J Math Chem (2011) 49:1092–1114
DOI 10.1007/s10910-011-9800-0

ORIGINAL PAPER

A Matrix Based Indexing HDMR method for multivariate data modelling

M. Alper Tunga

Received: 13 August 2010 / Accepted: 19 January 2011 / Published online: 1 February 2011

Abstract Modelling multivariate data of real life problems from engineering, chemistry, physics, mathematics or other related sciences, in which function values are known only at arbitrarily distributed points of the problem domain, is an important and complicated issue since there exist mathematical and computational complexities in the analytical structure construction process coming from the multivariance. The Plain High Dimensional Model Representation (HDMR) method expresses a multivariate problem in terms of less-variate problems. In this work, a Matrix Based Indexing HDMR method is developed to make the Plain HDMR philosophy employable for the multivariate data partitioning process. This new method will have the ability of dealing with less-variate data sets by partitioning the given data set into univariate, bivariate and trivariate data sets. Interpolating these partitioned data sets will construct an approximate analytical structure as the model of the given multivariate data modelling problem.

Keywords High dimensional model representation · Multivariate data modelling · Interpolation · Multidimensional problems · Approximation

1 Introduction

A multivariate data set including N number of independent variables (generally $N>3$) and m number of nodes with the associated function values is given to model the given system in data modelling problems of real life. Dealing with less-variate data sets such as univariate, bivariate or trivariate ones is much better than dealing with a multivariate

M. A. Tunga (✉)
Engineering Faculty, Software Engineering Department, Bahçeşehir University, Beşiktaş, 34349 İstanbul, Turkey
e-mail: alper.tunga@bahcesehir.edu.tr

set since this multivariance causes high mathematical and computational complexities especially in standard mathematical methods and computer based applications. Hence, one way to avoid these undesirable cases is to take divide–and–conquer methods into consideration. The mechanism constructed through this type of a method stands for multivariate data partitioning process. One of these methods is Plain (or Sobol) High Dimensional Model Representation (HDMR) Method [1] which is about representing a given multivariate function in terms of less–variate functions with a finite expansion.

The Plain HDMR method is then used by Professor Rabitz and his group in various research areas. Some HDMR based methods developed by this group are ANOVA-HDMR [2] where ANOVA is an acronym for analysis of variance used in statistics, cut-HDMR [3] and RS-HDMR [4] which is about random sampling.

Professor Metin Demiralp and his group also developed various HDMR based algorithms [5–10]. These methods were applied to multivariate interpolation problems, algebraic eigenvalue problem modelling, Schrödinger's equation, optimal control of harmonic oscillator, exponential matrix evaluation, evolution operators and so on.

Many other scientists developed various HDMR based methods for different research areas [11–20].

The Plain HDMR method can also be used in the development of a multivariate data partitioning algorithm [21–25]. However, because of the structure of this algorithm, the Plain HDMR method needs an orthogonal geometry in the given domain of the data modelling problem. That is, the function values must be known at all possible nodes of the given multivariate data set to be partitioned. In real life applications, the function values are usually given only at arbitrarily distributed nodes of the problem domain. This results in a need for another HDMR based method need for the multivariate data partitioning process. The first attempt to this end was the Generalized HDMR Method [22] and the second was the Lumping HDMR Method [23–25]. In fact, these two methods have disadvantages in their algorithms.

The Generalized HDMR Method works well for the modelling problems having purely additive nature. As the additivity dominance decreases and the multiplicativity features become dominant, the performance of the method gets worse dramatically. Additionally, the method requires solving a linear equation system and depending on the given data structure the equation system may turn out to be unsolvable. In addition to this, to determine the general structures of the bivariate and the higher variate Generalized HDMR components is very difficult since the possible structures will include integral equation systems. Of course, this process will increase the mathematical complexity of the method dramatically and prevent the performance improvement of the method by using the higher variate components.

The Lumping HDMR method has two important algorithms in its structure, one for the modelling part and the other one for the testing part. The performance of the modelling part is quite well while that of the testing part is average because of its unstable structure.

The aim of this work is to develop a new HDMR based method to bypass these disadvantages and to have more general, algebraic, stable and efficient method for the multivariate data partitioning process. Our new method has the same philosophy with the Lumping HDMR method [23] and allows us to produce an orthogonal indexing scheme and makes a one-to-one mapping between this new set and the given multi-

variate data set. Since we have an orthogonal geometry after the mentioned mapping process, we can partition the new multivariate data set into univariate, bivariate or trivariate data sets by using the Plain HDMR philosophy and determine an approximate analytical structure. This analytical structure has the independent variables of the index scheme and the mentioned mapping lets us know the locations of each node of the given data set in the index scheme. In this way, we can use that analytical structure for the given multivariate data set nodes. However, we do not know the index space locations of the nodes at which the functions values are not given. To find these function values is our main task. This point is the main problem of the Indexing based HDMR methods. For this purpose, a matrix representation mechanism is imposed into the method to define the relationship between the given original multivariate data set and the index data set appearing in the standard Indexing based HDMR Method [24]. The main issue of our new method is to find out the elements of this matrix and to bypass the main problem of the Indexing based HDMR methods developed before [24,25]. An overdetermined linear equation system solution is needed for this purpose. This work gives the details of this solution and the new form of the Indexing HDMR method. The new model is named as " Matrix Based Indexing HDMR (MI-HDMR)" method.

The paper is organized as follows. The second section covers the Plain HDMR method while the details of the Matrix Based Indexing HDMR Method are given in the third section. The fourth section includes several numerical implementations to show the performance of our new method and the concluding remarks are in the last section.

2 The plain HDMR method

The Plain HDMR method represents a given multivariate function in terms of less-variate functions. The method has a finite expansion for this purpose and the general structure of this expansion is given as follows

$$f(x_1,\ldots,x_N) = f_0 + \sum_{i_1=1}^{N} f_{i_1}(x_{i_1}) + \sum_{\substack{i_1,i_2=1 \\ i_1<i_2}}^{N} f_{i_1 i_2}(x_{i_1},x_{i_2}) + \cdots + f_{1\ldots N}(x_1,\ldots,x_N) \tag{2.1}$$

where $f(x_1,\ldots,x_N)$ is an square integrable multivariate function and the right hand side terms are the constant, univariate, bivariate and higher variate HDMR components of this function. The main task of this algorithm is to uniquely determine the general structures of these right hand side components. The following vanishing conditions are defined for this determination process

$$\int_{a_{i_k}}^{b_{i_k}} dx_{i_k} W_{i_k}(x_{i_k}) f_{i_1\ldots i_s}(x_{i_1},\ldots,x_{i_s}) = 0, \quad i_1 \le i_k \le i_s \tag{2.2}$$

and it is known that these vanishing conditions correspond to the following orthogonality conditions

$$\int_{a_1}^{b_1} dx_1 W_1(x_1) \cdots \int_{a_N}^{b_N} dx_N W_N(x_N) f_{i_1 \ldots i_s}(x_{i_1}, \ldots, x_{i_s}) f_{j_1 \ldots j_\ell}(x_{j_1}, \ldots, x_{j_\ell}) = 0$$
$$(s \neq \ell) \vee [(i_1 \neq j_1) \vee \ldots \vee (i_s \neq j_s)] \tag{2.3}$$

which means that any two of HDMR components are mutually orthogonal in the Hilbert space of functions. Here, $W(x_1, \ldots, x_N)$ is the product type weight function having a normalization criterion for easy determination of the HDMR components.

$$W(x_1, \ldots, x_N) \equiv \prod_{j=1}^{N} W_j(x_j),$$
$$\int_{a_j}^{b_j} dx_j W_j(x_j) = 1, \quad x_j \in \left[a_j , b_j \right], \quad 1 \leq j \leq N \tag{2.4}$$

The following operators are defined to obtain the general structures of the constant and univariate HDMR components

$$\mathscr{I}_0 F(x_1, \ldots, x_N) \equiv \int_{a_1}^{b_1} dx_1 W_1(x_1) \cdots \int_{a_N}^{b_N} dx_N W_N(x_N) F(x_1, \ldots, x_N) \tag{2.5}$$

$$\mathscr{I}_{k_1}(x_1, \ldots, x_N) \equiv \int_{a_1}^{b_1} dx_1 W_1(x_1) \cdots \int_{a_{k_1-1}}^{b_{k_1-1}} dx_{k_1-1} W_{k_1-1}(x_{k_1-1})$$
$$\times \int_{a_{k_1+1}}^{b_{k_1+1}} dx_{k_1+1} W_{k_1+1}(x_{k_1+1}) \cdots \int_{a_N}^{b_N} dx_N W_N(x_N) F(x_1, \ldots, x_N) \tag{2.6}$$

where $1 \leq k_1 \leq N$ and $F(x_1, \ldots, x_N)$ is an arbitrary square integrable function. Other operators for the higher variate HDMR components determination process can be defined similarly.

The general structures of HDMR components are obtained by using the abovementioned operators under the vanishing conditions and the weight function given in (2.2)

and (2.4), respectively

$$\begin{aligned} f_0 &= \mathscr{I}_0 f(x_1, \ldots, x_N) \\ f_{i_1}(x_{i_1}) &= \mathscr{I}_{i_1} f(x_1, \ldots, x_N) - f_0 \\ f_{i_1 i_2}(x_{i_1}, x_{i_2}) &= \mathscr{I}_{i_1 i_2} f(x_1, \ldots, x_N) - f_{i_1}(x_{i_1}) - f_{i_2}(x_{i_2}) - f_0 \\ f_{i_1 i_2 i_3}(x_{i_1}, x_{i_2}, x_{i_3}) &= \mathscr{I}_{i_1 i_2 i_3} f(x_1, \ldots, x_N) - f_{i_1 i_2}(x_{i_1}, x_{i_2}) - f_{i_1 i_3}(x_{i_1}, x_{i_3}) \\ &\quad - f_{i_2 i_3}(x_{i_2}, x_{i_3}) - f_{i_1}(x_{i_1}) - f_{i_2}(x_{i_2}) - f_{i_3}(x_{i_3}) - f_0 \end{aligned} \tag{2.7}$$

where $1 \leq i_1 < i_2 < i_3 \leq N$. We tend to use at most trivariate HDMR components in order not to increase the computational and mathematical complexities.

3 The Matrix Based Indexing HDMR (MI-HDMR) method

In this work, it is assumed that the function values are known at the arbitrarily distributed nodes of the problem domain and it is asked to construct a mathematical model to that problem. This model will allow us to evaluate the unknown function values at the other nodes of the domain. Hence, the given multivariate data set with the associated function values is named as " training data set" while the data set without function values is named as " testing data set".

The training data set structure can be defined as

$$\Theta_s \equiv \left(v_1^{(s)}, \ldots, v_N^{(s)}, \varphi_s\right), \quad \varphi_s \equiv f(v_1^{(s)}, \ldots, v_N^{(s)}), \quad 1 \leq s \leq m \tag{3.1}$$

where m is the number of nodes of the training data set.

The structure of the testing data set is given as follows

$$q_\ell = \left(\mu_1^{(\ell)}, \ldots, \mu_N^{(\ell)}\right), \quad 1 \leq \ell \leq t \tag{3.2}$$

where t is the number of testing nodes.

Since the weight function given in (2.4) is the product of univariate components and the HDMR components are mutually orthogonal, all the function values at all possible nodes of the problem domain are needed in the integration process of the Plain HDMR method. This corresponds to a cartesian product set with the function values at all the nodes of this set, that is, the structure of the data to be modelled has an orthogonal geometry.

This cartesian product set can be defined as

$$\begin{aligned} \mathscr{D} &\equiv \mathscr{D}_1 \times \mathscr{D}_2 \times \cdots \times \mathscr{D}_N \\ \mathscr{D}_j &\equiv \left\{\xi_j^{(k_j)}\right\}_{k_j=1}^{k_j=n_j} = \left\{\xi_j^{(1)}, \ldots, \xi_j^{(n_j)}\right\}, \quad 1 \leq j \leq N \end{aligned} \tag{3.3}$$

where each $\xi_j^{(k_j)}$ and each n_j value stand for a value coming from the domain of each independent variable of the space having an orthogonal geometry and the number of elements of each domain, respectively.

However, in real life problems, the training set does not include the function values at all nodes of the problem domain. These nodes at which the function values are known, are arbitrarily distributed inside the given domain as in (3.1). The mentioned structure has a non-orthogonal geometry. Hence, we need to construct a mechanism to obtain an orthogonal geometry from the given non-orthogonal geometry to apply the Plain HDMR method.

Imposing an indexing space having a cartesian product set which covers an orthogonal geometry and constructing a one-to-one mapping between the original space and the imposed index space like below is the first step of the Indexing based HDMR philosophy [23].

$$\begin{aligned}
\left(v_1^{(1)}, \ldots, v_{N-1}^{(1)}, v_N^{(1)}, \varphi_1\right) &\Longrightarrow \left(\xi_1^{(1)}, \ldots, \xi_{N-1}^{(1)}, \xi_N^{(1)}, \varphi_1\right) \\
\left(v_1^{(2)}, \ldots, v_{N-1}^{(2)}, v_N^{(2)}, \varphi_2\right) &\Longrightarrow \left(\xi_1^{(1)}, \ldots, \xi_{N-1}^{(1)}, \xi_N^{(2)}, \varphi_2\right) \\
&\vdots \\
\left(v_1^{(m-1)}, \ldots, v_{N-1}^{(m-1)}, v_N^{(m-1)}, \varphi_{m-1}\right) &\Longrightarrow \left(\xi_1^{(n_1)}, \ldots, \xi_{N-1}^{(n_{N-1})}, \xi_N^{(n_N-1)}, \varphi_{m-1}\right) \\
\left(v_1^{(m)}, \ldots, v_{N-1}^{(m)}, v_N^{(m)}, \varphi_m\right) &\Longrightarrow \left(\xi_1^{(n_1)}, \ldots, \xi_{N-1}^{(n_{N-1})}, \xi_N^{(n_N)}, \varphi_m\right) \qquad (3.4)
\end{aligned}$$

Here, the indexing is constructed through variables. Of course, indexes can be selected in different ways. The reasons in selecting the indexes as variables are to construct a simple matrix representation structure for the problem since this mapping structure is modelled by a matrix representation in our new method, to easily solve the linear equation system of that matrix representation and to use the similarity metrics effectively in the appropriate training node determination process for each testing node of the problem. These details are given in the rest of this section.

Another important point for this mapping procedure is to specify an appropriate ordering over the nodes of the original grid and then to construct a one-to-one mapping between the original nodes and the nodes of the index space. The proposed procedure for this purpose is to sort the nodes of the original space by function values in ascending or descending order. This is the best solution observed from several tests through several numerical implementations.

The main task for this mapping procedure is to specify the values of $n_1, n_2, \ldots, n_N$ which stand for the number of elements of each independent variable's domain in the index space. For this purpose, the prime factors of m, the number of nodes of the given training set, are evaluated and then these factors are reorganized by taking the problem's number of independent variables into consideration [23]. For instance, when m is 100 and there exist 4 independent variables in the problem, the n_j, $(1 \leq j \leq 4)$ values can be set as $n_1 = 2$, $n_2 = 2$, $n_3 = 5$ and $n_4 = 5$. Of course, this setting is not unique. On the other hand, if there exist 3 independent variables for another problem having the same value for m, then the specification for n_j, $(1 \leq j \leq 3)$ values may

be as $n_1 = 4$, $n_2 = 5$ and $n_3 = 5$. The elements of the domain of each independent variable in the index space are selected as $\xi_j^{(n_j)} \in \{1, 2, \ldots, n_j\}$ where $1 \leq j \leq N$.

The next step is to define a weight function whose general structure is given in (2.4). As we deal with the function values at the nodes of the given domain, the weight function must support this structure. Hence, the linear combinations of Dirac delta functions [26] are inserted into the product type weight function

$$W_j(x_j) \equiv \sum_{k_j=1}^{n_j} \alpha_{k_j}^{(j)} \delta\left(x_j - \xi_j^{(k_j)}\right),$$
$$\sum_{k_j=1}^{n_j} \alpha_{k_j}^{(j)} = 1, \quad x_j \in \left[a_j, b_j\right], \quad 1 \leq j \leq N \tag{3.5}$$

where α coefficients are for giving different importance for each node. The constraint obtained for these coefficients comes from the normalization criterion given in (2.4).

Using the weight function defined in (3.5) and relations given in (2.7), the components starting from the constant ending at the trivariate level can be determined for the data partitioning process.

The constant term is determined as

$$f_0 = \sum_{k_1=1}^{n_1} \sum_{k_2=1}^{n_2} \cdots \sum_{k_N=1}^{n_N} \left(\prod_{j=1}^{N} \alpha_{k_j}^{(j)} \right) f\left(\xi_1^{(k_1)}, \ldots, \xi_N^{(k_N)}\right) \tag{3.6}$$

while the general structure for the univariate components is as follows

$$f_{i_1}\left(\xi_{i_1}^{(k_{i_1})}\right) = \sum_{k_1=1}^{n_1} \cdots \sum_{k_{i_1-1}=1}^{n_{i_1-1}} \sum_{k_{i_1+1}=1}^{n_{i_1+1}} \cdots \sum_{k_N=1}^{n_N} \left(\prod_{\substack{j=1 \\ j \neq i_1}}^{N} \alpha_{k_j}^{(j)} \right) \times f\left(\xi_1^{(k_1)}, \ldots, \xi_{i_1}^{(k_{i_1})}, \ldots, \xi_N^{(k_N)}\right) - f_0 \tag{3.7}$$

where $1 \leq i_1 \leq N$. The bivariate component structure is obtained as

$$f_{i_1 i_2}\left(\xi_{i_1}^{(k_{i_1})}, \xi_{i_2}^{(k_{i_2})}\right) = \sum_{k_1=1}^{n_1} \cdots \sum_{k_{i_1-1}=1}^{n_{i_1-1}} \sum_{k_{i_1+1}=1}^{n_{i_1+1}} \cdots \sum_{k_{i_2-1}=1}^{n_{i_2-1}} \sum_{k_{i_2+1}=1}^{n_{i_2+1}} \cdots \sum_{k_N=1}^{n_N} \left(\prod_{\substack{j=1 \\ j \neq i_1 \wedge j \neq i_2}}^{N} \alpha_{k_j}^{(j)} \right) \times f\left(\xi_1^{(k_1)}, \ldots, \xi_{i_1}^{(k_{i_1})}, \ldots, \xi_{i_2}^{(k_{i_2})}, \ldots, \xi_N^{(k_N)}\right) - f_{i_1}\left(\xi_{i_1}^{(k_{i_1})}\right) - f_{i_2}\left(\xi_{i_2}^{(k_{i_2})}\right) - f_0 \tag{3.8}$$

where $1 \le i_1 < i_2 \le N$.

The following structure is obtained as the general structure of the trivariate components.

$$
\begin{aligned}
& f_{i_1 i_2 i_3}\left(\xi_{i_1}^{(k_{i_1})}, \xi_{i_2}^{(k_{i_2})}, \xi_{i_3}^{(k_{i_3})}\right) \\
& = \sum_{k_1=1}^{n_1} \cdots \sum_{k_{i_1-1}=1}^{n_{i_1-1}} \sum_{k_{i_1+1}=1}^{n_{i_1+1}} \cdots \sum_{k_{i_2-1}=1}^{n_{i_2-1}} \sum_{k_{i_2+1}=1}^{n_{i_2+1}} \cdots \sum_{k_{i_3-1}=1}^{n_{i_3-1}} \sum_{k_{i_3+1}=1}^{n_{i_3+1}} \\
& \quad \times \cdots \sum_{k_N=1}^{n_N} \left(\prod_{\substack{j=1 \\ j \neq i_1 \wedge j \neq i_2 \wedge j \neq i_3}}^{N} \alpha_{k_j}^{(j)} \right) f\left(\xi_1^{(k_1)}, \ldots, \xi_{i_1}^{(k_{i_1})}, \ldots, \xi_{i_2}^{(k_{i_2})}, \ldots, \xi_{i_3}^{(k_{i_3})}, \ldots, \xi_N^{(k_N)}\right) \\
& \quad - f_{i_1 i_2}\left(\xi_{i_1}^{(k_{i_1})}, \xi_{i_2}^{(k_{i_2})}\right) - f_{i_1 i_3}\left(\xi_{i_1}^{(k_{i_1})}, \xi_{i_3}^{(k_{i_3})}\right) - f_{i_2 i_3}\left(\xi_{i_2}^{(k_{i_2})}, \xi_{i_3}^{(k_{i_3})}\right) - f_{i_1}\left(\xi_{i_1}^{(k_{i_1})}\right) \\
& \quad - f_{i_2}\left(\xi_{i_2}^{(k_{i_2})}\right) - f_{i_3}\left(\xi_{i_3}^{(k_{i_3})}\right) - f_0
\end{aligned} \tag{3.9}
$$

These relations obtained in (3.7–3.9) allow us to partition the multivariate training data set into univariate, bivariate and trivariate data sets. The following pairs are the structures of the nodes of the univariate, bivariate and trivariate data sets, respectively.

$$
\begin{aligned}
& \left(\xi_{i_1}^{(k_{i_1})}, f_{i_1}\left(\xi_{i_1}^{(k_{i_1})}\right)\right), \quad \left(\xi_{i_1}^{(k_{i_1})}, \xi_{i_2}^{(k_{i_2})}, f_{i_1 i_2}\left(\xi_{i_1}^{(k_{i_1})}, \xi_{i_2}^{(k_{i_2})}\right)\right) \\
& \left(\xi_{i_1}^{(k_{i_1})}, \xi_{i_2}^{(k_{i_2})}, \xi_{i_3}^{(k_{i_3})}, f_{i_1 i_2 i_3}\left(\xi_{i_1}^{(k_{i_1})}, \xi_{i_2}^{(k_{i_2})}, \xi_{i_3}^{(k_{i_3})}\right)\right) \\
& \xi_{i_1}^{(k_{i_1})} \in \mathscr{D}_{i_1}, \quad \xi_{i_2}^{(k_{i_2})} \in \mathscr{D}_{i_2}, \quad \xi_{i_3}^{(k_{i_3})} \in \mathscr{D}_{i_3}, \quad 1 \le k_{i_1} \le n_{i_1}, \\
& 1 \le k_{i_2} \le n_{i_2}, \quad 1 \le k_{i_3} \le n_{i_3}, \quad 1 \le i_1 < i_2 < i_3 \le N
\end{aligned} \tag{3.10}
$$

Next step is to determine the analytical structures for the abovementioned pairs. For this purpose, the Lagrange coefficients [27] are used. The general structure of these coefficients are as follows

$$
L_{k_{i_1}}(x_{i_1}) \equiv \prod_{\substack{j=1 \\ j \neq k_{i_1}}}^{n_{i_1}} \frac{\left(x_{i_1} - \xi_{i_1}^{(j)}\right)}{\left(\xi_{i_1}^{(k_{i_1})} - \xi_{i_1}^{(j)}\right)} \tag{3.11}
$$

and these coefficients are inserted into the following multinomials in order to obtain the analytical structures of the univariate, bivariate and trivariate HDMR components.

$$
p_{i_1}(x_{i_1}) = \sum_{k_{i_1}=1}^{n_{i_1}} L_{k_{i_1}}(x_{i_1}) f_{i_1}\left(\xi_{i_1}^{(k_{i_1})}\right)
$$

$$
p_{i_1 i_2}(x_{i_1}, x_{i_2}) = \sum_{k_{i_1}=1}^{n_{i_1}} \sum_{k_{i_2}=1}^{n_{i_2}} L_{k_{i_1}}(x_{i_1}) L_{k_{i_2}}(x_{i_2}) f_{i_1 i_2}\left(\xi_{i_1}^{(k_{i_1})}, \xi_{i_2}^{(k_{i_2})}\right)
$$

$$p_{i_1 i_2 i_3}\left(x_{i_1}, x_{i_2}, x_{i_3}\right) = \sum_{k_{i_1}=1}^{n_{i_1}} \sum_{k_{i_2}=1}^{n_{i_2}} \sum_{k_{i_3}=1}^{n_{i_3}} L_{k_{i_1}}(x_{i_1}) L_{k_{i_2}}(x_{i_2}) L_{k_{i_3}}(x_{i_3}) \times f_{i_1 i_2 i_3}\left(\xi_{i_1}^{(k_{i_1})}, \xi_{i_2}^{(k_{i_2})}, \xi_{i_3}^{(k_{i_3})}\right) \tag{3.12}$$

Now, the HDMR approximants can be written in terms of the multinomials given in (3.12) as follows

$$\begin{aligned} s_1(x_1, \ldots, x_N) &= f_0 + \sum_{i_1=1}^{N} p_{i_1}(x_{i_1}) \\ s_2(x_1, \ldots, x_N) &= s_1(x_1, \ldots, x_N) + \sum_{\substack{i_1, i_2=1 \\ i_1<i_2}}^{N} p_{i_1 i_2}(x_{i_1}, x_{i_2}) \\ s_3(x_1, \ldots, x_N) &= s_2(x_1, \ldots, x_N) + \sum_{\substack{i_1, i_2, i_3=1 \\ i_1<i_2<i_3}}^{N} p_{i_1 i_2 i_3}(x_{i_1}, x_{i_2}, x_{i_3}) \end{aligned} \tag{3.13}$$

However, the function whose data is given may not always have an analytical structure. The data interpolation region of the given problem may have certain types of singularities and therefore, a spline interpolation technique is needed to obtain these multinomials given in (3.12). The natural cubic spline interpolation is used [27–29] to determine the analytical structures of the univariate components. The following cubic polynomials are defined

$$S_{i_1}^{(j)}(x_i) = a_0^{(i_1)} + a_1^{(i_1)}\left(x_{i_1} - \xi_{i_1}^{(j)}\right) + a_2^{(i_1)}\left(x_{i_1} - \xi_{i_1}^{(j)}\right)^2 + a_3^{(i_1)}\left(x_{i_1} - \xi_{i_1}^{(j)}\right)^3 \tag{3.14}$$

on each interval, $\left[\xi_{i_1}^{(j)}, \xi_{i_1}^{(j+1)}\right]$ where $1 \le i_1 \le N$ and $1 \le j \le n_{i_1}$. Here, the coefficients $a_0^{(i_1)}$, $a_1^{(i_1)}$, $a_2^{(i_1)}$ and $a_3^{(i_1)}$ are determined through a number of conditions [27–29]. The univariate HDMR approximant, $s_1(x_1, \ldots, x_N)$ given in (3.13), can be obtained by adding these splines to the constant HDMR component.

To determine the analytical structures of the bivariate components, the bicubic spline interpolation is used [27–29]. The bicubic spline function is given as

$$S_{i_1 i_2}^{(j_1, j_2)}(x_{i_1}, x_{i_2}) = \sum_{p=0}^{3} \sum_{q=0}^{3} a_{p,q}^{(i_1, i_2)}\left(x_{i_1} - \xi_{i_1}^{(j_1)}\right)^p \left(x_{i_2} - \xi_{i_2}^{(j_2)}\right)^q \tag{3.15}$$

in each cell

$$R_{i_1 i_2} = \left\{(x_{i_1}), (x_{i_2}) \,\middle|\, \xi_{i_1}^{(j_1)} \le x_{i_1} \le \xi_{i_1}^{(j_1+1)}, \xi_{i_2}^{(j_2)} \le x_{i_2} \le \xi_{i_2}^{(j_2+1)}\right\} \tag{3.16}$$

where $1 \leq i_1, i_2 \leq N$, $1 \leq j_{i_1} \leq n_{i_1}$ and $1 \leq j_{i_2} \leq n_{i_2}$. The coefficients of the function can be evaluated again by taking a number of conditions into consideration [27–29]. When these bicubic splines are added to the univariate HDMR approximant, the bivariate HDMR approximant, given in (3.13) as $s_2(x_1, \ldots, x_N)$ is obtained.

Since the multivariate analysis of spline interpolation becomes more complex when the number of independent variables of the problem increases, the trivariate HDMR approximant is not taken into consideration in the case of spline interpolation is used instead of Lagrange interpolation. Another important point is that the proposed method with spline interpolation will be applied to piecewise continuous functions. A possible discontinuity property in the given piecewise structure needs another HDMR based approach since the intervals including such singularities should be taken into consideration and either appropriate conditions for spline interpolation to the HDMR relations or appropriate HDMR relations to the spline interpolation conditions should be developed. These will be considered in future work.

These univariate, bivariate and trivariate approximants are the approximate analytical structures obtained through the univariate, bivariate and trivariate data sets partitioned by the Indexing HDMR method. The important point here is the elements of these data sets. Because, they are the elements of the index space. Hence, the domains of the independent variables of these approximants come from the index space, not from the original space because of the one-to-one mapping done between these two spaces. This means, we know the locations of the nodes of the original space in the index space and it results in finding the location of each testing node in the index space. The value of the function can be obtained at each testing node since the location of that node in the index space is known.

For this purpose, a matrix representation for the mapping procedure given in (3.4) is constructed. The coefficient matrix, $\mathbf{A}$, includes the nodes of the training data set while $\mathbf{b}$ covers the corresponding index space nodes. The general structure of $\mathbf{A}$ is as follows

$$\mathbf{A} = \begin{bmatrix} \nu_1^{(1)} & \nu_2^{(1)} & \cdots & \nu_{N-1}^{(1)} & \nu_N^{(1)} \\ \nu_1^{(2)} & \nu_2^{(2)} & \cdots & \nu_{N-1}^{(2)} & \nu_N^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \nu_1^{(m-1)} & \nu_2^{(m-1)} & \cdots & \nu_{N-1}^{(m-1)} & \nu_N^{(m-1)} \\ \nu_1^{(m)} & \nu_2^{(m)} & \cdots & \nu_{N-1}^{(m)} & \nu_N^{(m)} \end{bmatrix} \tag{3.17}$$

while the following structure is for $\mathbf{b}$.

$$\mathbf{b} = \begin{bmatrix} \xi_1^{(1)} & \xi_2^{(1)} & \cdots & \xi_{N-1}^{(1)} & \xi_N^{(1)} \\ \xi_1^{(1)} & \xi_2^{(1)} & \cdots & \xi_{N-1}^{(1)} & \xi_N^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \xi_1^{(n_1)} & \xi_2^{(n_2)} & \cdots & \xi_{N-1}^{(n_{N-1})} & \xi_N^{(n_N-1)} \\ \xi_1^{(n_1)} & \xi_2^{(n_2)} & \cdots & \xi_{N-1}^{(n_{N-1})} & \xi_N^{(n_N)} \end{bmatrix} \tag{3.18}$$

The rows of **A** and **b** represents the nodes of the training data set and the indexing scheme, respectively. The columns of these matrices correspond to the independent variables of the problem.

To present the mapping between **A** and **b**, we need to define a linear equation system like $\mathbf{Ax} = \mathbf{b}$. Since the dimensions of **A** and **b** are $m \times N$, the dimension of **x** will be $N \times N$. To obtain **x**, we have to specify the following equation systems

$$\mathbf{Ax}_i = \mathbf{b}_i, \quad 1 \le i \le N, \quad \mathbf{b} = [\mathbf{b}_1 \ \cdots \ \mathbf{b}_N], \quad \mathbf{x} = [\mathbf{x}_1 \ \cdots \ \mathbf{x}_N] \tag{3.19}$$

where each $\mathbf{b}_i$ and $\mathbf{x}_i$ stand for a column of the matrices, **b** and **x**, respectively. Now, we have N number of linear equation systems to construct the structure of the matrix, **x**. However, in real life applications, it is clear that $m >> N$. That is, the number of independent equations are more than the unknowns of the equation system. This means, we have overdetermined equation systems here.

To solve this system, we concatenate the matrix, **A** with dimension $m \times N$ and an identity matrix with dimension $m \times m$. Now, we have a new matrix as a coefficient matrix having a dimension of $m \times (m + N)$ [30].

$$\mathbf{A}' = \begin{bmatrix} \nu_1^{(1)} & \nu_2^{(1)} & \cdots & \nu_{N-1}^{(1)} & \nu_N^{(1)} & 1\,0\cdots 0\,0 \\ \nu_1^{(2)} & \nu_2^{(2)} & \cdots & \nu_{N-1}^{(2)} & \nu_N^{(2)} & 0\,1\cdots 0\,0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots\,\vdots\,\ddots\,\vdots\,\vdots \\ \nu_1^{(m-1)} & \nu_2^{(m-1)} & \cdots & \nu_{N-1}^{(m-1)} & \nu_N^{(m-1)} & 0\,0\cdots 1\,0 \\ \nu_1^{(m)} & \nu_2^{(m)} & \cdots & \nu_{N-1}^{(m)} & \nu_N^{(m)} & 0\,0\cdots 0\,1 \end{bmatrix} \tag{3.20}$$

As a result, when the equation systems given in (3.19) are solved, the **x** matrix is obtained with dimension $(m + N) \times N$.

$$\mathbf{x} = \begin{bmatrix} \eta_1^{(1)} & \eta_2^{(1)} & \cdots & \eta_{N-1}^{(1)} & \eta_N^{(1)} \\ \eta_1^{(2)} & \eta_2^{(2)} & \cdots & \eta_{N-1}^{(2)} & \eta_N^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \eta_1^{(m+N-1)} & \eta_2^{(m+N-1)} & \cdots & \eta_{N-1}^{(m+N-1)} & \eta_N^{(m+N-1)} \\ \eta_1^{(m+N)} & \eta_2^{(m+N)} & \cdots & \eta_{N-1}^{(m+N)} & \eta_N^{(m+N)} \end{bmatrix} \tag{3.21}$$

This matrix is the degenerate basic solution of the equation system and more than $m - N$ of the components in each column of **x** are zero [30].

The most important point of our new method is to find out the location of each testing node in the index space and for this purpose, the determination process of the matrix, **x**, is defined inside the method. Now, it is time to specify the usage of this matrix for the testing nodes. The dimension of the vector of each testing node, say **Q**, is $N \times 1$.

$$\mathbf{Q}_{(\ell)}^T = \left[\mu_1^{(\ell)} \ \mu_2^{(\ell)} \ \cdots \ \mu_{N-1}^{(\ell)} \ \mu_N^{(\ell)} \right], \quad 1 \le \ell \le t \tag{3.22}$$

Here, t is the number of testing nodes and T stands for the transpose of the matrix.

We need to evaluate $\mathbf{Q}_{(\ell)}^T\mathbf{x}$ to determine the location of each testing node in the index space. However, the dimensions of $\mathbf{Q}_{(\ell)}^T$ and $\mathbf{x}$ do not match and a mechanism should be developed by taking the philosophy used in the equation system solution process into consideration to bypass this case. Hence, the new form of $\mathbf{Q}_{(\ell)}^T$ should be as follows

$$\mathbf{Q}'_{(\ell)}{}^T = \left[\mu_1^{(\ell)}\ \mu_2^{(\ell)}\ \cdots\ \mu_{N-1}^{(\ell)}\ \mu_N^{(\ell)}\ \mu_{N+1}^{(\ell)}\ \mu_{N+2}^{(\ell)}\ \cdots\ \mu_{N+m}^{(\ell)}\right] \tag{3.23}$$

where $1 \le \ell \le t$. The $\mu_{N+1}^{(\ell)}, \ldots, \mu_{N+m}^{(\ell)}$ values correspond to the values of a line of the identity matrix concatenated to the matrix, $\mathbf{A}$. The question here is which line of that identity matrix will be concatenated to $\mathbf{Q}_{(\ell)}^T$ to determine the matrix, $\mathbf{Q}'_{(\ell)}{}^T$. This line will be the most appropriate training node's line and this appropriate training node of the testing node under consideration is specified by using the Euclidean distance metric [31].

$$d_\ell^{(j)} = \sqrt{\sum_{i_1=1}^{N}\left(v_{i_1}^{(j)} - \mu_{i_1}^{(\ell)}\right)^2}, \quad 1 \le \ell \le t, \quad 1 \le j \le m \tag{3.24}$$

The training node which has the shortest distance to the testing node under consideration is determined by using this metric. If there are several training nodes having the same property then the following Discrete metric [31] is used to obtain the best appropriate training node.

$$d_\ell^{(j)}\left(v_{i_1}^{(j)} - \mu_{i_1}^{(\ell)}\right) = \begin{cases} 0 & v_{i_1}^{(j)} = \mu_{i_1}^{(\ell)} \\ 1 & v_{i_1}^{(j)} \neq \mu_{i_1}^{(\ell)} \end{cases}, \quad 1 \le \ell \le t, \quad 1 \le i_1 \le N, \quad 1 \le j \le m \tag{3.25}$$

Now, the vector given in (3.23) is determined. The final step is the evaluation of $\mathbf{Q}'_{(\ell)}{}^T\mathbf{x}$ to obtain the location of the testing node in the index space. However, this work also covers another process before this multiplication to have better approximations through our new method, MI-HDMR. This process is about rebuilding the content of the matrix, $\mathbf{x}$. The elements of this matrix is replaced by new values obtained through the following relation

$$\eta_i'^{(k)} = \frac{v_k^{(app)}}{\mu_k^{(\ell)}}\eta_i^{(k)}, \quad 1 \le i \le N, \quad 1 \le k \le m+N, \quad 1 \le \ell \le t \tag{3.26}$$

where $v_k^{(app)}$ stands for the appropriate training node determined for the testing node under consideration. This relation updates the elements of the $\mathbf{x}$ matrix with respect to the ratio of the change between the considered testing node components and its appropriate training node components. Now, we have $\mathbf{Q}'_{(\ell)}{}^T$ given in (3.23) and the

following matrix.

$$\mathbf{x}' = \begin{bmatrix} \eta_1'^{(1)} & \eta_2'^{(1)} & \cdots & \eta_{N-1}'^{(1)} & \eta_N'^{(1)} \\ \eta_1'^{(2)} & \eta_2'^{(2)} & \cdots & \eta_{N-1}'^{(2)} & \eta_N'^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \eta_1'^{(m+N-1)} & \eta_2'^{(m+N-1)} & \cdots & \eta_{N-1}'^{(m+N-1)} & \eta_N'^{(m+N-1)} \\ \eta_1'^{(m+N)} & \eta_2'^{(m+N)} & \cdots & \eta_{N-1}'^{(m+N)} & \eta_N'^{(m+N)} \end{bmatrix} \tag{3.27}$$

Finally, the location of each testing node is obtained after the following evaluation.

$$\left[\vartheta_1^{(\ell)} \ \vartheta_2^{(\ell)} \ \ldots \ \vartheta_N^{(\ell)} \right] = \mathbf{Q}_{(\ell)}'^{T} \mathbf{x}', \quad 1 \le \ell \le t \tag{3.28}$$

As a result, we can construct the one-to-one mapping between each testing node and the index space nodes as follows.

$$\left(\mu_1^{(\ell)}, \mu_2^{(\ell)}, \ldots, \mu_N^{(\ell)}\right) \quad \Rightarrow \quad \left(\vartheta_1^{(\ell)}, \vartheta_2^{(\ell)}, \ldots, \vartheta_N^{(\ell)}\right), \quad 1 \le \ell \le t \tag{3.29}$$

Using the abovementioned location information the function value of each testing node can be evaluated through the approximants given in (3.13).

To examine whether our new method works well or not, a relative error analysis is done by using the following relation

$$\mathcal{N} = \frac{||f_{org} - f_{new}||}{||f_{org}||} \tag{3.30}$$

where f_{org} and f_{new} stand for the original multivariate function and the multivariate function obtained through the MI-HDMR approximant, respectively.

To determine the appropriate training node for each testing node of the given problem, the Euclidean distance metric and the Discrete metric are used and the general algorithm for the MI-HDMR method is given through these similarity metrics. However, some other alternative distance metrics can be also used for this process. In this work, these alternative metrics are selected as Manhattan Distance, Minkowski Distance, Canberra Distance, Bray Curtis (Sorensen) Distance, Cosine Distance [31,32]. The relations of these metrics in terms of our method's terminology are given as follows

$$d_\ell^{(j)} = \sum_{i_1=1}^{N} \left| \left(v_{i_1}^{(j)} - \mu_{i_1}^{(\ell)} \right) \right|, \quad 1 \le \ell \le t, \quad 1 \le j \le m \tag{3.31}$$

$$d_\ell^{(j)} = \sqrt[\lambda]{\sum_{i_1=1}^{N} \left| \left(v_{i_1}^{(j)} - \mu_{i_1}^{(\ell)} \right) \right|^{\lambda}}, \quad 1 \le \ell \le t, \quad 1 \le j \le m \tag{3.32}$$

$$d_{\ell}^{(j)} = \sum_{i_1=1}^{N} \frac{\left|\left(v_{i_1}^{(j)} - \mu_{i_1}^{(\ell)}\right)\right|}{\left|v_{i_1}^{(j)}\right| + \left|\mu_{i_1}^{(\ell)}\right|}, \quad 1 \le \ell \le t, \quad 1 \le j \le m \tag{3.33}$$

$$d_{\ell}^{(j)} = \frac{\sum_{i_1=1}^{N} \left|\left(v_{i_1}^{(j)} - \mu_{i_1}^{(\ell)}\right)\right|}{\sum_{i_1=1}^{N} \left(v_{i_1}^{(j)} + \mu_{i_1}^{(\ell)}\right)}, \quad 1 \le \ell \le t, \quad 1 \le j \le m \tag{3.34}$$

$$d_{\ell}^{(j)} = \frac{\sum_{i_1=1}^{N} v_{i_1}^{(j)} \mu_{i_1}^{(\ell)}}{\left(\sum_{i_1=1}^{N} \left(v_{i_1}^{(j)}\right)^2 \sum_{i_2=1}^{N} \left(\mu_{i_2}^{(\ell)}\right)^2\right)^{1/2}}, \quad 1 \le \ell \le t, \quad 1 \le j \le m \tag{3.35}$$

where the first relation is for Manhattan, the second for Minkowski, and the rest of them are for Canberra, Bray Curtis and Cosine, respectively. These metrics can be also used in the appropriate training node determination process instead of the Euclidean Distance and Discrete metrics. The performances of all these metrics are examined in the numerical implementations section through several examples.

4 Numerical implementations

To test the performance of our new method, several numerical implementations are given in this section. For this purpose, various multivariate testing functions and randomly selected domains for the independent variables of these functions are defined.

To obtain the numerical results, program codes are mostly written in Perl Programming Language [33]. To solve the linear equation system, $\mathbf{Ax} = \mathbf{b}$, MuPAD, Multi Processing Algebra Data tool [34] is used with 15-digits precision. This precision is provided by the environment variable DIGITS of MuPAD which determines the number of significant decimal digits in floating point numbers. The value assigned to DIGITS must be a positive integer smaller than 2^{31}. The default significant decimal digits value for Perl is 15 and it can be increased by using special data types and functions written for Perl.

In the implementations of this section, comparisons between univariate GHDMR approximant and the univariate, bivariate and trivariate MI-HDMR approximants are made and the results are given in a number of tables. The reason to deal with only the univariate GHDMR approximant, not the higher variate approximants, is the complex structure of the GHDMR algorithm [22] as mentioned in the introduction part of the paper. When we try to determine the bivariate and the higher variate component structures, we see that possible relations for these structures will include complicated and troublous integral equation systems. This results in very high mathematical and computational complexities, since there are no works about the determination process of this task in literature. The cost to evaluate the bivariate and trivariate GHDMR approximants is too expensive and this evaluation process is not applicable and realistic. Hence, we do not have any information to determine these higher variate GHDMR approximants now. On the other hand, we have the ability of easily determining the bivariate, trivariate or higher variate MI-HDMR approximants. This gives us the

chance of increasing the approximation performance for the MI-HDMR method while we do not have the same chance for the GHDMR method in practice. As a result, we can only use at most univariate GHDMR approximant in our comparisons to test the performance of our new method, MI-HDMR.

The selected multivariate testing functions are as follows

$$
\begin{aligned}
&f_1(x_1,\ldots,x_6)=\sum_{i=1}^{6}x_i, \quad f_2(x_1,\ldots,x_6)=\left[\sum_{i=1}^{6}x_i\right]^4,\\
&f_3(x_1,\ldots,x_6)=\left[\sum_{i=1}^{6}x_i\right]^7, \quad f_4(x_1,\ldots,x_6)=\prod_{i=1}^{6}x_i,\\
&f_5(x_1,\ldots,x_6)=\mathrm{e}^{x_1+x_3+x_3+x_4+x_5+x_6}, \quad f_6(x_1,\ldots,x_6)=\sum_{i=1}^{6}a_i x_i, \quad a_i=i,\\
&f_7(x_1,\ldots,x_6)=\prod_{i=1}^{6}x_i^{a_i}, \quad a_1=a_5=2,\ a_2=a_6=3,\ a_3=1,\ a_4=4
\end{aligned}
\tag{4.1}
$$

where each testing function has 6 independent variables. The first five testing functions have purely additive, dominantly additive, dominantly multiplicative, purely multiplicative and exponential natures, respectively. The last two testing functions, $f_6(x_1,\ldots,x_6)$ and $f_7(x_1,\ldots,x_6)$ are defined here to show the performance change of the proposed method when a different transformation is applied to the problem during the matrix representation of the mapping procedure. Originally, the method includes a process that sorts the nodes of the original space by function values in ascending order before constructing a one-to-one mapping between the original space's nodes and the index space's nodes.

The following domains including real values are defined for each independent variable.

$$
\begin{aligned}
&x_1\in\{0.6,1.3,2.4\}, && x_2\in\{5.3,6.1,6.5\},\\
&x_3\in\{2.5,3.4,3.9,4.8,6.0\}, && x_4\in\{0.2,0.4,1.9,2.3,3.5,4.6\},\\
&x_5\in\{1.5,2.6,3.8,4.1\}, && x_6\in\{3.0,3.2,3.7,4.9,5.1\}
\end{aligned}
\tag{4.2}
$$

The whole grid that can be constructed through these domains has 5400 nodes. This number stands for the number of elements of the cartesian product set produced by using the above domains. In our numerical implementations, we randomly select 960 nodes at which the function values are known as the training data set. The testing data set has 200 nodes and the sought function values at these nodes are asked. The training data set nodes and the testing data set nodes are selected through the whole grid by a pseudo random function written by the author. This function assigns the nodes from the whole grid to an array, then specifies the array indexes randomly by using the *srand()* and *rand()* functions of Perl [33] and selects the nodes appearing in the randomly specified indexes of the array.

Table 1 Relative error values of the approximants for the testing functions

	Training part				Testing part			
	$\mathcal{N}_{\bar{s}_1}$	$\mathcal{N}_{s_1}$	$\mathcal{N}_{s_2}$	$\mathcal{N}_{s_3}$	$\mathcal{N}_{\bar{s}_1}$	$\mathcal{N}_{s_1}$	$\mathcal{N}_{s_2}$	$\mathcal{N}_{s_3}$
f_1	**0.0**	0.0195	0.0169	0.0044	**0.0**	0.0298	0.0291	0.0271
f_2	0.0990	0.1420	0.0873	**0.0333**	**0.1020**	0.1611	0.1223	**0.1074**
f_3	0.2847	0.3565	0.1866	**0.0747**	0.3234	0.4184	0.2765	**0.2141**
f_4	0.4477	0.4912	0.2737	**0.1293**	0.4555	0.5266	0.3995	**0.3350**
f_5	0.8781	0.9283	0.8337	**0.6447**	0.8947	0.9441	0.8525	**0.8047**
f_6	**0.0**	0.0221	0.0199	0.0043	**0.0**	0.0372	0.0358	0.0307
f_7	0.8044	0.8571	0.6446	**0.3863**	0.8047	0.8425	0.6566	**0.5092**

The first step is to evaluate the prime factors of 960 and depending on the number of independent variables of the problem and these prime factors we have to specify the domains of the index space's independent variables. Since we have 6 independent variables the number of elements of each domain will be 2, 2, 3, 4, 4 and 5. Hence, the following indexing scheme is obtained.

$$\begin{aligned} &\xi_1 \in \{1,2\}, \quad \xi_2 \in \{1,2\}, \quad \xi_3 \in \{1,2,3\}, \\ &\xi_4 \in \{1,2,3,4\}, \ \xi_5 \in \{1,2,3,4\}, \ \xi_6 \in \{1,2,3,4,5\} \end{aligned} \tag{4.3}$$

The second step is to construct a cartesian product set by using these abovementioned domains and then a one-to-one mapping is constructed between this set and the original training data set of the given problem. Now, this new data set can be partitioned into univariate, bivariate and trivariate data sets through the MI-HDMR method and the univariate, bivariate and the trivariate approximants can be determined as the approximate analytical structures of the sought multivariate function. These approximants are the representations of the index space. Hence, when we need to evaluate the sought function values at the given testing nodes, we have to determine the location of each testing node in the index space. For this purpose, the **x** matrix of the given problem is obtained. Using that matrix the mentioned location of each testing node can be specified. Finally, these locations of the testing nodes allow us to use the obtained approximant to find out the sought function values and the performance of our new method can then be examined by using the relative error relation given in (3.30).

The performance of the data partitioning and finally analytical structure determination for the given data modelling problem are given in Table 1. The results are obtained for randomly constructed multivariate problems under the domains given in (4.2) by using the analytically known testing functions given in (4.1).

In Table 1, $\bar{s}_1$ stands for the univariate Generalized HDMR approximant while s_1, s_2 and s_3 stand for the univariate, bivariate and trivariate MI-HDMR approximants.

The given values in the tables are relative error values evaluated by using relation (3.30) and it is clear that a good performance is obtained as its value becomes close to zero. The best error values are marked in boldface. For instance, there is an error

value of 3% for the modelling part of the second testing function's problem obtained through the trivariate approximant.

The results of Table 1 says that the univariate GHDMR approximant works better than the univariate MI-HDMR approximant for all cases. However, the bivariate and the trivariate GHDMR approximants cannot be evaluated since it is not applicable to evaluate the relations for second and higher order GHDMR components as mentioned before. Hence, we have at most univariate approximation through GHDMR and we cannot improve the performance of this method in practice. On the other hand, we can use bivariate and trivariate MI-HDMR approximants to improve the performance of the representation of the given multivariate problem. This brings an important advantage for our new method with respect to GHDMR. In this sense, we may compare the relative error results of univariate GHDMR approximant and the trivariate MI-HDMR approximant obtained for each case and we can say that the Generalized HDMR approximant works well only for purely additive structures. As the additivity dominance of the sought structure decreases and the multiplicativity dominance increases then the MI-HDMR approximants become more efficient. The error values show that our new method works well for the dominantly additive natures while one may say that MI-HDMR works reasonably good for cases that are purely of additive nature. It can be also easily seen that the method works better than the Generalized HDMR method for the multiplicative natures. The performance of MI-HDMR method is average for the purely multiplicative structures while the Generalized HDMR method cannot represent these types. On the other hand, the exponential natures cannot be represented by either of them successfully because of the additive nature of the HDMR expansion. This can be observed by looking at the results given for the testing function, $f_5(x_1, \ldots, x_6)$ in Table 1. In addition, the functions $f_6(x_1, \ldots, x_6)$ and $f_7(x_1, \ldots, x_6)$ are given to test the perfomance of the method while different mapping procedures are taken into consideration, that is, the structures of the matrices of the $\mathbf{Ax} = \mathbf{b}$ equation system become different. Table 1 includes the results obtained through the method having a sorting process by function values in ascending order. A second way is to use the original space's nodes in the order as they are given in the problem, that is, in the unsorted structure. This results in the following relative error values of the testing part for the trivariate MI-HDMR approximants of these two testing functions, $f_6(x_1, \ldots, x_6)$ and $f_7(x_1, \ldots, x_6)$

$$\mathscr{N}_{f_6} = 0.1222, \quad \mathscr{N}_{f_7} = 0.8716 \tag{4.4}$$

where the results become dramatically worse.

To examine the general tendency of the relative error values for each testing function in both the training and the testing part of our new method, the Perl scripts are executed 30 times, that is, we construct 30 different multivariate problems by using the following domains for each testing function given in (4.1).

$$\begin{aligned} &x_1 \in \{0.6, 1.3, 2.4\}, && x_2 \in \{5.3, 6.1, 6.5\}, \\ &x_3 \in \{2.5, 3.4, 3.9, 4.8\}, && x_4 \in \{0.2, 0.4, 1.9, 2.3, 3.5, 4.6\}, \\ &x_5 \in \{2.6, 3.8, 4.1\}, && x_6 \in \{3.7, 4.9, 5.1\} \end{aligned} \tag{4.5}$$

Table 2 Arithmetic means of relative error values obtained for randomly constructed 30 multivariate problems

	Training part				Testing part			
	$\mathscr{N}_{\bar{s}_1}$	$\mathscr{N}_{s_1}$	$\mathscr{N}_{s_2}$	$\mathscr{N}_{s_3}$	$\mathscr{N}_{\bar{s}_1}$	$\mathscr{N}_{s_1}$	$\mathscr{N}_{s_2}$	$\mathscr{N}_{s_3}$
f_1	**0.0**	0.0151	0.0131	0.0039	**0.0**	0.0349	0.0343	0.0325
f_2	0.0878	0.1092	0.0621	**0.0214**	**0.1118**	0.1551	0.1326	**0.1150**
f_3	0.2094	0.2727	0.1429	**0.0598**	0.2357	0.2985	0.2250	**0.2049**
f_4	0.3831	0.4230	0.2033	**0.0802**	0.4041	0.4829	0.3512	**0.3091**
f_5	0.7643	0.8014	0.5632	**0.3095**	0.8216	0.8371	0.7135	**0.6535**
f_6	**0.0**	0.0151	0.0136	0.0032	**0.0**	0.0307	0.0303	0.0279
f_7	0.7146	0.7895	0.5263	**0.2634**	0.7716	0.8075	0.6686	**0.5498**

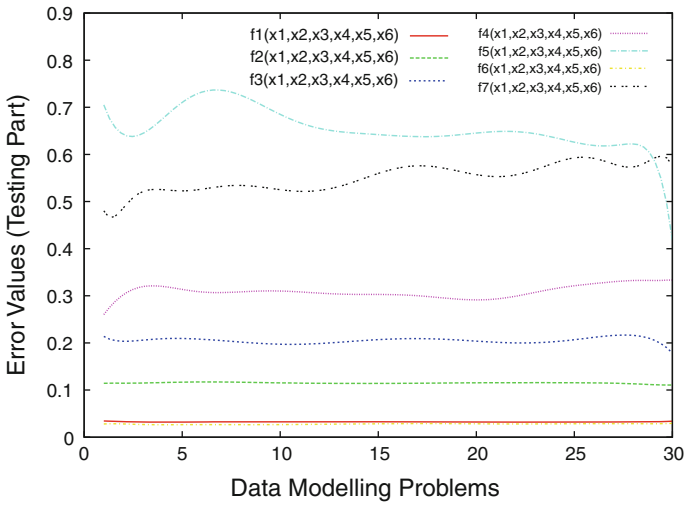

Fig. 1 Figure for the relative error values obtained in testing problems constructed for Table 2

The training data set has 240 randomly selected nodes over 1944 nodes which stands for all possible nodes of the problem domain. The number of testing nodes for each problem is 100. The arithmetic means of the relative error values obtained for the randomly constructed 30 different problems for each testing function are given in Table 2.

The MI-HDMR approximants are compared with at most the univariate GHDMR approximant because of the same reasons given for Table 1 and it is seen that the results of Table 2 are very parallel to the results given in Table 1. The method works impressively better than the Generalized HDMR method for the data modelling problems. Finally, the general tendency of the obtained error values for the trivariate MI-HDMR approximants of the 30 randomly constructed multivariate data modelling problems are given in Fig. 1.

Table 3 Relative error values of the trivariate approximant obtained by using alternative similarity metrics

	Testing part						
	Euclidean	Manhattan	Minkowski ($\lambda = 3$)	Minkowski ($\lambda = 5$)	Canberra	Bray Curtis	Cosine
$\mathcal{N}_{f_1}$	**0.0271**	0.0293	0.0283	0.0279	0.0296	0.0296	0.0371
$\mathcal{N}_{f_2}$	**0.1074**	0.1145	0.1104	0.1104	0.1151	0.1151	0.1468
$\mathcal{N}_{f_3}$	**0.2141**	0.2204	0.2207	0.2243	0.2220	0.2220	0.2781
$\mathcal{N}_{f_4}$	**0.3350**	0.3501	0.3662	0.3668	0.3497	0.3497	0.3538
$\mathcal{N}_{f_5}$	0.8047	**0.7221**	0.8187	0.8180	0.7258	0.7258	0.8895
$\mathcal{N}_{f_6}$	0.0307	0.0293	0.0283	**0.0279**	0.0296	0.0300	0.0371
$\mathcal{N}_{f_7}$	**0.5092**	0.5945	0.5098	0.5122	0.6014	0.6014	0.5662

Table 4 Arithmetic means of relative error values obtained for randomly constructed 30 multivariate problems by using alternative similarity metrics

	Testing part						
	Euclidean	Manhattan	Minkowski ($\lambda = 3$)	Minkowski ($\lambda = 5$)	Canberra	Bray Curtis	Cosine
$\mathcal{N}_{f_1}$	**0.0325**	0.0359	0.0342	0.0342	0.0364	0.0346	0.0507
$\mathcal{N}_{f_2}$	**0.1150**	0.1373	0.1334	0.1341	0.1361	0.1376	0.2052
$\mathcal{N}_{f_3}$	**0.2049**	0.2364	0.2589	0.2344	0.2418	0.2301	0.3227
$\mathcal{N}_{f_4}$	**0.3091**	0.3202	0.3239	0.3286	0.3266	0.3108	0.3774
$\mathcal{N}_{f_5}$	**0.6535**	0.6763	0.6628	0.6605	0.6505	0.6593	0.6988
$\mathcal{N}_{f_6}$	**0.0292**	0.0303	0.0299	0.0296	0.0310	0.0307	0.0469
$\mathcal{N}_{f_7}$	**0.5414**	0.5881	0.5915	0.5716	0.5654	0.5715	0.6930

The determination process of the appropriate training node for each testing node through the Euclidean Distance and Discrete metrics is not a unique solution for the mentioned purpose. There are, of course, some other alternative metrics that can be used in our new algorithm and these metrics are given through relations from (3.31) to (3.35). The relative error values obtained through the trivariate MI-HDMR approximant with the help of these metrics are given in Table 3. The relative error values appearing in the first column of Table 3 are from Table 1 to make an easy comparison between the new results and the results obtained through the Euclidean Distance metric. It can be examined that the algorithm with Euclidean Distance metric (with Discrete metric support) works better than the others instead of two cases which correspond to the exponential type and the additive type testing functions. The algorithm with Manhattan Distance and Minkowski metrics are the best choices to model the given multivariate data for these cases, respectively. However, these results are obtained for a randomly constructed multivariate data modelling problem. Table 4 includes arithmetic means of relative error values obtained for randomly constructed 30 multivariate problems to observe the general tendency of different similarity metrics and it is seen that the best results are always obtained through the MI-HDMR method including the Euclidean Distance and Discrete metrics as similarity metrics.

Table 5 Relative error values of the approximants for the testing functions (Lagrange interpolation vs. Spline interpolation)

	Training part				Testing part			
	$\mathscr{N}_{s_1}^{(Lag)}$	$\mathscr{N}_{s_2}^{(Lag)}$	$\mathscr{N}_{s_1}^{(Sp)}$	$\mathscr{N}_{s_2}^{(Sp)}$	$\mathscr{N}_{s_1}^{(Lag)}$	$\mathscr{N}_{s_2}^{(Lag)}$	$\mathscr{N}_{s_1}^{(Sp)}$	$\mathscr{N}_{s_2}^{(Sp)}$
f_8	0.1878	**0.1428**	0.3136	0.2894	0.2068	**0.2008**	0.3342	0.3336
f_9	0.0885	**0.0289**	0.1341	0.1041	0.1557	**0.1057**	0.2058	0.1677

To examine the performance of our new method in piecewise continuous functions, the following testing functions are selected

$$f_8(x_1,\ldots,x_6)=\begin{cases}\displaystyle\sum_{i=1}^{6}x_i^2, & x_1,x_5\le -2,\ x_2,x_3,x_4,x_6\le 0\\ \left[\displaystyle\sum_{i=1}^{6}x_i\right]^2, & x_1,x_5>-2,\ x_2,x_3,x_4,x_6>0\end{cases}$$

$$f_9(x_1,\ldots,x_6)=\begin{cases}\displaystyle\sum_{i=1}^{6}x_i, & x_1,x_5\le -2,\ x_2,x_3,x_4,x_6\le 0\\ \sin\left(\pi\displaystyle\sum_{i=1}^{6}(-1)^i x_i\right), & x_1,x_5>-2,\ x_2,x_3,x_4,x_6>0\end{cases} \tag{4.6}$$

where both testing functions have 6 independent variables. Again, we construct a grid having 5400 nodes and we randomly build a training data set of 960 nodes and a testing data set of 200 nodes. We use the method with both Lagrange interpolation and the cubic spline interpolation. Since we use at most bicubic spline function, the univariate and the bivariate MI-HDMR approximants are determined for the testing piecewise continuous functions. The relative error values obtained for these testing functions for a randomly constructed multivariate data modelling problem are given in Table 5. Table 6 shows the relative errors obtained through 30 randomly constructed multivariate data modeling problem having 240 training nodes and 100 testing nodes and includes a general performance comparison between the two different interpolation techniques. The superscripts *(Lag)* and *(Sp)* in the tables correspond to Lagrange and Spline, respectively. It can be easily examined from the tables Tables 5 and 6 that the Lagrange interpolation problem works better for all cases.

5 Concluding remarks

In this work, a new HDMR based method is developed to partition a multivariate data set into less-variate data sets to reduce the mathematical and computational complexities of real life problems. The Plain HDMR method has the ability of partitioning a multivariate data set with the help of an appropriate weight function. However, the prerequisite of this method is an orthogonal geometry need in the data structure of

Table 6 Arithmetic means of relative error values obtained for randomly constructed 30 multivariate problems (Lagrange interpolation vs. Spline interpolation)

	Training part				Testing part			
	$\mathcal{N}_{s_1}^{(Lag)}$	$\mathcal{N}_{s_2}^{(Lag)}$	$\mathcal{N}_{s_1}^{(Sp)}$	$\mathcal{N}_{s_2}^{(Sp)}$	$\mathcal{N}_{s_1}^{(Lag)}$	$\mathcal{N}_{s_2}^{(Lag)}$	$\mathcal{N}_{s_1}^{(Sp)}$	$\mathcal{N}_{s_2}^{(Sp)}$
f_8	0.1436	**0.1060**	0.3257	0.3163	0.2446	**0.2309**	0.3857	0.3668
f_9	0.1130	**0.0212**	0.1460	0.0985	0.1190	**0.0699**	0.1614	0.1230

the given problem. Hence, when we deal with HDMR based algorithms in multivariate data modelling problems, we need additional mechanisms for the partitioning process. Matrix based Indexing HDMR method is developed for this purpose. Our new method imposes an indexing scheme to the problem to construct a one-to-one mapping between the original domain and the index space and to obtain an orthogonal geometry from the given non-orthogonal geometry in order to use the plain HDMR philosophy in the partitioning process. The mapping allows us to know the location of each training node in the index space while there exists no location information for each testing node. Since the approximate analytical structure obtained through the MI-HDMR approximant has the independent variables of that index space, we should find the location of each testing node in the imposed index space to evaluate the sought function value at the testing node under consideration. To find that location first the appropriate training node should be determined for each testing node. For this purpose, several similarity metrics are used. The numerical implementations show us that the algorithm with the Euclidean Distance metric and the Discrete metric gives the best approximations to the sought analytical structures. Our new method also gives the ability of the determination of this information through a matrix representation whose elements are obtained through an overdetermined equation system.

The features taken into consideration in this mapping process are to have the same number of data in the original space and in the index space and to sort the function values given in the problem in ascending order. Of course, this type of a transformation over variables through a such sorting process does not always give the results better than the original ones. Hence, a method can be developed to choose better transformations to obtain better approximations as a future work.

Our new method works well for all types of additive structures and dominantly multiplicative structures. As the multiplicativity features of the structure of the problem become very dominant or the structure is purely multiplicative, the method's performance is average. However, for the exponential structures, since the method's expansion has an additive nature the MI-HDMR approximant becomes unsufficient. A new HDMR based method may be developed as future work to obtain better representations for these types of problems.

The approximation quality of the MI-HDMR method will become worse while the multivariate data modelling problem under consideration has a non-smooth structure since the method has an integration based algorithm. This property is also true for the other integration based algorithms. Hence, a huge number of training data set nodes are needed to model the given data through MI-HDMR. This results in some possible

modifications in the MI-HDMR algorithm to improve the performance of the method as the future work.

In Addition, the performance of our new method is tested for the data sets having certain singularities. For this purpose an alternative interpolation technique, cubic spline interpolation is used beside Lagrange interpolation and is applied to piecewise continuous functions. Our method with the Lagrange interpolation formula works better than the other. However, the piecewise structures having discontinuity properties are out of scope of this work because of the mismatch relations of spline interpolation and HDMR components and are said to be future work.

Acknowledgments Author is grateful to Professor Abdülbaki Baykara for his careful reading of the manuscript and his invaluable comments.

References

1. I.M. Sobol, Sensitivity estimates for nonlinear mathematical models. Math. Model. Comput. Exp. (MMCE) **1**, 407–414 (1993)
2. H. Rabitz, Ö.F. Alış, J. Shorter, K. Shim, Efficient input-output model representations. Comput. Phys. Comm. **117**, 11–20 (1999)
3. H. Rabitz, Ö. Alış, General foundations of high dimensional model representations. J. Math. Chem. **25**, 197–233 (1999)
4. G. Li, S.-W. Wang, H. Rabitz, Practical approaches to construct RS-HDMR component functions. J. Phys. Chem. A **106**, 8721–8733 (2002)
5. M. Demiralp, High dimensional model representation and its application varieties. Math. Res. **9**, 146–159 (2003)
6. M.A. Tunga, M. Demiralp, A factorized high dimensional model representation on the partitioned random discrete data. Appl. Num. Anal. Comp. Math. **1**, 231–241 (2004)
7. M.A. Tunga, M. Demiralp, A factorized high dimensional model representation on the nodes of a finite hyperprismatic regular grid. Appl. Math. Comput. **164**, 865–883 (2005)
8. B. Tunga, M. Demiralp, Hybrid high dimensional model representation approximants and their utilization in applications. Math. Res. **9**, 438–446 (2003)
9. M.A. Tunga, M. Demiralp, Hybrid high dimensional model representation (HHDMR) on the partitioned data. J. Comput. Appl. Math. **185**, 107–132 (2006)
10. M. Demiralp, Illustrative implementations to show how logarithm based high dimensional model representation works for various function structures. WSEAS Trans. Comput. **5**, 1339–1344 (2006)
11. T. Ziehn, A.S. Tomlin, A global sensitivity study of sulfur chemistry in a premixed methane flame model using HDMR. Int. J. Chem. Kinet. **40**, 742–753 (2008)
12. T. Ziehn, A.S. Tomlin, GUI-HDMR—a software tool for global sensitivity analysis of complex models. Environ. Model. Softw. **24**, 775–785 (2009)
13. J. Sridharan, T. Chen, Modeling multiple input switching of CMOS gates in DSM technology using HDMR. Proc. Des. Autom. Test Eur. **1**(3), 624–629 (2006)
14. B.N. Rao, R. Chowdhury, Probabilistic analysis using high dimensional model representation and fast fourier transform. Int. J. Comput. Methods Eng. Sci. Mech. **9**, 342–357 (2008)
15. R. Chowdhury, B.N. Rao, Hybrid high dimensional model representation for reliability analysis. Comput. Methods Appl. Mech. Eng. **198**, 753–765 (2009)
16. M.C. Gomez, V. Tchijov, F. Leon, A. Aguilar, A tool to improve the execution time of air quality Mmodels. Environ. Model. Softw. **23**, 27–34 (2008)
17. I. Banerjee, M.G. Ierapetritou, Design optimization under parameter uncertainty for general black-box models. Ind. Eng. Chem. Res. **41**, 6687–6697 (2002)
18. I. Banerjee, M.G. Ierapetritou, Parametric process synthesis for general nonlinear models. Comput. Chem. Eng. **27**, 1499–1512 (2003)
19. I. Banerjee, M.G. Ierapetritou, Model independent parametric decision making. Ann. Oper. Res. **132**, 135–155 (2004)

20. S. Shan, G.G. Wang, Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally- expensive black-Box functions. Struct. Multidiscip. Optim. **41**, 219–241 (2010)
21. M.A. Tunga, M. Demiralp, A new approach for data partitioning through high dimensional model representation. Int. J. Comput. Math. **85**, 1779–1792 (2008)
22. M.A. Tunga, M. Demiralp, Data partitioning via generalized high dimensional model representation (GHDMR) and multivariate interpolative applications. Math. Res. **9**, 447–462 (2003)
23. M.A. Tunga, M. Demiralp, in *Introductory Steps for an Indexing Based HDMR Algorithm: Lumping HDMR, 1st WSEAS International Conference on Multivariate Analysis and its Application in Science and Engineering*. (Istanbul, Turkey, May 27–30, 2008), pp. 129–135
24. M.A. Tunga, M. Demiralp, A reverse technique for lumping high dimensional model representation method. WSEAS Trans. Math. **8**, 213–218 (2009)
25. M.A. Tunga, in *An Indexing Based Approximation Method for Multivariate Interpolation Problems, The 1st International Symposium on Computing in Science & Engineering (ISCSE 2010)*. (Kusadasi, Aydin, Turkey, June 3–5 2010). pp. 56–60
26. A. Zemanian, *Distribution Theory and Transform Analysis, An Introduction to Generalized Functions, with Applications* (Dover Publications Inc., New York, 1987)
27. R.L. Burden, J.D. Faires, *Numerical Analysis* (Brooks/Cole, CA, 2001)
28. C. de Boor, *A Practical Guide to Splines, Revised edn* (Springer, New York, 2001)
29. E.V. Shikin, A.I. Plis, *Handbook on Splines for the User* (CRC Press, Florida, 1995)
30. D. Bertsimas, J.N. Tsitsiklis, *Introduction to Linear Optimization* (Athena Scientific, Belmont, MA, 1997)
31. V. Bryant, *Metric Spaces: Iteration and Application* (Cambridge University Press, New York, 1996)
32. K. Teknomo, *Similarity Measurement*. http://people.revoledu.com/kardi/tutorial/Similarity/
33. H.M. Deitel, P.J. Deitel, T.R. Nieto, D.C. McPhie, *How to Program Perl* (Prentice Hall, New Jersey, 2001)
34. W. Oevel, F. Postel, S. Wehmeier, J. Gerhard, *The MuPAD Tutorial* (Springer, New York, 2000)