

# SITUATING NEURAL NET AND FUZZY LOGIC MODELLING IN A UNIFORM SCHEME

JÜRGEN VAN GORP

JOHAN SCHOUKENS

*Department ELEC, Vrije Universiteit Brussel* *Department ELEC, Vrije Universiteit Brussel*  
*Pleinlaan 2, B-1050 Brussel, Belgium* *Pleinlaan 2, B-1050 Brussel, Belgium*  
*Email: Jurgen.Van.Gorp@vub.ac.be* *Email: Johan.Schoukens@vub.ac.be*

This paper introduces a general identification scheme for system modelling, clearly separating the model, cost function and optimization scheme. Although Neural Networks (NN) and Fuzzy Logic (FL) originate from the Artificial Intelligence domain, it is shown that they fit well in the identification scheme as just another nonlinear model. This paper contributes to the demystification of NN and FL. The gain for both modelling schemes is that they can profit from present knowledge within nonlinear system identification, such as cost function properties and optimization schemes. The gain for the classic identification theory is that techniques that do make sense, e.g. early stopping, prove to be applicable within linear system identification also. By placing FL and NN in a uniform modelling scheme, next to classic modelling, this paper also tends to contribute to the acceptance of both modelling domains within classic linear theory.

## I. A SCHEME FOR NONLINEAR MODELLING

Regardless of the fact that someone uses linear modelling [17], a NN mapping [4] [26] [29], a FL system [39] or any other nonlinear model, regardless of the fact that someone uses White Box, Black Box or Grey Box [7] [18] [26] modelling, a model is just a mathematical mapping from a given input to a given output, dependent on static or time-dependent model parameters.

Independent of the model used (linear, FL, RBF, Black Box, White Box, ...), the goal of modelling is to find the true model parameters  $\theta^*$  that minimize a cost function  $C$  which gives a measure how well the model fits a set of measurements.

In general this modelling (also called *identification*) is done in three major steps. First, basic decisions must be taken prior to the modelling: which measurements are needed for training? Which linear or nonlinear model is used, and which cost function is minimized? Every single one of these three choices has a direct influence on the subsequent modelling steps. Therefore, these three basic decisions are the most important choices to be made. The second step considers the minimization of the cost function and is more a technical problem once the three basic choices are made. The parameters are initialized and an optimization scheme is chosen. The experimenter must also decide upon when to stop optimization. Third comes a validation of the model. Usually the model is tested on a separate test set, or on new data. Optionally robustness and stability analysis is performed as a validation of the model. The analysis of the model robustness and stability doesn't necessarily come at the end of the modelling. It is possible to enforce these criteria even before the validation section by choosing the proper cost function.

The way how these sections are related to each other, is given in Fig. 1. In the following paragraphs, the sub parts of this scheme are discussed more in detail.

## II. DISCUSSION ON THE DIFFERENT SECTIONS OF THE MODELLING SCHEME

### A. Choose your model structure

There have been many fierce discussions in favour or against different models [1]. Yet, for any given continuous function that operates in  $\mathbb{R}$  it is possible to prove universal approximation properties (Kolmogorov's theorem) [6] [11] [13] [36] [38], such that from a theoretical point of view any basic BB model would fit the task.

From a practical point of view, however, the best fit is the one that needs the least number of parameters. A sine wave is best fitted using a sine model while an exponential is best fitted using an exponential model. Locally linear systems are best fitted using e.g. splines or a FL system. The best model that can be used to fit a functional, is the functional itself. Choosing the model is fully based upon typical transfer functions, or convergence properties whenever the parameters are optimized.

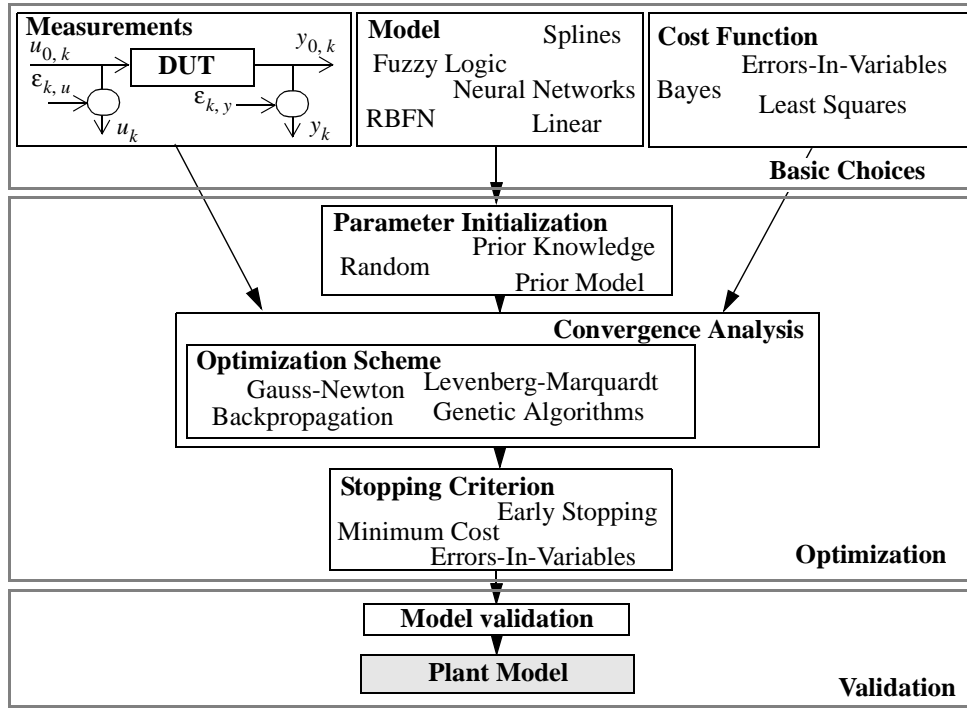


Fig. 1 Modeling scheme

Systems that are linear in the parameters

The most simple linear system is given as  $\mathbf{y} = \mathbf{A}\mathbf{u}$  such that  $\theta = \mathbf{A}$  and e.g. polynomial mappings have a transfer function  $\mathbf{y} = \mathbf{a}_0 + \mathbf{a}_1\mathbf{u} + \mathbf{a}_2\mathbf{u}^T + \dots$  such that  $\theta = [\mathbf{a}_0 \ \mathbf{a}_1 \ \mathbf{a}_2 \dots]^T$ . In both cases the transfer function is linear in the parameters, even if the relationship between the inputs  $\mathbf{u}$  and outputs  $\mathbf{y}$  is nonlinear. The transfer function is globally defined and the second derivative with respect to the parameters equals zero.

Neural networks

There is no general definition for NN systems that covers all possible NN topologies. Many forms exist, the most popular being the Multilayer Perceptron architecture with a transfer function  $\mathbf{y} = \sigma_1 \mathbf{A}_1 (\sigma_2 (\dots \sigma_n (\mathbf{A}_n \mathbf{u} + \beta_n) \dots + \beta_2) + \beta_1)$  with  $\sigma_i$  a nonlinear function that is applied element wise (e.g.  $\sigma(x) = \tanh(x)$ ). The parameter space is chosen as  $\theta = [\mathbf{A}_1(\cdot); \beta_1(\cdot); \mathbf{A}_2(\cdot); \beta_2(\cdot); \dots \mathbf{A}_n(\cdot); \beta_n(\cdot)]$ .

The input/output transfer function is typically a soft nonlinear relationship. Sometimes hard-bounded functions are used, such as  $\sigma(x) = \text{sat}(x)$  or  $\sigma(x) = \text{sign}(x)$ , mainly for pattern recognition [4] and classification. The optimization of the parameters suffers from local minima and is often badly conditioned.

Fuzzy Logic and Neurofuzzy systems

FLS vary a lot with respect to the transfer function, just as is the case for NN. Modelling is typically done with a fuzzification part, an inference part and a defuzzification part. Different models exist for each sub part. Two models rule within Fuzzy Logic: the Mamdani systems and the Takagi-Sugeno systems. The Takagi-Sugeno model is defined in [22] as

$$y = \frac{\sum_{i=1}^R [b_i(\mathbf{u})\mu_i(\mathbf{u})]}{\sum_{i=1}^R [\mu_i(\mathbf{u})]} \quad (1)$$

with  $\mu_i(\mathbf{u})$  the membership degree of the fuzzified input and  $b_i$  a  $\mathbf{u}$ -dependent functional, e.g. the relationship  $b_i = a_{i,0} + a_{i,1}\mathbf{u} + a_{i,2}\mathbf{u}^T + \dots$ .  $R$  is the number of rules that make up the inference mechanism. The Mamdani model defined as a Generalized Additive Fuzzy System in [13] uses the special case that  $b_i = a_{i,0}$ .

Usually FL systems are configured as locally linear systems with a smooth interpolation from one operating point to the other. The parameter space  $\theta$  includes the  $a_{i,j}$ , the rules that form the rule base (also called Knowledge Base), and the specific parameters that form the input and output fuzzy sets. The optimization depends on the chosen parameters, but is difficult due to local minima and redundancy within the many parameters. The latter is usually overcome by learning

each sub part separately, or by fixing a sub part, e.g. the Knowledge Base.

### The choice of a model

To fix the ideas, Fig. 2 shows a few transfer functions based on a 2-input, single output model. The figure shows the output of six different models, with the phase plane as the input. A normalized input  $u \in [-1, 1]$  and its normalized first derivative  $\dot{u} \in [-1, 1]$  are used for a mapping  $f \in [-1, 1]$  and the typical transfer functions are shown.

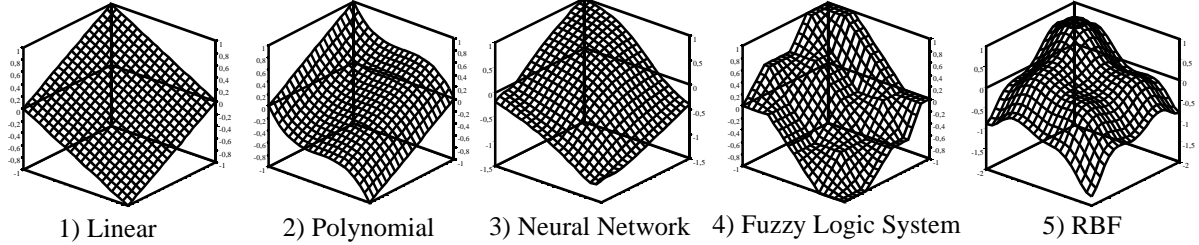


Fig. 2 Typical behaviour of transfer functions of different models

The above list is not in the least exhaustive. Sometimes very dedicated White Box models or complex Black Box models are used, such as Feature Space Mapping, Support Vector Machines, triangular nonlinear structures or saturating linear systems [10]. The very general definitions for NN and FL makes them appropriate for mixed models, such as Neurofuzzy systems [37] or mixed with PI and PD controllers. Overviews of other nonlinear systems are given in [8] [15] [18] and [28].

All of these models are based on the same principle: how to provide a mathematical relationship between a given input and a given output? The remaining task is the choice of the model parameters that should be learned or optimized.

### B. The need of carefully choosing the cost function

With no doubt, the least squares (LS) cost function is the most used to date. The cost function is easy to comprehend, and easy to apply. Define  $n$  as the length of the estimated output vector and  $N$  as the number of MIMO input-output measurement pairs  $(\mathbf{u}_k, \mathbf{y}_k)$  with  $\mathbf{u}_k \in \mathbb{R}^m$  and  $\mathbf{y}_k \in \mathbb{R}^n$ . The LS cost function is then defined as

$$C_{LS} \triangleq \frac{1}{N} \sum_{k=1}^N \left\{ \frac{1}{n} \sum_{i=1}^n [(f_i(\mathbf{u}_k, \theta) - y_{k,i})^2] \right\}. \quad (2)$$

Strictly spoken, the  $1/n$  and  $1/N$  factors are only needed to make the cost independent of the number of samples and output vector dimension, and it is possible to omit the factor during optimization. Assume that for each measurement the variance  $\sigma_{y,k}^2$  can be calculated. This variance can be used to normalize the cost function (2), giving the Weighted Least Squares [4] [19] [34] or Bayes [4] cost function

$$C_{WLS} \triangleq \frac{1}{N} \sum_{k=1}^N \left\{ \frac{1}{n} \sum_{i=1}^n \left[ \frac{(f_i(\mathbf{u}_k, \theta) - y_{k,i})^2}{\sigma_{y,k,i}^2} \right] \right\}. \quad (3)$$

A larger variance leads to a smaller influence of that measurement in the cost function, leading to a smaller model uncertainty. In practise the proper choice of the cost function is often neglected and a simple LS cost function is selected, because 'everyone does it like that'. The use of the WLS cost function already results in a smaller model uncertainty and should be preferred over the LS cost whenever output variances are available. None of these two costs, however, deals with noisy input samples.

Noisy input measurements can partly be dealt with, based on the Errors-In-Variables (EIV) cost function. The EIV cost function [31] [34] is defined as

$$C_{EIV} \triangleq \frac{1}{N} \sum_{k=1}^N \left\{ \frac{1}{n} \sum_{i=1}^n \left[ \frac{(f_i(\hat{\mathbf{u}}_k, \theta) - y_{k,i})^2}{\sigma_{k,i,y}^2} \right] + \frac{1}{m} \sum_{i=1}^m \left[ \frac{(\hat{u}_{k,i} - u_{k,i})^2}{\sigma_{k,i,u}^2} \right] \right\}. \quad (4)$$

with  $\hat{\mathbf{u}}_k$  the estimated values for the true inputs and  $m$  the length of the input vector. Note that the cost function must be minimized for both the BB parameters  $\theta$  and the estimated inputs  $\hat{\mathbf{u}}_k$ , thus enlarging the parameter space with  $m$  additional parameters. It has been used for linear systems, but was also applied on Neural Nets by Van Gorp et al. [33].

It should be stressed that the only way to implement stochastic properties in the model, is by choosing the proper cost

function. Neither the chosen model, nor the optimization method deal with the problem of noisy measurements. Choosing the wrong cost function inevitably leads to biasing and thus to faulty model parameters.

### C. Parameter initialization

The fastest way to find initial model parameters is by simply picking random values. The majority of the models are trained using such random starting parameters. For nonlinear problems this means that the optimization only guarantees that a local minimum of the cost function is reached. The best way of initialization is by starting close to the optimal solution, using a 'good guess'. In the field of FL the parameters are usually defined by linguistic rules, Fuzzy Patches [13] [14] or clustering methods [36]. Many times it is believed that these methods for initialization are sufficient to guarantee a satisfying performance, and further optimization of the Fuzzy Sets is omitted.

Another technique is the use of a simpler model or less demanding cost function prior to the demanded modelling. E.g. when using the Errors-In-Variables cost function, it is observed that the optimization suffers from many local minima [25] [31] [34] such that the initial parameters are best chosen as the results of an Output Error minimization, using the LS cost function. A much used technique is learning a linear model (with a global minimum for the model parameters) and then use the linear parameters as a starting point for a nonlinear model.

### D. Choose the optimization method

Although the name of the optimization scheme within different modelling fields is often chosen differently, the idea is usually the same: change the model parameters such that the resulting model performs better (such that the cost function is minimal). The optimization is done by recurringly updating the model parameters  $\theta(t+1) = \theta(t) + \Delta\theta(t)$ . Various methods exist for the calculation of the update vector  $\Delta\theta(t)$ . The choice is based on robustness of the optimization method, the memory needs, speed of optimization and the ability to avoid local minima. Nearly all current methods make, in one way or another, use of the Jacobian matrix  $\mathbf{J}(t) \in \mathbb{R}^{N \times P}$  with elements  $J_{i,j}(t)$  that are calculated as  $J_{i,j,k}(t) = \partial e_{i,k}(t) / (\partial \theta_j)$ . The index  $j$  equals  $j = 1, 2, \dots, P$  with  $P$  the number of model parameters that must be optimized. The error  $e_i$  depends on the chosen cost function. In the case that the LS cost function is used, the error would be  $e_{i,k}(t) = f_i(\mathbf{u}_k, \theta(t)) - y_{k,i}$  with  $i = 1, 2, \dots, N$ .

It would go beyond the scope of this paper to describe all optimization techniques. The most common optimization steps are the gradient descent (or Gaussian) update vector [26], defined as  $\Delta\theta(t) = -\mathbf{J}^T \mathbf{e}$  with the error  $\mathbf{e}$  a column vector that equals  $\mathbf{e} = [e_1, e_2, \dots, e_N]^T$ . The Newton step is calculated as  $\Delta\theta = -\mathbf{H} \backslash \mathbf{J}^T \mathbf{e}$  with  $\mathbf{H}$  the Hessian matrix with elements  $h_{i,j} = \partial^2 f_{BB} / \partial \theta_i \partial \theta_j$ . In practise the calculation of the full Hessian  $\mathbf{H}$  is often computational prohibitive. As a solution the off-diagonal parts of the Hessian are neglected, and further it is assumed that  $\mathbf{H} \cong \mathbf{J}^T \mathbf{J}$ . The resulting Gauss-Newton optimization step  $\Delta\theta$  is then calculated as  $\Delta\theta = -(\mathbf{J}^T \mathbf{J}) \backslash \mathbf{J}^T \mathbf{e}$ .

Even the use of  $\mathbf{J}^T \mathbf{J}$  instead of the Hessian, is computational demanding, and this is certainly the case for the inversion of the matrix. In case that the Hessian is not positive definite, Newton and Gauss-Newton methods are known to be easily trapped into a saddle-point, or even at the maximum of a functional rather than the minimum. The method can become unstable far more easier than the simple Steepest Descent algorithm. In order to make the Hessian matrix in the Gauss-Newton method positive definite, an identity matrix times a constant factor  $\lambda$  is added to it. The resulting Levenberg-Marquardt (LM) optimization step is then calculated as  $\Delta\theta = -(\mathbf{J}^T \mathbf{J} + \lambda \mathbf{I}) \backslash \mathbf{J}^T \mathbf{e}$  with  $\lambda$  the Marquardt-factor which is made adaptive. The need for inversion of the new Hessian matrix  $(\mathbf{J}^T \mathbf{J} + \lambda \mathbf{I})$  makes that LM needs a lot of calculation power and memory. Yet, the method is very robust against local minima.

The list given above is not exhaustive in the least. The use of the Metropolis algorithm, simulated annealing, Random Walk, error surfaces, Conjugate Gradients, Scaled Conjugate Gradients [4], more complex Gradient Descent methods [23], Boltzmann Learning [9], the Delta Rule [3], Line Search [9] [21], Linear Quadratic Programming, or even the use of Genetic Algorithms or Reinforcement Learning have not been addressed in detail in this section.

### E. Choose the stopping criterion

The decision when to stop the optimization can be taken apart from the optimization routine itself. In general one can state that the optimization is stopped when the cost function reaches its minimum.

Within BB modelling there is a severe risk for overfitting, due to severe overparametrizing. Therefore, a way must be found to handle overfitting. There are a number of methods described in the literature. For example model selection methods, which are hardly possible for BB models, and regularization methods. The most used method is the use of a validation set, which is also called early stopping [2] [20] or stopped training [24]. The measurement data is split into

three parts: the learning set, the validation set and the test set. The optimization of the BB parameters is performed on the learning set, while monitoring the validation set. Optimization is stopped when the error on the validation set increases. This technique is well known in the field of NN, but as shown in Fig. 1 it is also possible to use early stopping for other models.

The drawback of early stopping is the need for a validation set. In the particular case of sparse data, this can lead to a lack of measurement data for optimizing. Sometimes the validation set is just a single sample (leave-one-out technique) and a large number of models are mapped, with a different validation sample for each model. The use of the leave-one-out technique is highly questionable when used with noisy data. In his excellent paper Amari [2] shows that, under mild conditions, the size of the validation set should be no larger than  $M = (1/\sqrt{2N}) \cdot 100\%$  with  $M$  the size of the validation set in percent and  $N$  the size of the whole measurement set.

#### F. Model validation

Despite of the measurements made, and hours of optimization, no guarantee can be given that the resulting model complies with the users' needs. Therefore the model must be evaluated. This is usually done by evaluating the model on a test set.

A second step in the evaluation could be an analysis of the stability of the model. A separation must be made between absolute and relative stability. The latter is based upon phase or amplitude margins and is studied in detail for linear systems. To date, only absolute stability is available for nonlinear systems. Most stability theorems for nonlinear systems make use of a Lyapunov function [9] and expect the nonlinear function to be sector bounded [29]. The resulting stability theorem is usually too conservative for practical use but better theorems are yet to be found. The stability of NN is studied in [30]. FL systems are studied in [5] [12] and [14].

### III. CONCLUSION

This paper introduced a general identification scheme for system modelling, clearly separating the model, cost function and optimization scheme. The reference list shows how current linear and nonlinear modelling techniques easily fit into this modelling scheme. Although Neural Networks and Fuzzy Logic originate from the Artificial Intelligence domain, they also fit well in the identification scheme on the model level. This paper contributes to the demystification of both modelling techniques. The gain for both NN and FL is that they can profit from present knowledge within nonlinear system identification, such as cost function properties and optimization schemes. The gain for e.g. linear modelling is that both FL and NN can be used as nonlinear extensions to linear theory.

### IV. ACKNOWLEDGMENTS

This paper presents research results of the Belgian Programme on Interuniversity Poles of Attraction (IUAP 4/2), initiated by the Belgian State, Prime Minister's Office for Science, Technology and Culture; and the Flemish Government (GOA, IMMI2). The scientific responsibility rests with its authors.

### V. REFERENCES

- [1] Abramovitch D. Y., Bushnell L. G., "Report on the Fuzzy Versus Conventional Control Debate", IEEE Control Systems, June 1999, pp. 88 - 91.
- [2] Amari S.-I., Murata N., Müller K.-R., Finke M., Yang H. H., "Asymptotic Statistical Theory of Overtraining and Cross-Validation", IEEE Transactions on Neural Networks, Vol. 8, No. 5, September 1997, pp. 985 - 998.
- [3] Antognetti P., Milutinovic V. (Eds.), "Neural Networks, Concepts, Applications and Implementations", Prentice Hall Advanced Reference Series on Engineering, 1991.
- [4] Bishop C. M., "Neural Networks for Pattern Recognition", Clarendon Press, Oxford, 1995.
- [5] Cao S. G., Rees N. W., Feng G., "Stability Analysis of Fuzzy Control Systems", IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, Vol. 26, No. 1, February 1996, pp. 201 - 204.
- [6] Chen T., Chen H., "Approximation Capability to Functions of Several Variables, Nonlinear Functionals, and Operators by Radial Basis Function Neural Networks", IEEE Transactions on Neural Networks, Vol. 6, No. 4, July 1995, pp. 904 - 910.
- [7] Forssell U., Lindskog P., "Combining Semi-physical and Neural Network Modeling: an Example of its Usefulness", SYSID, July 1997, Kitakyushu, Japan, pp. 795 - 798.
- [8] Grossberg S., "Nonlinear Neural Networks: Principles, Mechanisms, and Architectures", Neural Networks, Vol. 1, 1988, pp. 17 - 61.
- [9] Haykin S., "Neural Networks, a Comprehensive Foundation", Prentice Hall, 1999.
- [10] Hindi H., Boyd S., "Analysis of Linear Systems with Saturation Using Convex Optimization", Proceedings of the 37th IEEE Conference on Decision & Control, Tampa, Florida, USA, December 1998, pp. 903 - 908.

- [11] Hornik K., "Approximation Capabilities of Multilayer Feedforward Networks", *Neural Networks*, Vol. 4, 1991, pp. 251 - 257.
- [12] Johansen T. A., "Fuzzy Model Based Control: Stability, Robustness, and Performance Issues", *IEEE Transactions on Fuzzy Systems*, Vol. 2, No. 3, August 1994, pp. 221 - 234.
- [13] Kosko B., "Fuzzy Systems as Universal Approximators", *IEEE Transactions on Computers*, Vol. 43, No. 11, November 1994, pp. 1329 - 1333.
- [14] Kosko B., "Global Stability of Generalized Additive Fuzzy Systems", *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, Vol. 28, No. 3, August 1998, pp. 441 - 452.
- [15] Krzyzak A., Linder T., Lugosi G., "Nonparametric Estimation and Classification Using Radial Basis Function Nets and Empirical Risk Minimization", *IEEE Trans. on Neural Networks*, Vol. 7, No. 2, March 1996, pp. 475 - 487.
- [16] Lewis F. L., Liu K., "Towards a Paradigm for Fuzzy Logic Control", *Automatica*, Vol. 32, No. 2, 1996, pp. 167 - 181.
- [17] Ljung L., "System Identification, Theory for the User", Prentice-Hall information and system sciences series, 1994.
- [18] Ljung L., "Some Aspects on Nonlinear Black-box Modeling in System Identification", Internal report, Dept. of Electrical Engineering, Linköping University, Sweden.
- [19] MacKay D. J. C., "Bayesian Methods for Adaptive Models", PhD thesis, California Institute of Technology, Pasadena, California, 1992.
- [20] MacKay D. J. C., "Hyperparameters: Optimize, or Integrate Out?", *Maximum Entropy and Bayesian Methods*, Kluwer Dordrecht, 1993.
- [21] Møller M. F., "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning", *Neural Networks*, Vol. 6, 1993, pp. 525 - 533.
- [22] Passino K. M., Yurkovich S., "Fuzzy Control", Addison-Wesley, 1997.
- [23] Pearlmutter B. A., "Gradient Calculations for Dynamic Recurrent Neural Networks: a Survey", *IEEE Transactions on Neural Networks*, Vol. 6, No. 5, September 1995, pp. 1212 - 1226.
- [24] Sarle W. S., "Stopped Training and Other Remedies for Overfitting", *Proceedings of the 27th Symposium on the Interface*, 1995, pp. 1 - 10.
- [25] Schoukens J., Pintelon R., Rolain Y., "Maximum Likelihood Estimation of Errors-In-Variables Models Using a Sample Covariance Matrix Obtained from Small Data Sets", *Recent Advances in Total Least Squares Techniques and Errors-In-Variables Modeling*, 1997, pp. 59 - 68.
- [26] Sjöberg J., "Non-Linear System Identification with Neural Networks", Linköping Studies in Science and Technology Dissertation No. 381, Sweden, 1995.
- [27] Sjöberg J., Zhang Qinghua, Ljung Lennart, Benveniste A., Delyon B., Glorennec P.-Y., Hjalmarsson H., Juditsky A., "Nonlinear Black-box Modeling in System Identification: a Unified Overview", *Automatica*, Vol. 31, No. 12, 1995, pp. 1691 - 1724.
- [28] Suykens J. A. K., Bersini H., "Neural Control Theory: an Overview", *Journal A*, Vol. 37, No. 3, 1996, pp. 4 - 10.
- [29] Suykens J. A. K., De Moor B. L. R., Vandewalle J. P. L., "Artificial Neural Networks for Modeling and Control of Non-Linear Systems", Kluwer Academic Publishers, 1996.
- [30] Tanaka K., "An Approach to Stability Criteria of Neural-Network Control Systems", *IEEE Transactions on Neural Networks*, Vol. 7, No. 3, May 1996, pp. 629 - 642.
- [31] Vandersteen G., "Identification of Linear and Nonlinear Systems in an Errors-In-Variables Least Squares and Total Least Squares Framework", PhD thesis, Vrije Universiteit Brussel, Belgium, April 1997.
- [32] Van Gorp J., "Steel Plant Modeling and Analysis Using a Neural Network", *ANNIE 1999, Intelligent Engineering Systems Through Artificial Neural Networks*, Volume 9, pp. 739 - 744.
- [33] Van Gorp J., Schoukens J., Pintelon R., "Adding Input Noise to Increase the Generalization of Neural Networks is a Bad Idea", *ANNIE 1998, Intelligent Engineering Systems Through Artificial Neural Networks*, Volume 8, pp. 127 - 132.
- [34] Van Gorp J., Schoukens J., Pintelon R., "The Errors-In-Variables Cost Function for Learning Neural Networks with Noisy Inputs", *ANNIE 1998, Intelligent Engineering Systems Through Artificial Neural Networks*, Volume 8, pp. 141 - 146.
- [35] Van Gorp J., Suykens J. A. K., "Representation of SISO Fuzzy Logic Control Systems Within NLq Theory", *ANNIE 1999, Intelligent Engineering Systems Through Artificial Neural Networks*, Volume 9, pp. 603 - 609.
- [36] Wang L., Langari R., "Complex Systems Modeling via Fuzzy Logic", *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, Vol. 26, No. 1, February 1996, pp. 100 - 106.
- [37] Werbos P. J., "Neurocontrol and Elastic Fuzzy Logic: Capabilities, Concepts, and Applications", *IEEE Transactions on Industrial Electronics*, Vol. 40, No. 2, April 1993, pp. 170 - 180.
- [38] Williamson R. C., Helmke U., "Existence and Uniqueness Results for Neural Network Approximations", *IEEE Transactions on Neural Networks*, Vol. 6, No. 1, January 1995, pp. 2 - 13.
- [39] Zadeh L. A., "Fuzzy Sets and Applications", John Wiley & Sons, 1987.