A Reverse Technique for Lumping High Dimensional Model Representation Method

M. ALPER TUNGA
Bahçeşehir University
Department of Software Engineering
Beşiktaş, 34349, İstanbul
TURKEY (TÜRKİYE)
alper.tunga@bahcesehir.edu.tr

METİN DEMİRALP
İstanbul Technical University
Informatics Institute
Maslak, 34469, İstanbul
TURKEY (TÜRKİYE)
demiralp@be.itu.edu.tr

Abstract: An orthogonal hyperprismatic grid whose all nodes are accompanied by the given function values can not be generally constructed due to the random nature of the given function data. This prevents the reduction of the single multivariate interpolation to more than one univariate or bivariate interpolations even approximately. It is generally quite difficult to determine an analytical structure for the target function in these problems. Lumping HDMR method is an indexing based High Dimensional Model Representation (HDMR) algorithm used to reconstruct these types of multivariate data by imposing an indexing scheme to obtain an orthogonal geometry for the given problem. By this way, the training of the given data can be accomplished. The next problem is to determine a reverse algorithm for the testing data. This work is about a new algorithm to find the correct coordinate of the given testing data in the orthogonal geometry obtained through Lumping HDMR.

Key–Words: Data Partitioning, Multivariate Analysis, High Dimensional Model Representation, Lumping HDMR

1 Introduction

High Dimensional Model Representation (HDMR) is a divide–and–conquer algorithm and is used to represent given multivariate functions in terms of low variate functions or to determine approximate analytical structures for the given multivariate interpolation problems. This method allows us not to tackle with the complexities of the standard mathematical methods coming with the multivariance. It also enables us to deal with low variate interpolation problems instead of the given multivariate interpolation problem.

This method is used in several works for different research areas since 1993 [1]. Afterwards, Rabitz [2] and Demiralp [3] developed certain basics of the HDMR method. HDMR based algorithms were also applied to the multivariate interpolation problems [4–7]. These methods are used to partition the given multivariate data into low variate data sets, preferring at most bivariate ones. The obtained low variate data sets are employed to obtain analytical structure for the sought multivariate function.

We deal with two different types of multivariate interpolation problems for engineering issues in our research areas. The first type covers the problems in which the values of the sought multivariate function are given at all nodes of an orthonormal hyperprismatic grid. That is, all nodes of the whole grid are used in the analytical structure determining method for the given problem. The HDMR method is used for this purpose [4, 5].

The second type is related to the problems in which all nodes of the predetermined grid are not involved in the coordinates of the given data, that is, the function values are given on the certain grid nodes randomly without filling whole grid. This makes it impossible to use the simplicity and the facilitation in the case of the grids whose nodes are fully accompanied by the given function values. In such cases, even plain HDMR does not work. This time, another version of HDMR, the Generalized HDMR (GHDMR) is used for the interpolation [6, 7].

The GHDMR method has also some technical problems in random data partitioning. There exists a set of linear equations, whose unknowns are the univariate GHDMR components of the sought function, to be solved in its algorithm. The number of linearly dependent equations appearing in this equation set sometimes causes serious problems related to high condition numbers of the matrices in the solution step. To avoid the mentioned difficulty Lumping HDMR, which is an indexing based HDMR algorithm, is developed by Tunga and Demiralp in 2008 [8]. That work was the training version of the interpolation part, that is, the determination of an analytical structure for the given problem by using an indexing scheme which

corresponds to the function value given original nodes of the problem.

Our aim in this work is to develop a new algorithm for the testing of the interpolation part. This new algorithm is expected to reveal the corresponding node for each index appearing in the indexing scheme used by Lumping HDMR. Hence, here we are trying to construct a reverse algorithm for Lumping HDMR to go back to the function value given nodes of the grid.

The HDMR and the Lumping HDMR methods are given in the second and third sections, not in all details. Our new algorithm is defined in the fourth section. A number of numerical implementations are also given before the concluding remarks to show the performance of our new algorithm.

2 The HDMR Method

The following equality is given as the HDMR expansion.

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

This expansion is a finite sum and is composed of a constant term, univariate terms, bivariate terms and so on. These are the HDMR components of a given multivariate function.

The following vanishing conditions are used to uniquely determine the right hand side components of that expansion

$$\int_{a_1}^{b_1} dx_1 \cdots \int_{a_N}^{b_N} dx_N W(x_1,...,x_N) f_i(x_i) = 0 \quad (2)$$

where $1 \le i \le N$. The weight function appearing in the vanishing conditions is assumed to be a product of univariate functions each of which depends on a different independent variable.

$$\begin{aligned} W(x_1,\ldots,x_N) &\equiv \prod_{j=1}^{N} W_j(x_j), \\ x_j \in [\,a_j\,,\,b_j\,], &\qquad 1 \le j \le N \end{aligned} \quad (3)$$

where each univariate factor is assumed to have 1-valued integral with respect to the related variable over the corresponding interval for easy determination of the HDMR components.

$$\int_{a_j}^{b_j} dx_j W_j(x_j) = 1, \qquad 1 \le j \le N \quad (4)$$

Using these properties of the weight function and the vanishing conditions, all components of HDMR can be uniquely determined. However the general tendency is to truncate HDMR at univariate or at most bivariate terms for practicality as long as the target function permits. Hence we deal with the determination of the constant and the univariate terms here.

To this end, the following operators can be written by using an arbitrary square integrable function, $F(x_1,\ldots,x_N)$, to determine the general structures of the constant HDMR term

$$\begin{aligned} \mathcal{I}_0 F(x_1,\ldots,x_N) \equiv & \int_{a_1}^{b_1} dx_1 W_1(x_1) \cdots \\ & \times \int_{a_N}^{b_N} dx_N W_N(x_N) F(x_1,\ldots,x_N) \end{aligned} \quad (5)$$

and the univariate HDMR terms

$$\begin{aligned} \mathcal{I}_m F(x_1,\ldots,x_N) \equiv & \int_{a_1}^{b_1} dx_1 W_1(x_1) \cdots \\ & \times \int_{a_{m-1}}^{b_{m-1}} dx_{m-1} W_{m-1}(x_{m-1}) \\ & \times \int_{a_{m+1}}^{b_{m+1}} dx_{m+1} W_{m+1}(x_{m+1}) \cdots \\ & \times \int_{a_N}^{b_N} dx_N W_N(x_N) F(x_1,\ldots,x_N) \end{aligned} \quad (6)$$

where $1 \le m \le N$. Other operators can be defined in a similar philosophy to determine the structures of the other HDMR terms, such as bivariate terms and so on.

When these operators are acted onto the both sides of the HDMR expansion by taking the vanishing conditions given in (2) under the product type weight function into consideration the following general structures for the constant and univariate terms are obtained respectively

$$\begin{aligned} f_0 &= \mathcal{I}_0 f(x_1,...,x_N) \\ f_m(x_m) &= \mathcal{I}_m f(x_1,...,x_N) - f_0 \end{aligned} \quad (7)$$

where $1 \le m \le N$. If we truncate HDMR by keeping constant term only then the approximation is called "Constant Approximation". The "Univariate Approximation" is defined by retaining the constant and univariate terms of HDMR. These HDMR truncations can be written more specifically as follows.

$$\begin{aligned} s_0(x_1,...,x_N) &= f_0 \\ s_1(x_1,...,x_N) &= f_0 + \sum_{i_1=1}^{N} f_{i_1}(x_{i_1}) \end{aligned} \quad (8)$$

3 Lumping HDMR

We assume that the given data are random or, in other words, not given at all points of a grid which is constructed via direct product of univariate meshes. That is, the information is randomly distributed over the nodes. The rational number obtained via dividing the number of given function values by the total number of grid nodes determines the sparsity of the information distribution. Especially in the case of sparsely distributed data, it is better to somehow gather all information in a cluster-like data structure. For this purpose, a virtual space where this clustering is realized can also be used. Each datum which is an $(N+1)$-tuple whose $(N+1)$-th component is the function's given value while the all remaining components form an N-tuple characterizing a point in N dimensional cartesian space of the independent variables, is indexed by a positive integer in an appropriate ordering which is of course not unique. Then the original coordinates are forgotten and the abovementioned index becomes the only variable to identify each datum. This univariance can be folded to create multivariance. If the number of indexes permits, an orthogonal hyperprismatic grid in a more than one dimensional virtual space is constructed such that its total number of nodes is equal to the number of indexes and there is a one-to-one relation between this grid's and the original one's nodes. Then HDMR can be applied to this orthogonal virtual geometry and its at most bivariate but preferably univariate truncation is used as an approximation for the partitioned data. In other words data given points of the original grid is lumped into the corresponding grid of the virtual space. We call this method "Lumping HDMR" [8] and the virtual space mentioned above "Index Space".

To better understand how an index space is built, we can consider the case where the number of given function values is 12. 12 has the prime factors, 2, 2, and, 3. Each factor can be considered as the number of the planes perpendicular to one of the edges of three dimensional orthogonal prismatic grid. This urges us to define three coordinates such that $x_1 \in \{1,2\}$, $x_2 \in \{1,2\}$, and $x_3 \in \{1,2,3\}$. Then, the nodes of the grid are given by the triples $(1,1,1)$, $(1,1,2)$, $(1,1,3)$, $(1,2,1)$, $(1,2,2)$, $(1,2,3)$, $(2,1,1)$, $(2,1,2)$, $(2,1,3)$, $(2,2,1)$, $(2,2,2)$, $(2,2,3)$.

In this example we construct a $2 \times 2 \times 3$ type grid in a three dimensional space. We could of course exchange the definitions of the independent variables x_1, x_2, x_3 and obtain 6 different grids. However, each of these grids are obtained from one of others by an appropriate rotation. In this sense, one can bring the uniqueness by choosing one of these possibilities as the essential index space.

The other flexibility for the index space construction is the choice of the dimension. We have constructed three dimensional spaces above. However we could use some binary products of the prime factors as single entities and reduce the index space dimension. For example, we could consider the expression $12 = 3 \times 4$ and a related two dimensional space where the independent variables are defined as $x_1 \in \{1,2,3\}$ and $x_2 \in \{1,2,3,4\}$. Then the grid nodes would be given by the ordered pairs $(1,1)$, $(1,2)$, $(1,3)$, $(1,4)$, $(2,1)$, $(2,2)$, $(2,3)$, $(2,4)$, $(3,1)$, $(3,2)$, $(3,3)$, $(3,4)$. We could also consider the two dimensional index space where the grid is 2×6 type beside the one dimensional space having 12 node linear grid. All these mean that there is a flexibility in the construction of the index space and the orthogonal prismatic grid embedded into this space. This nonuniqueness may be used to get most efficient lumping. What we know about the HDMR from experimentations is that it seems to work well in higher dimensions and higher node numbers in each direction defined by the independent variables.

The next step after the construction of the index space and the embedded orthogonal hyperprismatic grid is the indexing. It is of course not unique. However by choosing an appropriate ordering over the nodes of the original grid and over the index space's grid a one-to-one correspondence can be established between the node sets of these two grids.

To explain in general notation, first let us give the definition of the data of the variable x_j in index space as follows

$$\mathcal{D}_j \equiv \left\{ \xi_j^{(k_j)} \right\}_{k_j=1}^{k_j=n_j} = \left\{ \xi_j^{(1)}, \ldots, \xi_j^{(n_j)} \right\} \qquad (9)$$

where $1 \leq j \leq N$. The cartesian product of these sets contains N-tuples as the elements and defines the grid of the index space whose dimension is N

$$\mathcal{D} \equiv \mathcal{D}_1 \times \mathcal{D}_2 \times \cdots \times \mathcal{D}_N \qquad (10)$$

Now all points of this grid are accompanied by a given function value. Therefore we have a discrete structure which will be partitioned via HDMR. To provide the discreteness the following univariate weight functions are selected as the components of the overall weight function appearing in the HDMR algorithm. The Dirac delta function [9] is used in these structures because of the need for dealing with only the values of the sought function at the nodes of the above grid for the interpolation problem.

$$W_j(x_j) \equiv \sum_{k_j=1}^{n_j} \alpha_{k_j}^{(j)} \delta\left(x_j - \xi_j^{(k_j)}\right), \quad x_j \in [\, a_j \,,\, b_j \,], \qquad 1 \leq j \leq N \qquad (11)$$

Using this weight function the operators mentioned in the previous section can be applied to the both sides of the HDMR expansion by the help of the vanishing conditions and the given multivariate data is partitioned into low–variate data sets. In this work we deal with constant, univariate and at most bivariate terms.

After several integrations a constant value and univariate partitioned data set are obtained [5, 6].

The following equality is obtained to determine constant term, f_0

$$f_0 \equiv \sum_{\tau \in \mathcal{D}} \zeta(\tau) f(\tau) \tag{12}$$

where

$$\tau = \left(\xi_1^{(k_1)}, ..., \xi_N^{(k_N)}\right) \quad \zeta(\tau) = \alpha_1^{(k_1)} \cdots \alpha_N^{(k_N)}, \\ 1 \le k_j \le n_j, \quad 1 \le j \le N \tag{13}$$

The structure of the univariate components after some number of calculations is obtained as follows

$$f_m(\xi_m^{(k_m)}) = \sum_{\tau_m \in \mathcal{D}^{(m)}} \zeta_m(\tau_m) f(\tau_m, \xi_m^{(k_m)}) \\ - \sum_{\tau \in \mathcal{D}} \zeta(\tau) f(\tau) \tag{14}$$

where

$$\mathcal{D}^{(m)} \equiv \{\tau_m | \tau_m = (x_1, ..., x_{m-1}, x_{m+1}, ..., x_N), \\ x_j \in \mathcal{D}_j, 1 \le j \le N, j \ne m\}, \\ \tau_m = \left(\xi_1^{(k_1)}, ..., \xi_{m-1}^{(k_{m-1})}, \xi_{m+1}^{(k_{m+1})}, ..., \xi_N^{(k_N)}\right), \\ \zeta_m(\tau_m) = \alpha_1^{(k_1)} \cdots \alpha_{m-1}^{(k_{m-1})} \alpha_{m+1}^{(k_{m+1})} \cdots \alpha_N^{(k_N)}, \\ \xi_m^{(k_m)} \in \mathcal{D}_m, \ 1 \le k_m \le n_m, \ 1 \le m \le N \tag{15}$$

To this end, we have a constant value and n_m ordered pairs for the univariate function $f_m(x_m)$ [5, 6]. The next step is to interpolate these partitioned data to determine an analytical structure for the sought function.

4 Interpolation

To determine the overall structure of the function, an analytical structure should be defined or a calculation rule should be imposed on the interpolation. For this purpose, first a polynomial representation should be built for $f_m(x_m)$.

$$p_m(x_m) = \sum_{k_m=1}^{n_m} L_{k_m}(x_m) f_m\left(\xi_m^{(k_m)}\right), \\ \xi_m^{(k_m)} \in \mathcal{D}_m, \qquad 1 \le m \le N \tag{16}$$

Here $L_{k_m}(x_m)$s are Lagrange polynomials [10] which are independent of the function's structure. The explicit structures of these polynomials are given below

$$L_{k_m}(x_m) \equiv \prod_{\substack{j=1 \\ j \ne k_m}}^{n_m} \frac{\left(x_m - \xi_m^{(j)}\right)}{\left(\xi_m^{(k_m)} - \xi_m^{(j)}\right)}, \\ \xi_m^{(k_m)} \in \mathcal{D}_m, \quad 1 \le k_m \le n_m, \quad 1 \le m \le N \tag{17}$$

After the construction of Lagrange polynomials, univariate functions given by the relation (16) are uniquely determined within continous polynomial interpolation. These functions can be considered as univariate components of HDMR for the multivariate function, $f(x_1, ..., x_N)$. The expansion formed by the summation of these functions and the constant term provides the following multinomial approximation.

$$f(x_1, ..., x_N) \approx f_0 + \sum_{m=1}^{N} p_m(x_m) \tag{18}$$

This should be considered as a univariate additive decomposition approximation.

5 Reverting Lumping HDMR Results

The reverse algorithm for Lumping HDMR is based on the distance evaluations between the testing data and the training data. We use training data to construct an HDMR which is valid everywhere in an orthogonal hyperprismatic grid of the considered index space. Test data is also distributed in this grid but not at the nodes, instead, at certain internodal locations which are not known yet since the test data function values are not expressed in terms of the index coordinates. Therefore we need to find the appropriate locations for each test datum in the index space. To this end we can use the distances between each test datum and the all training data and evaluate the minimum distance. A single minimum value may not suffice to locate the testing datum in the index space. Then not only the shortest distance but first few shortest distance can be used to locate the testing point in the index space. This is done in a way such that there remains no doubt about the location. This holds of course for the criterion of shortest distance and some other criteria may be used in some other circumstances. We use shortest distance criterion in this work.

Using the structures of the training and the testing nodes as $(\xi_1, \xi_2, \ldots, \xi_N)$ and $(\mu_1, \mu_2, \ldots, \mu_N)$ respectively, we can rewrite the Euclidean distance for-

mula for our algorithm as follows.

$$d = \sqrt{\sum_{i=1}^{N}(\xi_i - \mu_i)^2} \tag{19}$$

Now, first, we will take the first testing node into consideration and evaluate the distances between this node and all training nodes. Then, we will find out the minimum distance value and select that training node as our source to determine the location of our testing node in our index space and say source index node for the index node of this determined training node. Next, for example, if we obtain a distance value of 0.6 and we have 5 independent variables, then we add 0.12 to each component of the source index node and assign this new node as an index node to the testing node. Of course, this addition process is not the only solution. This part must be improved to determine more realistic results. In this work, we deal only with the process. New algorithms for this step is left for the future work. However, this new node is the location of the testing node for the rest of our algorithm. This procedure will be repeated for all testing nodes to determine a location in the index space of our problem.

Once the location of the testing datum in the index space is determined then the rest is just a straightforward task, the evaluation of the function value via the analytic formula constructed by the univariately decomposed multivariate interpolation. Here we assume that all coordinate contributions are equally weighted.

These are, of course, very introductory steps for the reverse algorithm related to the Lumping HDMR. We still conduct certain works on this topic and show efforts to improve the efficiency of this algorithm.

6 Error Analysis

To examine the performance of this new method, in other words, to understand whether the obtained representations are or are not the acceptable solutions for the given engineering problems the following relative norm

$$\mathcal{N} = \frac{||f_{org} - f_{new}||}{||f_{org}||} \tag{20}$$

is evaluated. Here, f_{new} stands for the multivariate function obtained via Lumping HDMR method.

The norm values obtained close to zero by using this relation should be interpreted as the high performance of this new representation technique.

7 Numerical Implementations

The results of the numerical implementations given in this section are obtained by using certain program codes (scripts) written in MuPAD 4.0, Multi Processing Algebra Data tool [11, 12]. MuPAD codes have been run in 20 decimal digits numerical precision environment. These scripts are run on a PC of Core Duo T2050 1.60 GHz with a RAM capacity of 512 MB.

We select testing functions and the domains for independent variables of these testing functions to examine the performance of this new method. The analytical structures of our testing functions are as follows.

$$\begin{aligned} f_1(x_1,\ldots,x_5) &= \sum_{i=1}^{5} x_i, \\ f_2(x_1,\ldots,x_5) &= \left[\sum_{i=1}^{5} x_i\right]^2, \\ f_3(x_1,\ldots,x_5) &= \left[\sum_{i=1}^{5} x_i\right]^5 \end{aligned} \tag{21}$$

We have kept same the domains for each independent variable in all numerical implementations for simplicity. These domains are as follows. Each domain has a grid constructed by 1 increments starting from the lower bound up to and including the upper limit.

$$\begin{aligned} 1 \le x_1 \le 6, \qquad 3 \le x_2 \le 7, \qquad 4 \le x_3 \le 8, \\ 2 \le x_4 \le 6, \qquad 5 \le x_5 \le 8 \end{aligned} \tag{22}$$

The data set may have at most 3000 nodes when all nodes of the hyperprismatic grid corresponding to the prescribed domains are given in the problem. We construct different interpolation problems, having 1000 training nodes over this mentioned grid. These 1000 nodes are selected by a random function written by the authors. This selection is repeated several times and several different interpolation problems are constructed for each testing function having different nodes for training in order to examine the performance of the method more carefully. The relative error value is evaluated for each costructed problem. In average, as a result, the following relative error values are obtained for each testing function.

$$\begin{aligned} \mathcal{N}_{f_1} &= 0.11 \\ \mathcal{N}_{f_2} &= 0.21 \\ \mathcal{N}_{f_3} &= 0.48 \end{aligned} \tag{23}$$

The additive nature of the HDMR expansion causes obtaining better results for the multivariate interpolation problems having additive nature. As the nature of the problem begins to be less additive then the performance of the HDMR method hence, the performance of the Lumping HDMR method gets worse. This can be easily examined from the relative error values given

Table 1: Relative error values for the reverse algorithm.

	# of Testing Nodes		
	500	400	200
$f_1(x_1,\ldots,x_5)$	0.15	0.15	0.16
$f_2(x_1,\ldots,x_5)$	0.28	0.29	0.30
$f_3(x_1,\ldots,x_5)$	0.56	0.59	0.62

in (23) The main part of this work is the testing part of the Lumping HDMR, that is the reverse part of the method. The relative error results obtained for several interpolation problems having 500, 400 and 200 testing nodes through the reverse algorithm of the Lumping HDMR are given in Table 1. These results are the relative error values of the reverse algorithm including at most bivariate HDMR approximants. These results again are the averages of several runnings of the new algorithm through the constructed interpolation problems.

8 Concluding Remarks

This new algorithm is developed to bypass the disadvantages of the Generalized HDMR method. It bypasses the deadlock appearing in modelling, that is obtaining an analytical structure for the mentioned type of interpolation problems given in this work through Generalized HDMR. We always obtain an approximate analytical structure for the given multivariate interpolation problem. The training part and the reverse one, the testing part, of the algorithm gives acceptable results for the interpolation problems having additive nature.As the additivity of the function decreases and the multiplicativity nature of the sought function becomes more dominant then the performance of the algorithm gets worse.

For the future work, a better reverse algorithm should be developed for Lumping HDMR in order to have better approximations for the interpolation problems.

Acknowledgements: The second author is grateful to Turkish Academy of Sciences and both authors thank to WSEAS, for their supports.

References:

[1] I.M. Sobol, Sensitivity Estimates for Nonlinear Mathematical Models, *Mathematical Modelling and Computational Experiments (MMCE)*, 1, 1993, No.4.407.

[2] H. Rabitz and Ö. Alış, General Foundations of High Dimensional Model Representations, *J. Math. Chem.* 25, 1999, pp. 197–233.

[3] M. Demiralp, High Dimensional Model Representation and its Application varieties, *Mathematical Research* 9, 2003, pp. 146–159.

[4] M. Demiralp and M.A. Tunga, High Dimensional Model Representation of Multivariate Interpolation via Hypergrids, *The Sixteenth International Symposium on Computer and Information Sciences (ISCIS XVI)*, 2001, pp. 416–423.

[5] M.A. Tunga and M. Demiralp, A New Approach for Data Partitioning Through High Dimensional Model Representation, *Int. Journal of Computer Mathematics*, 2007, (accepted).

[6] M.A. Tunga and M. Demiralp, Data Partitioning via Generalized High Dimensional Model Representation (GHDMR) and Multivariate Interpolative Applications, *Mathematical Research* 9, 2003, pp. 447–462.

[7] M.A. Tunga and M. Demiralp, A Factorized High Dimensional Model Representation on the Partitioned Random Discrete Data, *Appl. Num. Anal. Comp. Math.* 1, 2004, pp. 231-241.

[8] M.A. Tunga and M. Demiralp, Introductory Steps for an Indexing Based HDMR Algorithm: Lumping HDMR, *1st WSEAS International Conference on Multivariate Analysis and its Application in Science and Engineering*, 2008, pp. 129–135.

[9] A.H. Zemanian, *Distribution Theory and Transform Analysis, An Introduction to Generalized Functions, with Applications*, Dover Publications Inc. New York, Reprint 1987.

[10] J.L. Buchanan and P.R. Turner, *Numerical Methods and Analysis*, McGraw-Hill, 1992.

[11] W. Oevel, F. Postel, S. Wehmeier and J. Gerhard, *The MuPAD Tutorial*, Springer, 2000.

[12] http://www.mupad.de