Memory Economy for Electronic Control Units: Compression of Conventional Look-up Tables

K. KNÖDLER, J. POLAND, A. MITTERER* and A. ZELL
WSI-RA Universität Tübingen
Sand 1, D - 72076 Tübingen
GERMANY
knoedler@informatik.uni-tuebingen.de http://www-ra.informatik.uni-tuebingen.de

Abstract: - The operation of modern technical systems is often controlled by electronic units. In the automotive sector lattice-like look-up tables are a common representation of non-linear multi-dimensional parameter functions of actuators. Regardless of the strongly restricted memory capacity of automotive electronic control units, manufacturers need to introduce additional functionalities to meet the new specifications. With an increasing number of adjustable actuators the number of parameters and thus the memory requirement within the control unit grows enormously. This work introduces a method based on evolutionary and classical optimization algorithms which reduces the memory requirement for 2-dimensional look-up tables.

Key-Words: - Automotive Electronic Control Unit, Memory Capacity, Look-up Table, nonlinear Optimization, Genetic Algorithm, Evolution Strategy.

1 Introduction

Modern technical systems are often controlled by individual electronic control units (ECU). In modern automotive vehicles there are more than 50 ECUs to control systems like the combustion engine, active and passive safety systems, etc. For the actual *base state*[1] of the system the ECU micro controller calculates the required parameter settings for all adjustable actuators and functions. Over the last years legal restrictions on economy and safety together with customer requests for performance have forced the manufacturers to introduce additional functionalities. Thereby advanced control systems require exponentially increasing numbers of parameters controlling the actuators (see e.g. [10] for the development of modern automotive combustion engines). Although electronic control units are equipped with high-performance micro controllers, there are strong restrictions on the complexity of control strategies. In the automotive sector, e.g. for ECUs of combustion engines, this is particularly due to relatively low computational power ($1 - 20$ MHz) and small memory capacity (~ 1 MByte). The latter is justified by very high development costs that are caused by hard robustness requirements, relatively small production quantities, and by the demand for long term availability. Therefore it is not possible to increase the capacity of memory chips for ECUs as much as necessary. Already existing and optimized data fields approximating parameter functions of actuators have to be modified to save memory.

2 Look-up Table Approximation

A common representation of a parameter function is a lattice-like data field with a set of parameter values at a full factorial rectangular grid of base states. This data field is often called *look-up table* [6]. For the control of high performance combustion engines, up to 800 1- and 2-dimensional look-up tables are necessary. The compression method is formalized for 2-dimensional look-up tables, but also applicable to other dimensionalities.

Let $\mathbb{T}$ define the set of 2-dimensional look-up tables, i.e. with 2 input variables and one output variable. For the following considerations input and output variables are assumed to be scaled to $[0, 1]$. An element $T \in \mathbb{T}$, $T := (t, G^t, P^t)$ is given

*A. Mitterer, BMW Group, D-80788 München, Germany. E-mail: alexander.mitterer@bmw.de.

[1]Here a *base state* is defined by a certain number of *base parameters* which are filtered and digitalized signals measured by certain sensors. It's also called *operating point*. The parameters completing the description of the total state of the system depend on these base parameters.

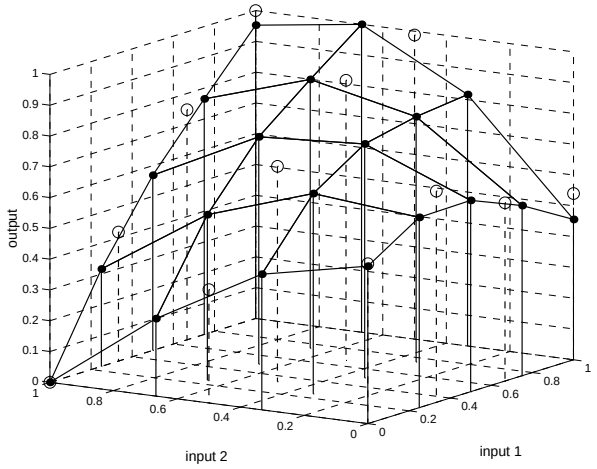

Fig. 1. A (5×4) look-up table (filled circles) with two inputs (base parameters) is compressed to size (4×3) (circles).

by its grid size $t := (t_1 \times t_2)$, the grid positions $G^t := \{G_1^t, G_2^t\}$, and the parameter values $P^t(G^t)$ at these positions. Figure 1 visualizes a look-up table with grid size $t = (5 \times 4)$, as an elementary example. For a new pair of input variables (a base state) $X := (X_1, X_2)$ with $G_{1m}^t \leq X_1 < G_{1(m+1)}^t$ and $G_{2n}^t \leq X_2 < G_{2(n+1)}^t$, the ECU micro controller calculates the demanded output $P(X)$ by a bi-linear interpolation, i.e. a weighted sum of the surrounding parameter values $P_{ij}^t := P^t(G_{1i}^t, G_{2j}^t)$:

$$\begin{aligned} P(X) = \quad & \hat{A}_{00} \cdot P_{m,n}^t + \hat{A}_{10} \cdot P_{m,n+1}^t \\ + \quad & \hat{A}_{01} \cdot P_{m+1,n}^t + \hat{A}_{11} \cdot P_{m+1,n+1}^t. \end{aligned}$$

The terms $\hat{A}_{ij} := A_{ij}/A$ correspond to the four areas visualized in figure 2 devided by the total area $A := A_{00} + A_{01} + A_{10} + A_{11}$.

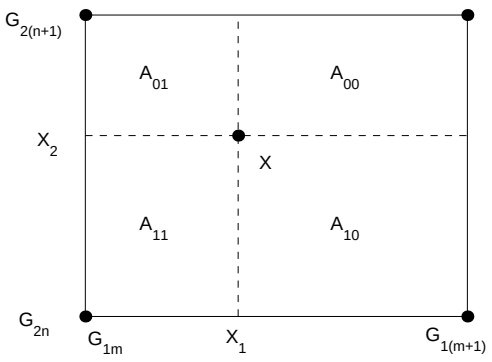

Fig. 2. Parameters for the bi-linear interpolation of $P(X)$ from a given look-up table $T := (t, G^t, P^t)$.

Recent studies of more flexible grid forms like the *associative datafield* with *scattered data* suggested in [14] might be better for the efficient approximation of functions with more than 2 input variables, but at least in two dimensions they often require more memory which is the critical point. The benefit of associated datafields concerning the online adaptability is ignored here, since associative datafields could not establish till today, i.e. conventional lattice-like look-up tables are still dominant especially in the automotive sector.

3 Neural Network Approximation

An alternative way to calculate required parameters for actuators is to evaluate an artificial neural network, e.g. a Multi-Layer Perceptron or a Radial Basis Function network, within the ECU ([8], [13], [11]). In this case an ECU micro controller calculates for every new base state X the demanded output $P(X)$ by evaluating the neural network: $P(X) = net(X)$. Look-up tables are then replaced by parameters of a neural network, e.g. weights and biases, which are determined in an offline or sometimes in a continuous online training process. The latter is only possible for very fast learning algorithms. Today, neural networks are still very rarely used for the control of automotive systems.

Later a rough comparison of the memory requirements for the parameters of a Multi-Layer Perceptron neural network on one side and for look-up tables on the other side will be shown.

4 Look-up Table Compression

This work concentrates on the number of grid points defining the size of a look-up table. Figure 1 visualizes the structure of a very small look-up table for an actuator's parameter function depending on two base parameters. For demonstration reasons, the grid size was chosen (5×4), the grid point positions are highlighted by the solid vertical lines. The filled circles mark the parameter values. The goal is to calculate a compressed look-up table that is sufficiently similar to the original one. The dashed lines and the circles in figure 1 indicate a possible new compressed look-up table of size (4×3). In the following sections the problem is formalized and the compression method is outlined. Consider the following problem: Let $U \in \mathbb{T}$ with $U = (u, G^u, P^u)$ be an original uncompressed look-up table defined at a grid G^u of

size $u = (u_1 \times u_2)$ with a set of $u_1 \cdot u_2$ parameter values P^u. How can a compressed look-up table $C \in \mathbb{T}$ with $C = (c, G^c, P^c)$ be constructed that approximates the original one sufficiently accurately? Assume a predetermined reduced grid size $c = (c_1 \times c_2)$ where at least one of c_1 and c_2 is smaller than the corresponding original value. Figure 1 displays a simple situation, where an original look-up table of size $u = (5 \times 4)$ has to be compressed to one with size $c = (4 \times 3)$ indicated by the dashed lines. Thereby the positions of the grid points defining the edge of the base range for the original and the compressed look-up table coincide. The positions of the other grid points fixing the interior sampling points are freely placeable for each dimension. In order to achieve a maximal similarity between the original look-up table and the compressed one, two entangled minimization tasks have to be performed. The following section introduces a suitable objective function for the optimization algorithms.

4.1 The Objective Function

An original uncompressed look-up table U is defined at the grid G^u of size $u = (u_1 \times u_2)$. There are $i = 1, \ldots, u_1$ grid point positions G^u_{1i} in the first dimension and $j = 1, \ldots, u_2$ positions G^u_{2j} in the second dimension. The corresponding parameter values are $P^u_{ij} := P^u(G^u_{1i}, G^u_{2j})$. The compressed look-up table C is defined at the grid G^c with $i = 1, \ldots, c_1$ positions G^c_{1i} and $j = 1, \ldots, c_2$ positions G^c_{2j} in the first and second dimension. The parameter values are $P^c_{ij} := P^c(G^c_{1i}, G^c_{2j})$, where the grid points are partly determined by the four relations $G^c_{11} = G^u_{11} = 0$, $G^c_{1c_1} = G^u_{1u_1} = 1$, $G^c_{21} = G^u_{21} = 0$, and $G^c_{2c_1} = G^u_{2u_1} = 1$ to fix the edges of the base range. After every new setting of the grid positions G^c the calculation of new parameter values P^c at this grid is required. In order to assess an objective function for the optimization problem, the function

$$\Phi(G^c, P^c) := \sum_{i,j=1}^{u_1,u_2} \left(\tilde{P}^u_{ij}(G^c, P^c) - P^u_{ij} \right)^2, \qquad (1)$$

is defined, where $\tilde{P}^u(G^c, P^c) := B(P^c(G^c), G^u)$ defines the bi-linear interpolated parameter values at the original grid G^u. A *Levenberg-Marquardt Algorithm* (see e.g. [3]) is used to calculate the set of parameter values $\tilde{P}^c$ which leads to the best least square curve fit $\tilde{P}^u_{opt}(G^c, P^c)$ of the parameter values P^u at the original grid G^u. The parameter values $\tilde{P}^c$ are then defined by the minimum of equation (1) with respect to P^c

$$\Phi(G^c, \tilde{P}^c) := \min_{P^c} \Phi(G^c, P^c). \qquad (2)$$

The *Levenberg-Marquardt Algorithm* uses the parameter values P^c_{ini} calculated by a simple interpolation between the original parameter values P^u that is based on triangulation as starting point for the least square curve fit.

Using equation (2), an objective function for the problem of optimal positioning new grid points G^c can now be defined by

$$\Phi(G^c) := \Phi(G^c, \tilde{P}^c). \qquad (3)$$

This function is minimized by means of evolutionary algorithms, i.e. evolution strategies and genetic algorithms (see [4], [12], [1]). For comparison reasons also classical optimization algorithms for constrained and unconstrained problems are applied (see e.g. [3]).

4.2 Compression Algorithms

Since the points defining the grid edge are already fixed to 0 and 1 respectively, the dimension of the optimization problem is reduced to $c_1 + c_2 - 4$. In the sequel different algorithms that are used to solve the problem of positioning the grid points G^c of the compressed look-up table C are described. All algorithms need an initial set of solutions $\{G^c_{ini}\}$ of the problem in order to start the optimization. For this purpose equidistant grids G^c on the base range are slightly distorted by shifting all points that lie on the same grid line by a small random number in the orthogonal direction (see figure 3).

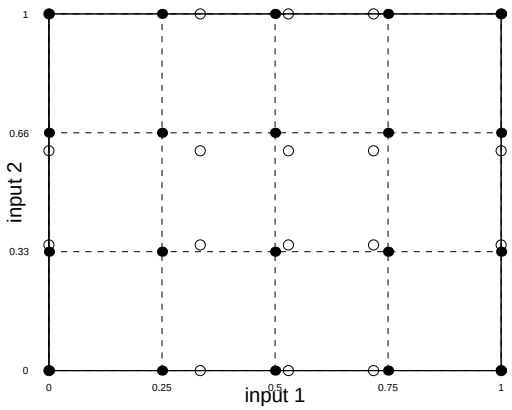

Fig. 3. A noisy rectangular grid (circles) derived from an equidistant grid of size (5×4) (filled circles).

4.2.1 Evolutionary Algorithms.

The problem of finding optimal new grid point positions $\tilde{G}^c$ is now defined by the minimum of equation (3) with respect to G^c

$$\Phi(\tilde{G}^c) := \min_{G^c} \Phi(G^c), \tag{4}$$

and can be solved by means of evolutionary algorithms. Of course evolution strategies suit well since the problem is continuous. An evolution strategy with covariance matrix adaptation [5] is used. The parameters are set to the standard values suggested in this work. For comparison reasons also a standard evolution strategy without covariance matrix adaptation is applied. A parent population with $\mu = 6$ individuals and an offspring population with $\lambda = 12$ individuals is chosen.

In order to apply genetic algorithms, a coding function with application specific bit resolution is introduced. Here, the grid point positions are encoded in gray code with a precision of 12 bits. An individual has a 2-dimensional matrix structure with $c_1 + c_2 - 4$ rows and 12 columns. A parent population of size $\mu = 40$ individuals and an offspring population with $\lambda = 40$ individuals is used. The algorithm parameters were set to an elite of 4 individuals, a tournament selection with the best $q = 4$ individuals, and either uniform or 2-point crossover. Uniform crossover on 2-dimensional encoded individuals is equivalent to uniform crossover for the 1-dimensional case. The 2-point crossover uses 2 crossover lines in each dimension and therefore separates the parents in up to 9 exchangeable sections.

4.2.2 Classical Optimization Algorithms.

Of course also classical optimization algorithms can be used to minimize the nonlinear multivariable function $\Phi(G^c)$ defined in equation (3). For example a *Sequential Quadratic Programming* algorithm (see e.g. [3]) is capable of handling the constrained optimization problem of positioning the grid points. The transformation of the search space from $[0, 1]^2$ to $]-\infty, \infty[^2$ by means of

$$\hat{G}^{(c)} = \tan((2 \cdot G^c - 1) \cdot \pi/2),$$

allows to apply optimization algorithms for unconstrained problems. Here, both a *Simplex Search Method* based on the one suggested by *Nelder* and *Mead* and a *BFGS Quasi Newton Method* (see e.g. [3] for this topic) are used.

5 An Application Example

As application example, the ignition timing angle as a control parameter for a modern combustion engine is considered. It depends on the base parameters, engine speed and relative air mass flow which together define the base state of the engine. An ignition timing angle look-up table of grid size $u = (24 \times 16)$ is compressed to size $c = (16 \times 10)$, i.e. 186 instead of 424 parameters. The points defining this look-up table are plotted in the upper part of figure 4. The lower part of figure 4 displays this look-up table calculated at a fine grid of size (101×103) by bi-linear interpolation (including the original grid positions).

All evolutionary and classical algorithms are implemented in the MATLAB environment. The linear interpolation based on triangulation and the *Levenberg-Marquardt Algorithm* for the least square curve fit are included in the *Matlab Optimization Toolbox* ([7]). The classical algorithms used for the positioning of the grid points (*Sequential Quadratic Programming*, *Simplex Search*, *BFGS Quasi Newton*) are also implemented in this toolbox. For the following results 30 runs of each algorithm were performed. The maximal number of generations, where the evolutionary algorithms converged, were 100 generations. In the case of the classical algorithms, the internal termination criteria were used.

5.1 Result: Compressed Look-up Table

Figure 6 shows the performance measured by equation (4) of the different algorithms for the compression of the ignition timing angle look-up table to size $c = (16 \times 10)$. All evolutionary algorithms find very good solutions. Note that the overall best result 0.00121 was found by the genetic algorithm with 2-point crossover. The best result for the CMA evolution strategy was 0.00125 and for the standard evolution strategy 0.00148. There are more significant differences in the mean and maximum values. Here, the genetic algorithms perform more robust than the evolution strategies. The classical algorithms yield significantly worse results (0.00189), especially the mean and maximum values are bad. The upper plot in figure 5 shows the overall best compressed look-up table, the lower plot the corresponding fine evaluation. The genetic algorithm requires much more computation time because of the high number of function evaluations, i.e. 4000. The CMA and the standard

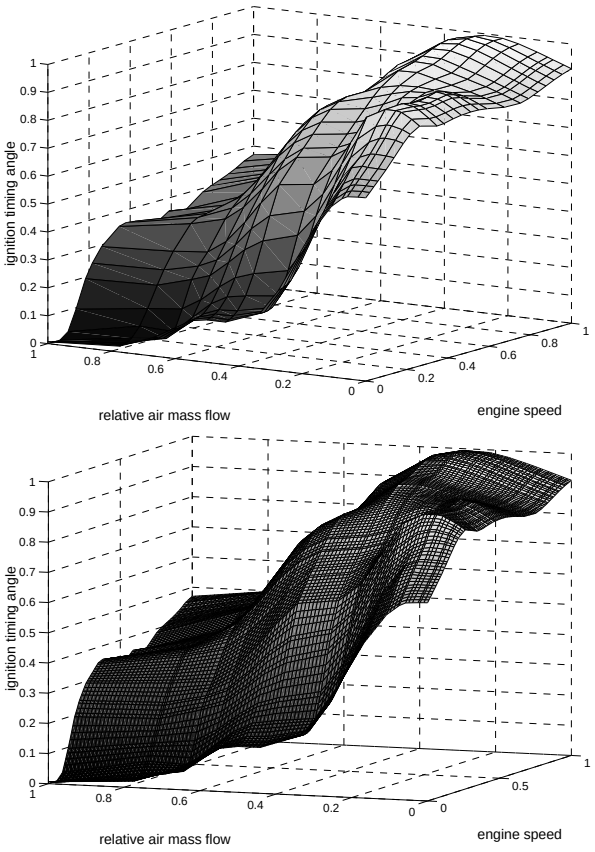

Fig. 4. Top: An ignition timing angle look-up table of grid size $u = (24 \times 16)$. Bottom: The look-up table evaluated at a fine grid of size (101×103).

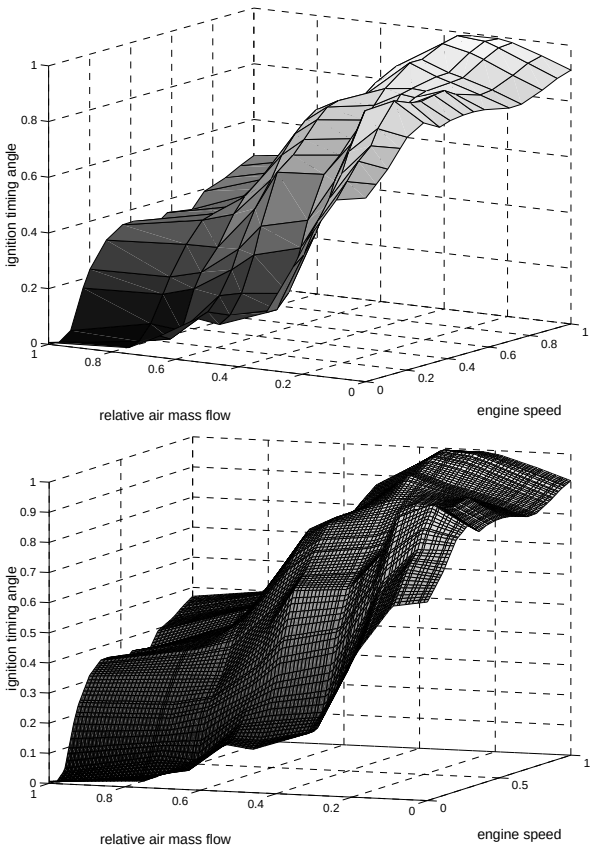

Fig. 5. Top: The best compressed look-up table for grid size $c = (16 \times 10)$. Bottom: The compressed look-up table evaluated at the fine grid.

evolution strategy took 1200, and in the mean, the classical optimization algorithms took 1400 function evaluations. But since the calibration of many technical systems requires best parameters for the actuators' control, computation time is less important and therefore also genetic algorithms are useable in practice.

5.2 Result: Neural Network

This section gives a rough comparison of the memory requirement for a compressed look-up table on one side and for parameters of a neural network approximation on the other side. For this purpose the ignition timing angle look-up table with grid size $u = (24 \times 16)$ is chosen. The number of parameters for the compressed look-up table with grid size $c = (16 \times 10)$, i.e. 186, and the number of parameters for a neural network that leads to a comparable quality at the original grid are considered. A feedforward neural network with 2 input neurons for the base parameters, 2 hidden layers consisting of 8 tanh neurons each, and with 1 linear output neuron for the actuator parameter is used. It takes 105 parameters, i.e. weights and biases, which have to be stored in the ECU. As training data set the original look-up table U is used. The *Levenberg-Marquardt* learning algorithm with bayesian regularization ([9], Neural Network Toolbox [2]) leads to a significantly worse sum of square errors of the original parameter values P^u than the best result of the presented method, i.e. 0.0042 instead of 0.0012. But the comparison of the parameter values at a fine grid with size $u = (101 \times 103)$ (including the original positions) shows that the neural approximation yields a better result, i.e. 0.30605 instead of 0.33912 for the sum of square errors. Therefore the neural network approximation of the look-up table yields a better global match of the original look-up table than the compressed look-up table.

6 Conclusions

The memory capacity of electronic control units can not stand the growing number of additional parameters, that have to be introduced in order

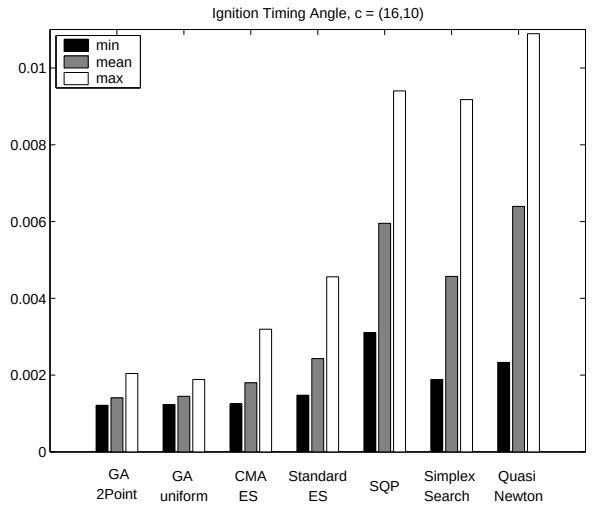

Fig. 6. The algorithms' performance for the compression of the ignition timing angle look-up table.

to fulfill new technical specifications. To provide memory for new parameters, a method for the optimal compression of already existing look-up tables was deduced. In the application example the evolutionary algorithms yield better and more robust results than the classical optimization algorithms. For higher dimensional search spaces, the evolutinary results were significantly better. Although the problem of compressing look-up tables is a continuous one, for larger grid sizes c in the mean genetic algorithms perform better than evolution strategies. The alternative method using artificial neural networks takes less memory capacity in order to reach a better global approximation than the optimized compressed look-up table.

Neural network approximations of look-up tables are very smooth and therefore optimal for the actuators' control. The missing physical relationsip between the network parameters and the actuator functions, and the low degree of transparency of neural networks will retain the dominance of conventional look-up tables at least for the next years.

Acknowledgments

We thank Thomas Fleischhauer and Frank Zuber-Goos for helpful discussions. This research has been supported by the BMBF (grant no. 01 IB 805 A/1).

References:

[1] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, 1996.

[2] H. Demuth and M. Beale. *Neural Network Toolbox User's Guide, Sixth Printing Revised for Version 4*. The Math Works Inc., 3 Apple Hill Drive, Natick, MA 01760-2098, 2000.

[3] R. Fletcher. *Practical Methods of Optimization*. Chichester, Wiley & Sons, 2nd edition, 1991.

[4] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[5] N. Hansen and A. Ostermeier. Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: The $(\mu/\mu_i, \lambda)$-cma-es. In *5th European Congress on Intelligent Techniques and Soft Computing*, pages 650–654, 1997.

[6] Robert Bosch GmbH (Hrsg.). *Kraftfahrzeugtechnisches Taschenbuch*. Springer-Verlag, New York, 22. auflage edition, 1992.

[7] The Math Works Inc. *Optimization Toolbox User's Guide, Fourth Printing for Version 2.1*. The Math Works Inc., 3 Apple Hill Drive, Natick, MA 01760-2098, 2000.

[8] U. Lenz and D. Schröder. Artificial intelligence for combustion engine control. In *SAE International Congress and Exposition, Detroit, USA, SAE Technical Paper No. 960328*, 1996.

[9] D. J. C. McKay. Bayesian interpolation. *Neural Computation*, 4:415–447, 1992.

[10] A. Mitterer. *Optimierung vielparametriger Systeme in der Antriebsentwicklung, Statistische Versuchsplanung und Künstliche Neuronale Netze in der Steuergeräteauslegung zur Motorabstimmung*. Number 434 in 12. VDI, VDI Verlag GmbH Düsseldorf, 2000. VDI Fortschritt-Berichte – Reihe 12: Verkehrstechnik/Fahrzeugtechnik.

[11] O. Nelles and R. Isermann. Identification of nonlinear dynamic systems – classical methods versus radial basis function networks. In *American Control Conference (ACC)*, 1995.

[12] I. Rechenberg. *Evolutionsstrategie '94*. frommann-holzboog, Stuttgart, 1994.

[13] C. Schäffner, D. Schröder, and U. Lenz. Application of neural networks to motor control. In *International Power Electronics Conference, IPEC '95, Yokohama, Japan. Proceedings pp. 46-51, Vol. 1*, 1995.

[14] M. Schmitt. Associative datafields in automotive control. In *3rd IEEE Conference on Control Applications, Glasgow (UK)*, pages 1239–1244, 1994.