off-line) is profitable or not. As a whole the model would work as a monitoring system issuing warnings if the predicted value(s) separates from the actual reading(s).

- Furthermore it is desirable investigating whether it is possible to foresee the emissions by means of adding new sensors in the combustion chamber.

6.3.2 Sensor validation

- One possible method could be training ANNs to recognize the specific CL patterns a failing sensor generates with help of data from saved sequences of sensor faults.

- A method tried, but not fully evaluated, is instead of having all input parameters predict all output parameters in one ANN, different ANNs are modeled for each output using all the other parameters (except itself) as input. By doing so the recovered value for the failing sensor will be correct (results have proven this) whilst the other parameters are affected slightly more since neither of them depend on themselves anymore. An alternative is combining these 2 methods, using the second model to provide a correct value once the first model has established the failing sensor.

Bibliography

[1] Alvarez, Henrik (2003): *Energiteknik*.

[2] Arriagada, Jaime (2001): *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.

[3] Arriagada, Jaime (2003): *On the Analysis and Fault-Diagnosis Tools for Small- Scale Heat and Power Plants*. Doctoral Thesis, Department of Heat and Power Engineering, Lund Institute of Technology, Lund University, Sweden.

[4] Arriagada, Jaime; Olausson, Pernilla (2000): *General Overview of the Artificial Neural Network Engineering for Energy System Applications*. Department of Heat and Power Engineering, Lund Institute of Technology, Lund University, Sweden.

[5] Diakunchak, I.S. (1992): *Performance Deterioration in Industrial Gas Turbines.* Gas and Steam Turbine Engineering, Power Generation Technology Division, Hamilton, Ontario, Canada.

[6] Energinet DK. Available: http://www.gastra.dk/dk/. [4th December 2005]

[7] Kröse, Ben; van der Smagt, Patrick (1996): *An Introduction to Neural Networks*.

[8] Mesbahi, Ehsan (2000): *Artificial Neural Networks for Fault Diagnosis, Modelling and Control of Diesel Engines*. Doctoral Thesis, Department of Marine Technology, University of Newcastle upon Tyne.

[9] E. Mesbahi, M. Genrup, M. Assadi (2005): *Fault prediction/diagnosis and sensor validation technique for a steam power plant.* Journal of Marine Engineering and Technology, No A7, 2005, ISSN 1476-1548, pp. 33-40.

[10] Principe, José C.; Euliano, Neil R.; Lefebvre, W. Curt (2000): *Neural and Adaptive Systems Fundamentals Through Simulations*.

[11] Rumelhart, D.E.; Hinton, G.E.; Williams R.J. (1986): *Learning Internal Representation by Error Propagation*.

[12] Saravanamuttoo, HIH; Rogers, GFC; Cohen, H (2001): *Gas Turbine Theory 5th Edition*.

Personal contacts:

[13] Arriagada, Jaime; jaime.arriagada@eon.se

[14] Assadi, Mohsen; mohsen.assadi@vok.lth.se

[15] Culjak, Vinko; vinko.culjak@lundsenergi.se

[16] Nilsson, Stefan; stefan.nilsson@lundsenergi.se

Appendix

The appendixes associated with this master thesis consist of a description of the settings for the models trained in chapter 5. Visual Basic codes for selected networks are also included.

The appendixes are:

- Appendix A: Settings for ANN GT models in section 5.2.
- Appendix B: Settings for ANN SV models in section 5.3.
- Appendix C: Visual Basic code for selected ANN models.

Appendix A

TABLE A.1 NETWORK SETTINGS FOR ANTI ICING MODEL.

Anti icing model (5.2.1)	
Hidden Neurons	8
Epochs	5000
Data sets (training/CV/testing)	60/15/25
Transfer function	tanh-sigmoid
Data normalization	-0.8:0.8

TABLE A.2 NETWORK SETTINGS FOR GT MODEL.

GT model (5.2.2)	
Hidden Neurons	7
Epochs	10000
Data sets (training/CV/testing)	60/15/25
Transfer function	tanh-sigmoid
Data normalization	-0.8:0.8

TABLE A.3 NETWORK SETTINGS FOR GT MODEL WITHOUT ANTI ICING.

GT without anti icing (5.2.2.2.1)	
Hidden Neurons	8
Epochs	10000
Data sets (training/CV/testing)	60/15/25
Transfer function	tanh-sigmoid
Data normalization	-0.8:0.8

TABLE A.4 NETWORK SETTINGS FOR GT MODEL WITH T_{FUEL}.

GT with T_{fuel} (5.2.2.2.2)	
Hidden Neurons	10
Epochs	10000
Data sets (training/CV/testing)	60/15/25
Transfer function	tanh-sigmoid
Data normalization	-0.8:0.8

TABLE A.5 NETWORK SETTINGS FOR GT MODEL WITHOUT P_{AMB}.

GT without p_{amb} (5.2.2.2.3)	
Hidden Neurons	10
Epochs	10000
Data sets (training/CV/testing)	60/15/25
Transfer function	tanh-sigmoid
Data normalization	-0.8:0.8

TABLE A.6 NETWORK SETTING FOR GT MODEL WITHOUT T_{AMB}.

GT without T_{amb} (5.2.2.2.3)	
Hidden Neurons	10
Epochs	10000
Data sets (training/CV/testing)	60/15/25
Transfer function	tanh-sigmoid
Data normalization	-0.8:0.8

TABLE A.7 NETWORK SETTINGS FOR GT MODEL WITHOUT RH.

GT without RH (5.2.2.2.3)	
Hidden Neurons	7
Epochs	10000
Data sets (training/CV/testing)	60/15/25
Transfer function	tanh-sigmoid
Data normalization	-0.8:0.8

TABLE A.8 NETWORK SETTING FOR GT EMISSIONS MODEL.

GT Emissions (5.2.3)	
Hidden Neurons	10
Epochs	5000
Data sets (training/CV/testing)	60/15/25
Transfer function	tanh-sigmoid
Data normalization	-0.8:0.8

Appendix B

TABLE B.1 NETWORK SETTINGS FOR GT SV MODEL.

GT SV (5.3.1)	
Hidden Neurons	9
Epochs	10000
Data sets (training/CV/testing)	60/15/25
Transfer function	tanh-sigmoid
Data normalization	-0.8:0.8

TABLE B.2 NETWORK SETTINGS FOR T_7 SV MODEL.

T_7 SV (5.3.2)	
Hidden Neurons	20
Epochs	10000
Data sets (training/CV/testing)	60/15/25
Transfer function	tanh-sigmoid
Data normalization	-0.8:0.8

Appendix C

The Visual Basic codes a seemingly complicated but a closer look is recommended in order improve understanding. The digits that the parameters are multiplied with are weights.

C.1 Visual Basic code for the ANN GT model

```
Function ANNGTmodel(AI As Double, RH As Double, pamb As Double, Tamb As Double) As Double()

Dim hidden1AxonPE1 As Double
hidden1AxonPE1 = tanh(((AI * 1.60000002384186 + -0.800000011920929) * 0.856087386608124) + ((RH * 5.99601902067661E-02 + -4.71483850479126) * -0.261825770139694) + ((pamb * 2.62618027627468E-02 + -25.374153137207) * 0.828892052173615) + ((Tamb * 0.187195107340813 + 4.07932139933109E-02) * -0.297290086746216) + -0.136315256357193)

Dim hidden1AxonPE2 As Double
hidden1AxonPE2 = tanh(((AI * 1.60000002384186 + -0.800000011920929) * -1.12067651748657) + ((RH * 5.99601902067661E-02 + -4.71483850479126) * -0.173274353146553) + ((pamb * 2.62618027627468E-02 + -25.374153137207) * -7.98191055655479E-02) + ((Tamb * 0.187195107340813 + 4.07932139933109E-02) * -0.956974565982819) + 0.106671445071697)

Dim hidden1AxonPE3 As Double
hidden1AxonPE3 = tanh(((AI * 1.60000002384186 + -0.800000011920929) * 1.44664859771728) + ((RH * 5.99601902067661E-02 + -4.71483850479126) * -0.429420620203018) + ((pamb * 2.62618027627468E-02 + -25.374153137207) * -0.867556214332581) + ((Tamb * 0.187195107340813 + 4.07932139933109E-02) * 0.527304351329803) + 1.84514725208282)

Dim hidden1AxonPE4 As Double
hidden1AxonPE4 = tanh(((AI * 1.60000002384186 + -0.800000011920929) * -0.773138582706451) + ((RH * 5.99601902067661E-02 + -4.71483850479126) * 7.55764469504356E-02) + ((pamb * 2.62618027627468E-02 + -25.374153137207) * 0.282254219055176) + ((Tamb * 0.187195107340813 + 4.07932139933109E-02) * -0.405942350625992) + -0.409856826066971)

Dim hidden1AxonPE5 As Double
hidden1AxonPE5 = tanh(((AI * 1.60000002384186 + -0.800000011920929) * 1.1805419921875) + ((RH * 5.99601902067661E-02 + -4.71483850479126) * -1.01674818992615) + ((pamb * 2.62618027627468E-02 + -25.374153137207) * -1.5696884393692) + ((Tamb * 0.187195107340813 + 4.07932139933109E-02) * -0.488603442907333) + -1.15312194824219)

Dim hidden1AxonPE6 As Double
hidden1AxonPE6 = tanh(((AI * 1.60000002384186 + -0.800000011920929) * -0.131374508142471) + ((RH * 5.99601902067661E-02 + -4.71483850479126) * -1.1653493642807) + ((pamb * 2.62618027627468E-02 + -25.374153137207) * -0.29467898607254) + ((Tamb * 0.187195107340813 + 4.07932139933109E-02) * -6.24055191874504E-02) + 1.08109045028687)

Dim hidden1AxonPE7 As Double
hidden1AxonPE7 = tanh(((AI * 1.60000002384186 + -0.800000011920929) * 1.45267856121063) + ((RH * 5.99601902067661E-02 + -4.71483850479126) * 0.178547412157059) + ((pamb * 2.62618027627468E-02 + -25.374153137207) * 0.343639612197876) + ((Tamb * 0.187195107340813 + 4.07932139933109E-02) * 1.90040695667267) + 1.90540206432343)

Dim outputAxonPE1 As Double

outputAxonPE1 = tanh((hidden1AxonPE1 * 0.309952020645142) + (hidden1AxonPE2 * 0.241807848215103) + (hidden1AxonPE3 * -0.349590629339218) + (hidden1AxonPE4 * 0.330598711967468) + (hidden1AxonPE5 * -0.309616386890411) + (hidden1AxonPE6 * 0.161656841635704) + (hidden1AxonPE7 * -1.36335892602801E-02) + -5.06226308643818E-02)

Dim outputAxonPE2 As Double

outputAxonPE2 = tanh((hidden1AxonPE1 * 0.334402859210968) + (hidden1AxonPE2 * 0.255088627338409) + (hidden1AxonPE3 * -0.379385650157929) + (hidden1AxonPE4 * 0.321410357952118) + (hidden1AxonPE5 * -0.319893270730972) + (hidden1AxonPE6 * 0.15807332098484) + (hidden1AxonPE7 * -4.41184686496854E-03) + -3.26674096286297E-02)

Dim outputAxonPE3 As Double

outputAxonPE3 = tanh((hidden1AxonPE1 * 0.50124990940094) + (hidden1AxonPE2 * 0.163747698068619) + (hidden1AxonPE3 * 0.359741926193237) + (hidden1AxonPE4 * 4.89916689693928E-02) + (hidden1AxonPE5 * 0.678045392036438) + (hidden1AxonPE6 * -0.751839399337769) + (hidden1AxonPE7 * 0.852939367294312) + 0.210933700203896)

Dim outputAxonPE4 As Double

outputAxonPE4 = tanh((hidden1AxonPE1 * 0.276841670274735) + (hidden1AxonPE2 * 0.560881853103638) + (hidden1AxonPE3 * -0.557926416397095) + (hidden1AxonPE4 * -0.216644659638405) + (hidden1AxonPE5 * -0.297554165124893) + (hidden1AxonPE6 * 5.76789230108261E-02) + (hidden1AxonPE7 * 1.95707064121962E-02) + 2.41462457925081E-02)

Dim outputAxonPE5 As Double

outputAxonPE5 = tanh((hidden1AxonPE1 * 0.109762564301491) + (hidden1AxonPE2 * -0.444078296422958) + (hidden1AxonPE3 * 0.020639643073082) + (hidden1AxonPE4 * -0.263317972421646) + (hidden1AxonPE5 * 9.79114845395088E-02) + (hidden1AxonPE6 * -5.40149360895157E-02) + (hidden1AxonPE7 * 0.337813079357147) + -8.36670100688934E-02)

Dim outputAxonPE6 As Double
```

```
outputAxonPE6 = tanh((hidden1AxonPE1 * 0.239147230982781) + (hidden1AxonPE2 * 0.53985857963562) + (hidden1AxonPE3 * -0.440748602151871) + (hidden1AxonPE4 * -0.085798978805542) + (hidden1AxonPE5 * -0.300403237342834) + (hidden1AxonPE6 * 0.119059436023235) + (hidden1AxonPE7 * -0.103935241699219) + -2.82940212637186E-02)

Dim outputAxonPE7 As Double

outputAxonPE7 = tanh((hidden1AxonPE1 * 0.227037310600281) + (hidden1AxonPE2 * -3.15963812172413E-02) + (hidden1AxonPE3 * -0.179051652550697) + (hidden1AxonPE4 * 0.565028727054596) + (hidden1AxonPE5 * -0.182256788015366) + (hidden1AxonPE6 * 9.96938943862915E-02) + (hidden1AxonPE7 * 1.91605389118195E-02) + 0.101101942360401)

mair = ((outputAxonPE1) - -10.1670608520508) / 0.148027956485748
pcout = ((outputAxonPE2) - -10.6906089782715) / 0.814041912555695
Tcout = ((outputAxonPE3) - -65.8213958740234) / 0.181393817067146
mng = ((outputAxonPE4) - -8.37121391296387) / 0.129603832960129
T7 = ((outputAxonPE5) - -56.3081817626953) / 0.111003182828426
Pel = ((outputAxonPE6) - -7.48312044143677) / 0.365363538265228
Qdh = ((outputAxonPE7) - -9.46672439575195) / 0.271301448345184

Dim temp(6) As Double
temp(0) = mair
temp(1) = pcout
temp(2) = Tcout
temp(3) = mng
temp(4) = T7
temp(5) = Pel
temp(6) = Qdh

ANNGTmodel = temp

End Function

Function tanh(x As Double) As Double
tanh = (Exp(x) - Exp(-x)) / (Exp(x) + Exp(-x))
End Function
```

C.2 Visual Basic code for the ANN GT SV model

```
Function SVGT(mair As Double, pcout As Double, Tcout1 As Double, Tcout2 As Double, Tcout3 As Double, mNG As Double, T7 As Double, Pel As Double) As Double()

Dim hidden1AxonPE1 As Double
hidden1AxonPE1 = tanh(((mair * 0.283942639827728 + -19.0240516662598) * 0.186793565750122) + ((pcout * 1.60965883731842 + -20.7742576599121) * -0.276158779859543) + ((Tcout1 * 0.285315930843353 + -104.006050109863) * -0.719080805778503) + ((Tcout2 * 0.282376259565353 + -103.124153137207) * 8.13071429729462E-02) + ((Tcout3 * 0.25128835439682 + -91.8735580444336) * 0.230839863419533) + ((mNG * 0.249221548438072 + -15.8253355026245) * 0.644202947616577) + ((T7 * 0.183783367276192 + -93.7759170532227) * 0.581934154033661) + ((Pel * 0.705902636051178 + -13.8697872161865) * -0.49749431014061) + 8.52500945329666E-02)

Dim hidden1AxonPE2 As Double
hidden1AxonPE2 = tanh(((mair * 0.283942639827728 + -19.0240516662598) * 0.885654091835022) + ((pcout * 1.60965883731842 + -20.7742576599121) * 0.131628468632698) + ((Tcout1 * 0.285315930843353 + -104.006050109863) * -0.338456898927689) + ((Tcout2 * 0.282376259565353 + -103.124153137207) * 0.150735124945641) + ((Tcout3 * 0.25128835439682 + -91.8735580444336) * -0.149701043963432) + ((mNG * 0.249221548438072 + -15.8253355026245) * -1.06168496608734) + ((T7 * 0.183783367276192 + -93.7759170532227) * -0.442954957485199) + ((Pel * 0.705902636051178 + -13.8697872161865) * -0.462285250425339) + 0.295386850833893)

Dim hidden1AxonPE3 As Double
hidden1AxonPE3 = tanh(((mair * 0.283942639827728 + -19.0240516662598) * -7.94500559568405E-02) + ((pcout * 1.60965883731842 + -20.7742576599121) * -0.208152160048485) + ((Tcout1 * 0.285315930843353 + -104.006050109863) * -0.227827876806259) + ((Tcout2 * 0.282376259565353 + -103.124153137207) * -0.244372978806496) + ((Tcout3 * 0.25128835439682 + -91.8735580444336) * -4.23047468066216E-02) + ((mNG * 0.249221548438072 + -15.8253355026245) * 1.40914416313171) + ((T7 * 0.183783367276192 + -93.7759170532227) * -0.625479161739349) + ((Pel * 0.705902636051178 + -13.8697872161865) * -6.03485703468323E-02) + 1.41242635250092)

Dim hidden1AxonPE4 As Double
hidden1AxonPE4 = tanh(((mair * 0.283942639827728 + -19.0240516662598) * 0.435048997402191) + ((pcout * 1.60965883731842 + -20.7742576599121) * 0.387996077537537) + ((Tcout1 * 0.285315930843353 + -104.006050109863) * 1.40017280355096E-02) + ((Tcout2 * 0.282376259565353 + -103.124153137207) * 0.745338320732117) + ((Tcout3 * 0.25128835439682 + -91.8735580444336) * -0.655076622962952) + ((mNG * 0.249221548438072 + -15.8253355026245) * -0.117332644760609) + ((T7 * 0.183783367276192 + -93.7759170532227) * 0.811586737632751) + ((Pel * 0.705902636051178 + -13.8697872161865) * 0.350319087505341) + 0.22914469242096)

Dim hidden1AxonPE5 As Double
hidden1AxonPE5 = tanh(((mair * 0.283942639827728 + -19.0240516662598) * -0.507831037044525) + ((pcout * 1.60965883731842 + -20.7742576599121) * 0.114923179149628) + ((Tcout1 * 0.285315930843353 + -104.006050109863) * -0.612038016319275) + ((Tcout2 * 0.282376259565353 + -103.124153137207) * 0.751324772834778) + ((Tcout3 * 0.25128835439682 + -91.8735580444336) * 0.656247735023499) + ((mNG * 0.249221548438072 + -15.8253355026245) * -0.179564371705055) + ((T7 * 0.183783367276192 + -93.7759170532227) * -0.385612189769745) + ((Pel * 0.705902636051178 + -13.8697872161865) * 0.696290850639343) + -0.269035398960114)

Dim hidden1AxonPE6 As Double
```

```
hidden1AxonPE6 = tanh(((mair * 0.283942639827728 + -19.0240516662598) * 0.420514643192291) + ((pcout * 1.60965883731842 + -20.7742576599121) * 7.71346613764763E-02) + ((Tcout1 * 0.285315930843353 + -104.006050109863) * -0.597773492336273) + ((Tcout2 * 0.282376259565353 + -103.124153137207) * -0.689587354660034) + ((Tcout3 * 0.25128835439682 + -91.8735580444336) * 0.166500806808472) + ((mNG * 0.249221548438072 + -15.8253355026245) * 1.10982775688171) + ((T7 * 0.183783367276192 + -93.7759170532227) * -9.67240631580353E-02) + ((Pel * 0.705902636051178 + -13.8697872161865) * -4.69693727791309E-02) + -2.45543432235718)

Dim hidden1AxonPE7 As Double
hidden1AxonPE7 = tanh(((mair * 0.283942639827728 + -19.0240516662598) * 0.910418212413788) + ((pcout * 1.60965883731842 + -20.7742576599121) * 0.235777229070664) + ((Tcout1 * 0.285315930843353 + -104.006050109863) * 0.931281566619873) + ((Tcout2 * 0.282376259565353 + -103.124153137207) * 0.277116507291794) + ((Tcout3 * 0.25128835439682 + -91.8735580444336) * 0.296788364648819) + ((mNG * 0.249221548438072 + -15.8253355026245) * 0.357601433992386) + ((T7 * 0.183783367276192 + -93.7759170532227) * -4.98955845832825E-02) + ((Pel * 0.705902636051178 + -13.8697872161865) * -0.15305195748806) + -0.186192944645882)

Dim hidden1AxonPE8 As Double
hidden1AxonPE8 = tanh(((mair * 0.283942639827728 + -19.0240516662598) * -0.157062873244286) + ((pcout * 1.60965883731842 + -20.7742576599121) * -0.378994852304459) + ((Tcout1 * 0.285315930843353 + -104.006050109863) * 0.104384206235409) + ((Tcout2 * 0.282376259565353 + -103.124153137207) * 0.600584626197815) + ((Tcout3 * 0.25128835439682 + -91.8735580444336) * -0.491678684949875) + ((mNG * 0.249221548438072 + -15.8253355026245) * 7.90735483169556E-02) + ((T7 * 0.183783367276192 + -93.7759170532227) * -0.764833152294159) + ((Pel * 0.705902636051178 + -13.8697872161865) * -0.652272462844849) + -0.4407799243927)

Dim hidden1AxonPE9 As Double
hidden1AxonPE9 = tanh(((mair * 0.283942639827728 + -19.0240516662598) * -2.81694307923317E-02) + ((pcout * 1.60965883731842 + -20.7742576599121) * 0.133533775806427) + ((Tcout1 * 0.285315930843353 + -104.006050109863) * 0.242000102996826) + ((Tcout2 * 0.282376259565353 + -103.124153137207) * 0.130913525819778) + ((Tcout3 * 0.25128835439682 + -91.8735580444336) * -0.05647998675704) + ((mNG * 0.249221548438072 + -15.8253355026245) * -1.12129139900207) + ((T7 * 0.183783367276192 + -93.7759170532227) * 0.618006527423859) + ((Pel * 0.705902636051178 + -13.8697872161865) * 1.77718177437782E-02) + 0.829651057720184)

Dim outputAxonPE1 As Double

outputAxonPE1 = tanh((hidden1AxonPE1 * 5.81776015460491E-02) + (hidden1AxonPE2 * 0.547296226024628) + (hidden1AxonPE3 * 0.45994570851326) + (hidden1AxonPE4 * 0.247182965278625) + (hidden1AxonPE5 * -9.30875241756439E-02) + (hidden1AxonPE6 * 0.324174761772156) + (hidden1AxonPE7 * 0.316172271966934) + (hidden1AxonPE8 * -0.337088912725449) + (hidden1AxonPE9 * -0.413593530654907) + -0.140029683709145)

Dim outputAxonPE2 As Double

outputAxonPE2 = tanh((hidden1AxonPE1 * -0.309754431247711) + (hidden1AxonPE2 * 0.188143268227577) + (hidden1AxonPE3 * 0.453961789608002) + (hidden1AxonPE4 * 0.405017226934433) + (hidden1AxonPE5 * 0.128594219684601) + (hidden1AxonPE6 * 0.392681956291199) + (hidden1AxonPE7 * 7.59483575820923E-02) + (hidden1AxonPE8 * -0.545533061027527) + (hidden1AxonPE9 * -0.390015959739685) + -4.34156805276871E-02)

Dim outputAxonPE3 As Double

outputAxonPE3 = tanh((hidden1AxonPE1 * -0.286762207746506) + (hidden1AxonPE2 * -0.516585052013397) + (hidden1AxonPE3 * -0.696186661720276) + (hidden1AxonPE4 * -0.254060328006744) + (hidden1AxonPE5 * -0.254992365837097) + (hidden1AxonPE6 * -0.673208057880402) + (hidden1AxonPE7 * 0.573885023593903) + (hidden1AxonPE8 * 0.304716885089874) + (hidden1AxonPE9 * 0.296641111373901) + 0.136229932308197)

Dim outputAxonPE4 As Double

outputAxonPE4 = tanh((hidden1AxonPE1 * 0.303903192281723) + (hidden1AxonPE2 * -0.111233025789261) + (hidden1AxonPE3 * -0.472114950418472) + (hidden1AxonPE4 * 0.37867745757103) + (hidden1AxonPE5 * 0.521689772605896) + (hidden1AxonPE6 * -0.524068236351013) + (hidden1AxonPE7 * 0.340634018182755) + (hidden1AxonPE8 * 0.758858799934387) + (hidden1AxonPE9 * 0.197478398680687) + 0.197895005345344)

Dim outputAxonPE5 As Double

outputAxonPE5 = tanh((hidden1AxonPE1 * 0.524922132492065) + (hidden1AxonPE2 * -0.108760222792625) + (hidden1AxonPE3 * -0.622990190982819) + (hidden1AxonPE4 * -0.799295902252197) + (hidden1AxonPE5 * 0.503470063209534) + (hidden1AxonPE6 * -0.18641559779644) + (hidden1AxonPE7 * 0.629288077354431) + (hidden1AxonPE8 * -0.285932332277298) + (hidden1AxonPE9 * 0.392000824213028) + 0.394834876060486)

Dim outputAxonPE6 As Double

outputAxonPE6 = tanh((hidden1AxonPE1 * 9.30584296584129E-02) + (hidden1AxonPE2 * -0.328386634588242) + (hidden1AxonPE3 * 0.712774872779846) + (hidden1AxonPE4 * 0.212189048528671) + (hidden1AxonPE5 * -0.010170585475862) + (hidden1AxonPE6 * 0.290061414241791) + (hidden1AxonPE7 * 0.12847700715065) + (hidden1AxonPE8 * -3.66333574056625E-02) + (hidden1AxonPE9 * -0.579593062400818) + 7.93368965387344E-02)

Dim outputAxonPE7 As Double

outputAxonPE7 = tanh((hidden1AxonPE1 * 0.55752694606781) + (hidden1AxonPE2 * -0.488017439842224) + (hidden1AxonPE3 * -0.63411945104599) + (hidden1AxonPE4 * 0.23355121910572) + (hidden1AxonPE5 * -0.171845778822899) + (hidden1AxonPE6 * -0.256543934345245) + (hidden1AxonPE7 * 3.97139927372336E-03) + (hidden1AxonPE8 * -0.179315105080605) + (hidden1AxonPE9 * 0.559850692749023) + -0.126849263906479)

Dim outputAxonPE8 As Double

outputAxonPE8 = tanh((hidden1AxonPE1 * -0.57448273897171) + (hidden1AxonPE2 * -5.04628084599972E-02) + (hidden1AxonPE3 * 0.496002405881882) + (hidden1AxonPE4 * 0.383888125419617) + (hidden1AxonPE5 * 0.243846669793129) + (hidden1AxonPE6 *
```

```
0.365563720464706) + (hidden1AxonPE7 * -0.163351893424988) + (hidden1AxonPE8 * -0.464382767677307) + (hidden1AxonPE9 * -
0.425772696733475) + 2.08311229944229E-02)

Omair = ((outputAxonPE1) - -19.0240516662598) / 0.283942639827728
Opcout = ((outputAxonPE2) - -20.7742576599121) / 1.60965883731842
OTcout1 = ((outputAxonPE3) - -104.006050109863) / 0.285315930843353
OTcout2 = ((outputAxonPE4) - -103.124153137207) / 0.282376259565353
OTcout3 = ((outputAxonPE5) - -91.8735580444336) / 0.25128835439682
OmNG = ((outputAxonPE6) - -15.8253355026245) / 0.249221548438072
OT7 = ((outputAxonPE7) - -93.7759170532227) / 0.183783367276192
OPel = ((outputAxonPE8) - -13.8697872161865) / 0.705902636051178

Dim temp(7) As Double

temp(0) = Omair
temp(1) = Opcout
temp(2) = OTcout1
temp(3) = OTcout2
temp(4) = OTcout3
temp(5) = OmNG
temp(6) = OT7
temp(7) = OPel

SVGT = temp

End Function

Function tanh(x As Double) As Double
tanh = (Exp(x) - Exp(-x)) / (Exp(x) + Exp(-x))
End Function
```