ECCOMAS Congress 2016
VII European Congress on Computational Methods in Applied Sciences and Engineering
M. Papadrakakis, V. Papadopoulos, G. Stefanou, V. Plevris (eds.)
Crete Island, Greece, 5–10 June 2016

A BENCHMARK OF CONTEMPORARY METAMODELING ALGORITHMS

Can Bogoclu[1] **& Dirk Roos** [2]

Institute of Modeling and
High-Performance Computing
Niederrhein University of Applied Sciences, Germany
[1] can.bogoclu@hs-niederrhein.de
[2] dirk.roos@hs-niederrhein.de

Keywords: surrogate model, anisotropic Kriging, moving least squares, radial basis functions, support vector regression, artificial neural networks

Abstract. *Growing popularity of probabilistic and stochastic optimization methods in engineering applications has vastly increased the number of sampling points required to obtain a solution. Depending on the complexity of the underlying physical model, this often proves to be a computationally burdensome challenge. In order to overcome this challenge, one possible approach is to use surrogate models (metamodels), which approximate the responses of the physical model in a given variable subspace.*

In the past years, many different metamodeling algorithms such as Gaussian process (Kriging), moving (weighted) least squares, radial basis functions, regression neural networks and support vector regression have been suggested as an alternative to ordinary least squares (polynomial regression). The choice of the best metamodeling algorithm for any application is not a trivial task. Although there has been previous research to compare at least some of these methods to some extent ([1, 2, 3]), only a small number of publications compare all of these algorithms over a large number of multidimensional functions with varying characteristics.

In this paper, a comparison of the aforementioned methods is carried out. Using well-known analytical test functions for optimization, this paper aims to shed some light on the question of which algorithm performs best under which conditions. Apart from the structure of analytical test functions, the influence of the number of sampling points and the amount of noise in the observations on the performance of metamodeling algorithms is investigated.

1 INTRODUCTION

In physics and engineering, nature is understood and described by mathematical models, which are abstractions of reality. Although the underlying reality does not change, these models are updated constantly, whenever a newer theory proposes better results and a better understanding of our observations.

In this respect, a metamodel or a surrogate model is a mathematical abstraction of a physical model that tries to mimic the dependencies between the input and output parameters. This abstraction is useful if the computational cost of the physical model is too high and a large number of calculations are needed, as is the case for model optimization and reliability analysis. Furthermore, finding metamodels to describe the experimental data is also possible, even if a physical model is not available or is too expensive to compute.

On the other hand, some information loss is inevitable because of the abstraction. The amount of this information loss or error depends on a number of aspects. Besides the validity of the physical model, the choice of metamodeling algorithm, number of sample points used to train the algorithm and the amount of numerical or experimental noise all contribute to the amount of error in a metamodel. Therefore, this benchmark aims to test some of the existing algorithms under different conditions. There are also other sources of error, such as the inclusion of unimportant variables and the method for the design of experiments but these issues are not addressed within this benchmark.

2 METAMODELING

Before beginning with the benchmark, the algorithms to be tested will be introduced briefly. Our aim here is not to give instruction for building these algorithms but to address similarities between these algorithms by not conforming to individual choice of notations for each method and using similar symbols for similar elements of each metamodel. For more detailed instructions on how to build these algorithms, we refer to the corresponding citations.

2.1 Ordinary least squares

Ordinary least squares (OLS) or polynomial regression is one of the earliest and easiest methods for surrogate modeling. First, the input vector is mapped to a polynomial. After that, the coefficients of each monomial term are determined. The prediction function is the sum of the multiplication of coefficients with their corresponding monomials.

Let $\mathbf{x} \in \mathbb{R}^n$ be an n-dimensional vector with elements $\mathbf{x} = [x_1, x_2, x_3, \ldots, x_n]$ and let $y = f(\mathbf{x})$ be the function that is trying to be approximated. The prediction function for a polynomial of degree d can be written as

$$\tilde{f}(\mathbf{x}) = \quad c_0 + c_1 x_1 + \cdots + c_n x_n + c_{n+1} x_1^2 + c_{2n} x_1 x_n + \cdots + c_{\frac{n(n+1)}{2}} x_n^2 + \cdots + c_k x_n^d$$

or in matrix form

$$\begin{aligned} \tilde{f}(\mathbf{x}) &= \mathbf{p}(\mathbf{x}, d) \cdot \mathbf{c} \\ \mathbf{c} &= [c_0, c_1, c_2, ..., c_k]^T \end{aligned}$$

where $\mathbf{p}(\mathbf{x}, d)$ is the polynomial mapping function of order d for the n-dimensional input vector $\mathbf{x}$. Coefficient vector $\mathbf{c}$ must be determined by using m sampling points. The least squares problem is underdetermined for $m < k$. The relationship between the number of monomial

terms k, number of input parameters n and polynomial degree d is given by

$$k = \binom{n+d}{d} = \frac{(n+d)!}{d!n!} \tag{1}$$

where $(\cdot)!$ denotes the factorial operation. If the matrix of sample points $\mathbf{X}^0$ and the vector of responses $\mathbf{y}^0$ has more than k entries, coefficient vector $\mathbf{c}$ can be obtained by minimizing

$$\min_{\mathbf{c}} \sum_{i=0}^{m} (y_i^0 - \tilde{f}(\mathbf{x}_i^0))^2 \tag{2}$$

the sum of squared errors (SSE). This is a quadratic system of equations, which can be solved easily by the Newton's method. The only free parameter of OLS is the polynomial degree d. Figure 1a, shows that the quality of the surrogate model can increase for larger values of d. On the other hand, polynomial degree d has a large influence on the number of sampling points m needed for the approximation. Eq. 1 shows that the number of monomials increase rapidly with the increasing number of input parameters n and the polynomial degree d. This represents a great setback for this method. Further details about OLS and how to efficiently solve least squares problems can be found in [4]. In this benchmark, d is varied between 2 and 4 and the best result is recorded.

2.2 Moving least squares

Another method which has proved to be useful for metamodeling is the so called moving least squares (MLS). Similar to OLS, the input vector is mapped to a polynomial. Instead of determining global coefficients $\mathbf{c}$, local coefficients $\mathbf{c}(\mathbf{x})$ are obtained by solving a weighted least squares problem [5]. The prediction function

$$\tilde{f}(\mathbf{x}) = \mathbf{P}(\mathbf{x}, d)^T \cdot \mathbf{c}(\mathbf{x})$$

is similar to OLS, except the coefficients $\mathbf{c}(\mathbf{x})$ are not constant but functions of the approximation point $\mathbf{x}$.

Coefficients $\mathbf{c}(\mathbf{x})$ depend on the weight function $w(\mathbf{x})$ which determine the influence domain $D := \{\mathbf{x}_i^0 \in D \mid \|\mathbf{x} - \mathbf{x}_i^0\|_2 \leq \theta_r\}$ with the radius θ_r. If the sample points lie in this domain, the weight function $q(\mathbf{x})$ takes values between $0 < q(\mathbf{x}) \leq 1$, while it is set to $q(\mathbf{x}) = 0$ outside of the influence domain. Following Gaussian function [6]

$$q(r_i) = \frac{e^{-(r_i/\alpha\theta_r)^2} - e^{-\alpha^{-2}}}{1 - e^{-\alpha^{-2}}}$$

is chosen as the weight function with shape parameter $\alpha = 0.5$ for this study. Parameter r_i denotes the Euclidean distance between $\mathbf{x}$ and the sample point $\mathbf{x}_i^0$. The weighted least squares problem can be written as

$$\min_{\mathbf{c}} \sum_{i=0}^{m} q(r_i)(y_i^0 - \tilde{f}(\mathbf{x}_i^0))^2$$

similar to Eq 2. It is easier to formulate and solve this system of equations in matrix form. Local coefficients can be directly obtained by solving

$$\begin{aligned} \mathbf{c}(\mathbf{x}) &= \mathbf{P}_0^T \mathbf{Q}(\mathbf{x}) \cdot \mathbf{y}^0 \cdot (\mathbf{P}_0^T \mathbf{Q}(\mathbf{x}) \mathbf{P}_0)^{-1} \\ \mathbf{Q} &= \operatorname{diag}(q(r_1), q(r_2), \ldots, q(r_m)) \\ \mathbf{P}_0 &= \left[\mathbf{P}(\mathbf{x}_1^0, d), \ldots, \mathbf{P}(\mathbf{x}_m^0, d)\right]^T \end{aligned} \tag{3}$$

where r_i denotes the distance between the prediction point and sample point $\mathbf{x}_i^0$. This expression must be computed for each prediction point. This is illustrated in Figure 1b for different sample points. Each colour represents the local least squares approximation of a prediction point. Although the prediction function may seem to be more complicated, better approximations with lower degrees of polynomials can be achieved by this method. The polynomial degree in Figure 1b and during this benchmark is set to 2. Compared to Figure 1a, it can be seen that the quality of approximation is much better for the same polynomial degree. On the other hand, the radius of the influence domain θ_r is a free parameter and must be determined by optimization. This process is called training. Training becomes computationally more expensive with increasing number of free parameters. Nevertheless, this one dimensional optimization problem can often be solved quite fast even with stochastic optimization methods.

2.3 Kriging (Gaussian process)

Kriging is an interpolation method, which uses the spatial correlation between the data points to obtain weights for any prediction point. The prediction function for the Kriging model can be written as [7]

$$\tilde{f}(\mathbf{x}) = \mathbf{P}(\mathbf{x}, d)\hat{\mathbf{c}} + \mathbf{q}(\mathbf{x})\mathbf{Q}(\mathbf{X}^0)^{-1}(\mathbf{y}^0 - \mathbf{P}_{0,d}^T\hat{\mathbf{c}}) \tag{4}$$

and consists of two additive terms.

The first term of the summation is a polynomial regression. In ordinary Kriging, the polynomial degree is zero, hence the first term is a constant value. This can also be seen as the expectation value of a Gaussian process that this model represents. In the case of universal Kriging, polynomial degree may be greater than zero. In this case, the Gaussian process has different expected values at different locations.

The second term of the summation is the interpolating part. If the model is trained well, this part is equal to the error of the polynomial regression at each observed point. There are different variations of the spatial correlation functions $\mathbf{q}(\mathbf{x})$ and the choice of the correlation function has great impact on the surrogate model. Squared exponential

$$q_{i,sq}(\mathbf{x}) = \exp\left(-\sum_{j=1}^{n}\theta_j\left(x_j - X_{i,j}^0\right)^2\right) \tag{5}$$

and absolute exponential functions

$$q_{i,abs}(\mathbf{x}) = \exp\left(-\sum_{j=1}^{n}\theta_j\left|x_j - X_{i,j}^0\right|\right) \tag{6}$$

are used in this benchmark. Notice the index j in Eq. 5 and Eq. 6. It denotes that there may be different values for the free parameter θ_j for different input dimensions. This is the difference between anisotropic and the other types of Kriging mentioned above. For the latter, θ_j is constant for all dimensions. Theoretically, the set of possible solutions for anisotropic Kriging includes the solutions for other types of Kriging (i.e. setting all polynomial coefficients to zero except the constant term and setting all values of θ_k to some constant). Hence the global optimal solution for anisotropic Kriging should be at least as good as the other types of Kriging. On the other hand, there is no guarantee that the chosen optimization algorithm will always find the global optimum. Universal and anisotropic Kriging are therefore both trained with both correlation functions. The best result for each function is shown in the benchmark.

To determine θ, the maximum likelihood function is used as an alternative to SSE as the objective function of optimization. After θ is found, coefficients $\hat{\mathbf{c}}$ can be obtained through the solution of quadratic equation systems as in OLS 2 and in MLS 3.

Figure 1c shows the first part of the Eq. 4 and the final solution of Kriging compared to OLS for a second order polynomial. It is noticeable that the polynomial provides only a plane for the mean value, while the correlation function accounts for the variance on each data point and makes this method an interpolation. The polynomial degree d for this method is also set to 2 during this benchmark.

2.4 Radial basis functions

Another way of metamodeling is to interpolate the data points with radial basis functions (RBF) of distance. The prediction function

$$\tilde{f}(\mathbf{x}) = \sum_{i=1}^{m} c_i q\left(\left\|\mathbf{x} - \mathbf{x}_i^0\right\|_2, \theta\right) \tag{7}$$

is a linear combination of radial basis functions $q(\cdot)$. In the above equation, $\mathbf{x}_i^0$ denotes the i-th of m data points and $\|\cdot\|_2$ the Euclidean norm. Function $q(\cdot)$ is the chosen radial basis function which involves a free parameter θ. The choice of the right RBF is crucial and there are many possibilities. Some authors even suggest using anisotropic [8] (like in anisotropic Kriging) or locally bounded [1] (like in MLS) radial basis functions. In our preliminary tests, we found that the following three RBF outperformed others for our test functions. Hence, we chose to use the multiquadratic RBF

$$q(\mathbf{x}, \mathbf{x}_i^0) \quad = \sqrt{\|\mathbf{x} - \mathbf{x}_i^0\|_2^2 + \theta^2}$$

the cubic RBF

$$q(\mathbf{x}, \mathbf{x}_i^0) \quad = (\|\mathbf{x} - \mathbf{x}_i^0\|_2 + \theta)^3$$

and the Gaussian RBF

$$q(\mathbf{x}, \mathbf{x}_i^0) \quad = \exp(-\theta \|\mathbf{x} - \mathbf{x}_i^0\|_2^2)$$

for our benchmark. Like for Kriging, the best solution is chosen with a posteriori knowledge and shown in the benchmark. After choosing the radial basis function and obtaining the free parameter θ, coefficients c_i can be determined by minimizing SSE as before and the metamodel is trained. Since there are equal number of coefficients to the data points, the solution is an interpolation between these points through the RBF. This process is visualized in Figure 1d for Gaussian RBF.

2.5 Support vector regression

Originally a classification algorithm, support vector machines seek to find the hyperplane that separates the data with a maximized minimum distance to the margin [9]. In ϵ-support vector regression (SVR) this margin is used as the approximation function with a desired precision ϵ. This is demonstrated in Figure 1e for a data set with an exaggerated amount of error for visual purposes. Furthermore, the flattest plane is sought that achieves the above condition if more than one exist.

For linearly separable data, the prediction function

$$\tilde{f}(\mathbf{x}) = \hat{\mathbf{c}}^T\mathbf{x} + b$$

is similar to OLS. However, the coefficients are obtained not through minimizing the SSE but by solving the following optimization problem

$$\min \frac{1}{2}\|\mathbf{c}\|_2^2 + \theta^P \sum_{i=r}^{m}(\zeta_i + \zeta_i^*) \tag{8}$$
$$\text{s.t.} \begin{cases} y_i^0 - \hat{\mathbf{c}}^T\mathbf{x} - b \leq \epsilon + \zeta_i \\ -y_i^0 + \hat{\mathbf{c}}^T\mathbf{x} + b \leq \epsilon + \zeta_i^* \\ \zeta_i, \zeta_i^* \geq 0 \end{cases}$$

to ensure flatness. ζ_i and ζ_i^* are slack variables in Eq. 8 which relax the constraints and the free parameter θ^P is the penalty term for the violation of the constraints with ϵ. This problem can be solved for a given free parameter ϵ and θ^P. Since we seek to optimize the free parameters, training of this metamodel becomes a two step optimization problem as in Kriging, MLS and RBF.

For the solution of nonlinear problems, the input space is mapped to a higher dimensional space by the mapping operator $\phi(\mathbf{x})$ for linearization. The prediction function is modified as

$$\tilde{f}(\mathbf{x}) = \hat{\mathbf{c}}^T\phi(\mathbf{x}) + b$$

for this purpose. Obtaining the mapping function $\phi(\mathbf{X})$ explicitly is often very difficult. Instead, the so called kernel trick is generally used to define the mapping function implicitly by defining a kernel function $q_{i,j}(\mathbf{x}_i, \mathbf{x}_j)$ such that the kernel function

$$q(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$$

replaces the mapping operator $\phi(\mathbf{x})$ and the prediction function

$$\tilde{f}(\mathbf{x}) = \sum_{i=r}^{m} c_i q(\mathbf{x}_i^0, \mathbf{x}) + b$$

can be expressed without $\phi(\mathbf{x})$. It can be seen that only $m - r$ of the m data points are used for the prediction. In this benchmark, r is set to zero since the number of sample points are relatively small. As kernel functions, sigmoid function

$$q(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\theta\mathbf{x}_i^T\mathbf{x}_j)$$

polynomial of degree $d = 2$

$$q(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T\mathbf{x}_j + \theta)^d$$

and the Gaussian radial basis function

$$q(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\theta^2}\right)$$

are used. As before, the best results for each test function are shown but in practical applications, choosing the right kernel may become exhausting. The optimization problem in Eq. 8 must be reformulated to include the kernel mapping. This can be found in [9] and will not be expressed here.

2.6 Artificial neural networks

An artificial neural network (ANN) is a network of mathematical mapping operators, called neurons. Neurons take an input (a scalar, a vector or a matrix) that is passed to them and map it to an output with the activation function (kernel) that is stored in them. The Network consists of one or more layers. Neurons on each layer are connected to the neighboring layers. A more detailed description can be found in [10]. Among other purposes like classification and optimization, an ANN can also be used for metamodeling. In this benchmark two variations of radial basis neural networks are tested for this purpose. Such an ANN consists of one input layer, one or more hidden layers and one output layer. For each input dimension, there is a neuron on the input layer. The first hidden layer contains the radial basis neurons and the output layer contains the approximated value for the given input (Figure 1f).

The prediction function is given as

$$\tilde{f}(\mathbf{x}) = \sum_{j=1}^{m} c_j \phi_j(\mathbf{x}) + b_j$$
$$\phi_j(\mathbf{x}) = \exp\left(-\theta_j \left\|\mathbf{x} - \mathbf{x}_j^0\right\|_2\right)^2$$

where ϕ_j is the output of j-th neuron on the hidden layer and m is the number of neurons on the hidden layer which is set to the number of data points in this benchmark. Free parameter θ is the so called bias of the neurons in the hidden layer and must be obtained through optimization. Coefficient vectors $\mathbf{c}$ and $\mathbf{b}$ can be found linearly by minimizing SSE for a given θ.

The similarity between the Gaussian radial basis function and such a network is obvious. Apart from the additive coefficients b_j, the prediction functions are the same, although the formulations are quite different. Such a network also interpolates the data.

Another more generalized formulation of a regression neural network is also tested in this benchmark as described in [11]. The main difference between both formulations are in the second layer so only the first part or the second layer

$$\tilde{f}(\mathbf{x}) = \sum_{j=1}^{m} \frac{\mathbf{y}_0^T \phi_j(\mathbf{x})}{\|\phi_j(\mathbf{x})\|_2}$$

of the prediction function is modified. As before, both of these methods are tested and the best results for each test function are shown in the benchmark.

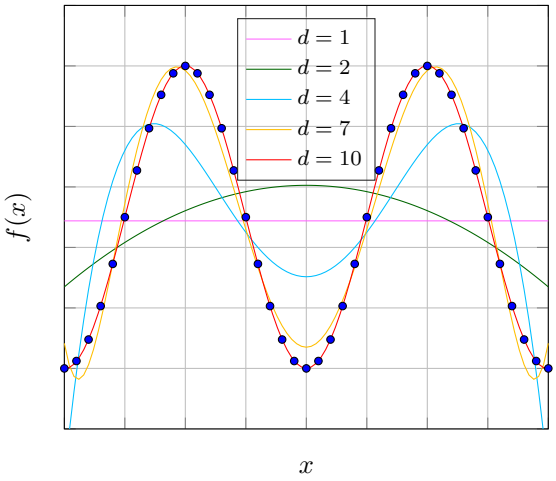
3 BENCHMARK

The benchmark consists of 15 test functions. DoEs are created for each function and each number of sample points by an optimized LHS method similar to [12], that takes both distance and correlation into account. First, the number of sample points is varied stepwise. Consecutively, Gaussian white noise is introduced to the model additively

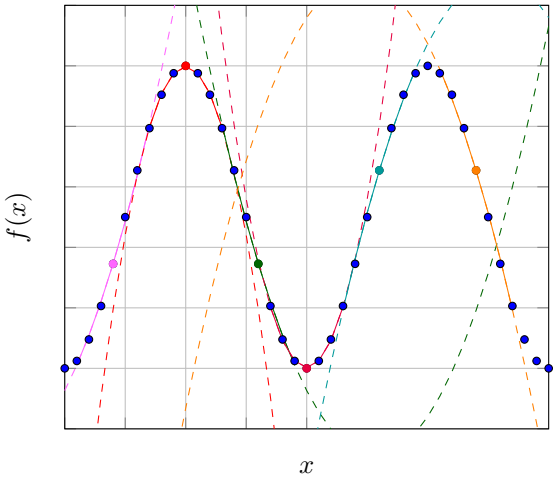
$$\hat{y}_i^0 = y_i^0(1 + \mathcal{N}(0, L_N))$$

and the level of noise L_N is varied stepwise for a fixed number of sample points.

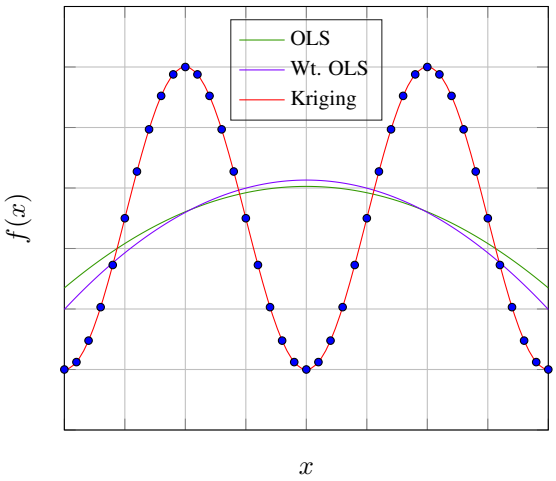
Furthermore, optimization of the metamodels is achieved by a particle swarm optimization algorithm to evade local minimums in combination with a local deterministic optimizer. A k-fold cross-validation is used to avoid overfitting. In this method, the training data is divided into



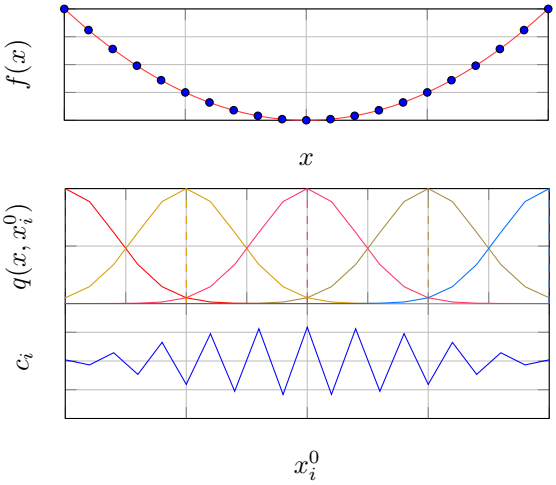
(a) OLS: Colours represent the polynomial order. Blue points represent the training data.



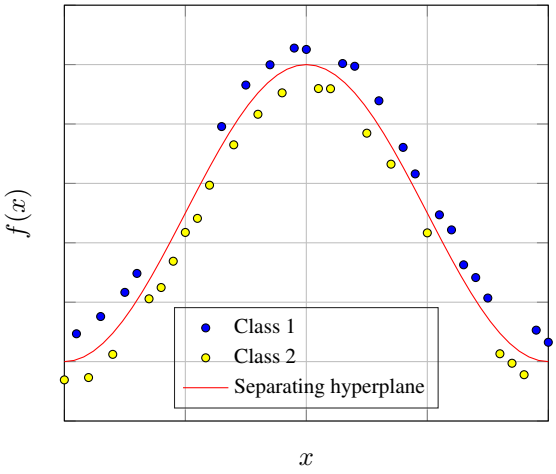
(b) MLS: Colours represent prediction points and their local approximation.



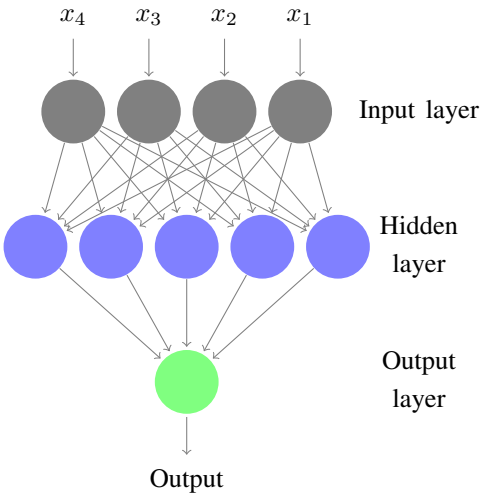
(c) Kriging (GP): The correlation matrix is used as weights for OLS. Errors of Wt. OLS are interpreted as variance and corrected by spatial correlation.



(d) RBF: The prediction function (top) is the linear combination of RBF (middle) and the coefficient vector c_i (bottom). Colours represent prediction points.



(e) SVR: Error on data points are exaggerated to visualize classification and the margin used as metamodel.



(f) ANN: All neurons are mathematical transformations that are interconnected through the layers.

Figure 1: Visualisation of metamodeling methods

k packages. $k-1$ packages are used for the training and one package for testing each iteration. The SSE is calculated by iterating over all test packages and used as the objective function for determining the free parameters. The number of folds k was set to 5.

Finally, metamodels are trained with the original and the standardized data to record the best result. Standardization of a variable $\mathbf{z}$ is done by

$$\overline{\mathbf{z}} = \frac{(\mathbf{z} - \mu_z)}{\sigma_z} \tag{9}$$

with mean vector μ_z and variation vector σ_z^2. Standardized data includes a standardized input matrix $\overline{\mathbf{x}}^0$ and a standardized output vector $\overline{\mathbf{y}}^0$.

3.1 Test functions

The following test functions were chosen for this benchmark:

1. Ackley function [13]:

$$f(\mathbf{x}) = -20\exp\left(-\frac{1}{5}\sqrt{\frac{1}{2}(x_1^2 + x_2^2)}\right) - \exp\left(\frac{1}{2}(\cos(2\pi x_1) + \cos(2\pi x_2))\right) + e + 20$$
$$-5 \leq x_{1,2} \leq 5$$

2. Beale function [14]:

$$f(\mathbf{x}) = (1.5 - x_1 + x_1x_2)^2 + \left(2.25 - x_1 + x_1x_2^2\right)^2 + \left(2.625 - x_1 + x_1x_2^3\right)$$
$$-4.5 \leq x_{1,2} \leq 4.5$$

3. Booth function [14]:

$$f(\mathbf{x}) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$$
$$-10 \leq x_{1,2} \leq 10$$

4. Bukin function no. 6 [15]:

$$f(\mathbf{x}) = 100\sqrt{|x_2 - 0.01x_1^2|} + 0.01|x_1 + 10|$$
$$-15 \leq x_1 \leq -5;\ -3 \leq x_2 \leq 3$$

5. Cross-In-Tray function [16]:

$$f(\mathbf{x}) = -10^{-4}\left(\left|\sin(x_1)\sin(x_2)\exp\left(\left|100 - \frac{\sqrt{x_1^2 + x_2^2}}{\pi}\right|\right)\right| + 1\right)^{0.1}$$
$$-5 \leq x_{1,2} \leq 5$$

6. Custom probability density function:

$$f(\mathbf{x}) = \left(1 + x_1^4 + 5x_2^4 + 2x_2^2\right)^{-1}$$
$$-3 \leq x_{1,2} \leq 3$$

7. Easom function [17]:

$$f(\mathbf{x}) = -\cos(x_1)\ \cos(x_2)\ \exp\left(-(x_1-\pi)^2-(x_2-\pi)^2\right)$$
$$0 \leq x_{1,2} \leq 6$$

8. Eggholder function [16]:

$$f(\mathbf{x}) = -(x_2+47)\sin\left(\sqrt{\left|x_2+\frac{x_1}{2}+47\right|}\right) - x_1\sin\left(\sqrt{|x_1-x_2-47|}\right)$$
$$-512 \leq x_{1,2} \leq 512$$

9. Goldstein-Price function [18]:

$$f(\mathbf{x})) = \left(1+(x_1+x_2+1)^2\left(19-14x_1+3x_1^2-14x_2+5x_1x_2+3x_2^2\right)\right)$$
$$\cdot\left(30+(2x_1-3x_2)^2\left(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2\right)\right)$$
$$-2 \leq x_{1,2} \leq 2$$

10. Holder Table function [16]:

$$f(\mathbf{x}) = -\left|\sin(x_1)\cos(x_2)\exp\left(|1-\pi^{-1}\sqrt{x_1^2+x_2^2}|\right)\right|$$
$$-3 \leq x_{1,2} \leq 3$$

11. Levy function No.13 [16]:

$$f(\mathbf{x}) = \sin^2(3\pi x_1)+(x_1-1)^2\left(1+\sin^2(3\pi x_2)\right)+(x_2-1)^2\left(1+\sin^2(2\pi x_2)\right)$$
$$-6 \leq x_{1,2} \leq 6$$

12. Michalewicz function [19]:

$$f(\mathbf{x}) = -\sum_{i=1}^{n}\sin(x_i)\sin^{20}\left(\frac{i\cdot x_i}{\pi}\right)$$
$$0 \leq x_{1,2} \leq \pi$$

13. Six hump camel function [20]:

$$f(\mathbf{x}) = 4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^42$$
$$-2 \leq x_{1,2} \leq 2$$

14. Three hump camel function [16]:

$$f(\mathbf{x}) = 4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^42$$
$$-5 \leq x_{1,2} \leq 5$$

15. Ursem function [21]:

$$f(\mathbf{x}) = -\sin(2x_1 - 0.5\pi) - 3\cos(x_2) - 0.5x_1$$
$$-2 \leq x_{1,2} \leq 2$$

Some of these functions are very easy to approximate (i.e. functions 2 and 3), while some are extremely challenging (i.e. functions 8 and 10). The majority of these functions should be moderately difficult to approximate for the tested metamodeling algorithms. In this way, some variance on the complexity of the structure of the test functions is achieved. The purpose of this variance is to reduce the bias of the benchmark such that no algorithm is favoured. Functions are expressed by means of equations and input bounds.

3.2 Results

Root mean squared error (RMSE) of 16384 validation points created from a Sobol sequence [22] is used as results in Table 1 and Table 2. Smaller values of RMSE represent a better model quality. The number of sample points are varied incrementally between 32, 64, 96 and 128 first. Consecutively, noise levels are varied between 1%, 2.5%, 5% and 10% incrementally while the DoEs with 128 points are reused. It can be argued that 10% noise is quite a lot but since we suggest using metamodels to model experimental data, such a high level of noise can still be interesting.

Besides the RMSE for each function, the mean value $mean_{Err}$ of RMSE and the standardized mean value $\overline{mean}_{Err}$ are displayed for each category (see Figure 2 and Figure 3). The mean value $mean_{Err}$ can be used as a measure to compare an algorithm with itself between the categories but it is biased since functions with greater prediction errors have a bigger impact on the results. Standardization is needed to equalize the weight of each function and make the score more independent of the magnitude of errors. Hence, each evaluation of a test function is standardized with the mean and the standard deviation of the root mean squared error of all algorithms for that evaluation. The mean value of the standardized error $\overline{mean}_{Err,k}$ of algorithm k can be expressed as shown in Eq. 9

$$\xi_{i,k} = \frac{RMSE_{i,k} - mean(RMSE_{i,all})}{std(RMSE_{i,all})}$$
$$\overline{mean}_{Err,k} = \frac{\sum_{i=1}^{15} \xi_{i,k}}{15}$$

and will be used for the categorical comparison of algorithms with each other. The variable $RMSE_{i,all}$ denotes the RMSE of all algorithms for the function i and does possess the same mean value as $mean_{Err}$ which denotes the mean of the RMSE along all functions in a category.

Samp.	Func.	ANN	GP	MLS	OLS	RBF	SVR
32	1	$7.17E-01$	$7.69E-01$	$7.83E-01$	$1.11E+00$	$\mathbf{6.95E-01}$	$7.83E-01$
	2	$8.48E+03$	$7.85E+03$	$1.19E+04$	$1.53E+04$	$\mathbf{3.30E+03}$	$1.82E+04$
	3	$2.41E-04$	$\mathbf{1.26E-13}$	$4.66E-13$	$6.24E-12$	$4.74E-04$	$2.37E+01$
	4	$1.37E+01$	$1.31E+01$	$1.49E+01$	$1.61E+01$	$1.31E+01$	$\mathbf{1.19E+01}$
	5	$2.03E-01$	$\mathbf{1.13E-01}$	$2.02E-01$	$2.11E-01$	$1.77E-01$	$2.04E-01$
	6	$\mathbf{6.11E-02}$	$6.58E-02$	$6.84E-02$	$1.51E-01$	$6.12E-02$	$1.04E-01$
	7	$\mathbf{5.40E-02}$	$6.43E-02$	$7.46E-02$	$1.40E-01$	$5.73E-02$	$6.09E-02$
	8	$\mathbf{2.91E+02}$	$3.23E+02$	$3.64E+02$	$3.51E+02$	$2.98E+02$	$3.18E+02$
	9	$\mathbf{1.41E+04}$	$1.92E+04$	$4.13E+04$	$5.13E+04$	$1.44E+04$	$3.74E+04$
	10	$2.72E+00$	$2.62E+00$	$4.92E+00$	$3.08E+00$	$2.69E+00$	$\mathbf{2.60E+00}$

Samp.	Func.	ANN	GP	MLS	OLS	RBF	SVR
64	11	$1.21E+01$	$1.10E+01$	$\mathbf{1.05E+01}$	$1.09E+01$	$1.11E+01$	$1.17E+01$
	12	$3.02E-01$	$\mathbf{1.28E-01}$	$2.40E-01$	$2.86E-01$	$2.33E-01$	$3.05E-01$
	13	$\mathbf{1.24E-01}$	$8.48E-01$	$2.84E+00$	$5.18E+00$	$2.16E-01$	$2.93E+00$
	14	$5.76E+01$	$\mathbf{6.32E-01}$	$2.15E+02$	$2.34E+02$	$5.60E+01$	$1.56E+02$
	15	$\mathbf{4.27E-02}$	$4.70E-02$	$1.46E-01$	$3.14E-01$	$3.21E-01$	$4.28E-02$
	1	$7.58E-01$	$\mathbf{6.58E-01}$	$7.17E-01$	$1.04E+00$	$6.73E-01$	$6.98E-01$
	2	$\mathbf{2.62E+02}$	$5.67E+03$	$6.29E+03$	$1.48E+04$	$5.48E+02$	$8.38E+03$
	3	$2.28E-04$	$\mathbf{1.17E-13}$	$4.95E-13$	$3.94E-12$	$3.05E-04$	$1.42E+01$
	4	$9.51E+00$	$8.18E+00$	$1.02E+01$	$1.54E+01$	$\mathbf{7.98E+00}$	$1.03E+01$
	5	$1.54E-01$	$\mathbf{8.48E-02}$	$2.02E-01$	$2.06E-01$	$1.48E-01$	$1.98E-01$
	6	$\mathbf{2.21E-02}$	$4.67E-02$	$3.24E-02$	$1.50E-01$	$2.25E-02$	$2.45E-02$
	7	$9.80E-03$	$3.34E-02$	$1.61E-02$	$1.39E-01$	$\mathbf{5.00E-03}$	$5.82E-03$
	8	$2.87E+02$	$2.80E+02$	$3.38E+02$	$3.45E+02$	$\mathbf{2.61E+02}$	$2.98E+02$
	9	$1.11E+03$	$5.79E+03$	$9.95E+03$	$4.83E+04$	$\mathbf{1.02E+03}$	$1.40E+04$
	10	$2.55E+00$	$\mathbf{2.12E+00}$	$2.88E+00$	$2.65E+00$	$2.59E+00$	$2.54E+00$
	11	$9.36E+00$	$\mathbf{8.47E+00}$	$9.86E+00$	$9.45E+00$	$1.05E+01$	$1.10E+01$
	12	$1.68E-01$	$\mathbf{4.15E-02}$	$1.84E-01$	$2.78E-01$	$1.61E-01$	$2.12E-01$
	13	$\mathbf{2.79E-03}$	$3.17E-01$	$1.40E+00$	$5.07E+00$	$1.66E-02$	$1.50E+00$
	14	$9.54E-01$	$\mathbf{4.58E-02}$	$8.06E+01$	$2.28E+02$	$1.24E+00$	$1.01E+02$
	15	$\mathbf{2.01E-02}$	$5.31E-02$	$5.41E-02$	$3.13E-01$	$2.14E-01$	$2.97E-02$
96	1	$7.55E-01$	$\mathbf{4.83E-01}$	$6.80E-01$	$1.03E+00$	$7.13E-01$	$6.78E-01$
	2	$\mathbf{9.80E+01}$	$1.52E+03$	$3.32E+03$	$1.43E+04$	$1.48E+02$	$4.69E+03$
	3	$1.22E-04$	$\mathbf{1.37E-13}$	$5.02E-13$	$5.05E-12$	$2.96E-04$	$8.21E+00$
	4	$8.52E+00$	$6.20E+00$	$1.20E+01$	$1.51E+01$	$\mathbf{5.18E+00}$	$9.10E+00$
	5	$1.39E-01$	$\mathbf{6.85E-02}$	$1.98E-01$	$1.99E-01$	$1.33E-01$	$1.49E-01$
	6	$1.57E-02$	$3.66E-02$	$2.42E-02$	$1.50E-01$	$\mathbf{1.17E-02}$	$1.44E-02$
	7	$\mathbf{6.07E-04}$	$1.22E-02$	$1.37E-02$	$1.39E-01$	$1.19E-03$	$1.91E-03$
	8	$2.99E+02$	$2.70E+02$	$2.90E+02$	$3.12E+02$	$\mathbf{2.53E+02}$	$2.99E+02$
	9	$\mathbf{1.43E+02}$	$1.11E+03$	$6.17E+03$	$4.83E+04$	$2.12E+02$	$1.01E+04$
	10	$2.36E+00$	$\mathbf{1.85E+00}$	$2.48E+00$	$2.53E+00$	$2.16E+00$	$2.46E+00$
	11	$9.27E+00$	$\mathbf{9.24E+00}$	$9.40E+00$	$9.78E+00$	$1.08E+01$	$9.50E+00$
	12	$1.36E-01$	$\mathbf{2.45E-02}$	$2.47E-01$	$2.78E-01$	$1.33E-01$	$1.37E-01$
	13	$\mathbf{2.30E-03}$	$1.93E-01$	$6.40E-01$	$4.97E+00$	$2.84E-03$	$3.92E-01$
	14	$6.18E-01$	$\mathbf{2.88E-03}$	$4.84E+01$	$2.34E+02$	$8.48E-01$	$8.11E+01$
	15	$1.47E-02$	$3.35E-02$	$1.87E-02$	$3.12E-01$	$2.15E-01$	$\mathbf{1.06E-02}$
128	1	$6.40E-01$	$\mathbf{3.63E-01}$	$6.54E-01$	$1.02E+00$	$6.88E-01$	$6.62E-01$
	2	$4.63E+01$	$7.29E+02$	$2.52E+03$	$1.43E+04$	$\mathbf{2.56E+01}$	$2.61E+03$
	3	$1.14E-04$	$\mathbf{1.18E-13}$	$5.34E-13$	$4.34E-12$	$2.14E-04$	$2.55E+00$
	4	$7.79E+00$	$6.03E+00$	$7.07E+00$	$1.51E+01$	$\mathbf{4.70E+00}$	$5.98E+00$
	5	$1.36E-01$	$\mathbf{6.34E-02}$	$1.51E-01$	$2.00E-01$	$1.32E-01$	$1.22E-01$
	6	$6.89E-03$	$2.29E-02$	$2.63E-02$	$1.51E-01$	$\mathbf{5.82E-03}$	$7.95E-03$
	7	$\mathbf{4.98E-05}$	$8.45E-03$	$1.43E-02$	$1.40E-01$	$2.04E-04$	$1.75E-03$
	8	$2.98E+02$	$2.45E+02$	$2.58E+02$	$3.06E+02$	$\mathbf{2.30E+02}$	$3.00E+02$
	9	$\mathbf{8.83E+01}$	$3.73E+02$	$9.43E+03$	$4.96E+04$	$9.26E+01$	$6.03E+03$
	10	$2.40E+00$	$\mathbf{1.78E+00}$	$2.18E+00$	$2.36E+00$	$1.98E+00$	$2.48E+00$
	11	$1.05E+01$	$9.79E+00$	$\mathbf{9.54E+00}$	$1.00E+01$	$1.15E+01$	$1.08E+01$
	12	$1.40E-01$	$\mathbf{1.40E-02}$	$7.25E-01$	$2.77E-01$	$1.12E-01$	$1.16E-01$
	13	$1.26E-03$	$1.33E-02$	$4.52E-01$	$4.98E+00$	$\mathbf{9.79E-04}$	$5.01E-01$
	14	$3.18E-01$	$\mathbf{2.41E-03}$	$4.88E+01$	$2.30E+02$	$3.16E-01$	$3.53E+01$
	15	$1.27E-02$	$2.57E-02$	$1.83E-02$	$3.11E-01$	$2.22E-01$	$\mathbf{1.25E-02}$

Table 1: Root mean squared error of prediction at 16384 validation points for various number of sample points

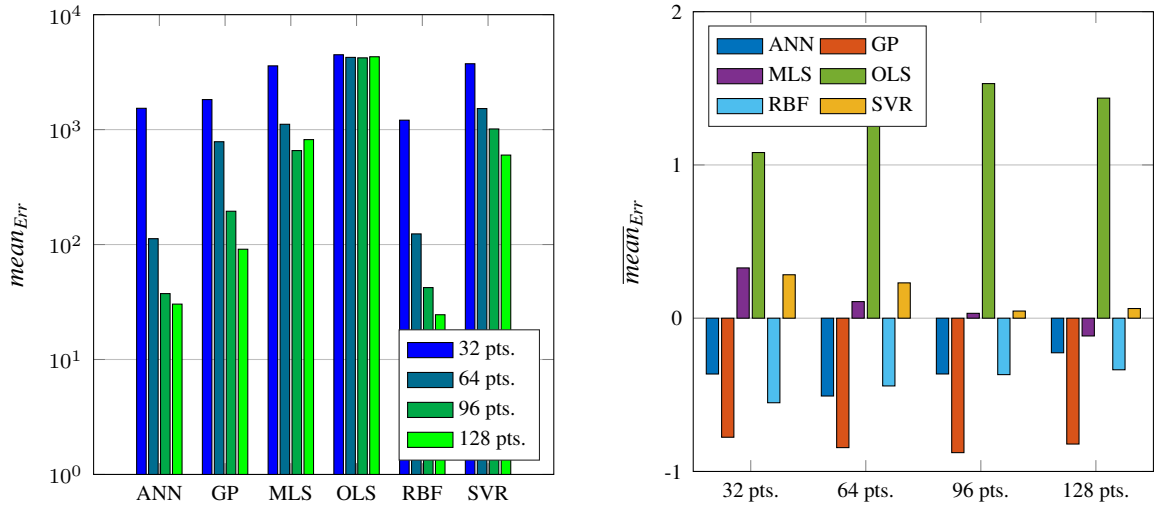


Figure 2: $mean_{Err}$ and $\overline{mean}_{Err}$ of the RMSE values for varied number of sample points

Table 1 shows the RMSE of 16384 validation points for a varying number of sample points. Smaller values of error are better and bold values represent the best result for each function. It can can be seen that the RMSE generally decreases with the increasing number of sample points. There are contradictions to this observation (i.e. Function 3), where the prediction error is already quite low for few samples and becomes larger because of oversampling.

Furthermore, it can be seen that different algorithms have the best score for different functions. Figure 2 shows that Kriging (GP) outperforms other algorithms on average, followed by ANN and RBF. OLS seems to have the poorest results in this part by a great margin.

Noise	Func.	ANN	GP	MLS	OLS	RBF	SVR
	1	$6.48E-01$	$\mathbf{3.98E-01}$	$6.63E-01$	$7.06E-01$	$6.98E-01$	$6.61E-01$
	2	$\mathbf{2.31E+02}$	$7.22E+02$	$2.56E+03$	$5.90E+03$	$2.79E+02$	$2.60E+03$
	3	$3.28E+00$	$\mathbf{2.72E+00}$	$2.86E+00$	$3.28E+00$	$3.26E+00$	$4.78E+00$
	4	$7.90E+00$	$6.05E+00$	$7.64E+00$	$1.49E+01$	$\mathbf{4.84E+00}$	$6.11E+00$
	5	$1.35E-01$	$\mathbf{6.56E-02}$	$1.53E-01$	$1.89E-01$	$1.33E-01$	$1.23E-01$
	6	$6.54E-03$	$7.19E-03$	$2.67E-02$	$1.19E-01$	$\mathbf{5.65E-03}$	$9.03E-03$
	7	$4.42E-03$	$5.92E-03$	$1.40E-02$	$1.14E-01$	$3.26E-03$	$\mathbf{1.85E-03}$
1 %	8	$2.98E+02$	$2.36E+02$	$2.58E+02$	$3.06E+02$	$\mathbf{2.30E+02}$	$3.00E+02$
	9	$\mathbf{2.17E+03}$	$7.25E+03$	$8.99E+03$	$3.36E+04$	$3.58E+03$	$5.69E+03$
	10	$2.26E+00$	$\mathbf{1.62E+00}$	$2.33E+00$	$2.26E+00$	$1.98E+00$	$2.48E+00$
	11	$1.05E+01$	$9.86E+00$	$\mathbf{9.54E+00}$	$1.02E+01$	$1.15E+01$	$1.09E+01$
	12	$1.25E-01$	$\mathbf{1.50E-02}$	$7.20E-01$	$2.68E-01$	$1.12E-01$	$1.17E-01$
	13	$\mathbf{1.21E-01}$	$2.42E-01$	$5.17E-01$	$4.57E-01$	$1.48E-01$	$6.32E-01$
	14	$5.12E+00$	$\mathbf{4.91E+00}$	$2.60E+02$	$5.21E+01$	$6.21E+00$	$5.82E+01$
	15	$4.08E-02$	$\mathbf{1.11E-02}$	$1.82E-02$	$1.31E-01$	$2.28E-01$	$1.18E-02$
	1	$7.21E-01$	$\mathbf{4.57E-01}$	$6.85E-01$	$7.18E-01$	$7.25E-01$	$6.63E-01$
	2	$\mathbf{5.21E+02}$	$1.35E+03$	$2.62E+03$	$5.90E+03$	$7.08E+02$	$2.72E+03$
	3	$9.06E+00$	$\mathbf{6.81E+00}$	$7.16E+00$	$8.19E+00$	$8.68E+00$	$1.20E+01$
	4	$9.04E+00$	$6.61E+00$	$8.58E+00$	$1.50E+01$	$\mathbf{5.50E+00}$	$6.31E+00$
	5	$1.40E-01$	$\mathbf{7.51E-02}$	$1.58E-01$	$1.90E-01$	$1.38E-01$	$1.28E-01$
	6	$7.25E-03$	$8.38E-03$	$2.75E-02$	$1.19E-01$	$\mathbf{6.69E-03}$	$1.16E-02$
	7	$9.11E-03$	$1.13E-02$	$1.36E-02$	$1.14E-01$	$6.10E-03$	$\mathbf{3.99E-03}$
2.5 %	8	$2.98E+02$	$2.36E+02$	$2.58E+02$	$3.06E+02$	$\mathbf{2.30E+02}$	$2.81E+02$
	9	$\mathbf{5.97E+03}$	$1.21E+04$	$8.58E+03$	$3.34E+04$	$6.07E+03$	$9.89E+03$
	10	$2.26E+00$	$\mathbf{1.63E+00}$	$2.33E+00$	$2.26E+00$	$1.98E+00$	$2.47E+00$

Noise	Func.	ANN	GP	MLS	OLS	RBF	SVR
	11	$1.02E+01$	$9.66E+00$	$\mathbf{9.54E+00}$	$1.02E+01$	$1.14E+01$	$1.01E+01$
	12	$1.25E-01$	$\mathbf{1.50E-02}$	$7.12E-01$	$2.68E-01$	$1.12E-01$	$1.18E-01$
	13	$\mathbf{2.95E-01}$	$3.64E-01$	$6.65E-01$	$5.14E-01$	$3.74E-01$	$7.50E-01$
	14	$\mathbf{1.10E+01}$	$1.13E+01$	$2.61E+02$	$5.23E+01$	$1.28E+01$	$3.58E+01$
	15	$4.05E-02$	$1.96E-02$	$1.85E-02$	$1.31E-01$	$2.42E-01$	$\mathbf{1.46E-02}$
	1	$7.43E-01$	$7.13E-01$	$7.26E-01$	$7.47E-01$	$7.99E-01$	$\mathbf{6.69E-01}$
	2	$\mathbf{1.02E+03}$	$2.13E+03$	$2.77E+03$	$5.91E+03$	$1.15E+03$	$2.60E+03$
	3	$1.56E+01$	$\mathbf{1.36E+01}$	$1.43E+01$	$1.64E+01$	$1.65E+01$	$1.49E+01$
	4	$9.08E+00$	$7.77E+00$	$1.03E+01$	$1.51E+01$	$\mathbf{7.06E+00}$	$8.36E+00$
	5	$1.48E-01$	$\mathbf{1.01E-01}$	$1.73E-01$	$1.93E-01$	$1.48E-01$	$1.45E-01$
	6	$1.13E-02$	$1.21E-02$	$2.97E-02$	$1.19E-01$	$\mathbf{1.03E-02}$	$1.93E-02$
	7	$1.48E-02$	$1.81E-02$	$1.34E-02$	$1.14E-01$	$1.04E-02$	$\mathbf{6.76E-03}$
5 %	8	$2.99E+02$	$2.36E+02$	$2.59E+02$	$3.07E+02$	$\mathbf{2.30E+02}$	$2.99E+02$
	9	$\mathbf{6.52E+03}$	$1.23E+04$	$1.48E+04$	$3.31E+04$	$7.75E+03$	$1.06E+04$
	10	$2.26E+00$	$\mathbf{1.63E+00}$	$2.18E+00$	$2.26E+00$	$1.98E+00$	$2.46E+00$
	11	$1.02E+01$	$\mathbf{6.76E+00}$	$9.54E+00$	$1.02E+01$	$1.15E+01$	$1.04E+01$
	12	$1.33E-01$	$\mathbf{2.02E-02}$	$6.99E-01$	$2.69E-01$	$1.12E-01$	$1.19E-01$
	13	$\mathbf{5.94E-01}$	$6.27E-01$	$9.63E-01$	$6.52E-01$	$7.51E-01$	$1.10E+00$
	14	$\mathbf{2.10E+01}$	$2.23E+01$	$6.25E+01$	$5.35E+01$	$2.74E+01$	$4.52E+01$
	15	$3.24E-02$	$2.99E-02$	$2.07E-02$	$1.30E-01$	$2.20E-01$	$\mathbf{1.93E-02}$
	1	$8.05E-01$	$9.19E-01$	$1.03E+00$	$8.38E-01$	$1.02E+00$	$\mathbf{6.98E-01}$
	2	$\mathbf{1.61E+03}$	$3.52E+03$	$3.14E+03$	$5.95E+03$	$3.29E+03$	$2.78E+03$
	3	$3.71E+01$	$2.72E+01$	$2.86E+01$	$3.28E+01$	$3.54E+01$	$\mathbf{2.31E+01}$
	4	$1.13E+01$	$1.11E+01$	$1.18E+01$	$1.56E+01$	$1.07E+01$	$\mathbf{1.03E+01}$
	5	$1.69E-01$	$\mathbf{1.65E-01}$	$2.29E-01$	$2.02E-01$	$1.83E-01$	$1.66E-01$
	6	$1.99E-02$	$2.09E-02$	$3.43E-02$	$1.19E-01$	$\mathbf{1.94E-02}$	$3.22E-02$
	7	$2.69E-02$	$2.53E-02$	$1.45E-02$	$1.14E-01$	$1.67E-02$	$\mathbf{1.21E-02}$
10 %	8	$2.98E+02$	$2.37E+02$	$2.61E+02$	$3.07E+02$	$\mathbf{2.31E+02}$	$2.94E+02$
	9	$\mathbf{9.79E+03}$	$1.61E+04$	$1.61E+04$	$3.28E+04$	$1.60E+04$	$1.63E+04$
	10	$2.27E+00$	$\mathbf{1.66E+00}$	$2.20E+00$	$2.27E+00$	$2.00E+00$	$2.45E+00$
	11	$1.07E+01$	$\mathbf{7.42E+00}$	$9.57E+00$	$1.03E+01$	$1.04E+01$	$1.13E+01$
	12	$1.28E-01$	$\mathbf{3.05E-02}$	$6.73E-01$	$2.69E-01$	$1.15E-01$	$1.22E-01$
	13	$\mathbf{1.00E+00}$	$1.21E+00$	$1.01E+00$	$1.47E+00$	$1.02E+00$	$1.80E+00$
	14	$\mathbf{4.12E+01}$	$4.44E+01$	$2.64E+02$	$5.90E+01$	$5.82E+01$	$5.71E+01$
	15	$5.80E-02$	$3.87E-02$	$2.73E-02$	$1.30E-01$	$2.15E-01$	$\mathbf{2.64E-02}$

Table 2: Root mean squared error of prediction at 16384 validation points for 128 sample points and varied levels of Gaussian noise

Results for varying noise levels are displayed in Table 2. Here it can be seen that increasing noise levels increase the prediction error and some algorithms are effected more than the others. Interestingly, the only exception is SVR; data with 1% noise seems to train a better metamodel than data with no noise for some test functions and on average.This can be explained by the fact that the results of SVR without noise were not as good on average. Furthermore SVR is the only tested algorithm with free parameters to define process error (ϵ and θ^P in Eq. 8).

As before, Kriging seems to perform better than other algorithms followed by ANN and RBF on average (Figure 3) . SVR becomes comparably good for higher levels of noise. Kriging, MLS, OLS and SVR also seem to be least affected from increasing levels of noise, since the relative increase in mean value is considerably smaller than other methods. OLS seems to consistently achieve the poorest results, although the difference gets smaller with increasing levels of noise.

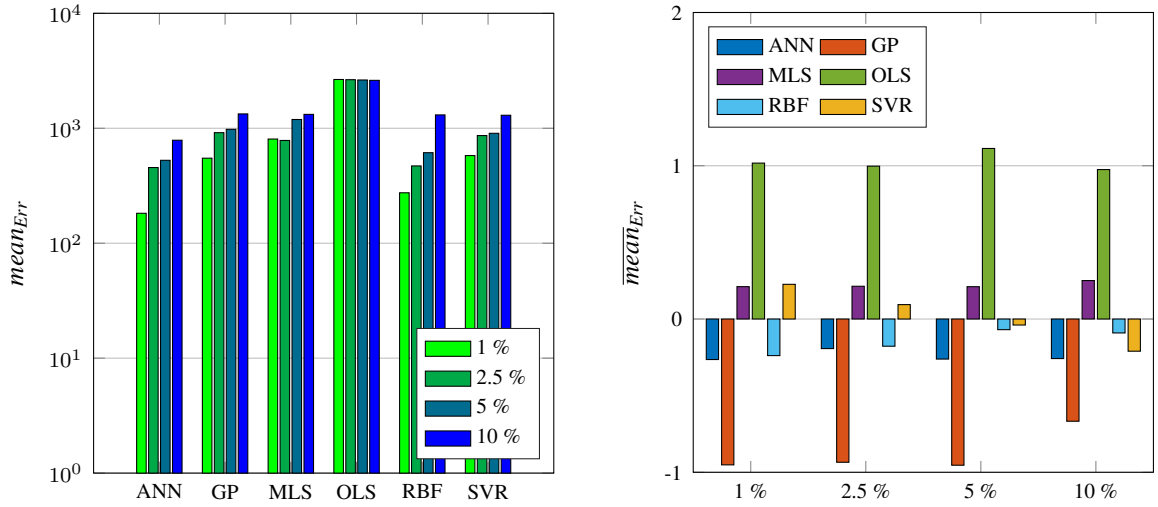


Figure 3: $mean_{Err}$ and $\overline{mean}_{Err}$ of the RMSE values for varied levels of Gaussian Noise

4 CONCLUSION

Six metamodeling algorithms are tested with 15 analytical functions in 8 different cases. A total of 720 results are included in this benchmark. The following observations can be made based on these results:

- Prediction error has a negative correlation with the number of sample points and a positive correlation with the noise level. On the other hand, it is observed that oversampling may worsen the quality of some metamodels. Furthermore, a small amount of Gaussian noise (1%) has a positive impact on the model quality of support vector regression for some test functions.

- On average, Kriging is the most accurate method followed by radial basis functions and neural networks independent of varying number of sample points and noise level. Conversely, every method except ordinary least squares delivers the best results for at least one of test functions. This shows that the axiom ”No free lunch theorem” [23] holds for metamodeling too.

- Kriging, moving least squares and support vector regression handle the noise best since relative increase of the prediction error is the smallest for these methods.

- Cross-validation helps avoid overfitting even for pure interpolating algorithms like Kriging and radial basis functions. This can be seen in the results for noisy data (Table 2).

Furthermore, the following observations were made during the benchmark:

- Standardizing data often but not always decreases the error.

- Finding the global optimum for anisotropic Kriging may sometimes be challenging for the optimizer. Therefore isotropic Kriging has better results for some functions.

- Gaussian radial basis function is often but not always the best of all tested radial basis functions.

- Radial basis function is the best of all tested kernels for support vector regression considering this benchmark.
- A better designed neural network for a specific problem may yield better results for that problem than a generalized network like in this benchmark.

We conclude that Kriging, neural networks and radial basis functions in this order are, on average, the best choices for unknown problems, at least when the constraints in this benchmark regarding the number of sample points and noise levels are fulfilled and the structural complexity is similar to those of the chosen test functions. Moreover, testing different algorithms may yield better results in some cases. Further work should be done regarding the effects of dimensionality on the quality of metamodels.

5 ACKNOWLEDGEMENT

The authors would like to express their thanks to the German Federal Ministry of Education and Research for their support for this project. Support identification no.: IN2013-671-221.

REFERENCES

[1] Krishnamurty, T., Comparison of Response Surface Construction Methods for Derivative Estimation Using Moving Least Squares, Kriging and Radial Basis Functions, *46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference*, 2005

[2] Li, Y.F.; Ng, S.H.; Xie, M. and Goh, T.N., A systematic comparison of metamodeling techniques for simulation optimization in Decision Support Systems, *Applied Soft Computing*, vol.10, 1257 - 1273, 2010

[3] Luo, J. and Lu, W., Comparison of surrogate models with different methods in groundwater remediation process, *Journal of Earth System Science*, vol.123, 1579 - 1589, 2014

[4] Rao, C. R. and Toutenburg, H., Linear Models: Least Squares and Alternatives, Second Edition, *Springer Verlag*, 1999

[5] Levin, D., The Approximation Power of Moving Least Squares, *Mathematics of Computation*, vol. 67, no. 224, 1998

[6] Most, T. and Bucher, C., A Moving Least Squares weighting function for the Element-free Galerkin Method which almost fulfills essential boundary conditions, *Structural Engineering and Mechanics*, vol. 21, no. 3, 2005

[7] Krishnamurty, T. and Romero, V. J., Construction of Response Surface with Higher Order Continuity and its Application to Reliability Engineering, *43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2002

[8] Dinh, H. Q.; Turk, G. and Slabaugh, G., Reconstructing surfaces using anisotropic basis functions, *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol.2, 2001

[9] Smola, A. J. and Schlkopf,B., A Tutorial on Support Vector Regression, *Statistics and Computing*,2003

[10] Rojas ,R., Neural Networks - A Systematic Introduction, *Springer Verlag*, 1996

[11] Wasserman, P.D., Advanced Methods in Neural Computing, *Van Nostrand Reinhold*, 1993

[12] Joseph, R. V. and Hung. Y., Orthogonal-Maxmin Latin Hypercube Designs, *Statistica Sinica*, vol. 18, 2008

[13] Ackley, D. H., A Connectionist Machine for Genetic Hill-Climbing, *Kluwer Academic Publishers*, 1987

[14] Jamil, M. and Yang, X.S. A Literature Survey of Benchmark Functions For Global Optimization Problems, *Blekinge Institute of Technology*, 2013

[15] Bukin, A. D. , New Minimization Strategy For Non-Smooth Functions, *Budker Institute of Nuclear Physics*, 1997

[16] Mishra, S. K., Some New Test Functions for Global Optimization and Performance of Repulsive Particle Swarm Method, *Department of Economics, North-Eastern Hill University*, 2006

[17] Easom, E. E., A Survey of Global Optimization Techniques, *University of Louisville*, 1990

[18] Goldstein, A. A. and Price, I. F., On Descent From Local Minima, *Mathematics of Computation*, vol. 25, no. 115, 1971

[19] Michalewicz, Z., Genetich Algorithms + Data Structures = Evolution Programs, *Springer-Verlag*, 1992

[20] Dixon, L. C. and Szego, G. P., Towards Global Optimisation 2, *Elsevier Science Ltd. North Holland*, 1978

[21] Ursem, R., Multinational Evolutionary Algorithms, *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol.3, 1999

[22] Dalal, I.L.; Stefan, D. and Harwayne-Gidansky, J., Low Discrepancy Sequences for Monte Carlo Simulations on Reconfigurable Platforms, *Application-Specific Systems, Architectures and Processors, 2008. ASAP*, 2008

[23] Wolpert, D.H. and Macready, W.G., No Free Lunch Theorems for Optimization, *IEEE Transactions on Evolutionary Computation* vol. 1, 1997