Response Surface Mesh with the Outer Input Method

Francisco Daniel Filip Duarte (danielduarte1181@gmail.com)
Independent https://orcid.org/0000-0003-0523-161X

Research Article

Keywords: response surface mesh, metamodel segmentation, artificial neural network, kriging

Posted Date: December 3rd, 2021

DOI: https://doi.org/10.21203/rs.3.rs-971011/v3

Title: Response Surface Mesh with the Outer Input Method

Author: Francisco Daniel Filip Duarte
Affiliation: Independent
Email address: danielduarte1181@gmail.com, danielduarte@vmxsoftwares.com
Orcid: https://orcid.org/0000-0003-0523-161X

Highlights:

1. Segmentation of any response surface with any number of design coordinates, and type of distribution such as Cartesian, random or clustered, in order to reduce training times.
2. Increases Kriging model training speed hundreds of times, for a 71x71 dataset, while maintaining high accuracy: from 80 minutes hour to 9 seconds.
3. Generates a response surface mesh, where individual mesh elements can be parallelized, and trained or updated independently.

Response Surface Mesh with the Outer Input Method

F. D. Filip Duarte [1],
danielduarte1181@gmail.com

[1] Independent, www.filipduarte.com

ABSTRACT: **Artificial intelligence in general and optimization tasks applied to the design of very efficient structures rely on response surfaces to forecast the output of functions, and are vital part of these methodologies. Yet they have important limitations, since greater precisions require greater data sets, thus, training or updating larger response surfaces become computationally expensive or unfeasible. This has been an important bottle neck limitation to achieve more promising results, rendering many optimization and AI tasks with a low performance.**

To solve this challenge, a new methodology created to segment response surfaces is hereby presented. Differently than other similar methodologies, this algorithm named outer input method has a very simple and robust operation, generating a mesh of near isopopulated partitions of inputs which share similitude. The great advantage it offers is that it can be applied to any data set with any type of distribution, such as random, Cartesian, or clustered, for domains with any number of coordinates, significantly simplifying any metamodel with a mesh ensemble.

This study demonstrates how one of the most known and precise metamodel denominated Kriging, yet with expensive computation costs, can be significantly simplified with a response surface mesh, increasing training speed up to 567 times, while using a quad-core parallel processing. Since individual mesh elements can be parallelized or updated individually, its faster operational speed has its speed increased.

Keywords: response surface mesh, metamodel segmentation, artificial neural network, kriging.

1. Introduction

Artificial intelligence (AI) plays an important role in society presently, due to its capacity to automate several complex or laborious tasks which before were only accomplished by humans. Present design practices of aeronautical, spatial and automotive structures also rely on metamodels used in AI to achieve approximations of precise and expensive models. Several metamodels are available in the public domain, yet the precision and the computation cost of these metamodels are often an object of research.

The outer input method (OIM) is a novel algorithm here described, which with a very simple methodology can create a mesh for a data set with any type of distribution or number of coordinates. It can be applied to any metamodel, thus expensive metamodels can be segmented and have their training time significantly reduced without compromising accuracy.

2. Related work

In the field of AI, Rumelhart et al [1]. described the backpropagation method, which is applied to train multi-level artificial neural networks (ANNs). These ANNs are a metamodel which consists of a mathematical model that mimics the flow of information observed in organic neuronal tissues of several living species, allowing many living organisms to have intelligence. This methodology has been very efficient in text, image, and speech recognition, among other applications. Among some of its generalized applications, ANNs can be applied to forecast the response of a mathematical function, called regression, and for category classification.

Deep learning [2] (DL), is a technique created by Lecun, Bengio, and Hinton, where many layers of ANNs are trained over data set to recognize several levels of patterns. It has unsupervised training, different than ANNS, and has many important applications for the automation of complex and challenging tasks, substituting human work in many fields of activity, with similar or greater levels of intelligence. DL and other areas of AI have as their basic building blocks, response surfaces such as ANNs, Kriging [3] (KR), support vector machine [4] (SVM), polynomial regression [5], radial basis functions [6] among others, and often require large parallel computing infrastructure and large data sets for their often computationally expensive training tasks.

Several global optimization algorithms (GOAs) are popular choices to find the extrapolated points of a function. These GOAs [7], [8], [9], [10], [11], have broad applications in engineering design, logistics, AI, among other areas.

In optimization, a metamodel, which is also called a response surface or surrogate model, can be applied to reduce computation costs by dismissing unpromising candidate solutions and suggesting promising alternatives. To evaluate the efficiency of a response surfaces, many benchmark functions were created [6].

Yet precise metamodels often require large data sets, and also are expensive in computation costs. Some studies aiming to reduce training times of KR have been published. Fuhg et al. in 2020 [12] published a review of the state-of-the-art of adaptative sampling methods for KR. Bouhlel et al. researched ways to reduce the computation cost of the KR model [13], by substituting the inversion of a matrix in the KR methodology with a kernel with few parameters defined by least squares and using adaptative sampling to not overload the KR model. In another study, van Stein et al. [14] described several methodologies to segment a KR model into clusters. Some of these techniques are based on slicing the domain in a Cartesian segmentation, yet those methodologies have limited application for data set with clustered distribution. Other methodologies, as described by Wang and Simpson [15] include a fuzzy analysis of the landscape of the function, to segment in locally optimal regions. Yet these proposed methodologies cannot be applied to any type of data set, and have several operational coefficients which must be adjusted by the user according to each function.

Wang et al. [16] in 2017 explored similar application of data sampling selection, including domain landscape analysis for KR clustering applied to optimization tasks. Liu et al. [17] described experiments to reduce computation time of the KR model for large data sets, by applying global and local sparse approximations of the overall inputs. Another alternatives presented include selecting subsets of the data set to reduce KR computation costs, and nesting KR, as described by Bachoc et al. [18] in 2021. A popular methodology also in the same filed of research, is the adaptative sampling [19], which Chellappa et al, in 2020., applied to the reduced basis method. This adaptative sampling consists of a surrogate error model generated to efficiently create sub samples of the parameter domain. There are many other publications available in the topic, aiming to reduce metamodel computation costs, which most often are based on a type of clustering or selective sampling the data set. Yet these methods often do not have a very simple implementation, requiring many operational parameters to be adjusted according to the data set, or cannot be applied to any type of data set distribution.

3. Methodology

The OIM generates a response surface mesh (RSM) for a normalized data set with a Cartesian, random, or clustered distribution, by separating the data set into elements within regions of similitude. This is achieved by consecutively dividing elements of data considered large, in two elements of data set, by its approximately middle section in the design space. For this the data set must be normalized, having each input between 0 and 1.

The advantage of this method is that it is very simple and robust, working equally with data sets with any number of coordinates or type of distribution while refining the mesh in regions of the domain with greater data density. By not overpopulating elements, it facilitates the generation of simplified response surface segments, and it allows to apply mesh refinement in regions of greater interest, simply by populating those regions. It also allows parallelization, and individual mesh elements to be updated separately.

The algorithm starts by assuming all the data set as the initial element. If the element size in terms of the number of inputs is greater than the maximum element size, it is sectioned. To section the element, initially is necessary to calculate the average position of the data set element, also called center position (CP). Secondly, the distance of each input with the CP is measured, and the point with greater distance with the CP is denominated outer input. It provides a rough estimation of the direction in which the variance of the position of the inputs is expected to be greater. The vector generated from CP to OI is denominated outer input vector (OIV), and is normal to a plane denominated approximated middle plane of the data set, which is used to divide the data set into two elements. The third step is to identify which points of the data set belong to each side of the plane, generating the element segmentation.

This process is repeated to each generated element until each element has a local population lower than the maximum allowed size. As an alternative that can be implemented, it is possible to define a minimum element size, to avoid error in the computation of each response surface element. Thus, if any element has a population lower than the minimum allowed size, additional points are added from the adjacent element.

Once all mesh elements are generated, each polarization vertex (PV) is defined by the average position of all inputs of a single element. Thus, each polarized mesh element (PME) is defined by the points of the data set which are closer to each PV, as in the Voronoi diagram, which is also called Dirichlet partition{Formatting Citation}. A given input forecast will be then calculated by the response surface element trained with the PME which has closer PV to the input. Another possible variation is to substitute the outer input vector by the eigenvector of the covariance matrix of the dataset.

The OIM is presented at Algorithm 1:

Algorithm 1: Outer Input Method (OIM)

```
1:  procedure OIM (Pop, par)
2:  load data set and maximum element size parameter
3:  while any cluster element has been segmented at the previous iteration, do
4:      for each cluster element, do
5:          if population size of the element > maximum element size, do
6:              calculate the CP of the cluster element
7:              measure the distance of the CP with all data inputs
8:              identify the OIV
9:              divide the element in two elements by the approximated middle plane
10:         end
11:     end
12: end
13: Generate PVs
14: Identify PMEs
```

The OIM also can be applied to generate a mesh for a finite element model or similar mathematical models, discretize the domain of a function and identify its most competitive outputs, among other applications which require domain partitioning. The OIM also can be applied to generate a mesh for a finite element model or similar mathematical models, discretize the domain of a function and identify its most competitive outputs, among other applications which require domain partitioning. Figure 1 displays the elements generated by the OIM segmentation, and Figure 2 displays the respective generated PMEs. Figure 3 displays the OIM segmentation of a data set with cluster distribution.

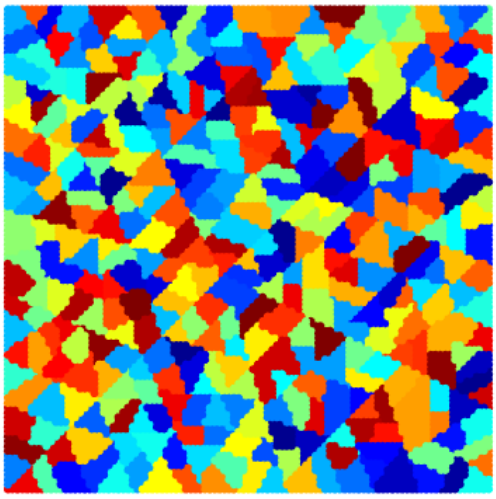

Figure 1 - A 100 x 100 input domain segmented by the OIM. Resulting segments of the OIM operation over a Cartesian domain, with maximum element size equal to 40 inputs.

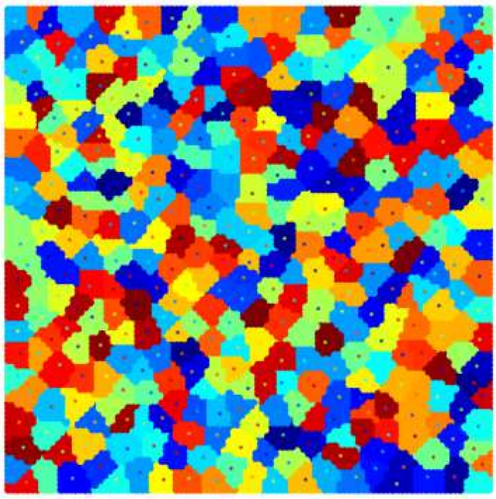

Figure 2 - Generated PMEs of the 100 x 100 domain. Each element defines the Voronoi partition of the domain, with the respective PVs points illustrated in different colors. These PMEs are generated from the segmented elements of figure 1, with the PVs of each PME identified in different colors

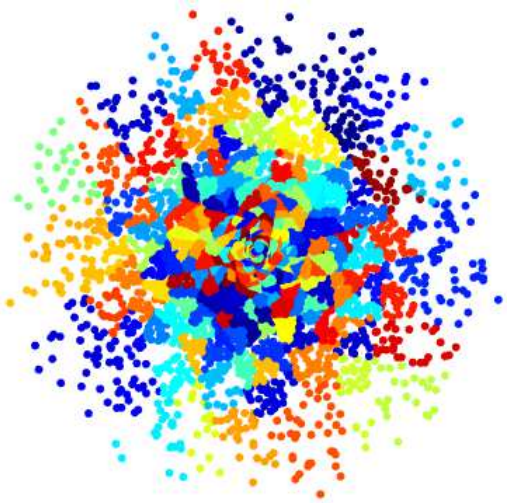

Figure 3 – A dataset with cluster distribution segmented by the OIM. A clustered data set segmented by the OIM by regions of similitude and maximum element size, showcasing how the OIM can be applied to any dataset with any type of distribution. It is possible to see that the mesh is refined in regions of greater input density.

4. Numerical experiment

In order to compare the efficiency of the response surface segmented by the OIM and the surface non-segmented, three benchmark functions used for this purpose are selected [6]. The regression metamodels applied in this experiment are the SVM which has fast a response and reasonable accuracy, and the KR, also called Gauss Process, very popular for its great accuracy, yet very expensive for larger datasets. Each function has its 2D domain defined in a Cartesian grid of 71 x 71 points, and the performance parameters including the overall training time are presented in tables 1 to 3. The metrics to measure performance are the average error of the output, and the variance of the error. Further parameters are the R square, RMAE, and RAAE, which are described by Jin, Chen, and Simpson [21]. It is important to note that for the RSM, each PME of the metamodel was trained in parallel using a quad-core computer, and the overall training time includes the mesh generation. The OIM has the only operational parameter named maximum element size, which in this experiment is equal to 50 inputs. The metamodels selected are the SVM and KR, and their OIM combinations. The performance comparison between the OIM segmented and non-segmented models are presented in Tables 1 to 3:

Holder Table function				
Metamodel	SMV	OIM-SMV	KR	OIM-KR
N Segments	1	133	1	133
Total training time	0.57 sec	8.63 sec	52 min 1.99 sec	7.38 sec
Mean error	1.116E-01	2.484E-02	1.712E-03	5.056E-04
Variance	2.416E-02	1.410E-03	1.332E-05	2.097E-06
R Square	8.776E-01	9.969E-01	1.000E+00	1.000E+00
RAAE	2.491E-01	3.680E-02	2.392E-03	7.051E-04
RMAE	1.798E+00	3.534E-01	5.421E-02	2.776E-02

Table 1 – Results for the Booth function

Ackley function				
Metamodel	SMV	OIM-SMV	KR	OIM-KR
N Segments	1	133	1	133
Total training time	0.71 sec	19.3 sec	37 min 53.27 sec	13.22s sec
Mean error	7.287E-02	4.333E-02	4.166E-02	8.830E-05
Variance	8.983E-03	3.212E-03	2.747E-03	1.624E-08
R Square	9.560E-01	9.916E-01	9.934E-01	1.000E+00
RAAE	1.607E-01	6.932E-02	6.475E-02	1.251E-04
RMAE	1.260E+00	4.559E-01	3.937E-01	1.194E-03

Table 2 – Results for Ackley function

Custom probability density function				
Metamodel	SMV	OIM-SMV	KR	OIM-KR
N Segments	1	133	1	133
Total training time	0.61 sec	7.63 sec	1 h:23 min 16.61 sec	8.81 sec
Mean error	3.367E-01	6.334E-03	1.640E-04	2.040E-04
Variance	6.525E-01	4.530E-04	1.476E-07	8.533E-08
R Square	-6.699E+00	9.877E-01	1.000E+00	1.000E+00
RAAE	1.127E+00	3.294E-02	8.230E-04	1.022E-03
RMAE	1.505E+01	1.293E+00	1.244E-02	1.198E-02

Table 3 – Results for the Custom probability function

From tables 1 to 3 is possible to see that the segmentation has increased the precision of the SVM, yet the mesh generation increased its evaluation time from less than one second to about 7 seconds. At Table 3, the SMV gave a large average error of 34%, yet its OIM combination gave an average error of 0.6 %, indicating the OIM can benefit SVM models. All other metrics also indicate the same, that the combination of the SVM with OIM benefits precision without significantly increasing computation time.

The KR model demonstrates in the experiment why is very popular among the metamodels, since it is very precise, like we see from the average error metric. Yet, is also very expensive in terms of computation time, ranging from 38 minutes up to 83 minutes for the 71 x 71 domain training. Its combination with the OIM significantly reduce training times to 13 seconds or less, a speed increment up to 567 times. Accuracy metrics show the KR combined with OIM has increased accuracy for the first two functions, and a slight reduction for the third. The large training speed gained, while maintaining very good accuracy, and having a very simple and robust implementation, demonstrates OIM is an efficient tool which can benefit several optimization and AI practices.

5. Conclusions

As demonstrated in the experiment, with the OIM, a SMV model can be segmented, increasing the accuracy for SMV, and reducing training time for KR. Maybe one of the great combinations of the OIM with metamodels, as demonstrated in this study, is to segment a KR model, and provide its great accuracy for larger datasets with much faster operation. With the OIM segmentation, KR training computation

was reduced from about 80 minutes to 9 seconds, a speed increase up to 567 times without compromising accuracy. With the OIM, KR models for larger datasets also can be parallelized and be scaled up to as computer memory and parallel processing allows, without excessively increasing training costs. Also, it is important to note the OIM mesh generation can perform even faster if coded in a language faster than MATLAB, like JAVA or C++.

Adding to this that the OIM can be implemented on any dataset with any type of distribution, it is demonstrated it is an important alternative for optimization and AI studies, to significantly reduce training costs of metamodels, and lead AI tasks and optimization studies to greater levels of performance.

6. References

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986, doi: 10.1038/323533a0.

[2] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553. Nature Publishing Group, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[3] C. E. Rasmussen, "Gaussian Processes in Machine Learning," in *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*, O. Bousquet, U. von Luxburg, and G. Rätsch, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 63–71.

[4] C. Cortes and V. Vapnik, "Support-Vector Networks," *Mach. Learn.*, vol. 297, pp. 273–297, 1995.

[5] R. Jin, W. Chen, and T. W. Simpson, "Comparative studies of metamodeling techniques under multiple modeling criteria," *8th Symp. Multidiscip. Anal. Optim.*, 2000, doi: 10.2514/6.2000-4801.

[6] C. Bogoclu and D. Roos, "A benchmark of contemporary metamodeling algorithms," *ECCOMAS Congr. 2016 - Proc. 7th Eur. Congr. Comput. Methods Appl. Sci. Eng.*, vol. 2, no. June, pp. 3344–3360, 2016, doi: 10.7712/100016.2039.7645.

[7] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *Proc. IEEE Int. Conf. Neural Networks*, pp. 1942–1948, 1995.

[8] C. R. Reeves and J. E. Rowe, *Genetic Algorithms - Principles and Perspectives. A guide to GA Theory.*, vol. 4, no. 1. 2003.

[9] T. H. A. Scollen, "Simulated Annleaing, Introduction, Application and Theory," *Nove Sci. Publ. Inc. New York*, 2018.

[10] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces," *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, 1997, doi: 10.1023/A:1008202821328.

[11] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems," *Eng. Comput.*, vol. 29, no. 1, pp. 17–35, 2013, doi: 10.1007/s00366-011-0241-y.

[12] J. N. Fuhg, A. Fau, and U. Nackenhorst, "State-of-the-Art and Comparative Review of Adaptive Sampling Methods for Kriging," *Arch. Comput. Methods Eng.*, vol. 28, no. 4, pp. 2689–2747, 2021, doi: 10.1007/s11831-020-09474-6.

[13] M. A. Bouhlel, N. Bartoli, A. Otsmane, and J. Morlier, "Improving kriging surrogates of high-dimensional design models by Partial Least Squares dimension reduction," *Struct. Multidiscip. Optim.*, vol. 53, no. 5, pp. 935–952, 2014, doi: 10.1007/s00158-015-1395-9.

[14] B. van Stein, H. Wang, W. Kowalczyk, M. Emmerich, and T. Bäck, "Cluster-based Kriging approximation algorithms for complexity reduction," *Appl. Intell.*, vol. 50, no. 3, pp. 778–791, 2020, doi: 10.1007/s10489-019-01549-7.

[15] G. G. Wang and T. Simpson, "Fuzzy clustering based hierarchical metamodeling for design space reduction and optimization," *Eng. Optim.*, vol. 36, no. 3, pp. 313–335, 2004, doi: 10.1080/0305215031000163991l.

[16] H. Wang, B. van Stein, M. Emmerich, and T. Bäck, "Time Complexity Reduction in Efficient Global Optimization Using Cluster Kriging," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2017, pp. 889–896, doi: 10.1145/3071178.3071321.

[17] H. Liu, J. Cai, Y.-S. Ong, and Y. Wang, "Understanding and comparing scalable Gaussian process regression for big data," *Knowledge-Based Syst.*, vol. 164, pp. 324–335, 2019, doi: https://doi.org/10.1016/j.knosys.2018.11.002.

[18] F. Bachoc, N. Durrande, D. Rullière, and C. Chevalier, "Properties and comparison of some Kriging sub-model aggregation methods," Feb. 2021. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01561747.

[19] S. Chellappa, L. Feng, and P. Benner, "An Adaptive Sampling Approach for the Reduced Basis Method." 2020.

[20] A. Jara, "Theory and computations for the Dirichlet process and related models: An overview," *Int. J. Approx. Reason.*, vol. 81, pp. 128–146, 2017, doi: https://doi.org/10.1016/j.ijar.2016.11.008.

[21] R. Jin, W. Chen, and T. W. Simpson, "Comparative studies of metamodelling techniques under multiple modelling criteria," *Struct. Multidiscip. Optim.*, vol. 23, no. 1, pp. 1–13, 2001, doi: 10.1007/s00158-001-0160-4.