Associative Datafields in Automotive Control
by Manfred Schmitt¹, Thorsten Ulrich² and Henning Tolle³

ABSTRACT

This work presents an associative datafield structure that has been developed for automotive control applications by the Institute of Control Engineering, Department of Control Systems Theory and Robotics at the Technical University of Darmstadt and the Robert Bosch GmbH. In contrast to the state-of-the-art lattice-like datafields, the new system permits the modelling of multi-dimensional non-linear process and/or controller characteristics with respect to computational performance and storage capacity provided by automotive control units. Furthermore, the associative datafield allows the compensation of wear and tear and manufacturing tolerances by learning, i.e., on-line adaptation of its contents. The paper describes the principles of the associative datafield and different update algorithms. Results of simulations using realistic data from car engines are discussed.

Keywords—associative access, scattered data interpolation, real-time capability, local structure adaptation

I. INTRODUCTION

Current regulations and customer requirements concerning exhaust gas emission, road safety and comfort have led to an increasing demand for advanced control systems in automotive engineering. As a result of this trend, engine management systems, such as ignition and injection control, idle speed control and cylinder selective knock control have been developed. Safety systems like anti-lock and traction control are also established today. These systems are highly dependent on the non-linear characteristics of the combustion engine and therefore require non-linear control strategies. Since motor vehicles are subject to wear and tear, the behaviour of the controlled processes is time-variant. That is why long-term stability and constant control quality can only be achieved using self-adaptive, i.e., learning control systems.

Despite of the fact that control units are equipped with high-performance microcontrollers, computational power and storage capacity still restrict the complexity of control strategies. This leads to the necessity of very efficient internal representations of non-linear process and/or controller behaviour. An approach to this problem is the utilization of datafields, which represent a non-linear function by a set of discrete data points. The input of a datafield is a vector that is comprised of the current values of the variables which describe the state of the process. This vector is called the setpoint in this context. The output, i.e., the estimated response of the process or the value of the control variable, is calculated by interpolating between a number of data points.

II. LATTICE-LIKE DATAFIELDS

The state-of-the-art datafield-type which today is often applied in automotive control units is a lattice-like structure. Datapoints are located on a n-dimensional grid which defines the topological neighborhood relations. Because of the regular positioning of datapoints the coordinates in the n-dimensional input space need not explicitly be stored. For a given setpoint \( \mathbf{z} = (z_1, z_2, ..., z_n)^T \), \( 2^n \) datapoints which enclose the setpoint can be found by a simple address calculation. Figure 1 illustrates the distribution of datapoints in a two-dimensional input space.

![Fig. 1. Datapoint distribution of a two-dimensional lattice-like datafield. The regular datapoint positioning allows the calculation of the memory location of datapoints without coordinate storage. The \( 2^2 = 4 \) datapoints used for interpolation at the given setpoint are marked by solid lines.](image)

Because of the simple mapping between the input space and memory locations, the computational effort involved in selecting a set of datapoints for output interpolation is very small. This leads to short response times which allow real-time operation in automotive applications. Simulations on a SUN 4/80 workstation have proved an average response time of about 50 \( \mu \)sec for a datafield with a two-dimensional input space and 120 datapoints.

On the other hand, there are some essential disadvantages of the lattice-like structure which prevent the implementation of advanced learning control systems:

- Since the density of datapoints can hardly be adjusted to the characteristics of the represented function, memory expense grows exponentially with the input space dimension. Practical applications therefore consider only two input variables.
- Reverse access, i.e., access with swapped input and output variables, is difficult to perform and requires higher computational effort than normal access.
- Locations of datapoints are prescribed by the n-dimensional grid. For this reason datafield update can solely be performed by modifying output values of datapoints. It is impossible to insert new datapoints at arbitrary locations or to modify positions of existing datapoints.

However, the modeling of multi-dimensional functions and the capability of on-line adaptation are essential for the implementation of advanced control concepts. The inclusion of previously ignored or roughly compensated variables improves control quality and allows the integration
III. THE ASSOCIATIVE DATAFIELD

To overcome the drawbacks of lattice-like datafields, we are developing a new datafield structure which allows arbitrary data point distribution ("scattered data"): Each data point is considered as a complete information unit defined by $n$ coordinates in the $n$-dimensional input space and one or more output variables. Access to the datafield is carried out in an associative manner. For a given $n$-dimensional input vector (setpoint), $(n+1)$ data points are associated and used for interpolation of the output value (figure 2).

![Fig. 2. Example of a two-dimensional datafield with arbitrary data point distribution ("scattered data"). In contrast to a lattice-like structure, storage of positions and output values of each data point is required, since no topological neighborhood information is implied. The $(n+1) = 3$ datapoints used for interpolation at the given setpoint are marked by solid lines.]

A. The Association Algorithm

"Association of datapoints" means selection of a subset of datapoints for output interpolation at a given setpoint based on heuristic similarity criteria. These criteria are:

1. the distance between a datapoint and the setpoint,
2. the distribution of datapoints relative to the setpoint.

The first criterion assumes that the reliability of a datapoint decreases with the growth of distance. In general, the distance between two points $\mathbf{x}_A = (x_{A1}, \ldots, x_{An})^T$ and $\mathbf{x}_B = (x_{B1}, \ldots, x_{Bn})^T$ in a normalized $n$-dimensional space is given by the Minkowsky-Metric of order $p$:

$$ L_p(\mathbf{x}_A, \mathbf{x}_B) = \left( \sum_{j=1}^n |x_{Aj} - x_{Bj}|^p \right)^{\frac{1}{p}} $$

For $p = 2$ the Euclidean distance is obtained. The computational effort (calculation of the square root) involved in this metric is relatively high. That is why we use the linear ($p = 1$) Minkowsky-Metric, which is known as the "City-Block-Distance" [1]. Figure 3a shows the lines of constant distance to a setpoint in the two-dimensional input space which result from Euclidean and City-Block-distance measurement.

The second criterion used for datapoint association considers the distribution of the selected datapoints relative to the setpoint. Again, this criterion is based on a heuristic assumption: The probability to achieve a high-quality representation of the exact output value at the setpoint increases, if the setpoint is surrounded by the selected datapoints. The $n$-dimensional area which is enclosed by a set of $(n+1)$ points can be defined as their cartesian or as their convex frame. Figure 3b illustrates these definitions.

B. Output Interpolation

The selection of $(n+1)$ datapoints allows the computation of a $n$-dimensional hyperplane, i.e. a linear output interpolation. In order to achieve short response times, we have implemented an approximately linear interpolation rule which is given by the formula

$$ y(\mathbf{z}_{\text{exp}}) = \left( \frac{\sum_{i=1}^{n+1} \left( \frac{1}{\prod_{k=1, k 
eq i}^{n+1} L_p(\mathbf{z}_{\text{exp}}, \mathbf{z}_k)} \right)}{\sum_{i=1}^{n+1} \left( \prod_{k=1, k 
eq i}^{n+1} L_p(\mathbf{z}_{\text{exp}}, \mathbf{z}_k) \right)} \right) $$

where $\mathbf{z}_k$ denotes the positions of the $(n+1)$ selected datapoints the output values of which are described by $y_k$. The setpoint is referred to as $\mathbf{z}_{\text{exp}}$. $L_p$ signifies the distance between the setpoint and the positions of the $(n+1)$ selected datapoints according to equation (1). The interpolated surface that results from this rule is illustrated in figure 4.

C. Efficient Implementations

The selection of datapoints based on distance and distribution criteria requires the examination of all datapoints in...
memory. Using conventional computer architectures, this task must be performed sequentially. Thus, the growth of the response time is proportional to the total number \( N \) of datapoints. Real-time capability can therefore only be achieved for small \( N \). The implementation of associative datafields on the basis of the computational power provided by automotive control units therefore necessitates efficient strategies to accelerate the associative access.

A profitable approach to this problem is the **segmentation of the input space** [2]. The idea of this concept is the definition of distinct areas (segments) in the input space, each of which contains a subset of datapoints. The association is carried out in a two-stage hierarchical algorithm. In the first step, the segment which contains the current setpoint is identified. The examination of the datapoints which are assigned to this segment takes place in the second step. The segments can either be hypercuboids or multi-dimensional feature areas. The datafield structure which results from a hypercuboid-segmentation is called AMOS (**A**ssociative Memory with Orthogonal Segmentation). In case of a two-dimensional input space, the rectangular segments are shown in figure 5.

![Fig. 5. Example of AMOS: Associative datafield with two-dimensional input space which is subdivided in 9 rectangles.](image)

The identification of the hypercuboid which encloses the current setpoint is similar to an access to a lattice-like datafield. Thus, the AMOS structure promises short response times, since there is only little effort involved in the segment-identification. The average response time is given as:

\[
T \sim c + \frac{N}{m} \tag{3}
\]

In this relationship, \( N \) denotes the total number of datapoints, \( m \) denotes the number of segments. The average response time is proportional to the average number of datapoints per segment \( \frac{N}{m} \) plus a constant effort \( c \) involved in the selection of the current segment. If the setpoint is located close to one of the boundaries \( b_i(x_j) \), it might be impossible to find \( N + 1 \) enclosing datapoints. To satisfy the distribution criterion, datapoints which are assigned to adjacent segments must be considered. The additional effort necessary in these situations is neglected in (3).

The limiting parameter for real-time capability is not the average but the maximum computational effort involved in datafield accesses. Thus, the response time of the datafield should be constant, i.e., independent from the location of the setpoint. A datafield with input space segmentation can meet this requirement if each segment contains nearly the same number of datapoints. In figure 5 the segments contain either 3 or 4 datapoints. However, if the variance of datapoint-density is higher than in the example above, i.e., if there are clusters of datapoints and/or datapoints are diagonally distributed in the input space, the AMOS-structure cannot achieve a nearly constant number of datapoints per segment.

Since datafields with pronounced clusters of datapoints allow efficient modelling of multi-dimensional non-linear functions, we have developed a second type of input space segmentation. The idea of this approach is to classify datapoints on the basis of their location in the \( n \)-dimensional input space, i.e., datapoints with similar positions are assigned to the same segment. The center of all datapoints assigned to one segment is used as a reference point for identifying this segment. Figure 6 shows an example of a datafield with a two-dimensional input space, which is subdivided in 5 segments. Segments are also called **feature areas** in this context, because the segmentation is based on the feature of location.

![Fig. 6. Example of AMNS: Associative datafield with two-dimensional input space which is subdivided in 5 feature areas.](image)

The selection of the current segment is carried out by calculating the distances between the current setpoint and all segment-reference points. In fact, the same algorithm which is applied for datapoint-selection can be used for segment-identification. Consequently, the \( (n + 1) \) nearest segments are selected. The datapoints which are assigned to the nearest segment-reference point are examined in the second step. The remaining \( n \) segments are additionally considered in case that the datapoints not in the first segment do not enclose the setpoint, i.e., do not satisfy the criterion of distribution. The fundamental operation of the associative access to the datafield is similar to that of neural networks used for pattern-recognition (nearest-neighbourhood classification, [1]). Therefore, the datafield structure that results from an input space segmentation in \( n \)-dimensional feature areas is called AMNS (**A**ssociative Memory with Neural **S**egmentation). The segment-reference points are referred to as **neurons**. The average response time of AMNS is proportional to the number of segments \( m \), which have to be examined in the first step, and to the average number of datapoints per segment \( \frac{N}{m} \), which determines the computational effort of the second step.

\[
T \sim m + \frac{N}{m} \tag{4}
\]

The optimal number of segments is therefore given as:

\[
\frac{\partial T}{\partial m} = 0 \Rightarrow m_{opt} = \sqrt{N}, \quad \frac{\partial^2 T}{\partial m^2} = \frac{2N}{m^3} > 0 \tag{5}
\]

The shortest response time is achieved if the number of segments is equal to the square root of the number of datapoints.
D. The engine torque model - an automotive application

As an example of an automotive application of associative datafields, the modelling of the torque $M$ provided by a combustion engine is discussed. Significant input variables are the position of the throttle valve $\alpha_{thr}$ and the number of revolutions per minute $n$.

The engine torque model $M = f(\alpha_{thr}, n)$ is needed for traction control systems, which require information about the current torque or the optimal throttle-valve position. The input space of this system is comprised of only two variables. As far as memory expense is concerned, a lattice-like datafield would therefore be a suitable choice. However, the necessity of reverse datafield queries using $n$ and $M$ as inputs and $\alpha_{thr}$ as output as well as the necessity of efficient on-line adaptation for the compensation of wear and tear, makes the associative datafield adequate for this application. Figure 7 shows the interpolated surface of an associative engine-torque datafield consisting of $N = 64$ datapoints.

![Fig. 7. Interpolated surface of an associative datafield with 64 datapoints representing an engine-torque model.](image)

IV. ON-LINE ADAPTATION OF DATAFIELDS

Automotive applications put superior requirements to control systems. High control quality must be achieved in order to comply with regulations concerning exhaust gas emissions or to implement safety-related functions such as anti-skid or traction-control. Furthermore, the system must be able to cope with fast changing operational conditions (setpoints). To meet these requirements, automotive control systems often combine open- and closed-loop control concepts [3]. The block diagram of a control system of this type is illustrated in figure 8. The control variable $u$, which manipulates the plant, is composed of a datafield output $u_D$ and a controller output $u_C$. The datafield represents a nonlinear open-loop control strategy. It allows fast reaction to changing setpoints, since the corresponding datafield output is available spontaneously. The controller implements a linear control algorithm and guarantees accuracy in steady-state and slow transient conditions. Besides, disturbances $Z$, that affect the plant, are suppressed by closed-loop control.

Datafields in state-of-the-art applications only represent the steady-state behaviour of the controlled process. Generally, dynamic process characteristics can be considered in the datafield if past values of the plant input and output signals are used as additional inputs of the datafield. For this purpose a short-term memory (STM in figure 8) must be implemented [4].

Since automotive processes are subject to wear and tear, their characteristics are time-variant. However, the control strategy, which is represented by the datafield and the linear control algorithm, is optimized for specific process characteristics. Thus, control quality will decrease continuously with growing operation time. This problem can be solved if either the linear control algorithm or the datafield is adapted to the time-variant process. Our approach follows the concept of the learning control loop introduced by W.T. Miller [5], which uses a linear time-invariant controller and an adaptive associative memory system. In the learning control system illustrated in figure 8, the time-invariant controller provides a teacher signal for datafield adaptation. The control variable $u$ which is corrected by the controller output $u_C$ is fed back to the datafield and used for updating its contents.

A. Local Structure Adaptation

The adaptation of the datafield to time-variant process behaviour requires a local update-strategy. The variation of the characteristics of the process is probably different for each operational condition (setpoint). Therefore, a subset of datapoints must be manipulated according to the teacher signal and the corresponding setpoint-data. Instead of a global transformation which is applied to the output of the datafield, the datafield’s structure has to be modified.

Though the teacher signal is provided at every access to the datafield, datafield update cannot be performed each time, because the computational effort involved in structural manipulation is too high. Furthermore, datafields in today’s automotive applications do not consider dynamic process behaviour. Therefore, the controller output carries reliable update-information only in steady-state conditions or phases of slow setpoint movement. Since sensor-signals are superposed with noise in real-world applications, an appropriate filtering has to be implemented. With respect to steady-state datafields, we have developed the following procedure:

While the setpoint is continuously located in one area of the input space, the mean values of the input variables (setpoint-data) and of the teacher signal are computed. Alternatively, first order low-pass filters can be applied. The adaptation of the datafield is invoked by a rapid setpoint movement. The expansion of the area within which changing setpoints are considered to represent steady-state operation depends on the local density of datapoints. The update-algorithm is triggered if the setpoint leaves the range of validity of two adjacent groups of datapoints, i.e. $(n + 2)$
Update algorithms for the associative datafield can be based on two different strategies:

1. Manipulation of existing datapoints.
2. Insertion of new datapoints.

In the following section the SAND-Algorithm, which affects existing datapoints by applying an inverted interpolation-rule, is presented. Section IV-A.2 deals with the MILL-Strategy that makes use of the possibility to insert datapoints at arbitrary locations.

A.1 The SAND-Algorithm

At the end of a phase of steady-state operation, average or low-pass filtered values of the setpoint variables and of the teacher signal are available for datafield update. In the following, \( z_c \) signifies the mean (or low-pass filtered) setpoint. The corresponding value of the teacher signal is denoted by \( \overline{u} \). Since \( z_c \) is typically not identical with the location of an existing datapoint, the algorithm must extrapolate the update information upon one or more datapoints. Because the reliability of the teacher information decreases with growing distance to \( z_c \), it is plausible to manipulate a number of datapoints that are located close to \( z_c \). This update strategy is therefore called SAND (Structure Adaptation of Nearest Datapoints). We have implemented several variations of SAND which affect a different number of datapoints and apply different formulas for datapoint manipulation [7]. The implementation which has proved to be most efficient uses the association algorithm (section III-A) to select \((n+1)\) datapoints \( d_i \) and modifies their output values \( y_i \). The formula used for datapoint adaptation is based on the idea that the degree of modification of a datapoint \( d_i \) should be antiproportional to the distance between its location \( z_c \) and the setpoint \( z_c \). This strategy leads to an adaptation-rule that can be interpreted as the inversion of the interpolation-rule (2). With utilization of the definitions

\[
\sum_{i=1}^{n+1} w_i = \prod_{j=1, j \neq i}^{n+1} L_p(z_c, z_j) \quad i = 1, \ldots, n+1
\]

\[
\sum_{i=1}^{n+1} w_i^2
\]

\[
\delta = \left( \sum_{i=1}^{n+1} w_i \right)^{-1} \left( \sum_{i=1}^{n+1} w_i^2 \right) \quad i = 1, \ldots, n+1
\]

\[
\Delta y_i = \gamma \cdot \delta \cdot w_i \cdot e(z_c)
\]

\[
y_i^{\text{new}} = y_i^{dl} + \Delta y_i, \quad i = 1, \ldots, n+1
\]

\[
e(z_c) \text{ signifies the difference between the teacher signal } \overline{u} \text{ and the response value } \overline{y}(z_c) \text{ before the update is performed:}
\]

\[
e(z_c) = \overline{u} - \overline{y}(z_c)
\]

The gain \( \gamma \) in formula 8 determines the degree of datapoint modification. If \( \gamma \) is chosen to be 1.0, the interpolated output value at the setpoint \( z_c \) will exactly match with the teacher information \( u \) after the update. However, simulations with noisy signals have proved, that a gain \( \gamma < 1 \) leads to higher robustness and attenuated convergence of the algorithm. It can be shown that the adaptation-rule which is given by equations 6, 7 and 8 is equivalent to the normalized LMS-algorithm introduced by Heiss [8] for the adaptation of one-dimensional input-output maps. Criteria for the convergence of the algorithm are also derived in [8].

A.2 The MILL-Algorithm

The associative datafield allows the insertion of datapoints at arbitrary locations in the input space. The mean (or low-pass filtered) values of the controller output (teacher signal) and the setpoint variables can be considered as a new datapoint that represents the current steady-state behaviour in this particular operational condition. Extrapolation of the teacher information, as it is carried out by the SAND-algorithm, is not required.

Since the amount of memory that is provided for datafield storage is limited, a new datapoint can only be inserted if an existing one is deleted simultaneously. To maintain overall-accuracy, an efficient update algorithm therefore must select the datapoint which carries the smallest amount of information. This datapoint is overwritten when the next datafield adaptation is performed. The update strategy based on this approach is called MILL (Minimum Information Loss Learning).

The implementation of the MILL-algorithm requires a definition of the amount of information a datapoint carries. Our concept is based on the assumption that significant differences between the function represented by the datafield and the actual process behaviour only occur in small partitions of the input space. The majority of datapoints are considered to represent the process characteristics almost exactly. The amount of information \( I(d_i) \) a datapoint \( d_i \) carries can therefore be defined as the difference between its output value \( y_i \) and the interpolated value at the position of \( d_i \) that is obtained if \( d_i \) is deleted:

\[
I(d_i) = |y_i - \overline{y}(z_c)|
\]

In this formula \( z_c \) denotes the location of \( d_i \) in the n-dimensional input space, \( \overline{y}(z_c) \) denotes the datafield's response without consideration of the datapoint \( d_i \). Figure 10 illustrates this definition for two datapoints \( d_1 \) and \( d_2 \). The error that occurs if \( d_1 \) is deleted is greater than the corresponding difference at \( d_2 \). Thus, the amount of information represented by \( d_2 \) is considered to be larger than that carried by \( d_1 \).
Fig. 10. Definition of the amount of information that is represented by single datapoints.

From all $N$ datapoints the datapoint $d_k$, which carries the smallest amount of information, is selected for deletion:

$$d_k = \arg \min_{d_i \in \{1, \ldots, N\}} \{I(d_i)\} \quad i = 1, \ldots, N \quad (12)$$

When the update algorithm is invoked, a decision must be taken whether or not the insertion of a new datapoint and the simultaneous deletion of $d_k$ improves datafield accuracy. Therefore the absolute error $|e(d_k)|$ according to (10) has to be compared with the amount of information $I(d_k)$ that is lost if $d_k$ is overwritten.

The average absolute error of the datafield, which is achieved if the new datapoint overwrites $d_k$, will be reduced if the following condition is true:

$$|e(d_k)| > I(d_k) \quad (13)$$

Otherwise, the datafield is not manipulated.

A.3 Simulations of Structural Datafield Adaptation

To demonstrate the efficiency of the update algorithms SAND and MILL, a simulation of a structural datafield adaptation is presented. The simulation starts with a datafield that represents the engine torque model (figure 7). The desired function $f^{des}$ that should be approximated after a period of adaptation, is defined as the sum of the function $M = f^{act}(\alpha_{thr}, n)$, which is represented before learning is performed, and a two-dimensional sinusoidal function $\Delta M = f^s(\alpha_{thr}, n)$. The amplitude of $f^s$ is chosen to be 10% of the full range of the output variable $M$.

$$f^{des}(\alpha_{thr}, n) = f^{act}(\alpha_{thr}, n) + f^s(\alpha_{thr}, n) \quad (14)$$

$$f^s(\alpha_{thr}, n) = 0.1 \cdot \sin(\alpha_{thr}) \cdot \sin(n)$$

The update information used in this simulation is generated by means of the analytic function $f^s$, whereas realistic signals (\alpha_{thr} and n) from a car engine are used as inputs (setpoints).

Both adaptation strategies lead to a good approximation of the desired function in this simulation [6]. The performance of the MILL-algorithm is slightly higher: The average absolute error is reduced by 83% of the initial value, whereas the SAND-algorithm achieves an error reduction of 77%.

On the other hand, the computational effort required by the MILL-strategy is considerably higher than that of the SAND-algorithm. The operations which must be performed applying the adaptation rule (8) are similar to those for output interpolation.

V. Conclusion

New efficient schemes for data representation for automotive applications are proposed. These scattered datafields can represent non-linear characteristics with less datapoints than the up to now used lattice-like datafields. Therefore, the new structure allows the application of datafields of higher dimension than the today common value two. By means of the proposed update algorithms SAND and MILL, the scattered datafields can also be adapted easily if required by changes of the non-linear characteristics.

REFERENCES


* These study thesis are only available at the library of the Institute of Control Engineering at the Technical University of Darmstadt.