

SURROGATE MODELING TECHNIQUES APPLIED TO ENERGY SYSTEMS

Anna Ściążko



UNIVERSITY OF ICELAND



**University
of Akureyri**



SURROGATE MODELING TECHNIQUES APPLIED TO ENERGY SYSTEMS

The Master Thesis was supported by a grant from Iceland, Liechtenstein and Norway through the EEA Financial Mechanism - Project PL0460.

Anna Ściążko

A 30 ECTS credit units Master's thesis

Supervisors

Dr. Francois Marechal

PhD student Mr. Matthias Dubuis

Dr. Adam Roman



A Master's thesis done at
RES | The School for Renewable Energy Science
in affiliation with
University of Iceland &
University of Akureyri

Akureyri, January 2011

Surrogate Modeling Techniques Applied To Energy Systems

A 30 ECTS credit units Master's thesis

© Anna Ściążko, 2011

RES | The School for Renewable Energy Science

Solborg at Nordurslod

IS600 Akureyri, Iceland

Telephone + 354 464 0100

www.res.is

Printed in 2011

at Stell Printing in Akureyri, Iceland

ABSTRACT

This Master Thesis investigates the possibility of using the surrogate modeling techniques in the complex energy system modeling problems. The different mathematical (nearest-neighbor, linear, spline, cubic Hermite and polynomial interpolation as well as polynomial fitting) and artificial intelligence (neural network) methods were introduced and implemented.

Prepared paper includes information from many different fields of science: energy science, mathematics and computer science. All of them were used to prepare multi-science analysis of surrogate modeling problem.

Several different surrogate models were created for two different energy systems. First of them was gas turbine with recirculation, syngas production and CO₂ capture; second system was the steam network. The energy system descriptions were provided in the Thesis.

Implemented models were analyzed and their errors were found. The results helped in generalizing the features of surrogate models of each type. The conclusion are focused on the following topics: methods of choosing the best initial sets of points, method of elimination errors in the training process, validity space of the model and possibilities of connecting two algorithms: genetic optimization and creation of surrogate model. Gathered information will be used in the second part of the project – implementing the generic tool to create surrogate models for any energy system model. The second part of the project will be realized in Poland at Jagiellonian University in cooperation with École Polytechnique Fédérale De Lausanne. In the Thesis, the process of creating the automotive computer tool is designed. Moreover the possible positive influence, not only for Poland, but for any problem examined by this tool, is described.

PREFACE

Today many problems are solved with computer simulations. They help us to model very complicated systems in relatively short time. Run so complex analysis and optimization would not be possible in any other way. But nowadays we need better and better solutions which allows for working with bigger speed and effectiveness. One of method to achieve that is to use better computers, but it is not always possible.

Other way of finding the more effective way of computer computations is introducing the surrogate models of complex energy systems. In this paper a several surrogate modeling techniques are investigated and implemented in the real life problem.

The Thesis is good example of merging a various fields of science (energy science, mathematics and computer science) in order to obtain the best results.

All of the analyzed problems are strictly connected with energy science, but the methods used to model and analyze them are more connected with terms as interpolation, error analysis or artificial intelligence, neural networks, genetic algorithms and surrogate models. This paper shows how important it is to find a connection between seemingly unrelated brands of science.

TABLE OF CONTENTS

1	Theory part.....	11
1.1	Modeling and optimization techniques.....	11
1.1.1	Motivation.....	11
1.1.2	Energy integration	11
1.1.3	Optimization.....	12
1.2	Surrogate models – introduction	16
1.3	Models evaluation methods	16
1.3.1	Norm of residual.....	17
1.3.2	Correlation coefficient r	17
1.3.3	Mean square error.....	18
1.4	Mathematical models.....	18
1.4.1	Interpolation problem definition	18
1.4.2	Nearest-neighbor interpolation.....	19
1.4.3	Linear interpolation	20
1.4.4	Polynomial interpolation.....	22
1.4.5	Spline interpolation	26
1.4.6	Cubic Hermite interpolation	28
1.4.7	Polynom fitting.....	30
1.5	Artificial intelligence models.....	32
1.6	Neural networks	32
1.6.1	Introduction.....	32
1.6.2	Simple neuron	33
1.6.3	Transfer functions.....	34
1.6.4	Neuron with vector input	35
1.6.5	Multilayer neural network	35
1.6.6	Back propagation algorithm.....	37
1.7	Genetic programming	38
2	One Decision variable model.....	40
2.1	Flow sheet and model description.....	40
2.2	Problem description.....	44
2.3	Modeling problem description	45

2.4 Results.....	50
2.4.1 Results for one output (times, errors, graphs).....	50
2.4.2 Results for selected methods for all output parameters.....	67
2.4.3 Interpretation of results.....	73
3 Multi decision variable model.....	75
3.1 Flow sheet and model description.....	75
3.2 Problem description.....	80
3.3 Modeling problem description.....	80
3.4 Results.....	84
4 Generalization.....	103
4.1 Points choosing.....	103
4.2 Model training methods – errors elimination.....	104
4.3 Surrogate model errors – validity.....	105
4.4 Connection with optimization genetic algorithm.....	105
5 Surrogate model – project developing in Poland.....	107
6 Conclusions.....	109
References.....	111
Appendix A.....	1

LIST OF FIGURES

Figure 1.1 Pinch point technology – rules.....	12
Figure 1.2 Genetic algorithm schema (Genetic algorithms - Introduction)	13
<i>Figure 1.3 Crossover operator in genetic algorithm.....</i>	<i>14</i>
Figure 1.4 Mutation operator in genetic algorithm.....	14
Figure 1.5 Mutation types (Stranz & Martin, 1997)	15
Figure 1.6 The role of crossover and mutation operators in genetic algorithm (Genetic Algorithms : General Idea)	15
Figure 1.7 Nearest-neighbour interpolation of function $\cos(x)$ based on 10 equidistant nodes in the interval $[0,10]$	20
Figure 1.8 Linear interpolation between two given points P_1 and P_2	21
Figure 1.9 Linear interpolation of function $\cos(x)$ based on 10 equidistant nodes in the interval $[0,10]$	22
Figure 1.10 Interpolation polynoms of function $f(x) = \max(0, 1 - x)$ in interval $[-4,4]$ based on different types of nodes: equidistant(A,C) and Chebyshev(B,D) and different number of nodes: 5 nodes (A, B) and 33 nodes (C,D).	25
Figure 1.11 Cubic spline interpolation of function $\cos(x)$ based on 10 equidistant nodes in the interval $[0,10]$	28
Figure 1.12 Cubic Hermite interpolation of function $\cos(x)$ based on 10 equidistant nodes in the interval $[0,10]$	29
Figure 1.13 Cubic Spline Interpolation (spline) versus Cubic Hermite Interpolation (pchip) (MATLAB documentation, pchip command)	30
Figure 1.14 Polynomial fitting for function $\cos x$ - red line – in the interval $[0,12]$ by polynoms of different degree: blue – first degree, green – second degree, yellow – third degree, violet – fourth degree.	31
Figure 1.15 Simple neuron (Beale, Hagan, & Demuth, 2010).....	33
Figure 1.16 Transfer functions used in Neural Network (Beale, Hagan, & Demuth, 2010).....	34
Figure 1.17 Neuron with vector input (Beale, Hagan, & Demuth, 2010).....	35
Figure 1.18 Neuron with vector input: abbreviated notation (Beale, Hagan, & Demuth, 2010).....	35
Figure 1.19 Simple representation of multilayer Network (Stergiou & Siganos).....	36
Figure 1.20 One layer of a neural network in normal and abbreviated notation (Beale, Hagan, & Demuth, 2010)	36
Figure 1.21 Multilayer network (Beale, Hagan, & Demuth, 2010).....	37
Figure 1.22 Approximation network with one hidden layer(Beale, Hagan, & Demuth, 2010).....	37

Figure 1.23 Genetic programs in tree structure for one decision variable model (Sreekanth & Datta, 2010).....	39
Figure 2.1 Gas turbine with recirculation schema – GT_SEQ.....	40
Figure 2.2 The amount of H ₂ to be added to improve flame stability.....	41
Figure 2.3 Syngas production schema – REACT	42
Figure 2.4 CO ₂ capture black box model for chemical absorption with monoethanolamines (Dubuis & Tock, 2010).....	43
Figure 2.5 Flow-sheeting model which can be use to simulate multi-pressure, split-fraction CO 2 capture (Bernier, Marechal, & Samson, 2009).....	43
Figure 2.6 The multi-objective problem (gas turbine power plant with CO ₂ capture with Flue Gas Recirculation). A - program schema, B simplified problem.....	44
Figure 2.7 Nearest-neighbor interpolation for different number of nodes for Power Output	51
Figure 2.8 Linear interpolation for different number of nodes for Power Output.....	52
Figure 2.9 Spline interpolation for different number of nodes for Power Output.....	53
Figure 2.10 Cubic Hermite interpolation for different number of nodes for Power Output	54
Figure 2.11 Polynomial interpolation for different number of nodes (and different degree of interpolation polynom) for Power Output	56
Figure 2.12 Polynomial fitting of polynom degree 5 for different number of nodes for Power Output	57
Figure 2.13 Polynomial fitting of polynom degree 10 for different number of nodes for Power Output	58
Figure 2.14 Design of the neural Network with 2 neurons (experiment 1,2,3) and with 10 neurons (experiment 4) in hidden layer (from MATLAB interface)	59
Figure 2.15 Fit graph for Experiment 1	60
Figure 2.16 Regression graph for Experiment 1.....	61
Figure 2.17 Performance and Training State graphs for Experiment 1	62
Figure 2.18 Fit graph for Experiment 2	62
Figure 2.19 Regression graph for Experiment 2.....	63
Figure 2.20 Performance and Training State graphs for Experiment 2	64
Figure 2.21 Fit graph for Experiment 3	64
Figure 2.22 Regression graph for Experiment 3.....	65
Figure 2.23 Performance and Training State graphs for Experiment 3	65
Figure 2.24 Fit graph for Experiment 4	66
Figure 2.25 Regression graph for Experiment 4.....	66
Figure 2.26 Performance and Training State graphs for Experiment 4	67
Figure 2.27 Surrogate model on basis of 4 knots for F_421_T.....	68
Figure 2.28 Surrogate model on basis of 6 knots for F_421_T.....	68

Figure 2.29 Surrogate model on basis of 11 knots for F_421_T.....	69
Figure 2.30 R values for spline surrogate model based on 4 knots for all output parameters (ordered from smallest one).....	69
Figure 2.31 R values for spline surrogate model based on 6 knots for all output parameters(ordered from smallest one).....	70
Figure 2.32 R values for spline surrogate model based on 11 knots for all output parameters(ordered from smallest one).....	70
Figure 2.33 Design of the neural Network for one input, many output problem (from MATLAB interface).....	71
Figure 2.34 Regression graph for one input, many output problem.....	72
Figure 2.35 Performance and Training State graphs for one input, many output problem.....	72
Figure 2.36 Regression graph for additional test set.....	73
Figure 3.1 Superstructure of gas turbine power plant with CO 2 capture (Dubuis & Tock, 2010).....	75
Figure 3.2 Superstructure of a steam network with one expansion level and heat consumption and rejection (Girardin, Dubuis, & Marechal).....	76
Figure 3.3 Bad example of Steam Network Configuration (Dubuis & Tock, 2010).....	78
Figure 3.4 Good example of Steam Network Configuration (Dubuis & Tock, 2010)	79
Figure 3.5 Pareto Curve for Power Output – Investment Cost for different configurations of Steam Network (Girardin, Dubuis, & Marechal).....	79
Figure 3.6 Pareto Curve for Power Output – Levelized Electricity Cost for different configurations of Steam Network (Girardin, Dubuis, & Marechal)	80
Figure 3.7 Schema of data preparation for surrogate model creation.....	82
Figure 3.8 The schema of surrogate model.....	83
Figure 3.9 Confusion matrix for neural network surrogate models – classifying part – bigger testing set	86
Figure 3.10 Performance and Training State graphs for neural network surrogate models – classifying part – bigger testing set.....	87
Figure 3.11 Regression graph for neural network surrogate models – power fitting part – bigger testing set	88
Figure 3.12 Performance and Training State graphs for neural network surrogate models – power fitting part – bigger testing set.....	89
Figure 3.13 Regression graph for neural network surrogate models – cost fitting part – bigger testing set	90
Figure 3.14 Performance and Training State graphs for neural network surrogate models – cost fitting part – bigger testing set.....	90
Figure 3.15 Regression graph for neural network surrogate models – power fitting part – bigger testing set, version 2	92
Figure 3.16 Regression graph for neural network surrogate models – cost fitting part – bigger testing set, version 2, training	93

Figure 3.17 Regression graph for neural network surrogate models – cost fitting part – bigger testing set, version 2, retraining 1	94
Figure 3.18 Regression graph for neural network surrogate models – cost fitting part – bigger testing set, version 2, retraining 2	95
Figure 3.19 Confusion matrix for neural network surrogate models – classifying part – smaller testing set.....	96
Figure 3.20 Regression graph for neural network surrogate models – power fitting part – smaller testing set.....	97
Figure 3.21 Regression graph for neural network surrogate models – cost fitting part – smaller testing set.....	98
Figure 3.22 Confusion matrix for neural network surrogate models – classifying part – complex problem.....	100
Figure 3.23 Regression graph for neural network surrogate models – power fitting part – complex case; 25 neurons in the hidden layer	101
Figure 3.24 Regression graph for neural network surrogate models – power fitting part – complex case; 25 neurons in the hidden layer	102
Figure 4.1 Surrogate modeling with optimization algorithm (Sreekanth & Datta, 2010) .	106

LIST OF TABLES

Table 1.1 Rest of interpolations for different interpolation conditions.....	26
Table 1.2 Comparison of brain and computer (Orr, 1999).....	32
Table 2.1 All input and output variables in the surrogate model with example of values...45	
Table 2.2 Properties of different neural network surrogate models	59
Table 2.3 Neural network surrogate models: MSE and Regression R values.....	60
Table 2.4 Properties of neural network surrogate models for one input, many output problem.....	71
Table 2.5 Neural network surrogate models for one input, many output problem: MSE and Regression R values	71
Table 3.1 Input parameters of the simple Steam Network model	81
Table 3.2 Input parameters of the simple Steam Network model	84
Table 3.3 Properties of neural network surrogate models – classifying part – bigger testing set	85
Table 3.4 Neural network surrogate models – classifying part – bigger testing set: MSE and Regression R values.....	85
Table 3.5 Properties of neural network surrogate models – power fitting part – bigger testing set	87
Table 3.6 Neural network surrogate models – power fitting part – bigger testing set: MSE and Regression R values.....	87
Table 3.7 Properties of neural network surrogate models – cost fitting part – bigger testing set	89
Table 3.8 Neural network surrogate models –cost fitting part – bigger testing set: MSE and Regression R values	89
Table 3.9 Properties of neural network surrogate models – power fitting part – bigger testing set, version 2	91
Table 3.10 Neural network surrogate models – power fitting part – bigger testing set, version 2: MSE and Regression R values.....	91
Table 3.11 Properties of neural network surrogate models – cost fitting part – bigger testing set, version 2	92
Table 3.12 Neural network surrogate models – cost fitting part – bigger testing set, version 2: MSE and Regression R values.....	92
Table 3.13 Properties of neural network surrogate models – classifying part – smaller testing set	95
Table 3.14 Neural network surrogate models – classifying part – smaller testing set: MSE and Regression R values.....	96

Table 3.15 Properties of neural network surrogate models – power fitting part – smaller testing set	96
Table 3.16 Neural network surrogate models – power fitting part – smaller testing set: MSE and Regression R values	97
Table 3.17 Properties of neural network surrogate models – cost fitting part – smaller testing set	98
Table 3.18 Neural network surrogate models – cost fitting part – smaller testing set: MSE and Regression R values.....	98
Table 3.19 Properties of neural network surrogate models – classifying part – complex problem.....	99
Table 3.20 Neural network surrogate models – classifying part – complex problem: MSE and Regression R values.....	99

1 THEORY PART

In the first part of the Master Thesis several important information used in the practical problems are introduced. In this chapter several mathematical and artificial intelligence techniques connected with modeling problems are presented. This work lies on the crossroads of different disciplines. Presented topics are connected with energy science, computer science and mathematics.

1.1 Modeling and optimization techniques

There are many modeling and optimization techniques used in energy engineering software. To introduce some of them: the different methods of building the models and solving them, energy integration techniques or optimization algorithms. Some of those methods are described below. The greatest emphasis is placed on the genetic optimization algorithms, because they are used in the existing software and they can be correlated with the surrogate modelling algorithms.

1.1.1 Motivation

In many cases we need computer models to solve complicated problems in the topic of energy engineering. Real model preparation is not always possible because of time, money and physical possibilities.

Moreover in many cases we consider very complex systems. There is huge number of possibilities for designing such problem. Often we have several competitive objective functions in the system (like for example cost of the system and its environmental influence). Computer models let for many problem evaluations. We can use them in the optimization problems to find the best system design.

1.1.2 Energy integration

The energy integration technique is also called the pinch technology or heat integration. This method helps in optimal designing of energy systems. It helps in minimizing energy consumption and maximizing the internal heat recovery. The detailed description of this methodology can be found for example in (Linhoff & Townsend, 1982) and (Kemp, 2007). In the theory part only the main idea of pinch point technology is shown.

The pinch point technology allows for calculating the thermodynamically attainable energy targets for a process and helps in identification methods to achieve them.

One of the most important things is pinch temperature, which is the most constrained point in the process. There are three most important rules of pinch point technology presented in the Figure 1.1:

- Do not transfer heat through pinch point;
- Do not cool the process above the pinch point;
- Do not heat the process below the pinch point.

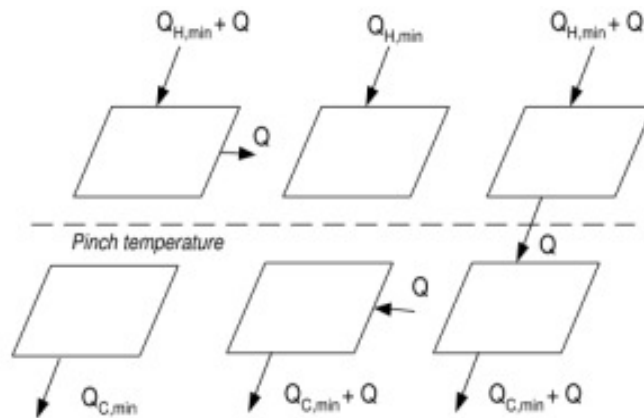


Figure 1.1 Pinch point technology – rules

1.1.3 Optimization

Optimization is the method of finding the best solution for given problem. Problem is evaluated by special objective functions. Optimization can be understood as searching the function extremums (minimums or maximums, depending of problem formulation). There is two type of optimization: mono-objective or multi-objectives. Multi-objective optimization helps in finding the best decision in case of several competing targets (for example it can be finding a compromise between production cost and quality). Optimization results can be shown in the Pareto curve, which shows the connection between different objective functions.

To optimize a function we can use many different algorithms. One of them is genetic algorithm. We will introduce a basic theory about this method. Genetic algorithm can be also used in the building a surrogate model (this concept will be explain in the separate chapter). Genetic algorithms are used in the LENI software (MOO – the tool implementing multi-objective optimization). Good understanding of genetic algorithm is especially important - the next step of this Master Thesis will be incorporating surrogate modeling into the LENI software and combination surrogate models with optimization algorithms (to make the process of learning the surrogate model more efficient).

The genetic and evolutionary algorithms were invented by John Holland in 60. and 70.XX. These kinds of algorithms try to mimetic the natural evolution process and to solve problems in the similar way as evolution do. They use the biological evolutionary mechanisms, such as natural selection, survival of the fittest, inheritance, reproduction and mutation. They can be understood as the compromise between the stochastic solution searching and working on the basis of previous results. There is many publications on this subject, more details can be found in (Goldberg, 1989).

Basic concepts connected with genetic algorithms are:

- Method of saving the parameters of each individual – chromosome;
- Method to rating the individuals - the fitness function ;
- Random selection of initial population;
- Selection operator;

- Crossover operator;
- Mutation operator.

The basic genetic algorithm schema is shown in the Figure 1.2. Of course there is many methods to improve the basic algorithm, but the most important stages are following: first step is initialization of the population of chromosomes - they are generated randomly. Then for all chromosomes the cost function (fitness function) is evaluated. On the basis of this information the parent chromosomes are selected. Then the crossover and mutation operator makes the children population. The new population is created from new children population and selected individuals (best ones) of the old population. If the good enough individual is found (if the problem is solved) algorithm ends. If not, the algorithm goes back to the point 2 (cost function evaluation) (Dideková, 2009).

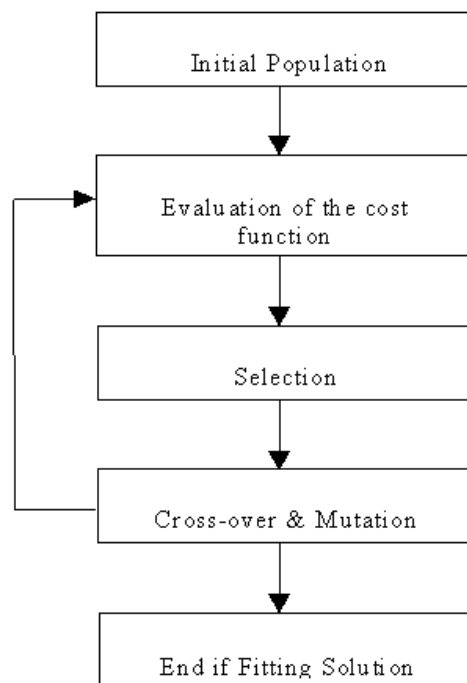


Figure 1.2 Genetic algorithm schema (Genetic algorithms - Introduction)

Let's take a closer look for each element of genetic algorithm.

Each individual (the description of problem solution) is modeled in computer memory – this description is called the chromosome. In all of practical problems used in the practical part of the Thesis the individuals can be seen as energy system's models (in fact the input parameters which defines models). Descriptions of individuals, as well as descriptions of cost functions which can be used, will be introduced in each case. The fitness function has to be chosen appropriately to the given problem. The function does not evaluate the chromosomes directly, but the phenotype of each individual. This grade is saved as the feature of each individual.

The initial population is built with n points from problem parameters space X . For genetic algorithm we are choosing the number of individuals and then, selecting them randomly. The next important step is checking the correctness of creating individuals.

Selection operator is used when the stop condition is not fulfilled – it means that the good enough individual was not found and the number of algorithm iteration is smaller than the established one. As a result of selection the new population is created from parents population. The most popular selection methods are:

- Fitness proportionate selection (roulette-wheel selection);
- Tournament selection;
- Truncation selection.

Crossover is a genetic operator which makes new generation from previous one. It uses two individuals from parent population to create two offspring individuals. There are many types of crossover operator, for example single point crossover or two point crossover, which are presented in the

Figure 1.3.

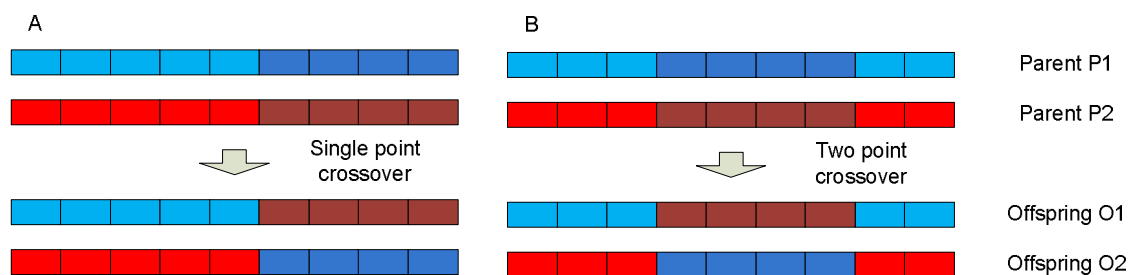


Figure 1.3 Crossover operator in genetic algorithm

Mutation is uncommon phenomena, but it is very important part of algorithm. It enters diversity into the solution set. The mutation operator changes value of randomly chosen part of chromosome. In many cases results are quite bad, but sometimes the output chromosome is suited better to environment – it gives better problem solution. Mutation helps when algorithm stops in the local extremums (it helps in finding global one). The main idea of mutation operator is shown in the Figure 1.4. Figure 1.5 presents the different types of mutation.

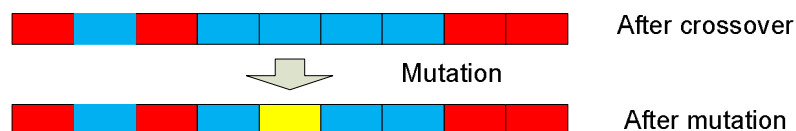


Figure 1.4 Mutation operator in genetic algorithm

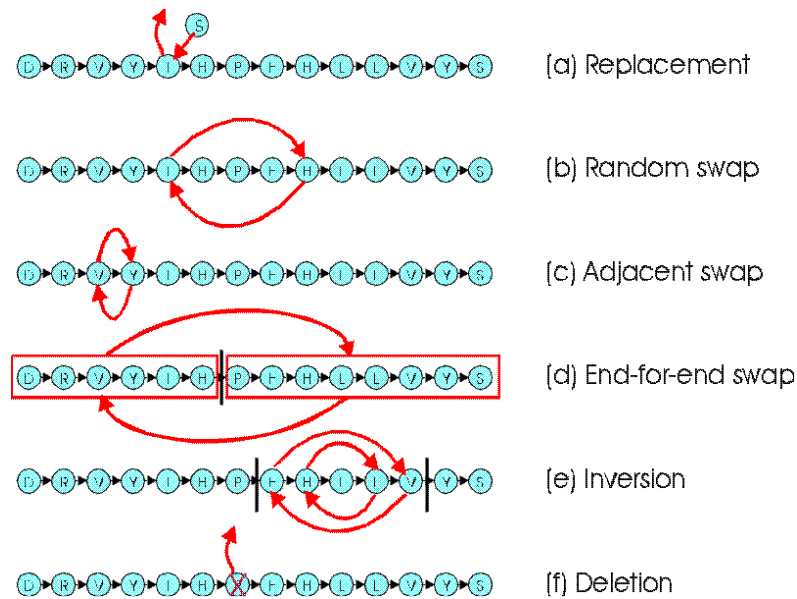


Figure 1.5 Mutation types (Stranz & Martin, 1997)

The main idea about working of crossover and mutation operators is presented in the Figure 1.6. The crossover improves the solutions which were found previously – it is process similar to climbing at the top a hill that is not the highest (local maximum). On the other hand mutation lets for trying random solutions from the problem space. Most mutation will be bad and individuals will die. However occasionally a higher peak may be found.

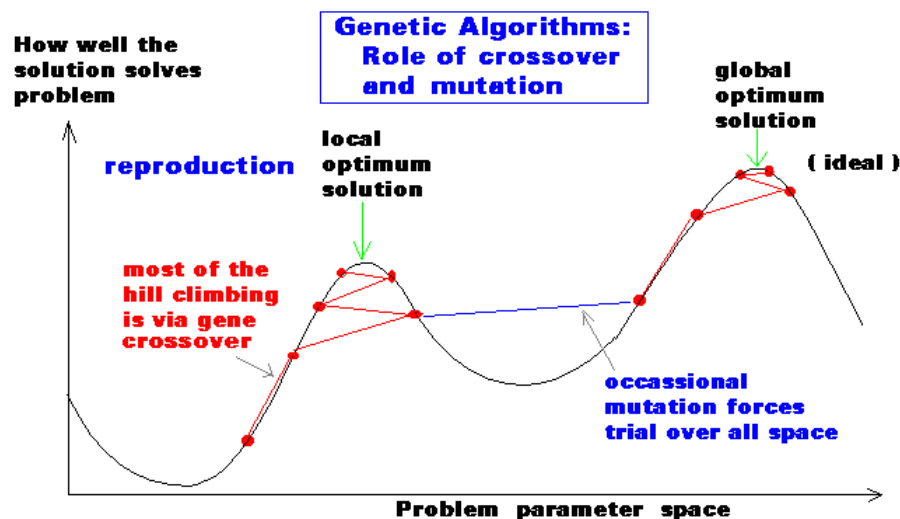


Figure 1.6 The role of crossover and mutation operators in genetic algorithm (Genetic Algorithms : General Idea)

1.2 Surrogate models – introduction

Optimization of energy systems is a field in continuous development, and one which requires a lot of computation time and memory. Moreover, recent research in this domain demonstrates that new methods that include uncertainties in the design of such systems are needed. This means sampling of uncertain parameters and increasing the necessary computational resources. The role of surrogate models may be a way to include uncertainty in the analysis of energy systems. Surrogate models possess the following crucial properties:

- They can provide a more efficient model, allowing more iteration in optimization and uncertainty analysis.
- Depending on the methods used, they can provide even analytical formulation what would open the door to uncertainty propagation.

It should be noted that the word ‘model’ will be assigned to the original pre-existing model, which was developed based on flow sheeting and integration software. The expression ‘surrogate model’ will represent the model that will be developed in this study.

Today, many energy system evaluation projects perform costly and lengthy laboratory experiments and/or complex computer models are used. Due to their complexity, a single evaluation of the model can require several hours of CPU processing time on a high-performance computer. Because of this, many modern engineering designs use surrogate models: (Tenne & Armfield, 2008) and (Won & Ray, 2005)

As the examples show, great results can be achieved when surrogate models are used (Fernandes, 2006)

There are several books and articles connected with surrogate methods theory (Caballero & Grossmann, 2008). There are mathematical (Kincaid & Chene, 2002) and artificial intelligence theory (Sivanandam, Sumathi, & Deepa, 2006). The real challenge is to apply the theory to a real life project. For each project the methodology has to be studied and the best method has to be chosen.

There are also several MATLAB Toolboxes available for surrogate models and for neural networks. The manual for them will be studied and used during model development for this project (Beale, Hagan, & Demuth, 2010).

1.3 Models evaluation methods

We have to introduce the methods of evaluating the quality of surrogate model.

To evaluate the model we normally use the additional auxiliary testing set.

In the mathematical models we can distinguish two different set of points: training set (nodes of interpolation) and testing set. In the case of artificial intelligence model (we will focus on the neural network model) we have three different sets: training, validation and testing set. Training and validation set will be described in the next chapters.

The testing set should cover the whole decision variable space (we should have a possibility to test all variants of the problem). However, in many cases the testing set is

chosen randomly – it is caused by a big complexity of the considered problem. In each of practical problems the detailed information about used testing set will be given.

There are several different measurements which can be used to grade the surrogate model. Some of them (used in this Master Thesis) are introduced below.

1.3.1 Norm of residual

The norm of residuals is often used as a measure of the goodness of fit. It is especially useful when we have to compare different fits.

The residuals d_k are the difference between the real data value y_k and the output of the surrogate model in given point $f(x_{1,k}, x_{2,k}, \dots, x_{n,k})$:

$$d_k = y_k - f(x_{1,k}, x_{2,k}, \dots, x_{n,k}).$$

The norm (in this context the L2-norm) of the vector z :

$$z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix},$$

can be expressed as:

$$||z||_2 = (|z_1|^2 + |z_2|^2 + \dots + |z_n|^2)^{1/2}.$$

So the norm of the residuals is the square root of the sum of squared d_k values. It can be calculated from:

$$norm = \left(\sum_{k=1}^n |d_k|^2 \right)^{1/2} = \sqrt{\sum_{k=1}^n |y_k - f(x_{1,k}, x_{2,k}, \dots, x_{n,k})|^2}.$$

Generally the smaller norm of residuals means, the fit is better.

1.3.2 Correlation coefficient r

Another common measurement of the “goodness” of a correlation is the correlation coefficient r . The square of r is the fraction of the variance in the dependent variables that is explained by the correlation. The mathematical form to express the correlation coefficient is following:

$$r^2 = 1 - \frac{norm^2}{(n-1)s^2};$$

where s is the standard deviation:

$$s = \sqrt{\frac{\sum(f(x_{1,k}, x_{2,k}, \dots, x_{n,k}) - \bar{f})^2}{n-1}},$$

And \bar{f} is the mean (average) of all output values.

The closer to 1 is the correlation coefficient, the better surrogate model fits the original function.

1.3.3 Mean square error

The mean squared error is default performance function of the feedforward neural network. This is the averaged squared error between the real values and output of the surrogate model.

The mean squared error is defined in the following way:

$$MSE = \frac{1}{n} \sum_{k=1}^n |y_k - f(x_{1,k}, x_{2,k}, \dots, x_{n,k})|^2.$$

This measurement is similar to the norm of residuals.

1.4 Mathematical models

In this chapter several mathematical techniques will be described. Mathematical formulas can be useful in building surrogate models. Although in many cases they are very complicated and it takes a lot of memory and time to find a satisfactory solution by them.

In this chapter, we will focus on one dimension approximation problem - the aim of the theory part is to present the main ideas and methods which can be used in surrogate modeling. The precise mathematical theory which is behind the used methods can be found in Numerical Analysis books (Kincaid & Chene, 2002). Some of the presented methods can be generalized for higher dimension problems. Although finding the smooth interpolants for multivariable functions is a difficult problem because of some unusual features of the multivariate problems. This features shows even in the bivariaite cases (two independent variables).

1.4.1 Interpolation problem definition

One of the way in which we can build a surrogate model is using the mathematical solution. There are several methods which can be used but this work is focusing on the

interpolation methods. They are a good example of the mathematical surrogate modeling. There are different methods of interpolation like, for example, linear, polynomial or spline interpolation. Chosen methods will be presented and implemented in one of the analyzed real life problems.

The interpolation problem is one of the basic problems of Numerical Mathematics and interpolation has also many engineering usages. The simplest explanation of the term interpolation is imagining the process of interpolation like an inverse of tabularizing a function. When we discretize a function we have an analytical form of a function and then we can use this form, build a table of values of this function in concrete points. On the other hand, when we interpolate, we know only values of the function in some, certain points, and using them we can find an analytical form of this function. This definition of the interpolation is consistent with the description of the surrogate modeling problem (we can generate only finite number of real examples but we would like to know the model values for different decision variable sets).

The formal definition of the interpolation problem is following:

For given $n \in \mathbb{N}$, x_0, x_1, \dots, x_n ($x_i \neq x_j$ for $i \neq j$) and function f defined in points $\{x_0, x_1, \dots, x_n\}$ find a function I_n satisfying the following conditions:

$$I_n(x_i) = f(x_i), (i = 0, 1, \dots, n).$$

Points x_0, x_1, \dots, x_n are called interpolation nodes and the function I_n is called an interpolation function.

The examples of solution for interpolation problem are shown below.

Another very important problem connected with interpolation is the interpolation error. The interpolation function has the same value as function f only in the nodes. In other points of interval in which we interpolate, there could be differences between interpolation function and the relevant function. This error of interpolation is called the rest of interpolation.

1.4.2 Nearest-neighbor interpolation

Nearest-neighbor interpolation is the one of the simplest method of interpolation. Sometimes it is called point sampling or proximal interpolation. This method can be used in one dimension problems as well as in the more complicated ones. Easy to implement algorithm selects the closest point and just assign its value like a solution of interpolation problem. Algorithm in the basic version does not include the values of other points in the neighborhood to the final solution so as a result we have piecewise-constant interpolation function instead of the continuous one.

The nearest-neighbor algorithm as well as its modifications (k-nearest-neighbor algorithms) are often use in the pattern recognition problems and are classified as the simplest of the machine learning algorithms. Simplicity of this solution is connected with

rather big interpolation error in case of fitting the functions. The example of the use of this method can be seen in the Figure 1.7 .

This method is rarely used in the one dimension problems. We can use other interpolation methods, like for example linear interpolation, which is not more difficult and almost always gives better results. On the other hand in multi dimension problems we can consider nearest-neighbor interpolation because of its simplicity and speed. But if we would like to have good results with this method we need many samples – interpolation nodes. However problems considered in the next part of the Thesis need more precise and faster methods (the nodes generation is connected with time) so the different surrogate techniques have to be applied.

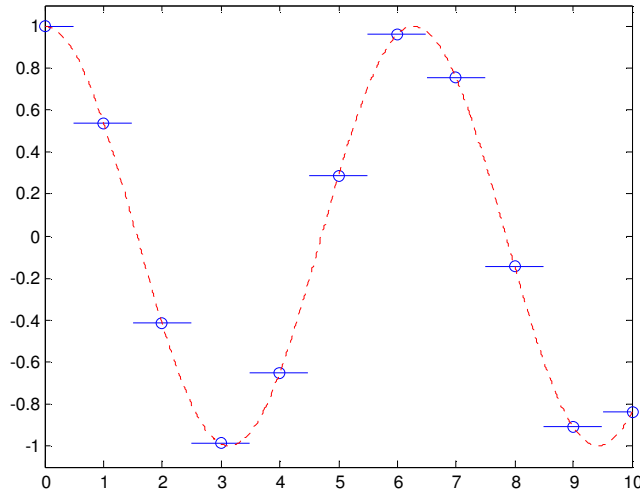


Figure 1.7 Nearest-neighbour interpolation of function $\cos(x)$ based on 10 equidistant nodes in the interval $[0,10]$.

1.4.3 Linear interpolation

Second simple method of interpolation is the linear interpolation (sometimes it is called *lerp*). It is old and often used method – it has been used since antiquity in Mesopotamia and Greek in mathematical and astronomical computations.

In case of two interpolation nodes $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ the linear interpolant is the straight line between them. This situation is illustrated in the Figure 1.8. We can easily notice that the following proportion is true (geometrical derivation):

$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1}.$$

From this equation we have the linear interpolation formula. The y value of the interpolated point $P(x, y)$, where x is in the interval (x_1, x_2) , is equal:

$$y = y_1 + (x - x_1) \cdot \frac{y_2 - y_1}{x_2 - x_1}.$$

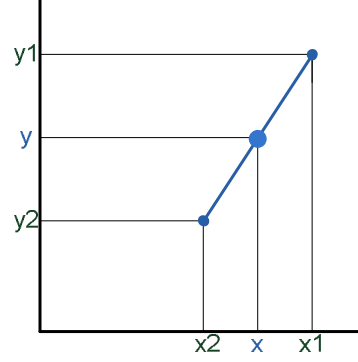


Figure 1.8 Linear interpolation between two given points P_1 and P_2

The interpolation of bigger set of data points is presented in the Figure 1.9. The interpolation is concatenation of linear interpolants build for each pair of consecutive points. The final form of interpolation function has the following features:

- Continuous curve;
- Discontinuous derivative;
- Differentiability class C^0 .

Linear interpolation is simple in comparison to other methods, but we have to remember about the interpolation error. The error of approximation R_T is defined as:

$$R_T = f(x) - p(x),$$

where:

$$p(x) = f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1}(x - x_1)$$

is linear interpolation polynom. We can prove by using Rolle's theorem that the following expression is true:

$$|R_T| \leq \frac{(x_2 - x_1)^2}{8} \max_{x_1 \leq x \leq x_2} |f''(x)|,$$

if original function f has continuous second derivative. The error of linear interpolation depends on the maximum value of second derivative of original function and the length of

interpolation interval – for longer interval and “curvier” function we have worse extrapolation.

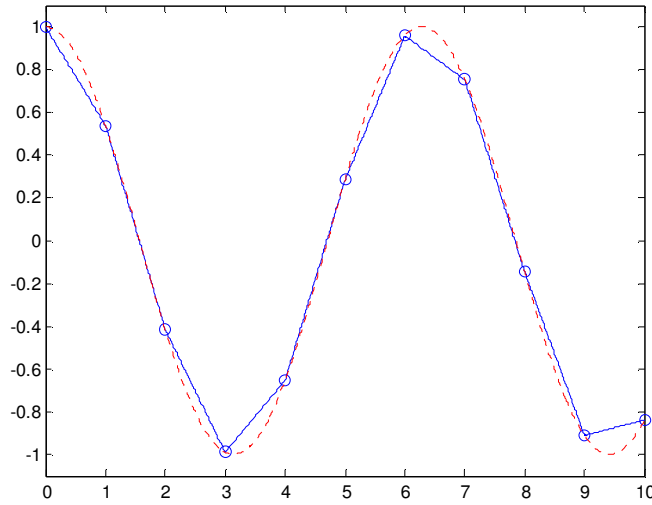


Figure 1.9 Linear interpolation of function $\cos(x)$ based on 10 equidistant nodes in the interval $[0, 10]$.

1.4.4 Polynomial interpolation

Linear interpolation for two nodes is the simple case of polynomial interpolation. Polynomial interpolation is more complicated method than previous forms of interpolation. We can obtain relatively good results with it, however, certain conditions must be satisfied.

Interpolation polynomial definition is following:

By interpolation polynomial we mean polynomial degree at most n , which takes values w_0, w_1, \dots, w_n in pair wise different points x_0, x_1, \dots, x_n . Numbers x_0, x_1, \dots, x_n we called interpolation nodes.

This definition is consistent with Interpolation problem. There is several ways of construction the interpolation polynoms. We will show two basic methods of constructing the polynomial, which satisfy the interpolation problem conditions, to prove that the problem have the polynomial solution.

We will start from the Lagrange form of interpolation polynomial. Using this form we can in the simplest way show how the interpolation polynomial works. We have to think about the following polynomial $L_n(x)$:

$$L_n(x) = f(x_0)\lambda_0(x) + f(x_1)\lambda_1(x) + \dots + f(x_n)\lambda_n(x) = \sum_{k=0}^n f(x_k)\lambda_k(x),$$

where:

$$\lambda_k(x) = \frac{(x-x_0)(x-x_1)\dots(x-x_{k-1})(x-x_{k+1})\dots(x-x_n)}{(x_k-x_0)(x_k-x_1)\dots(x_k-x_{k-1})(x_k-x_{k+1})\dots(x_k-x_n)}.$$

The polynomial $L_n(x)$ is built with using values of function f in given points and auxiliary polynoms $\lambda_k(x)$. We can notice that polynomial $\lambda_k(x)$ is equal to 1, for x equals x_k , when i is k and $\lambda_k(x)$ is 0, when i is not equal to k :

$$\lambda_k(x_i) = \begin{cases} 0 & \text{for } i \neq k \\ 1 & \text{for } i = k \end{cases}, \quad i = 0, 1, \dots, n$$

So we can see that value of interpolation polynomial L_n in points x_0, x_1, \dots , is the same as value of the function f in those points. In this way we have shown that the Lagrange problem has a solution. What's more it is very easy to prove from Fundamental Theorem of Algebra that this solution is unambiguous.

Computing a coefficient of interpolation polynomial or its value in a concrete point is rather complicated and laborious. Of course we can find algorithms which help us to do it, but the whole process of computing still will be very time-consuming. For this reason we introduce the Newton form of interpolation polynomial. Each polynomial can be presented in the Newton form, which means the sum of the products of auxiliary polynoms $p_k(x)$ and suitable coefficients b_k . We can also write an interpolation polynomial in the Newton form:

$$L_n(x) = \sum_{k=0}^n b_k p_k(x),$$

where:

$$\begin{aligned} p_0(x) &\equiv 1 \\ p_k(x) &= (x-x_0)(x-x_1)\dots(x-x_{k-1}) \quad (k=1, 2, \dots). \end{aligned}$$

In this case, we call coefficient b_k differential ratios of function f with nodes x_0, x_1, \dots, x_n . The simplest way to compute each differential ratio is using the following recursive formula:

$$\begin{aligned} f[x_l, x_{l+1}, \dots, x_{l+k}] &= \frac{f[x_{l+1}, x_{l+2}, \dots, x_{l+k}] - f[x_l, x_{l+1}, \dots, x_{l+k-1}]}{x_{l+k} - x_l} \\ f[x_l] &= f(x_l), \quad l = 0, 1, \dots, n \end{aligned}$$

We can notice that to compute a differential ratio rank k we need two differential ratios rank $k-1$ with other nodes. When we know how we can find coefficients b_k , we can write the final form of the Newton interpolation polynomial:

$$L_n(x) = f[x_0] + \sum_{i=1}^k f[x_0, x_1, \dots, x_i] (x - x_0)(x - x_1) \dots (x - x_{i-1}).$$

We can see that it is built from polynoms $p_k(x)$ and differential ratios. Finding the coefficients of polynom in this form is easier than in case of the Lagrange form. To make it in the simplest way we can use the algorithm of building the table of differential ratios. When we interpolate on the basis of equidistant nodes the interpolation process is even easier (in case of Lagrange form of interpolation polynom) – instead of differential ration we are using finite differences and the formulas are simplify.

Another important problem of polynomial interpolation is the polynomial interpolation rest $R(x)$. For any continuous function $f(x)$ and any pairwise points x_0, x_1, \dots, x_k the following formula holds:

$$f(x) - L_n(x) = R(x),$$

$$R(x) = f[x, x_0, \dots, x_n] (x - x_0)(x - x_1) \dots (x - x_n).$$

Precise calculating the interpolation error is rather complicated and we have to look for other methods of finding the rest of interpolation. Especially when it comes to practical solutions, we only have to know an approximate maximum of error.

The second important problem of interpolation is the following: how should we choose nodes if we want to have the minimal rest of interpolation? Until now, we didn't give any special conditions to interpolation nodes. There was only one important thing – when we interpolate function on the interval $[a, b]$ all nodes also have to belong to that interval. For other nodes we have other interpolation polynoms, which approximate a given function more or less precisely. We have to find nodes which give us the best approximation. In other words we want to find a polynom which is closest to given function. As a matter of fact, those conditions are satisfied by polynom based on the nodes which are roots of the Chebyshev polynom. The Chebyshev polynom can be defined by trigonometric identity:

$$T_n(x) = \cos(n \cdot \arccos x) = \cosh(n \cdot \operatorname{arccosh} x).$$

We can introduce also a formula, which helps us to find needed nodes:

$$x_k = \frac{b-a}{2} \cdot \cos \frac{2k+1}{2(n+1)} \pi + \frac{a+b}{2}.$$

The differences between polynomial interpolation in different cases can be seen in the Figure 1.10. Four different situations are presented:

- A. Polynomial interpolation based on the 5 equidistant nodes;
- B. Polynomial interpolation based on the 5 Chebyshev nodes;
- C. Polynomial interpolation based on the 33 equidistant nodes;
- D. Polynomial interpolation based on the 33 Chebyshev nodes.

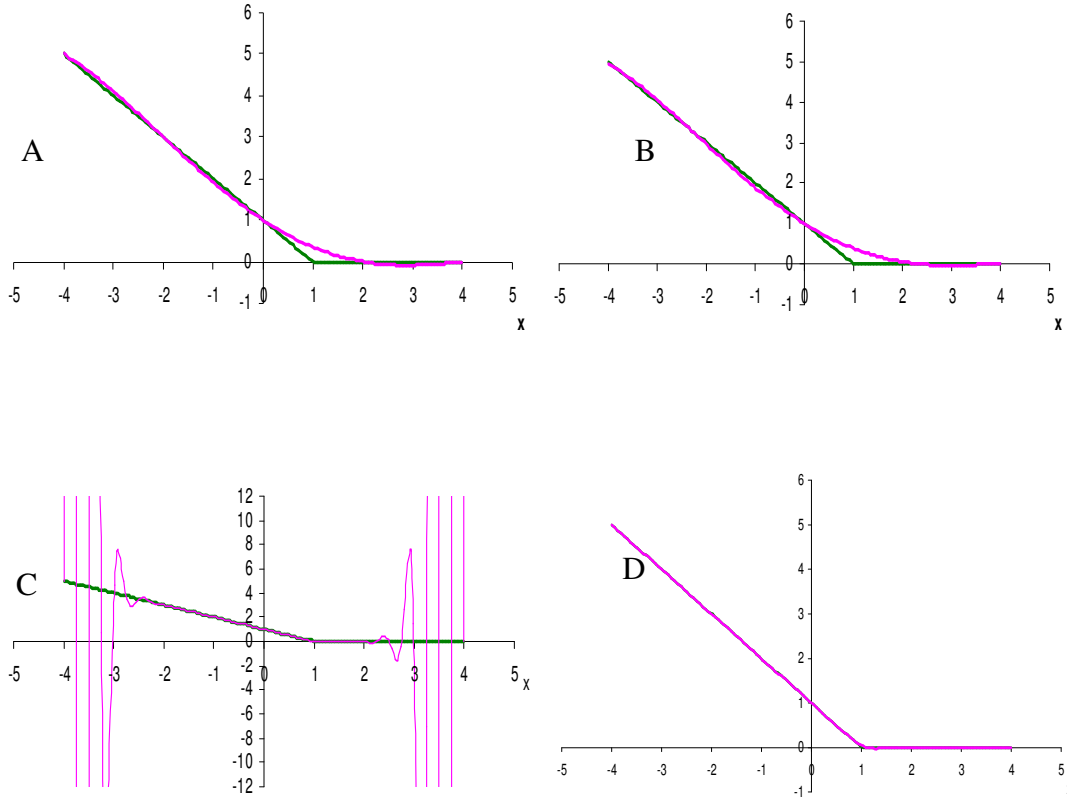


Figure 1.10 Interpolation polynoms of function $f(x) = \max\{0, 1-x\}$ in interval $[-4, 4]$ based on different types of nodes: equidistant(A,C) and Chebyshev(B,D) and different number of nodes: 5 nodes (A, B) and 33 nodes (C,D).

For 5 nodes the results are similar for both cases: equidistant and Chebyshev ones. For 33 Chebyshev nodes we have very good results with minimal rest of interpolation. On the other hand in the case of 33 equidistant nodes the interpolation error is huge. This is a good illustration of the Runge Phenomenon. It is the situation when with growing values of n (that is growing number of interpolation nodes), the maximum error of interpolation grows infinitely. The rest of interpolations are shown in the Table 1.1 Rest of interpolations for different interpolation conditions.

Table 1.1 Rest of interpolations for different interpolation conditions.

	Type of nodes	Number of nodes	Error maximum
A	5	Equidistant	0,34
B	5	Chebyshev	0,37
C	33	Equidistant	101216
D	33	Chebyshev	0,042

To sum up, the smaller rather than bigger number of equidistant nodes gives better results. With the Chebyshev nodes, on the other hand, it is the other way around, that is the more nodes, the better results.

1.4.5 Spline interpolation

The next method of numerical mathematics is spline interpolation. This type of interpolation is in some details similar to linear interpolation and in other to polynomial one. In this case the interpolation function is special type of piecewise low degree polynom – spline (but the polynom pieces fits smoothly together). There are several advantages of this type of interpolation over the standard polynomial interpolation:

- Interpolation error can be minimized with using low degree polynoms for the spline. The rest of interpolation is smaller than in case of linear interpolation;
- The Runge effect is not present in this case;
- Interpolant is smooth;
- It is easier to find a value of the interpolating function (evaluate it) than in case of high degree polynoms.

Formal definition of spline according to (Kincaid & Chene, 2002):

A spline function consists of polynomial pieces on subintervals joined together with certain continuity conditions. Formally, suppose that $n+1$ points x_0, x_1, \dots, x_n have been specified and satisfy $x_0 \leq x_1 \leq \dots \leq x_n$. These points are often called knots in case of spline interpolation. Suppose also that an integer $k \geq 0$ has been prescribed. A spline function of degree k having knots x_0, x_1, \dots, x_n is a function S such that:

- On each interval $[x_{i-1}, x_i]$, S is a polynomial of degree $\leq k$;
- S has a continuous $(k - 1)$ st derivative on $[x_0, x_n]$.

So we are looking for a spline of degree n function in following form:

$$S(x) = \begin{cases} S_0(x) & x \in [x_0, x_1] \\ S_1(x) & x \in [x_1, x_2] \\ \vdots & \\ S_{n-1}(x) & x \in [x_{n-1}, x_n] \end{cases}.$$

$S_i(x)$ is a polynom of degree k .

The spline interpolation of degree 0 has the same principals as nearest-neighbor interpolation. Linear spline interpolation (spline of degree 1), on the other hand, is consistent with linear interpolation process, which was described earlier.

Now let focus on the spline interpolation of degree 3 (cubic spline interpolation). This method will be presented in practical usages in the next part of the Thesis.

For cubic splines we have following requirements:

- $S(x_i) = f(x_i)$ - standard condition for interpolation
- Continuously differentiable condition for $i = 1, \dots, n - 1$:

$$\begin{aligned} S_{i-1}(x_i) &= S_i(x_i), \\ S'_{i-1}(x_i) &= S'_i(x_i), \\ S''_{i-1}(x_i) &= S''_i(x_i). \end{aligned}$$

Moreover we have to remember about the conditions for first and last knot. We can give them as: $S'(x_0) = u$ and $S'(x_n) = v$ - the result is clamped cubic spline. The second possibility is: $S''(x_0) = 0$ and $S''(x_n) = 0$ - the natural cubic spline .

In the case of cubic spline interpolant has the form of separate cubic polynoms for each interval with the different coefficient for each of them ($x \in [x_i, x_{i+1}]$):

$$S_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i.$$

The example of the cubic spline interpolation can be seen in the Figure 1.12. We can observe, that in the case of interpolation of function $\cos(x)$ based on 10 equidistant nodes in the interval $[1,10]$ it gave great results. The functions are almost identical.

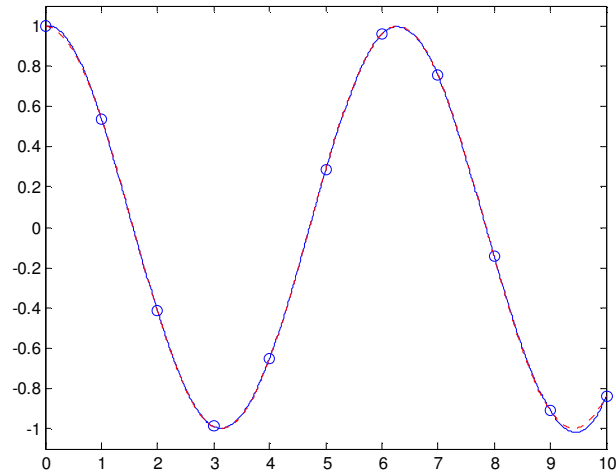


Figure 1.11 Cubic spline interpolation of function $\cos(x)$ based on 10 equidistant nodes in the interval $[0, 10]$.

To sum up – the spline interpolation has many advantages and is popular in practical solutions. It is easy to implement, fast and produce smooth curves (for example in case of cubic spline). In case of polynomial interpolation we have continuous solution but in many cases for high degree polynomials there are big interpolation errors next to the ends of interpolation intervals. In case of cubic spline those problems are not present. But on the other hand it is only piecewise continuous – when the problem is sensitive for changes in the higher derivatives (for cubic splines higher than second) the interpolation errors can occur.

1.4.6 Cubic Hermite interpolation

The cubic Hermite interpolation find an interpolant $P(x)$ in each interval on the basis of the given value and slopes at the two endpoints. The values in the knots of interpolant and the function are the same: $P(x_i) = f(x_i)$. The second condition is that the first derivative $P'(x)$ is continuous (but the second one $P''(x)$ is not). The second important thing is choosing the slopes in the knots. $P(x)$ shape is compatible with the shape and monotonicity of the data. It can be observed especially in the local extremums – the interpolant act as the original function (it preserves function monotonicity and extremums)(MATLAB documentation, pchip command). The example of cubic Hermite interpolation can be seen in the Figure 1.12

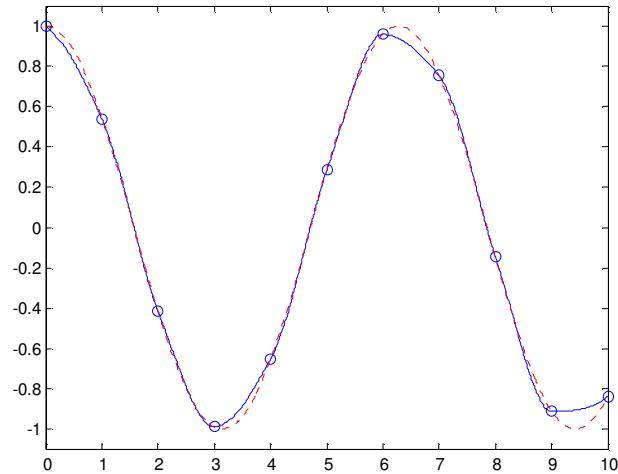


Figure 1.12 Cubic Hermite interpolation of function $\cos(x)$ based on 10 equidistant nodes in the interval $[0, 10]$.

The cubic Hermite interpolation is in many aspects similar to cubic spline interpolation. They construct interpolants in very similar way. The difference is in the way in which the slopes at the knots are chosen (in case of cubic spline the second derivative is continuous). The results of this behavior are following:

- It is easier to build Hermite interpolant than cubic spline one;
- The evaluation time of both of them is very similar;
- Spline interpolation gives smoother results because $S''(x)$ is continuous;
- Spline is better solution in situation when we interpolate a smooth function;
- When the interpolated function is not smooth (when it has a point of a non-continuity) it is better to choose Hermite interpolation because it do not have so big effect of overshoots and oscillation. (MATLAB documentation, pchip command)

The Figure 1.13 shows the differences between cubic spline interpolation and cubic Hermite interpolation.

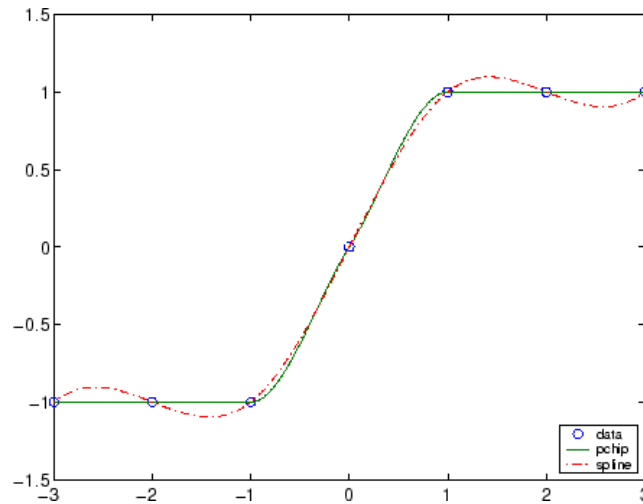


Figure 1.13 Cubic Spline Interpolation (spline) versus Cubic Hermite Interpolation (pchip) (MATLAB documentation, pchip command)

1.4.7 Polynom fitting

Interpolation is a part of the wider brand of mathematic called curve fitting (or regression analysis). When we interpolate we would like to find a function which exactly fits the data points. In case of the curve fitting we are looking for the function which closely fits the data, for example in a least squares sense. Last squares sense means the minimization the sum of squared residuals (residual is the difference between a real value in a point and value of fitted function in that point)

The curve fitting involves interpolation as well as smoothing – process of finding a smooth function which approximates the data points. Another important term connected with regression analysis is extrapolation. Extrapolation is using the approximated function beyond the interval in which the beginning data points was situated. The uncertainty and errors are in this case much bigger than in the case of standard situation.

The example of regression analysis is shown in the Figure 1.14. We can notice that the worst error is in the case of first degree polynom (it is straight line). From the other hand in case of fourth degree polynom we can observe big rest on the ends of approximation interval.

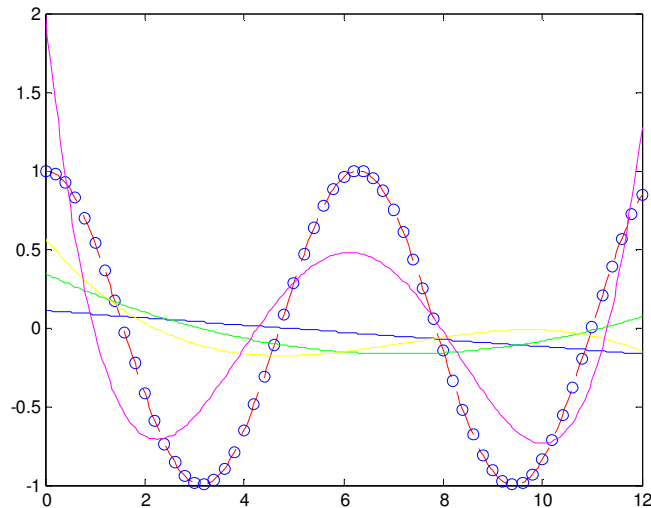


Figure 1.14 Polynomial fitting for function $\cos(x)$ - red line - in the interval $[0,12]$ by polynomials of different degree: blue - first degree, green - second degree, yellow - third degree, violet - fourth degree.

There are several reasons to choose curve fitting instead of polynomial interpolation. When we have many nodes it is better to use regression analysis (instead of high dimension polynomial interpolation) because of:

- The Runge Phenomena – situation in which high order polynomials can have oscillatory character and the rest of interpolation grows infinitely;
- The high dimension polynomials can fit the data precisely but in many cases more important is the effect of averaging the results (what can be done by fitting the smaller degree polynomial);
- The exact interpolation of high dimension polynomial can be hard and time consuming. In many cases we need only approximate solution which can be good enough to practical problems;
- The high order polynomials are less smooth than simpler one (they have more inflection points).

Two situations were discussed: when we have more nodes than degree of interpolation and when those two numbers are the same. The situation when we would like to fit the polynomial which degree is higher than number of nodes is the most complicated one. Imagine that we would like to fit the straight line (first degree polynomial) to one point. There is infinite number of possible solutions. So finding a good method of evaluating solutions and choosing one is hard.

Finding the best degree of polynomial (which fits the data the best) has to be done experimentally, but we can assume that we should choose the smallest possible degree, which gives us a satisfying result.

1.5 Artificial intelligence models

The second very important techniques of surrogate modeling is artificial intelligence approach. When we are talking about artificial intelligence we think about computer programming methods which imitate the real biology organisms or processes.

The methods which can be used in the surrogate modeling are neural networks, supported vector machines, machine learning and genetic programming.

Below the basic concepts of neural networks and genetic programming will be introduced.

The main part of the Thesis – description of the surrogate model for complicated energy system will be connected with neural networks problems.

1.6 Neural networks

1.6.1 Introduction

Many scientists have a big interest in neural networks. They are the great instrument to solve problems from different brands of science. We can understand neural networks as the modern computing systems, which architecture and method of processing information is similar to the biological prototype – human brain. In the Table 1.2 the differences between human brain and computer are presented. Neural networks are a way in which computer try to imitate brain processes. Information processed by the network has the form of the numerical data and because of that the neural networks can be used to model many different types of systems.

The first neural network – perception – was invented by Frank Rosenblatt in 1957 at the Cornell Aeronautical Laboratory. It can be seen as a simple linear classifier build on the basis of the biological model, which was able to learn.

Table 1.2 Comparison of brain and computer (Orr, 1999)

	Processing element	Element size	Energy use	Processing speed	Style of computation	Fault tolerant	Learns	Intelligent, conscious
Brain	10^{14} synapses	10^{-6} m	30 W	100 Hz	Parallel, distributed	Yes	Yes	Usually
Computer	10^8 transistors	10^{-6} m	30 W (CPU)	10^9 Hz	Serial, centralized	No	A little	Not (yet)

There are seven main parts in the process of creating the Neural Network:

- Data collecting;
- Network creation;
- Network configuration;
- Initializing weights and biases;
- Network training;
- Network validating

- Network using.

In the practical part of the Master Thesis all of this step will be used and demonstrated. To understand good all of those steps we have to look into the theory behind neural networks.

1.6.2 Simple neuron

The simplest and most important element of neural network is simple, single input neuron. The schema of the neuron is presented in the Figure 1.15.

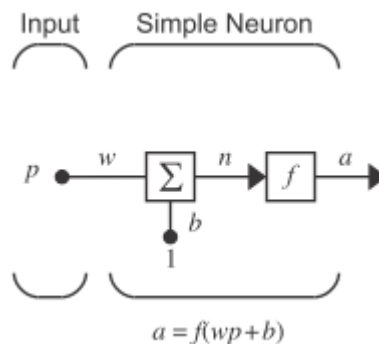


Figure 1.15 Simple neuron (Beale, Hagan, & Demuth, 2010)

There are several important values presented in the graph: input p , weight w , bias b , net input n , transfer function f and output a .

The values of bias and weight are adjustable. The training network process is in fact a process of finding the appropriate values of this two parameters.

Now we will take a closer look on the mathematical operations which take place in the neuron. Firstly the scalar input is multiplied by a weight. The product of this operation is added to the bias value – as a result we get a net input. We can understand a bias value like an additional weight for input equals 1. The next operation is finding the value of transfer function for the net input - the result of this operation is output value. So we can write:

$$a = f(n) = f(p \cdot w + b).$$

The names of described processes are respectively: the weighting function, the net input function and the transfer function.

The most popular weighting function is multiplying the input value by a weight, however sometimes different function are used. As a net input function we use normally the summation of weighted inputs with the bias. Transfer function depends from the type of the considered problem.

1.6.3 Transfer functions

There are many different transfer functions used in the design of neural networks. Two of them are presented in the Figure 1.16. Those transfer functions – linear and log sigmoid transfer function - are commonly used in the practical solutions, especially in the problems connected with fitting the functions.

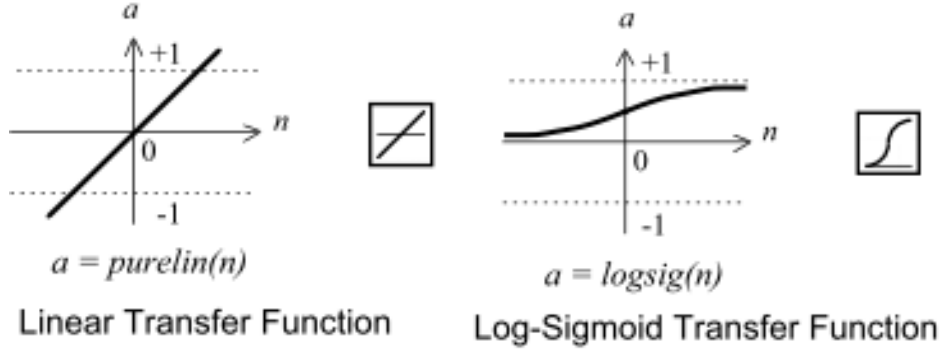


Figure 1.16 Transfer functions used in Neural Network (Beale, Hagan, & Demuth, 2010)

First example is linear transfer function. This type of the neuron is often used in the last layer of the multilayer approximation network.

The second example – sigmoid transfer function (log-sigmoid function, also known as a logistic function) – is commonly used in the hidden layers of multilayer networks. This function takes the input (any value between plus and minus infinity) and give the output value in the range (0,1). Sigmoid function is prized because its derivative is easy to calculate. This fact is helpful in calculating and updating the weights in the process of training.

The sigmoid transfer function is represented by the following formula:

$$\sigma(t) = \frac{1}{1 + e^{-\beta t}}.$$

For $\beta = 1$ the derivative of this function can be calculated from:

$$\frac{d\sigma(t)}{dt} = \sigma(t)[1 - \sigma(t)].$$

And for $\beta \neq 1$, using $\sigma(\beta, t) = \frac{1}{1+e^{-\beta t}}$, we have the following formulation:

$$\frac{d\sigma(\beta, t)}{dt} = \beta[\sigma(\beta, t)[1 - \sigma(\beta, t)]].$$

1.6.4 Neuron with vector input

In this subchapter the neuron with many inputs values is presented. The individual input elements: p_1, p_2, \dots, p_R are multiplied by weights: $w_{1,1}, w_{1,2}, \dots, w_{1,R}$. Their sum is $\mathbf{W}\mathbf{p}$ (the dot product of the single row matrix \mathbf{W} and the vector \mathbf{p}). The schema of the single neuron with many inputs is presented in the Figure 1.17.

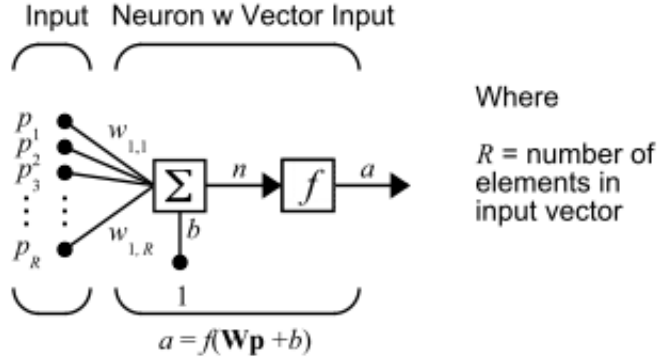


Figure 1.17 Neuron with vector input (Beale, Hagan, & Demuth, 2010)

Additional value in the neuron is the bias – it is summed with the weighted inputs to form the net input n . Then to get the value of output we have to use the transfer function:

$$a = f(n) = f(\mathbf{W} \cdot \mathbf{p} + b) = f(w_{1,1} \cdot p_1 + w_{1,2} \cdot p_2 + \dots + w_{1,R} \cdot p_R + b).$$

The presented form of notation is quite complicated and detailed. Because of that simpler form is introduced – it is shown in the Figure 1.18. This abbreviated notation is especially useful when we have to represent complicated multilayer networks.

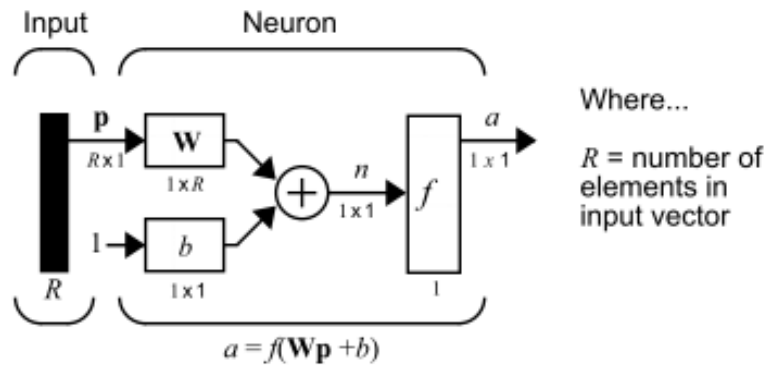


Figure 1.18 Neuron with vector input: abbreviated notation (Beale, Hagan, & Demuth, 2010)

1.6.5 Multilayer neural network

The very simple idea of multilayer network is presented in the Figure 1.19. It shows a neural network with one input layer, one hidden layer and one output layer.

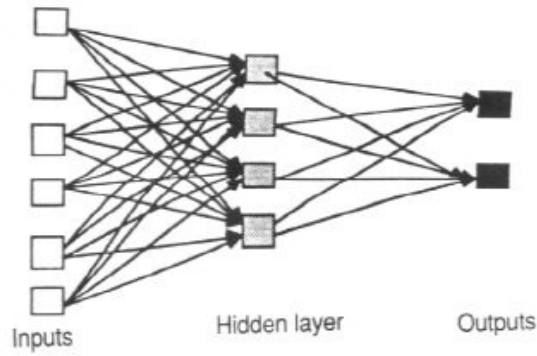


Figure 1.19 Simple representation of multilayer Network (Stergiou & Siganos)

Now we will explain more precisely the concept of layer and multilayer network.

Generally, two or more neurons can be connected to create a layer. A network can contain one or more layers. A layer of network includes the weights, the multiplication and summation operations, the bias b and the transfer function f . The array of inputs - vector \mathbf{p} - is not included in or called a layer.

First we will focus on the single layer of neurons. An examples of different notations of one-layer network with R input elements and S neurons follows are presented in the Figure 1.20.

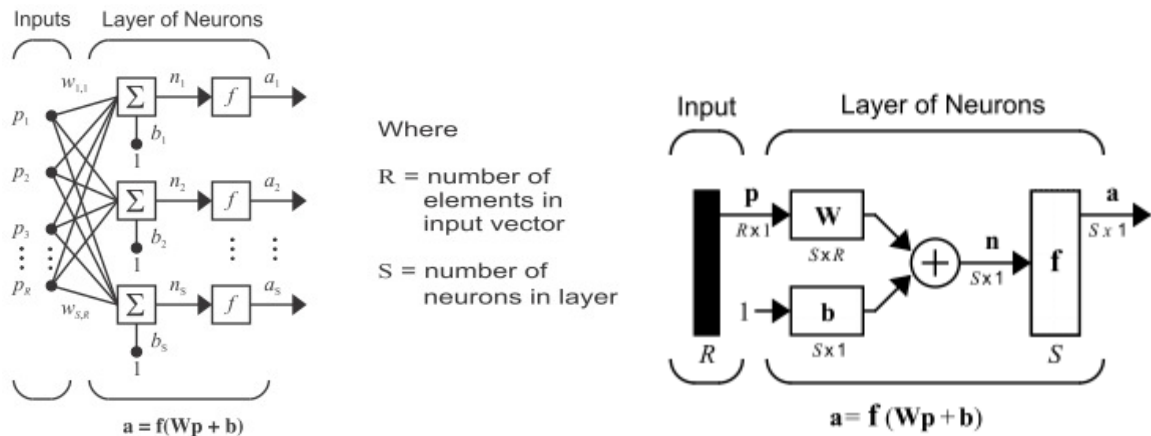


Figure 1.20 One layer of a neural network in normal and abbreviated notation (Beale, Hagan, & Demuth, 2010)

In this network, each element of the input vector \mathbf{p} is connected to each neuron input through the weight matrix \mathbf{W} .

The multilayer network schema a is presented in the Figure 1.21.

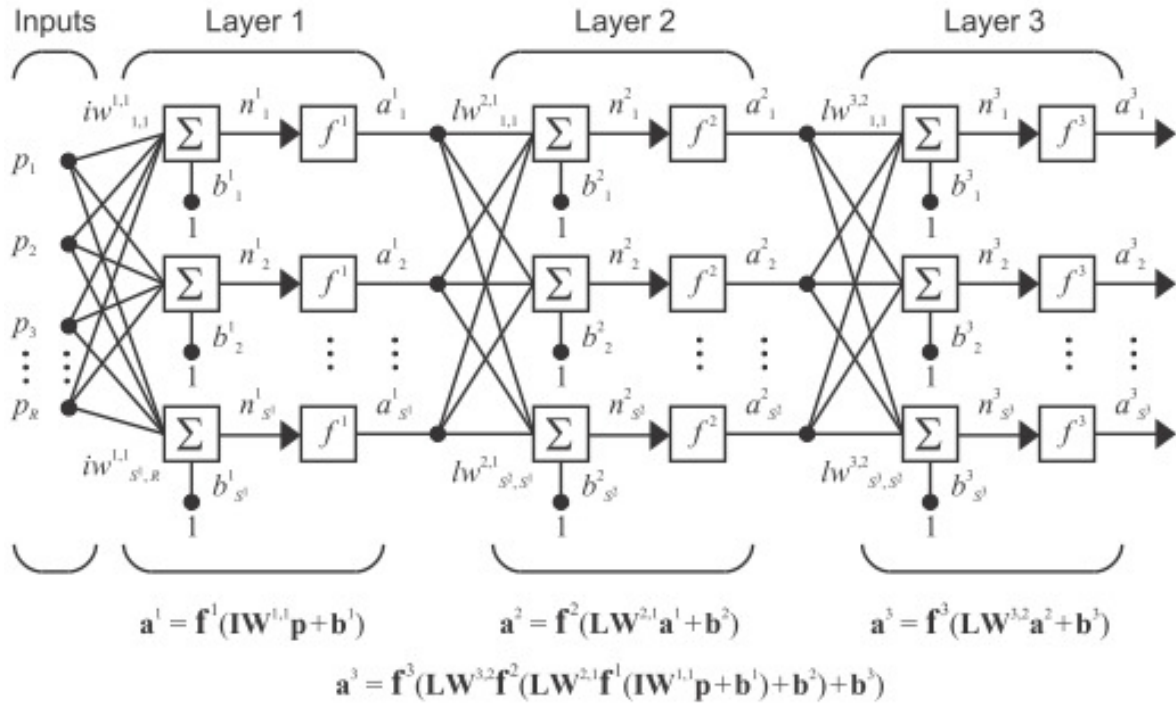


Figure 1.21 Multilayer network (Beale, Hagan, & Demuth, 2010)

The next figure, Figure 1.22, shows the basic neural network model used in the practical problems in the Master Thesis. As it is said in the (Beale, Hagan, & Demuth, 2010) this network can be used as a general function approximator. It can arbitrarily well approximate any function with a finite number of discontinuities (of course we have to remember about sufficient neurons in the hidden layer).

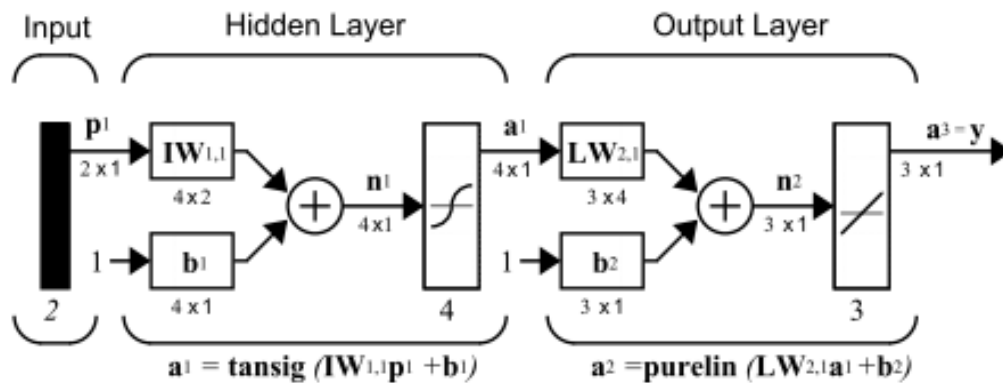


Figure 1.22 Approximation network with one hidden layer(Beale, Hagan, & Demuth, 2010)

1.6.6 Back propagation algorithm

The back propagation training is one of the most popular methods of training neural networks. The first step in this problem is to prepare the appropriate sets of the data: training and validating one. Training set should precisely characterize investigated problem

(it is the best situation). The training set has two parts: the vectors of input data (values which are taken by the network) and the vectors of target data (the values which should be given by a network).

The initial values of weights and biases are chosen randomly.

The back propagation algorithm can be described in the following way:

The chosen input vector is transformed by the network. After that the network output value is compared with the target vector for this input. We have to check if the answer of the network is correct, and if not how big is the error. Next, this error is propagated in the network but in the reverse direction than the input vector (from output to the input layer). On the basis of this error the weights in each neuron are corrected to decrease the error of the answer in the next iteration. This procedure is repeated until the error of the network is smaller than assumed earlier value.

The same procedure has to be done for all of the input vectors from training set. Processing of the whole training set is called an epoch. After each epoch the error in validation set is calculated. The minimum value of validation error from all epoch is saved. If during six (this number was used in the practical part of the Thesis) consecutive epochs the error in validation set does not decrease, the training process is stopped.

Good and detailed description of back propagation algorithm can be find in (Bernacki, Włodarczyk, & Gołda, 2004)

1.7 Genetic programming

Genetic programming can be used not only to optimize a model but also to build a surrogate model. As it was mentioned before the genetic algorithms apply the Darwin Evolution Theory to the regression model over a series of generations (Sreekanth & Datta, 2010). All genetic algorithms need specific input data. In this case each individual is a specified surrogate model – in other words it is a syntactically correct computer programs which behave as a conventional model. All of those individuals are built with parametric constants, specified inputs, operators (addition, subtraction, multiplication, etc.) and function (ex. trigonometric function). The example of individuals can be seen in the Figure 1.23.

To evaluate individuals we have to prepare the testing set of input points for original model and the values of output in this point of the real model. Then we examine each individual by checking the difference (error) between the individual output (it is program – surrogate model so we can find its output for input point) and real output for all points from testing set.

The genetic algorithm works in the standard way: cross-over, mutation, duplication and delete operators are used to prepare next population. This action leads to finding the best suited individuals – the best surrogate models which have the smallest error. Figure 1.23 shows how the children population (Figure 1.23 c, d) is created from parent population (Figure 1.23 a, b): they are created as a result of cross-over and mutation operators applied to the two individuals from parent population. In the Figure 1.23 d is presented interesting situation: variable y is eliminated from the equation, so this individual gives the constant output for any input point.

The children population is tested with the previously described way and reproduced according to the results. After many iterations of the algorithm the functions evaluate and the good surrogate model can be found.

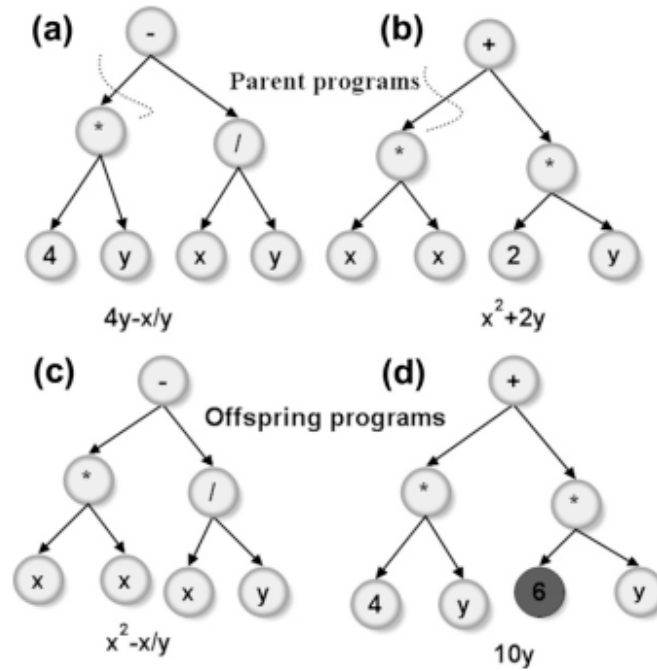


Figure 1.23 Genetic programs in tree structure for one decision variable model (Sreekanth & Datta, 2010)

2 ONE DECISION VARIABLE MODEL

The second chapter of the Thesis consist the analysis of the relatively simple real life problem (with only one decision variable). In this part the theoretical assumptions used in model are described. Also the surrogate model is prepared and evaluated. Different model techniques are introduced into the practical problem and differences in those methods are pointed.

2.1 Flow sheet and model description

In this part of the Thesis the energy system with installed gas turbines will be considered. There are several ways to reduce CO₂ from such system. One of them is to capture and sequester CO₂ by using monoethanolamine (MEA). To make this process efficient we need high concentration of CO₂. To make the CO₂ concentration high in exhaust gases (they are going to sorbent process) we can re-inject a part of flue gases in the compressor.

Gas turbine model with recirculation

The first part of the system which we should analyze is gas turbine model with recirculation. The schema of this system is shown in the Figure 2.1.

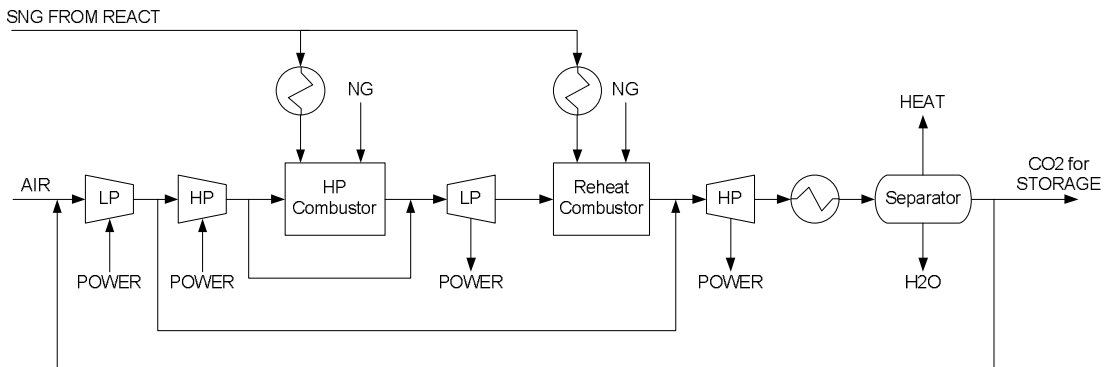


Figure 2.1 Gas turbine with recirculation schema – GT_SEQ

All details about the streams and other elements in the system are given in the reports according the computer models used in the practical part of the Thesis (Dubuis & Tock, 2010).

Modeling an recirculation in the system as similarly to reality as possible is not a simple task. Several assumptions were made in order to prepare a model of an recirculation impacts in the system. All of the assumptions were made on the basis of nominal condition presented in (Dubuis & Tock, 2010). The assumptions are following:

- Volumetric flow rate is constant - it is equal to $400 \text{ m}^3/\text{s}$. This assumption is made due to maintain velocity triangle in the compressor;
- Split fraction of the streams after LP and HP compressors are constant and equal respectively $S_1 = 0.77$ and $S_2 = 0.61$ (this assumption is made to avoid detailed simulation; the splitting fraction of these flows are given by the geometry and the type of fluid);
- The temperatures of streams before the turbines are maximally equal to $T_1 = 1100^\circ\text{C}$ (LP turbine) and $T_1 = 1300^\circ\text{C}$ (HP turbine). This assumption is made because the capacity of the blade cooling system. Those temperatures are controlled by the air excess in the combustor

Moreover, we should remember that volumetric flow rate is not constant in the turbines. It is because the inlet temperatures are maintained constant. The velocity triangles in the turbines will change.

It is also assumed that isentropic efficiency of the turbines is constant, but in fact, it should decrease because of non-optimal flow conditions connected with recirculation. Also the work produced by the turbines is overestimated.

The turbine models, as well as compressor models are built on the basis of mass and energy balances and isentropic efficiencies.

To improve the flame stability we need to add H_2 inlet streams. The amount of H_2 to be added as a function of the excess O_2 left after combustion is shown in the Figure 2.2.

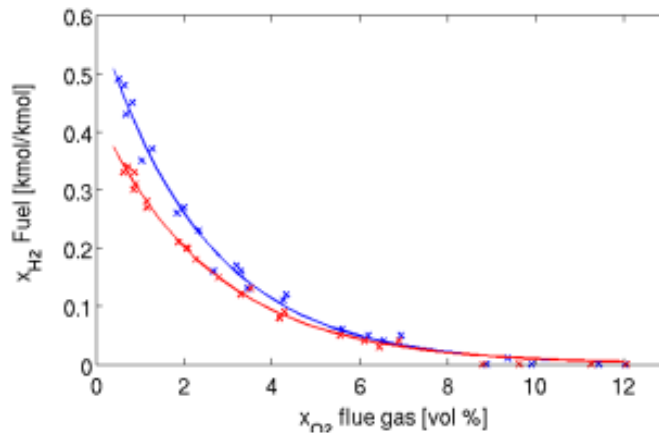


Figure 2.2 The amount of H_2 to be added to improve flame stability

The blue curve is for adding pure H_2 and red one represents adding of a syngas (60% of H_2 and 40% of CO). In case of syngas it is mixed with natural gas. The syngas is more available as the product of autothermal reforming of natural gas, so this situation will be considered.

Syngas production

As it was mentioned, to maintain flame stability, it is necessary to add H₂ to the fuel and the hydrogen is introduced by producing syngas. The unit to produce syngas was also modeled and its schema is presented in the Figure 2.3.

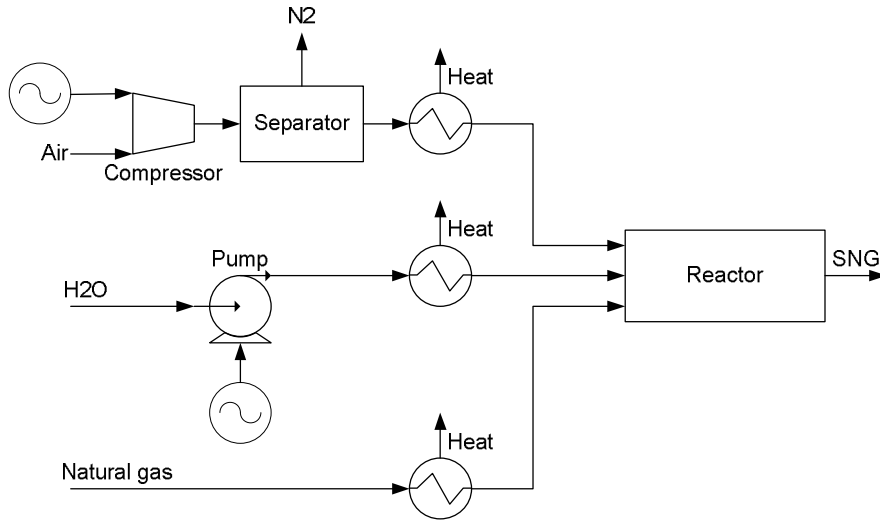


Figure 2.3 Syngas production schema – REACT

The syngas production is modeled as synthesis in the membrane reactor. The oxygen for partial oxidation process is provided from the air. The main process reactions are:

- Steam methane reforming SMR ($\Delta h_r^0 = -206 \text{ kJ/mol}$):

$$\text{CH}_4 + \text{H}_2\text{O} \leftrightarrow \text{CO} + 3\text{H}_2$$
- Partial oxidation POX ($\Delta h_r^0 = 36 \text{ kJ/mol}$):

$$\text{CH}_4 + 1/2 \text{O}_2 \leftrightarrow \text{CO} + 2\text{H}_2$$
- Auto-thermal reforming where the heat for the endothermic reforming is satisfied by the exothermic partial oxidation.

These reactions are followed by a one or two step water-gas shift reaction to convert CO to CO₂ and additional H₂ (WGS: $\text{CO} + \text{H}_2\text{O} \leftrightarrow \text{CO}_2 + \text{H}_2$, $\Delta h_r^0 = 41 \text{ kJ/mol}$).

The overall reaction is autothermal and it is assumed that there is no oxygen at the reactor outlet. The reactions are operated at 950 °C and 30 bar with a steam to carbon ratio initially set to 2.

The natural gas combustion and recovering heat from H₂ streams leaving the reactor at high temperature help in satisfy heat demands of H₂ production. Moreover we have to remember, that the temperature of the H₂ entering the combustion chamber can not be

higher than 200 – 250⁰C. So there can be used heat exchanger to preheat the reactants in those streams.

The MEA unit

The last part of the system to be modeled is MEA unit. In used model is was assumed as a black box. The schema is represented in the Figure 2.4.

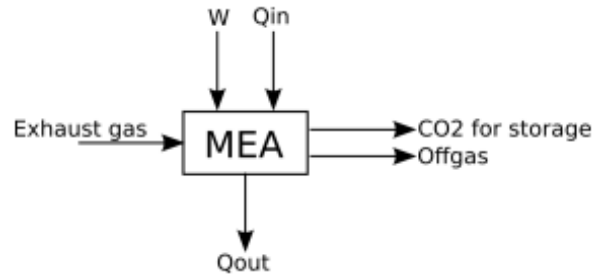


Figure 2.4 CO₂ capture black box model for chemical absorption with monoethanolamines (Dubuis & Tock, 2010)

In fact, this problem can be modeled in much more complicated way presented in the Figure 2.5.

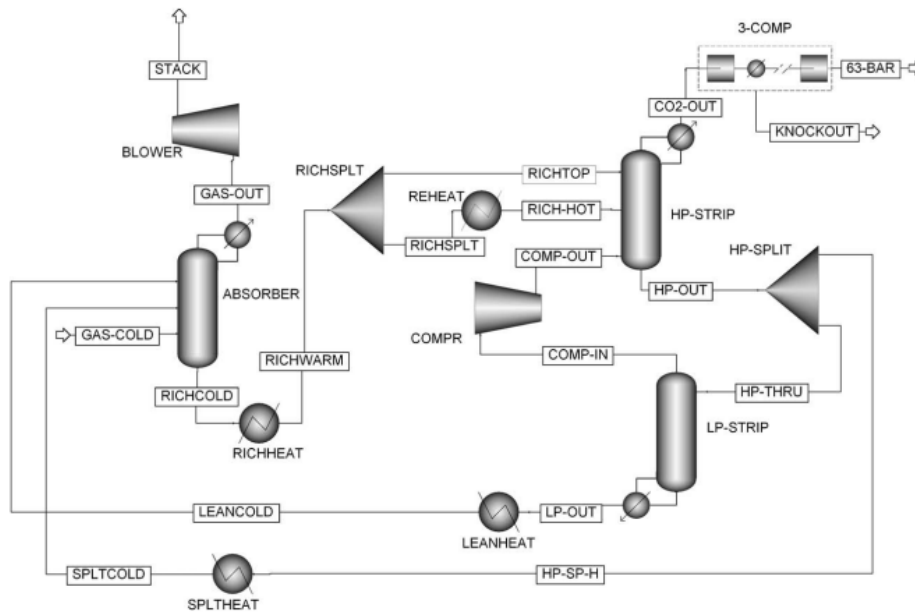


Figure 2.5 Flow-sheeting model which can be use to simulate multi-pressure, split-fraction CO₂ capture (Bernier, Marechal, & Samson, 2009)

The model which is based on flowsheeting software is very time and memory-consuming. A simpler model is based on the following assumptions. First of all the amount of CO₂ captured is equal to 90% (it is the data basis on the literature). The heat and mechanical power consumption have the following values:

- Thermal heat need at 150: 3.7 MJ/kg CO₂ captured;
- Thermal heat demand at 110: 3.7 MJ/kg CO₂ captured;
- Mechanical power: 1 MJ/kg CO₂ captured.

2.2 Problem description

The first problem which was analyzed in the Thesis is simple model with only one decision variable. The modeled system is gas turbine power plants designs with CO₂ capture with Flue Gas Recirculation (FGR). The decision variable in this case is the Recirculation Fraction and all output values depend on this parameter.

In the next chapters different surrogate models will be introduced and evaluated. This simple problem is an excellent example of techniques used in surrogate modeling. Moreover it is a good introduction to the more complicated cases which can be solved by using similar methodology. The surrogate models of the simple system will show differences between mathematical and artificial intelligence approaches. Also the methods of finding interpolation error and evaluating the surrogate model will be introduced.

We have to remember about one more big advantage of one decision variable surrogate models. In practical problems we often prepare the sensitivity analysis – we want to know the influence of one decision variable for the output function. So even simple mathematical or artificial intelligence models can help in preparing this curves. After generating very small number of points and building a surrogate model we can have very precise sensitivity analysis data.

Moreover the prepared surrogate model can be used in real life problem. The existing model is used to optimize the gas turbine power plant system in LENI EPFL laboratory. The optimization process is complicated especially because the computer model is built with several parts. The main optimization program uses data from two separate models. In the Figure 2.6.A the schema of this system is presented. The VALI model as well as ASPEN model are both flowsheet software. The main part of the system, gas turbine power plant, is modeled in VALI software. The CO₂ storage module is build in ASPEN technology. Figure 2.1.B presents the simplified problem model which was used to prepare the surrogate model.

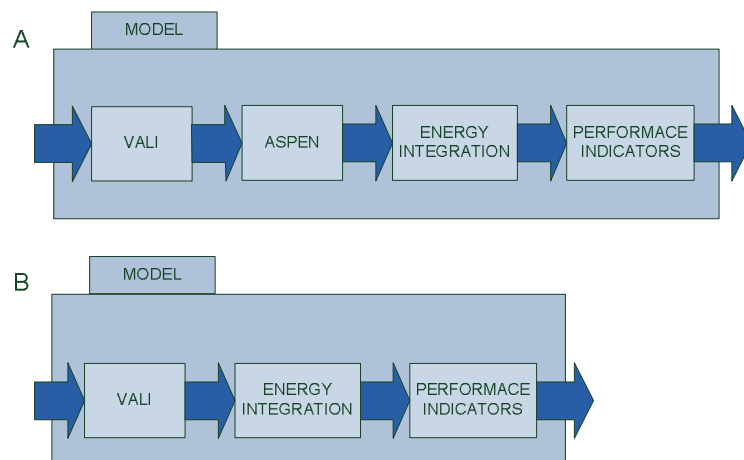


Figure 2.6 The multi-objective problem (gas turbine power plant with CO₂ capture with Flue Gas Recirculation). A - program schema, B simplified problem.

Generating data by both parts of the model is very laborious and time consuming. For example, generation output values from one decision variable point with OSMOSE procedure and VALI model takes approximately 7 seconds. So for 100 points we need about 700 seconds – it mean over 11 minutes. And the optimization algorithm calculates thousands of points in both VALI and/or ASPEN modules.

The surrogate model will be prepared for the gas turbine power plant (originally modeled in VALI). This should make the optimization process much faster. It will be possible to analyze more points and run more iteration of optimization algorithm. So also the results of the optimization should be more precise.

2.3 Modeling problem description

In this part of the Master Thesis the system with only one input parameter was considered. All output parameters are the function of one variable: recirculation ratio.

In the first part of the modeling process only one output value was considered. It was the power output from the turbine. The different methods of building surrogate model were implemented for this problem. The mathematical models (nearest-neighbor interpolation, linear interpolation, cubic spline interpolation, cubic Hermite interpolation and polynom fitting) as well as artificial intelligence models (neural network with one neuron in input layer and one neuron in output layer) were prepared. Also the different numbers of nodes of interpolation was checked. All of this method was evaluated and the precision of the different surrogate models was compared.

After the method's evaluation the best one from mathematical methods – spline interpolation - was chosen to use in more complicate model. The second model had the same input value like simpler one: recirculation ratio, but it had many more output parameters. These parameters are needed by energy integration and performance evaluation (hot and cold stream definition, energy stream...). The output parameters analyzed in the problem are shown in the Table 2.1. Table 2.1 presents also the example of values of all output variables for recirculation ratio equals 30%. For the same problem data (input and output variables) the second surrogate model was prepared. The second model was based on the neural network with one neuron in input layer and 79 neurons in output layer.

Table 2.1 All input and output variables in the surrogate model with example of values.

<i>Model</i>	<i>Tag name</i>	<i>Output variable</i>	<i>Unit</i>	<i>Value</i>
INPUT				
GT_SEQ	RECY_FRAC	Recirculation fraction	-	0,3
OUTPUT				
GT_SEQ	TURB2_OUT_T	Temperature after HP turbine	K	926,14
GT_SEQ	TURB2_OUT_HFLOWTOT	Total enthalpy flow after HP turbine	kW	425595

GT_SEQ	COOL_OUT_T	Temperature before separator	K	298,15
GT_SEQ	COOL_OUT_HFLOWTOT	Total enthalpy flow before separator	kW	22325,1
REACT	F_1_002O2_T	Temperature after compressor and N ₂ separation	K	855,2
REACT	F_1_002O2_HFLOWTOT	Total enthalpy after compressor and N ₂ separation	kW	205,66
REACT	F_1_003_T	Temperature of oxygen before reactor	K	1223,15
REACT	F_1_003_HFLOWTOT	Total enthalpy of oxygen before reactor	kW	354,18
REACT	F_420_T	Temperature after pump	K	298,19
REACT	F_420_HFLOWTOT	Total enthalpy after pump	kW	2,86
REACT	F_421_T	Temperature of water before reactor	K	1223,15
REACT	F_421_HFLOWTOT	Total enthalpy of water before reactor	kW	4413,33
REACT	F_410_T	Temperature of input NG	K	298
REACT	F_410_HFLOWTOT	Total enthalpy of input NG	kW	24752,1
REACT	F_411_T	Temperature of NG before reactor	K	1223,15
REACT	F_411_HFLOWTOT	Total enthalpy of NG before reactor	kW	26264
GT_SEQ	H21_T	Temperature of SNGfor HP combustor before preheating	K	1223,15
GT_SEQ	H21_HFLOWTOT	Total enthalpy of SNGfor HP combustor before preheating	kW	8061,34
GT_SEQ	H21A_T	Temperature of SNGfor HP combustor after preheating	K	473,15
GT_SEQ	H21A_HFLOWTOT	Total enthalpy of SNGfor HP combustor after preheating	kW	7123,4
GT_SEQ	H22_T	Temperature of SNGfor reheat combustor before preheating	K	1223,15
GT_SEQ	H22_HFLOWTOT	Total enthalpy of SNGfor	kW	22970,2

		reheat combustor before preheating		
GT_SEQ	H22A_T	Temperature of SNG for reheat combustor after preheating	K	523,15
GT_SEQ	H22A_HFLOWTOT	Total enthalpy of SNG for reheat combustor after preheating	kW	20468,5
GT_SEQ	STORAGE_WRCO2	Partial mass flow rate of CO ₂ to storage	kg/s	124370
GT_SEQ	STORAGE_WRH2O	Partial mass flow rate of H ₂ O to storage	kg/s	23056,8
GT_SEQ	STORAGE_WRN2	Partial mass flow rate of N ₂ to storage	kg/s	908493
GT_SEQ	STORAGE_WRO2	Partial mass flow rate of O ₂ to storage	kg/s	115366
GT_SEQ	NGAS_TOT_MASSF	Mass flowrate of NG for both combustors	kg/s	11,32
GT_SEQ	NGAS_TOT_LHVWT	Mass low heat value of NG for both combustors	kJ/kg	49195,2
REACT	F_410_MASSF	Mass flowrate of NG	kg/s	0,45
REACT	F_410_LHVWT	Mass low heat value of NG	kJ/kg	50001,2
REACT	H2_TOT_MASSF	Mass flowrate in SNG from reactor	kg/s	1,82
REACT	H2_TOT_LHVWT	Mass low heat value in SNG from reactor	kJ/kg	11003,5
GT_SEQ	COMB1_IN_MRO2	O ₂ partial flow rate before HP combustor	kmol/s	1,34
GT_SEQ	COMB1_OUT_MRO2	O ₂ partial flow rate after HP combustor	kmol/s	0,69
GT_SEQ	TURB1_OUT_MRO2	O ₂ partial flow rate after LP turbine	kmol/s	1,55
GT_SEQ	COMB2_OUT_MRO2	O ₂ partial flow rate after reheat combustor	kmol/s	0,78
REACT	W_401_POWER	Power input for pump	kW	3,49
REACT	W_1_001_POWER	Power input for compressor	kW	953,25
GT_SEQ	W_1_POWER	Power input for LP compressor	kW	225348
GT_SEQ	W_2_POWER	Power for HP compressor	kW	37908,1
GT_SEQ	W_3_POWER	Power output from LP turbine	kW	76719,8

GT_SEQ	W_4_POWER	Power output from HP turbine	kW	411485
GT_SEQ	W_5_POWER	Total power output	kW	224948
GT_SEQ	NGAS_TOT_MRC1	SNGinput partial flow rate of methane	kmol/s	0,48
GT_SEQ	NGAS_TOT_MRC2	SNGinput partial flow rate of ethane	kmol/s	0,12
REACT	F_410_MOLF	Molar flowrate of NG	kmol/s	0,028
GT_SEQ	SEP_OUT_WRCO2	O ₂ partial mass flow rate from separator	kg/s	49,35
GT_SEQ	SEP_OUT_MRCO2	O ₂ partial flow rate from separator	kmol/s	1,12
GT_SEQ	RECIRC_MRCO2	CO ₂ partial flow rate in recirculation stream	kmol/s	0,34
GT_SEQ	STORAGE_WRCO2	CO ₂ partial mass flow rate to storage	kg/s	124370
GT_SEQ	STORAGE_MRCO2	CO ₂ partial flow rate to storage	kmol/s	0,78
MEA	FUMEEES_WFCO2	CO ₂ compound weight fraction in fumes	-	0,12
MEA	FUMEEES_MASSF	Mass flow rate of fumes	kg/h	1171290
MEA	CO2_CAPT	Ratio of CO ₂ captured by MEA process	-	0,9
MEA	CO2_CAPTE_MASSF	CO ₂ mass flow rate from MEA	kg/s	31,093
MEA	CO2_CAPTE_MOLF	CO ₂ molar flow rate from MEA	kmol/s	0,71
REACT	F_1_001_MASSF	Mass flow rate of air to compressor	kg/s	1,59
REACT	F_1_002_P	Pressure after compressor	bar	30
REACT	F_411_MRC1	Partial flow rate of methane in NG before reactor	kmol/s	0,028
REACT	H2_TOT_MRC1	Partial flow rate of methane in SNG from reactor	kmol/s	0,0014
REACT	H2_TOT_P	Pressure of SNG from reactor	bar	30
REACT	H2_TOT_T	Temperature of SNG from reactor	K	1223,15
REACT	H2_TOT_MOLF	Molar flow rate of SNG from reactor	kmol/s	0,14

REACT	F_420_P	Pressure after pump	Bar	30
REACT	F_1_003_VOLF	Volumetric flow rate of oxygen before reactor	m ³ /s	2356,23
GT_SEQ	COMB1_IN_MASSF	Mass flow rate before HP combustor	kg/s	224,68
GT_SEQ	COMB1_OUT_T	Temperature after HP combustor	°C	1425,05
GT_SEQ	TURB1_OUT_MRN2	N ₂ partial flow rate after LP turbine	kmol/s	9,94
GT_SEQ	COMB2_OUT_T	Temperature after reheat combustor	°C	1518,72
GT_SEQ	COMP1_OUT_P	Pressure after LP compressor	bar	20
GT_SEQ	COMP1_IN_P	Pressure before LP compressor	bar	1
GT_SEQ	COMP1_IN_MASSF	Mass flow rate before LP compressor	kg/s	479,47
GT_SEQ	C1_EFFIC	Efficiency of LP compressor	-	0,83
GT_SEQ	COMP2_OUT_P	Pressure after HP compressor	Bar	30
GT_SEQ	COMP2_IN_P	Pressure before HP compressor	bar	20
GT_SEQ	COMP2_IN_MASSF	Mass flow rate before HP compressor	kg/s	370,15
GT_SEQ	C2_EFFIC	Efficiency of HP compressor	-	0,88

To evaluate the methods for many outputs the set of 200 random points was generated. All correlation coefficients r were calculated on the basis of this set. The interpolation nodes was generated as equidistant point in the interval of interpolation (actually the set of 100 equidistant points was generated in the beginning and then needed number of nodes was chosen from the whole set). The one output models (in the first part of modeling process) were evaluated on the basis of the 100 equidistant points.

In this part of Master Thesis several MATLAB files was generated: beginning from point generation scripts and functions to create the surrogate models, ending on the analysis and results plotting functions. Moreover the intuition function to find the value of surrogate model for given input was prepared (it can be used to incorporate surrogate model into the more complicated optimization program).

All the mathematical methods was implemented on the basis of MATLAB function. Such function as:

- `pp = interp1(x,Y,method,'pp')` with different values of parameter `method`:
 - 'nearest' – nearest-neighbor interpolation
 - 'linear' - Linear interpolation (default)
 - 'spline' - Cubic spline interpolation
 - 'cubic' - Piecewise cubic Hermite interpolation

This function was used to perform the nearest-neighbor interpolation, linear interpolation, cubic spline interpolation and cubic Hermite interpolation;

- `p = polyfit(x,y,n)`

This function was used to perform the polynomial interpolation and polynomial fitting.

The neural network used to prepare artificial intelligence surrogate model was implemented with Neural Network Toolbox 7. Detailed description of this Toolbox and all the functions can be found in (Beale, Hagan, & Demuth, 2010).

2.4 Results

This chapter presents the results of surrogate modeling of one variable problem.

2.4.1 Results for one output (times, errors, graphs)

The first analyzed situation was the problem with one input and one output. Our aim was to predict the value of the tag `W_5_POWER` on the basis of tag `RECY_FRAC`. There were several methods used to find the surrogate model: mathematical ones as well as artificial intelligence ones.

Mathematical models

The mathematical methods, which were introduced previously, were implemented to find the surrogate models for analyzed system. Below the results are described and also the graphs are shown. Those graphs are excellent example of all dependences and features of mathematical interpolation described in the theoretical section.

On the graphs it can be seen that with growing value of recirculation fraction, the value of power output of the system also grows. Some of the physical dependences will be explained in the next subchapter. However, all of them can not be explained due to model complexity. This underlines the advantage of surrogate model being to represent tendencies without having to model all relations between variables.

All the results will be presented in the graphs. For each method the several options for number of nodes was investigated. In cases of nearest-neighbor interpolation, linear interpolation the interpolation functions was prepared for 2 and 11 equidistant nodes. Cubic spline interpolation and cubic Hermite interpolation was prepared for 2, 5, 8 and 11 equidistant nodes. For polynomial interpolation 6 different interpolation polynomials are

presented (in this case the degree of the polynom depends on the number of nodes). The last method is polynomial fitting. Two different polynomials (polynom of the degree 5 and degree 10) were fitted to the data points (in different situations it was 5, 10, 15 or 20 equidistant nodes).

All the graphs have the similar outlook. On the upper graph the real data and interpolation function are presented. Also the interpolation nodes are pointed. The second graph (lower one) shows the interpolation error – the graph presents the residuals in function of recirculation fraction. The residuals are the difference between the real data and data interpolated by the surrogate model. The graph shows also the value of norm of residuals and the correlation coefficient between data and surrogate model.

The first built model was nearest-neighbour interpolation model. The results can be seen in the Figure 2.7. The interpolation function is built with the horizontal lines with the middle point in the nodes. The biggest interpolation error can be seen in the middle of interval between neighboring nodes (this is caused by the shape of the function – it is monotonically; if there would be any local extrema this property changed). The bigger slope of the original function causes the bigger interpolation error. This feature of nearest-neighbor interpolation can be seen on the each of the graph. The interpolation errors are bigger in the right side of interpolation interval than in the left side and the slope of the function is also steeper there. Moreover it can be seen that in the nodes the residuals are equal to 0. On the right side of the node the residuals are positive and on the left side they are negative (this is caused by the original data shape).

It can be seen also that the bigger number of interpolation nodes makes the interpolation error smaller. The norms of residuals are smaller and the correlation coefficients are bigger with the growing number of nodes. When there are more parts of interpolation function they interpolate data with bigger precision.

The value of norm of residuals for shown example of 11 nodes is over 2022. The correlation coefficient for this case is 0.99434.

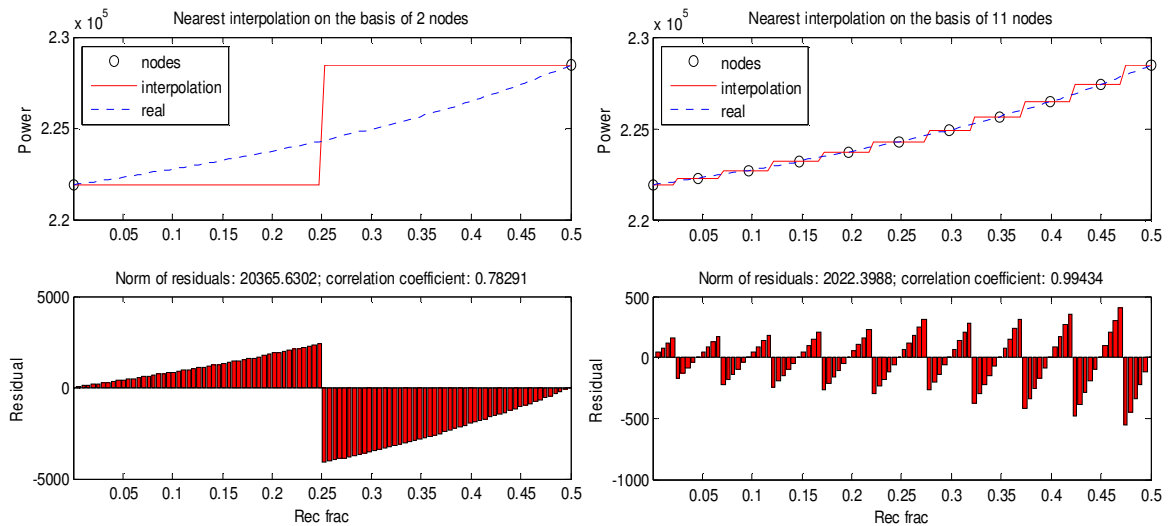


Figure 2.7 Nearest-neighbor interpolation for different number of nodes for Power Output

The next analyzed mathematical interpolation method was linear interpolation. The graphs illustrating this method are shown in the Figure 2.8. The interpolation function is also built with the linear parts (as in the nearest – neighbor interpolation method), but in this case the lines connect the neighboring nodes.

As previously the biggest interpolation errors can be seen in the middle of interval of interpolation (this is caused by the shape of the original data). The interpolation errors are smaller when the interpolation intervals are shorter. The error distribution (shape of the residual graphs) is different than in the nearest-neighbor method. All of the residuals are smaller than 0 (and this feature is also connected with shape of original data function).

The value of norm of residuals we are getting in the case of 11 nodes is equal to 76.4. In the same case the correlation coefficient is 0.99999.

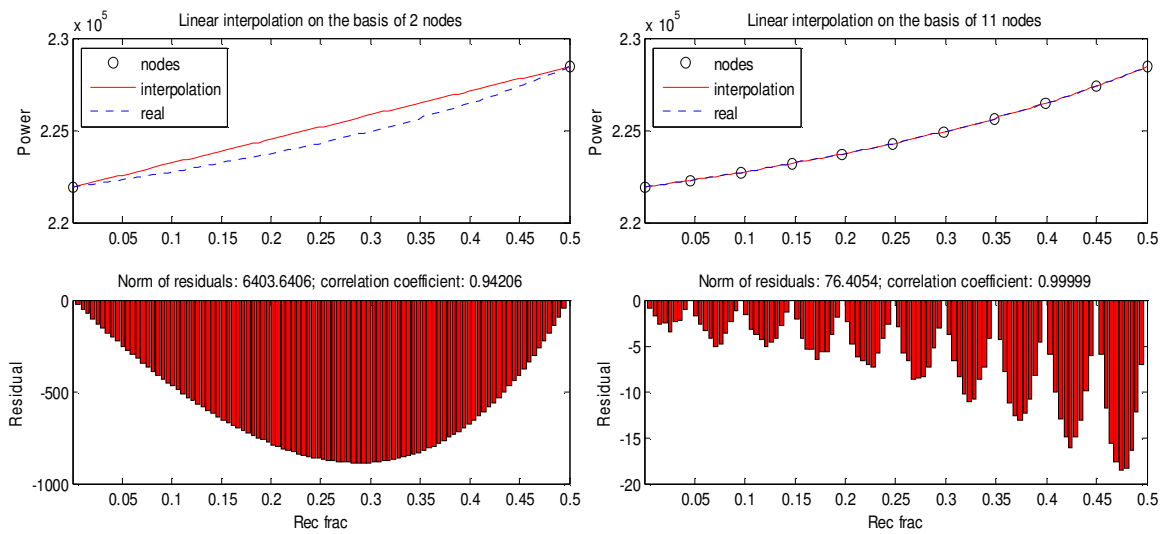


Figure 2.8 Linear interpolation for different number of nodes for Power Output

The cubic spline interpolation results are shown in the Figure 2.9. We can get really good results with this method (the best from all analyzed mathematical methods).

In case of 2 nodes the interpolation function is the same as in the case of linear interpolation. Because of that we will focus on the more complicated cases. For more nodes the errors distribution is less uniform. The interpolation function is continuous (the first and second derivatives are constant in the nodes for both neighboring functions). In the each interval the function has the form of the polynom degree 3. This explains non-uniformly distribution of errors.

The best interpolation results we are getting in the case of 11 modes. The norm of residuals is equal to 3.3 and correlation coefficient is 1.

This interpolation method was chosen to use in the next part of the problem (modeling many inputs problem). It gives the best and the most predictable results from all of the discussed mathematical methods. It is also better choose than cubic Hermite interpolation, because the interpolation real data has the shape of continuous functions and cubic spline interpolation takes care of second derivative in the nodes (cubic Hermite interpolation does not).

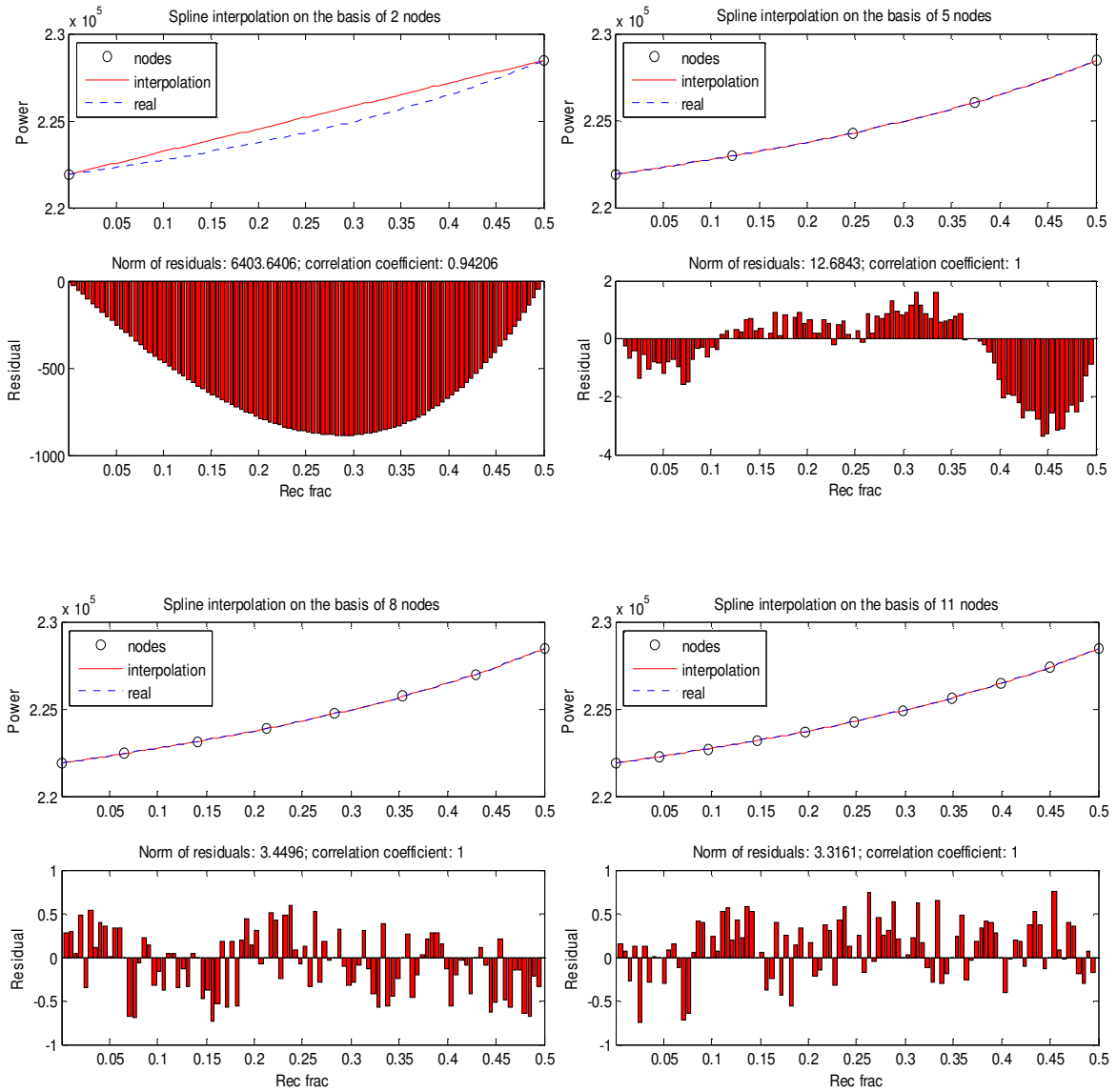


Figure 2.9 Spline interpolation for different number of nodes for Power Output

The next described method is cubic Hermite interpolation. The result of this method are shown in the Figure 2.10. As it was mentioned earlier this method is similar to the cubic spline interpolation, but it does not take care of second derivative in the nodes. Analyzed original data function is continuous in the whole interval, so we are getting slightly worse results than for spline interpolation.

The best results we are getting for 11 nodes. The norm of residuals in this case is 4 and the correlation coefficient is equal to 1.

Moreover on the graphs we can notice that the biggest interpolation errors occur next to the ends of interpolation interval (it can be seen especially well in the case of small number of nodes). It is caused because there is any other data to compare for the extreme points.

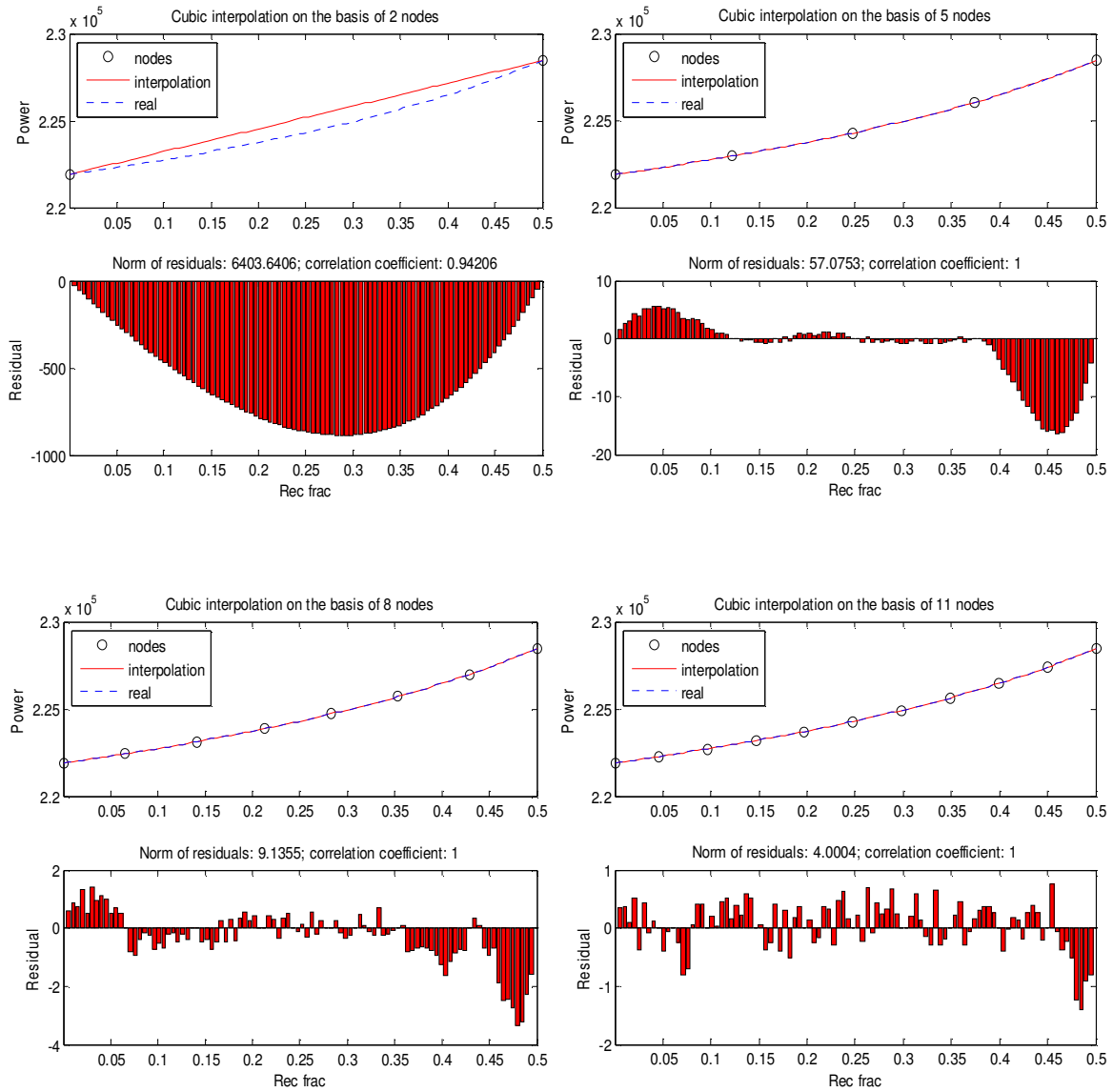


Figure 2.10 Cubic Hermite interpolation for different number of nodes for Power Output

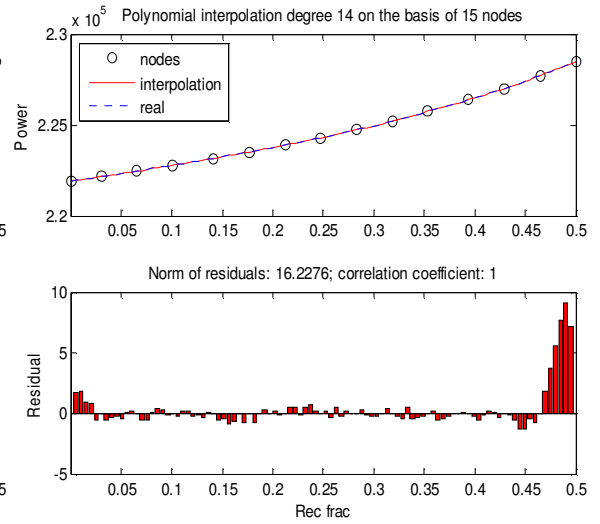
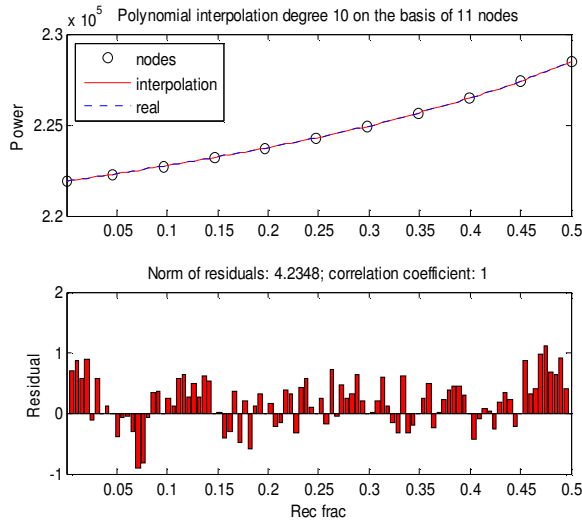
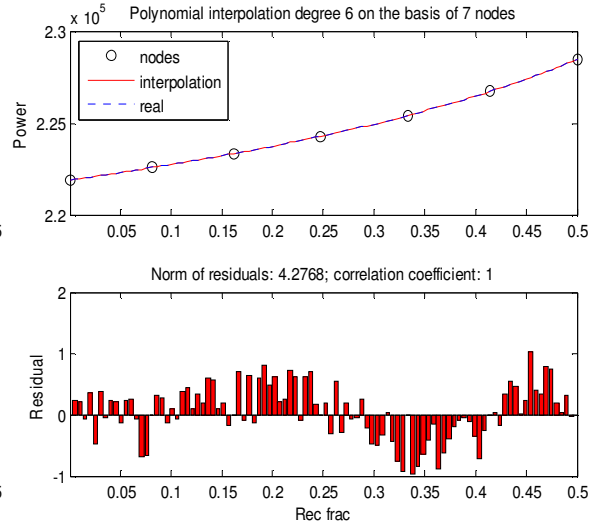
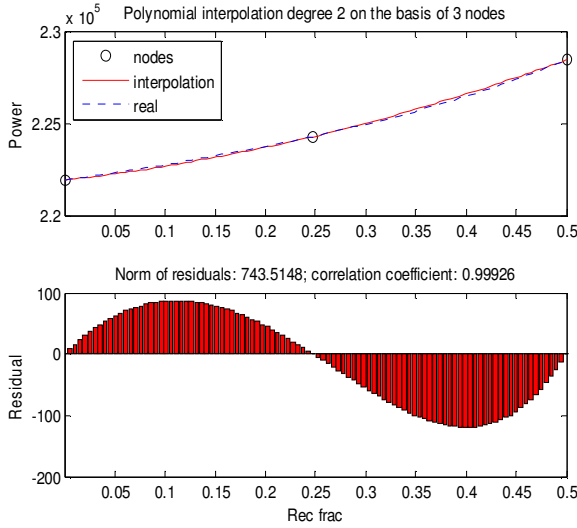
Another method of interpolation is polynomial interpolation. In this case we have continuous function in the whole interval. But in this case a polynomial's degree depends on the number of nodes: if we have many nodes, we need also high polynomial degree. This is resulting in the big error of interpolation in case of many nodes (it can be seen like an over fitting). We have to find a compromise between precision of interpolation and the possibility of Runge effect.

The polynomial interpolation examples are shown in the Figure 2.11. There is six different interpolation polynomials presented:

- 3 nodes (square polynomial);
- 7 nodes (6th degree polynomial);
- 11 nodes (10th degree polynomial);
- 15 nodes (14th degree polynomial);

- 19 nodes (18th degree polynom);
- 23 nodes (12nd degree polynom).

Firstly the interpolation error is getting smaller with the number of nodes. We are getting very good results for the polynom based on 11 nodes. The norm of residuals is 4.23 and correlation coefficient is 1. But from this moment, with growing number of nodes the interpolation error grows fast. Especially big errors can be seen near ends of interval of interpolation. It is good illustration of Runge effect.



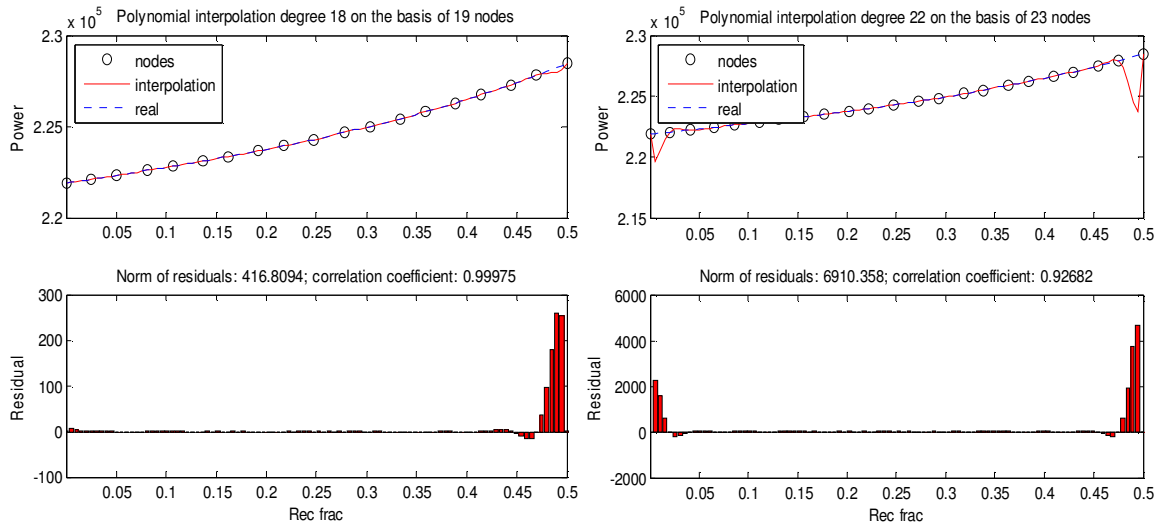


Figure 2.11 Polynomial interpolation for different number of nodes (and different degree of interpolation polynomial) for Power Output

The last mathematical method is polynomial fitting. The biggest difference between this method and previous ones is that the polynomial does not have to have the same value as original function in the nodes. The example of interpolating data with polynomial degree 5 can be seen in the Figure 2.12.

As we can observe on the graphs below the polynomial fitting is giving really good results. The interpolation for 15 nodes is almost as good as spline interpolation (the norm of residuals is 3.41 and correlation coefficient is equal to 1). On the other hand we have to remember that a process of finding interpolation polynomial in this case is much more complicated than in the case of spline interpolation. In spline interpolation we are looking for small degree polynomials for each interval. When we fit a polynomial we have high degree polynomial and we try to fit it closely to the data to minimize the error in the least square sense. Finding the appropriate polynomial is more time-consuming and moreover the polynomial evaluation for given points is also more complicated.

One more thing should be noticed – when the 5th degree polynomial is fitted to 5 nodes it is ambiguous. To fit this kind of polynomial unambiguously we need at least 6 interpolation nodes.

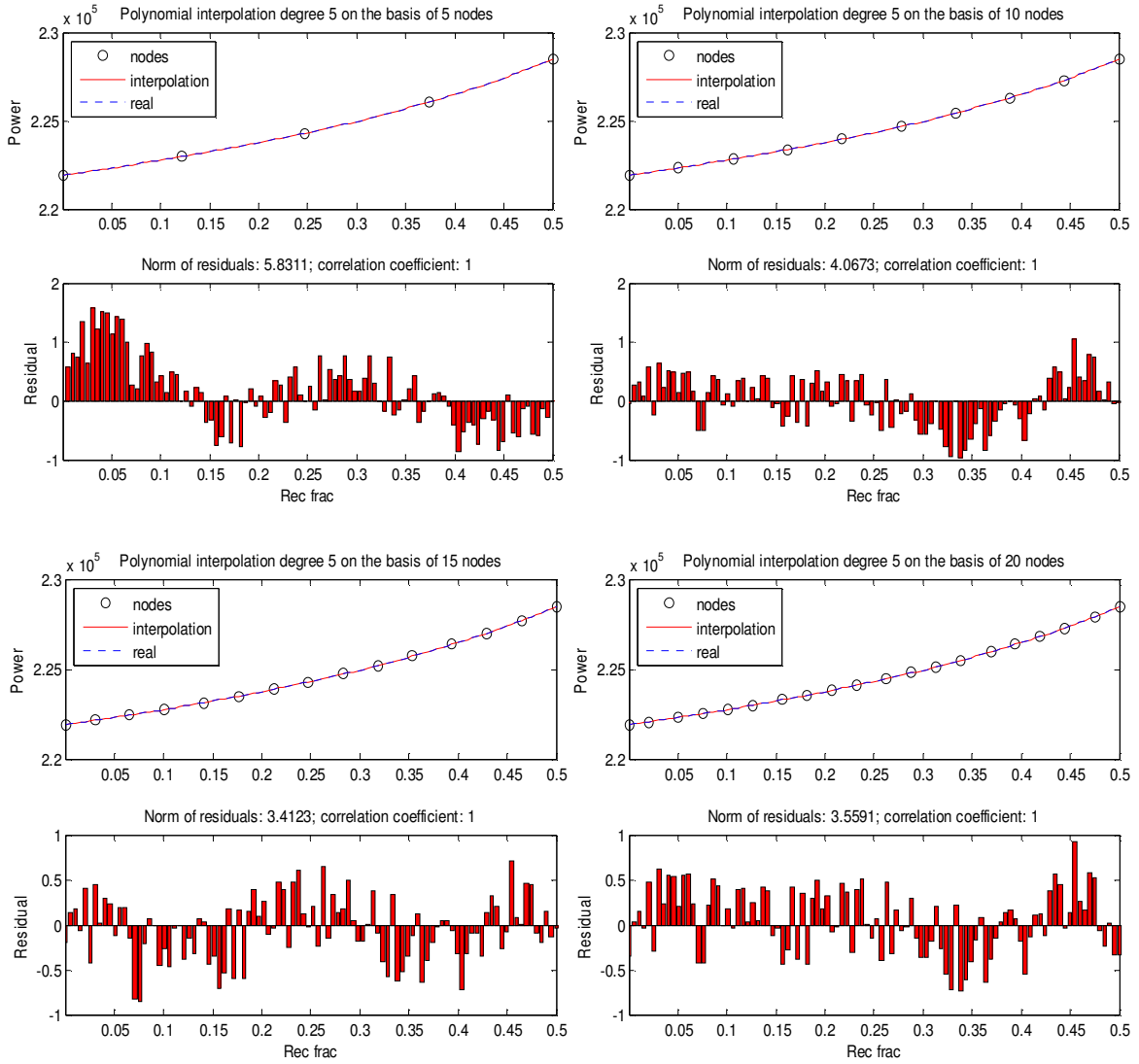


Figure 2.12 Polynomial fitting of polynomial degree 5 for different number of nodes for Power Output

The fitting of polynomial degree 10 is presented in the Figure 2.13. This situation is analogous to the previous one. As before the best results can be observed for 11 nodes (the norm of residuals is 3.4 and correlation coefficient is equal to 1). This results are slightly better than for 5th degree polynomial, but the difference is really small. On the other hand the 10th degree polynomial is even more complicated in using that previous one.

In this case the fitting result is not unambiguous for 5 and for 10 nodes.

For 5 and for 10 nodes fitting we can observe on the residual graph some tendencies. This tendencies shows us the approximate shape of a interpolation polynomial in comparison to the original function. For bigger number of nodes the residuals are more irregular. It is like that because the more nodes were incorporated into the process of creating the interpolation polynomial.

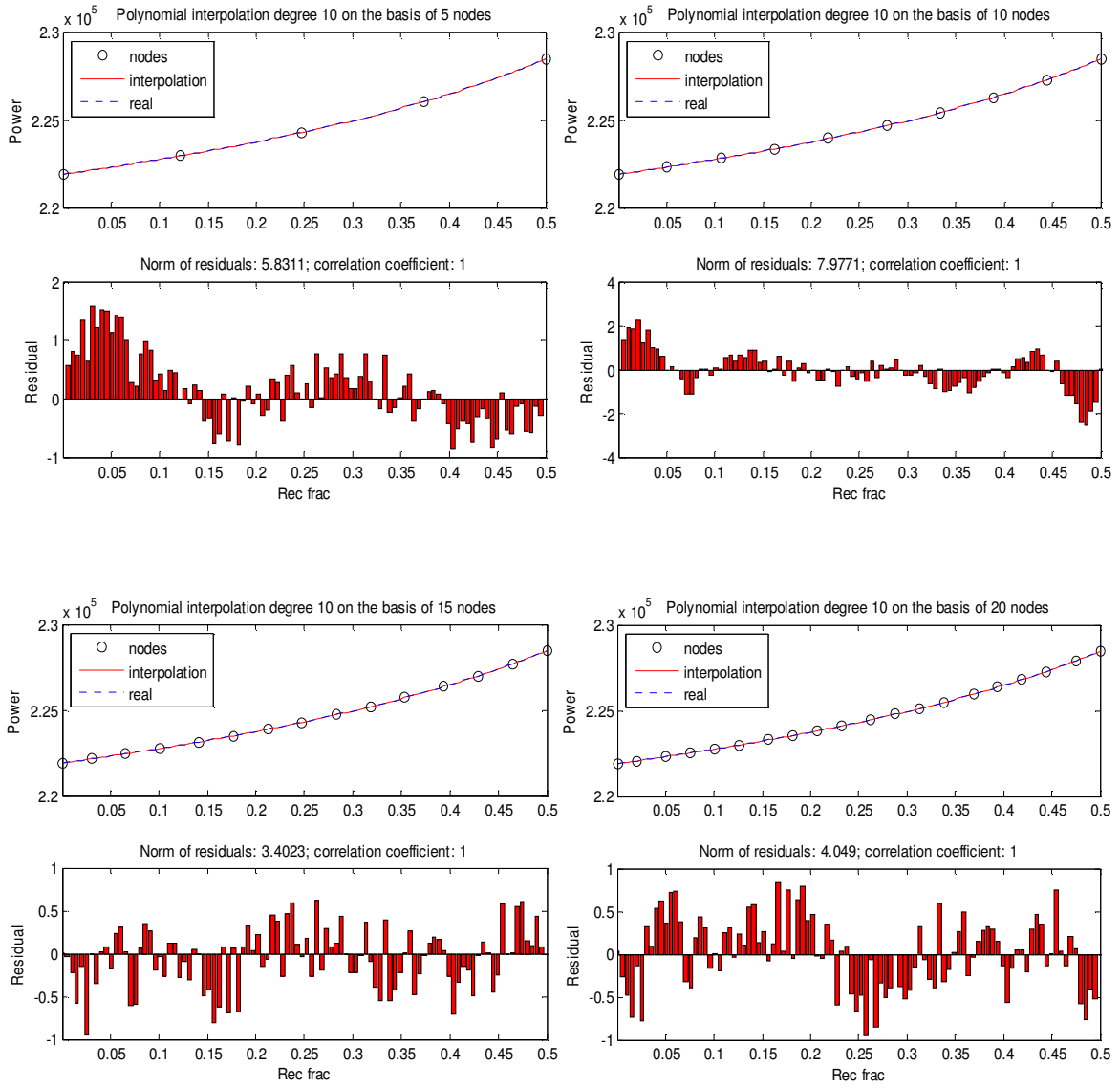


Figure 2.13 Polynomial fitting of polynom degree 10 for different number of nodes for Power Output

Neural network

Second type of surrogate modeling is artificial intelligence modeling. As it was mentioned previously this Thesis is focused on the neural network models. The neural network surrogate solution was implemented for simple problem with one input and one output. As it was shown there are many basic mathematical methods which can be used to model this problem with very good results. The neural network seems to be too complex and complicated tool to use in such easy case. In the real life problems in fact it is better solution to use the mathematical model in case like this one (it is faster and easier to implement).

However, in this Thesis, the neural network model was implemented, tested and analyzed in details. Below the description of different experiments is presented. It is much easier to show many features of neural network on the example of simple problem with one input and one output. In this case those features can be understood by intuition – for more

complicated cases they are not so obvious. This subchapter is an introduction to more complex problems, which will be presented in the next chapters.

There was many experiments done, but below only 4 of them are presented in the details.

For all of the presented experiments there was 5 points in the training set, 2 points in the validation set and 93 points in testing set. All of them were randomly chosen from set of 100 equidistant points generated in the beginning. The experiment with other division for different set was also done and their results will be briefly described at the end of subchapter. For all of the experiments the maximum number of iteration was equal 1000.

The design of the used neural networks is presented in the *Figure 2.14*. In three first experiments the neural network was built from one input, 2 neurons in hidden layer, 1 neuron in output layer and one output. In the fourth experiment the hidden layer had 10 neurons.

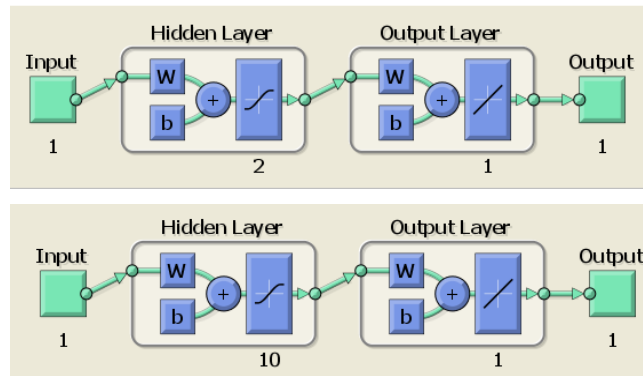


Figure 2.14 Design of the neural Network with 2 neurons (experiment 1,2,3) and with 10 neurons (experiment 4) in hidden layer (from MATLAB interface)

The results of four experiments are listed in the

Table 2.2. The most important data are norm of residuals and correlation coefficient which allows for evaluation of each model and also for comparison neural network models with mathematical ones.

Table 2.2 Properties of different neural network surrogate models

<i>Experiment</i>	<i>Neurons in hidden layer</i>	<i>Iteration number</i>	<i>Time [s]</i>	<i>Norm of residuals</i>	<i>Correlation coefficient</i>
1	2	1000	22	12.37	1
2	2	10	1	2511.23	0.989
3	2	7	1	9124.22	0.743
4	10	6	1	12972.21	0.844

Table 2.3 contains data for each of sets of points (training, validation, test and all). There are presented the values of mean squared error MSE and Regression value R.

Table 2.3 Neural network surrogate models: MSE and Regression R values

Experiment		Training	Validation	Test	All
	Points	5	2	93	100
1	MSE	0.94	0.62	1.58	1.58
	Regression R	1	1	1	1
2	MSE	2519.41	307.872	67667.61	67667.61
	Regression R	1	1	0.997	0.997
3	MSE	1330765	4456,54	823533,8	823533,8
	Regression R	0.997	1	0.902	0.896
4	MSE	0	6101841	1678220	1678220
	Regression R	1	1	0.889	0.889

Now we will focus on the individual experiments. Each of them lets for illustrating some interesting features of neural network.

The first experiment gives the best results. It shows that with appropriate neural network and the good chosen training and validation set we can built a good model. During the experiments sometimes even better results was achieved, but this example is representative one. We have to remember that the quality of neural network is in part a random value. It strongly depends from the staring weights in network, as well as from the distribution of the points from each set among the whole interval.

Figure 2.15 shows the real point and results from neural network surrogate model. We can notice that there is almost no difference between those two sets. The norm of residuals for this case is equal 12.37 and the correlation coefficient is 1. On the graph we can also observe the distribution of the training and validation points. As it can be seen the training points cover the whole interval. If we would choose better training points (for example equidistant ones) the final results would be even better.

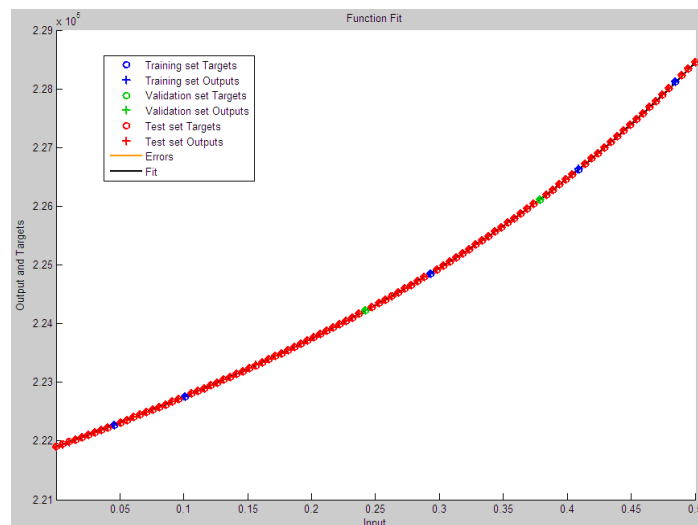


Figure 2.15 Fit graph for Experiment 1

The regression graphs for the Experiment 1 are shown in the Figure 2.16. They show differences between real results (targets) and output of the neural network. The graph presents also the regression line for each data set. If this line connects left down and right upper edges of the graph, it means that the network gives good results. If the point lies in the distance of that line it means that there are mistakes in the surrogate model (for a perfect fit, the data should fall along a 45 degree line, where the network outputs are equal to the targets).

If we want to improve the results given by the neural network we should retrain the existing one. The new training process starts with the weights from previous one. So the better results can be achieved. It can help especially in the situation when the training process ended as a result of achieving maximum number of iterations (not because of validation checking).

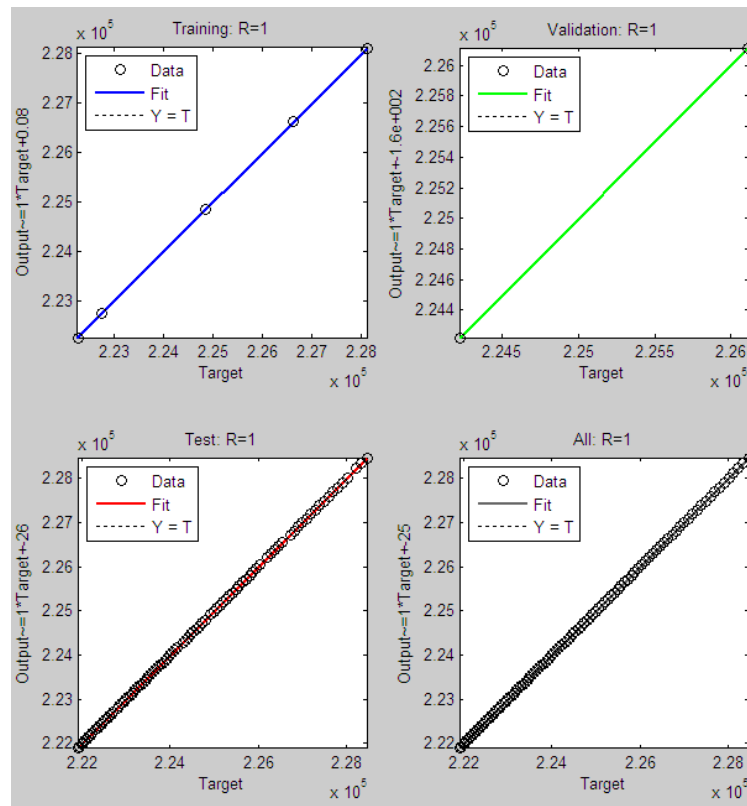


Figure 2.16 Regression graph for Experiment 1

The last two graphs presents details about training process – they are shown in the Figure 2.17 The first graph shows performance of the training, validation, and test sets. The second graph presents the details of network parameters in each epoch. In first experiment there was 1000 epochs (maximum allowed number). It means that the stop condition was not reached (the MSE for validation set should be constant or bigger than best value during 6 consecutive epochs). There is one more thing which should be noticed. The difference of MSE for all point in first and last epoch – it changed from about 10^8 to 1.

The accuracy for this network is very good, but the training process took 22 seconds, what is very big number for such a simple problem like this one. The good think is that the time

of training does not depends linearly from the complexity of the problem (sometimes we can get really good results for complicated models in relatively short time).

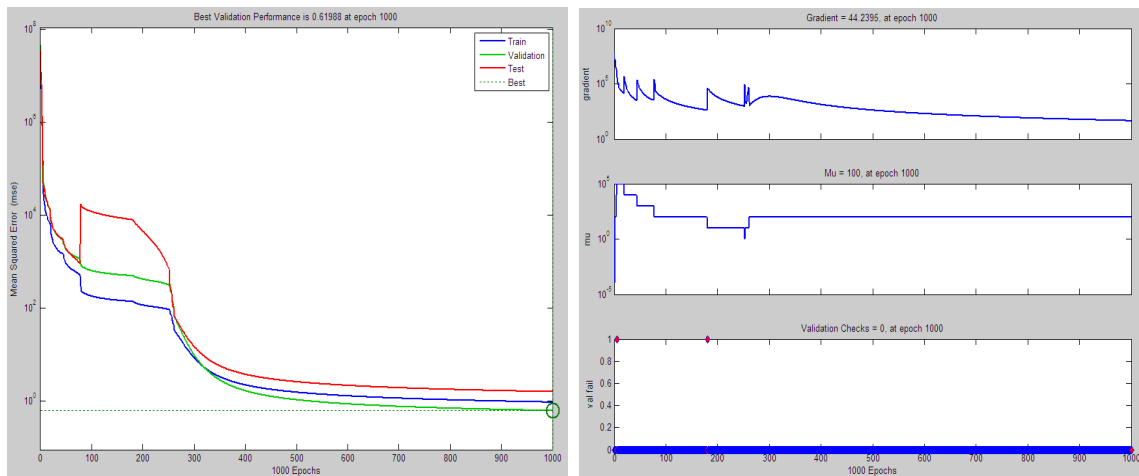


Figure 2.17 Performance and Training State graphs for Experiment 1

In the next experiment the training set did not cover the whole interval. There is no training point in the left end as it can be seen in the Figure 2.18. It can be noticed also that in left part the differences between real data and network output are the biggest. In the process of learning the network was basis on the connection between training points, which does not corresponds to the relation between points in the left part of interval.

The smallest value of MSE is achieved for validation set, but the value for training set is also small in relation to MSE of test set.

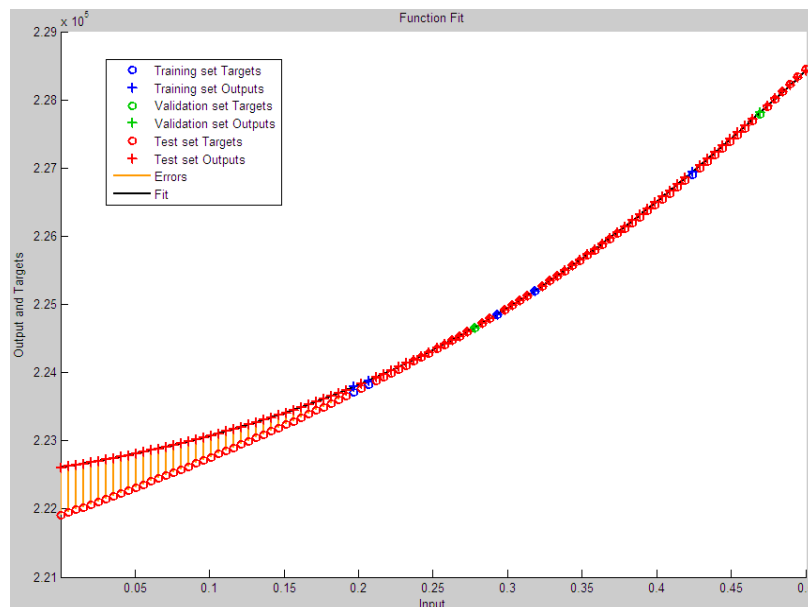


Figure 2.18 Fit graph for Experiment 2

In the Figure 2.19 we can see that test set the regression line does not cover the 45° line. It means that there are mistakes in the surrogate model. The same situation can be observed on the regression graph for all points.

To achieve better results we should check which points gives us the worst results and add the points from their neighborhood to the training set.

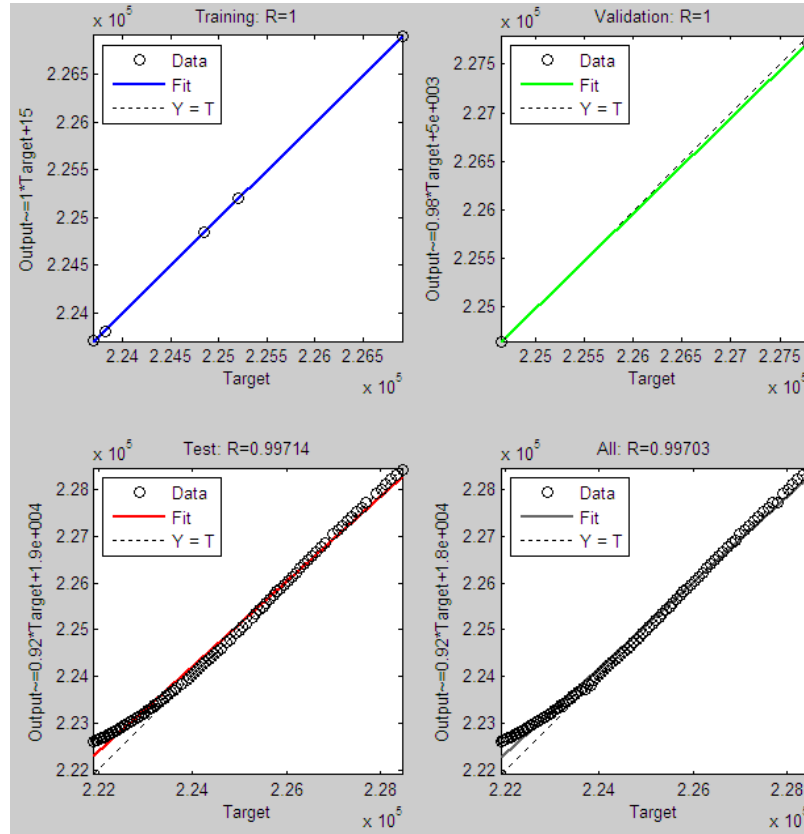


Figure 2.19 Regression graph for Experiment 2

Figure 2.20 shows detail information about the training process. There were only 10 iterations of training algorithm. The validation error started growing in 4th epoch. This behavior could be changed by adding more points to training set as by adding points to validation set.

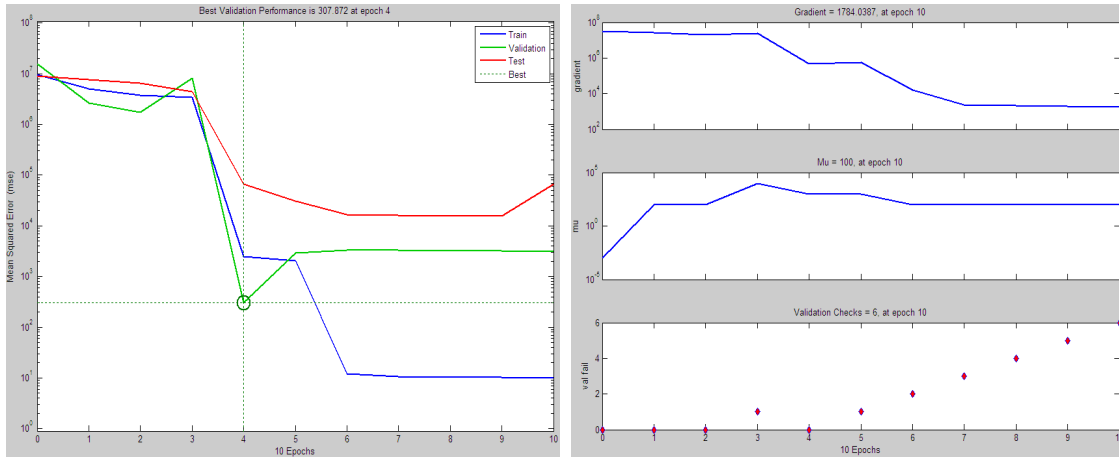


Figure 2.20 Performance and Training State graphs for Experiment 2

The next analyzed case is presented in the Figure 2.21. The training points are focused in the middle of interval. In this case two factors had an influence for big surrogate model's error: bad training points distribution and bad starting weights.

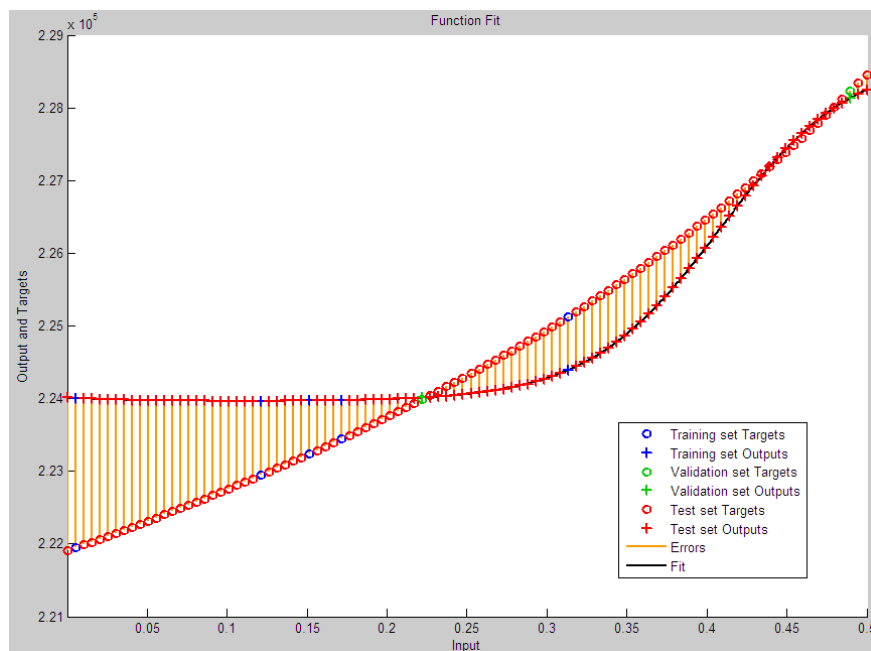


Figure 2.21 Fit graph for Experiment 3

Any of the regression graphs presented in the Figure 2.22 does not have appropriate shape. Big errors can be seen in each part of the interval.

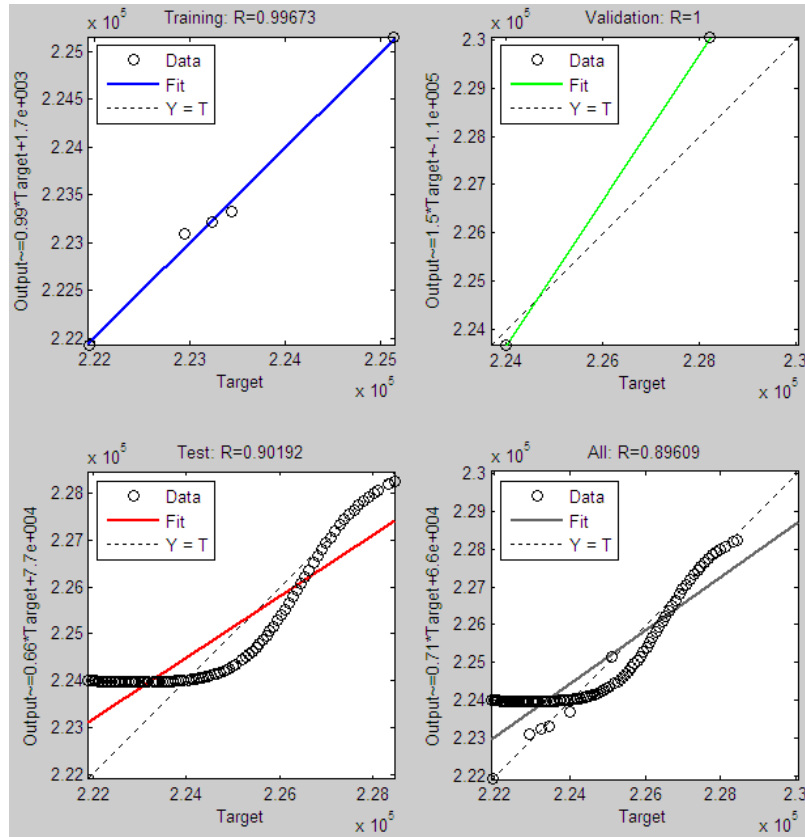


Figure 2.22 Regression graph for Experiment 3

In this case there were only 7 epochs and the detailed information about each of them are presented in the Figure 2.23.

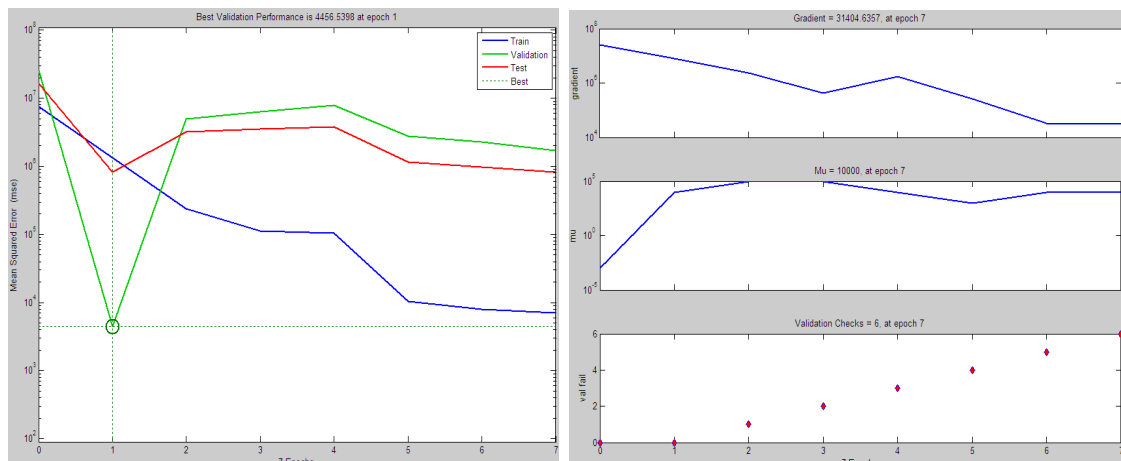


Figure 2.23 Performance and Training State graphs for Experiment 3

The interesting results can be observed in the last experiment. In this case there was bigger number of neurons in the hidden layer (10). It is a big number in comparison to only one input value. The results show that neural network is overfitted. It gives great results for training set, but for other points MSE values are huge. The fit graph for this experiment is presented in the Figure 2.24.

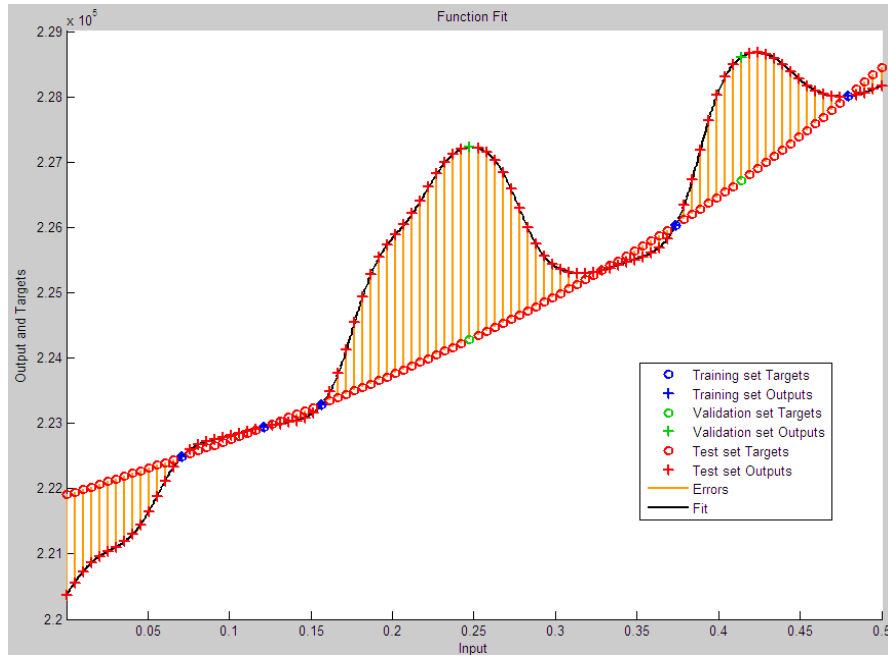


Figure 2.24 Fit graph for Experiment 4

Figure 2.25 shows the regression graphs for the last experiment. As it was said before in the training set the points lay on the perfect fit line. For other sets there are big fluctuations.

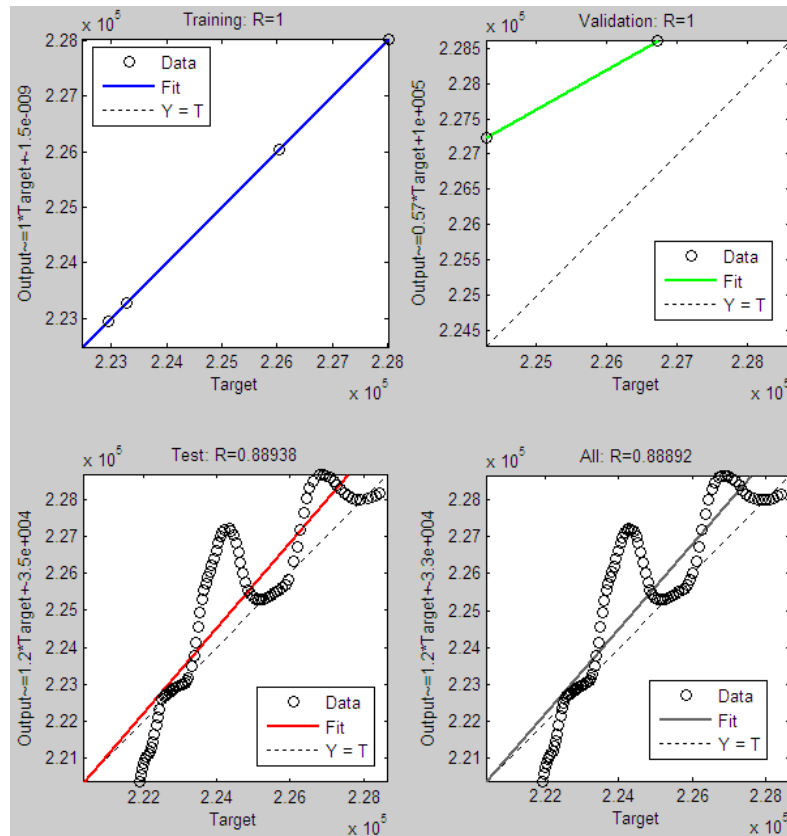


Figure 2.25 Regression graph for Experiment 4

The last graph in this paragraph, Figure 2.26, shows the details of training process for fourth experiment. It is significant that the training error decreases very fast and validation error is constant. Because of that there is only 6 epochs.

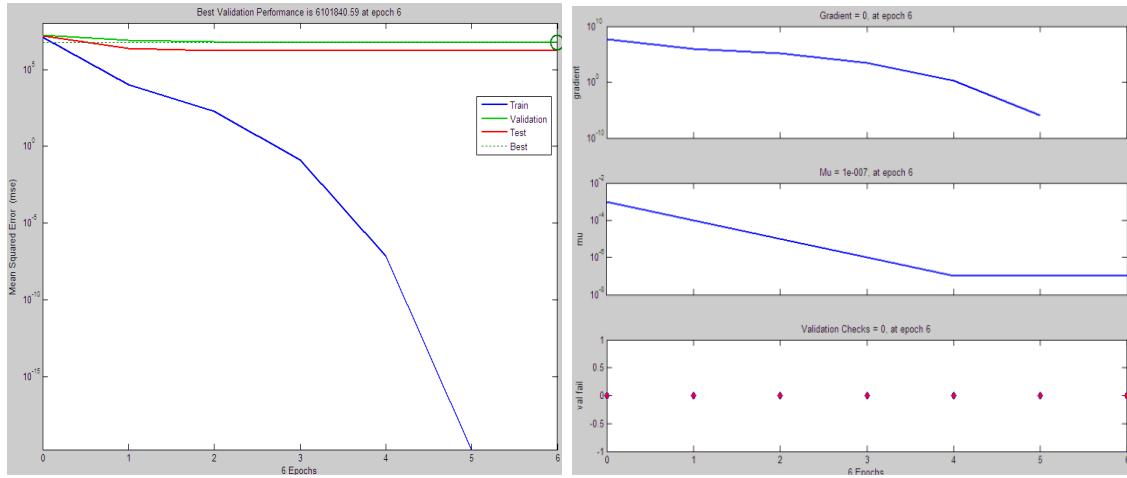


Figure 2.26 Performance and Training State graphs for Experiment 4

This problem is great illustration of overfitting effect: the performance of the training set is good and in the same time the test set gives big errors. The solution which can improves results is decreasing the number of nodes.

On the other hand if training performance is poor we should try to do one of two things:

- Increasing number of points in training and validation set;
- Increasing number of neurons in hidden layer.

We should remember that bigger number of neurons in hidden layer let network to solve more complicated problems. On the other hand it can cause an overfitting effect and require more computation.

The same rule is true also for bigger number of hidden layers: more layers require more computation but they allow for solving complex problems.

During practical analyzes of this problem there were created more networks than presented four. The different numbers of points in training and validation sets were tested. The conclusion is that the quality of surrogate model improves with bigger number of training and validating points. But much more important is that the points should cover the whole interval. For properly distributed points their number does not have so big impact for final results.

2.4.2 Results for selected methods for all output parameters

After detailed analyze of one input, one output surrogate model there was a time to build the model which take into account all of input parameters. There were chosen two methods: spline interpolation and neural network.

Spline interpolation

Firstly the mathematical models for all input parameters for 4 knots were built. The result of this interpolation are shown in Appendix 1. As it can be seen in some of the cases there are differences between real data and function output. It is caused by very small number of interpolation knots. We choose the worst interpolation model – for input parameter F_{421_T} . It is shown in the Figure 2.27. The correlation coefficient for this case is equal to 0.98. As it can be seen the biggest interpolation error is in the last interpolation interval (between two last nodes), because there is the biggest slope of original data.

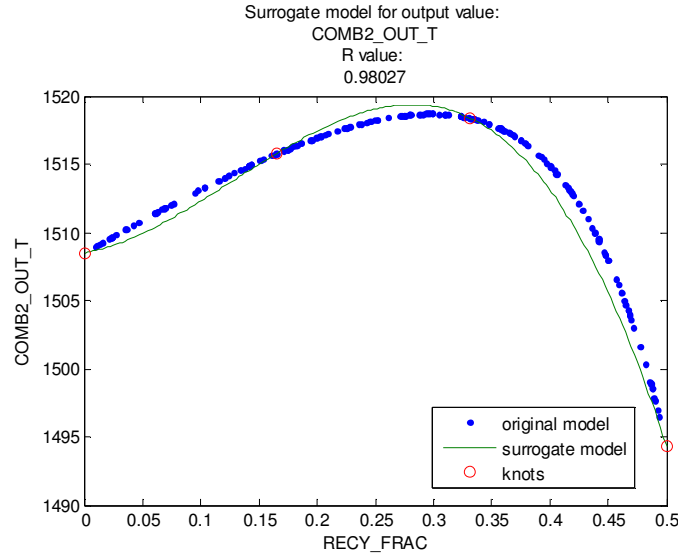


Figure 2.27 Surrogate model on basis of 4 knots for F_{421_T}

As it was said the reason of the mistakes is the small number of interpolation knots. It was checked how the increasing this number influences the quality of the model. In the Figure 2.28. For 6 knots the correlation coefficient is equal to 0.999. So it is quite good result.

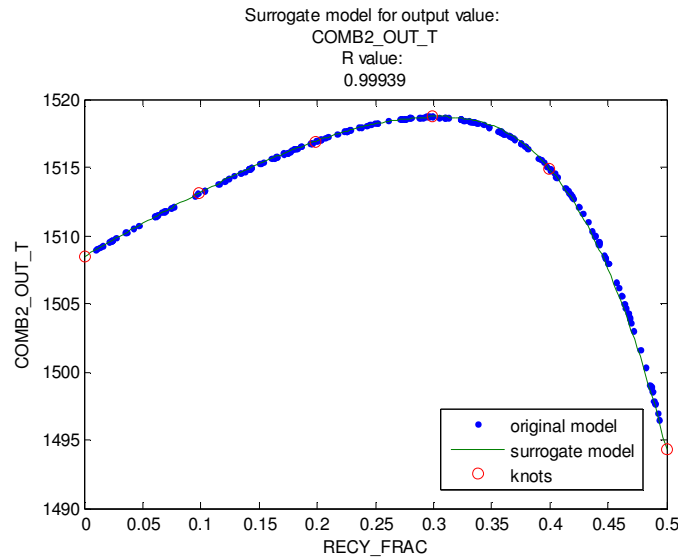


Figure 2.28 Surrogate model on basis of 6 knots for F_{421_T}

We checked how many knots is needed to have correlation coefficient equals 1. The result is shown in the Figure 2.29. So for the input parameter, which gave the worst result in first interpolation, we need 11 knots to have really good results. In this case there is 10 intervals of interpolation (each one of them is 0.05).

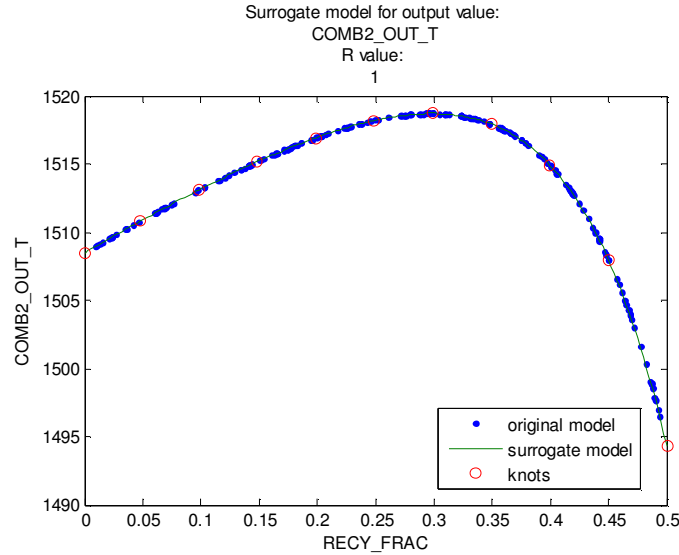


Figure 2.29 Surrogate model on basis of 11 knots for F_{421_T}

In the figures: Figure 2.30, Figure 2.31, Figure 2.32 there are presented correlation coefficients for all model output parameters, ordered from the worst one. It can be seen that for 4 knots all coefficient are better than 0.98. For 6 knots this number is much better – all coefficients are bigger than 0.999 and many of them is even equal to 1. For 11 knots on the other hand all coefficients are equal to 1.

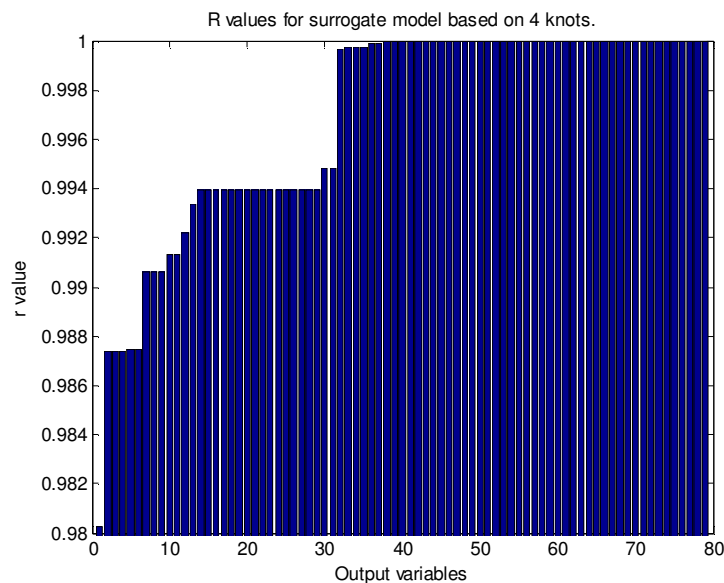


Figure 2.30 R values for spline surrogate model based on 4 knots for all output parameters (ordered from smallest one)

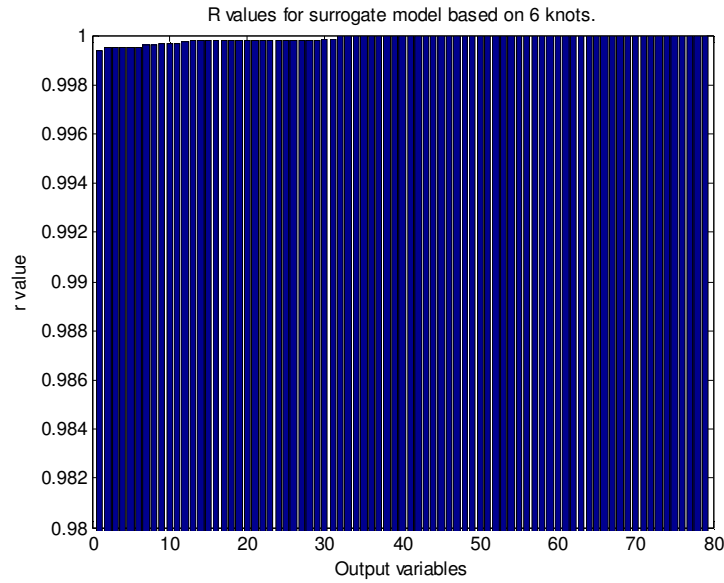


Figure 2.31 R values for spline surrogate model based on 6 knots for all output parameters(ordered from smallest one)

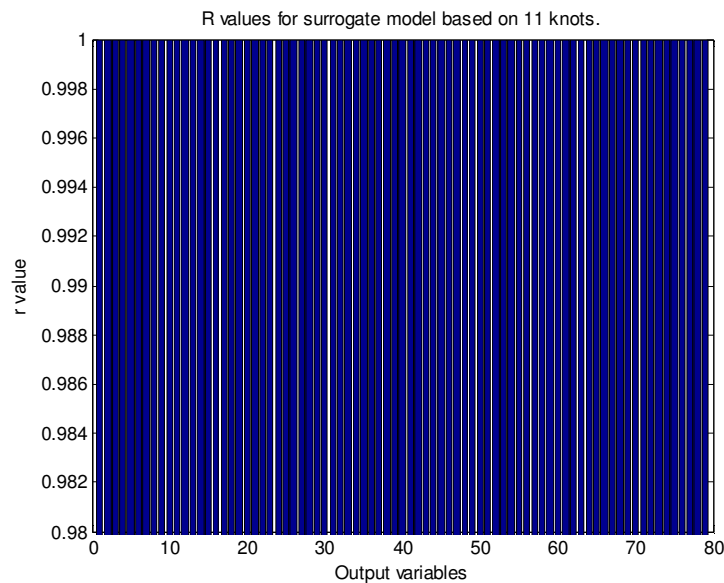


Figure 2.32 R values for spline surrogate model based on 11 knots for all output parameters(ordered from smallest one)

The final version of the spline interpolation surrogate model which was prepared to use in real-life model bases on the 11 knots – it makes surrogate model high quality and efficient.

Neural network

The last stage of surrogate modeling one decision variable problem was building neural network model for many output parameters. There was build one network for all of them: network is build with one input, 3 neurons in hidden layer, 79 neurons in output layer and 79 outputs. Figure 2.33 shows it design.

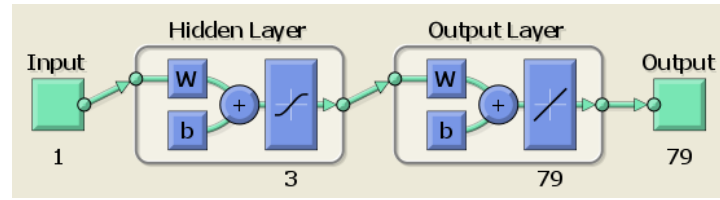


Figure 2.33 Design of the neural Network for one input, many output problem (from MATLAB interface)

All real output data was normalized. If the raw data would be used, some of the output parameters influences the final result more, some less (it depends from their absolute values). Moreover the neural network, which was used in this analysis, is better suited for parameters in range $[-1,1]$. From the set of all data, 20 samples was chosen randomly for training set and 10 for validation set. The rest was used to testing the network.

The time of training network was long – it was over 7 minutes. There was done 1000 iterations of the algorithm (Table 2.4).

Table 2.4 Properties of neural network surrogate models for one input, many output problem

Neurons in hidden layer	Iteration number	Time
3	1000	7 min 11s

Table 2.5 contains data for each of sets of points (training, validation, test and all). There are presented the values of mean squared error MSE and Regression value R.

Table 2.5 Neural network surrogate models for one input, many output problem: MSE and Regression R values

	Training	Validation	Test	All
Points	20	10	170	200
MSE	$1.8 \cdot 10^{-7}$	$2.09 \cdot 10^{-7}$	$6.39 \cdot 10^{-7}$	$6.39 \cdot 10^{-7}$
Regression R	1	1	1	1

The regression graphs for the neural network are presented in the Figure 2.34. The network generation time was very long but it resulted in high quality network. The points from all sets cover the best fit line.

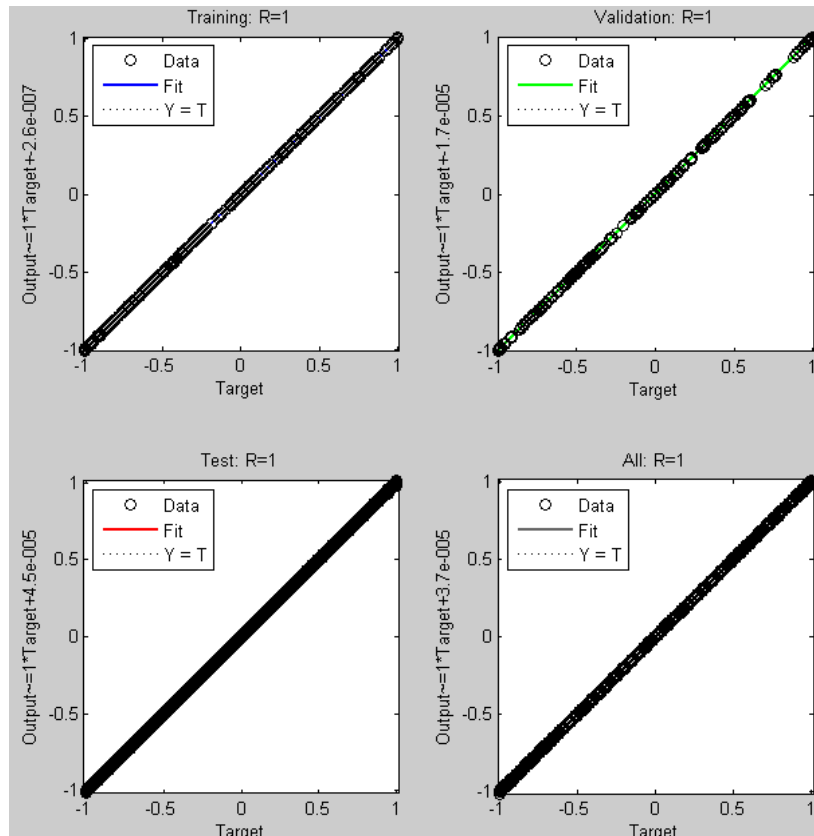


Figure 2.34 Regression graph for one input, many output problem

The process of the network training can be seen in the Figure 2.35. We can notice that MSE value firstly drops very rapidly for all sets and then it approaching slowly the minimum. The stop condition was not fulfill in this case. We can notice, that we could stop training process much earlier. In future research introducing the new stop condition should be considered. The process of training the network can be stopped when MSE has smaller value then requested limit.

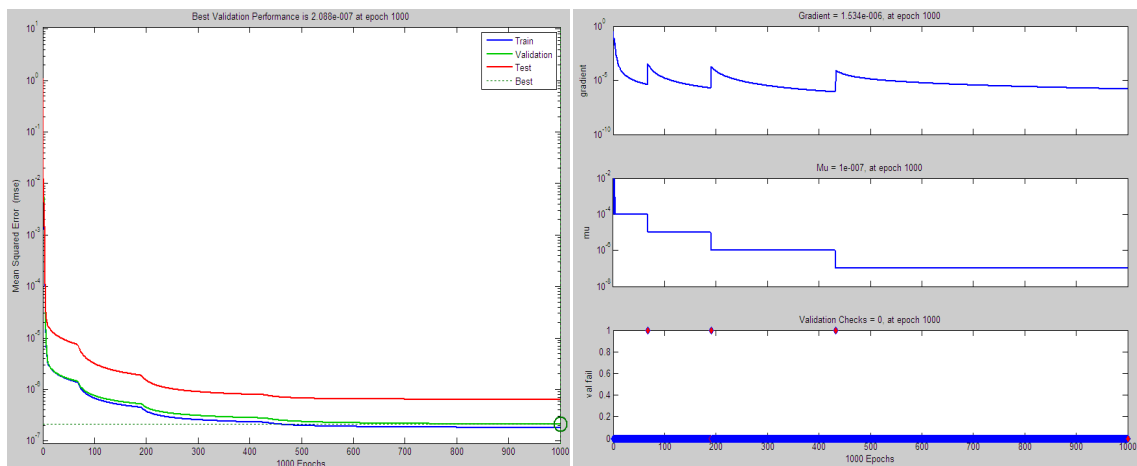


Figure 2.35 Performance and Training State graphs for one input, many output problem

Figure 2.36 shows the results of additional test. Created neural network was tested on the set of 200 random points. As it can be seen the results are as good as before.

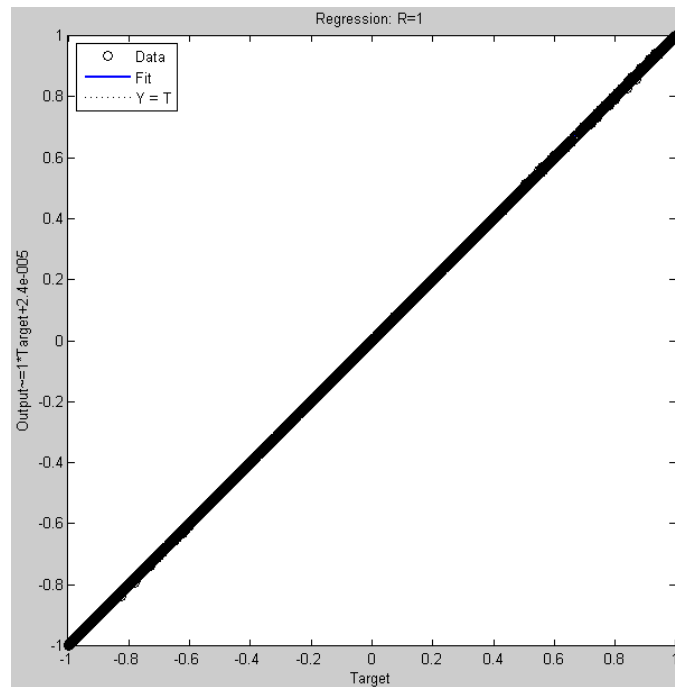


Figure 2.36 Regression graph for additional test set.

We should remember that the quality of the network was evaluated for all parameters together. If this model would be used in real life problem we should also check the quality of each output value. To do it we can for example calculate the norm of residuals and correlation coefficient for all from output parameters separately. It was checked that for all of the output parameters norm of residuals is smaller than 0.05 (remember about normalization of the output values). All of the correlation coefficients are equal to 1.

2.4.3 Interpretation of results

In this subchapter several physical aspects of the model will be described.

All of the modeled tags, as well as analyzed system, were described earlier. The tags were chosen to be modeled on the basis of the other parts of the model – it means that we modeled all the tags which are used in other parts of the system.

During the process of choosing tags it was not checked if some of them do not have constant values. In fact it turned out that some of them are imposed as constant. The examples of these tags can be some temperatures, as for example COOL_OUT_T tag, or ratio of CO2 captured by MEA process. It was checked that also for this variables surrogate model gives good results – there is any additional disturbances or oscillations.

All of the interesting results (not imposed as a constant) are presented in the Appendix A. On the basis of those graphs we can analyze the influence of recirculation ration for different output parameters.

Below we will show the example of such analysis. We will focus on the net electricity production:

- The gas composition depends on the FGR and syngas injection. This processes decreasing $\kappa=c_p/c_v$ of the working fluid. It influences power produced by the turbines and power consumed by the compressor. Both of them are lower with increasing FGR;
- The LHV of fuel mixture (lower heating value) decrease has no effect since the inlet temperatures of gas turbine are fixed and the pressure drop of fuel injection is not considered. The increasing fuel mass flow compensates the lower LHV;
- H_2 injection with increasing FGR changes the inlet mass flow of the gas turbine. The mechanical power increases proportionally, what compensate and overweight the calorific factor decrease effect;
- With increasing value of FRG we need to consume more energy to produce syngas.

As it was shown there are many parameters in the system which influence the final shape of the dependences. Interpretation of those connections is not a trivial task.

3 MULTI DECISION VARIABLE MODEL

3.1 Flow sheet and model description

The second practical problem analyzed in the Thesis is connected with the previously described system. The whole problem is to model the power plants with post-combustion CO₂ capture, consisting of: a gas turbine with flue gas recirculation, a CO₂ capture unit (chemical absorption with amines) and a steam network (Dubuis & Tock, 2010). The whole system is presented in the Figure 3.1.

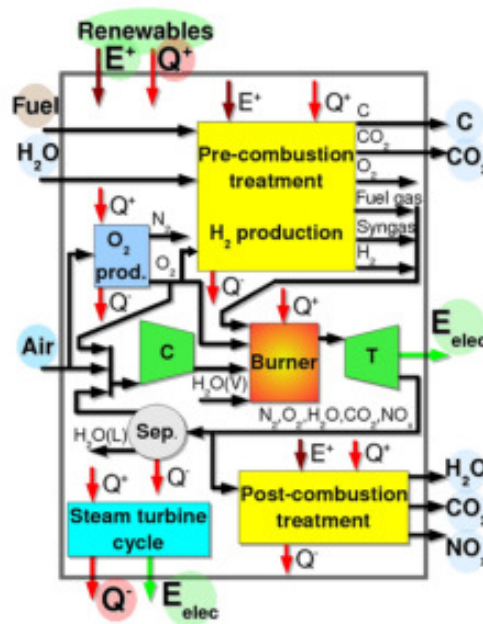


Figure 3.1 Superstructure of gas turbine power plant with CO₂ capture (Dubuis & Tock, 2010)

The first three parts: syngas production, gas turbine with flue gas recirculation and a CO₂ capture unit have been described in the previous chapter. In this part of the work we will focus on the last part of the system: steam network and more globally utilities conception.

In the system there are several additional hot and cold streams at different temperature levels from the rest of the process. The idea is to recover as much heat as possible from hot streams to feed cold streams and produce power. As a cold source, a cooling water system is included as well as an eventual boiler if hot demands can't be covered by steam network.

There are two possible methods of modeling a steam network.

First of them is process flowsheet simulation. In this case we are preparing the detailed model of the system. It lets for building very precise and transparent simulation. On the other hand this is very time-consuming method in evaluation as well as in preparation. Moreover this solution is not flexible when we want to optimize the complex

superstructure and need one model per configuration. It is hard to apply this model to complex Heat Exchanger Network.

Second approach is the process integration techniques. It allows to include heat sink and source in the steam network design. Moreover, it allows to represent every configuration (steam injection or draw off ...). This method is much more effective – it is less time and memory consuming what allows for better optimization. One of the biggest advantages of this method is easy adding of the new streams to the problem.

The model used in the Master Thesis is based on the second approach. This method lets for relatively easy optimization of the level of different pressure and temperatures involved in the steam network. It allows also for good approximation of the generated power, efficiency and costs of the total network.

Figure 3.2 presents the superstructure of the steam network. We have to remember that it is only the representation of the steam network, which is useful to optimize it (for example the stream L – let down flow – is introduced to help solver to converge).

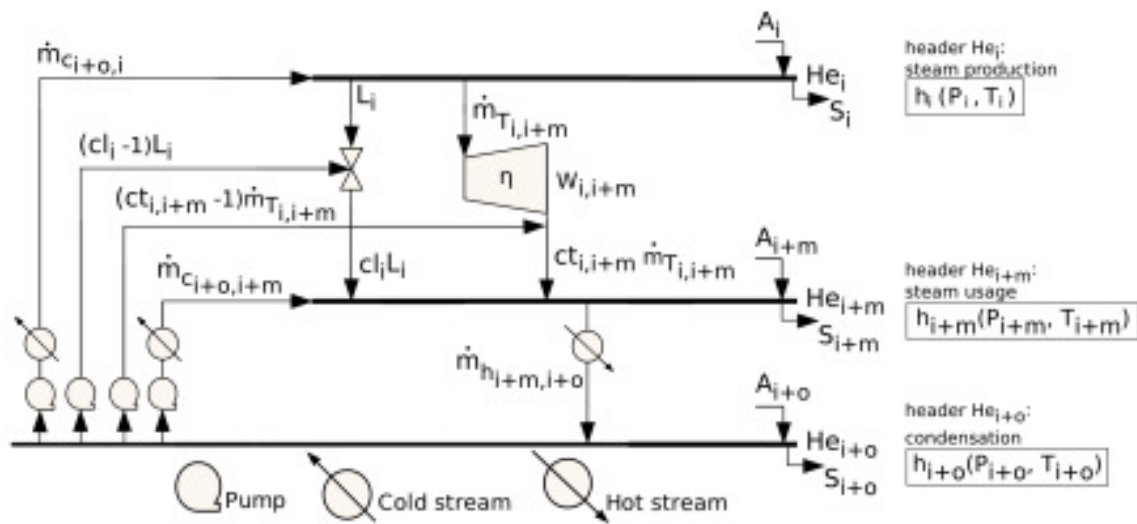


Figure 3.2 Superstructure of a steam network with one expansion level and heat consumption and rejection (Girardin, Dubuis, & Marechal)

When we focus on the structure of steam network we can notice important elements which define the system:

- Headers in the system are defined by their properties: temperature, pressure, steam quality ($x > 0.85$) and their type. Each header is defined as one of the following:
 - Production header in which steam is injected. In the Figure 3.2 it is represented as header i ;
 - Usage header which can receive or distribute steam. It is represented as header $i + m$;

- Condensation header in which steam is condensed. After that the steam is sent back to Heat Recovery Steam Generator. In the Figure 3.2 it is represented as header $i + o$.
- The different mass flow rates involved in:
 - Steam turbine (m_T);
 - Let down flow (L);
 - Steam recovered from heat reject (A);
 - Steam consumed by heat sink (S).

Steam injections (A) or extractions (S) can in the header be understood as the energy demand or additional processes (like for example the devices to CO₂ capture).

The simplest correct model of the steam network has the following elements:

- One steam production level;
- One condensation level;
- Minimum pressure level observed in the list is a condensation header.

The performance indicators desired for the optimization process in this system were cost and efficiency of the system. In the surrogate model also those two parameters will be used as the targets values. In the model we can optimize the flowrates as well as several other values: pressure, temperature and number of different header of a steam network.

The thermodynamic efficiency of the cycle can be calculated from the following formula:

$$\eta_{steam} = \frac{\dot{Q}_{HRSG} - \sum S + \sum A}{\dot{W}_{tot}}.$$

On the other hand to calculate the value of the isentropic efficiency of the steam expansion between two pressure levels we can use (Dubuis & Tock, 2010):

$$\eta_{k,k+1} = 0.919 - 0.549 \cdot \left(1 - \frac{P_k - P_{k+1}}{P_k} \right).$$

The optimization of the system was prepared and described in (Girardin, Dubuis, & Marechal) and (Dubuis & Tock, 2010). Some of the results are presented below. Those graphs can introduce several concepts about the Steam Network and make them easier to understand.

Firstly the Integrated Composite Curves for the Steam Network integrated with MEA and syngas production process are presented in the Figure 3.3 and Figure 3.4. They show respectively the examples of bad and good design of Stem Network.

The first example shows the system with one steam turbine. Steam is produced at 100 bar and 50 bar; the condensation is at 5.7 bar. In the system we do not go below 5.7 bar because of feeding MEA process. Presented system produces 57 MWe. As it can be seen in the Integrated Composite Curve graph this design is not good integrated and there are big losses in the system.

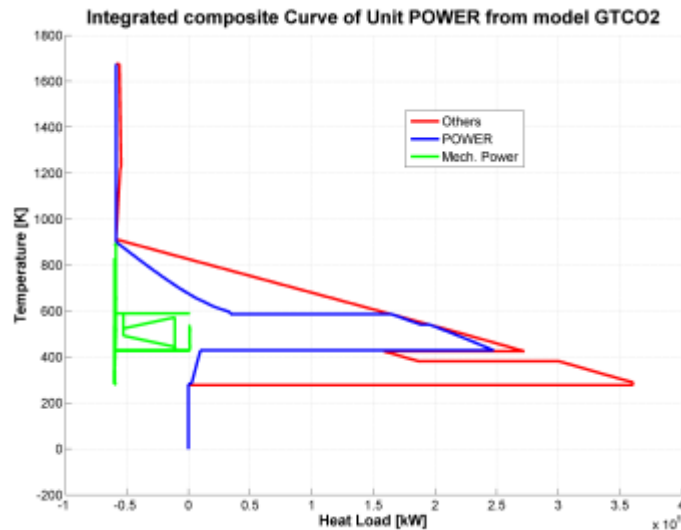


Figure 3.3 Bad example of Steam Network Configuration (Dubuis & Tock, 2010)

The second example, on the other hand, shows system which is much better integrated. In this system the total power production is equal to 89 MWe. The system also consists in one steam turbine. The condensation in this case is from 200 bar to 0.04 bar. The steam is produced at 50 bars. Moreover there is also draw off to feed MEA in the system at 5.7 bar.

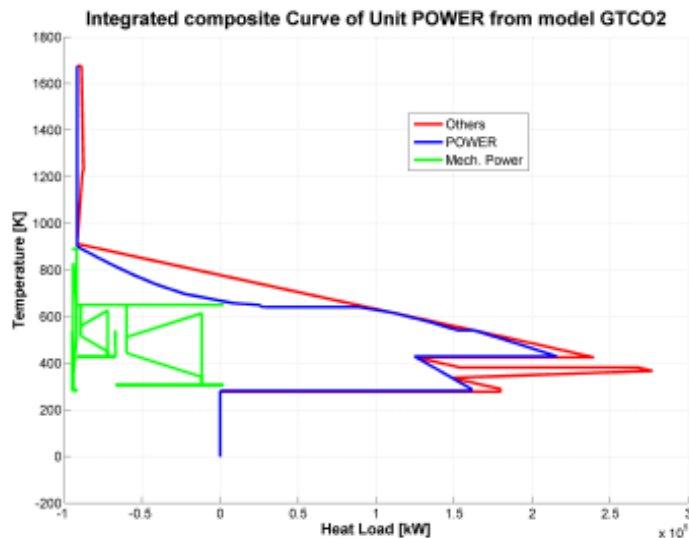


Figure 3.4 Good example of Steam Network Configuration (Dubuis & Tock, 2010)

Those examples show how important is to find the optimal parameters of the system.

Below the examples of the optimization for described system are shown. Actually those graphs shows only the results which can be obtained as a result of optimization process. We will show where the surrogate model can be used.

Graphs show the visualization of three typical steam network configurations. As a objectives functions the investment cost and total power production are assumed. The results are presented in the Figure 3.5. The base case (Case 3 in the graph) is designed as two steam turbines model (working in ranges 60-0.04 bar and 3.3-0.04 bar). There is a draw off in the second turbine at 0.2 bar.

We have to remember that this kind of optimization is very time and memory consuming. For this type of computations it takes about 30 hours to make 1000 iterations on 40 processors (2.8 GHZ). It would be very helpful to use surrogate model instead real one. We could get even more promising results.

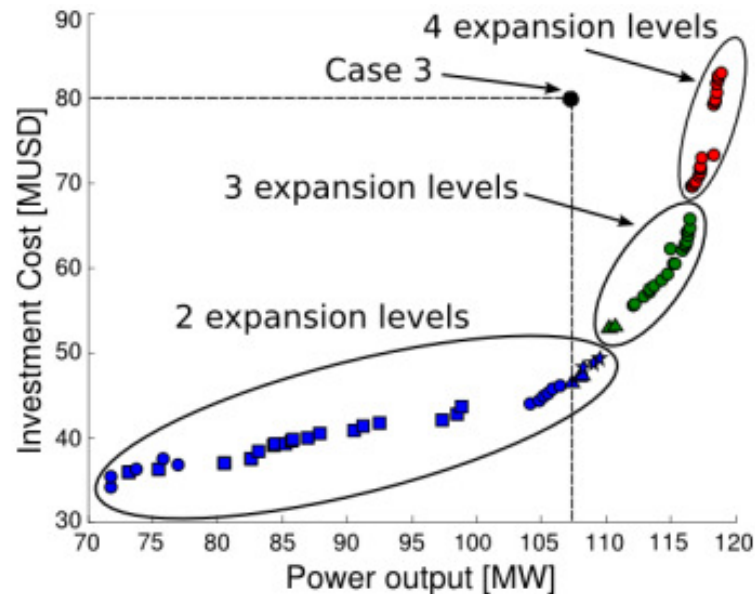


Figure 3.5 Pareto Curve for Power Output – Investment Cost for different configurations of Steam Network (Girardin, Dubuis, & Marechal)

Another possibly optimization is shown in the Figure 3.6. It represents the levelized electricity cost. Comparison of two graphs let us to find the best solution for the system. In this case the optimum configuration consists in a 3 expansion levels. An electricity cost are about 46.5 \$/MWh and an investment cost are 59.2 M\$. The power output is 115 MW. This type of decision making process is characteristic for optimization tasks.

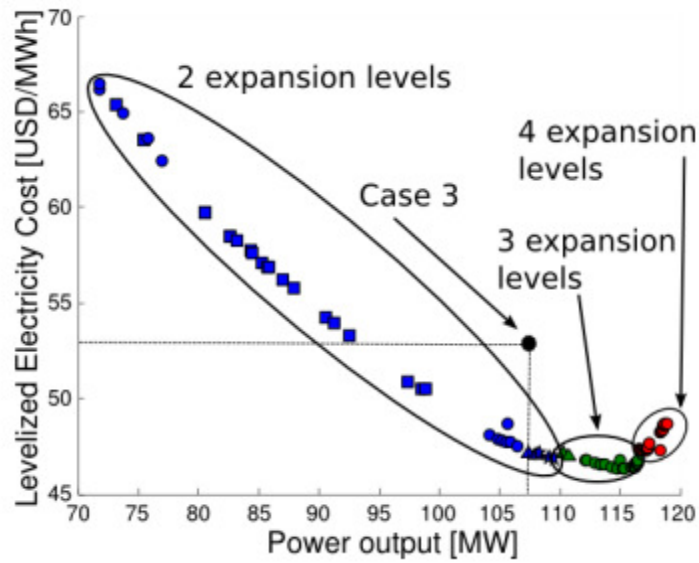


Figure 3.6 Pareto Curve for Power Output – Levelized Electricity Cost for different configurations of Steam Network (Girardin, Dubuis, & Marechal)

3.2 Problem description

In this part Steam Network was analyzed and the surrogate model for it was prepared. Actually the problem was divided for two parts.

The first was the simpler situation with smaller number of input values. This situation represented the simple Steam Network with only one steam production level and only one condensation level.

The second model was prepared for more complicated problem – more complex Steam Network design. All inputs parameters will be described in the next subchapter.

The prepared surrogate model can be used in the studies concerning the Steam Networks. They can improve the number of iteration of genetic algorithm which are possible to run. So also the results of optimization would be more accurate.

This problem introduces moreover several problems connected with surrogate modeling (connected strictly with the software used in LENI). Sometimes there is a big problem with generation appropriate points to train the network. This problem was very good visible in the first simpler network design. In the next chapter the detailed description and used solution will be described.

This problem – Steam Network – is moreover a very good introduction to the task which will be implemented during next year. The final goal of the project is preparing the tool to automotive generation of surrogate model for complex energy systems. The Steam Network has all features which characterize this type of problems.

3.3 Modeling problem description

In this chapter the prepared models are described in the details.

For both of them there was two output values – total power production and total cost of the system. The input values depend from the model.

First surrogate model was prepared for the simpler problem. There were following inputs values take into account:

- H1_Pressure;
- H1_SuperheatingDT;
- H1_ReheatSuperheatingDT;
- H1_doReheat;
- D1_Pressure;
- D1_SuperheatingDT;
- D2_SuperheatingDT;
- D2_Pressure;
- C1_Pressure.

So there is 9 different independent input variables in the system. This situation is much more complicated than problem from previous chapter when we had only one input value.

It was decided to prepare only one type of surrogate model in this case – the neural network model. Actually the solution was more complicated than one simple neural network.

The first problem which appeared was with training points generation. The intervals in which we chose points were not good defined. The first used intervals are shown in the Table 3.1.

Table 3.1 Input parameters of the simple Steam Network model

<i>Paramether</i>	<i>Unit</i>	<i>Minimum value</i>	<i>Maximum value</i>
H1_Pressure	Bar	20	30
H1_SuperheatingDT	⁰ C	20	100
H1_ReheatSuperheatingDT	⁰ C	0	100
H1_doReheat	-	0	1
D1_Pressure	Bar	40	180
D1_SuperheatingDT	⁰ C	0	80
D2_Pressure	Bar	20	40
D2_SuperheatingDT	⁰ C	0	30
C1_Pressure	Bar	0.02	1

During point generation it turned out that there is much more point (the set of input variables) for which the system did not converge than the good ones. There were only about 20% of points for we had the reasonable results.

Because of that the first step of preparing the surrogate model was generation of the 2000 random points. Then on the basis of this set, the simple neural network was created – its task was to check if the point is converging or not. This simple network was used only to create new set of points, which were converging.

The new set of 1640 points was created in the following way: firstly we create new random points of input values. Then the simple network is checking if the point is converging or not (of course the network made some mistakes, as a result in final set there was about 5% points, that finally turn out to be not converged). If the point is not converged (negative answer of network) we randomly choose new point. If the answer of network is positive we run the OSMOSE and Energy Integration for that point and save its value as well as value of costs and power generated by the defined system. Then the data points from both sets (random and converging points) were used to create the final surrogate model. Only the points with positive convergence, were used to create fitting neural network. This algorithm of data preparation is shown in the Figure 3.7.

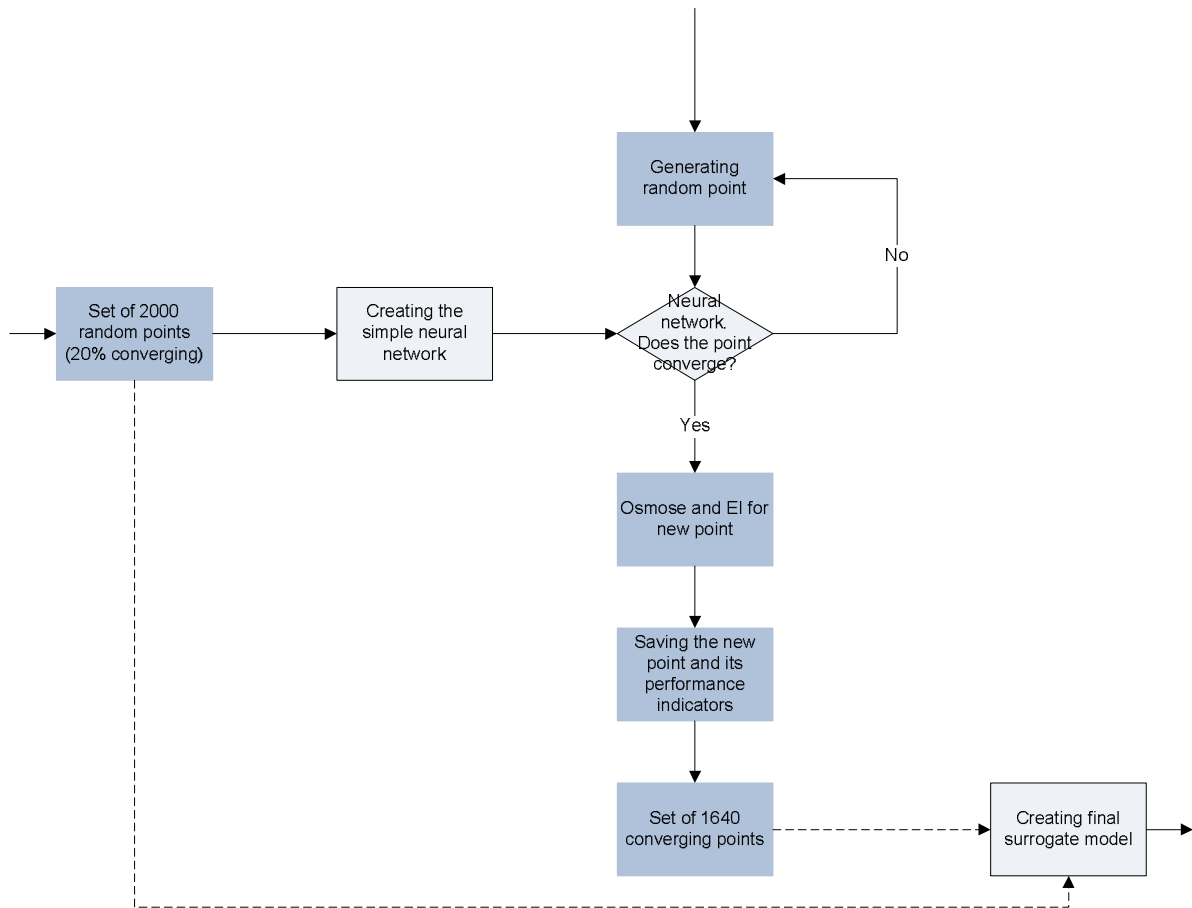


Figure 3.7 Schema of data preparation for surrogate model creation.

For process of training a network all of the input and targets values was normalized to the interval [-1,1]. This was done with Neural Network Toolbox function:

$$[Y, PS] = \text{mapminmax}(X, YMIN, YMAX) .$$

This normalization helps network to fit better to the data. It is connected with shape of activation function in the neurons.

Moreover when we have the neural network for two or more output parameters normalization helps in showing the real errors of both function, for example in the regression graphs. If the data were not normalized, the output with bigger absolute values could have bigger influence for final results.

However, in this problem, for each objective function (cost and power) the different surrogate model was created. Each of the final surrogate models consists of two different neural networks. The first of them is classification network. To train this network the whole set of training points was used. It shows if the input point is converging or not. If not, the answer of surrogate model is: the point does not converge – the system is incorrect. If yes, the point is sent to the second part of the surrogate model – second neural network. To train it only the converging points were used. This network computes the value of performance indicator for given point. The schema of surrogate model, and method in which it works is presented in the Figure 3.8.

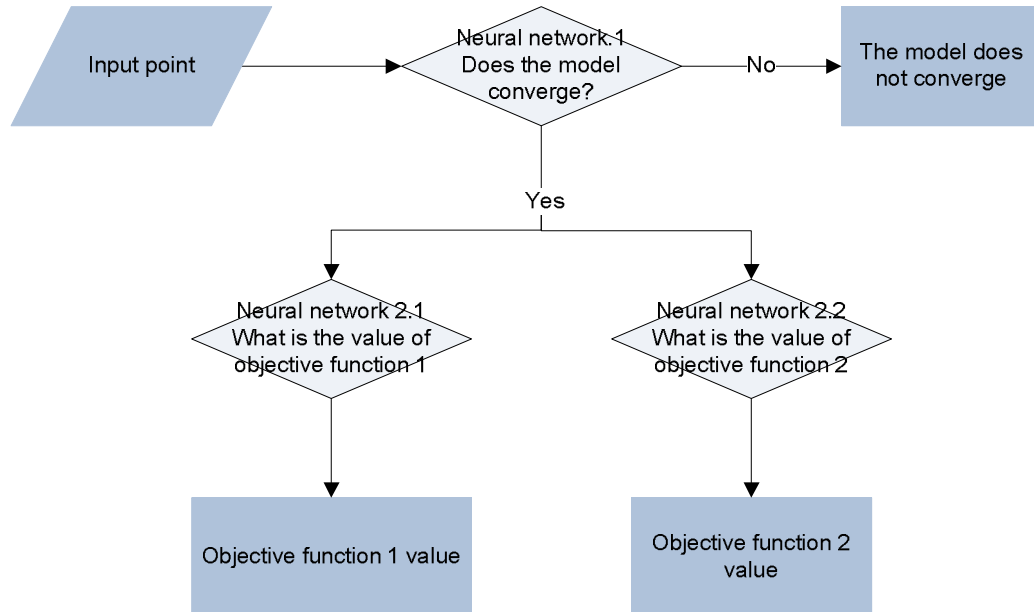


Figure 3.8 The schema of surrogate model

Very similar methodology was used in the case of more complicated model. There was only one difference. In the set of the randomly generated points about 50% were converged. So the auxiliary neural network was not used in this case. We were working on the points generated in the beginning.

In the more complicated case there were much more decision variables (23). Some of them were normal number variables, but some represented decision if a unit is present in the system or not. The list with all decision variables and the intervals for which we generated points is shown in the Table 3.2.

Table 3.2 Input parameters of the simple Steam Network model

<i>Parameter</i>	<i>Unit</i>	<i>Minimum value</i>	<i>Maximum value</i>
H1_Pressure	Bar	2	30
H2_Pressure	Bar	30	140
H3_Pressure	Bar	100	180
H1_SuperheatingDT	⁰ C	20	100
H2_SuperheatingDT	⁰ C	20	150
H3_SuperheatingDT	⁰ C	20	100
H1_ReheatSuperheatingDT	⁰ C	0	100
H1_doReheat	-	0	1
H2_ReheatSuperheatingDT	⁰ C	0	100
H2_doReheat	-	0	1
H3_ReheatSuperheatingDT	⁰ C	0	100
H3_doReheat	-	0	1
D1_Pressure	Bar	40	180
D2_Pressure	Bar	20	40
D3_Pressure	Bar	5	60
D4_Pressure	Bar	0.02	5
D5_Pressure	Bar	0.02	1
D1_SuperheatingDT	⁰ C	0	80
D2_SuperheatingDT	⁰ C	0	30
D3_SuperheatingDT	⁰ C	0	80
D4_SuperheatingDT	⁰ C	0	30
D5_Vapf	-	0.85	1
C1_Pressure	Bar	0.02	1

3.4 Results

In this subchapter the results of surrogate modeling of complicated multivariable problem are presented. In this problem we focused also on the surrogate model with one hidden layer. During the surrogate modeling process we get good results for simpler problem and much worse for the complex one.

Firstly the information connected with simpler version of the system are shown. Several network configurations were analyzed and described. The graphs show characteristic features of neural network surrogate models. We have to remember, that each of surrogate models consists actually with three neural networks:

- Classifying part;
- Power fitting part;
- Cost fitting part.

Simpler problem

The first analyzed model was prepared on the basis of big training set (we used 2548 points to train the network and 1092 points to test the network). The first, classifying network was consisted of 10 neurons in hidden layer. The details about the neural network are presented in the Table 3.3.

Table 3.3 Properties of neural network surrogate models – classifying part – bigger testing set

<i>Neurons in hidden layer</i>	<i>Iteration number</i>	<i>Time [s]</i>
10	75	10

This type of the network is different than networks used in the previous part of the Thesis. The classifying network shows to which class the point belong: to the converging or to not converging points. As an output we have a single value, if it is close to 1 it means that point is converging. It is more close to 0 the point does not converge.

The numbers according to size of each set of points as well as information about the accuracy of network in each set are presented in the Table 3.4. It can be seen that biggest precision is reached in the training set (here the situation is similar as in the most of fitting networks).

Table 3.4 Neural network surrogate models – classifying part – bigger testing set: MSE and Regression R values

	<i>Training</i>	<i>Validation</i>	<i>Test</i>	<i>All</i>
Points	1820	728	1092	3640
Accuracy	99,6%	98,5%	98,4%	99%

The confusion matrix of this problem is presented in the Figure 3.9. The confusion matrix sows 4 groups of points:

- Points originally converging, classifying as converging (good classification);
- Points originally converging, classifying as not converging (bad classification);
- Points originally not converging, classifying as converging (bad classification);
- Points originally not converging, classifying as not converging (good classification).

To create the surrogate model the similar number of converging (1940) as not converging points (1719) were used.

In the set of 3640 points only 35 are wrongly classified. It is only 1% of all points. The worse classification can be seen in the set of points originally not converging, classifying as converging than in the set of points originally converging, classifying as not converging. It is probably connected with the fact that in the whole space there is much more originally not converging points than converging ones. Because of the method of generating points there is much bigger sampling density in the “converging parts” of the space.

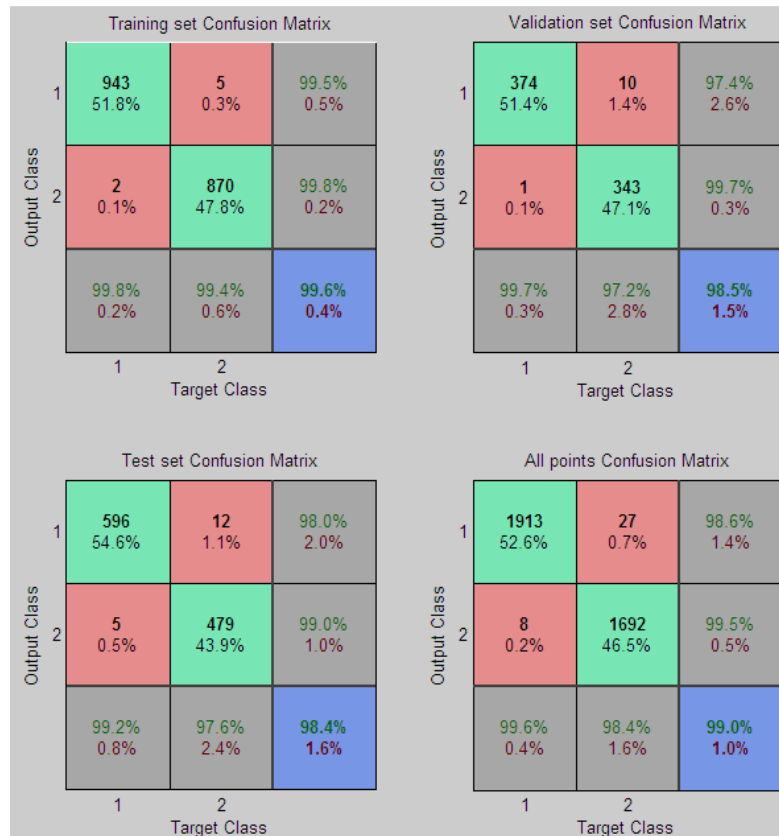


Figure 3.9 Confusion matrix for neural network surrogate models – classifying part – bigger testing set

The details about the proces of learning the calsyffing network are presented in the **Figure 3.10**. On the graph with training state we can notice that validation error behaved rather changeabllly.

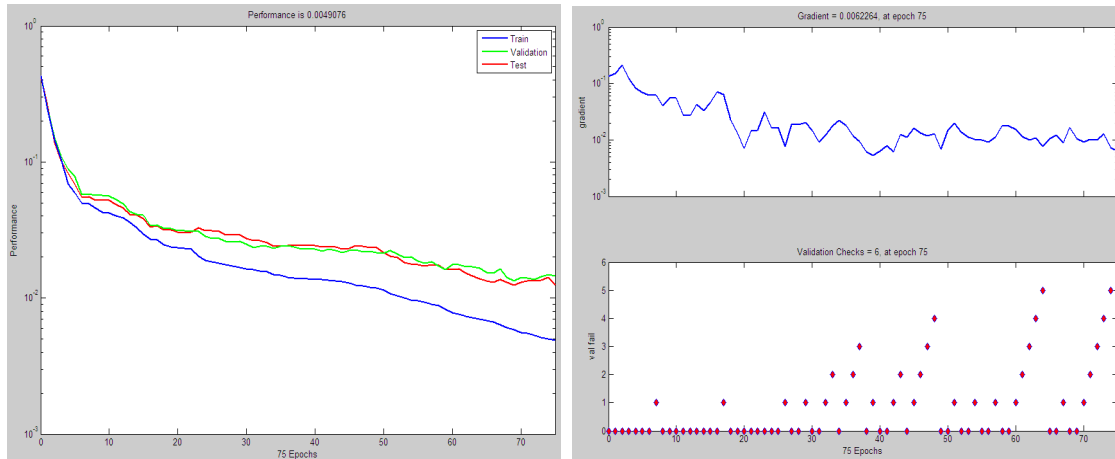


Figure 3.10 Performance and Training State graphs for neural network surrogate models – classifying part – bigger testing set

When the classifying network had been ready, the power and cost fitting models were created.

Details about first of them – cost fitting surrogate model are presented in the Table 3.5. The network is consist of 20 neurons in hidden layer. All of the model parameters output as well as input ones were normalized. But still – for such big problem (with big number of samples) the norm of residuals is relatively small. The correlation coefficient for this problem – 0.99 is also very good.

Table 3.5 Properties of neural network surrogate models – power fitting part – bigger testing set

<i>Neurons in hidden layer</i>	<i>Iteration number</i>	<i>Time [s]</i>	<i>Norm of residuals</i>	<i>Correlation coefficient</i>
20	47	6	2.56	0.99

The Table 3.6 presents the rest of the important information about power fitting network. It shows the number of points used to train the network (1122) and to validate it (198). To test the network all of the converging points from testing set of classifying network were used (601 points). The mean square error is small – it is equal only to 0.003. The regression R is over 0.99.

Table 3.6 Neural network surrogate models – power fitting part – bigger testing set: MSE and Regression R values

	<i>Training</i>	<i>Validation</i>	<i>Test</i>	<i>All</i>
Points	1122	198	601	1921
MSE	0.0013	0.0033	0.0074	0.0034
Regression R	0.997	0.992	0.982	0.992

The Figure 3.11 presents the regression graphs for all of the sets. As it can be seen there is several mistakes in the model, but still the fit is very good. To improve the behavior of the network we should check the points where there are biggest errors and give more training examples from their neighborhood.

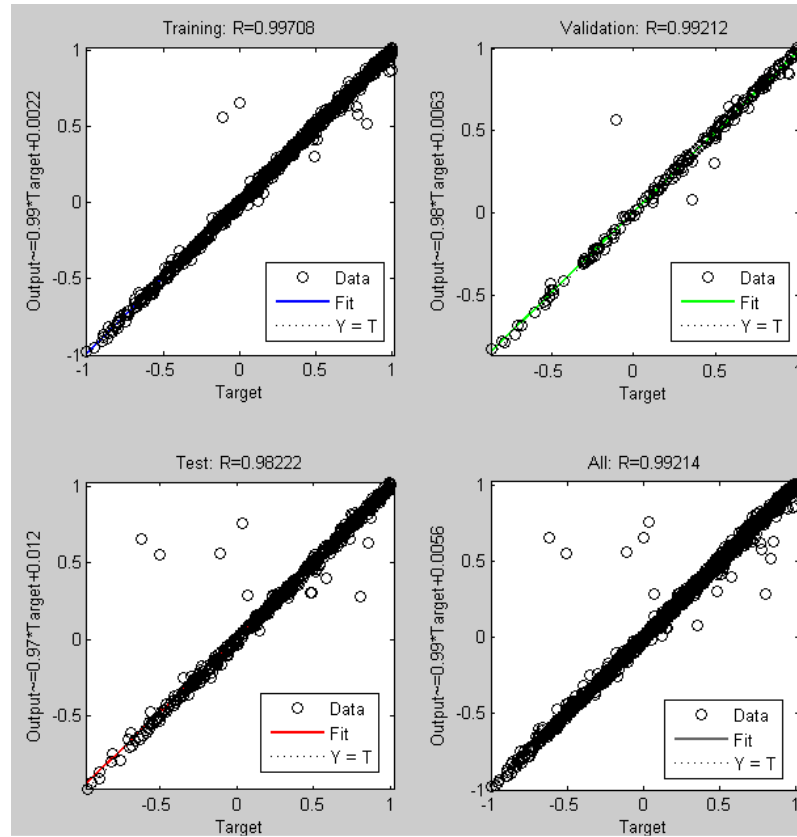


Figure 3.11 Regression graph for neural network surrogate models – power fitting part – bigger testing set

The training process details for this problem are presented in the Figure 3.12.

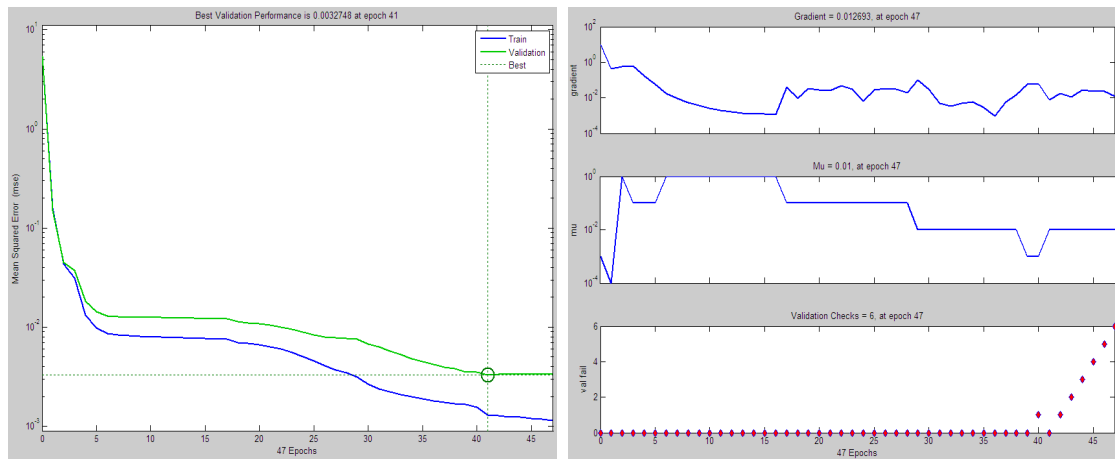


Figure 3.12 Performance and Training State graphs for neural network surrogate models – power fitting part – bigger testing set

The second objective function – total cost – are presented below. In the Table 3.7 the properties of the neural network model are shown. As previously the network has 20 neurons in the hidden layer and its properties (norm of residuals and correlation coefficient) are good.

Table 3.7 Properties of neural network surrogate models – cost fitting part – bigger testing set

Neurons in hidden layer	Iteration number	Time [s]	Norm of residuals	Correlation coefficient
20	48	6	2.42	0.993

The results for each of the sets are shown in the Table 3.8. The sizes of each of the sets are the same as previously. The Regression for all points is equal to 0.993.

Table 3.8 Neural network surrogate models –cost fitting part – bigger testing set: MSE and Regression R values

	Training	Validation	Test	All
Points	1122	198	601	1921
MSE	0.00092	0.0083	0.0053	0.0031
Regression R	0.999	0.982	0.987	0.993

The regression graphs for the cost fitting network are shown in the Figure 3.13. There are several errors of the network in the training set as well as in the rest of the sets. To improve the quality of the network we could retrain the network. The results of such process will be

shown in the next analysis. Another method is trying to find more training examples in the most sensitive part of the space (the same solution as previously).

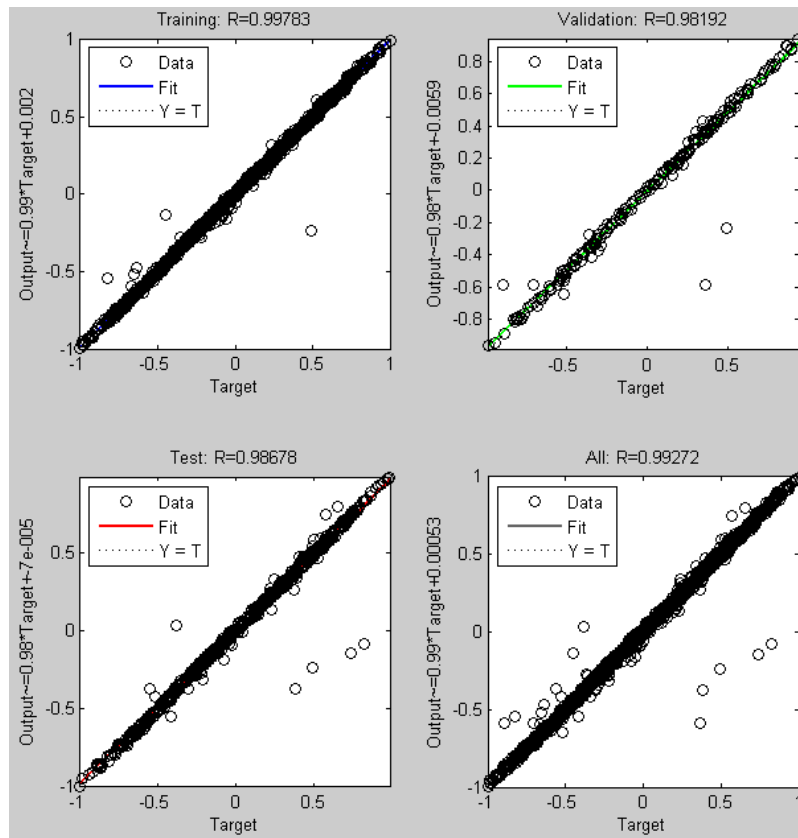


Figure 3.13 Regression graph for neural network surrogate models – cost fitting part – bigger testing set

The training details for this network are shown in the Figure 3.14.

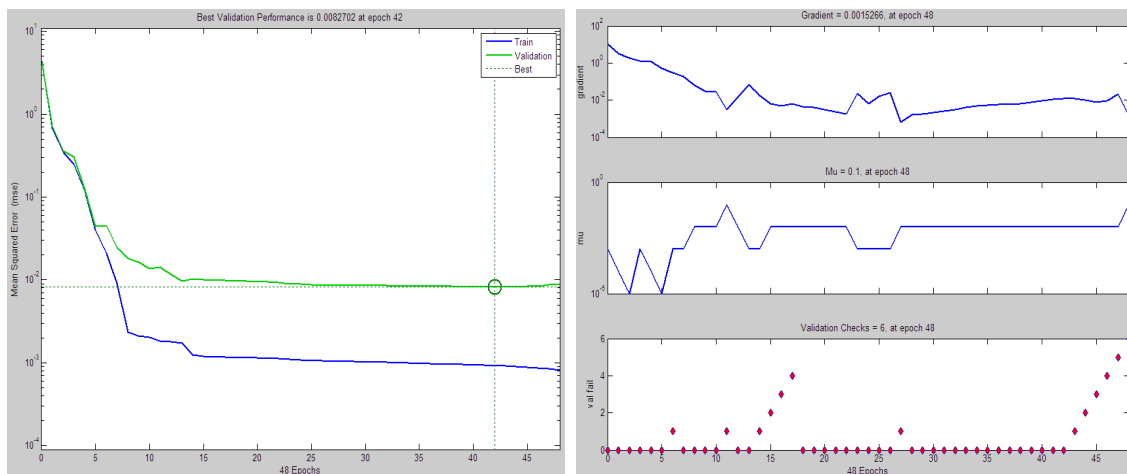


Figure 3.14 Performance and Training State graphs for neural network surrogate models – cost fitting part – bigger testing set

As the second analysis the same situation was investigated. Actually the first network is the same one as previously. We changed only fitting networks. We checked the influence of the different number of neurons in the hidden layer for the overall results. In this case we used 30 neurons inside the network (earlier there were 20 neurons). The details about the network are presented in the Table 3.9. The norm of residuals is better than previously (1.45 in comparison to 2.56). The same situation is in the case of correlation coefficient (0.997 in comparison to 0.99).

Table 3.9 Properties of neural network surrogate models – power fitting part – bigger testing set, version 2

<i>Neurons in hidden layer</i>	<i>Iteration number</i>	<i>Time [s]</i>	<i>Norm of residuals</i>	<i>Correlation coefficient</i>
30	72	16	1.45	0.997

Table 3.10 shows learning process for this situation. The number of points is the same as previously. The regression coefficients are also slightly better than in the case of smaller number of neurons in the hidden layer (for example 0.997 in comparison to 0.992 for all points).

Table 3.10 Neural network surrogate models – power fitting part – bigger testing set, version 2: MSE and Regression R values

	<i>Training</i>	<i>Validation</i>	<i>Test</i>	<i>All</i>
Points	1122	198	601	1921
MSE	0.00013	0.00038	0.0031	0.00109
Regression R	0.9997	0.9992	0.993	0.997

As it can be seen in the Figure 3.15 the biggest error are in the test set and they are irregular. The fit for the training and validation set is almost ideal.

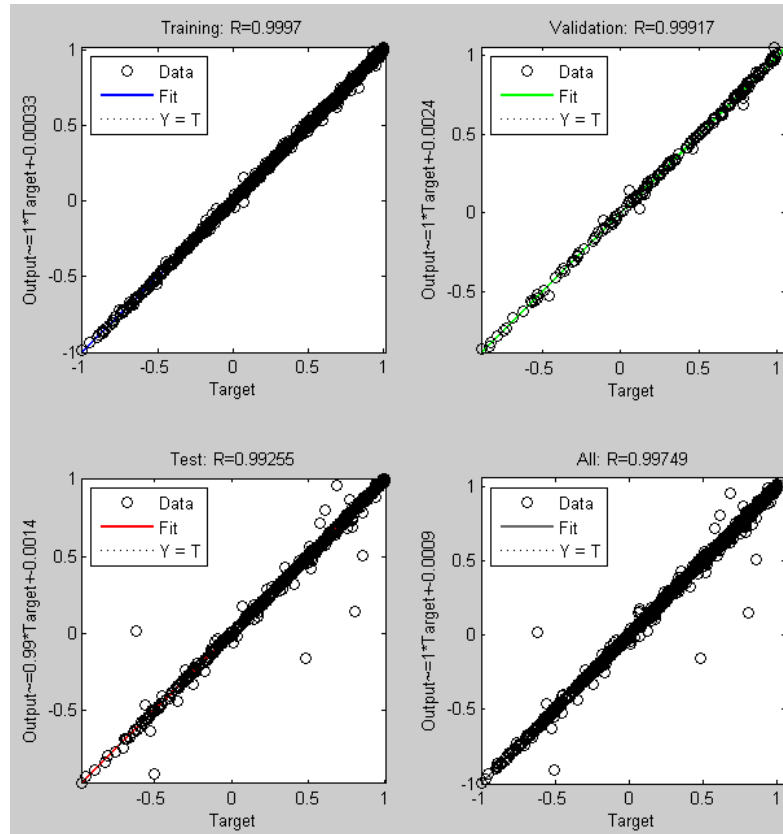


Figure 3.15 Regression graph for neural network surrogate models – power fitting part – bigger testing set, version 2

The next example shows interesting correlation. The first prepared network was quite bad, but retraining gave us much better results. The final information about the network are presented in the Table 3.11 and in the Table 3.12.

Table 3.11 Properties of neural network surrogate models – cost fitting part – bigger testing set, version 2

Neurons in hidden layer	Iteration number	Time [s]	Norm of residuals	Correlation coefficient
30	28 (retraining)	6	3.59	0.984

Table 3.12 Neural network surrogate models – cost fitting part – bigger testing set, version 2: MSE and Regression R values

	Training	Validation	Test	All
Points	1122	198	601	1921
MSE	0.000008	0.00011	0.021	0.0067
Regression R	1	0.999	0.95	0.98

Figure 3.16 shows first attempt to build the surrogate model. Result are bad and big errors appear even in the training set.

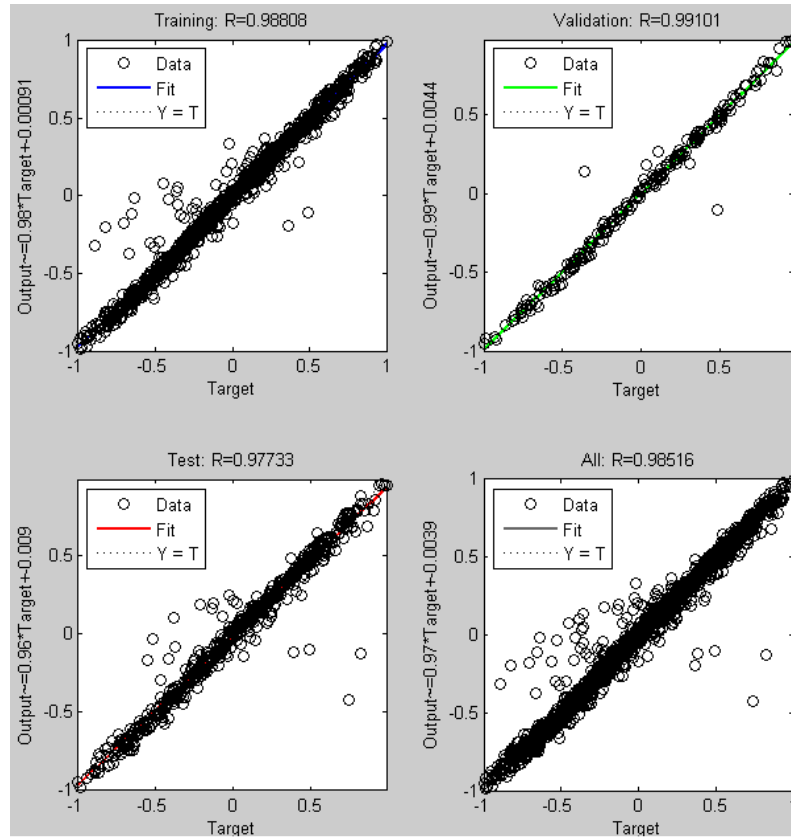


Figure 3.16 Regression graph for neural network surrogate models – cost fitting part – bigger testing set, version 2, training

After the first retraining process (we train the network again but as initial weights we use the weights from previous network) the results are much better. They are presented in the Figure 3.17. There is only one bad thing in the model. There are two points in testing set with big error. The values of their output are bigger than 1 and smaller than -1 respectively. This is very bad behavior – when we use surrogate model to optimize a problem, the results can be wrong. The optimization algorithm can find non-existent extremums. One of ideas to solve this problem is to use the connection of two algorithms: optimization and surrogate modeling algorithm. This solution will be described in the Generalization chapter.

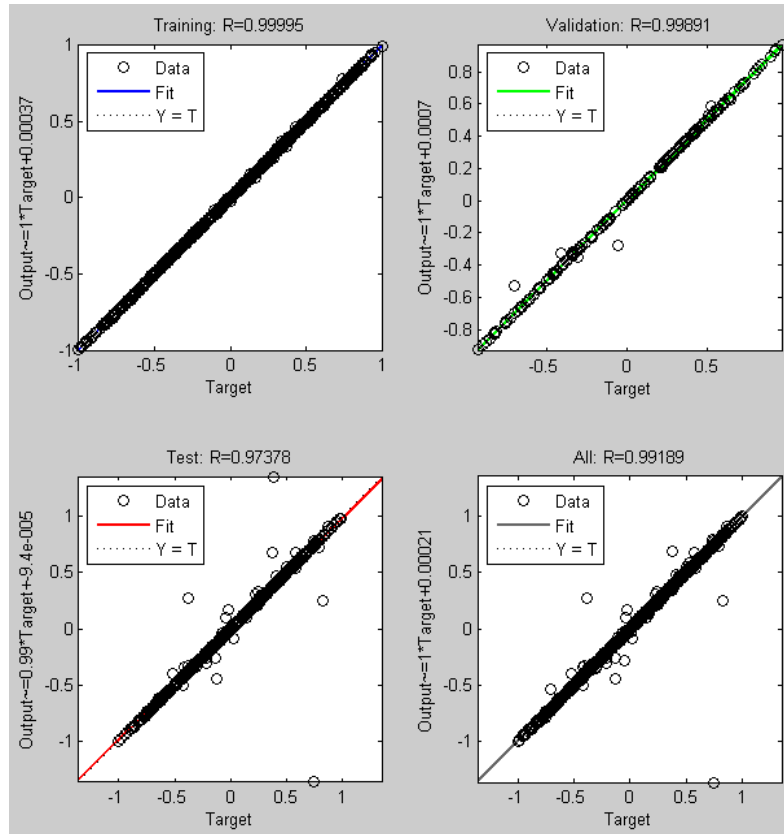


Figure 3.17 Regression graph for neural network surrogate models – cost fitting part – bigger testing set, version 2, retraining 1

We decided to check if we can improve the surrogate modeling results even better. We retrained the network one more time. The result graphs are presented in the Figure 3.18. The approximation for the most of the points is almost ideal. But from the other hand the result for extremums points are even worse – they can hardly influence the optimization process.

The interesting result is that when we retrain the network again error does not change. The shape and the all quality indicators for testing set are almost ideal. The network can not be learnt anything new with the same set of input points.

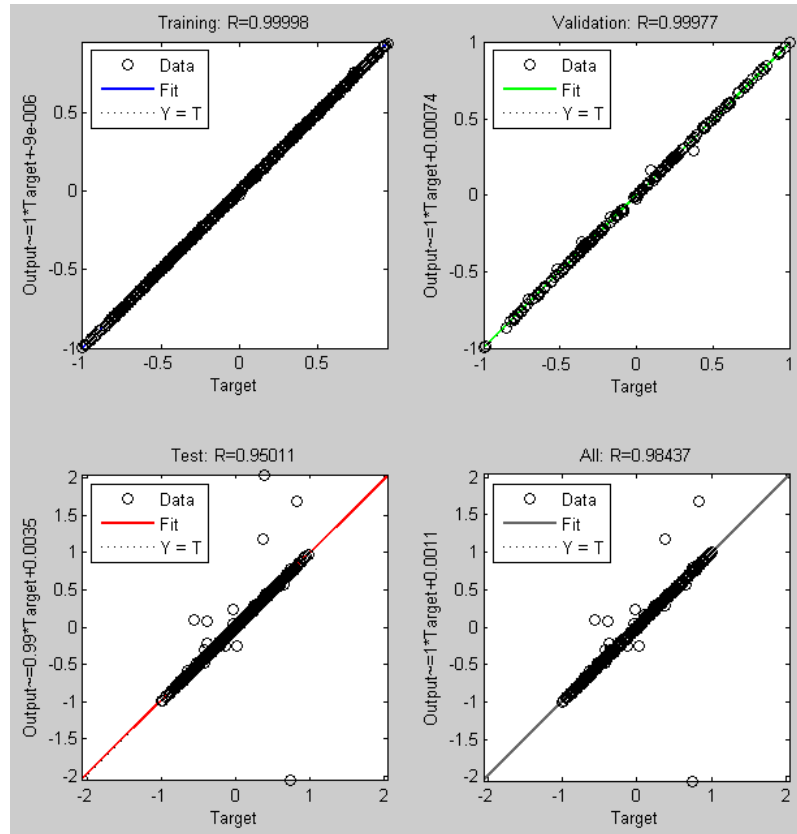


Figure 3.18 Regression graph for neural network surrogate models – cost fitting part – bigger testing set, version 2, retraining 2

The last presented experiment is connected with the number of training and validation points. Now we used much smaller learning sets. All of the results are presented in the same convection as before.

For the classifying network the number of neurons in the hidden layer is 10 as in the first analyzed case. The details about network are presented in the Table 3.13.

Table 3.13 Properties of neural network surrogate models – classifying part – smaller testing set

Neurons in hidden layer	Iteration number	Time [s]
10	62	9

Table 3.14 shows the training process details. In this case we used almost two times smaller training and validation sets than previously (1092 and 364 respectively in comparison to 1820 and 728 points). This change influences the quality of the model for each of the sets of points (for example 99% in comparison to 97,9% for all points).

Table 3.14 Neural network surrogate models – classifying part – smaller testing set: MSE and Regression R values

	<i>Training</i>	<i>Validation</i>	<i>Test</i>	<i>All</i>
Points	1092	364	2184	3640
Accuracy	98,9%	97,3%	97,6%	97,9%

The Figure 3.19 shows confusion matrixes for all sets. There is 75 points classified wrongly among all of the points.

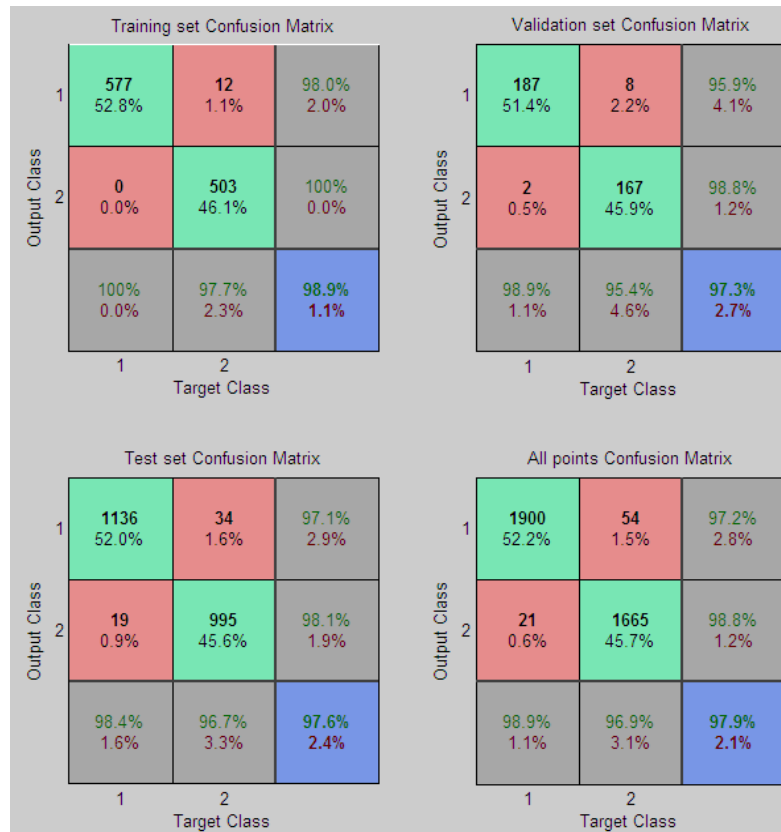


Figure 3.19 Confusion matrix for neural network surrogate models – classifying part – smaller testing set

The results of the power and cost fitting functions are shown below. Firstly the details about power fitting surrogate model are presented in the Table 3.15 and in the Table 3.16 (we are using the bigger number of neurons in the hidden layer).

Table 3.15 Properties of neural network surrogate models – power fitting part – smaller testing set

<i>Neurons in hidden layer</i>	<i>Iteration number</i>	<i>Time [s]</i>	<i>Norm of residuals</i>	<i>Correlation coefficient</i>
30	68	23	3.67	0.984

Table 3.16 Neural network surrogate models – power fitting part – smaller testing set: MSE and Regression R values

	Training	Validation	Test	All
Points	651	115	1155	1921
MSE	0.000068	0.00010	0.012	0.0070
Regression R	0.9997	0.9998	0.974	0.985

Figure 3.20 shows the shape of regression function for the objective function of total power in the system. The fit is very good except of a few points. They are quite bad fitted – this is the situation when the can disturb the process of optimization. The reason for this fact is that the data were not cover the whole searching space (we have less training examples now so probability of situation like this is bigger).

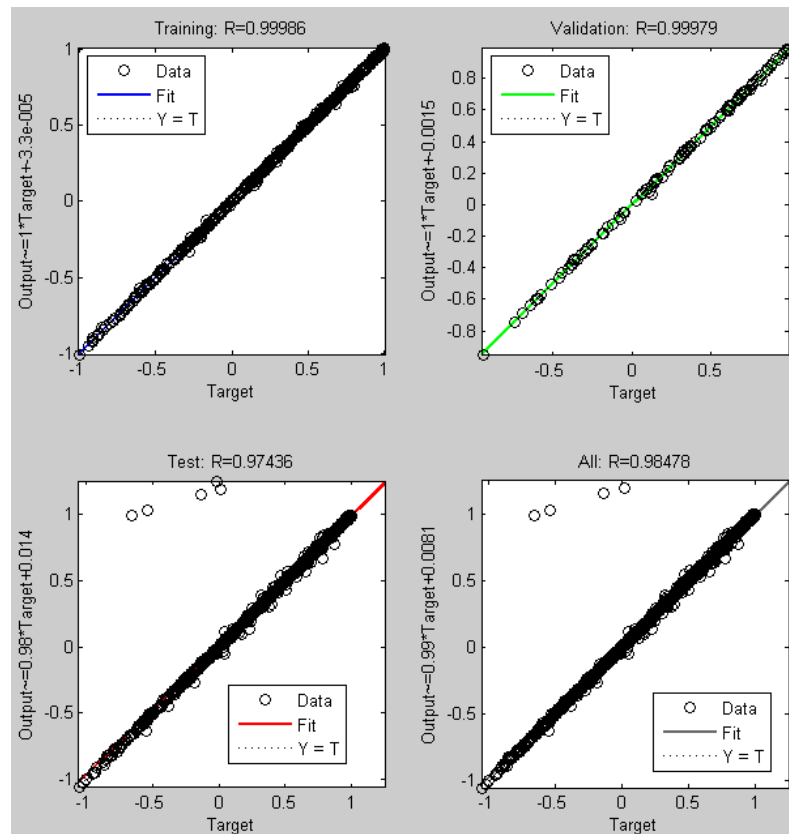


Figure 3.20 Regression graph for neural network surrogate models – power fitting part – smaller testing set

Table 3.17 and Table 3.18 shows characteristic of the cost fitting surrogate model.

Table 3.17 Properties of neural network surrogate models – cost fitting part – smaller testing set

Neurons in hidden layer	Iteration number	Time [s]	Norm of residuals	Correlation coefficient
30	67	23	4.034	0.98

Table 3.18 Neural network surrogate models – cost fitting part – smaller testing set: MSE and Regression R values

	Training	Validation	Test	All
Points	651	115	1155	1921
MSE	0.00031	0.00021	0.014	0.0085
Regression R	0.9993	0.9995	0.968	0.981

Results in this case are similar as for power production model. Overall fitting is quite good (it could be improve a little but by the retraining the network). But there are 5 points which do not follow the tendency. They can be observed in the Figure 3.21.

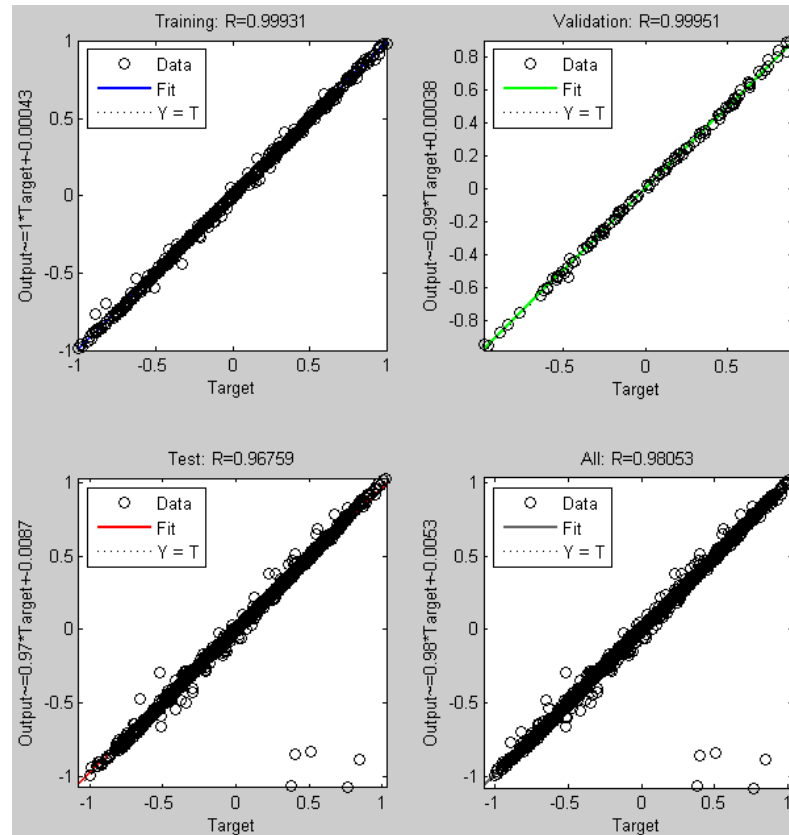


Figure 3.21 Regression graph for neural network surrogate models – cost fitting part – smaller testing set

Complex problem

As it was mentioned before results in this case are much worse than for the simpler case. In this problem there is 23 decision variables. It is huge number and our task was in fact to approximate the function in 24-dimension space.

This system is in fact not the most popular solution (in practical problems). It is much more complicated than popular used systems. Normally we use solutions which are between simple and complex approach presented in this work.

The bottleneck of the surrogate modeling process was real point's generation time. Preparing one point by OSMOSE and Energy Integration software took over 1 minute. It was 1906 points generated – this took over 32 hours. For this number of points several neural networks' configurations were investigated. Any of them did not give satisfactory results. In some configuration it is possible to achieve good fit in the training set, but the quality of the model for the validation and test set is bad.

We will present only the results for classification network and for the power fitting network (the results for the cost fitting model are analogical).

The quality of the classification network could be accepted. The details are presented in the Table 3.19 and Table 3.20.

Table 3.19 Properties of neural network surrogate models – classifying part – complex problem

<i>Neurons in hidden layer</i>	<i>Iteration number</i>	<i>Time [s]</i>
25	28	4

Table 3.20 Neural network surrogate models – classifying part – complex problem: MSE and Regression R values

	<i>Training</i>	<i>Validation</i>	<i>Test</i>	<i>All</i>
Points	1620	191	95	1906
Accuracy	87,6%	83,8%	87,4%	87,2%

The confusion matrixes for this case are shown in the Figure 3.22. For all of the sets the accuracy of the model is over 80%.

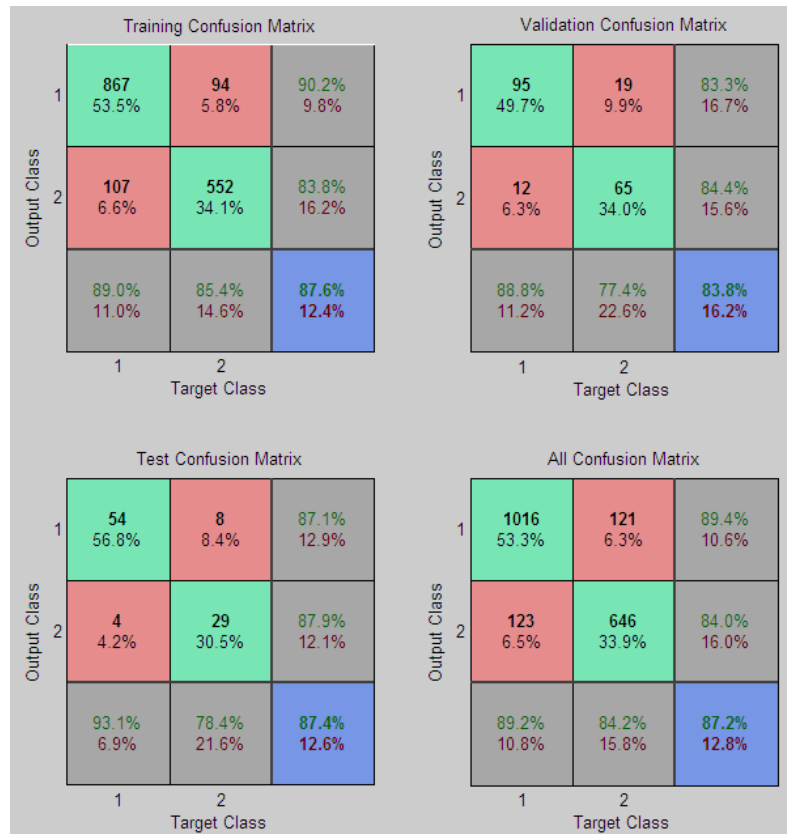


Figure 3.22 Confusion matrix for neural network surrogate models – classifying part – complex problem

Much worse situation takes place when we try to approximate the power production. The first attempt is shown in the Figure 3.23. The model has been prepared for 25 neurons in the hidden layer. There can be observed some tendency in the data but the model is bad.

To prepare this model there were used 797 points in training set and 171 points in the validation set. The surrogate model was tested on the basis of the 171 points.

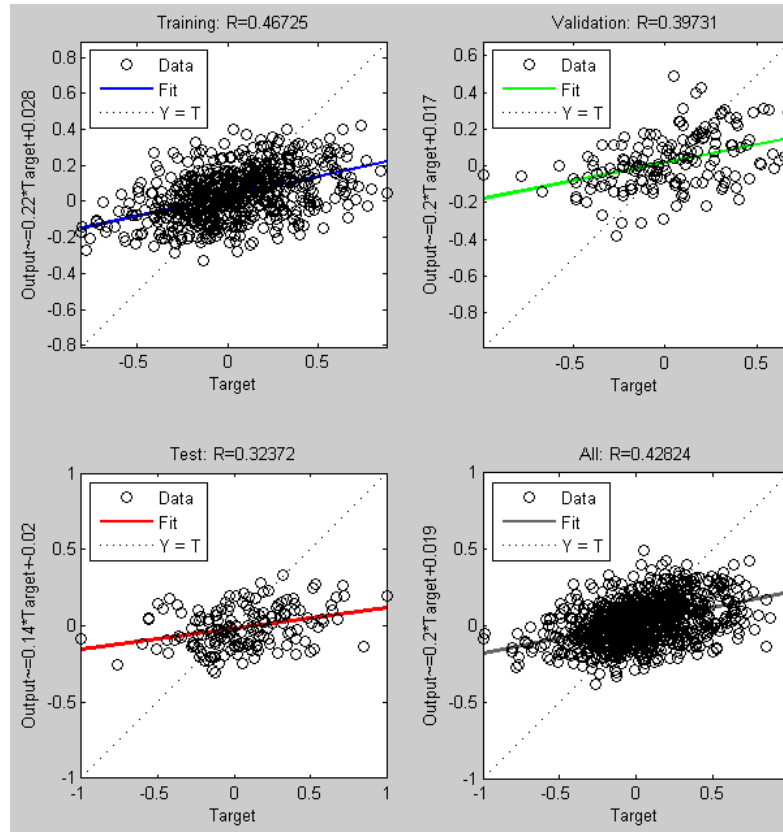


Figure 3.23 Regression graph for neural network surrogate models – power fitting part – complex case; 25 neurons in the hidden layer

When we increase the number of neurons in the hidden layer (for example 60 neurons) we can achieve big precision in the training sets. This situation is presented in the Figure 3.24. This situation shows the effect of over fitting the surrogate model.

There are several solutions in this case to improve the quality of the model. First of them is to collect more training data, which better cover the space. The second method is to use different method of learning the model (for example, choose different validation set in each iteration of the algorithm). The last idea is to use special methods of sampling – instead of random selection use the declared algorithm.

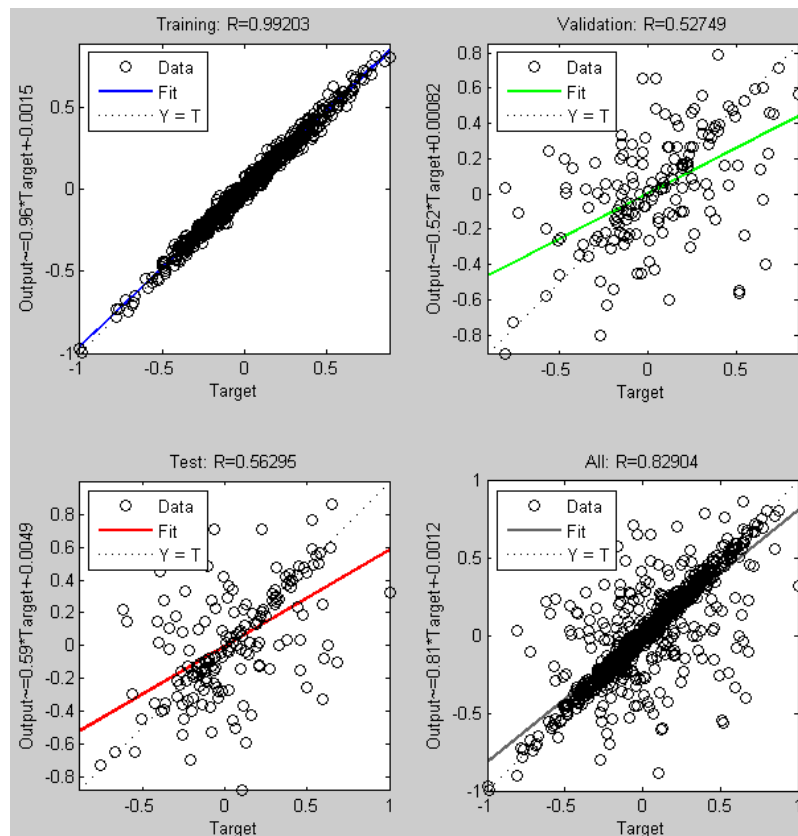


Figure 3.24 Regression graph for neural network surrogate models – power fitting part – complex case; 25 neurons in the hidden layer

4 GENERALIZATION

In this chapter some general observations connected with surrogate modeling problem are collected.

4.1 Points choosing

The point choosing methods are important in the case of mathematical as well as artificial intelligence modeling techniques. The better the training points cover the space, the better results we can obtain.

The good example of appropriate points influence was shown for polynomial interpolation – for Chebyshev nodes we could obtain much better results than for equidistant or random one. Moreover with Chebyshev nodes we could obtain the Runge phenomena.

Another important thing was to choose the optimal number of nodes. If we have too less of nodes surrogate models would not be precise. On the other hand – if we have too many nodes, their generation time will be very long. This period with the time of preparing and using surrogate model can make building surrogate model uneconomic. It can be better to generate values of chosen points in the standard way (flowsheeting models).

In the analyzed examples it can be seen that the biggest errors appear in the parts of the space where the enough number of samples was not chosen. The problems appear also very often next to the edges of the searching space. It is caused by the lack of reference points for that parts of the space.

The good solution could be equidistant sampling of the whole space. Unfortunately in many cases this is not possible. We have to remember that with growing dimension of the space, the number of equidistant samples grows exponentially. However, this type of sampling is good in simple problems with one or two decision variables. This simple problem can be found in the sensitivity analysis or very simple multi-objective optimizations.

However, in practical life we are forced to solve much more complicated problems. There are several ways to deal with them. In this work the simple random sampling was used. We can get satisfactory results with this approach, but normally we need to prepare many samples and, as it was shown in the last example in practical part, this is not always possible.

Other solution is to select new points in the correlation to previously chosen. For example if we will choose a point, we can not choose another one in the assumed distant. This assumption would prevent choosing many points only in one part of the whole space – the sampling distribution would be more equal.

Another idea is to choose points accordingly to the gradient in the model. If we notice that there is a big gradient in some parts of the space, we chose more point in that neighborhood. This algorithm could be connected with random point choosing to avoid miss local extremums.

Two methods which are described above need to be carefully investigated in the future work.

4.2 Model training methods – errors elimination

In this subchapter we will focus mainly on the artificial intelligence surrogate models based on the neural networks. Although some of the presented methods can be used successfully in the mathematical modeling techniques.

First method of improving the surrogate model quality was described in the previous section and it requires choosing optimal training points. But it is not always possible. When we are using the surrogate model to accelerate the optimization process, we can think about connecting both algorithms: optimization and building the surrogate model. This method will be described in the separated subchapter.

But we can obtain great results also by careful design of surrogate model: choosing the best mathematical method or finding the best network configuration.

Several properties and methods for improving neural network surrogate models were described in the practical part of the Thesis. Here we will collect all of them:

- Choosing appropriate number of layers. More complicated network can solve complex problems more effectively. On the other hand they are more memory and time consuming (especially in the training process);
- Hidden neurons number – they have similar tendency like the number of layers. When there is too less neurons in the hidden layer the network can have a problem with finding a good solution. On the other hand, for bigger network (more hidden neurons) we need more time to prepare them. And moreover they are in many cases over-fitted. Normally the maximum of the input and output values is good approximation for optimal number of neurons in the hidden layer;
- There are many types of activation function in the neurons – two of them were detailed presented (linear and semilog function). The basic usages of this function were described (neurons in output and hidden layer respectively). Other types of activation function will be investigated in the future work;
- To improve the behavior of the neural network we can use the variable validation set. It is especially good method when we have not got many input points (for example their generation is time-consuming). Firstly we divide the whole set of points for n sets. In each iteration of training algorithm (epoch) different set of initial group is used as validation set. Other sets ($n-1$) are used as training set;
- When the satisfactory solution is not found we can try to retrain the network. Retrain is very similar to normal training process. Only difference is in the starting weights. In the beginning we start with the randomly initialized values of weights and bias. When the neural network is retrained the initial weights are pre-trained, so it is easier to find good surrogate model from that point;
- There can be also find several methods of defining stop condition. Stop condition tells us when we should end the training process. One of the methods was presented in the Thesis (ending when validation error does not decrease in defined number of

iteration). Other stop condition is assumed number of epochs or assumed maximum error of the model.

All of this method can be used in the building optimal surrogate model. Some of them can be used in the generic tool to build such models.

4.3 Surrogate model errors – validity

Another very important problem connected with surrogate modeling is finding the part of the whole decision variable space in which the model is valid. In another words we should introduce the measurement with let us know how big errors can we except for analyzed point of decision variables.

As it was shown in the practical part of the Master Thesis, in many cases, when we have appropriate set of training points chosen, we can obtain very good results by using surrogate modeling. In case of simple problem it is better to use simple mathematical methods, which can give even better results than more complicated in the conceptual and computational senses artificial intelligence surrogate models. We should notice, that in many cases very small differences between real values and surrogate model outputs does not influence the final results (for example, when we are thinking about system with power production approximately equals to MW, the differences in single W or even kW are not so important). More important thing is that the surrogate model should behave as real one in context of overall shape (when we are using surrogate model to optimize a function the location of model extremums is one of the most important things).

Generally the surrogate model gives the good results in the part of the decision variable space which is appropriately covered by training samples. It is logical – the surrogate model have only that information, which was earlier given to it. It can not forecast the shape – it can only generalize the available information.

Moreover we should realize that the surrogate models are much better in interpolation values inside the problem ranges (inside the input point sampling area) than beyond the ends of the range. All of the presented models had the biggest problems with points next to the ends. When we do not have any reference points finding real value of the model is even harder.

There is one more important feature of surrogate models which should be noticed in this subchapter. In some of the examples we observed appear of the additional extremums. It could be observed especially well in the problems with normalized output values. All of the output could be included in the range $[-1,1]$. But unexpectedly the strange output values were given by surrogate models – some of them were equal to -2 some to 2. Appearance of these additional points can destroy the whole result of optimization of a function.

Because of that the good idea is connecting the optimization and surrogate modeling algorithms. This solution can help in avoiding described situations.

4.4 Connection with optimization genetic algorithm

The idea how we can improve a surrogate model, which was mentioned before, is to connect both algorithms: optimization and surrogate ones.

The schema of connected algorithm is presented in the Figure 4.1. This method was described for example in (Sreekanth & Datta, 2010).

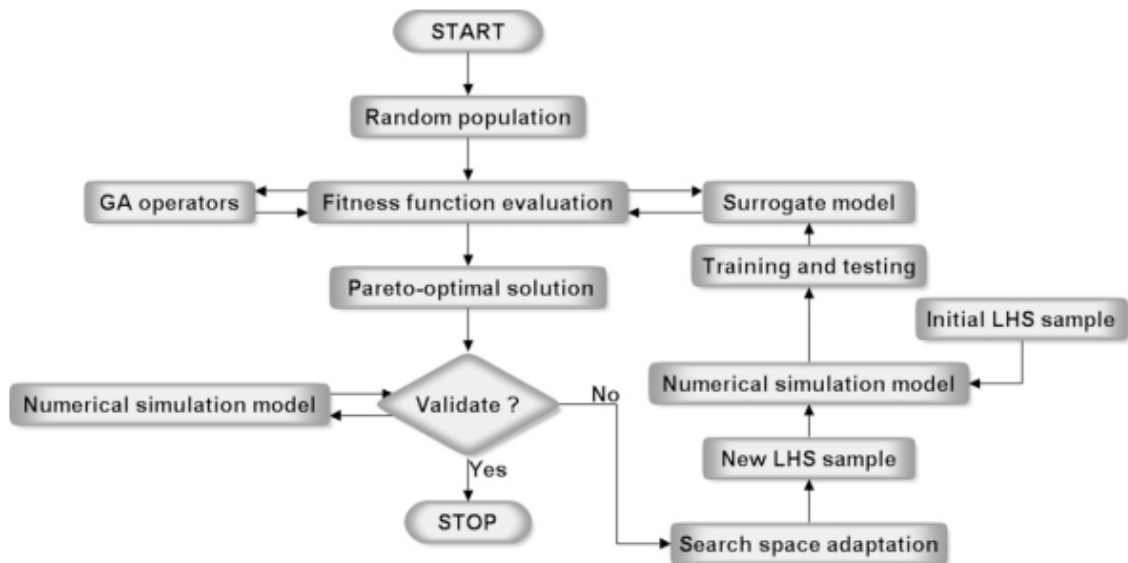


Figure 4.1 Surrogate modeling with optimization algorithm (Sreekanth & Datta, 2010)

The main idea is following: we are training the surrogate model with initial random samples. Then we are using surrogate model prepared in this way to optimize the random initial population of genetic algorithm.

When several iteration of genetic algorithm is done, we are using the individuals from the current population to retrain the surrogate model. It guarantee us that in the most neuralgic part of the decision variable space the surrogate model gives appropriate output values with small error. It destroys also the false extremum values in the model.

This relearning of the surrogate model can be repeated until satisfactory results will be obtained.

5 SURROGATE MODEL – PROJECT DEVELOPING IN POLAND

The Master Thesis was prepared with support of the EEA Financial Mechanism Grants: “Professional Partnership between the Republic of Poland and the Republic of Iceland in the Utilization of Renewable Energy Resources: Graduate Education, Practical Training and Renewable Energy Research”. Important thing to mention is the work plan of the project, which will be continued in Poland during next year.

The whole project methodology can be presented in the following plan:

1. A model was proposed (a gas turbine with recirculation and a steam network).
2. The learning and validation set have been generated on the basis of the MINLP (Mixed-Integer Non-Linear Program) model. The surrogate models have been developed in a MATLAB environment (consistent with other LENI tools). The chosen and developed method is Artificial Neuron Network. For simpler problem (a gas turbine with the recirculation) also mathematical programming was used.

For given method the different parameters were tested and the best solution was chosen. For ANN the parameters examples are:

- Network architecture – type of ANN;
- Number of neurons in the hidden layers;
- Activation function.

The ANN was developed with the Neural Network Toolbox for MATLAB.

3. The surrogate model was built. Some of the questions that was answered are:
 - Optimization is performed through an evolutionist algorithm. Does the surrogate model make enough evaluation of the real model or shall it be completed by more data?
 - How can we choose optimal learning and validation sets?
 - In the decision variables space, what is the validity domain of the surrogate model?
 - What is its precision for a given set of data? The question behind this one is ‘what is the surrogate model deviation, compared to uncertainty influence’?
 - What is the procedure to build such a model? Can it be made in parallel with optimization?
4. Generalize the method (make it useable for any model). This means:
 - Define a test (to check if the model is a possible candidate for such approach);
 - Define which parameters can be set as default parameters, and which have to be defined by the user;
 - Implementing it to build a general tool.
5. Incorporate the method into existing LENI tools.

In the context of this Master Thesis, the goal was to reach point 3 from the research plan. The remaining research points and the development of the surrogate model tool will be continued after the completion of the RES Master Thesis. The final goal of the project is to develop a tool that can be used to build surrogate models for complex energy systems. This tool will have to be user friendly and generic enough to be used on different models by different users (non-expert). However this represents a lot of work that cannot be achieved during four months time. Therefore the constrained goal of this project was to study a detailed model, and develop methods that can be generalized to any kind of problem.

The project will be continued after writing the RES thesis – follow-up work will be used to prepare a master thesis project for the author home university in Poland (academic year 2011-2012, thesis advisor: Dr. Adam Roman). As the result of these two projects the final goal (building surrogate model tool) will be achieved.

The second, very important, part of the project will be developed and implemented in Poland.

Moreover we have to remember that this project is devoted to developing a tool and a methodology which can be useful in other practical problems. This toll will help in analyzing complicated energy systems in faster and more effective way. It can improve their quality, what can be beneficial not only for Poland but for any examined by this tool problem.

6 CONCLUSIONS

The Master Thesis presents methods of preparing the surrogate models for complex energy systems. Different methods were introduced and evaluated. Also possible applications of presented theoretical solutions were shown.

The analysis prepared during Master Thesis provides information about practical implementation methods of surrogate models into the energy system field. The Thesis is an example of implementation of mathematical theory into real world problem.

The final goal of the project is to develop a tool that can be used to build surrogate models for complex energy systems. This tool will have to be user friendly and generic enough to be used on different models by different users (non-expert). In this Master Thesis project first part of the work was done. The detailed models have been studied and results were generalized to any kind of a problem. The different methods: mathematical as well as artificial intelligence methods were incorporated into the surrogate modeling problems. The generalization and conclusions according to different problems connected with surrogate modeling are presented in the Generalization chapter. The conclusion are focused on the following topics: choosing the best training set, method of elimination errors in the training process, validity space of the model and possibilities of connecting two algorithms: genetic optimization and creation of surrogate model.

The prepared surrogate models and general conclusions will be used to develop a new tool which will be incorporated into LENI software. The tool will be implemented at Jagiellonian University in Poland. This tool will provide new possibilities for optimization and analysis strategies of complex energy systems. Using a surrogate model instead of the real one gives the chance to perform more optimization strategies and to perform more complex analysis. This tool will find many subsequent applications and will be used in the future LENI projects.

The automatic tool can improve the quality and the speed of analysis of energy systems connected with polish condition (a few of this type analysis were prepared in LENI). Moreover the information presented in this Thesis can be used in preparing surrogate models for specific problems related to Poland or other countries.

REFERENCES

- Beale, M., Hagan, M., & Demuth, H. (2010). *Neural Network Toolbox™ 7, User's Guide*. online: The MathWorks.
- Bernacki, M., Włodarczyk, P., & Gołda, A. (2004, September 6). Retrieved January 15, 2011, from Principles of training multi-layer neural network using backpropagation: http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html
- Bernier, E., Marechal, F., & Samson, R. (2009). Multi-objective design optimization of a natural gas-combined cycle with carbon dioxide capture in a life cycle perspective. *Energy*.
- Caballero, J., & Grossmann, I. E. (2008). An Algorithm for the Use of Surrogate Models in Modular Flowsheet Optimization. *AIChE J* , 54, 2633.
- Dideková, Z. (2009, October 14). *Portal Posterus*. Retrieved January 15, 2011, from Neural control of non-linear processes designed by genetic algorithms: <http://www.posterus.sk/?p=3138>
- Dubuis, M., & Tock, L. (2010). *GTCO2 Project: Progress report*.
- Fernandes, F. (2006). Optimization of Fischer-Tropsch Synthesis Using Neural Networks. *Chem. Eng. & Tech.* , 29,449.
- Girardin, L., Dubuis, M., & Marechal, M. (n.d.). On the use of process integration techniques to generate optimal steam cycle configurations for the power plant industry.
- Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley.
- Kemp, I. C. (2007). *Pinch Analysis and Process Integration*. Oxford, U.K.: Butterworth-Heinemann.
- Kincaid, D., & Chene, W. (2002). *Numerical Analysis: Mathematics of Scientific Computing, 3rd Edition*. Providence, RI.: American Mathematical Society.

Linhoff, B., & Townsend, D. (1982). *A user guide on process integration for the efficient use of energy*. The Institution of Chemical Engineers.

MATLAB documentation, pchip command.

Orr, G. (1999). *Willamette University web page*. Retrieved January 15, 2011, from Neural Networks: <http://www.willamette.edu/~gorr/classes/cs449/intro.html>

Sivanandam, N., Sumathi, S., & Deepa, S. (2006). *Introduction To Neural Networks Using MATLAB 6.0*. Dehli: Tata Mgraw Hill.

Sreekanth, J., & Datta, B. (2010). Multi-objective management of saltwater intrusion in coastal aquifers using genetic programming and modular neural network based surrogate models. *Journal of Hydrology* , 393, 245-256.

Stergiou, C., & Siganos, D. (n.d.). Retrieved January 15, 2011, from NEURAL NETWORKS : http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html

Stranz, D., & Martin, L. (1997, September 25). Retrieved January 15, 2011, from Derivation of Peptide Sequence from Mass Spectral Data using the Genetic Algorithm: <http://www.abrf.org/JBT/Articles/JBT0004/JBT0004.html>

Tenne, Y., & Armfield, S. (2008). A framework for memetic optimization using variable global and local surrogate models. *SOFT COMPUTING* , Volume 13, Numbers 8-9, 781.

Won, K., & Ray, T. (2005). A Framework for Design Optimization Using Surrogates. *Eng. Optim.* , 37,685. .

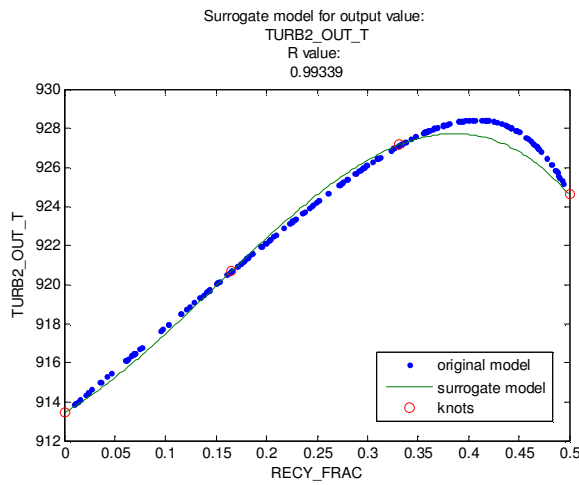
Genetic algorithms - Introduction:

<http://cgm.cs.mcgill.ca/eden/PrimitiveGenetics/Introduction.htm>. Retrieved January 15, 2011.

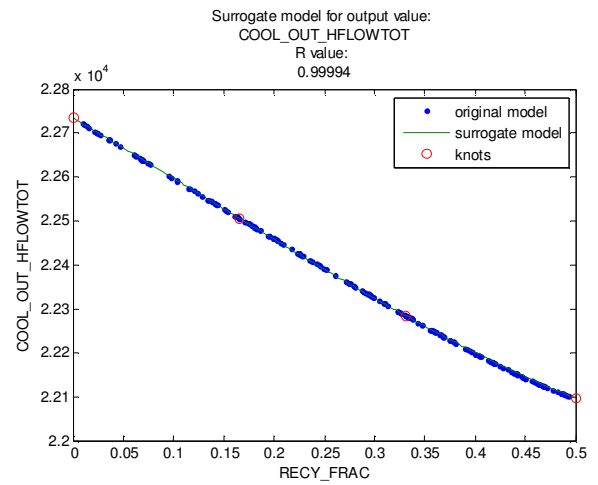
Genetic Algorithms : General Idea: <http://www.sussex.ac.uk/space-science/Nature/ga.html>
Retrieved January 15, 2011.

APPENDIX A

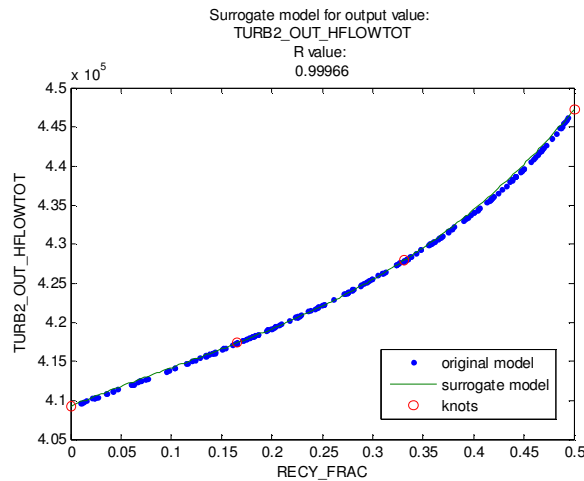
Comparison of surrogate model based on spline interpolation and original one variable model - gas turbine power plants designs with CO₂ capture with Flue Gas Recirculation (FGR). The original model is represented by set of 200 randomly chosen points determined by the value of decision variable – recirculation ratio. Appendix consist graphs for all output parameters analyzed in the Thesis.



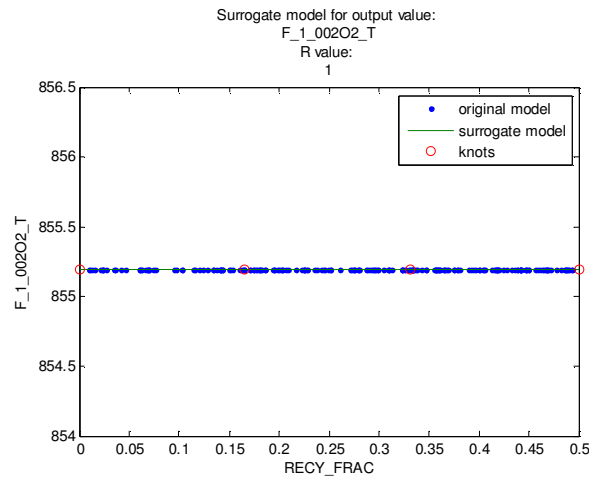
A. 1 Surrogate model on basis of 4 knots for TURB2_OUT_T



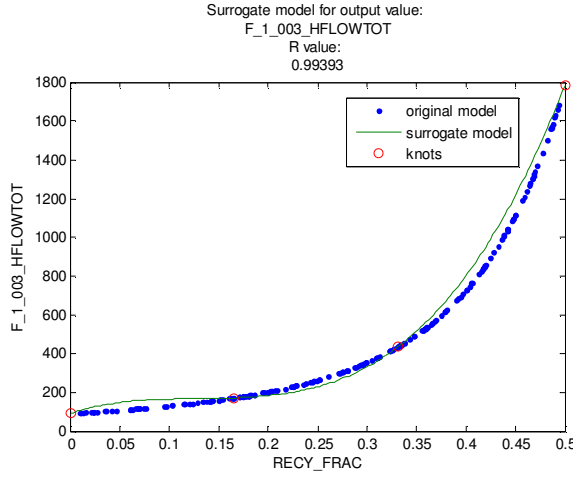
A. 3 Surrogate model on basis of 4 knots for COOL_OUT_HFLOWTOT



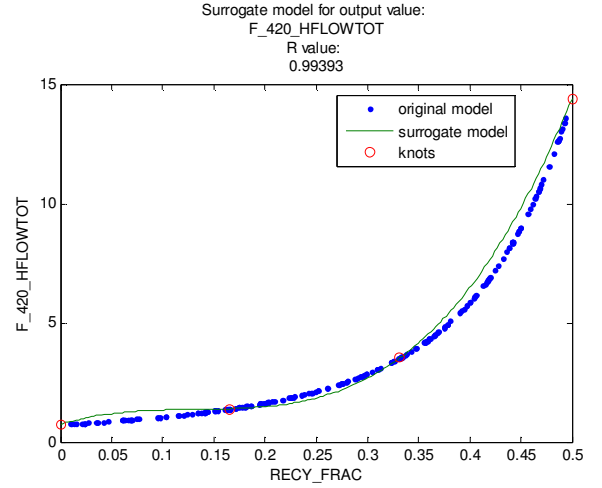
A. 2 Surrogate model on basis of 4 knots for TURB2_OUT_HFLOWTOT



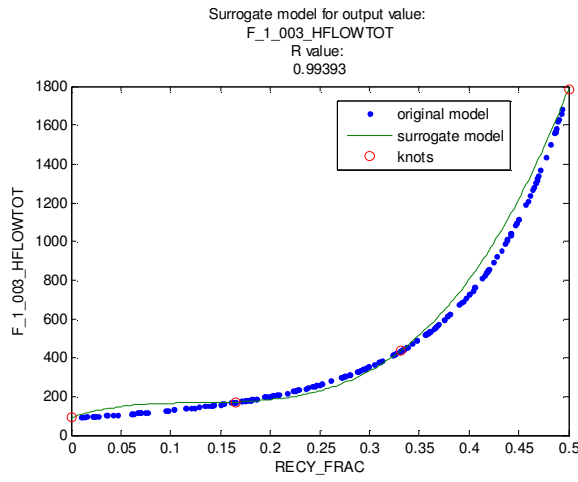
A. 4 Surrogate model on basis of 4 knots for F_1_002O2_T



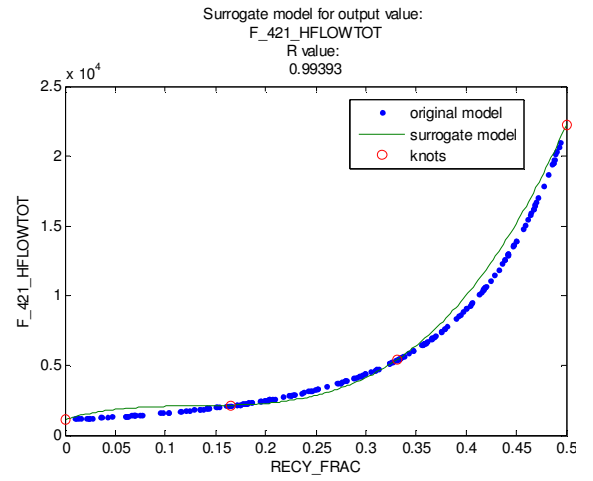
A. 5 Surrogate model on basis of 4 knots
for F_1_002O2_HFLOWTOT



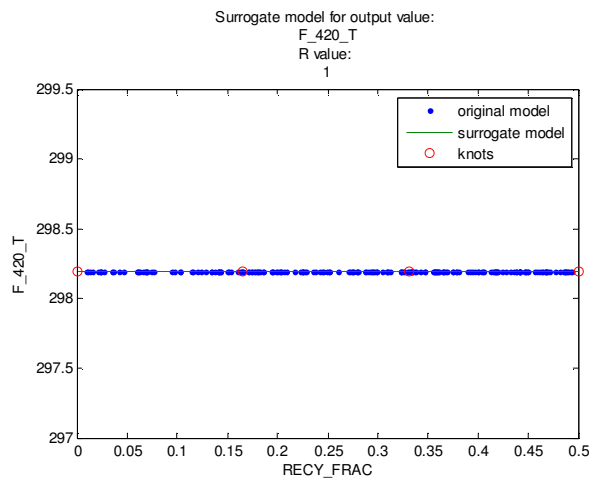
A. 8 Surrogate model on basis of 4 knots
for F_420_HFLOWTOT



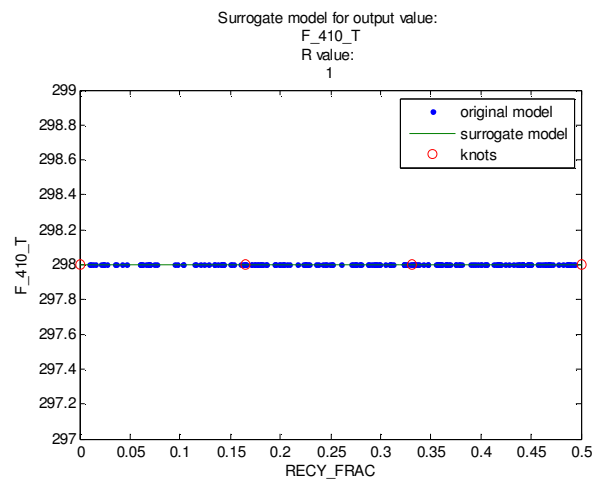
A. 6 Surrogate model on basis of 4 knots
for F_1_003_HFLOWTOT



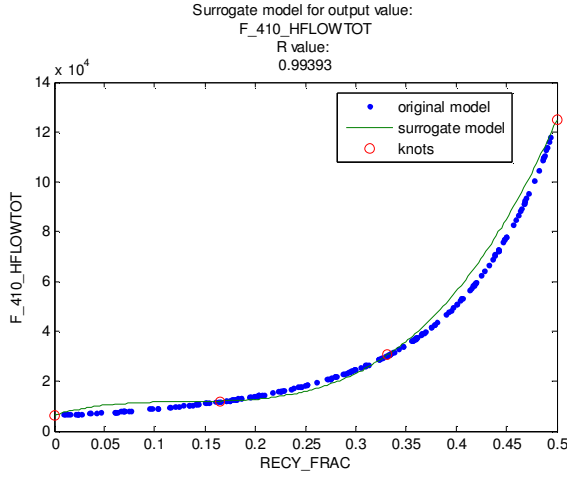
A. 9 Surrogate model on basis of 4 knots
for F_421_HFLOWTOT



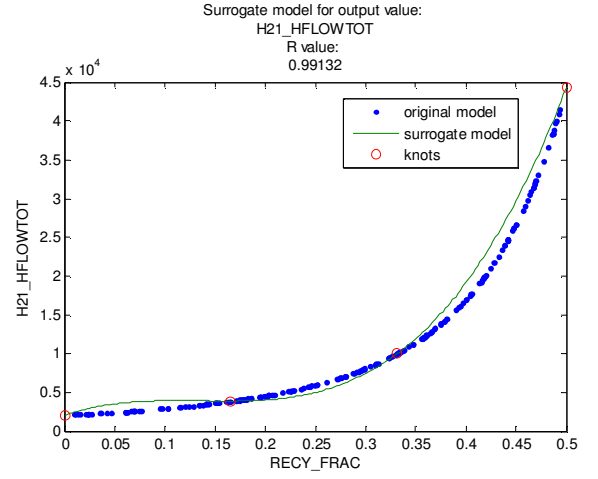
A. 7 Surrogate model on basis of 4 knots
for F_420_T



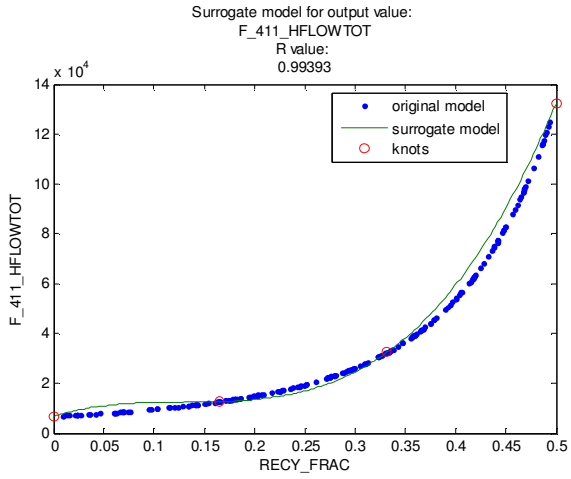
A. 10 Surrogate model on basis of 4
knots for F_410_T



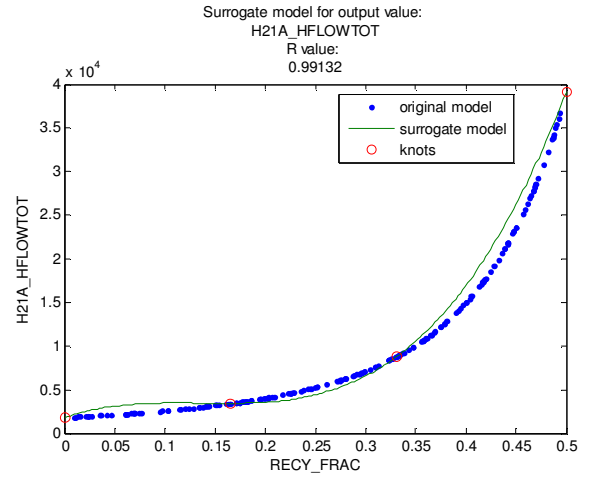
A. 11 Surrogate model on basis of 4 knots for F_410_HFLOWTOT



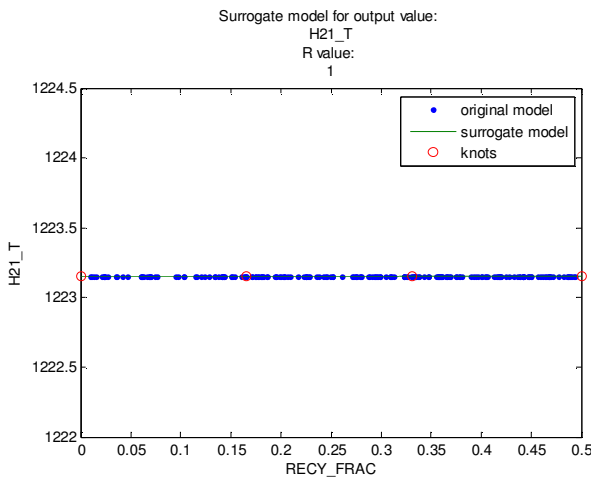
A. 14 Surrogate model on basis of 4 knots for H21_HFLOWTOT



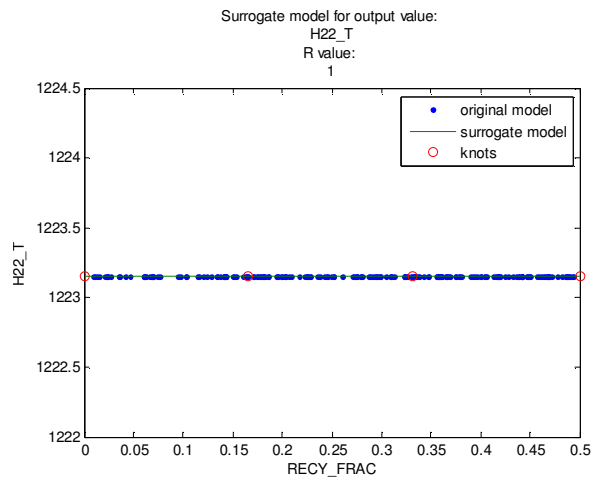
A. 12 Surrogate model on basis of 4 knots for F_411_HFLOWTOT



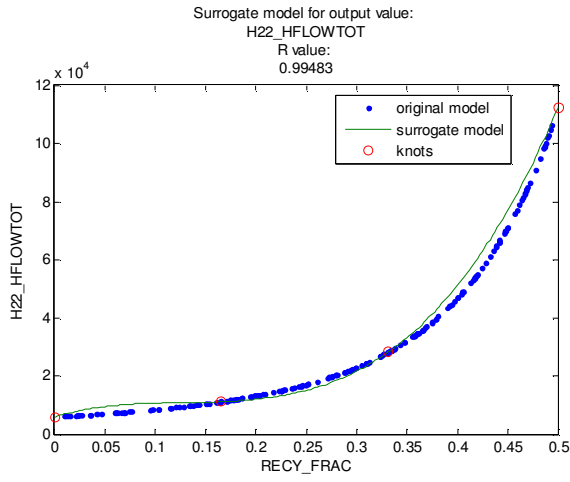
A. 15 Surrogate model on basis of 4 knots for H21A_HFLOWTOT



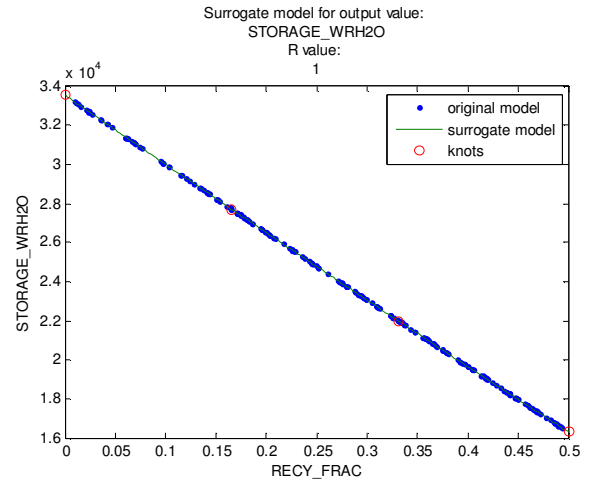
A. 13 Surrogate model on basis of 4 knots for H21_T



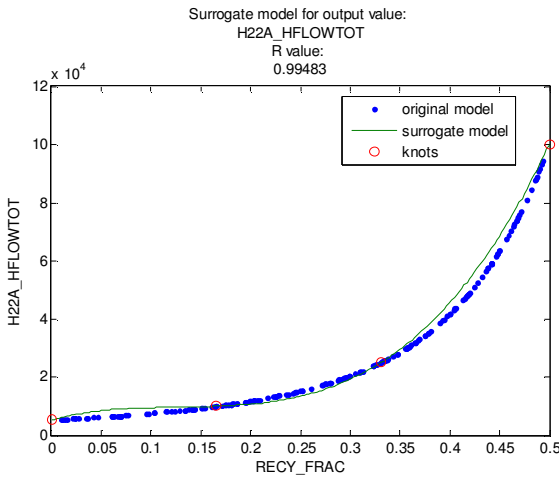
A. 16 Surrogate model on basis of 4 knots for H22_T



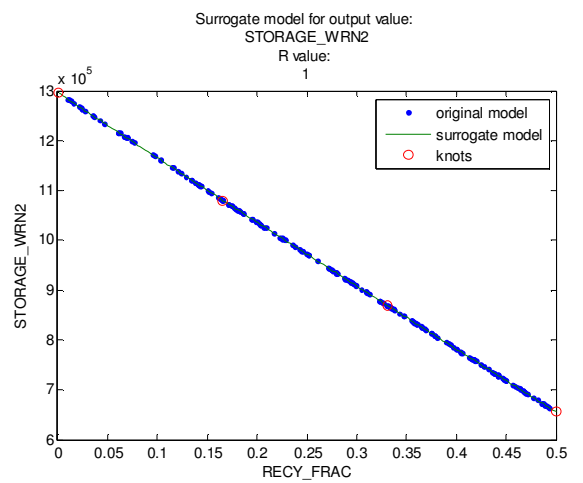
A. 17 Surrogate model on basis of 4 knots for H22_HFLOWTOT



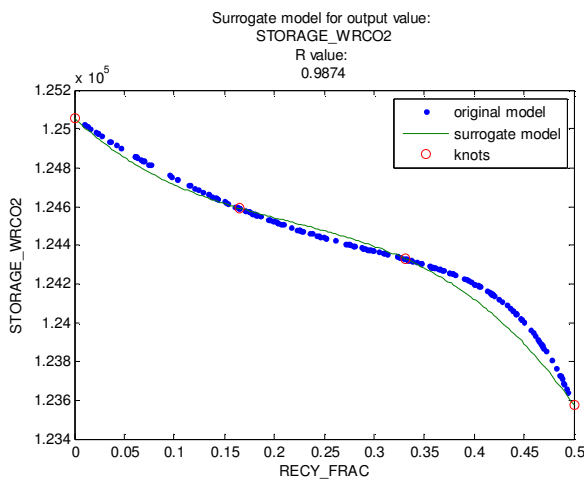
A. 20 Surrogate model on basis of 4 knots for STORAGE_WRH2O



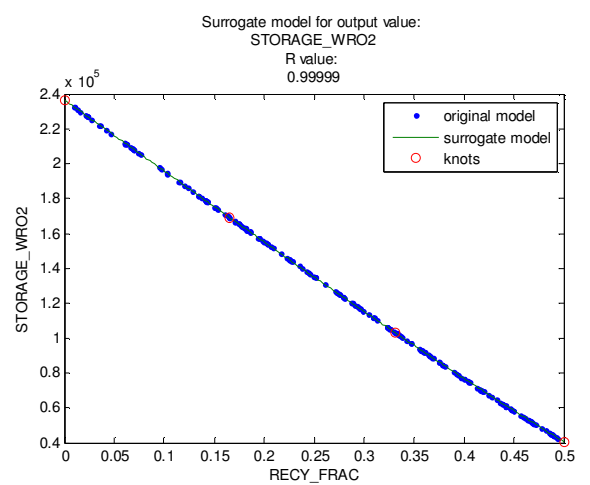
A. 18 Surrogate model on basis of 4 knots for H22A_HFLOWTOT



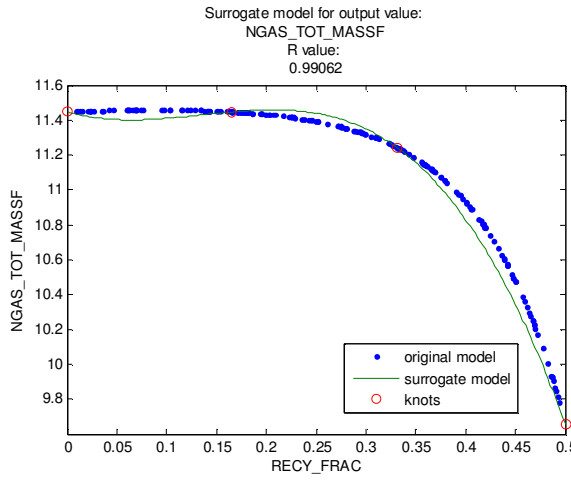
A. 21 Surrogate model on basis of 4 knots for STORAGE_WRN2



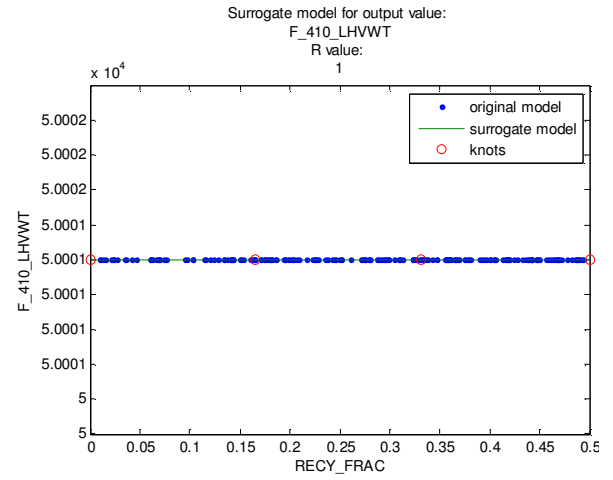
A. 19 Surrogate model on basis of 4 knots for STORAGE_WRCO2



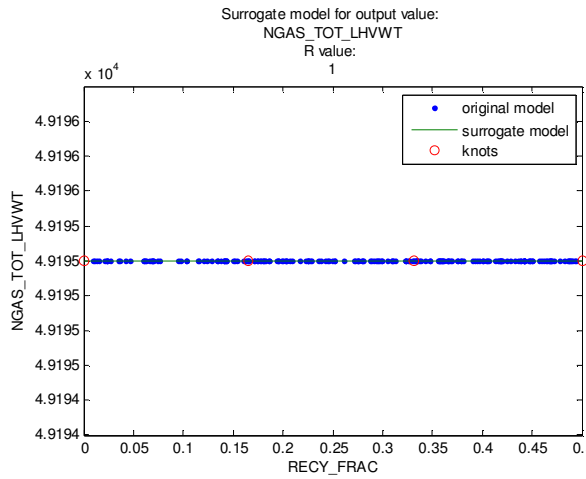
A. 22 Surrogate model on basis of 4 knots for STORAGE_WRO2



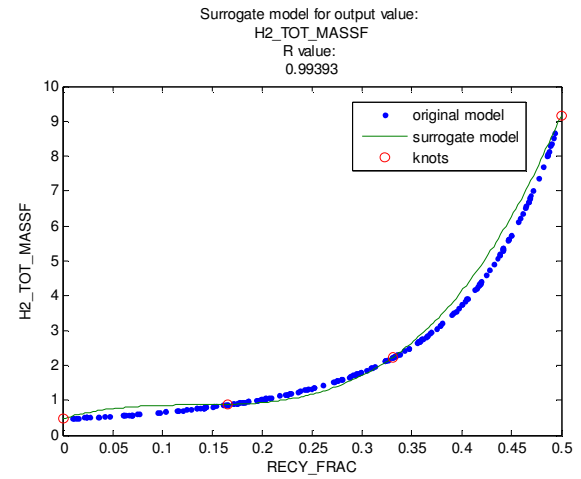
A. 23 Surrogate model on basis of 4 knots for NGAS_TOT_MASSF



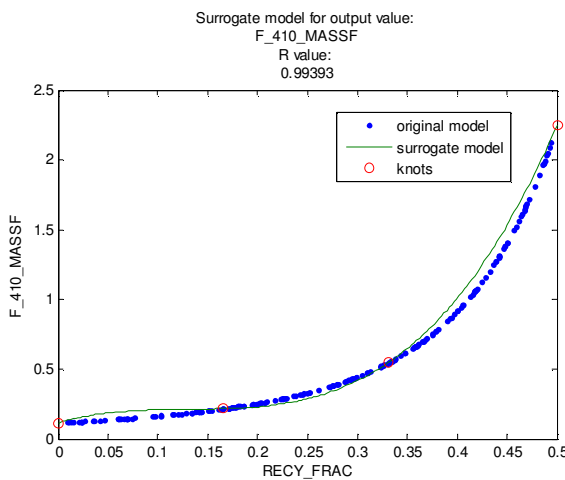
A. 26 Surrogate model on basis of 4 knots for F_410_LHVWT



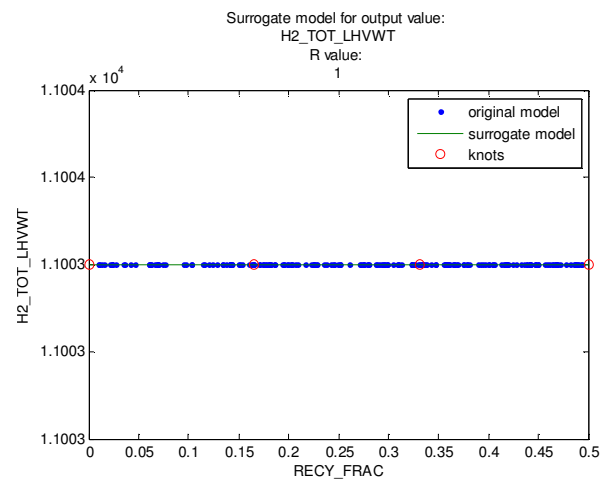
A. 24 Surrogate model on basis of 4 knots for NGAS_TOT_LHVWT



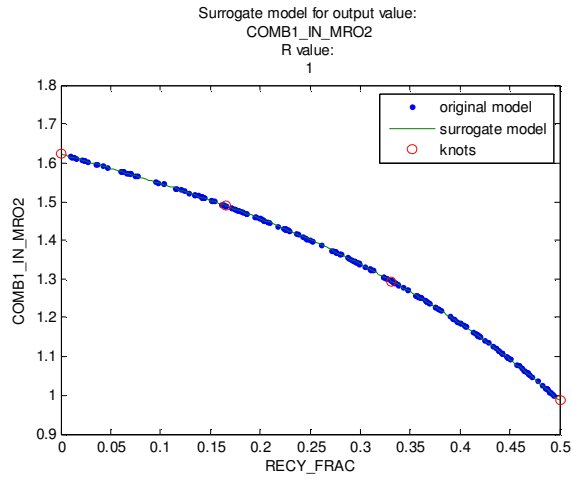
A. 27 Surrogate model on basis of 4 knots for H2_TOT_MASSF



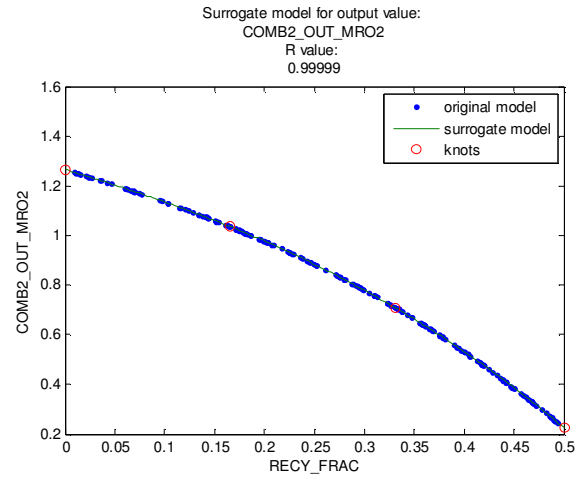
A. 25 Surrogate model on basis of 4 knots for F_410_MASSF



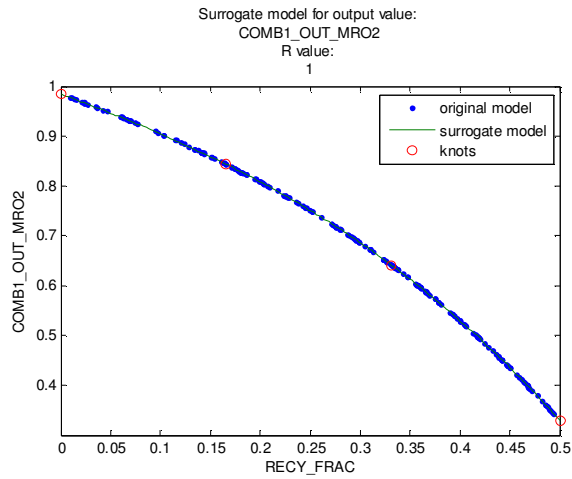
A. 28 Surrogate model on basis of 4 knots for H2_TOT_LHVWT



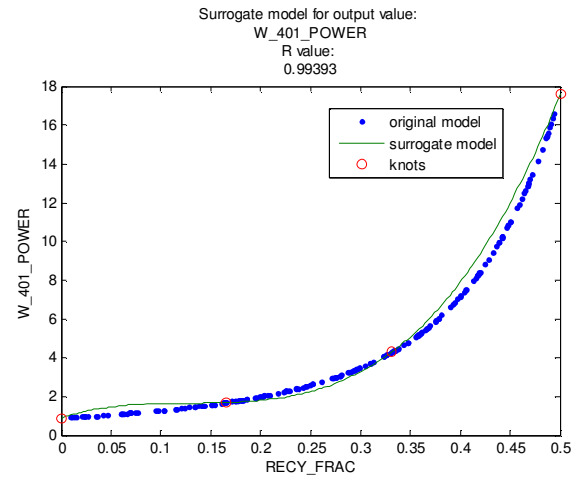
*A. 29 Surrogate model on basis of 4 knots
for COMB1_IN_MRO2*



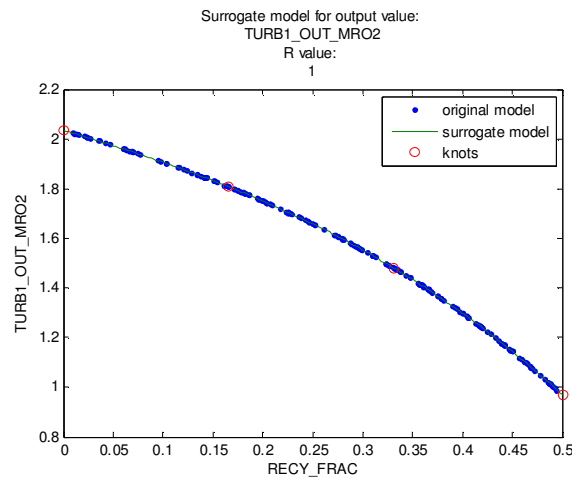
*A. 32 Surrogate model on basis of 4 knots
for COMB2_OUT_MRO2*



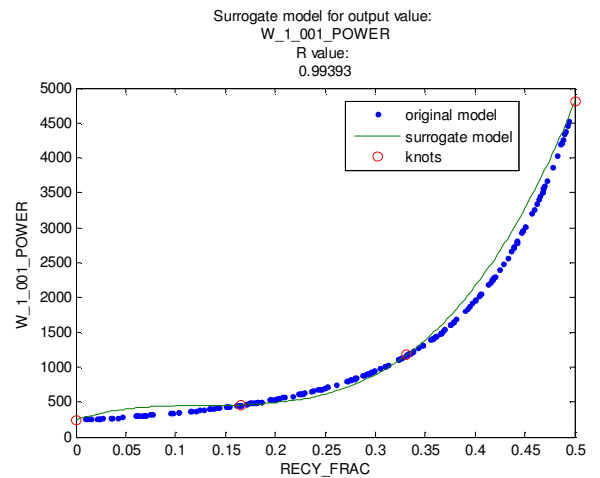
*A. 30 Surrogate model on basis of 4 knots
for COMB1_OUT_MRO2*



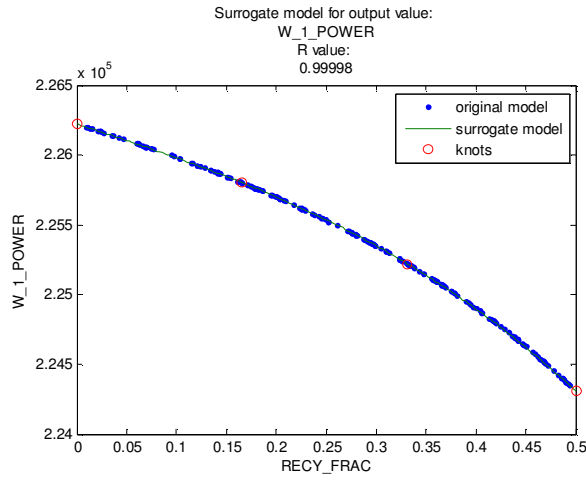
*A. 33 Surrogate model on basis of 4 knots
for W_401_POWER*



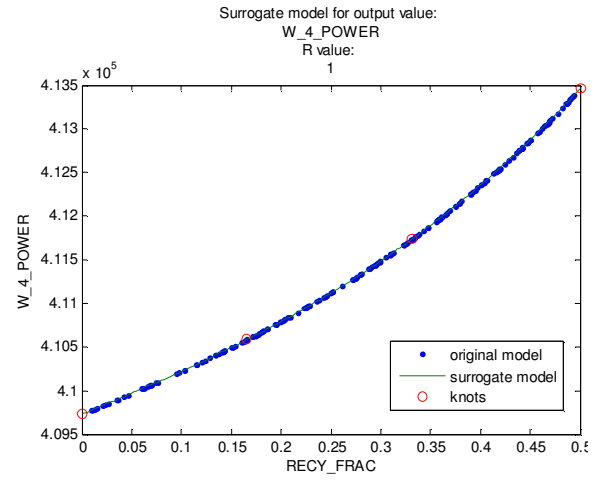
*A. 31 Surrogate model on basis of 4 knots
for TURB1_OUT_MRO2*



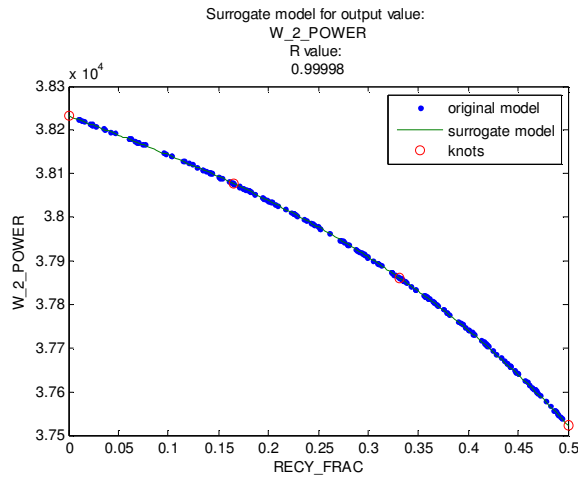
*A. 34 Surrogate model on basis of 4 knots
for W_1_001_POWER*



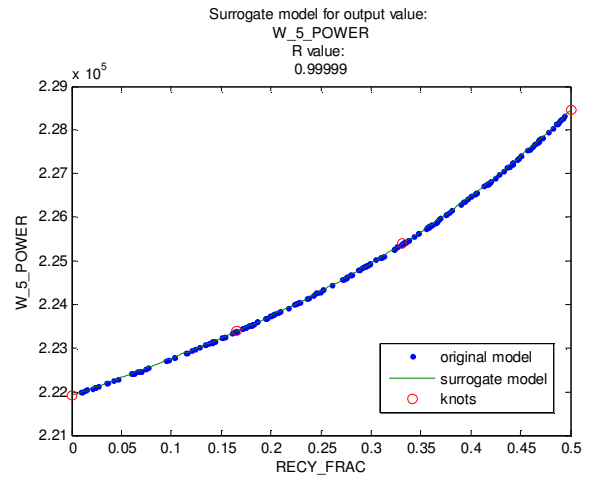
A. 35 Surrogate model on basis of 4 knots for W_1_POWER



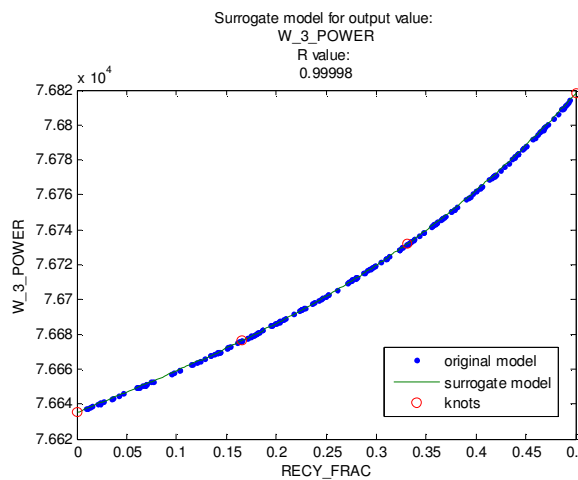
A. 38 Surrogate model on basis of 4 knots for W_4_POWER



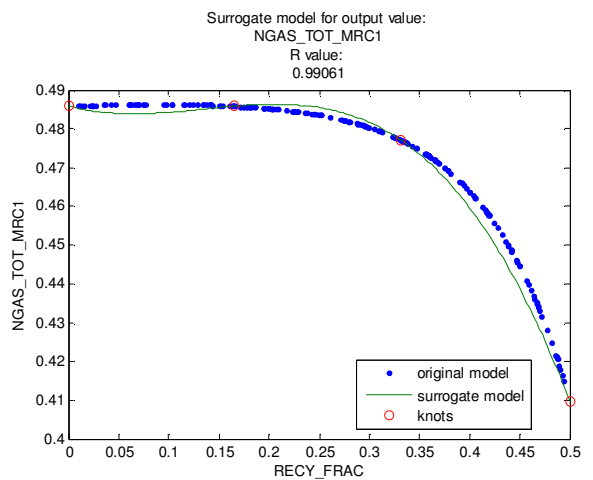
A. 36 Surrogate model on basis of 4 knots for W_2_POWER



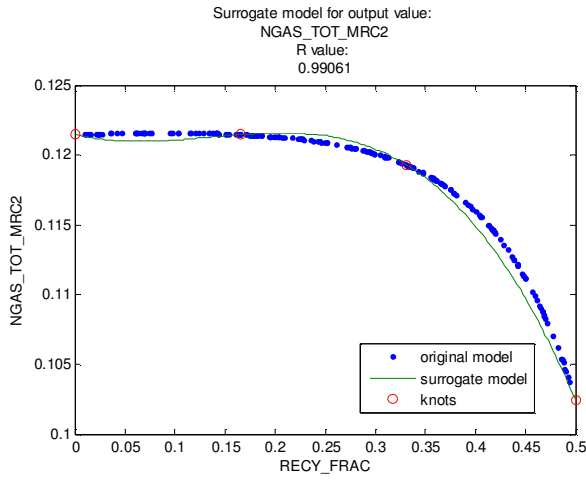
A. 39 Surrogate model on basis of 4 knots for W_5_POWER



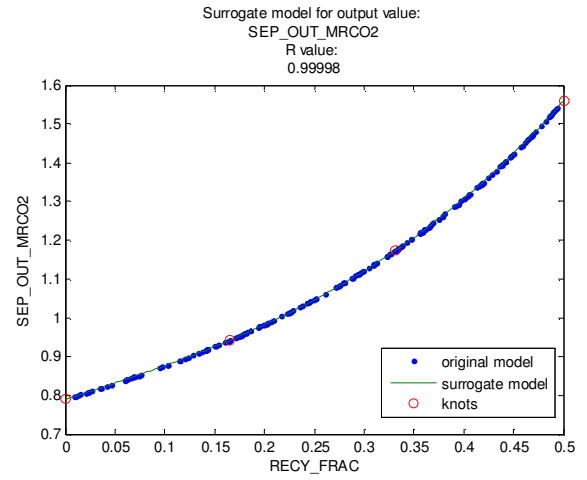
A. 37 Surrogate model on basis of 4 knots for W_3_POWER



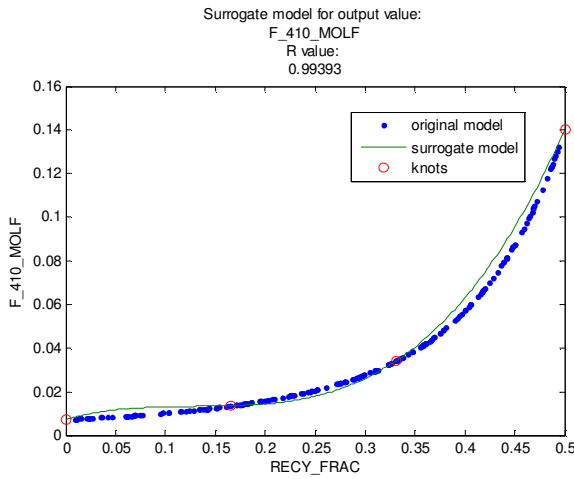
A. 40 Surrogate model on basis of 4 knots for $NGAS_TOT_MRC1$



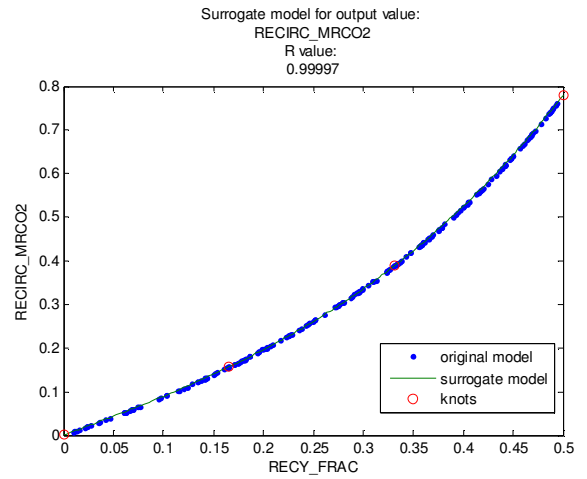
A. 41 Surrogate model on basis of 4 knots for NGAS_TOT_MRC2



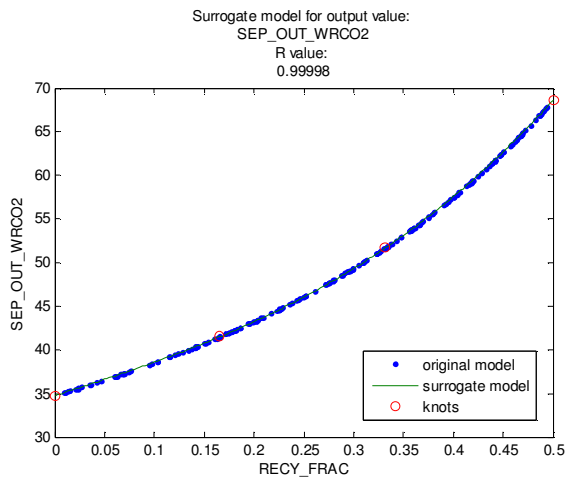
A. 44 Surrogate model on basis of 4 knots for SEP_OUT_MRCO2



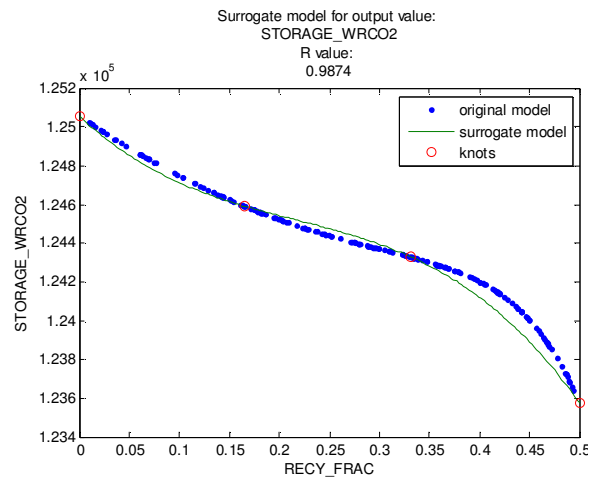
A. 42 Surrogate model on basis of 4 knots for F_410_MOLF



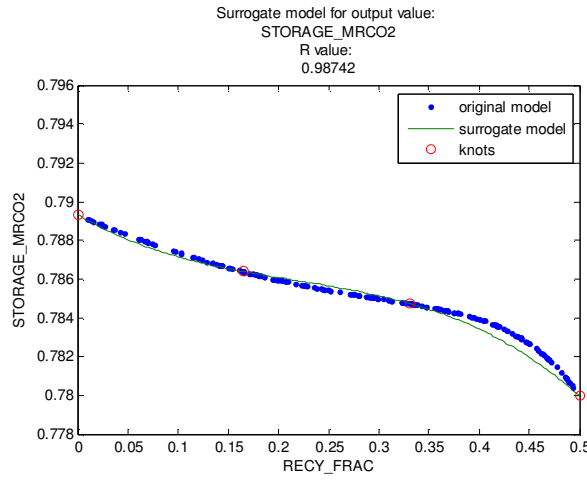
A. 45 Surrogate model on basis of 4 knots for RECIRC_MRCO2



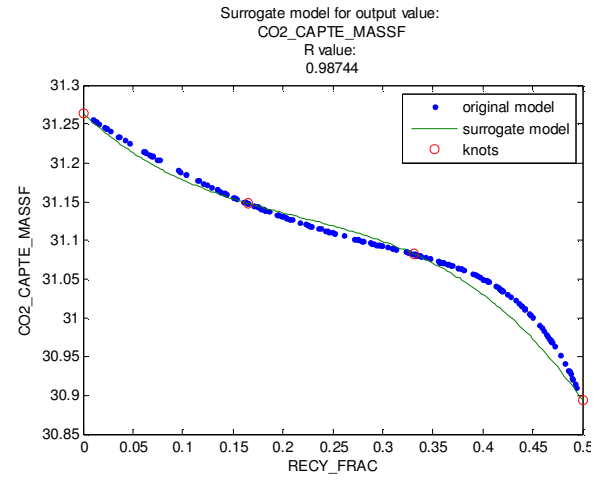
A. 43 Surrogate model on basis of 4 knots for SEP_OUT_WRCO2



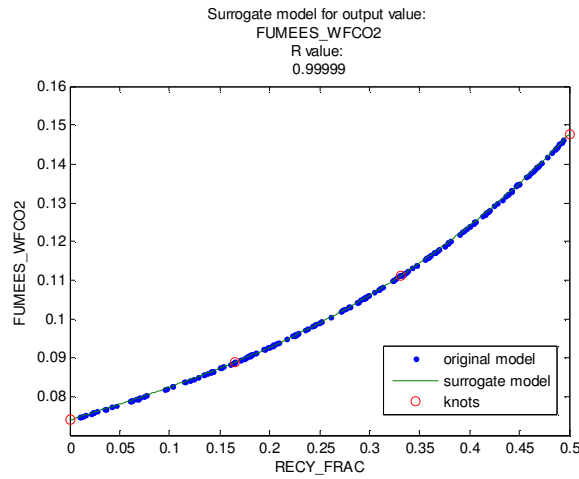
A. 46 Surrogate model on basis of 4 knots for STORAGE_WRCO2



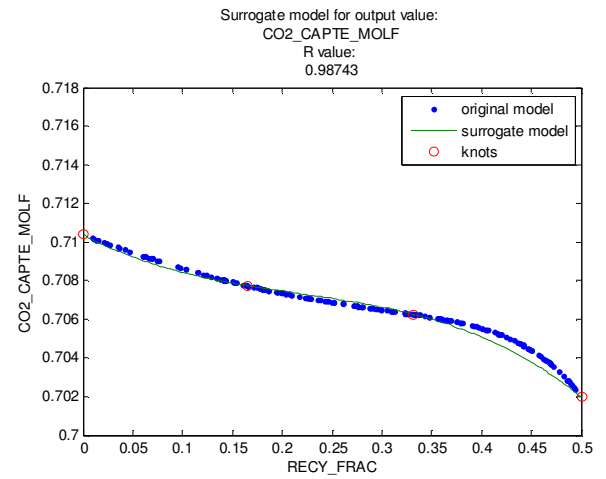
A. 47 Surrogate model on basis of 4 knots for STORAGE_MRCO2



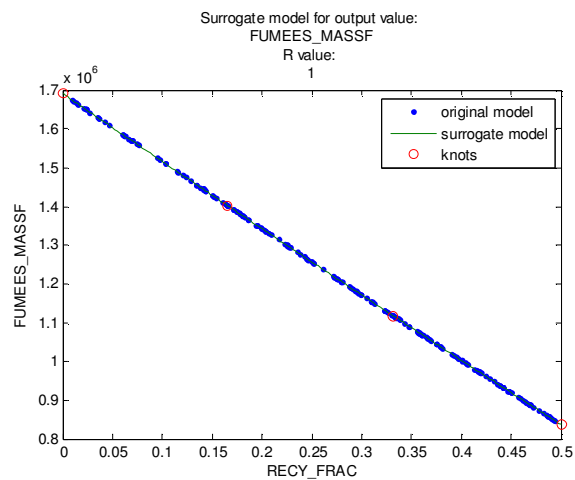
A. 50 Surrogate model on basis of 4 knots for CO2_CAPTE_MASSF



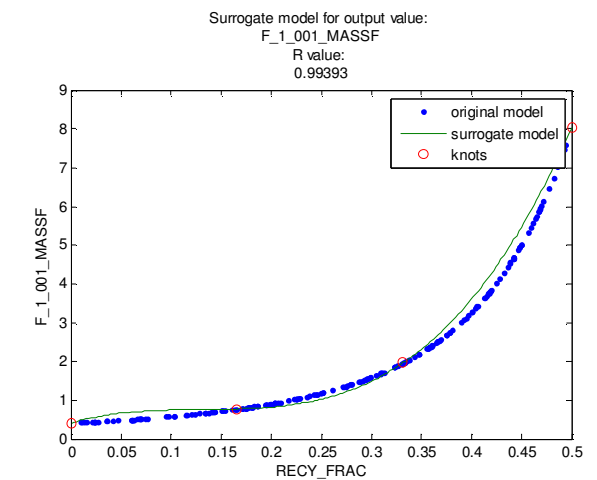
A. 48 Surrogate model on basis of 4 knots for FUMEES_WFCO2



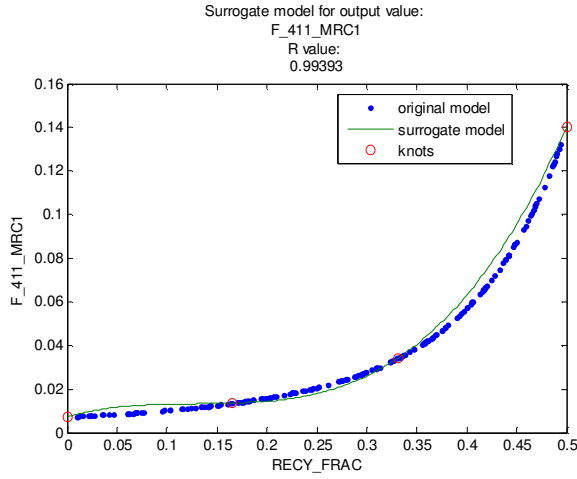
A. 51 Surrogate model on basis of 4 knots for CO2_CAPTE_MOLF



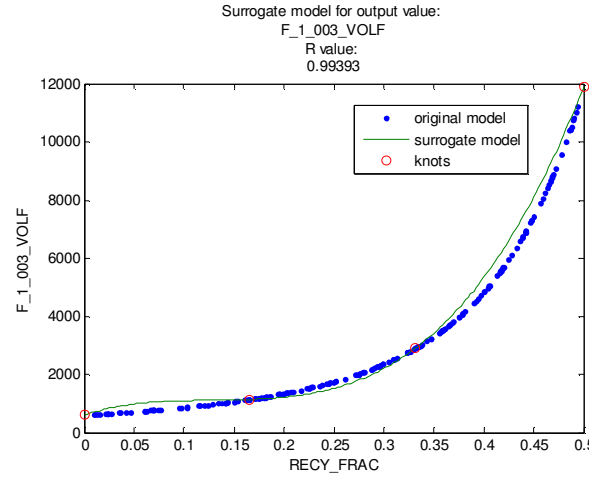
A. 49 Surrogate model on basis of 4 knots for FUMEES_MASSF



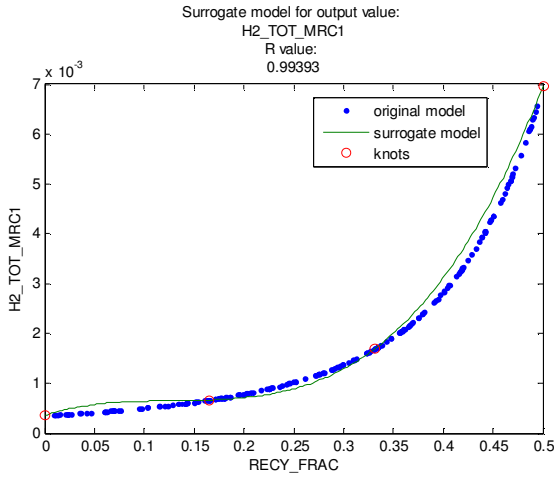
A. 52 Surrogate model on basis of 4 knots for F_1_001_MASSF



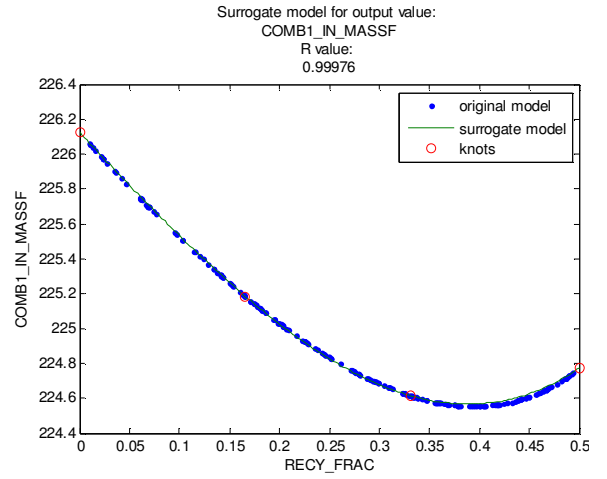
A. 53 Surrogate model on basis of 4 knots for F_411_MRC1



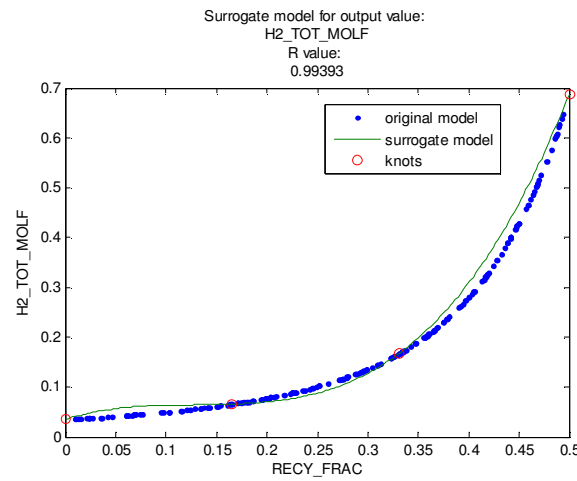
A. 56 Surrogate model on basis of 4 knots for F_1_003_VOLF



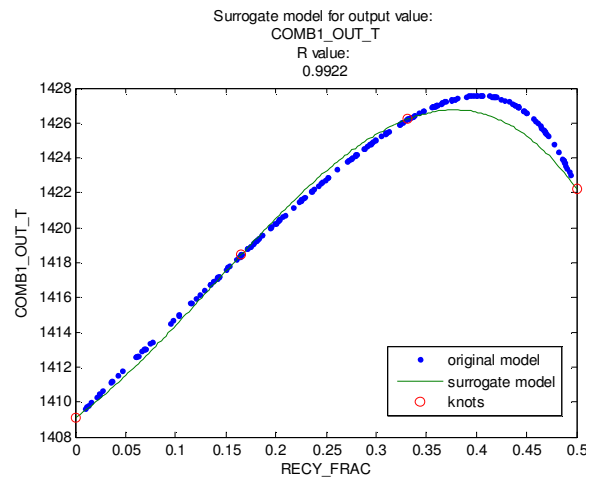
A. 54 Surrogate model on basis of 4 knots for H2_TOT_MRC1



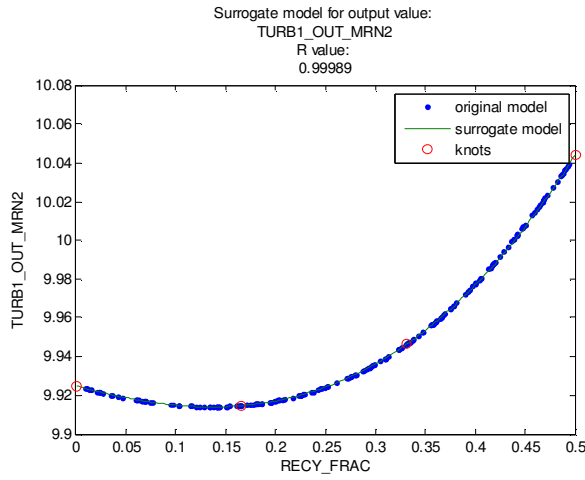
A. 57 Surrogate model on basis of 4 knots for COMB1_IN_MASSF



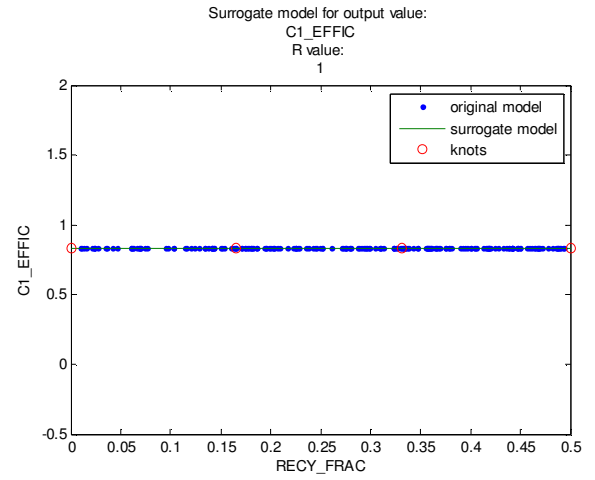
A. 55 Surrogate model on basis of 4 knots for H2_TOT_MOLF



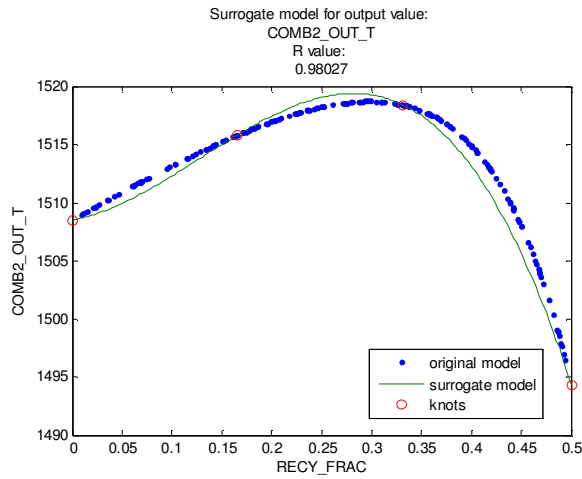
A. 58 Surrogate model on basis of 4 knots for COMB1_OUT_T



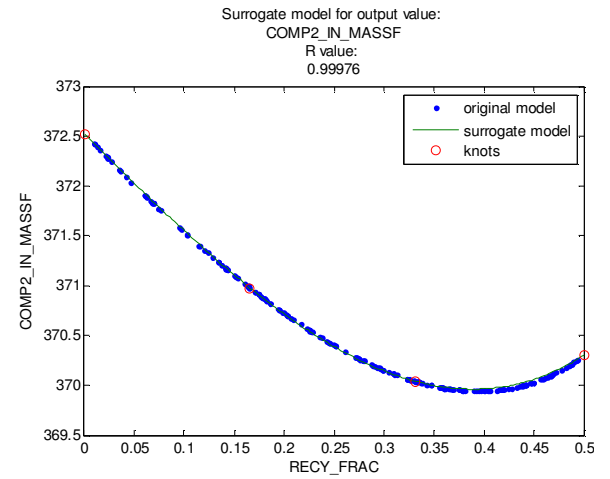
*A. 59 Surrogate model on basis of 4 knots
for TURB1_OUT_MRN2*



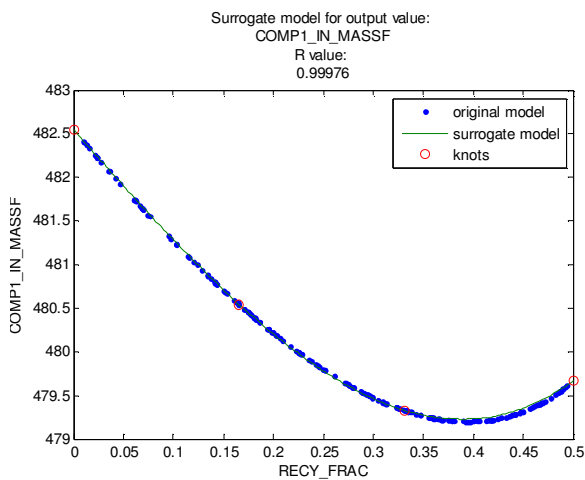
*A. 62 Surrogate model on basis of 4 knots
for C1_EFFIC*



*A. 60 Surrogate model on basis of 4 knots
for COMB2_OUT_T*



*A. 63 Surrogate model on basis of 4 knots
for COMP2_IN_MASSF*



*A. 61 Surrogate model on basis of 4 knots
for COMP1_IN_MASSF*