(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) **EP 1 972 767 A1**

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
24.09.2008 Bulletin 2008/39

(51) Int Cl.:
F02D 41/14 *(2006.01)* ***F02D 41/24*** *(2006.01)*

(21) Application number: **07104811.0**

(22) Date of filing: **23.03.2007**

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IS IT LI LT LU LV MC MT NL PL PT RO SE SI SK TR
Designated Extension States:
AL BA HR MK RS

(71) Applicant: **Ford Global Technologies, LLC**
Dearborn, MI 48126 (US)
Designated Contracting States:
DE GB SE

(72) Inventor: **Larsson, Erik**
420 17 Olofstorp (SE)

(74) Representative: **Romare, Laila Anette**
Albihns AB
P.O. Box 142
401 22 Göteborg (SE)

(54) A method for adapting a combustion engine control map

(57) The invention relates to a method for adapting a combustion engine control map, which map comprises a set of nodes where each node is represented by a local model. The method involves the steps of receiving a measured sample of an engine parameter at an operating point, generating artificial samples in the coordinates of local models located adjacent the operating point, and updating the local models on both sides of the operating point using an update algorithm. This allows an engine control map to be updated over a relatively large area or region.

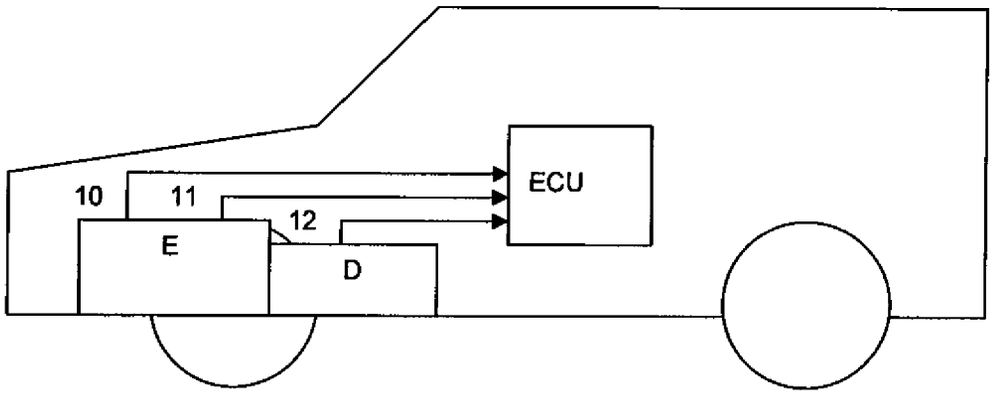

FIG. 15

EP 1 972 767 A1

Description

TECHNICAL FIELD

[0001] According to a preferred embodiment, the invention relates to a method for adapting a combustion engine control map, which map comprises a set of nodes where each node is represented by a local model. Algorithms may be used for generating artificial samples and updating the engine control map based on a measured sample of an engine parameter.

BACKGROUND ART

[0002] The automotive industry is given ever increasing demands to lower emissions and fuel consumption on the vehicles which they produce. When these demands continue to grow it becomes both cost effective and necessary to improve common engineering solutions. One important means to lower emissions and fuel consumption is to improve the control system of the engines.

[0003] Contemporary engine control systems contain a considerable amount of static maps. The maps are linear or nonlinear functions of one or several variables, which often describe a physical phenomenon or a function with no apparent physical interpretation used by the control system. The static maps are used by e.g. nonlinear controllers and static feed forward controllers. Some static maps are adapted online with some optimization algorithm acting on an incoming sample. These samples are referred to as *measurement samples,* although they might not be generated by a physical measurement. The reasons for the need of online adaptation are manifold. Three important reasons are aging of engines, mechanical differences between engines, and that the map can be dependent of many variables which are not practically possible to include as input variables. The inclusion of many variables is not practical partly due to an undesired necessity for high dimensional maps and partly due to uncertainty of how the variables influence the map.

[0004] The object of the invention is to improve the adaptation process to achieve more accurate maps. Improvements of adaptive static maps in accordance with the invention will in turn result in an improvement of the control system and consequently lowered emissions and fuel consumption.

DISCLOSURE OF INVENTION

[0005] The above problems are solved by a method according to claim 1 and a vehicle comprising an electronic control unit for implementing said method, according to claim 19.

[0006] According to a preferred embodiment, the invention relates to a method for adapting a combustion engine or driveline control map, which map is expressed by a basis function equation. The method involves the steps of:

- receiving a measured sample of an engine parameter at an operating point,
- updating local models adjacent of the operating point using an update algorithm,
- generating artificial samples in coordinates of local models located remote from the operating point and the said adjacent local models, and
- updating the said remote local models using an update algorithm.

[0007] Artificial samples are generated in coordinates of local models in a map expressed by a basis function equation, which is described in connection with Equation 1 below.

[0008] In a preferred embodiment, generating artificial samples can be done using an actualizing algorithm, such as a local pattern regression model (LPRM).

[0009] According to a preferred example of this embodiment, the method involves updating the local models in a map represented by a look-up table. The local models can be updated using a recursive least squares (RLS) algorithm, a direct adjustment (DA) algorithm, a least means squares (NLMS) or a normalized lest means squares (LMS) algorithm

[0010] According to a further preferred example of the embodiment, the method involves updating the local models in a map represented by a local linear neuro-fuzzy model (LLNFM). In this case the coordinates of a local models i (LM_i) is simply given by the center of the validity function of such a local linear model. The local models can be updated using a recursive least squares (RLS) algorithm, a least means squares (NLMS) or a normalized lest means squares (LMS) algorithm.

[0011] In an alternative embodiment, generating artificial samples can be done using a tent roof tensioning (TRT) algorithm.

[0012] According to a first example of this embodiment, the method involves updating the local models in a map represented by a local linear neuro-fuzzy model (LLNFM). The local models can be updated using a recursive least squares a (RLS) algorithm, a least means squares (NLMS) or a normalized lest means squares (LMS) algorithm.

[0013] According to a further example of this embodiment, the method involves updating the local models in a map represented by a look-up table. The local models can be updated using a recursive least squares (RLS) algorithm, The local models can be updated a least means squares (NLMS) or a normalized lest means squares (LMS) algorithm.

[0014] In addition, the invention relates to a vehicle comprising an electronic control unit for controlling a combustion engine or a vehicle driveline and sensors for measuring at least one engine or driveline related parameter, where the electronic control unit is provided with a map of measured or estimated samples for the said at least one parameter. The map provided in the electronic control unit is adapted using the above method.

[0015] The invention also relates to a combustion engine comprising an electronic control unit for controlling said combustion engine and sensors for measuring at least one engine parameter. The electronic control unit may be provided with a map of measured or estimated samples for at least one of the said engine parameters. The map or maps may be adapted using the method described above.

[0016] Static maps with online adaptation will be viewed in a comprehensive perspective and thus referred to as adaptive static maps, where different components of the maps are reviewed from the perspective to design better adaptive maps.

1 GENERAL BACKGROUND

[0017] Three major components can be distinguished in the design of adaptive static maps, these are given below.

- Map representation
- Update algorithm (local adaptation)
- Actualization (spreading)

[0018] By static maps are meant nonlinear memory less functions that perform a mapping from input space X to output space Y, i.e. $\hat{y}(x)\mathrm{X}\rightarrow\mathrm{Y}$, where $X \subset R^D$, $Y \subset R$ and D denotes the dimension of the input space and the map. Moreover no extrapolation outside the defined input space is considered. The output space is always one-dimensional whereas the dimension of the input space is arbitrary, though only one and two dimensions will be considered here.

[0019] Some general theory on which nonlinear static maps rely is introduced below. As mentioned above the map will receive measurement samples during operation. The measurement samples consist of input output pairs (x_i,y_i) and the learning algorithm has no control of the operating point of these, thus passive learning. The input coordinate of the latest samples is also referred to as operating point $\xi = x_i$. The process which adapts the map with regard to these samples is referred to as online adaptation. Initial parameter optimization and online adaptation is distinguished by the terms optimization and adaptation, respectively.

[0020] The measurement samples can be received in three different ways with respect to time:

1. Periodically incoming samples
2. Periodically incoming samples except for non-steady state situations
3. Sporadically incoming samples

[0021] Some of the algorithms must be adjusted depending on which of these sampling situations is given by the application. Here the second type of sampling situation will be considered.

[0022] The online adaptation with respect to these samples is done by an update algorithm. In most map architectures the update algorithm acts locally on the map. Hence update algorithms applied on measurement samples will only adjust the map in a small region around the sample. This problem is aggravated by the fact that in many applications the measurement samples are rarely or never received in large areas of the input space. Hence one problem to be solved by the invention relates to development of adaptive static maps which adapt larger areas of the map than the known methods which only update the map locally around the measurement samples.

[0023] The primary method that will be considered here for solving this problem is by employing actualization algorithms, which spread the adjustment to larger regions of the map. Hence the main purpose of the invention is to provide improved actualization algorithms. The actualization algorithms should fulfil the following requirements:

- Spread adjustments to larger areas of the map
- To integrate some form of a priori-information of possible variations in the map
- The algorithm should not depend explicitly on the form of the map
- It should not spread information to undesired regions of the map

- Reasonable memory and computation time requirements
- It should be scalable
- It must remain stable, i.e. the map must remain bounded

[0024] In the description below, measured and estimated maps are distinguished by denoting the measured map with $y(x)$ and the estimated map with $\hat{y}(x)$. It is also assumed that the measured map $y(x)$ includes additive noise, i.e. $y = y_u + n$, where the noise free measured map is denoted with $y_u(x)$. Hence the goal of a map is to be as close to $y_u(x)$ as possible.

[0025] Two major time variables are used; t represents a continuous time variable while k represents a discrete time variable that counts the order of the incoming samples. Variables and functions that are indexed or followed by brackets with k or t always refer to time, discrete and continuous respectively. Indexation with i or j refer to distinct variables, functions or values.

[0026] A list of abbreviated terms, constants, functions, sets and variables used in this text are given under the section "Notations" below.

2 - MAP REPRESENTATION

[0027] Static non-linear functions can be represented with many different model architectures. This chapter describes two different architectures which are suitable for adaptive static maps. The first is classic look-up tables and the other is called local linear neuro-fuzzy models (LLNFM) which is a modern architecture based on fuzzy logic.

[0028] Many map representations can be written on the basis function framework indicated in equation (1), where the output $\hat{y}$ is a sum of *basis functions* $\Phi_i(x,\theta_i^{(nl)})$ weighted with functions $L_i(x,\theta_i)$ which are linear in its parameters θ , these are referred to as *linear functions.* The values of the basis functions are determined by the input x and its parameters $\theta_i^{(nl)}$. The basis functions are in general nonlinear in its parameters $\theta_i^{(nl)}$. Various map representations differ from each other by having different types of linear models and basis functions. The basis function is often written in an abbreviated form without the parameters. With this framework it is clear that the parameters of nonlinear map representations can be separated in two categories; the basis function parameters $\theta^{(nl)}$ and the linear model parameters θ. This convenient fact is made use of in the optimization and adaptation of the map. The map is expressed by a basis function equation

$$\hat{y}(x) = \sum_{i=1}^{M} L_i(x,\theta_i)\Phi_i(x,\theta_i^{(nl)}) \tag{1}$$

where

θ_i is a height parameter of node i,

$L_i(x,\theta_i)$ is a linear function in the basis function framework,

$\Phi_i(x,\theta_i^{(nl)})$ is a basis function in the basis function framework, where (nl) indicates that the function is non-linear,

M is the number of nodes in a one-dimensional map or a LLNFM of arbitrary dimension

[0029] Perhaps the most important performance measure of a map representation is how well it approximates the measured map. The measure is defined as model error and is given in equation (2), where y_u represents a noise free measured map. This measure can be estimated from the training data used for the optimization of the map, which will be discussed next. The measurement samples which are used for estimation of the map are assumed to have additive noise n, with variance σ_n , i.e. $y = y_u + n$. The model error can further be decomposed into bias error and variance error according to equation (3). This is described in Nelles, O. (2001). *Nonlinear System Identification.* Springer-Verlag, Berlin Heidelberg, hereinafter referred to as [Nelles].

$$e_{\text{model}} = \left| y_u(x) - \hat{y}(x) \right| \tag{2}$$

$$e^2_{\text{model}} = e^2_{\text{bias}} + e_{\text{var}} \tag{3}$$

[0030] The bias error is due to the inflexibility of the model. The flexibility of the model is determined by how well the structure of the map representation describes the measured map and it grows with the number of parameters in the model. The variance error arises from having parameter values which deviate from their optimal values. Variance error is reduced by having a large number of training data S and minimizing the variance σ_n of the noise in the training samples, furthermore the variance error increases with the number of parameters in the model. In equation (4) a general approximate relation of how these variables influence the variance error is given, this hold generally regardless of map representation as described in [Nelles].

$$e_{\text{var}} \approx \sigma_n^2 \frac{\#\ parameters}{\text{S}} \tag{4}$$

[0031] Thus determining the number of parameters in a model is based on a trade-off between the bias and variance error. This trade-off will not be addressed in further detail here. It is though important to realize what can be accomplished with online adaptation. Every adaptation algorithm that is considered here will only change the values of the linear model parameters in the map representation of equation (1). Hence the minimization of the variance error with respect to the parameters θ of the linear function is the conceivable overall goal of the adaptation algorithms. Variance error is from here on always considered with respect to the estimation of the linear model parameters.

[0032] Deriving an analytical expression of the variance error of the model output $\hat{y}(x)$ due to the error in the estimation of the linear model parameters θ is done by taking the covariance of the output of equation (5), where each diagonal element gives the variance error of the model output at every sample (x_i, y_i) used in the estimation of the parameters θ [Nelles]. Note that X is the regressors used for the estimation of the parameters θ and that they are general regressors and are unique for the specific map architectures. The size of the diagonal elements of cov{θ} gives the variance error of the parameters of the linear function.

$$\begin{aligned} e_{\text{var}} &= \text{cov}\{\hat{y}\} \\ &= E\{(\hat{y} - E\{\hat{y}\})(\hat{y} - E\{\hat{y}\})^T\} \\ &= E\{(X(\theta - E\{\theta\}))(X(\theta - E\{\theta\}))^T\} \\ &= X\,\text{cov}\{\theta\}\,X^T \end{aligned} \tag{5}$$

2.1 Look-Up Tables

[0033] The most widely used map representation in industrial applications is look-up tables [Nelles]. Look-up tables consist of interpolation nodes which are distributed on a grid and are located by a coordinate c and each node is associated with a height θ. The output of a look-up table is given by interpolation between the nodes which spatially surrounds the input coordinate in each dimension and their belonging heights. Linear interpolation is used in the one dimensional case, see equation (6) and Figure 1a-b, whereas area (bilinear) interpolation is used in two dimensional maps, see equation (7) and Figure 2, where the areas are given by equations (8)-(12)

$$\hat{y}(x) = \theta_i \frac{(c_{i+1} - x)}{c_{i+1} - c_i} + \theta_{i+1} \frac{(x - c_i)}{c_{i+1} - c_i} \tag{6}$$

$$\hat{y}(x) = \theta_{i,j}\frac{A_{i+1,j+1}}{A} + \theta_{i+1,j}\frac{A_{i,j+1}}{A} + \theta_{i,j+1}\frac{A_{i+1,j}}{A} + \theta_{i+1,j+1}\frac{A_{i,j}}{A} \quad (7)$$

$$A_{i+1,j+1} = (c_{1,i+1} - x_1)(c_{2,j+1} - x_2) \quad (8)$$

$$A_{i,j+1} = (x_1 - c_{1,i})(c_{2,j+1} - x_2) \quad (9)$$

$$A_{i+1,j} = (c_{1,i+1} - x_1)(x_2 - c_{2,j}) \quad (10)$$

$$A_{i,j} = (x_1 - c_{1,i})(x_2 - c_{2,j}) \quad (11)$$

$$A = (c_{1,i+1} - c_{1,i})(c_{2,j+1} - c_{2,j}) \quad (12)$$

[0034] Look-up tables can be written on the basis function framework. The height parameters correspond to the linear functions and the spatial interpolation between the nodes corresponds to the basis functions. Only the basis functions $\Phi_{i,j}$ that are within the active interpolation area are non-zero, see equation (13) where uniformly two dimensional look-up tables are assumed.

[0035] Compare equation (7) above and the formal basis function given by equation (14), which both describes the output of a two dimensional look-up table.

$$\begin{aligned} &\Phi_{i,j}(x) \neq 0\,, \quad \forall\, i,j \in \left\{ i,j : \left(\left| c_{1,i} - x_1 \right| \leq c_{1,2} - c_{1,1} \right) \wedge \left(\left| c_{2,j} - x_2 \right| \leq c_{2,2} - c_{2,1} \right) \right\} \\ &\Phi_{i,j}(x) = 0\,, \quad else \end{aligned} \quad (13)$$

$$\hat{y}(x) = \sum_{i=1}^{M_1} \sum_{j=1}^{M_2} \theta_{i,j} \Phi_{i,j}\left(x, \begin{bmatrix} c_{1,i} \\ c_{2,j} \end{bmatrix}\right) \quad (14)$$

[0036] The nodes are usually uniformly distributed in the input space, but it is also possible to have a non-uniform distribution [Nelles]. From here on, when referring to look-up tables, a uniform distribution is assumed. It is also assumed that all nodes are fixed a priori. The optimization of the location and number of nodes will not be addressed here.

[0037] Determining the initial heights θ of the map is a pure linear optimization task. It is done by minimizing the sum of squared errors in equation (15) over all measurement samples S and solved by the least squares algorithm of equation

(16) [Nelles].

$$\sum_{i=1}^{S}(y_i-\hat{y}_i)^2 \tag{15}$$

$$\theta=(X^TX)^{-1}X^Ty \tag{16}$$

[0038] Where

$$X=\begin{bmatrix}\Phi_1(x_1) & \Phi_2(x_1) & \dots & \Phi_M(x_1)\\ \Phi_1(x_2) & \Phi_2(x_2) & \dots & \dots\\ \dots & \dots & \dots & \dots\\ \Phi_1(x_S) & \Phi_2(x_S) & \dots & \Phi_M(x_S)\end{bmatrix},\quad y=\begin{bmatrix}y_1\\ y_2\\ \dots\\ y_S\end{bmatrix},\quad \theta=\begin{bmatrix}\theta_1\\ \theta_2\\ \dots\\ \theta_M\end{bmatrix} \tag{17}$$

[0039] In the beginning of the current section a general expression in equation (5) of the variance error of the output due to the estimation of the parameters θ in the linear function was given. By applying this expression to the output of a look-up table where linear parameters θ are estimated by equation (16), the following expression appears in equation (18).

$$\begin{aligned}e_{\text{var}} &= X\,\text{cov}\{\theta\}\,X^T\\ &= X\,E\left\{\left((X^TX)^{-1}X^T(y-E\{y\})\right)\left((X^TX)^{-1}X^T(y-E\{y\})\right)^T\right\}X^T\\ &= X\left((X^TX)^{-1}X^TE\{n\}\left(E\{n\}\right)^TX(X^TX)^{-1}\right)X^T\\ &= X\left((X^TX)^{-1}X^TE\left\{nn^T\right\}X(X^TX)^{-1}\right)X^T\\ &= X\left(\sigma_n^2(X^TX)^{-1}\right)X^T\end{aligned} \tag{18}$$

[0040] Notice that $y-E\{y_u+n\}=y-y_u=n$ and in the two last equalities it is assumed that noise is white which implies $E\{n\}$ ($E\{n\}=E\{nn^T\}$ and $E\left\{nn^T\right\}=\sigma_n^2 I$. The diagonal in the last expression within the outermost brackets of equation (18) gives the variance error of the estimated parameters θ .

[0041] In Nelles several properties of look-up tables are given, some of the more relevant are stated here. Three positive important benefits of using look-up tables are that they have high evaluation speed, the parameters of the linear models can be optimized fast, and they are simple to implement. Two negative properties are non-smoothness and that they suffer severely from the curse of dimensionality. With curse of dimensionality is meant that the memory requirements grow fast with the dimension of the map.

2.2 Local Linear Neuro-Fuzzy Models

[0042] This map representation has been developed in parallel in various scientific fields [Nelles] and is generally less well-known than look-up tables. Readers who are familiar with fuzzy logic will recognize that *local linear neuro fuzzy models* (LLNFM) are equivalent to first order Takagi-Sugeno with axis-orthogonal Gaussian membership functions and

product operator used for conjunction [Nelles]. In order to clarify some concepts relating to fuzzy logic, the LLNFMs are described in detail below.

[0043] LLNFMs can also be written on the basis function framework given above. The basis functions in LLNFMs are normalized Gaussian functions, see equation (19). These Gaussian functions are also referred to as validity functions, because they give the size of how much their respective local models are affecting the output. By normalized are meant that the sum of all *M* basis functions always ad up to one, see equation (20). It is also assumed that the Gaussian functions are axis orthogonal, i.e. the parameters which determine the width and position of the function in each dimension are independent of each other.

$$\Phi_i(x) = \frac{\mu_i(x)}{\sum_{j=1}^{M} \mu_j(x)}, \quad \mu_i(x) = \exp\left(-\frac{1}{2}\left(\frac{(x_1 - c_{i1})}{\sigma_{i1}^2} + \ldots + \frac{(x_D - c_{iD})}{\sigma_{iD}^2} \right) \right) \tag{19}$$

$$\sum_{i=1}^{M} \Phi_i(x) = 1, \quad \forall x \tag{20}$$

[0044] Here $c_{i,j}$ is the center position of the validity function *i* in dimension j and $\sigma_{i,j}$ determines the width of validity function *i* in dimension *j*.

[0045] The linear functions in the basis function framework are here made up of local linear models (LLM) with respect to the input coordinate x, see equation (21). By forming the basis function representation, the output of the LLNFM results in equation (22). The Gaussian validity functions are always non-zero, thus all the LLMs always contribute to the output, although in varying degree with respect to the input coordinate.

$$L_i(x, \theta_i) = \theta_{i0} + \theta_{i1} x_1 + \ldots + \theta_{iD} x_D \tag{21}$$

$$\hat{y}(x) = \sum_{i=1}^{M} (\theta_{i0} + \theta_{i1} x_{i1} + \ldots + \theta_{iD} x_{iD}) \Phi_i\left(x, \begin{bmatrix} c_i \\ \sigma_i \end{bmatrix}\right) \tag{22}$$

[0046] In the general basis function framework, the model parameters can be divided into two categories; those in the basis function and those in the linear function. The parameters in the linear function are clearly the LLM parameters $\theta_{i,j}$ while the parameters in the basis function are the center positions $c_{i,j}$ of the validity functions and their widths $\sigma_{i,j}$, . This is advantageous because if the validity functions are specified, which is done by $c_{i,j}$ and $\sigma_{i,j}$, the linear model parameters are easily optimized with least squares.

[0047] There are two different approaches for the optimization of the linear model parameters, a global and a local. The global approach optimizes all linear model parameters with respect to all the measurement samples simultaneously. The second approach neglects the overlapping of the validity functions and optimizes the local linear models separately. In [Nelles] the two approaches are compared; the global approach has a smaller bias error while the local has a smaller variance error and it also has lower computational complexity, $O(MD^3)$ compared to $O(M^3D^3)$, and is more robust against noise. The same source states that the local approach is superior to the global variant in most applications, consequently local optimization will be considered from here on. The global approach is particularly unsuitable for the online adaptation, which is the focus here; because of its high computational complexity and its less robust behavior (more about online adaptation of LLNFM see Section 33). The local estimation is derived from the loss function in equation (23) for LLM *i* over all S sample pairs(x_j, y_j). Note that the errors are weighted with the current validity functions. This optimization problem is solved with weighted least squares, in equation (24), where the parameter vector, weight matrix, and regressor matrix are given by equation (25). Observe that the regressor matrices are independent of *i*, since all data samples are

used in the estimation of every LLM.

$$J_i = \sum_{j=1}^{S} \Phi_i(x_j)(y_j - \begin{bmatrix} 1 & x_{j1} & \dots & x_{jD} \end{bmatrix} \begin{bmatrix} \theta_{j0} \\ \theta_{j1} \\ \dots \\ \theta_{jD} \end{bmatrix})^2 \tag{23}$$

$$\theta_i = \left(X^T Q_i X \right)^{-1} X^T Q_i y \tag{24}$$

$$Q_i = \begin{bmatrix} \Phi_i(x_1) & 0 & \dots & 0 \\ 0 & \Phi_i(x_2) & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \Phi_i(x_S) \end{bmatrix}, X = \begin{bmatrix} 1 & x_{11} & \dots & x_{1D} \\ 1 & x_{21} & \dots & x_{2D} \\ \dots & \dots & \dots & \dots \\ 1 & x_{S1} & \dots & x_{SD} \end{bmatrix}, y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_S \end{bmatrix}, \theta_i = \begin{bmatrix} \theta_{i0} \\ \theta_{i1} \\ \dots \\ \theta_{iD} \end{bmatrix} \tag{25}$$

[0048] If the widths $\sigma_{i,j}$ of the validity functions are small, the map will have small transition phases between the LLMs and thus less smooth steps occur in the map. On the other hand if they are large, the function will loose local accuracy. The center coordinates must be distributed by a structure optimization algorithm. A fast algorithm for this task is presented in [Nelles] which is called *Local Linear Model Tree* (LOLIMOT). It optimizes the structure of the map by clustering the input space incrementally. Because the emphasis here is placed on online adaptation, where only the linear model parameters are adapted, the structure optimization will not be pursued in further detail.

[0049] From the general expression of the variance error of the model output given in equation (5), an expression for LLNFM follows equation (26), where it is assumed that the estimation is done with the local method given above [Nelles]. The parameter variance error for each LLM is given in equation (27), where the two last equalities are based on the assumption of white noise. The regressor and weight matrix are given by equation (25).

$$e_{\text{var}} = \text{cov}\{\hat{y}\} = \sum_{i=1}^{M} Q_i \, \text{cov}\{\hat{y}_i\} = \sum_{i=1}^{M} Q_i X \, \text{cov}\{\theta_i\} X^T \tag{26}$$

$$\begin{aligned} &\text{cov}\{\theta_i\} \\ &= E\left\{ \left((X^T Q_i X)^{-1} X^T Q_i (y - E\{y\}) \right) \left((X^T Q_i X)^{-1} X^T Q_i (y - E\{y\}) \right)^T \right\} \\ &= (X^T Q_i X)^{-1} X^T Q_i E\{n\} \left(E\{n\} \right)^T Q_i X \, (X^T Q_i X)^{-1} \\ &= (X^T Q_i X)^{-1} X^T Q_i E\{nn^T\} Q_i X \, (X^T Q_i X)^{-1} \\ &= \sigma_n^2 \left(X^T Q_i X \right)^{-1} X^T Q_i Q_i X \left(X^T Q_i X \right)^{-1} \end{aligned} \tag{27}$$

[0050] Some relevant beneficial properties of LLNFM are fast linear model parameter optimization, easily controlled smoothness of the map, and the curse of dimensionality is low [Nelles]. Hence in applications with high dimensional

maps the LLNFM is a better choice than look-up table if memory requirements are crucial. One relevant negative property is that they only have medium evaluation speed [Nelles]. Look-up tables are by far more frequently used in applications, thus look-up tables will be the most considered map representation in the remaining chapters.

3 - UPDATE ALGORITHMS

[0051] In Section 2 above it was explained that the parameters in the map representations can be separated in two categories; basis function parameters and linear model parameters. Due to this fact and the fact that linear optimization methods are robust, fast, and easy to implement, only the linear model parameters will be adapted during online operation. This further implies that only the variance error of the map with respect to the linear model parameters is reduced by the online adaptation.

[0052] When look-up tables are updated the heights of the two most adjacent nodes in each dimension will change their values, see also Figures 1a, 1b and 2. When LLNFMs are updated only the linear model parameters of those LLMs which have a valid function larger than a given threshold in the current operating point of the sample are updated, i.e. $\Phi_i(x) > \Phi_{thr}$.

3.1 Direct Adjustment

[0053] The most straightforward way to update a look-up table is to change the values of the height parameters so that the map has the same value in the operating point $x=\xi$ as the new measurement, according to equation (28).

[0054] This update algorithm is most often used in applications with look-up tables [Nelles]. Here the height parameters of the current interpolation area are given the same value as the new sample. Note however that when samples are received precisely on the coordinate of a height parameter; only that parameter is updated. This feature is used for the actualization algorithms presented in section 4.

$$\hat{y}_{k+1}(x_i) = \begin{cases} y_k(\xi), & for\ x_i = \xi \\ \hat{y}_k(x_i), & for\ x_i \neq \xi \end{cases} \tag{28}$$

[0055] According to the *stabilitylplasticity dilemma* there is a tradeoff in learning systems between the speed of the adaptation ("plasticity") and the ability of good noise attenuation ("stability") [Nelles]. In the *direct adjustment* (DA) algorithm the plasticity is maximized and the stability is minimized, because all the earlier measurements are discarded while merely the newest sample determines the current estimate. The estimated variance error of a look-up table in the coordinate of the latest measurement sample (x_i,y_i) in the immediate time after the update with DA, is simply the variance of the measurement noise of the sample, i.e. $\mathrm{cov}\{\hat{y}_{k+1}(x_i)\} = \sigma_i^2$. Hence the algorithm is unsuitable in applications where the noise level is significant. Furthermore when actualization methods are applied, the error in the measurement will be spread to large areas of the map. However, its evident advantages are extreme low computational requirements and simplicity.

3.2 Normalized Least Mean Squares

[0056] The most commonly used update algorithm for online adaptation generally, is the *normalized least mean squares* (NLMS) algorithm. This is described in Vogt, M., Müller, N., and Isermann, R. (2004). On-Line Adaptation of Grid-Based Look-up Tables Using a Fast Linear Regression Technique. Journal of Dynamic Systems, Measurement, and Control, December 2004, Vol. 126, hereinafter referred to as [Vogt et al., 2004]. It is a linear first order optimization method and it has very low computational requirements [Nelles]. The update algorithm given in equation (29) is applied on the height parameter of node *i* in a one dimensional look-up table, where the learning rate η must be set within $0 < \eta < 2$ [Vogt et al., 2004]. If the measurement sample is located between node *i* and *i*+1, in one dimensional look-up tables, both node *i* and *i*+1 is updated according to equation (29). This is done analogously in two dimensional look-up tables, where the four nodes which belong to the current interpolation area are updated. If the denominator is omitted the algorithm is simply called *least mean squares.*

$$\theta_i(k+1) = \theta_i(k) + \eta\,\varepsilon(k)\frac{\Phi_i(x(k))}{\sum_{j=1}^{M}\Phi_j^2(x(k))}, \quad \varepsilon(k) = y_k(\xi) - \hat{y}_k(\xi), \quad x(k) = \xi \tag{29}$$

3.3 Recursive Least Squares

3.3.1 Description

[0057] The *recursive least squares* (RLS) algorithm is a linear second order optimization method [Nelles]. It is summarized in (30)-(32), where *X* and θ are general regressors and parameters respectively. Its time complexity is of the order O(#*parameters*^{2}) and it is also extended with forgetting factor λ, and weight factor *w* [Nelles].

$$\theta(k) = \theta(k-1) + \gamma(k)\varepsilon(k), \quad \varepsilon(k) = y_k(\xi) - X^T(k)\theta(k-1) \tag{30}$$

$$\gamma(k) = \frac{1}{X^T(k)P(k-1)X(k) + \lambda / w(k)}P(k-1)X(k) \tag{31}$$

$$P(k) = \frac{1}{\lambda}(I - \gamma(k)X^T(k))P(k-1) \tag{32}$$

[0058] The forgetting factory λ enables the algorithm to follow time-variant processes. The value of it is determined by the *stability*/*plasticity* trade-off; good noise attenuation (large λ) versus fast learning (small λ). Usually the value is set between 0.9 and 1. As mentioned above the algorithm also has a weight factor which determines how much a sample should influence the estimation.

[0059] If the algorithm is given many samples in the same operating point, equation (32) will be reduced to approximately equation (33). This is described in Åström, K.J. and Wittenmark, B. (1989). Adaptive Control. Addison-Wesley Publishing Company, hereinafter referred to as [Åström and Wittenmark, 1989]. This makes the covariance matrix *P* grow with an exponential rate. A simple way to overcome this problem is to have a restriction in the algorithm which stops it when the residual ε or *PX* becomes smaller than a dead-zone [Åström and Wittenmark, 1989].

$$P(k) = P(k-1)/\lambda \tag{33}$$

3.3.2 Implementation

[0060] In [Vogt et al., 2004] it is shown that the RLS (the modified version presented below) converges faster than NLMS while the memory requirements are only twice as high when it is applied on look-up tables. Moreover the RLS has a convenient way of weighing the leverage of the samples in the estimation which can be used in actualization algorithms. Consequently RLS and DA will be the standard updating algorithms from here on. Next, two ways of how the RLS can be implemented in look-up tables and LLNFMs will be described.

[0061] An approach of how the RLS can be implemented on look-up tables is given in [Vogt et al., 2004], called *Modified RLS,* where a two dimensional map is considered. It is assumed that a sample only affects the surrounding four nodes within the current interpolation area, see equations (34)-(38) and Figure 2.

$$\tilde{\theta} = [\theta_{i,j}, \theta_{i,j+1}, \theta_{i+1,j}, \theta_{i+1,j+1}]^T \quad (34)$$

$$\tilde{\theta}(k) = \tilde{\theta}(k-1) + \gamma(k)\varepsilon(k), \quad \varepsilon(k) = y_k(x) - X^T(k)\tilde{\theta}(k-1) \quad (35)$$

$$\gamma(k) = \frac{1}{X^T(k)P(k-1)(k) + \lambda / w(k)} P(k-1)X(k) \quad (36)$$

$$P(k) = \frac{1}{\lambda}(I - \gamma(k)X^T(k))P(k-1) \quad (37)$$

$$X(k) = [\Phi_{i,j}(x(k)), \Phi_{i,j+1}(x(k)), \Phi_{i+1,j}(x(k)), \Phi_{i+1,j+1}(x(k))] \quad (38)$$

[0062] This continues until measurements are received in another interpolation area.

[0063] Then the diagonal elements in the covariance-matrix *P* are stored in a variance matrix *V* in memory. The new diagonal elements in the *P* -matrix for the new interpolation area are given by the stored variance matrix, while the covariance elements (non-diagonal) are set to zero. Initially the variance elements can be given relatively large values (100-1000) which will give a fast initial convergence. High variance values indicate uncertain values. The flow chart of the modified RLS algorithm is depicted in Figure 3.

[0064] **LLNFM**. As mentioned in Section 22, the optimization of the linear model parameters is done locally, i.e. the local linear models are adapted one at a time. But contrary to the offline optimization which optimizes all LLM to every sample, the online adaptation adapts only those LLM whose validity function is larger than a threshold $\Phi_i(x) > \Phi_{thr}$ in the operating point of each sample.

[0065] The choice of this strategy is based on two arguments. The first is obviously a lower computational demand and the other is to ensure robustness to insufficient excitation of the map. Otherwise, if the samples are not uniformly distributed to all LLM, the heights of the non-excited LLM will converge to the height of the incoming samples despite the small size of their validity functions in the operating point, and thereby causing destructive learning of the non-excited LLM. Non-uniform distribution of the samples is an assumption in the problem description here, see Section Therefore Φ_{thr} should be large enough so that besides the most active LLM, at most its neighboring LLM are updated. If no separate actualization algorithm is used, it may be advantageous to have a Φ_{thr} small enough so that the neighboring LLM are updated, which will give a local actualization, this is referred to as *broad updating.* But when separate actualization algorithms are used, only the most active LLM is updated, to avoid a mixture of two different actualizations.

[0066] The RLS algorithm applied on a one dimensional LLNFM with the local adaptation strategy described above is given in equations (39)-(43). Note that the regressor has ones in the left column due to the bias parameter; also note that the samples are weighted with their validity functions. The covariance matrix for every LLM is always kept in memory.

$$\theta(k) = \theta(k-1) + \gamma(k)\varepsilon(k), \quad \varepsilon(k) = y_k(x) - X^T(k)\theta(k-1) \quad (39)$$

$$\gamma(k) = \frac{1}{X^T(k)P(k-1)X(k) + \lambda / \Phi_i(x(k), C)} P(k-1)X(k) \quad (40)$$

$$P(k) = \frac{1}{\lambda}(I - \gamma(k)X^T(k))P(k-1) \quad (41)$$

$$\theta = [\theta_{i0}, \theta_{i1}] \quad (42)$$

$$X(k) = [1, x(k)] \quad (43)$$

BRIEF DESCRIPTION OF DRAWINGS

[0067] In the following text, the invention will be described in detail, in some cases with reference to the attached drawings. These schematic drawings are used for illustration only and do not in any way limit the scope of the invention. In the drawings:

Figure 1 a shows a linear interpolation between two nodes in a one dimensional look-up table;

Figure 1b shows the basis functions of the two current nodes;

Figure 2. shows an interpolation area of a two dimensional look-up table;

Figure 3. shows a flow chart of the modified RLS algorithm;

Figure 4. shows tent roof tensioning with DA on a look-up table;

Figure 5a-5e show examples of LPRM, artificial samples;

Figure 6a-6b show example of possible boundaries for permitted actualization in two dimensional look-up tables;

Figure 7 shows an actualization cycle of LPRM on 2-D look-up tables;

Figure 8a-d shows a common start map (Fig.8a) and the result of three simulations using this start map;

Figure 9 shows an example of a realization of the driving cycle;

Figure 10 shows a histogram of the drive cycle realization in Fig. 9;

Figure 11 a-d show snapshots of simulations of a first map with TRT, which are taken at the first sample k=1; during the short period of higher engine speed k=60; the highest point of the map k=300 and the last sample k=500;

Figure 12a-d show snapshots of simulations of a first map with LPRM, which are taken at the first sample k=1; during the short period of higher engine speed k=60; the highest point of the map k=300 and the last sample k=500;

Figure 13a-d show snapshots of simulations of a first map with TRT, which are taken at the first sample k=1; during

the short period of higher engine speed k=60; the highest point of the map k=300 and the last sample k=500;

Figure 14a-d shows snapshots of simulations of a first map with LPRM, which are taken at the first sample k=1; during the short period of higher engine speed k=60; the highest point of the map k=300 and the last sample k=500; and

Figure 15 shows a schematic illustration of a combustion engine comprising a control unit provided with a map on which the method can be implemented.

EMBODIMENTS OF THE INVENTION

[0068] The online adaptation with respect to periodically incoming samples is done by an update algorithm. In most map architectures the update algorithm acts locally on the map. As stated above, update algorithms applied on measurement samples will only adjust the map in a small region around the sample. This problem is aggravated by the fact that in many applications the measurement samples are rarely or never received in large areas of the input space. Hence one problem to be solved by the invention relates to development of adaptive static maps which adapt larger areas of the map than the known methods which only update the map locally around the measurement samples.

[0069] The primary method that will be considered here for solving this problem is by employing actualization algorithms, which spread the adjustment to larger regions of the map. Hence the purpose of the invention is to provide improved actualization algorithms. The actualization algorithms should fulfil the following requirements:

- Spread adjustments to larger areas of the map
- To integrate some form of a priori-information of possible variations in the map
- The algorithm should not depend explicitly on the form of the map
- It should not spread information to undesired regions of the map
- Reasonable memory and computation time requirements
- It should be scalable
- It must remain stable, i.e. the map must remain bounded

4 - ACTUALIZATION ALGORITHMS

[0070] If merely an update algorithm is used in online adaptation, the map will only be adapted locally where the new measurement samples are received. Therefore actualization algorithms should be employed. Section 41 describes the *tent roof tension* (TRT) algorithm which was presented in Heiss, M. (1997). Online Learning or Tracking of Discrete Input-Output Maps. IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-PART A: SYSTEMS AND HUMANS, VOL. 27, NO. 5, September 1997, hereinafter referred to as [Heiss, 1997] and suggest a modified version which is applicable to the current case, while Section 42 will introduce a new algorithm for spreading adjustments to larger portions of the map. For simplicity the algorithms are presented in their one-dimensional versions (D=1), while the two-dimensional cases are commented at the end.

As mentioned in Section 1.2 a framework has been developed here that gives a clear demarcation between the map representation, the update algorithm, and the actualization algorithm. This is made possible by having the actualization algorithms realize the actualization over the map by placing artificial samples ${}^{a}y^{(i)}$ in the coordinates of general local models (LM), independent of map representation. The LMs are thereafter updated with the preferred update algorithm. When referring to specific local models explicitly, the indexation is written at the upper right part of the variable or function in brackets e.g. $\hat{y}^{(i)}$ or $x^{(i)}$. The value of a local model $\hat{y}^{(i)}$ is defined as the map value in the coordinated $x^{(i)}$ of the local model. Hence artificial samples are always generated in the coordinates of the local models. This further implies that the information in the actualization is an estimate of the map value in the coordinates of the LMs and no information is given of how the details of the map around the coordinates of the LMs should look like. Therefore these details should be kept unchanged if possible. Thanks to this framework the actualization algorithms are described in general terms and are applicable on both map representations presented above and compatible with the update algorithms given in Section 2. To simplify notation the update of a general update algorithm is abbreviated with an operator (44), indicated below, where the operator acts on the sample $y_k(\xi)$ and $\hat{y}_k$ which signifies all the information of the estimated map, that is, both the map and the variance/covariance matrices stored in memory.

$$\hat{y}_{k+1} = \mathcal{U}\{\hat{y}_k, y_k(\xi)\} \tag{44}$$

[0071] If look-up tables are used as map representation, the general LMs refer to the individual nodes in the map. Note that when DA is used, the update of a measurement sample will adjust the two surrounding LMs in each dimension, while artificial samples only affect the LM on which the sample is placed. This holds for the modified RLS as well. However, when an artificial sample in say LM *i* is updated with the RLS, the height parameter of the neighboring node *i*+1 (or *i*-1) will not change its value, but its belonging variance value in *V* do change in the update of the *P*-matrix. Hence the variance value of *i*+1 (or *i*-1) should be kept unchanged in the update.

[0072] The LLNFM architecture is based on local linear models (LLM), thus the general LMs simply refer to these given by the architecture. The coordinate of LM *i* is simply given by the center of its validity function, i.e. $x^{(i)} = c_i$.

[0073] Following the same line of thought as for look-up tables, the artificial samples should only change the bias parameter of the actualized LLM and the rake parameter should be unchanged. If the RLS is used as update algorithm, only the bias parameter and its variance value should be changed. The other values; the rake parameter and its variance value and also the non-diagonal covariance values in the P-matrix should be kept unchanged in the update.

[0074] With this framework established, where the actualization algorithms generate artificial samples, a central question arises; how the actualization algorithms should estimate the values of the artificial samples. The remaining of this chapter will show how the two described actualization algorithms solve this problem quite differently.

4.1 Tent Roof Tensioning

4.1.1 Original Version

[0075] The algorithm was given in [Heiss, 1997] where it is implemented on a look-up table with discrete input space i.e. $X \subset \grave{} \ ^{D}$. Thus no interpolation is done between the nodes, each allowed input is associated with a single height and the update is done with simple *direct adjustment* (DA). The algorithm begins by updating the current operating point $x = \xi$ of the map. Thereafter the surrounding points which are within the distance *r* from the updated point will be adapted. The adaptation is based on a linear interpolation between the recently updated point $\hat{y}_{k+1}(\xi)$ and the points which are located at the distance r from the former. The linear interpolation gives the appearance of a tent roof with its center in the operating point $\hat{y}(\xi)$; which has given the algorithm its name. The algorithm is summarized in Algorithm 4.1.

Algorithm 4.1: Tent Roof Tensioning

[0076]

$$\begin{aligned}
&\hat{y}_{k+1}(\xi) = y(\xi) \\
&\hat{y}_{k+1}(x_i) = y(\xi) + \frac{\hat{y}_k(\xi + r) - y(\xi)}{r}(x_i - \xi), \quad if\ 0 < x_i - \xi < r \\
&\hat{y}_{k+1}(x_i) = y(\xi) + \frac{\hat{y}_k(\xi - r) - y(\xi)}{r}(\xi - x_i), \quad if\ 0 < \xi - x_i < r \\
&\hat{y}_{k+1}(x_i) = \hat{y}_k(x_i), \quad else
\end{aligned}$$

[0077] A problem can occur in case the operating point $x = \xi$ is located near a boundary of the map and the point $\xi \pm r \notin X$ is outside the map. This can be solved by horizontal extrapolation of the end points, i.e. if $\xi + r > M$ then $\hat{y}(\xi+r) = \hat{y}(M)$ and analogously when $x<1$ [Heiss, 1997]. Additionally [Heiss, 1997] gave an analytical proof which guarantees that the algorithm is stable and that it converges to a small error band around the measured map.

4.1.2 Modified Version

[0078] As recently mentioned the algorithm was in its original version given with the DA as its update algorithm and

a discrete input space. This section gives a suggestion of how the algorithm can be modified to the framework of generating artificial samples in the coordinates of LMs, which was presented above. This will make it applicable on both LLNFM and look-up tables with continuous input space and compatible with the preferred update algorithm.

[0079] **Look-up table**. When a sample is received in operating point $x=\xi$ between local models i and /+1 the preferred update algorithm updates the two LMs as described in Section 2. Thereafter artificial samples ${}^{a}y^{(j)}$ are created in the center of every LM located between the LMs i and i-r, and between i+1 and i+r+1. Subsequently the LMs within the "tent roof" are updated with the update algorithm. Note that the tent roof is formed by linear interpolation from the values of local models i and i-r in one direction and from local models i+1 and i+r+1 in the other direction. Figure 4 gives an example of the TRT applied on a look-up table with DA, using Algorithm 4.2. The radius of the tent roof is set to r=4 and consequently 3 artificial samples are created in each direction of the operating point.

Algorithm 4.2: Modified Tent Roof Tensioning

[0080]

- Update operating point

$$\hat{y}_{k+1} = \mathcal{U}\{\hat{y}_k, y_k(\xi)\}$$

- Create artificial samples

$${}^{a}y_{k+1}^{(j)} = \hat{y}_{k+1}^{(i+1)} + \frac{\hat{y}_k^{(i+1+r)} - \hat{y}_{k+1}^{(i+1)}}{x^{(i+1+r)} - x^{(i+1)}}(x^{(j)} - x^{(i+1)}), \quad \forall\ j \in \{i+2, i+3, i+4, ..., i+r\}$$

$${}^{a}y_{k+1}^{(j)} = \hat{y}_{k+1}^{(i)} + \frac{\hat{y}_k^{(i-r)} - \hat{y}_{k+1}^{(i)}}{x^{(i)} - x^{(i-r)}}(x^{(i)} - x^{(j)}), \quad \forall\ j \in \{i-1, i-2, i-3, ..., i+1-r\}$$

- Update actualized LMs

$$\hat{y}_{k+1} = \mathcal{U}\{\hat{y}_k, {}^{a}y_{k+1}^{(j)}\}, \quad \forall\ j \in \{i+2, i+3, i+4, ..., i+r\} \cup \{i-1, i-2, i-3, ..., i+1-r\}$$

[0081] Figure illustrates a tent roof tensioning with DA on a look-up table, wherein artificial samples are indicated by (`), a measurement sample by (∗), an estimated map before adaptation by (_._), and an estimated map after adaptation by (__) .

[0082] When the algorithm is applied on two dimensional look-up tables the algorithm starts by placing a base for the tent, which forms a square around the updated local models (i_1,i_2),(i_1,i_2+1), (i,+1,i_2), (i_1+1,i_2+1). The square is given by connecting nodes between the nodes (i_1, -r,i_2 -r),(i, -r,i_2 + 1 + r),(i_1 +1 + r,i_2 -r),(i_1 +1 + r,i_2 + 1+ r), which are the corners of the square. The tent roof is subsequently formed by linear interpolation between the base square and the square formed by the nodes in the interpolation area of the operating point (i_1,i_2),(i_1,i_2+1),(i_1+1,i_2),(i_1+1,i_2+1), and all the nodes within this roof are given artificial samples, with values given by the tent roof in the coordinate of the actualized LMs.

[0083] **LLNFM**. Implementing the algorithm on a one dimensional LLNFM is very similar to the case of a look-up table given above. The tent roof is formed by linear interpolation between LM $\hat{y}^{(i)}$ and LMs $\hat{y}^{(i-r)}$, $\hat{y}^{(i+r)}$. Note that the tent roof is in general asymmetric with respect to $\hat{y}^{(i)}$ due to the non-uniform distribution of the LMs, i.e.$|x^{(i)} - x^{(i-r)}| \neq |x^{(i)} - x^{(i+r)}|$.

[0084] If the LLNFM is of two dimensions, a simple solution is to form a tent base with the distance r irrespective of

the LMs, i.e. $x^{\left(\begin{bmatrix} i_1 \\ i_2 \end{bmatrix}\right)} \pm \begin{bmatrix} r \\ j \end{bmatrix}, x^{\left(\begin{bmatrix} i_1 \\ i_2 \end{bmatrix}\right)} \pm \begin{bmatrix} j \\ r \end{bmatrix}, j \in \begin{bmatrix} -r & +r \end{bmatrix}$, and actualize all LM within the tent base.

[0085] A more sophisticated solution is to form the tent base around the operating point by first identifying all LLM with a validity function equal to a threshold in the coordinated $x^{(i)}$ of the LM where the operating point lies, i.e. ∇j, *where* $\Phi_j(x^{(i)}) = \Phi_{tentbase}$. The tent base is thereafter formed by drawing straight lines between the coordinates of all the identified LM *j* which surround LM *i*. The values on the tent base needed for the tent roof can be read directly from the value of the map in the needed coordinate. All local models / within this tent base, i.e. $\forall l \neq i$, *where* $\Phi_l(x^{(i)}) \geq \Phi_{tentBase}$, are subsequently actualized analogously to the case of two dimensional look-up tables. Hence the radius of the tent base is determined by $\Phi_{tentBase}$ instead of by r. Moreover the tent base has not the form of a square but of a polygon, due to the non-uniform distribution of the local models.

4.2 Local Pattern Regression Models

[0086] The first subsection will give the basic version of the algorithm and the second will discuss various extensions to it.

4.2.1 Basic Version

[0087] In this algorithm there exists a local straight line model in every transition between two adjacent LMs (equation (45)), these are referred to as *regression models.* They give an estimated relationship between the values of two neighboring LMs. In this actualization algorithm the values of the artificial samples are given by these local pattern regression models. If e.g. an artificial sample is to be generated in LM *i*+1 the value of it is estimated from the value of a neighboring LM e.g. *i*, with the regression model between the two of them. Note that equation (45) is independent of the input coordinate x of the map, merely the value of LM i determines the value of the artificial sample in LM *i*+1. Moreover, there is a unique regression model in both directions between every adjacent LM, i.e. $\left(\beta_0^{(i,i+1)}, \beta_1^{(i,i+1)}\right)$ and $\left(\beta_0^{(i+1,i)}, \beta_1^{(i+1,i)}\right)$. The estimated value of $\hat{y}^{(i+1)}$ in equation (45) is denoted with a capital letter to distinguish that it is an estimation.

$$Y^{(i+1)} = \beta_0^{(i,i+1)} + \beta_1^{(i,i+1)} \hat{y}^{(i)} \tag{45}$$

[0088] The basic version of the algorithm starts by updating the operating point $\hat{y}_{k+1}(\xi)$which is between say local models *i* and *i*+1. Thereafter artificial samples are created in local models *i*-1 and *i*+2 and the update algorithm is applied on these samples. Subsequently artificial samples are created in local models *i*-2 and *i*+3 from the levels of local models *i*-1 and *i*+2 respectively. This continues until the whole map is adapted. The basic version of the *local pattern regression models* (LPRM) algorithm, implemented on a one-dimensional look-up table is summarized in an iterative form in Algorithm 4.3.

[0089] The individual regression models are defined by two parameters; rakeβ_1 and bias β_0. With these, different patterns between the two belonging local models can be represented. A change of the height in LM *i* can for example result in a large or a small height change in LM *i*+1. It is even possible to have a negative rake parameter, which results in a height increase in *i*+1 when the height of *i* decreases. With this discussion in mind and remembering that each transition has a unique regression model, one may conclude that these simple regression models can give complex patterns in the actualization of the map. This conclusion is verified by the simulations in Section 5.

Algorithm 4.3: Basic Local Pattern Regression Models

[0090]

$\hat{y}_{k+1} = \mathcal{U}\{\hat{y}_k, y_k(\xi)\}$ / Update operating point

$loop\ j = i+1...M-1$ / All local models right of the operating point

${}^{a}y_{k+1}^{(j+1)} = \beta_0^{(j,j+1)} + \beta_1^{(j,j+1)}\hat{y}_{k+1}^{(j)}$ / Create artificial sample

$\hat{y}_{k+1} = \mathcal{U}\{\hat{y}_k, {}^{a}y_{k+1}^{(j+1)}\}$ / Update actualized local model

end / right models

$loop\ j = i...2$ / All local models left of the operating point

${}^{a}y_{k+1}^{(j-1)} = \beta_0^{(j,j-1)} + \beta_1^{(j,j-1)}\hat{y}_{k+1}^{(j)}$ / Create artificial sample

$\hat{y}_{k+1} = \mathcal{U}\{\hat{y}_k, {}^{a}y_{k+1}^{(j-1)}\}$ / Update actualized local model

end / left models

[0091] Figures 5a -5e illustrate an example of LPRM, artificial samples indicated by (`), a measurement sample by (∗), an estimated map before adaptation by (-·-), and an estimated map after adaptation by (__). A simple example of the algorithm with the DA used as update algorithm is illustrated in Figure 5a 5e. The example starts with the map receiving a measurement sample y between LM 1 and LM 2 in Fig. 5a. These samples are subsequently updated with the DA algorithm. Thereafter an artificial sample ${}^{a}y^{(3)}$ is created in LM 3 with the value given by a regression model that estimates the value in LM3 from LM2 (regression model 2-3), as shown in Fig. 5b and LM 3 is accordingly updated with respect to the artificial sample, as shown in Fig.5c. The same procedure follows by forming an artificial sample in LM 3 based on the level of LM 2 and their intermediate regression model. Fig. 5d shows an estimation of an artificial sample in LM 4 from LM 3. In Fig 5e, LM 4 receives artificial sample.

4.2.2 Extended Version

[0092] In this section three extensions to the basic version are presented. The first is a method which weighs the artificial samples according to their uncertainty; the second extension is a way to limit the actualization to areas which have not received real samples within a predetermined time; while the third extension is a method to adapt the regression models online. Some extensions and associated problems assume that RLS is used as update algorithm, thus the RLS will be the standard update algorithm in this subsection. In the end of the subsection different modifications due to map representation and map dimension will be discussed.

[0093] **Weighing artificial samples**. One problem with the basic version is that the uncertainty of the values of the generated artificial samples increases with the distance from the operating point. This is because the generated samples are based on estimations from linear regression models and these are associated with an uncertainty, i.e. their confidence

intervals.

[0094] Here it is assumed that the training data used for the optimization is normally distributed and mean is equal to the regression model. This implies that the confidence intervals follow the Student-t distribution. This is described in Milton, J.S. and Arnold, J.C. (1995). Introduction to Probability and Statistics - *Principles and Applications for Engineering and the Computing Sciences,* hereinafter referred to as [Milton and Arnold, 1995]. Equations (46)-(48) [Milton and Arnold, 1995] gives the confidence interval $conf_{i,i+1}$ when an artificial sample is generated in LM *i*+1 and estimated from the level of LM *i*. The parameter $t_{\alpha/2}$ is given by the student-t distribution with respect to the number of samples *N* and the degree of confidence100(1-α). The choice of the degree of confidence is arbitrary because the confidence intervals will be transformed, which is described below.

$$conf_{i,i+1} = t_{\alpha/2} S \sqrt{1 + \frac{1}{N} + \frac{(y^{(i)} - \overline{y}^{(i)})^2}{S_{y^{(i)}y^{(i)}}}} \tag{46}$$

$$S_{y^{(i)}y^{(i)}} = \sum_{j=1}^{N} (y_j^{(i)} - \overline{y}_j^{(i)})^2 \tag{47}$$

$$S = \sqrt{\frac{\sum_{j=1}^{N} (\beta_0^{(i,i+1)} + \beta_1^{(i,i+1)} y_j^{(i)} - y_{j+1}^{(i)})^2}{N-2}} \tag{48}$$

[0095] This uncertainty can be taken into consideration by making use of the possibility of weighing the samples in the RLS-algorithm. The sizes of the weights are based on the sum of the confidence intervals of the used regression models between the artificial sample and the operating point. The values of the weights *w* used in the update must have the relationship to the sum of confidence intervals *confSum* given in conditions (49) below. The sum of confidence intervals *confSum* is formed by summing all confidence intervals associated with every estimated artificial sample from the operating point in LM 1 to the latest artificial sample in LM *i*+1, according to equation (50). This transformation is done with a decreasing exponential function according to equation (51).

$$\begin{aligned} w &\to 0, \quad when \quad confSum \to \infty \\ w &\to 1, \quad when \quad confSum \to 0 \end{aligned} \tag{49}$$

$$confSum_{1\ldots i+1} = conf_{i,i+1} + conf_{i-1,i} + \ldots + conf_{1,2} \tag{50}$$

$$w^{(i,i+1)} = \exp\{-\delta \times confSum_{1\ldots i+1}\} \tag{51}$$

[0096] The choice of the exponential function is based on the following arguments. The function is a well known and an easily predictable function; it will quickly converge to zero when *confSum* begins to grow and this convergence rate is easily determined by a constant δ. Other choices of transformations are naturally possible.

[0097] The above solution requires N sample pairs stored in each transition. This extra memory requirement can be

omitted if the regression models are not reestimated online (which is described below) or the accuracy of the confidence intervals is of negligible importance. Then the confidence intervals can be given a fixed value.

[0098] There is no use to update areas when the sum of confidence intervals has grown to a point when the weight has become so small that the effect of the artificial samples is insignificant. So a restriction $w > w_{limit}$ is included in the algorithm; partly because the artificial samples makes the RLS update to forget older, more significant samples, and replace them with highly uncertain artificial samples and partly to save computation time.

[0099] **Restricting actualization**. Another problem with the basic version is that it generates artificial samples to areas which have recently been updated by real measurement samples. It must be clear that the artificial samples are only an estimate of the measured map; the ideal situation is to receive real samples in all LMs regularly. Hence the spreading of artificial samples should be restricted to areas that have not been updated with real measurement samples during a predetermined time T_{act}. Consequently it is necessary to measure the time $K^{(i)}$ since the last time the separate local models were updated with a measurement sample. The spreading cycle should neither continue on the other side of a newly updated LM, because the newly updated LM has recently actualized those areas with more accurate artificial samples. Therefore the actualization cycle is stopped when a newly updated local model is reached. Here a central definition must be stated. Two types of changes that occur in the measured map can be distinguished:

- Expected changes; the map varies in the way that is expected by the regression models $y_k^{(i)} = Y^{(i)}$, (probably more frequent).
- Structural changes; the map varies in a new and unknown way, *contrary to the estimation of the regression models* $y_k^{(i)} \neq Y^{(i)}$.

[0100] The parameter T_{act} is determined by minimizing the error of the map $e_{model} = |y^{(i)} - y^{(i)}|$. When artificial samples are not generated in LM *i* the local approximate maximum error in the LM is reached just before actualization is allowed, i.e. $e_{avg}^{(i)} = \dot{y}_{avg} T_{act}$, where $\dot{y}_{avg}$ is the average change rate of the map. On the other hand when artificial samples are generated in local model *i* the error converges to the error in the regression model $e_{reg}^{(i)} = \left| y^{(i)} - Y^{(i)} \right|$. The error in the regression model depends e.g. on the accuracy of the offline optimization, the number of samples *N* used for the regression, the rate of structural changes in the map, and the frequency of the optional online re-estimation of the regression models, which will be described below. Thus the optimization of parameter T_{act} is complex and depends on many uncertain parameters. Summarizing the trade-off; if $\dot{y}_{avg}$ is high T_{act} should be small and if the structural change rate is believed to be high or the regression models are uncertain in any way, T_{act} should be large.

[0101] Due to the forgetting factor λ in the RLS update algorithm, real measurement samples will be forgotten when artificial samples are spread to the local model, which starts when $K^{(i)} > T_{act}$. Therefore λ may be given a higher value $\lambda=\lambda_{act}$ when it is acting on artificial samples so that real measurements won't be forgotten too quickly.

[0102] **Regression model adaptation**. It is possible to adapt the regression models online, so that they can keep track of structural changes in the map. This is done by storing *N* pairs of heights of adjacent local models, where the levels are measured at the same time. How these height pairs are collected is described below. By replacing older stored sample pairs with new ones, the local pattern regression model can be re-estimated.

The theory of estimating linear regression models is based on the Gauss-Markov assumptions, which are given below.

- The random residuals ε_i have expected value 0
- They are independent
- They are normally distributed
- They all have the same variance, i.e. homogeneous variance

[0103] Here only simple straight-line linear regression models have been used, though it is possible to create more complex regression models. The rationale for this is that two adjacent local models ought to vary in a similar way and to keep the model as simple as possible and to minimize memory and computational requirements. However if the variation pattern diverges from a linear pattern it will result in a biased estimation of the model parameters. This is described in Raw lings, J.O., Pantula, S.G., and Dickey, D.A. (1998). Applied Regression Analysis - A Research Too/.

Springer-Verlag New York, Inc, hereinafter referred to as [Rawlings et al., 1998].

[0104] Structural changes in the map will have the effect that none of the Gauss-Markov assumptions are valid except normality. This could be interpreted that older measurements are more uncertain than newer ones i.e. heterogeneous variance $\text{var}(\varepsilon_i) \neq \text{var}(\varepsilon_j)$. The negative effect on the estimation caused by heterogeneous variance can be reduced by using weighted least-squares in the regression model estimation [Rawlings et al., 1998]. The weights are set so that newer samples have bigger impact on the estimation than the older ones. If the rate of structural changes is fast the weight difference should be greater, on the other hand if they are slow the weight difference can be set smaller which will give the older values greater influence on the model estimation. This weight decrease is solved by the "weight decrease" algorithm given in the Appendix.

[0105] In the basic least-squares estimation, outliers have a big leverage on the values of the model parameters, because the estimation is based on squared errors. This could have the effect that one or a few less accurate measurements could severely distort the estimation. But the outliers should not completely be discarded because they could be the result of structural changes. This problem is solved by the introduction of a limit on the distance an outlier is permitted to be from the expected value given by the regression model. If an outlier is measured outside this limit, it is discarded and an artificial measurement will be placed on the current limit. Furthermore maximum and minimum values must be set on the parameters of the regression models or by forming formal inequality constraints on the output of the regression models, to ensure stability.

[0106] In the case of non-normality only the estimation of the confidence intervals will be affected while the parameter estimates are unaffected. Moreover normal distribution is a reasonable assumption in most cases [Rawlings et al., 1998].

[0107] The number of saved samples *N* used for the estimation of the regression models is determined by the following pros and cons. The major benefits with many samples are robustness against noise and more accurate models, while the benefits with few samples are smaller memory requirements and faster adaptation to structural changes.

[0108] **Collecting new samples for the regression models**. Note first that the samples used for the estimation of the regression models are the heights of the LMs, not the incoming measurement samples. Samples for the re-estimation of the regression models are only collected if two adjacent local areas are updated with real samples within a predetermined period $T_{collect}$.

[0109] Otherwise changes may occur in the map during the time between the two samples which leads to incorrect estimation. Thus an upper limit of how fast the map changes, "maximum change rate", must be determined before implementation, which is defined by equation (52).

$$\dot{y}_{\max} \equiv \max_{x,t} \{ | \frac{\partial y_t(x)}{\partial t} | \} \tag{52}$$

[0110] With this value known and a maximum error tolerance $e_{collect}$ specified, the time limit $T_{collect}$ between the two samples can be determined. For clarification; the error tolerance refers here to the error in the collected sample pair due to variation in the map between the two measurements, which is defined by equation (53). By setting the change rate to $\dot{y}_{\max}$ and integrating the time period $T_{collect}$ in equation (54) follows.

$$\int_T | \dot{y}_t(x) | \, dt = e \tag{53}$$

$$T_{collect} = \frac{e_{collect}}{\dot{y}_{\max}} \tag{54}$$

[0111] The essence of this is that maps with high change rates must have small $T_{collect}$ and for maps with low change rates the time limit can be set to a higher value. Thus maps with high change rates and low sampling frequency will seldom reestimate their regression models while maps with low change rates and high sampling frequency will reestimate their regression models more often. To avoid the occurrence of too high change rate and thus large errors in the regression models a restriction is included in the extended Algorithm 4.4, under task 1. Another cause of the same error is actualization between the two collected pairs which can occur if $T_{act} < T_{collect}$.

[0112] Another problem that can lead to incorrect model estimation, when RLS is used, is if one of the local models

$\hat{y}^{(i)}$ has been updated frequently and its value has converged to values of the samples, while the adjacent model $\hat{y}^{(i-1)}$ has just been updated once during a long period. This single sample may not change the height of the local model near the correct level because of the inertia in the RLS-algorithm. Note that this problem does not exist when DA is used as update algorithm. This is solved by keeping track of the time $K^{(i-1)}$ since a measurement was received in the LM and if enough time has passed since the last update one can assume that the latest measurement is much more relevant than the old ones. Thus artificial samples should be created so that the bias level will be adjusted to the proximity of the newest measurement. Thereafter the regression model can be re-estimated. See Algorithm 4.4 under task 3 for details. When the updating is done of these artificial samples the forgetting factor can advantageously be given a smaller value so that the old samples are forgotten and hence the next incoming sample won't be dominated by the recently generated artificial samples.

[0113] A third problem that can lead to incorrect estimation of the regression models due to how the samples are collected is if the measurement error is high. This is especially a problem when estimating models where the measurements occur seldom.

[0114] It is important how the old samples are replaced by the new ones. If the old samples are replaced in an aged order there is a high risk that the samples will cluster in a small region, which will lead to an uncertain estimation in regions outside the cluster. To overcome this problem the data used for the estimation should be present in a wide range of the movement of the bias levels. The pseudo code for the algorithm can be found in the Appendix.

Algorithm 4.4: *Extended Local Pattern Regression Models*

[0115]

1. Start, check change rate: One measurement sample $y_k(\xi)$ is received between LMs i and i+1. If the change rate of the map is too high readjust the incoming sample to an acceptable value.

$$if\ \left|y(\xi)-\hat{y}(\xi)\right| > \left(K^{(i)}\frac{(c_{i+1}-x)}{c_{i+1}-c_i} + K^{(i+1)}\frac{(x-c_i)}{c_{i+1}-c_i}\right)\dot{y}_{\max} \text{ / Too high change rate}$$

$$y(\xi)=\hat{y}(\xi)\pm\left(K^{(i)}\frac{(c_{i+1}-x)}{c_{i+1}-c_i} + K^{(i+1)}\frac{(x-c_i)}{c_{i+1}-c_i}\right)\dot{y}_{\max}$$

end

2. Update the current LMs $\hat{y}_{k+1} = U\{\hat{y}_k, y_k(\xi)\}$.

3. Readjust old LM (not used with DA): If the current LMs have not been updated for a long time, make sure that the map is sufficiently close to the latest measurement.

$$if\ \dot{y}_{avg}\left(K^{(i)}\frac{(c_{i+1}-x)}{c_{i+1}-c_i}+K^{(i+1)}\frac{(x-c_i)}{c_{i+1}-c_i}\right)>\varepsilon_{old}$$ / If enough time has passed

$$while\ \varepsilon_{old}<\left|y(\xi)-\hat{y}(\xi)\right|$$

$$\hat{y}_{k+1}=\mathcal{U}\{\hat{y}_k,y(\xi)\}$$ / Use λ_{old} in the update

if (no change in θ *was done)*

break; / To avoid infinite loop due to dead-zone in RLS

end

end

end

4. Check requirements for re-estimation: If the requirements for adaptation of any of the belonging regression models should be adjusted. Requirements:

a) If any of the current LM $\hat{y}^{(i)}$, $\hat{y}^{(i+1)}$ neighboring LMs $\hat{y}^{(i-1)}$, $\hat{y}^{(i+2)}$ have been updated with measurement samples within time window $T_{collect}$.
b) No non-steady state situation between the two measurements has occurred.

5. Re-estimate regression models: If the requirements in step four are fulfilled for LM i-1 or LM i+2, save the values of the two adjacent LMs and re-estimate the two regression models between them. Assume LM i+2 fulfills the requirements in 3.

a) If any of the two bias levels is an outlier, e.g. $|\bar{Y}^{(i+2)}_{|\hat{y}_{i+1}}-\hat{y}^{(i+2)}|>outlierLimit$, adjust the value of it/them to an acceptable value. Do not use the adjusted sample for updating the map later, but use it for estimation the regression model.
b) Save these values for the estimation of the regression models between the two LMs and discard an old sample pair with the replacement algorithm given in appendix.
c) Adjust the weight matrix for the estimation of the regression model so that newer samples have greater leverage in the LS-algorithm. Use the algorithm given in appendix.
d) Estimate the two regression models in both directions. Note that the statistics S_{yy} ,S and, $\bar{y}$ change their values due to the re-estimation. Ensure that the regression model parameters are within their allowed values.

6. *Create artificial samples:* Spread artificial samples from LM i and i+1 with recursive functions to all the adjacent LMs and beyond. The values of the samples are given by estimation from the bias level of LM i and i+1 and their respective regression models. The samples are placed in the coordinates of the LMs. The actualization is given below in an iterative form.

$loop\ j = i+1...M-1$ / All local models right of the operating point

$${}^{a}y_{k+1}^{(j+1)} = \beta_0^{(j,j+1)} + \beta_1^{(j,j+1)}\hat{y}_{k+1}^{(j)}$$ / Create artificial sample

$$w^{(j,j+1)} = \exp\left\{-\delta \times confSum_{i+1\,..\,j+1}\right\}$$ / Calculate weight

(Used in RLS update)

$$\hat{y}_{k+1} = \mathcal{U}\{\hat{y}_k, {}^{a}y_{k+1}^{(j+1)}\}$$ / Update actualized local model

end / right models

$loop\ j = i...2$ / All local models left of the operating point

$${}^{a}y_{k+1}^{(j-1)} = \beta_0^{(j,j-1)} + \beta_1^{(j,j-1)}\hat{y}_{k+1}^{(j)}$$ / Create artificial sample

$$w^{(j,j-1)} = \exp\left\{-\delta \times confSum_{i\ldots j-1}\right\}$$ / Calculate weight

(Used in RLS update)

$$\hat{y}_{k+1} = \mathcal{U}\{\hat{y}_k, {}^{a}y_{k+1}^{(j-1)}\}$$ / Update actualized local model

end / left models

The recursive functions will continue to spread artificial samples beyond the adjacent LMs until some restriction is fulfilled, when that happens the recursive function will stop and will not continue beyond the LM where the restriction was met. The restrictions are given below:

a) If the end of the map is reached.
b) If a LM recently has been updated with a real sample, i.e. $K^{(j)} < T_{act}$.
c) If the weight $w^{(j,j-1)}$, which is based on the sum of the confidence intervals, has diminished to an insignificant value.

7. Wait: Here the algorithm has completed its cycle and will wait for the next incoming measurement sample.

Summary of parameter settings

[0116]

λ	Forgetting factor used for updating measurement samples (RLS)
λ_{art}	Forgetting factor used for updating artificial samples from regression models (larger) (RLS)
λ_{old}	Forgetting factor used for updating measurement samples due to $\dot{y}_{avg}\, K^{(i)} > \varepsilon_{old}$ (smaller) (RLS)
δ	Determines the convergence rate of the weights (RLS)
w_{limit}	Minimum weight for continued actualization (RLS)
T_{act}	Time limit of how early it is permitted to actualize since the latest received measurement in the current LM
$T_{collect}$	Maximum time limit between two collected samples for reestimating regression models
$\dot{y}_{avg}$	Probable average change rate of the map (RLS)
$\dot{y}_{\max}$	Highest tolerated change rate
ε_{old}	Highest tolerated residual between not recently updated LM and new measurement sample (RLS)
$\beta_{0_{\max}}, \beta_{1_{\max}}$	Highest allowed values on the regression parameters
$\beta_{0_{\min}}, \beta_{1_{\min}}$	Smallest allowed values on the regression parameters
outlierLimit	Outlier limit for regression model reestmation
N	Number of stored samples used for estimation of the regression models

[0117] **Modifications**. If the algorithm is applied on two dimensional look-up tables it starts by creating a public variable in the form of a matrix *A* which keeps track of which LMs that are allowed to be actualized. Therefore each time a LM is updated it changes its corresponding *A* -value to forbidden. The *A* -matrix is initialized before each actualization cycle so that the actualization is it stops at newly updated LMs (j_1,j_2), when $K^{(j_1,j_2)} < T_{collect}$. Moreover a boundary could be formed with respect to the LM of the operating point (i_1,i_2) and a newly updated LM (j_1,j_2) so that no actualization is done beyond this. Figures 6a-6b show example of possible boundaries for permitted actualization in two dimensional look-up tables, empty nodes start point of actualization, wherein filled nodes represent $K^{(j_1,j_2)} \geq T_{act}$, and striped nodes represent $K^{(j_1,j_2)} < T_{act}$. In Figure 6a a boundary is formed by cross section of the map, while Fig. 6b shows triangular shaped boundaries. Figure 6a shows an example of this where the boundary is formed by stopping actualization passed the cross section of the map with respect of the newly updated LM. Other geometrical boundaries are possible, e.g. forming a triangular shape from (j_1, j_2) with respect to (i_1,i_2), as shown in Figure 6b.

[0118] The actualization is done by initializing actualization cycles from the local models (i_1, i_2), $(i_1, i_2 +1)$, $(i_1 +1, i_2)$, $(i_1 +1, i_2 +1)$, which are associated with the interpolation area of the operating point. Algorithm 4.5 provides the actualization cycles in an iterative form. Figure 7 depicts an example of the actualization procedure in a two dimensional look-up table. Otherwise the algorithm is straightforwardly derived from the one dimensional case.

Algorithm 4.5: *2-D (look-up table) LPRM*

/ Actualization cycle from LM (i_1,i_2)

[0119]

$loop\ j_1 = i_1 ... 2$ / Leftward

$$ {}^{a}y_{k+1}^{(j_1+1,i_2)} = \beta_0^{((j_1,i_2),(j_1+1,i_2))} + \beta_1^{((j_1,i_2),(j_1+1,i_2))} \hat{y}_{k+1}^{(j_1,i_2)} $$

$$ \hat{y}_{k+1} = \mathcal{U}\{\hat{y}_k, {}^{a}y_{k+1}^{(j_1+1,i_2)}\} $$

$loop\ j_2 = i_2 ... 2$ / Downward

if $|i_2 - j_2| > |i_1 - j_1|$

break;

end

$$ {}^{a}y_{k+1}^{(j_1,j_2-1)} = \beta_0^{((j_1,j_2),(j_1,j_2-1))} + \beta_1^{((j_1,j_2+1),(j_1,j_2-1))} \hat{y}_{k+1}^{(j_1,j_2)} $$

$$ \hat{y}_{k+1} = \mathcal{U}\{\hat{y}_k, {}^{a}y_{k+1}^{(j_1,j_2-1)}\} $$

end

end

$loop\ j_2 = i_2 ... 2$ / Downward

$$ {}^{a}y_{k+1}^{(i_1,j_2-1)} = \beta_0^{((i_1,j_2),(i_1,j_2-1))} + \beta_1^{((i_1,j_2),(i_1,j_2-1))} \hat{y}_{k+1}^{(i_1,j_2)} $$

$$ \hat{y}_{k+1} = \mathcal{U}\{\hat{y}_k, {}^{a}y_{k+1}^{(i_1,j_2-1)}\} $$

$loop\ j_1 = i_1 ... 2$ / Leftward

if $|i_1 - j_1| > |i_2 - j_2| + 1$

break;

end

$$ {}^{a}y_{k+1}^{(j_1-1,j_2)} = \beta_0^{((j_1,j_2),(j_1-1,j_2))} + \beta_1^{((j_1,j_2),(j_1-1,j_2))} \hat{y}_{k+1}^{(j_1,j_2)} $$

$$ \hat{y}_{k+1} = \mathcal{U}\{\hat{y}_k, {}^{a}y_{k+1}^{(j_1-1,j_2)}\} $$

end

end

/ **Actualization cycle from LM** (i_1, i_2 +1)

[0120]

$loop\ j_2 = i_2 + 1 ... M_2 - 1$ / Upward

$$ {}^{a}y_{k+1}^{(i_1,j_2+1)} = \beta_0^{((i_1,j_2),(i_1,j_2+1))} + \beta_1^{((i_1,j_2),(i_1,j_2+1))} \hat{y}_{k+1}^{(i_1,j_2)} $$

$$ \hat{\mathbf{y}}_{k+1} = \mathcal{U}\{\hat{\mathbf{y}}_k, {}^{a}y_{k+1}^{(i_1,j_2+1)}\} $$

$loop\ j_1 = i_1 ... 2$ / Leftward

if $|i_1 - j_1| > |i_2 + 1 - j_2| + 1$

break;

end

$$ {}^{a}y_{k+1}^{(j_1-1,j_2)} = \beta_0^{((j_1,j_2),(j_1-1,j_2))} + \beta_1^{((j_1,j_2),(j_1-1,j_2))} \hat{y}_{k+1}^{(j_1,j_2)} $$

$$ \hat{\mathbf{y}}_{k+1} = \mathcal{U}\{\hat{\mathbf{y}}_k, {}^{a}y_{k+1}^{(j_1-1,j_2)}\} $$

end

end

$loop\ j_1 = i_1 ... 2$ / Leftward

$$ {}^{a}y_{k+1}^{(j_1+1,i_2+1)} = \beta_0^{((j_1,i_2+1),(j_1+1,i_2+1))} + \beta_1^{((j_1,i_2+1),(j_1+1,i_2+1))} \hat{y}_{k+1}^{(j_1,i_2+1)} $$

$$ \hat{\mathbf{y}}_{k+1} = \mathcal{U}\{\hat{\mathbf{y}}_k, {}^{a}y_{k+1}^{(j_1+1,i_2+1)}\} $$

$loop\ j_2 = i_2 + 1 ... M_2 - 1$ / Upward

if $|i_2 + 1 - j_2| > |i_1 - j_1|$

break;

end

$$ {}^{a}y_{k+1}^{(i_1,j_2+1)} = \beta_0^{((i_1,j_2),(i_1,j_2+1))} + \beta_1^{((i_1,j_2),(i_1,j_2+1))} \hat{y}_{k+1}^{(i_1,j_2)} $$

$$ \hat{\mathbf{y}}_{k+1} = \mathcal{U}\{\hat{\mathbf{y}}_k, {}^{a}y_{k+1}^{(i_1,j_2+1)}\} $$

end

end

/ **Actualization cycle from LM** (i_1 + 1, i_2)

[0121]

$loop\ j_2 = i_2 ... 2$ / Downward

$$ {}^{a}y_{k+1}^{(i_1+1,j_2-1)} = \beta_0^{((i_1+1,j_2),(i_1+1,j_2-1))} + \beta_1^{((i_1+1,j_2),(i_1+1,j_2-1))} \hat{y}_{k+1}^{(i_1+1,j_2)} $$

$$ \hat{\mathbf{y}}_{k+1} = \mathcal{U}\{\hat{\mathbf{y}}_k, {}^{a}y_{k+1}^{(i_1+1,j_2-1)}\} $$

$loop\ j_1 = i_1 + 1 ... M_1 - 1$ / Rightward

if $|i_1 + 1 - j_1| > |i_2 - j_2| + 1$

break;

end

$$ {}^{a}y_{k+1}^{(j_1+1,j_2)} = \beta_0^{((j_1,j_2),(j_1+1,j_2))} + \beta_1^{((j_1,j_2),(j_1+1,j_2))} \hat{y}_{k+1}^{(j_1,j_2)} $$

$$ \hat{\mathbf{y}}_{k+1} = \mathcal{U}\{\hat{\mathbf{y}}_k, {}^{a}y_{k+1}^{(j_1+1,j_2)}\} $$

end

end

$loop\ j_1 = i_1 + 1 ... M_1 - 1$ / Rightward

$$ {}^{a}y_{k+1}^{(j_1+1,i_2)} = \beta_0^{((j_1,i_2),(j_1+1,i_2))} + \beta_1^{((j_1,i_2),(j_1+1,i_2))} \hat{y}_{k+1}^{(j_1,i_2)} $$

$$ \hat{\mathbf{y}}_{k+1} = \mathcal{U}\{\hat{\mathbf{y}}_k, {}^{a}y_{k+1}^{(j_1+1,i_2)}\} $$

$loop\ j_2 = i_2 ... 2$ / Downward

if $|i_2 - j_2| > |i_1 + 1 - j_1|$

break;

end

$$ {}^{a}y_{k+1}^{(j_1,j_2-1)} = \beta_0^{((j_1,j_2),(j_1,j_2-1))} + \beta_1^{((j_1,j_2+1),(j_1,j_2-1))} \hat{y}_{k+1}^{(j_1,j_2)} $$

$$ \hat{\mathbf{y}}_{k+1} = \mathcal{U}\{\hat{\mathbf{y}}_k, {}^{a}y_{k+1}^{(j_1,j_2-1)}\} $$

end

end

/ **Actualization cycle from LM** (i_1 +1,i_2 +1)

[0122]

$loop\ j_2 = i_2 + 1 \ldots M_2 - 1$/ Upward

$${}^{a}y_{k+1}^{(i_1+1,j_2+1)} = \beta_0^{((i_1+1,j_2),(i_1+1,j_2+1))} + \beta_1^{((i_1+1,j_2),(i_1+1,j_2+1))}\hat{y}_{k+1}^{(i_1+1,j_2)}$$

$$\hat{y}_{k+1} = \mathcal{U}\{\hat{y}_k, {}^{a}y_{k+1}^{(i_1+1,j_2+1)}\}$$

$loop\ j_1 = i_1 + 1 \ldots M_1 - 1$/ Rightward

if $|i_1 + 1 - j_1| > |i_2 + 1 - j_2| + 1$

break;

end

$${}^{a}y_{k+1}^{(j_1+1,j_2)} = \beta_0^{((j_1,j_2),(j_1+1,j_2))} + \beta_1^{((j_1,j_2),(j_1+1,j_2))}\hat{y}_{k+1}^{(j_1,j_2)}$$

$$\hat{y}_{k+1} = \mathcal{U}\{\hat{y}_k, {}^{a}y_{k+1}^{(j_1+1,j_2)}\}$$

end

end

$loop\ j_1 = i_1 + 1 \ldots M_1 - 1$/ Rightward

$${}^{a}y_{k+1}^{(j_1+1,i_2+1)} = \beta_0^{((j_1,i_2+1),(j_1+1,i_2+1))} + \beta_1^{((j_1,i_2+1),(j_1+1,i_2+1))}\hat{y}_{k+1}^{(j_1,i_2+1)}$$

$$\hat{y}_{k+1} = \mathcal{U}\{\hat{y}_k, {}^{a}y_{k+1}^{(j_1+1,i_2+1)}\}$$

$loop\ j_2 = i_2 + 1 \ldots M_2 - 1$/ Upward

if $|i_2 + 1 - j_2| > |i_1 + 1 - j_1|$

break;

end

$${}^{a}y_{k+1}^{(i_1,j_2+1)} = \beta_0^{((i_1,j_2),(i_1,j_2+1))} + \beta_1^{((i_1,j_2),(i_1,j_2+1))}\hat{y}_{k+1}^{(i_1,j_2)}$$

$$\hat{y}_{k+1} = \mathcal{U}\{\hat{y}_k, {}^{a}y_{k+1}^{(i_1,j_2+1)}\}$$

end

end

[0123] Figure 7 shows an actualization cycle of LPRM on 2-D look-up tables, with LMs within the interpolation area of the operating point, indicated by circles (o), an outer loop, indicated by filled arrows (__), and an inner loop, indicated by dashed arrows (_._).

[0124] If the algorithm is applied on one dimensional LLNFM no major modifications are needed. Note however that in the two dimensional case there are generally many transitions between adjacent LM in each dimension, due to the

non uniform distribution of the LLM. The actualization cycle can be implemented similar to Algorithm 4.5, but here the cycle starts from one LM instead of four as in the case of look-up tables.

4.2.3 Memory Requirements

[0125] The additional memory requirements needed when the *local pattern regression models* algorithm is used is analyzed here. The analysis is merely done with look-up tables used as map representation. This is because the number of transitions between adjacent LM is arbitrary in two and higher dimensional LLNFM, while the number of transitions are always twice in each dimension (non-border LMs) in look-up tables. Furthermore the small number of constants which doesn't depend on the number of local models in omitted.

[0126] The memory requirements for the basic version of the algorithm are two parameters β_0,β_1 for each transition between every neighboring LMs. Note that there is one regression model in each direction in every transition, but because the regression models are easily invertible it is sufficient to save one of them. How they are inverted is shown in Algorithm 4.6 below. When the algorithm is applied to one-dimensional look-up tables, the number of transitions adds up to (M-1) and thus 2(M-1) parameters are stored in memory. In two dimensional look-up tables the number of transitions in the first dimension is (M_1-1) M_2 and in the other dimension M_1 (M_2-1), thus the total number of stored parameters is 2 (M_1 (M_2-1) + (M_1-1) M_2).

[0127] The extended version needs some additional stored parameters. First every LM needs to keep track of the time since they last received a measurement sample $K^{(i)}$, thus M additional variables in one dimensional look-up tables and M_1 M_2 in two dimensional tables. When online adaptation of the regression models are incorporated in the algorithm it needs to store 2N samples for each transition, which sums up to 2N(M-1) in one dimensional maps and 2N(M_1 (M_2-1) + (M_1-1) M_2) in two dimensional case. The total number of stored variables and constants is given below. In the extended version the regression model parameters can be calculated each time they are needed and thereby saving memory at the expense of computation time. The expressions below give the minimum memory requirement.

[0128] Basic version for one dimensional look-up tables:

- 2(M-1)

[0129] Extended version for one dimensional look-up tables:

- 2N(M-1) + M

[0130] Basic version for two dimensional look-up tables:

- 2(M_1(M_2-1) + (M_1-1) M_2)

[0131] Extended version for two dimensional look-up tables:

- $2N(M_1 (M_2 -1) + (M_1 -1) M_2) + M_1 M_2 = 2N(2 M_1M_2 - M_1 - M_2) + M_1 M_2$

4.2.4 Offline Optimization

[0132] The initial optimization of the regression models is done by generating samples from two maps, which is the a-priori information for the regression model parameters. These are referred to as the first and second boundary maps $y_{boundary1}$ (X), $y_{boundary2}$(X), They are supposed to enclose a maximum probable variation interval (max ($y_{0\le t<\infty}(x)$),min ($y_{0\le t<\infty}(x)$)), that the map will have during operation. These maps might be given by the start map and a probable future map measured from e.g. a used engine or just a qualified guess. It is important that the boundary maps approximately enclose the variation interval. Otherwise the accuracy of the artificial samples decrease, which is reflected by larger confidence intervals. This occurs when the heights of the local models are far from the average bias level $\bar{y}$ of the N samples used for estimation of the regression models. Another consequence of bad placement of the boundary maps is that the replacement algorithm used in the re-estimation of the regression models may only replace one of a few of the stored samples. Thus it is important that the boundary maps closely demarcate the interval of the bias variations of the map. If one of the boundary maps is known to be near the center of the probable variation interval of the map, it should be moved to the probable boundary of the interval. In case this is done the details in the map should be linearly extrapolated with respect to the other map.

[0133] The offline optimization algorithm below generates samples by linear interpolation between the two boundary maps. Each local model is given N samples and these are of course placed in the center of the local models $x^{(i)}$.

Algorithm 4.6: Offline Beta Optimization

[0134]

$$for\ v = 1...N$$

$$for\ i = 1...M$$

$$samples(i,v) = \frac{y_{boundary2}(x^{(i)})v + y_{boundary1}(x^{(i)})(N-v)}{N}$$

$$end$$

$$end$$

$$for\ i = 1...M-1$$

$$\begin{bmatrix} \beta_0^{(i,i+1)} \\ \beta_1^{(i,i+1)} \end{bmatrix} = (X^T Q X)^{-1} X^T Q\, samples(i+1,:)^T, \quad X = \begin{bmatrix} 1 & samples(i,1) \\ 1 & samples(i,2) \\ \dots & \dots \\ 1 & samples(i,N) \end{bmatrix}, \quad Q = I$$

$$\beta_0^{(i+1,i)} = -\frac{\beta_0^{(i,i+1)}}{\beta_1^{(i,i+1)}}$$

$$\beta_1^{(i+1,i)} = \frac{1}{\beta_1^{(i,i+1)}}$$

$$end$$

5 - SIMULATION

[0135] The purpose of the simulations described below is to explain how the algorithms work, detect weaknesses and strengths of different algorithms, and to develop general strategies for designing adaptive maps. The chapter consists of three simulations studies, where each Example consists of a few simulations. Example 1 will demonstrate some fundamentals of the nature of LPRMs. Example 2 will evaluate and demonstrate the nature of different adaptive strategies on real world engine maps, and Example 3 evaluates adaptive strategies on drive cycles on real engine maps with different levels of noise.

5.1 Simulation - Example 1

[0136] This example will demonstrate the capacity of LPRM to actualize different patterns. The Example is implemented on LLNFM with six LLMs and the update is done with RLS. Three simulations are done, where the second boundary map $Y_{boundary2}$ is different in each simulation; otherwise they all share the same properties. The simulations were done

by placing 20 equal measurement samples (x = 0.2, y = 14) close to the second boundary map y_{init2} and within LM 2. Thus all changes outside LM 2 are done by actualization. Figure 8a shows the common initial map for all three simulations with $Y_{boundary1}$ =sin(2πx)+10 and the map before the simulation. The estimated start map is indicated with a full line (___) and the boundary map by a dashed line (_._).

[0137] Figure 8b illustrates the end of the first simulation where the second boundary map differs from the first boundary map by having a larger bias term. In Figure 8b the final map uses $y_{boundary2}$ =sin(2πx)+13 wherein the estimated map is indicated with a full line (___),the boundary maps by a dashed line (_._), the artificial samples by circles (o), and the measurement sample by an asterisk (∗). The result of the second simulation is given in Figure 8c, where the sine function in the second boundary map was multiplied with a larger factor. Figure 8c shows the final map with $y_{boundary2}$ =4sin (2π*x*)+10, wherein the estimated map is indicated with a full line (___),the boundary maps by a dashed line (_._), the artificial samples by circles (o), and the measurement sample by an asterisk (∗). The last simulation in Figure 8d shows a combination of the two preceding simulations with both a larger bias term and a larger factor multiplied on the sine function. Figure 8c shows the final map with $y_{boundary2}$ =3sin(2π*x*)+11, wherein the estimated map is indicated with a full line (-), the boundary maps by a dashed line (_._), the artificial samples by circles (o), and the measurement sample by an asterisk (∗).

[0138] Figures 8b-c contain two boundary maps, where the relatively lower amplitude map is referred to as boundary map 1 and the higher amplitude map is referred to as boundary map 2.

[0139] The simulations above show that the LPRM can actualize patterns, which estimates the initial maps with good accuracy. Bear in mind that all measurement samples in the simulations had the same value and coordinate. This ability to represent different patterns can be understood by remembering that each local pattern regression model has a set of two unique parameter values rake and bias. Note also that the values of the artificial samples can either decrease or increase when the measurement samples increase, see Figures 8b and 8c in input values larger than 0.5, where they increase in Figure 8b and decrease in Figure 8c. In Section 4 it was mentioned that the actualization should only change the value of the LMs, i.e. the appearance around the coordinate of the LMs should not be changed by the actualization. In the case of LLNFM the rake parameter should not be changed by the actualization. This phenomenon is clear in Figure 8c, where the actualized LLMs have preserved their rake parameter from the start map and only the bias parameter has changed. Compare the rake of the map around the artificial samples with boundary map 1.

5.2 Simulation - Example 2

[0140] The simulations in this example are done on engine maps from the control system of an IC engine. The original maps were of two dimensions and gave the control system an estimated value of volumetric efficiency, from a given engine speed and intake manifold pressure. Here they are projected to one dimensional maps, by giving the manifold pressure a fixed value. The goal of this simulation is to evaluate how different actualization algorithms perform and how they work.

[0141] The simulations need a drive cycle which should be a model of how the values of the measurement samples of engine speed could vary in a real world drive cycle. For the drive cycle to be appropriate it should fulfill some requirements. Firstly the speed has inertia, thus the driving cycle should incorporate some dynamics. Secondly the excitation should hover around an engine speed which is common during normal driving conditions and during a short time excite a higher engine speed, which could correspond to an acceleration phase. Thirdly more than one driving cycle should be generated, so that possible flaws can be detected.

[0142] The drive cycle model is realized by a moving average (MA) time series model, given in equation (55) and the engine speed is accordingly given in equation (56). The MA-process is driven by white normally distributed noise with variance 1. The model is designed to give a simple model, which fulfills the above mentioned requirements. The process reaches the higher engine speed by adding a constant to the input noise during a short time in the drive cycle. The drive cycle ends by disengaging the MA-process and giving the input a stable and low engine speed so that all cycles ends in the same position, for better visual comparison.

$$z_{MA}(k) = \sum_{i=1}^{6}(1.1-0.1i)z_{MA}(k-i)+0.2v(k)\,, \quad z_{MA}(k)=0, \forall\, k<0 \tag{55}$$

$$x(k) = 100 z_{MA}(k) + 2200 \tag{56}$$

[0143] The driving cycle consist of 500 measurement samples and between samples 50 and 70 the MA-process will be fed with an additional constant in its input noise, resulting in higher engine speed. An example of a realization of the driving cycle is given in Figure 9 and Figure 10 showing the histogram of the drive cycle realization in Figure 8a-d, with 40 containers of samples, which gives an approximate density function of the drive cycle.

[0144] The following simulations compare the TRT with the LPRM (with re-estimation of the regression models). No measurement noise will be added and the DA will be used as update algorithm. Two different maps will be tested in the simulations. The first map, "Map 1", is a perfect match with the boundary maps used as a priori-information, as indicated in Figures 11a-d and 12a-d. The other map, "Map 2", is an alteration of Map 1 which does not fully agree with the boundary maps, as indicated in Figures 13a-d and 14a-d.

[0145] During the first 300 samples the real map, from which the samples are generated, will rise from a level close to the lower boundary map to a level close to the higher boundary map. The map will then decrease to its initial position during the remaining 200 samples. The movement of the Map 1 is done by linear interpolation between the initial maps using 300 equidistant steps up and 200 equidistant steps down. The movement of Map 2 is done with the same method except that it will be altered from Map 1 by an addition of a sine function. The initial estimated map will have the same values as the lower boundary map. The parameter settings of the simulation are given in the Appendix.

[0146] Figures 11a-d to 14a-d show snapshots of the simulations, which are taken at the first sample k=1; during the short period of higher engine speed k=60; the highest point of the map k=300 and the last sample k=500.

[0147] Figure 11 a shows a snapshot at sample k=1 of simulation of Map 1 with TRT, estimated map indicated by a full line (__), the measured map indicated by a dashed line (---), the boundary maps indicated by dash-dotted lines (-·-), artificial samples indicated by circles (o), and the measurement sample indicated by an asterisk (∗).

[0148] Figure 11 b. Snapshot at sample k=60 of simulation of Map 1 with TRT, estimated map indicated by a full line (___), the measured map indicated by a dashed line (---), the boundary maps indicated by dash-dotted lines (-·-), artificial samples indicated by circles (o), and the measurement sample indicated by an asterisk (*).

[0149] Figure 11c. Snapshot at sample k=300 of simulation of Map 1 with TRT, estimated map indicated by a full line (___), the measured map indicated by a dashed line (---), the boundary maps indicated by dash-dotted lines (-·-), artificial samples indicated by circles (o), and the measurement sample indicated by an asterisk (*).

[0150] Figure 11d. Snapshot at sample k=500 of simulation of Map 1 with TRT, estimated map indicated by a full line (___), the measured map indicated by a dashed line (---), the boundary maps indicated by dash-dotted lines (-·-), artificial samples indicated by circles (o), and the measurement sample indicated by an asterisk (∗).

[0151] Figure 12a. Snapshot at sample k=1 of simulation of Map 1 with LPRM, estimated map indicated by a full line (___), the measured map indicated by a dashed line (---), the boundary maps indicated by dash-dotted lines (-. -), artificial samples indicated by circles (o), and the measurement sample indicated by an asterisk (∗).

[0152] Figure 12b. Snapshot at sample k=60 of simulation of Map 1 with LPRM, estimated map indicated by a full line (__), the measured map indicated by a dashed line (---), the boundary maps indicated by dash-dotted lines (__.__), artificial samples indicated by circles (o), and the measurement sample indicated by an asterisk (*).

[0153] Figure 12c. Snapshot at sample k=300 of simulation of Map 1 with LPRM, estimated map indicated by a full line (__), the measured map indicated by a dashed line (---), the boundary maps indicated by dash-dotted lines (_._), artificial samples indicated by circles (o), and the measurement sample indicated by an asterisk (∗).

[0154] Figure 12d. Snapshot at sample k=500 of simulation of Map 1 with LPRM, estimated map indicated by a full line (__), the measured map indicated by a dashed line (---), the boundary maps indicated by dash-dotted lines (-·-), artificial samples indicated by circles (o), and the measurement sample indicated by an asterisk (∗).

[0155] Figure 13a. Snapshot at sample k=1 of simulation of Map 2 with TRT, estimated map indicated by a full line (__), the measured map indicated by a dashed line (---), the boundary maps indicated by dash-dotted lines (_._), artificial samples indicated by circles (o), and the measurement sample indicated by an asterisk (∗).

[0156] Figure 13b. Snapshot at sample k=60 of simulation of Map 2 with TRT, estimated map indicated by a full line (__), the measured map indicated by a dashed line (---), the boundary maps indicated by dash-dotted lines (- · -), artificial samples indicated by circles (o), and the measurement sample indicated by an asterisk (*).

[0157] Figure 13c. Snapshot at sample k=300 of simulation of Map 2 with TRT, estimated map indicated by a full line (___), the measured map indicated by a dashed line (---), the boundary maps indicated by dash-dotted lines (_._), artificial samples indicated by circles (o), and the measurement sample indicated by an asterisk (*).

[0158] Figure 13d. Snapshot at sample k=500 of simulation of Map 2 with TRT, estimated map indicated by a full line (__), the measured map indicated by a dashed line (---), the boundary maps indicated by dash-dotted lines (_._), artificial samples indicated by circles (o), and the measurement sample indicated by an asterisk (∗).

[0159] Figure 14a. Snapshot at sample k=1 of simulation of Map 2 with LPRM, estimated map indicated by a full line (__), the measured map indicated by a dashed line (---), the boundary maps indicated by dash-dotted lines (_._), artificial samples indicated by circles (o), and the measurement sample indicated by an asterisk (∗).

[0160] Figure 14b. Snapshot at sample k=60 of simulation of Map 2 with LPRM, estimated map indicated by a full line (__), the measured map indicated by a dashed line (---), the boundary maps indicated by dash-dotted lines (_._), artificial

samples indicated by circles (o), and the measurement sample indicated by an asterisk (∗).

[0161] Figure 14c. Snapshot at sample k=300 of simulation of Map 2 with LPRM, estimated map indicated by a full line (__), the measured map indicated by a dashed line (---), the boundary maps indicated by dash-dotted lines (_._), artificial samples indicated by circles (o), and the measurement sample indicated by an asterisk (∗).

[0162] Figure 14d. Snapshot at sample k=500 of simulation of Map 2 with LPRM, estimated map indicated by a full line (--), the measured map indicated by a dashed line (---), the boundary maps indicated by dash-dotted lines (_._), artificial samples indicated by circles (o), and the measurement sample indicated by an asterisk (∗).

[0163] From the figures above, which shows snapshots of the LPRM, it can be seen that artificial samples are not generated in every possible LM in each snapshot. This is due to the parameter T_{act} =10 which hinders actualization passed LMs which have received measurement samples within the last 10 incoming samples.

[0164] Figures 14a-d shows a good illustration of the effects of re-estimation of the regression models. By comparing the pattern of the actualization of figures (a) and (d), it is clear that the regression models have been reestimated. It also shows that re-estimation has only been done in LMs which are in the range of the engine speeds in the drive cycle. Hence no re-estimation has been done in LMs located at engine speeds higher than 4000 and lower than 1700, which results in a preserved actualization pattern from the boundary maps.

[0165] Another noteworthy phenomenon can be found when comparing Figures 14 b, c, and d. In Figure 14b the regression models of the higher engine speeds are re-estimated, which is verified by the new actualization pattern in Figure 14d. But the new actualization pattern is not notably present in Figure 14c, which lies between Figure 14b and Figure 14d in time. This can be explained by the fact that the re-estimation was done by replacing sample pairs at the lower end of the height variation interval of the map, i.e. close to the lower boundary map, whereas the sample pairs closer to the upper boundary map is preserved. To achieve adaptation of the actualization pattern close to the upper boundary map, some sample pairs in this region needs to be replaced as well. This has happened in the region of normal engine speeds around 2200 where the actualization pattern lies between the upper boundary map and the real map, see Figure 14c. Although no artificial samples are generated in this snapshot, this conclusion can be drawn from knowing that no measurement samples have been received at low engine speeds, see the drive cycle. That is, merely artificial samples have been received at these low engine speeds.

Table 5.1. Average error sum over five simulations

Actualization	**Map 1**	**Map 2**
No actualization	11640	16180
TRT	10720	15380
LPRM - without re-estimation	1698	8501
LPRM - with re-estimation	1985	7186

[0166] A sum of the errors between the real map and the estimated map in 200 equidistant points over the maps total range measured after each received sample will be used as a performance measure. The numbers in Table 5.1 show the average of this error sum over five simulations in each given example. In addition to the examples depicted in Figures 11a-d to 14a-d, Table 5.1 also gives the average error sum of the case of no actualization and the LPRM with no re-estimation allowed.

[0167] The performance measure used here is a bit blunt because it does not weigh the errors after the frequency they are read. The map is more frequently read in normal cruising speeds than at e.g. extremely high engine speeds. A larger radius in the TRT than used in the simulations above gave a slightly smaller error sum. But it also resulted in more significant destructive learning in the range of normal cruising speed. Consequently the radius of the tent was kept smaller then its optimum with respect to the sum of errors.

[0168] Although the deviation of Map 2 with regard to the boundary maps, the TRT was outperformed by the LPRM with both allowed and restricted re-estimation. Comparing the performance of the LPRM on the two different maps, it is clear that the quality of the a priori-information is crucial for the performance of the adaptation. Furthermore as expected, allowing re-estimation of the regression models gave a lower error sum if the a priori-information is poor. But it is slightly higher in the case of perfect a priori-information. It is clear that the incorporated a priori-information gives an edge to LPRM. It is also clear that this holds for somewhat poor a priori-information as well.

5.3 Simulation - Example 3

[0169] This simulation Example will compare adaptive strategies at different levels of measurement noise on the drive cycle from the previous Example and this will be done on both maps from the same Example. All the actualization methods in Table 5.1 will be evaluated with both RLS and DA as update algorithms and each combination are simulated

five times and from that an average error sum is formed. No spatial filtering was done in the simulations. The noise is white normally distributed with three different values of standard deviation; low (σ_n=0.01), high (σ_n=0.06), and very high (σ_n=0.15). Compare the standard deviations with the values of the maps in Figures 11a-d to 14a-d. The noise level should also be compared with the change rate of the map and the variation interval of the map. Tables 5.2-5.3 show the result of the simulations on the two different maps. For parameter settings, see Appendix.

Table 5.2. Average error sum over five simulations, map 1

Actualization	σ_n =0.01	σ_n =0.06	σ_n =0.15
No actualization (DA)	11900	12400	13760
No actualization (RLS)	15070	15120	15380
TRT (DA)	10580	10780	12020
TRT (RLS)	11120	11430	11870
LPRM (DA) re-estimation	2238	4760	13110
LPRM (RLS) re-estimation	5707	6026	6091
LPRM (DA) no re-estimation	1874	4558	12120
LPRM (RLS) no re-estimation	4430	4644	5368

Table 7.3. Average error sum over five simulations, map 2

Actualization	σ_n **=0.01**	σ_n **=0.06**	σ_n **=0.15**
No actualization (DA)	16130	16680	18230
No actualization (RLS)	18310	18640	18620
TRT (DA)	15340	16080	17530
TRT (RLS)	15110	15590	16240
LPRM (DA) re-estimation	7270	8279	15750
LPRM (RLS) re-estimation	9413	9492	1005
LPRM (DA) no re-estimation	8589	9607	15295
LPRM (RLS) no re-estimation	8965	9304	9366

[0170] No restrictions on the beta-parameters where set and the parameters ($\dot{y}_{\max}$ ε_{old} , *outlierLimit*) used to avoid incorrect re-estimation of the regression models where given values with high acceptance levels on the change rate of the map. Hence with more restrictive parameter settings the performance for the simulations with re-estimation would have performed better. Furthermore the number of sample pairs used for estimation of the regression models was set to N=5 in the simulations, a higher number should give a better performance with re-estimation in high noise levels.

[0171] The change rate in the simulations is probably much higher than real world applications with slow variations. The map sweeps the interval between the two boundary maps in just 500 steps. This situation should give benefits to DA, with its high plasticity. Which is verified in the simulations where DA performs surprisingly well compared to RLS, where the forgetting factor was set to the relatively low value 0.9. Merely at very high noise levels does RLS typically perform better than DA. The simulations also showed that high spatial frequency can appear with the RLS if the initial variance elements are large, thus they should be given relatively small values.

[0172] Figure 15 shows a schematic illustration of a vehicle comprising a combustion engine (E), a driveline (D) and an electronic control unit (ECU) for controlling said combustion engine (E) and said driveline (D). Sensors (10, 11, 12) for measuring at least one engine or driveline parameter are connected to the electronic control unit (ECU) in order to provide measured samples. The electronic control unit (ECU) is provided with maps of measured or estimated samples for at least one of the said engine or driveline parameters. The maps are adapted using the method described above.

APPENDIX

A Pseudo Code

A.1 Local Pattern Regression Models

Algorithm: *Replacement of old estimation samples*

Input: sample pair (y_i, y_{i+1}), N pairs of estimation samples ($y^{(i)}_{j=1...N}, y^{(i+1)}_{j=1...N}$)

Output: updated estimation samples ($y^{(i)}_{j=1...N}, y^{(i+1)}_{j=1...N}$)

dist="large number"

$for\ j = 1...N$ / Find which of the old saved samples, y, that is closest to y

$if\ \left|y_i - y_j^{(i)}\right| < dist$

$index = i$

$dist = \left|y_i - y_j^{(i)}\right|$

end

end

$if\ y_i < y_j^{(i)}$

$if\ \left|y_i - y_{j-1}^{(i)}\right| < 0.1\left|y_{j+1}^{(i)} - y_{j-1}^{(i)}\right|$

$y_{j-1}^{(i)} = y_i$

else

$y_j^{(i)} = y_i$

end

else

if $\left|y_i - y_{j+1}^{(i)}\right| < 0.1\left|y_{j+1}^{(i)} - y_{j-1}^{(i)}\right|$

$y_{j+1}^{(i)} = y_i$

else

$y_j^{(i)} = y_i$

end

end

(Observe, always do the evaluation of which *j* sample to replace in one of the LMs, the left one *i* (or the right one *i*+1), to keep the sample pairs in order)

Algorithm: *Weight decrease*

Input: weight vector $w^{(i,i+1)}$ used in estimating regression models between LM *i* and LM *i*+1, position of the new sample *j*

Output: weight vector $w^{(i,i+1)}$

weightDifference = $0.9\frac{1}{N}$ / weight difference used in simulations

currentWeight = $w_j^{(i,i+1)}$

for $v = 1...N$

if $w_v^{(i,i+1)} > currentWeight$

$w_v^{(i,i+1)} = w_v^{(i,i+1)} - weightDifference$

end

end

$w_j^{(i,i+1)} = 1$

B Matlab Code

B.1 Spatial Filters

```
% Generates filter
[b a]=butter(6,0.65)

% Filter operator
theta = filtfilt(b,a,theta);

% Power spectrum
THETA = fft(theta,512);
Pyy = THETA.* conj(THETA)/512;
f = 2*(0:256)/512;
plot(f,Pyy(1:257),'k')
```

C Parameter Settings

C.1 Simulation Study 2

M=21

r=4

N=5

T_{act} =10

$T_{collect}$ =1

$outlierLimit$ =1

$\beta_{0_{\max}}, \beta_{1_{\max}}$ =inf

$\beta_{0_{\min}}, \beta_{1_{\min}}$ =-inf

C.2 Simulation Study 3

M=21

r=4	% only used in TRT
N=5	
δ =0.2	
w_{limit} =0.01	% only used in LPRM
T_{act} =10	% only used in LPRM
$T_{collect}$ =1	% only used in LPRM – with reestimation
$outlierLimit$ =1	% only used in LPRM – with reestimation
$\beta_{0_{\max}}, \beta_{1_{\max}}$ =inf	% only used in LPRM – with reestimation
$\beta_{0_{\min}}, \beta_{1_{\min}}$ =-inf	% only used in LPRM – with reestimation
λ =0.90	% only used in (RLS)
λ_{art} =0.93	% only used in LPRM (RLS) – with reestimation
λ_{old} =0.85	% only used in LPRM (RLS) – with reestimation
$\dot{y}_{avg}$ =0.002	% only used in LPRM (RLS) – with reestimation
$\dot{y}_{\max}$ =0.4	% only used in LPRM – with reestimation
ε_{old} =0.1	% only used in LPRM (RLS) – with reestimation
$t_{\alpha/2}$ =2	% only used in LPRM

NOTATIONS

Abbreviated Terms

[0173]

LLNFM	Local linear neuro-fuzzy models
LM	General local models
LLM	Local linear models in the LLNFM representation
LPRM	Local pattern regression models
LOLIMOT	Local linear model tree
DA	Direct adjustment
NLMS	Normalized least mean squares
RLS	Recursive least squares
LC	Local correction
TRT	Tent roof tensioning

Constants, Functions, Sets, and Variables

[0174] $x \in X \{X \subset \; ^{D}\}$ Input space (independent variable)

$x(k)$ The independent variable at time k

$x^{(i)}$ Coordiante of LM *i*
$x = \xi$ x-value at the operating point

$y \in Y$ {Y⊂ } Output space
$y(x) : X \rightarrow Y$ Measured map
$y_u(x) : X \rightarrow Y$ Noise free measured map
$\dot{y}(x) : X \rightarrow Y$ Estimated map
$\dot{y}_k(x)$ Map at time *k*
$\dot{y}^{(i)}$ Map value in LM *i*

$^{a}y_k^{(i)}$ Artificial sample placed in LM *i at* time *k*

$\hat{y}_k$ Stored map and variance/covariance matrices stored in memory.
n Additive measurement noise
c_i Coordinate of node *i* (one-dimensional look-up table)
$c_{1,i}$ Grid line in dimension 1 at location *i* (two-dimensional look-up table)
$(c_{1,i}\ c_{2,j})$ Coordinate of node (*i,j*) (two-dimensional look-up table)
θ_i Height parameter of node *i* (one-dimensional look-up table)
$\theta_{i,j}$ Height parameter of node (*i,j*) (two-dimensional look-up table)
θ_{i0} Bias parameter of LLM *i* (LLNFM)
θ_{iD} Rake parameter in dimension D of LLM *i* (LLNFM)

$\Phi_i(x, \theta_i^{(nl)})$ Basis function in the basis function framework

$L_i(x,\theta_i)$ Linear function in the basis function framework
M Number of nodes in a one-dimensional map or a LLNFM of arbitrary dimension
$M_1 \times M_2$ Number of nodes in a two-dimensional map
S Number of samples for initial optimization
P(*k*) Covariance matrix in the RLS algorithm at time *k*
$w^{(i,i+1)}$ Weight used in the RLS algorithm when local model *i* is updated with an artificial sample generated from model *i*+1
$K^{(i)}$ Number of measurement samples since LM received a measurement
$\beta_0^{(i,i+1)}$ Regression model parameter (bias), from model *i* to *i*+1
$\beta_1^{(i,i+1)}$ Regression model parameter (rake), from model *i* to *i*+1
$Y^{(i+1)}$ Estimation of bias level in *i*+1 from regression model
Q Weight matrix
δ Determines the convergence rate of the weights
w_{limit} Minimum weight for continued actualization
T_{act} Time limit of how early it is permitted to actualize since the latest received measurement in the current LM
$T_{collect}$ Maximum time limit between two collected samples for reestimating regression models
$\dot{y}_{avg}$ Average change rate of the map
$\dot{y}_{max}$ Highest tolerated change rate
ε_{old} Highest tolerated residual between updated LM and new measurement sample
$\beta_{0max}, \beta_{1max}$ Highest allowed values on the regression parameters
$\beta_{0min}, \beta_{1min}$ Lowest allowed values on the regression parameters
outlierLimit Outlier limit for regression model reestmation
N Number of stored samples used for estimation of the regression models

Claims

1. Method for adapting a combustion engine or a vehicle driveline control map, which map is expressed by a basis function equation (Eq. 1), **characterized by** the method involving the steps of:

- receiving a measured sample of an engine parameter at an operating point,

- updating local models adjacent of the operating point using an update algorithm,
- generating artificial samples in coordinates of local models located remote from the operating point and the said adjacent local models, and
- updating the said remote local models using an update algorithm.

2. Method according to claim 1, **characterized by** generating artificial samples in coordinates of local models in a map expressed by the equation

$$\hat{y}(x) = \sum_{i=1}^{M} L_i(x, \theta_i)\Phi_i(x, \theta_i^{(nl)}) \qquad \text{(Eq.1)}$$

where

θ_i is a height parameter of node *i*,
$L_i(x,\theta_i)$ is a linear function in the basis function framework,

$\Phi_i(x, \theta_i^{(nl)})$ is a basis function in the basis function framework, where (nl) indicates that the function is non-linear,
M is the number of nodes in a one-dimensional map or a LLNFM of arbitrary dimension

3. Method according to claim 1, **characterized by** generating artificial samples using an actualizing algorithm.

4. Method according to claim 2 or 3, **characterized by** generating artificial samples using a local pattern regression model (LPRM).

5. Method according to claim 4, **characterized by** updating the local models in a map represented by a look-up table.

6. Method according to claim 5, **characterized by** updating the local models using a recursive least squares (RLS) algorithm.

7. Method according to claim 5, **characterized by** updating the local models using a direct adjustment (DA) algorithm.

8. Method according to claim 5, **characterized by** updating the local models using a least means squares (NLMS) or a normalized lest means squares (LMS) algorithm.

9. Method according to claim 4,**characterized by** updating the local models in a map represented by a local linear neuro-fuzzy model (LLNFM).

10. Method according to claim 9, **characterized by** updating the local models using a recursive least squares (RLS) algorithm.

11. Method according to claim 9, **characterized by** updating the local models using a least means squares (NLMS) or a normalized lest means squares (LMS) algorithm.

12. Method according to claim 2 or 3, **characterized by** generating artificial samples using a tent roof tensioning (TRT) algorithm.

13. Method according to claim 12, **characterized by** updating the local models in a map represented by a local linear neuro-fuzzy model (LLNFM).

14. Method according to claim 13, **characterized by** updating the local models using a recursive least squares a (RLS) algorithm.

15. Method according to claim 13, **characterized by** updating the local models using a least means squares (NLMS)

or a normalized lest means squares (LMS) algorithm.

16. Method according to claim 12, **characterized by** updating the local models in a map represented by a look-up table.

17. Method according to claim 16, **characterized by** updating the local models using a recursive least squares (RLS) algorithm.

18. Method according to claim 16, **characterized by** updating the local models using a least means squares (NLMS) or a normalized lest means squares (LMS) algorithm.

19. Vehicle comprising an electronic control unit (ECU) for controlling a combustion engine (E) or a vehicle driveline and sensors (10, 11, 12) for measuring at least one engine or driveline related parameter, where the electronic control unit (ECU) is provided with a map of measured or estimated samples for the said at least one engine or driveline related parameter, **characterized in that** the map is adapted using the method of claim 1.

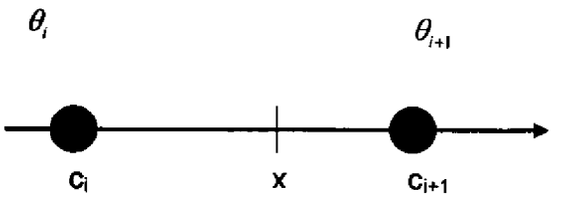

FIG. 1a

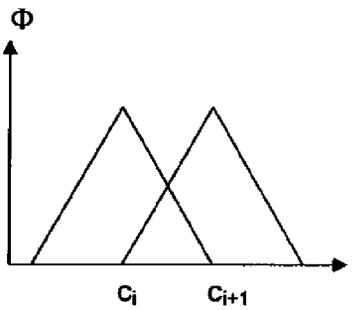

FIG. 1b

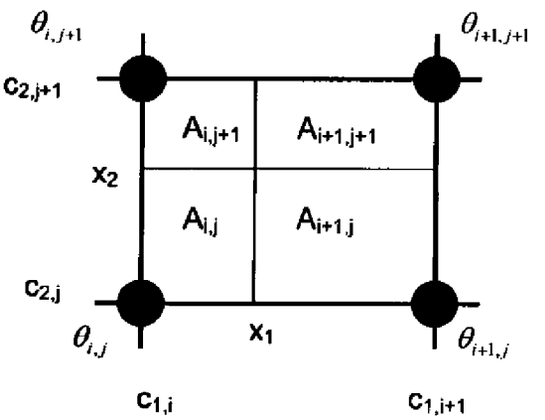

FIG. 2

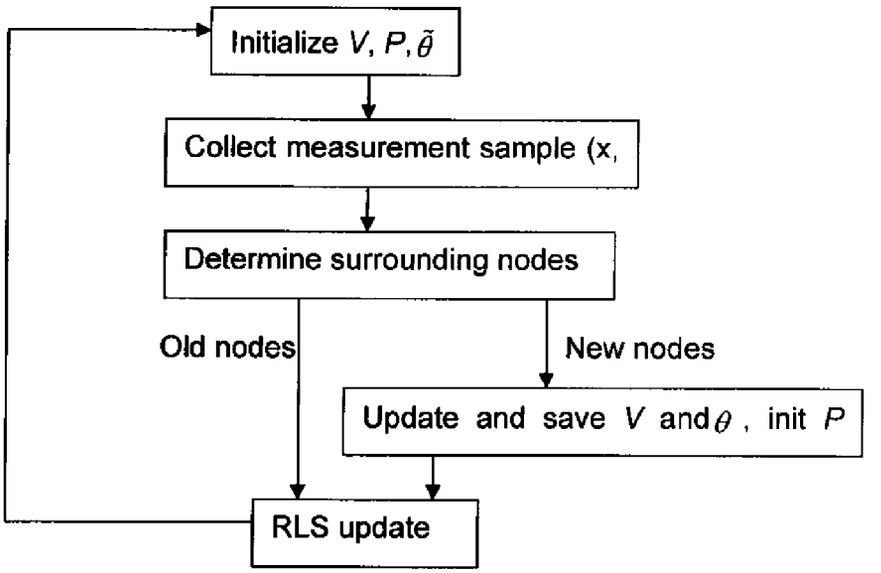
Initialize $V, P, \tilde{\theta}$
Collect measurement sample (x,
Determine surrounding nodes
Old nodes
New nodes
Update and save V and θ, init P
RLS update

FIG. 3

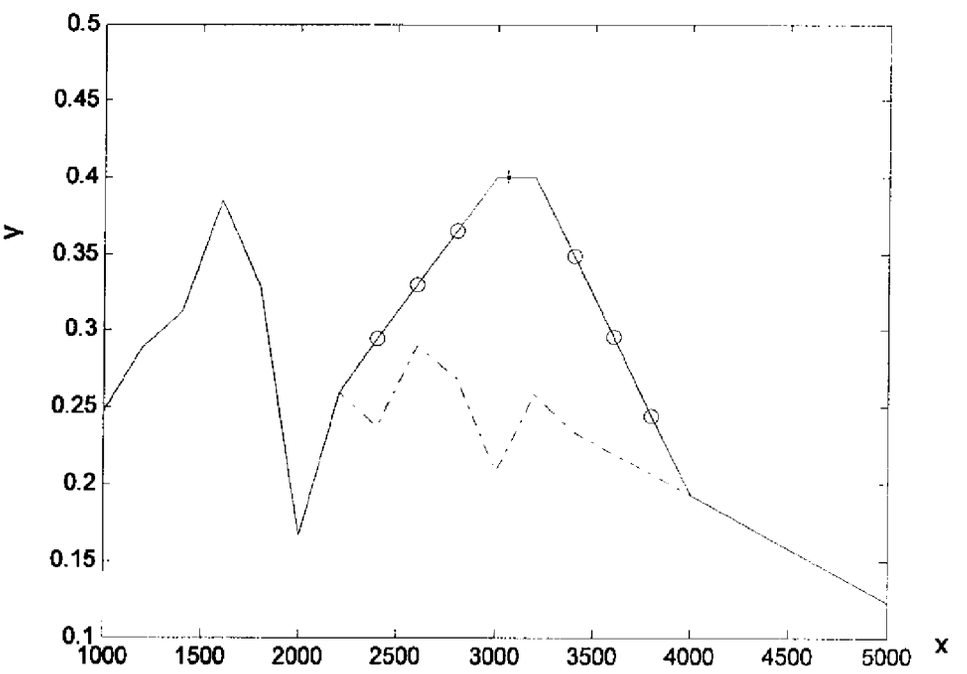
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
y
1000
1500
2000
2500
3000
3500
4000
4500
5000
x

FIG. 4

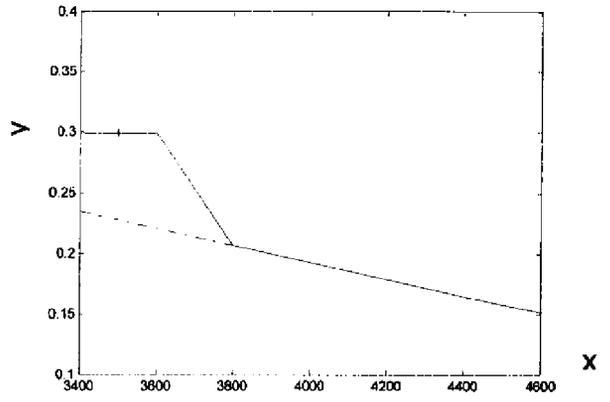

FIG. 5a

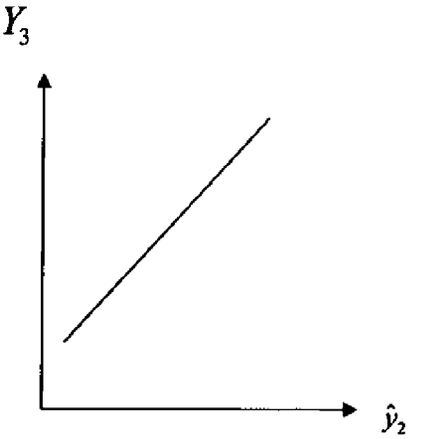

FIG. 5b

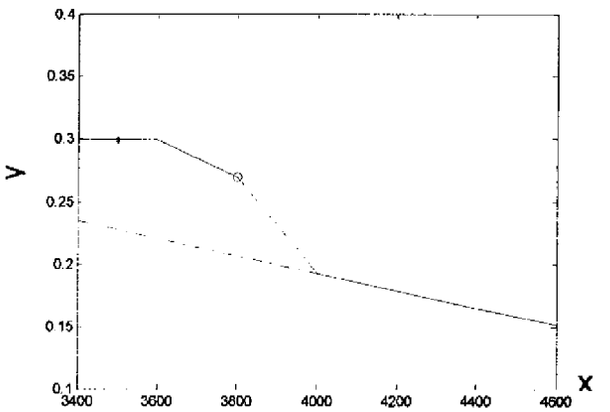

FIG. 5c

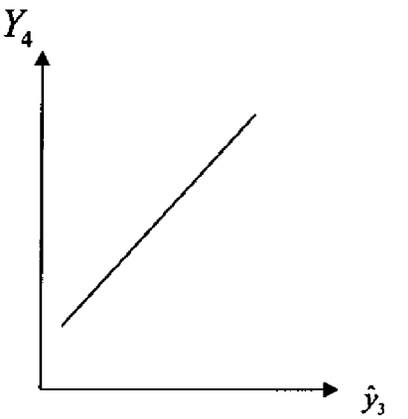

FIG. 5d

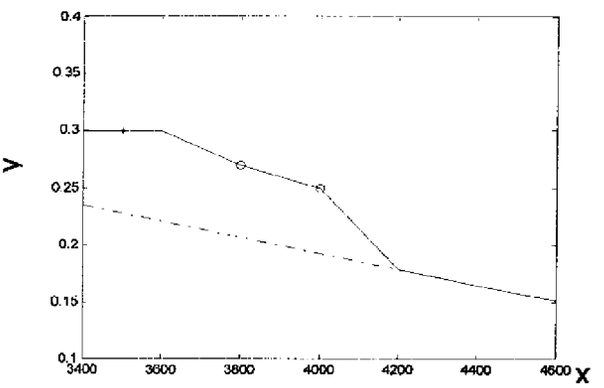

FIG. 5e

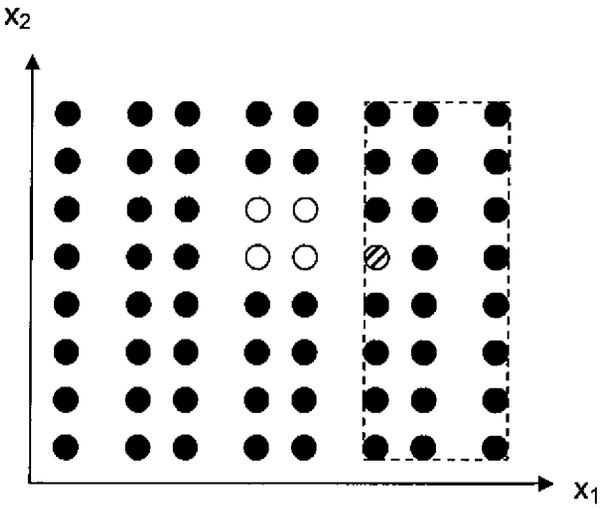

FIG. 6a

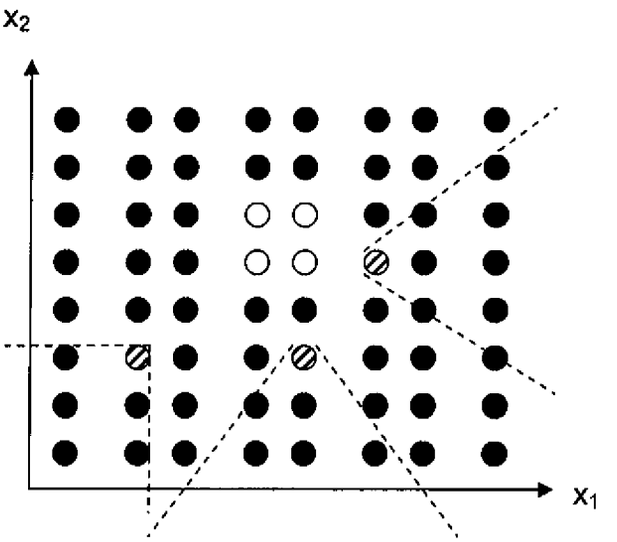

FIG. 6b

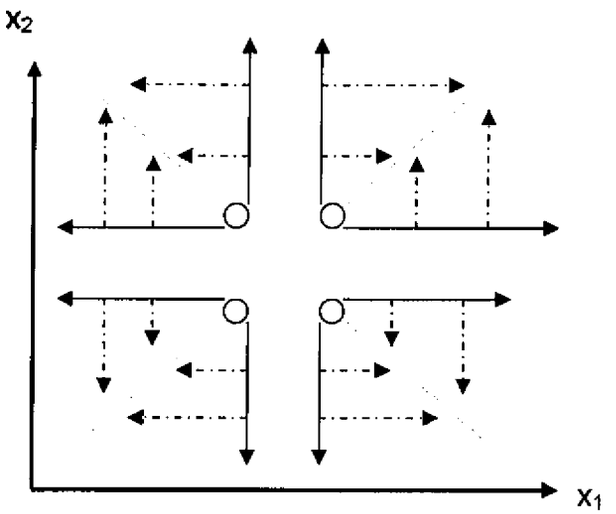

FIG. 7

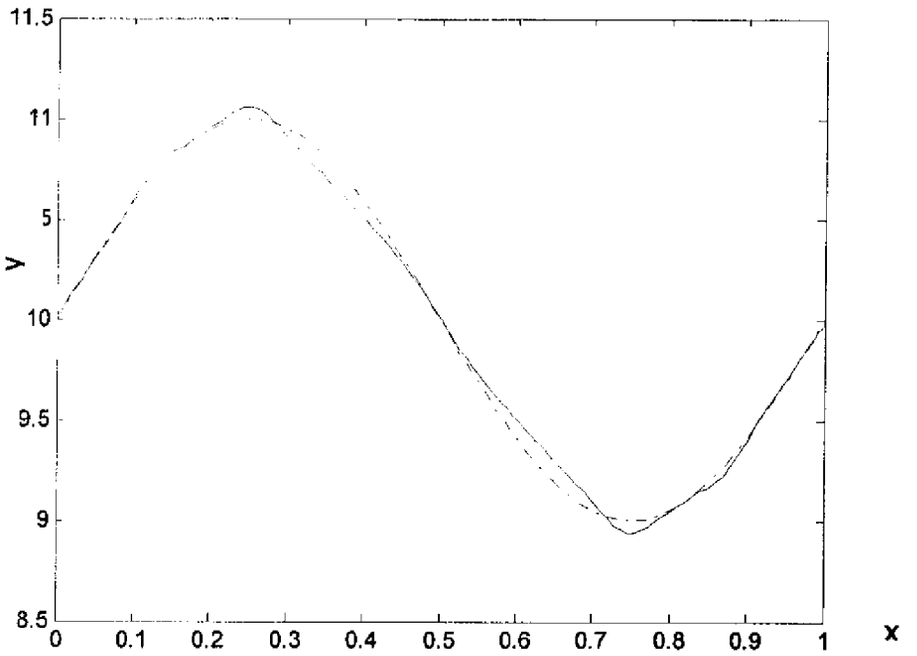
11.5
11
5
10
9.5
9
8.5
y
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
x

FIG. 8a

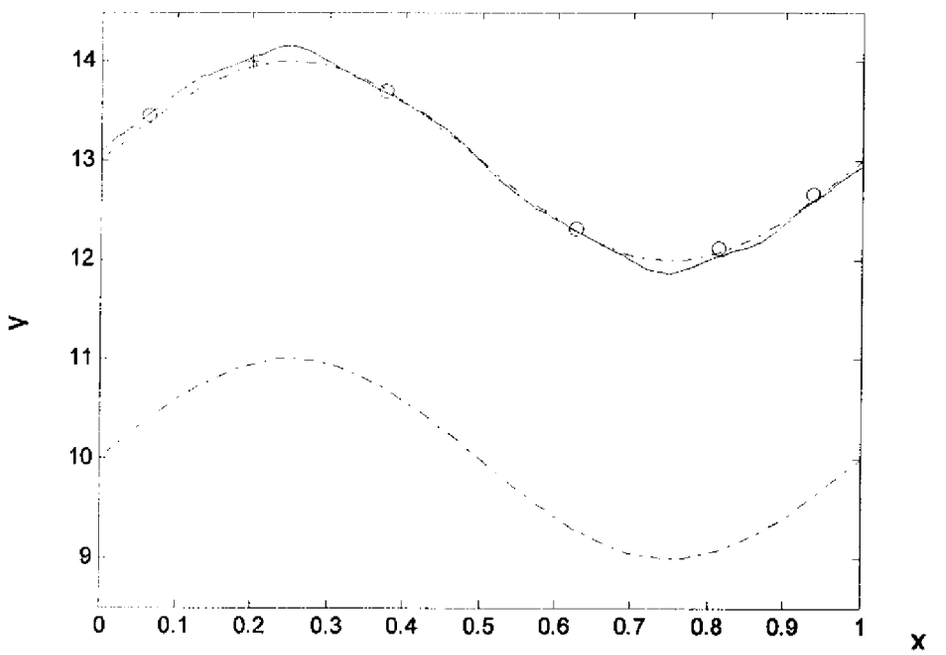
14
13
12
11
10
9
y
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
x

FIG. 8b

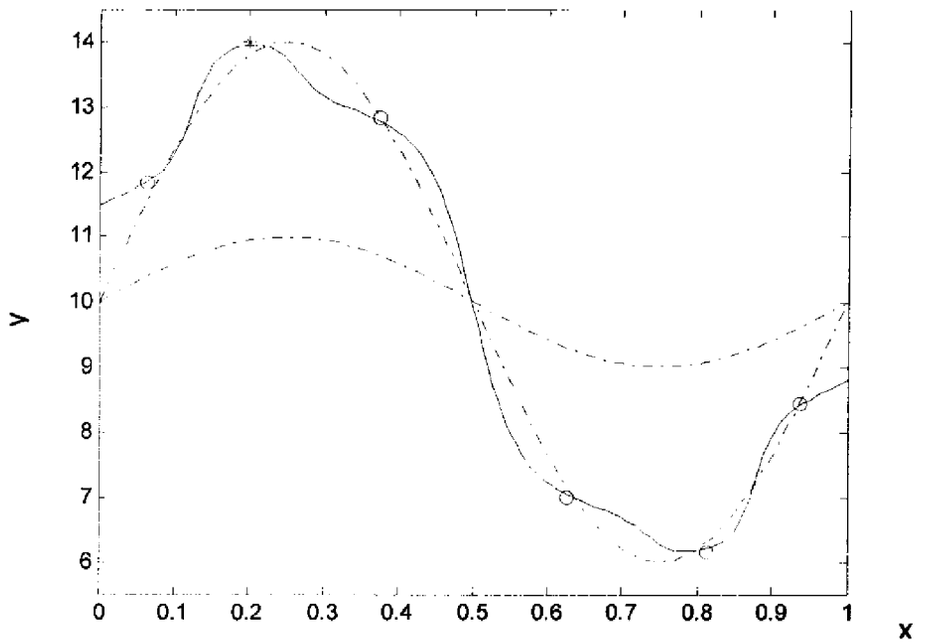
14
13
12
11
10
9
8
7
6
y
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
x

FIG. 8c

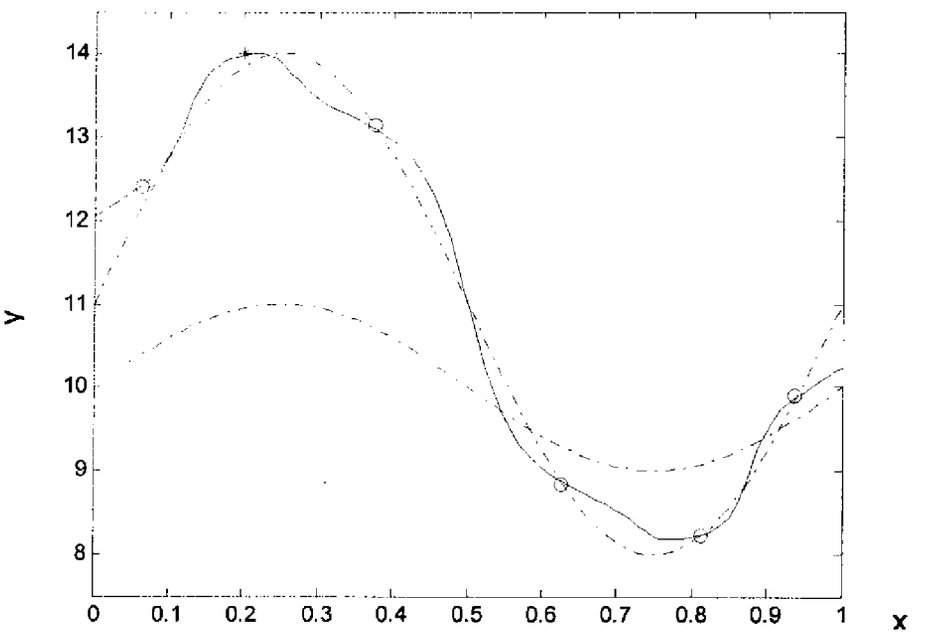
14
13
12
11
10
9
8
y
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
x

FIG. 8d

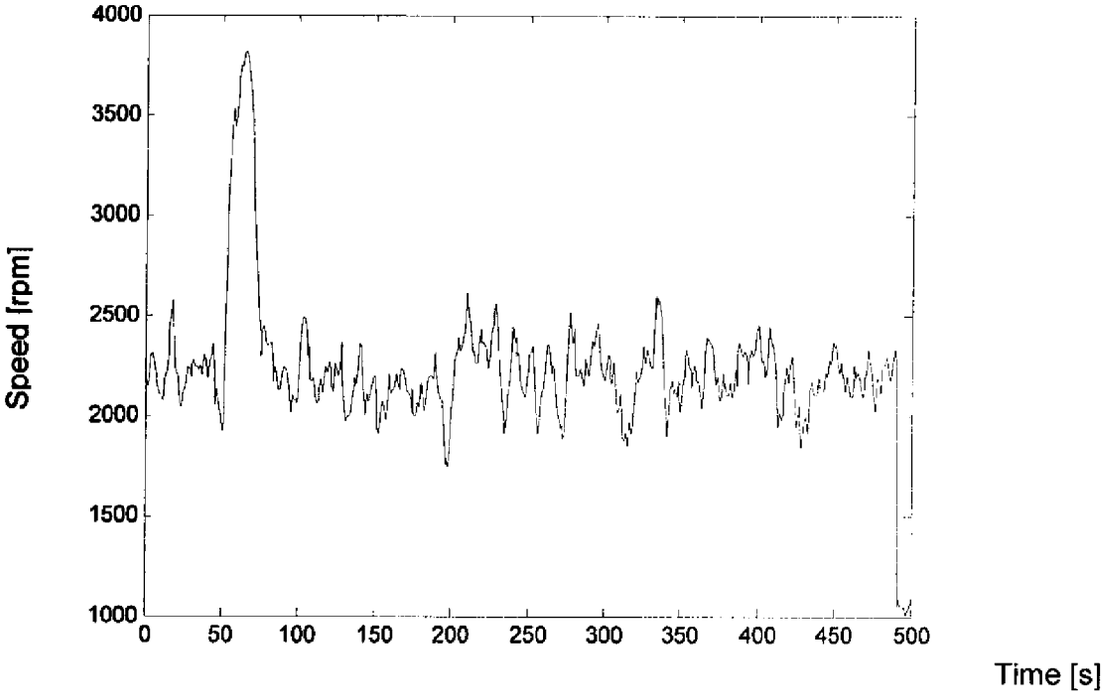
Speed [rpm]
4000
3500
3000
2500
2000
1500
1000
0
50
100
150
200
250
300
350
400
450
500
Time [s]

FIG. 9

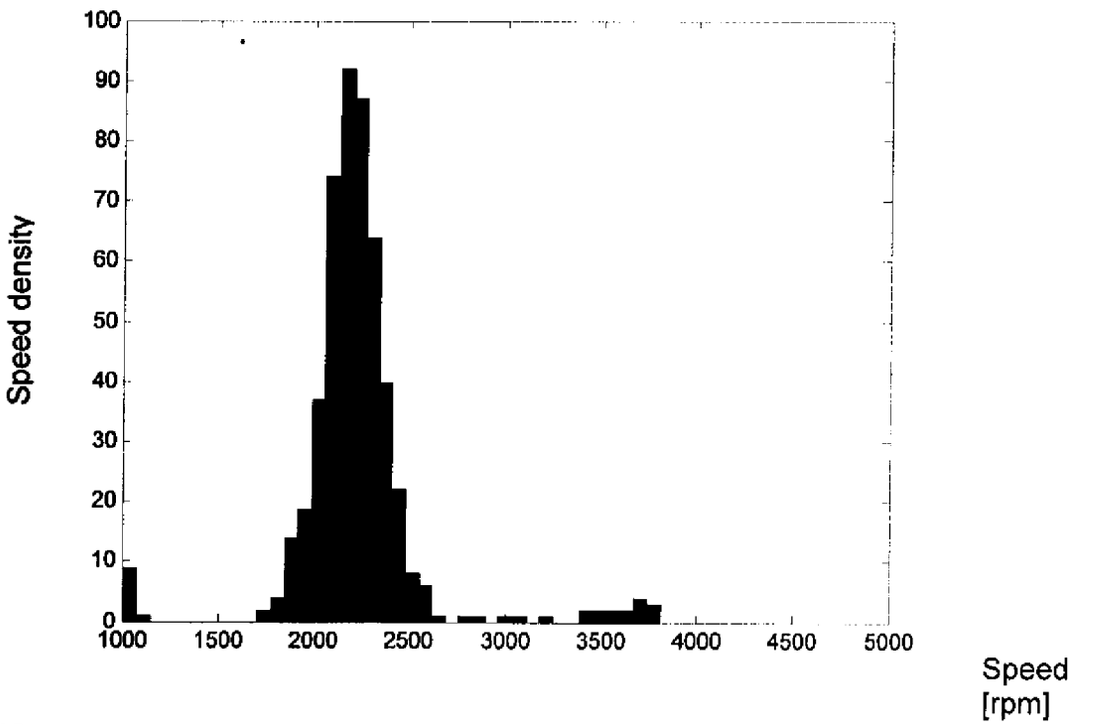
Speed density
100
90
80
70
60
50
40
30
20
10
0
1000
1500
2000
2500
3000
3500
4000
4500
5000
Speed
[rpm]

FIG.10

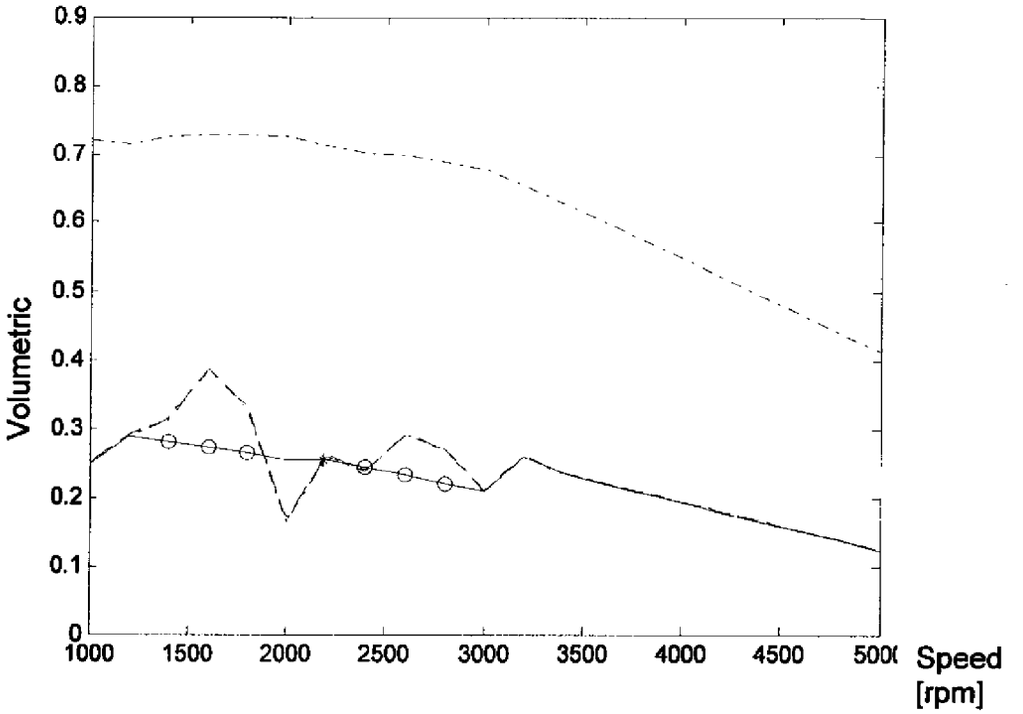

FIG. 11a

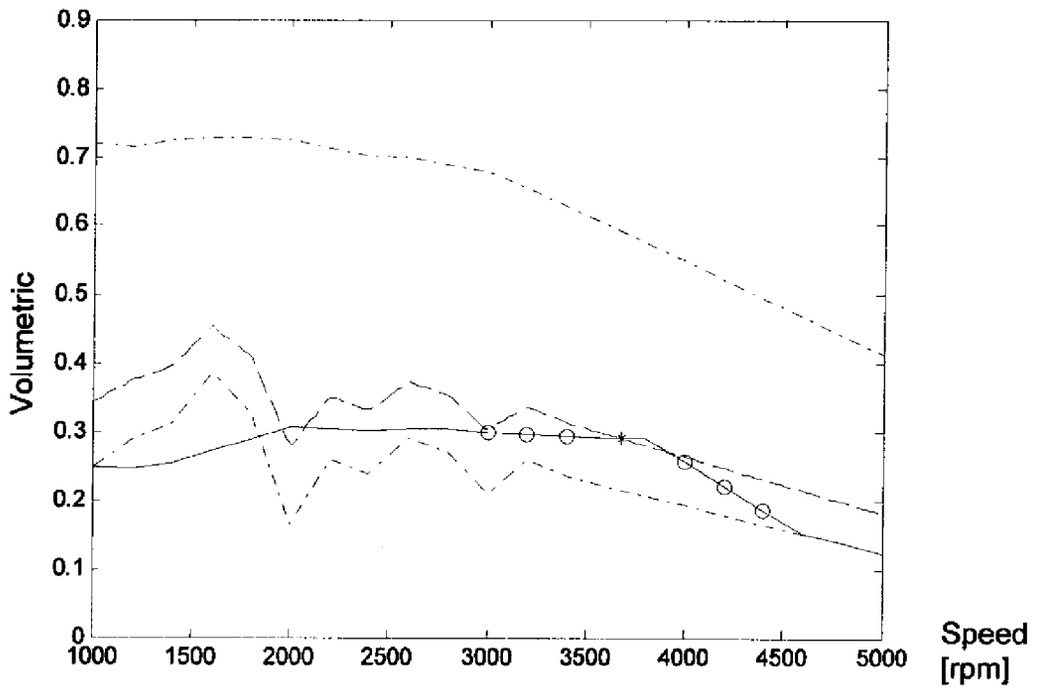

FIG. 11b

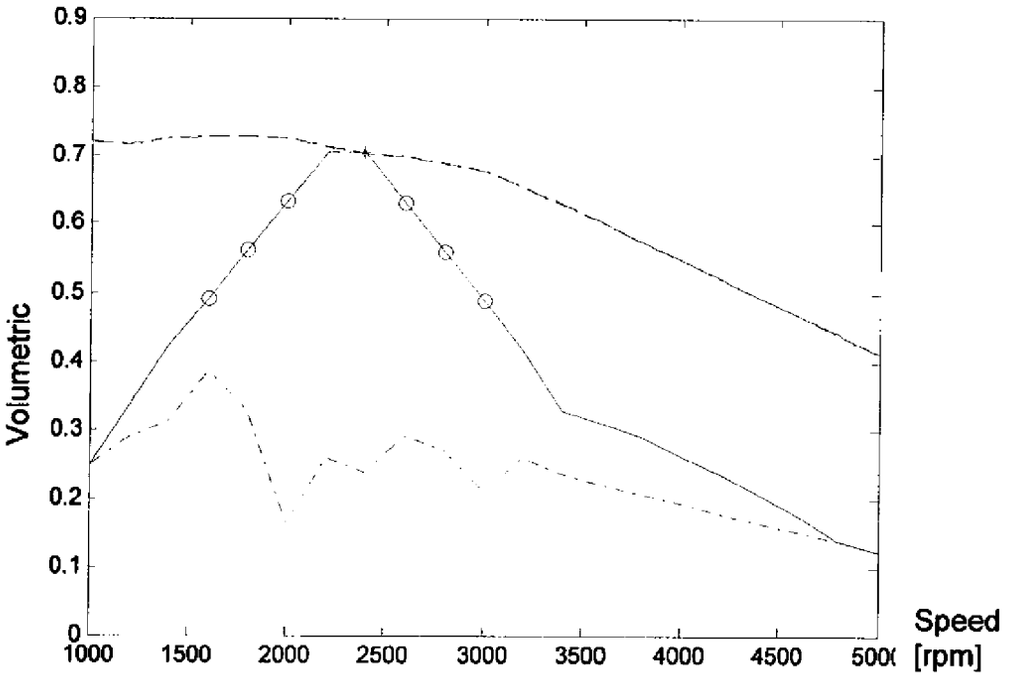

FIG. 11c

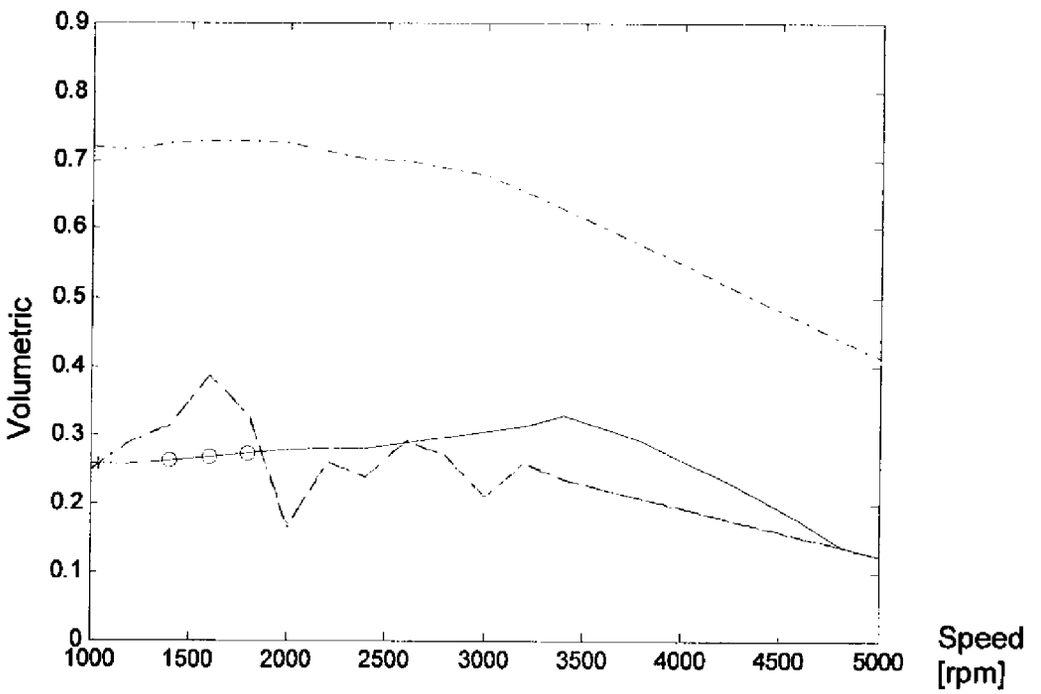

FIG. 11d

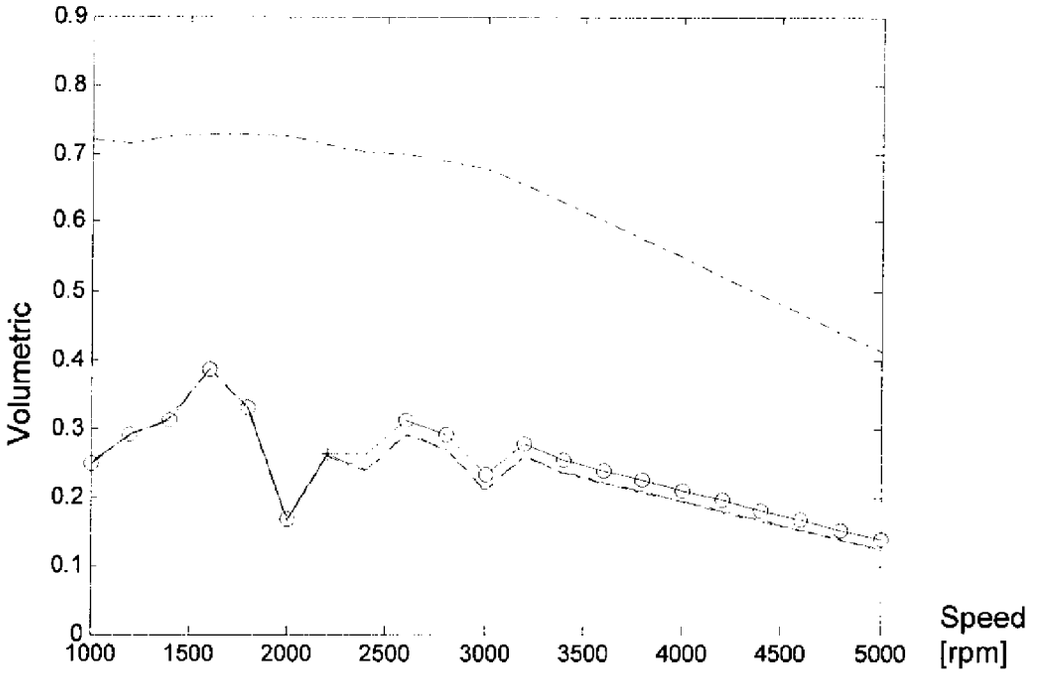

FIG. 12a

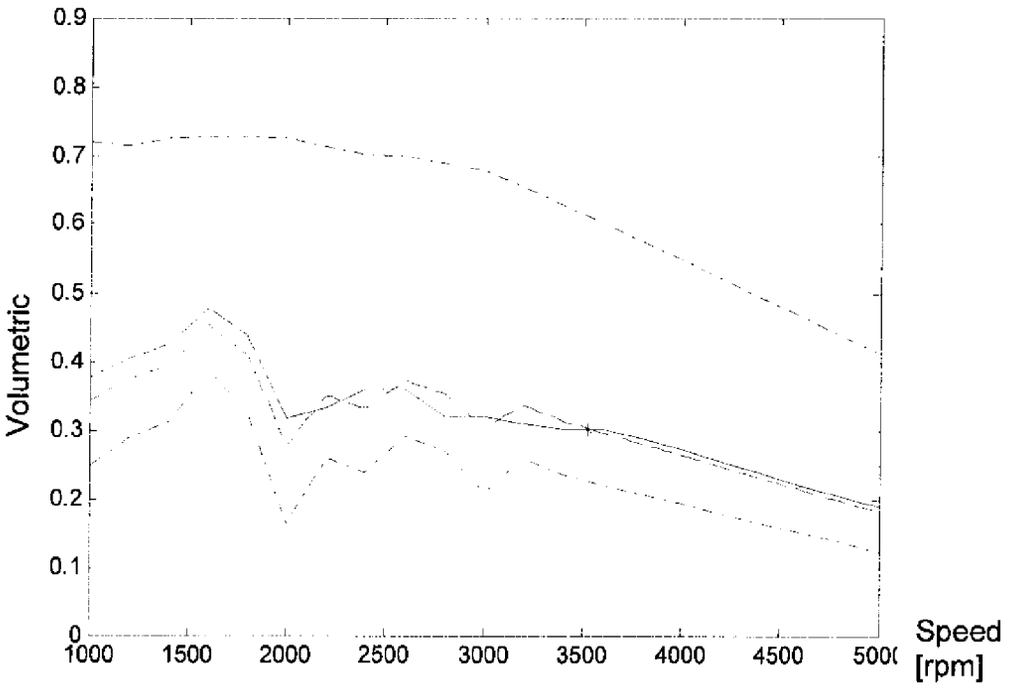

FIG. 12b

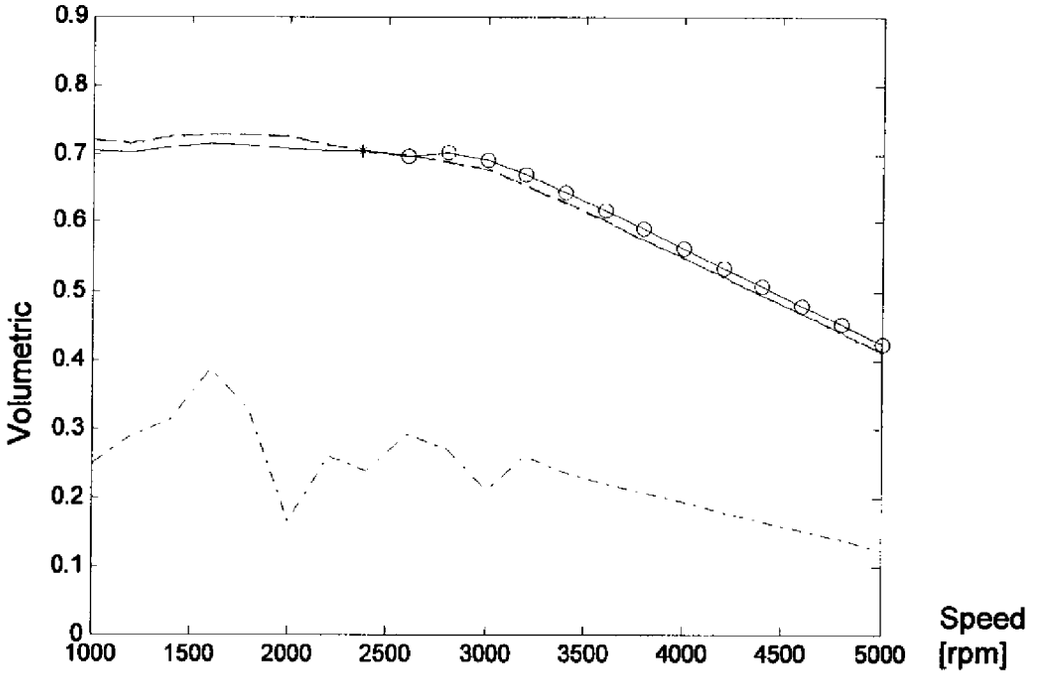

FIG. 12c

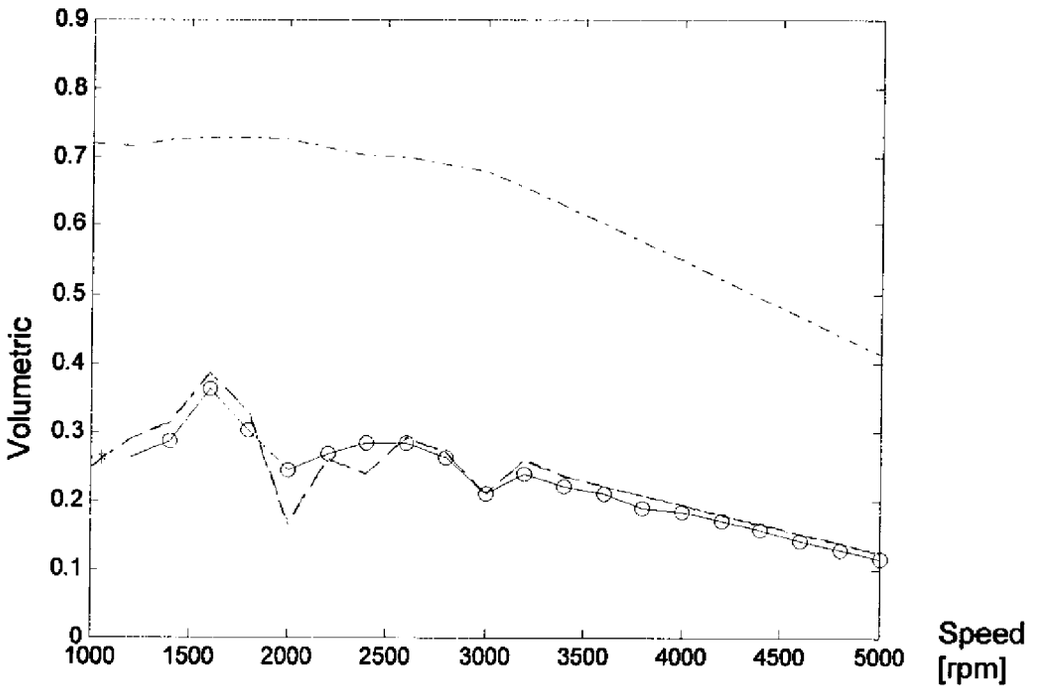

FIG. 12d

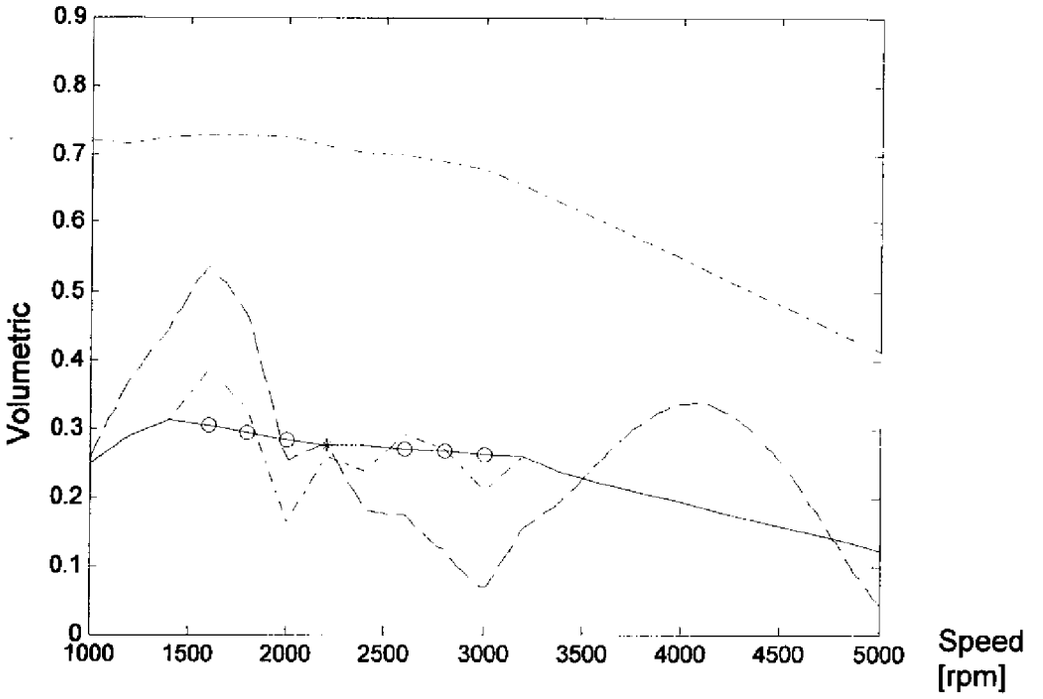

FIG. 13a

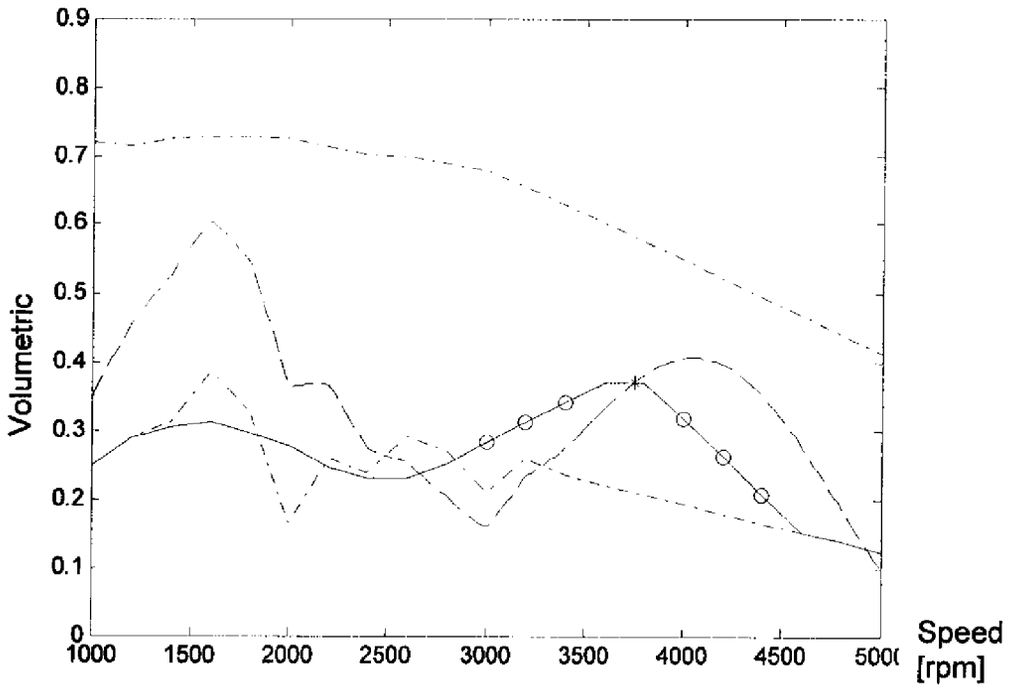

FIG. 13b

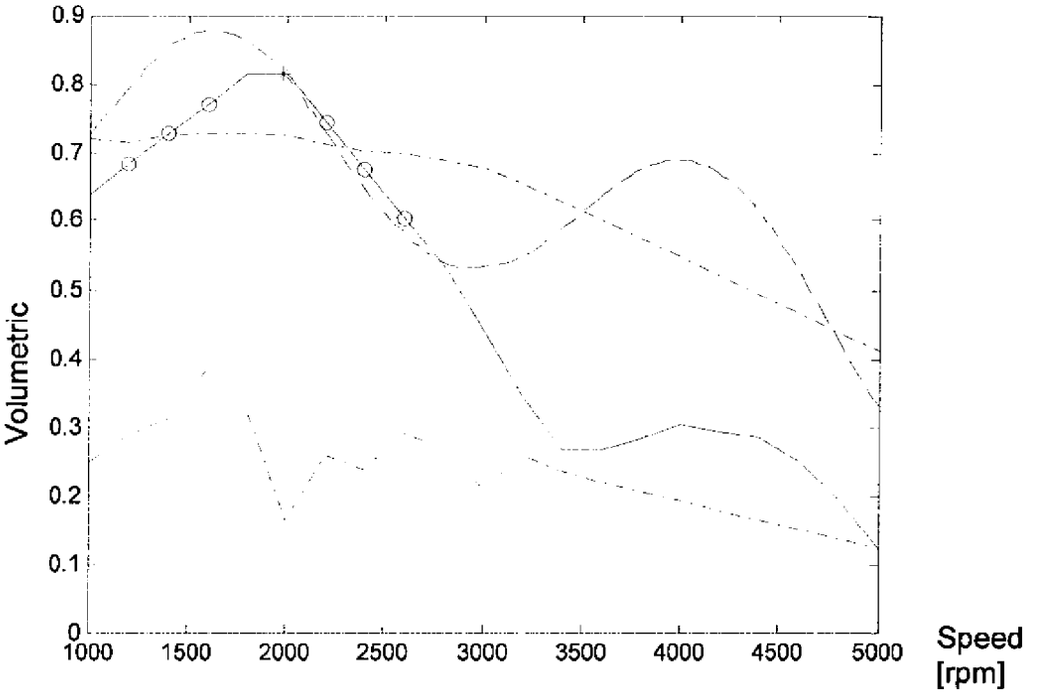

FIG. 13c

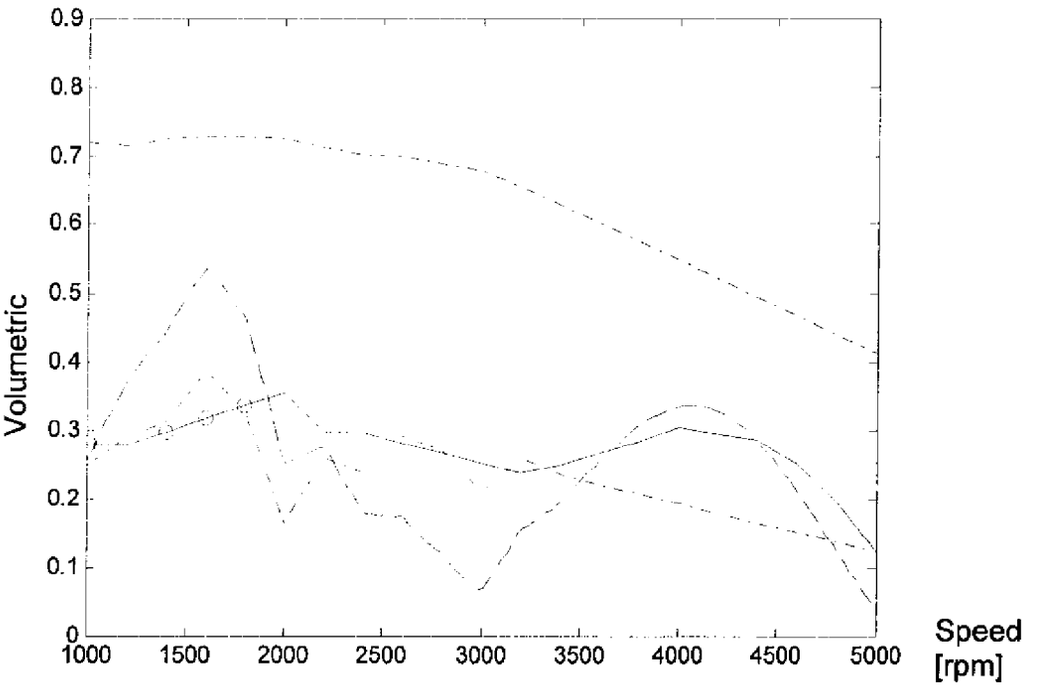

FIG. 13d

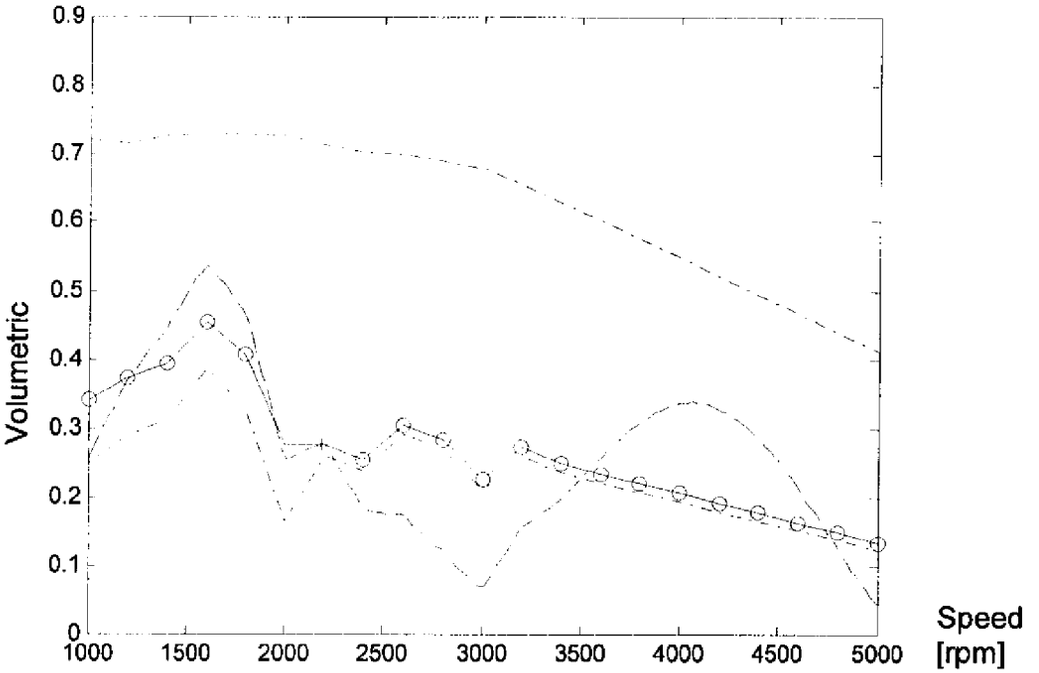

FIG. 14a

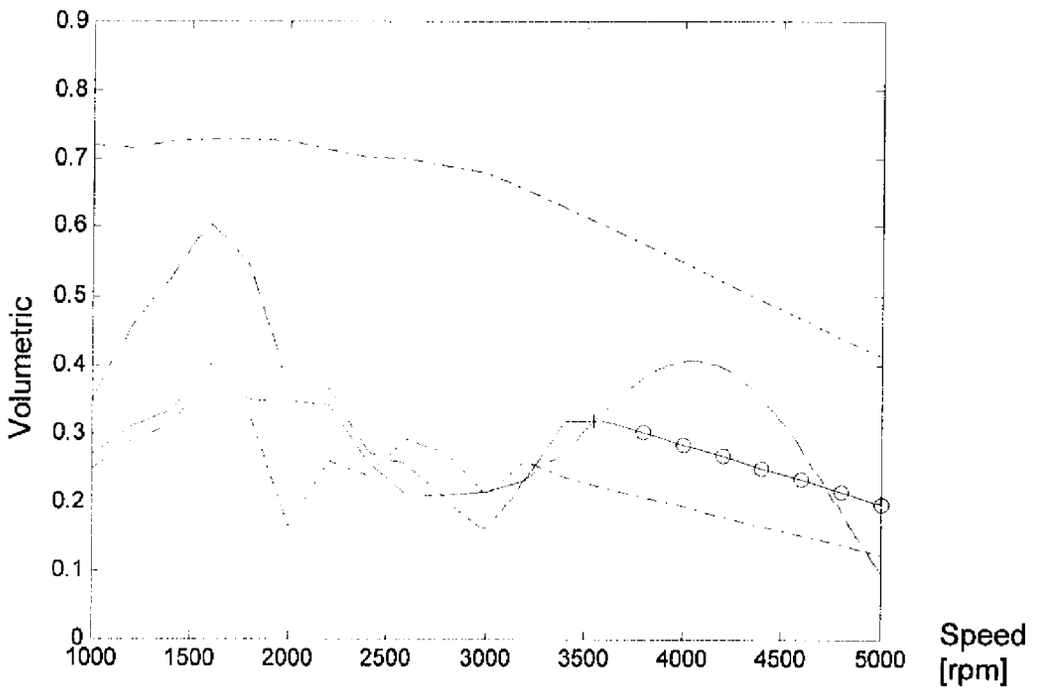

FIG. 14b

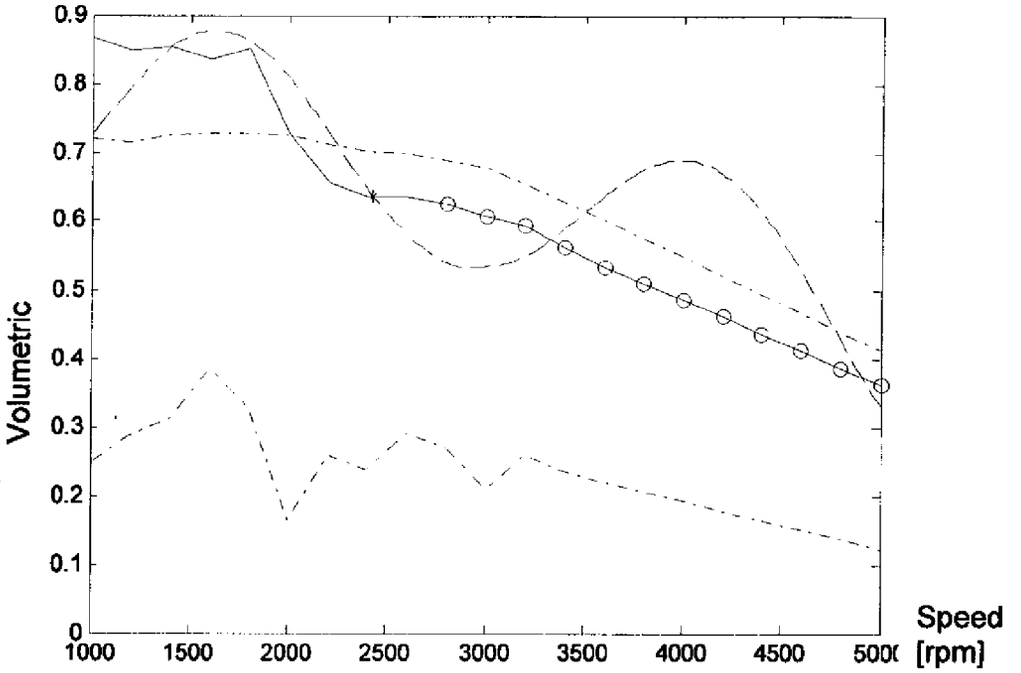

FIG. 14c

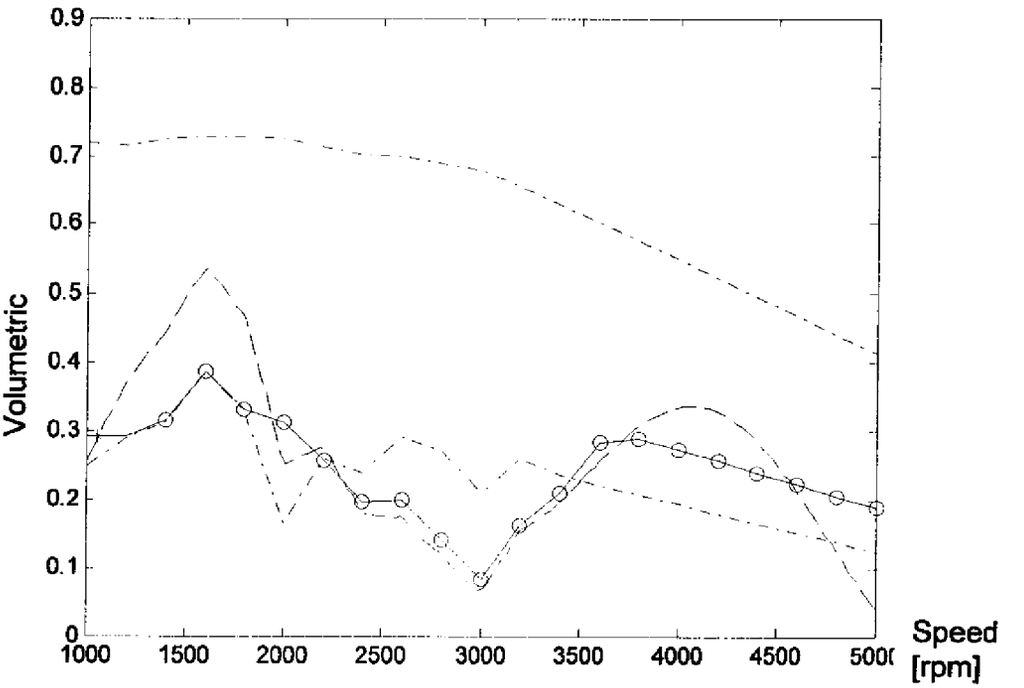

FIG. 14d

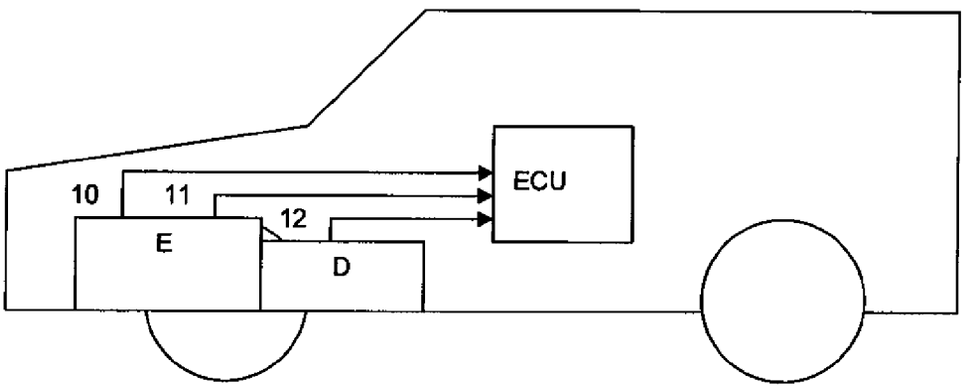

FIG. 15

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 07 10 4811

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
X	DE 101 40 376 A1 (FORD GLOBAL TECH INC [US]) 14 March 2002 (2002-03-14) * page 2, paragraphs 9,11 * -----	1,3,12, 19	INV. F02D41/14 F02D41/24
E	EP 1 772 611 A (DELPHI TECH INC [US]) 11 April 2007 (2007-04-11) * column 2, line 33 - line 34 * * column 2, line 46 - line 49 * -----	1,3,12, 19	
			TECHNICAL FIELDS SEARCHED (IPC) F02D

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
Munich	17 August 2007	Jackson, Stephen

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 07 10 4811

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

17-08-2007

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
DE 10140376 A1	14-03-2002	US 6363317 B1	26-03-2002
EP 1772611 A	11-04-2007	US 2007078588 A1	05-04-2007

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Non-patent literature cited in the description

- **VOGT, M. ; MÜLLER, N. ; LSERMANN, R.** On-Line Adaptation of Grid-Based Look-up Tables Using a Fast Linear Regression Technique. *Journal of Dynamic Systems, Measurement, and Control,* December 2004, vol. 126 **[0056]**
- **ÅSTRÖM, K.J. ; WITTENMARK, B.** Adaptive Control. Addison-Wesley Publishing Company, 1989 **[0059]**
- **HEISS, M.** Online Learning or Tracking of Discrete Input-Output Maps. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-PART A: SYSTEMS AND HUMANS,* September 1997, vol. 27 (5 **[0070]**
- **MILTON, J.S. ; ARNOLD, J.C.** *Introduction to Probability and Statistics,* 1995 **[0094]**
- **RAW LINGS, J.O. ; PANTULA, S.G. ; DICKEY, D.A.** Applied Regression Analysis - A Research Too. Springer-Verlag, 1998 **[0103]**