The fanovaGraph Package: Visualization of Interaction Structures and Construction of Block-additive Kriging Models

Jana Fruth, Olivier Roustant, Thomas Muehlenstaedt

The `fanovaGraph` Package: Visualization of Interaction Structures and Construction of Block-additive Kriging Models

J. Fruth[a], O. Roustant[b], T. Muehlenstaedt[c]

[a] *Faculty of Statistics, TU Dortmund University, Vogelpothsweg 87, 44221 Dortmund, Germany*
[b] *Ecole Nationale Supérieure des Mines, FAYOL-EMSE, LSTI, F-42023 Saint-Etienne, France*
[c] *W. L. Gore & Associates GmbH, 85640 Putzbrunn, Germany*

Abstract

FANOVA graphs have a broad range of application in model interpretation, sensitivity analysis and analysis of computer experiments. They give a clear and easy to understand visualization of function structures, reveal block-additive decompositions and also effectively improve Kriging model predictions by kernel adaption. These several fields are combined in a new released R package `fanovaGraph`. This tutorial presents the implemented methods step by step together with theoretical background and illustrations. Additionally it includes methods in simulation and the crucial question of threshold decision, where graphical solutions are suggested.

Keywords: Sensitivity analysis, FANOVA decomposition, Sobol indices, High-order interactions, Superset importance, Additive structure

1. Introduction

Many computer experiments behave like black box functions in the sense that relations between input and output factors are not obvious from the function. If the experiment is expensive, usually a meta model like a Kriging emulator (but also other models like Neural Networks or Random Forests) is

Email addresses: `fruth@statistik.tu-dortmund.de` (J. Fruth), `roustant@emse.fr` (O. Roustant), `tmuehlen@wlgore.com` (T. Muehlenstaedt)

estimated from a low number of runs as a replacement for the experiment. Then again, the meta model provides a function which needs to be explored. To analyse and interpret those black box functions, common approaches are scatterplots or sensitivity analysis methods, which become problematic in the presence of interactions (see for instance [11] for a review). This problem has recently been approached by using graphical displays of the interaction structure (see [5] and [7]).

The R package `fanovaGraph` [3] is an efficient tool for a clear and easy to understand graphical visualization of the global interaction structure of a black box function. Interactions are depicted by a so-called FANOVA graph, whose edge thicknesses indicate the interaction strength. Estimators for these interactions, based on the superset importance estimation in [6], have been shown to be asymptotical normal and efficient in [4].

Following the procedure suggested in [7], the detected interaction structure (block-additive decomposition) by the FANOVA graph can further be used to construct block-additive Kriging models. The package `fanovaGraph` therefore contains in a second part all methods for block-additive Kriging analysis based on the package `DiceKriging` [10]. This includes estimation, prediction and simulation of those models.

The paper is organised as follows. At first the main procedure is demonstrated at an example function allowing an easy access and short overview. Section 3 then gives detailed information about the procedure's steps and explains the statistical background. New confidence intervals for the estimators are presented. In Section 4, the question of choosing the right threshold value for the graph edges is adressed and a graphical decision plot for the threshold is suggested. Finally Section 5 shows how further models, here Neural Networks and Support Vector Machines can be analyzed by the package.

Throughout this paper, we use the following six dimensional function as a toy example for the "unknown" black box function.

$$f(\mathbf{x}) = \cos((1, x_1, x_3, x_5)\beta^T) + \sin((1, x_2, x_4, x_6)\gamma^T)$$

with $\beta = (-0.8, -1.1, 1, 1.1)$ and $\gamma = (-0.5, 1, 0.9, -1.1)$ and $\mathbf{x} \in [-1, 1]^6$. The function has the block-additive form

$$f(\mathbf{x}) = g(x_1, x_3, x_5) + h(x_2, x_4, x_6).$$

```
> d <- 6
> domain <- c(-1, 1)
> fun <- function(x) {
+      beta <- c(-0.8, -1.1, 1, 1.1)
+      gamma <- c(-0.5, 1, 0.9, -1.1)
+      g <- cos(cbind(1, x[, c(1, 3, 5)]) %*% beta)
+      h <- sin(cbind(1, x[, c(2, 4, 6)]) %*% gamma)
+      return(g + h)
+  }
```

2. A demo

Before detailing the functions of `fanovaGraph`, it is worth giving an overview of their usage in a practical case study. We presume basic knowledge about Kriging and sensitivity analysis and refer to [11], [2].

There are two main goals of `fanovaGraph`:

1. To provide a description of the interaction structure of a black box function, say f, visualized in a so-called *FANOVA graph*, which at the same time gives a block-additive decomposition of the function.
2. To use the block-additive decomposition given by the FANOVA graph to build an advanced Kriging type metamodel f_{app}.

Of course the steps 1 and 2 can be done independently: When only the interaction structure of the function f is of interest (step 1) or when external information is available to construct a block-additive kernel (step 2). This section shows the frequent situation where both steps 1 and 2 are done sequentially: the modeling of a costly black box function, e.g. a finite element computer experiment. This situation is presented carefully in `fanovaGraph` by the way of two *demos*. We detail it here by means of the 6D example presented in Section 1.

When dealing with a costly function, even the first step requires a metamodel of f. Such functional approximation can be built in different ways, by using linear models, neural networks, polynomial chaos expansion, Kriging models (or equivalently Gaussian process regression), splines, etc. In the following, the metamodel is based on a Kriging model built from the package `DiceKriging` based on a maximin design sampled with the package `lhs` [1], but other solutions might have been chosen as well. The code corresponding to the initialization step is shown below.

```
> library(lhs)
> L01 <- maximinLHS(100, d)
> x <- L01 * (domain[2] - domain[1]) + domain[1]
> y <- fun(x)
> KM <- km(~1, design = data.frame(x), response = y)
```

Obviously, the accuracy of the first metamodel should be verified before continuing further. This step is not shown here, for the sake of brevity. The model's prediction function now serves as a metamodel. It is in `fanovaGraph` implemented by the `kmPredictWrapper` function, based on `predict` from `DiceKriging`. The model (here `KM`) must be given as an extra argument.

The usual first step of a sensitivity analysis, estimation of first order Sobol indices, can then be done with the package `sensitivity` [8]. The result, visualized in Figure 1, shows that all 6 variables are influent, and indicates the presence of interactions.

```
> i1 <- fast99(model = kmPredictWrapper, factors = d,
+      n = 3000, q = "qunif", q.arg = list(min = domain[1],
+      max = domain[2]), km.object = KM)
> plot(i1)
```

However, this analysis does not show the way how variables interact and the functions two additive blocks are not revealed. To do so, we estimate a FANOVA graph. This is done in `fanovaGraph` by the `estimateGraph` function. The obtained graph object `g` can then be plotted:

```
> g <- estimateGraph(f.mat = kmPredictWrapper, d = d, n.tot = 30000,
+       q.arg = list(min = domain[1], max = domain[2]), km.object = KM)
> plot(g, plot.i1=FALSE)
```

The corresponding figure can be seen in Figure 2, on the left. Clearly, two groups of variables are visible, and can be identified by choosing an appropriate threshold. How to choose the threshold value is detailed in Section 4. Here $\delta = 0.01$ seems to be a suitable value, as shown on the same figure, on the right.

```
> g.cut <- threshold(g, delta = 0.01, scale = TRUE)
> plot(g.cut, plot.i1=FALSE)
```

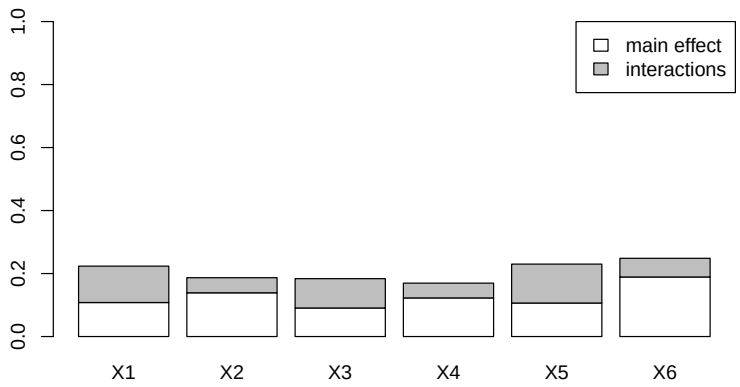
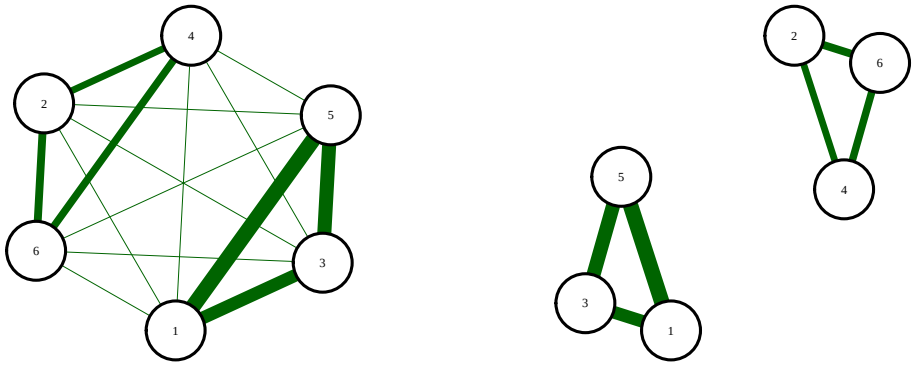


Figure 1: Main effects (white box) and total effects (large box) for the 6D-function f, obtained with the package `sensitivity` ([8]).



(a) Without thresholding

(b) With threshold $\delta = 0.01$

Figure 2: The estimated FANOVA graph for the 6D function f.

We can use the information for the second part to build up a block-additive Kriging model. Mathematically, the two unconnected (additive)

parts are the two cliques of the graph g

```
> Cliques <- g.cut$cliques
> print(Cliques)
[[1]]
[1] 1 3 5

[[2]]
[1] 2 4 6
```

The clique structure suggests that a good kernel for this function, among kernels of stationary processes, has the block-additive structure:

$$k(\mathbf{h}) = k_{135}(\mathbf{h}_{135}) + k_{246}(\mathbf{h}_{246})$$

with the set notation $\mathbf{h}_I = \{h_i, i \in I\}$ for all set I, and where k_{135}, k_{246} are kernels defined over 3-dimensional subspaces. Several kernel types are available, and we choose for both the Gaussian tensor-product kernel, assuming that f is a smooth function. The estimation of the corresponding Kriging model can now be done with the `MLoptimConstrained` function of `fanovaGraph`, and predictions computed with `yhat`. Finally, the results are compared to the 'standard' Kriging model, based on a 6D Gaussian tensor-product kernel, and confirm the clear improvement in prediction of the new Kriging model (Figure 3). Note that at no point we made any priori assumptions about the block-additive form of the black-box function.

```
> parameter <- kmAdditive(x, y, cl = Cliques)
> xpred <- matrix(runif(d * 1000, domain[1], domain[2]), ncol = d)
> y_new <- predictAdditive(xpred, x, y, parameter, cl = Cliques)
> y_old <- kmPredictWrapper(xpred, km.object = KM)
> y_exact <- fun(xpred)


> par(mfrow = c(1, 2))
> plot(y_exact, y_old, asp = 1, xlab="y, exact",
+       ylab="y, predicted", main="Standard Kernel")
> abline(0, 1)
> plot(y_exact, y_new[, 1], asp = 1, xlab="y, exact",
+       ylab="y, predicted", main="Modified Kernel")
> abline(0, 1)
```

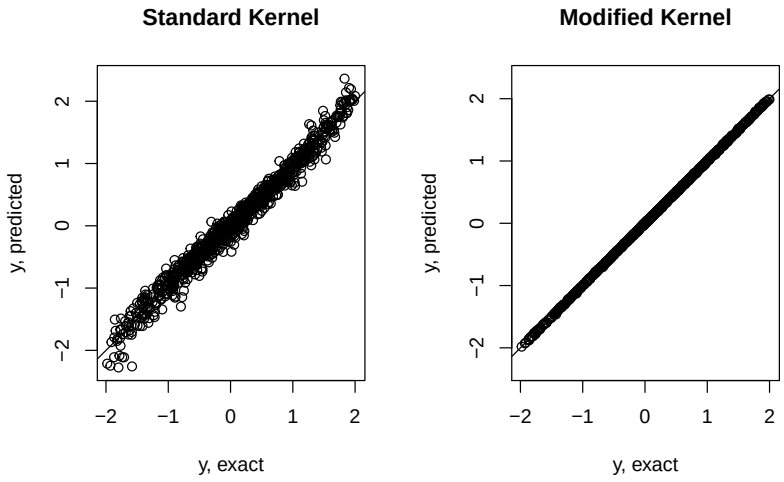


Figure 3: Predictions over a 1000-point uniform test set of two Kriging models: A standard one, based on a 6D Gaussian tensor-product kernel, and a new one, based on a sum of two 3D Gaussian tensor-product kernels $k_{135} + k_{246}$.

3. Detailed description of functions

Let us now generally introduce the procedure and its implementation in `fanovaGraph`. Say that the function under investigation f is defined over d factors $x_1, \ldots, x_d$, which can be described as independent random variables with probability measure ν. Usually, if no special distribution is predetermined, a uniform distribution in the factor domain is chosen for ν.

3.1. Estimation of indices

The key point in building the FANOVA graph is the estimation of the edge thicknesses. An edge shall represent the interaction strength of the two factors that are represented by the vertices it combines. Therefore a Sobol-based index is chosen, that measures the overall influence of the two factors, called total interaction index (TII) in [4]. For two factors x_i and x_j it is defined by the sum of all Sobol indices that contain both indices:

$$\mathfrak{D}_{ij} := \sum_{I \supseteq \{i,j\}} D_I, \tag{1}$$

where D_I is the unscaled Sobol index of the group of factors $(X_k)_{k\in I}$. Liu and Owen (2006) show, that this index can be computationally obtained by

$$\begin{aligned}\mathfrak{D}_{ij} &= \frac{1}{4}E\left[\Delta_{i,j}(z_{i,j}, x_{i,j}, x_{-\{i,j\}})^2\right] \\ &= \frac{1}{4}\int \left[\Delta_{i,j}(z_{i,j}, x_{i,j}, x_{-\{i,j\}})\right]^2 d\nu_{i,j}(z_{i,j})d\nu(\mathbf{x}) \qquad (2)\end{aligned}$$

with $\Delta_{i,j}(z_{i,j}, x_{i,j}, x_{-\{i,j\}})) =$

$$f(x_i, x_j, x_{-\{i,j\}}) - f(x_i, z_j, x_{-\{i,j\}}) - f(z_i, x_j, x_{-\{i,j\}}) + f(z_i, z_j, x_{-\{i,j\}})$$

and where z_i (resp. z_j) is an independent copy of x_i (resp. x_j).
Since f is assumed to be a black box function, the total interaction index can not easily be computed by (2). One way to estimate it is by partly fixed Monte Carlo integration of the integrals. Denote by $\mathbf{x}^k$ and $\mathbf{z}^k$, $k = 1, \ldots, N$ two independent samples of (sufficiently large) size N drawn from ν. Then the total interaction index is estimated by

$$\begin{aligned}\widehat{\mathfrak{D}}_{ij} = \frac{1}{4} \times \frac{1}{N}\sum_{k=1}^{N} \big[f(x_i^k, x_j^k, x_{-\{i,j\}}^k) - f(x_i^k, z_j^k, x_{-\{i,j\}}^k) \\ - f(z_i^k, x_j^k, x_{-\{i,j\}}^k) + f(z_i^k, z_j^k, x_{-\{i,j\}}^k)\big]^2. \qquad (3)\end{aligned}$$

In `fanovaGraph` the TII are estimated by `estimateGraph` and can be extracted from the output by `$ tii` or respectively `$ tii.scaled` for the TII scaled by the overall variance. To get insight about the certainty of the estimates, asymptotic 95% confidence intervals are computed additionally exploiting the estimator's asymptotic normal distribution of the Liu and Owen estimator in (3). Here an example code for the previously defined prediction function of the 6D example. Interactions of variables of the same block (e. g. $X1$ and $X3$) are clearly higher than from different blocks:

```
> g <- estimateGraph(f.mat = kmPredictWrapper, d = d,
+      n.lo = 1000, q.arg = list(min = domain[1],
+      max = domain[2]), method = "FixLO", km.object = KM)
> head(g$tii)
          totalInt    Std.Error        lower        upper
X1*X2 0.0007741586 6.330495e-05 0.0006500832 0.0008982341
```

```
X1*X3 0.0411745532 3.110340e-03 0.0350783986 0.0472707078
X1*X4 0.0010549226 1.050348e-04 0.0008490582 0.0012607871
X1*X5 0.0553378302 4.649570e-03 0.0462248410 0.0644508194
X1*X6 0.0013846099 1.167610e-04 0.0011557626 0.0016134572
X2*X3 0.0009118153 1.316350e-04 0.0006538154 0.0011698151
```

Due to the consistency of the estimator, the higher the sample number, the more exact is the estimation. In `estimateGraph` the number of Monte Carlo samples can be specified directly by `n.lo` or in terms of overall function evaluations by `n.tot`. Further parameters are the dimension size `d`, the input distribution `q` (defaults to uniform distribution) with parameters `q.arg` and the estimation method `method`. Beside the one from Liu and Owen, three further estimation methods, based on FAST, RBD-FAST and Sobol's pick and freeze method, are implemented. See [4] for a detailed description.

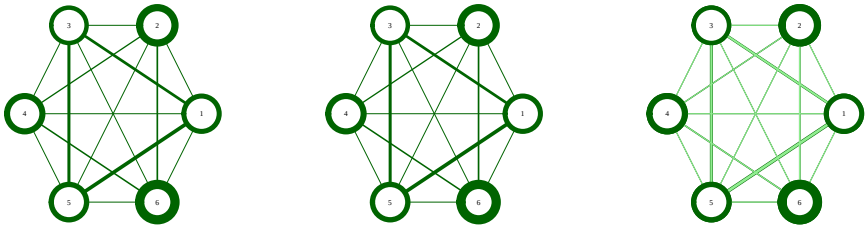
3.2. Plots

Total interaction indices are easily visualized in the FANOVA graph. Input variables are represented by the graph's vertices and are connected by an edge if the total interaction index is positive. The thickness of an edge is relative to the size of the corresponding total interaction index. With the `plotiGraph` function, graph objects obtained from `estimateGraph` can directly be plotted. There are several modifications for the graph plotting. The position of the indices in the plot is determined by the `layout` argument, adopted from `igraph0`, which searches by default for the best way. To compare graphs, the layout can be fixed. Standard main effect Sobol indices can be represented additionally by the vertex thicknesses. Moreover, pure second order Sobol indices (D_{ij}) can be added as additional inner edges since $D_{ij} < \mathfrak{D}_{ij}$ by definition. They can give insight about the amount of active interactions of orders higher then two.

```
> plot(g, ploti1 = FALSE, layout = layout.circle)

> plot(g, ploti1 = TRUE, layout = layout.circle)

> plot(g, ploti1 = FALSE, i2 = c(0, 0.03222, 0, 0.04093,
+      0, 0, 0.0135, 0, 0.023, 0, 0.03222, 0, 0, 0.01753, 0),
+      layout = layout.circle)
```



(a) Standard graph (b) Additional main effects (c) Additional second order effects

Figure 4: Different modifications of FANOVA graph plots.

3.3. Block additive Kriging

As shown in Section 2, the information brought by the FANOVA graph about total interactions can be used to build a suited Kriging model. More precisely, the clique decomposition of the graph provides a block-additive kernel of the form (see [7] for details):

$$k(\mathbf{h}) = \sum_{i=1}^{L} k_{C_i}(\mathbf{h}_{C_i}) \tag{4}$$

where $C_1, \ldots, C_L$ are the cliques, and $k_{C_1}, \ldots, k_{C_L}$ the corresponding kernels, associated to independent stationary centered Gaussian processes $Z_{C_1}, \ldots, Z_{C_L}$. Note, that the cliques can have common vertices. In `fanovaGraph`, each kernel k_{C_i} has a tensor-product structure:

$$k_{C_i}(\mathbf{h}_{C_i}) = \sigma^2_{C_i} \prod_{j \in C_i} k_1(h_j; \theta_{C_i,j}) \tag{5}$$

where $\sigma^2_{C_i}$ is the variance of Z_{C_i}, k_1 is a 1-dimensional kernel, and the $\theta_{C_i,j}$'s ($j \in C_i$) denote the other covariance parameters.
The implementation of k_{C_i} relies on `DiceKriging`. In particular, k_1 can be chosen among a list of stationary kernels (at present time: Exponential, power-exponential, Matérn 3/2, Matérn 5/2 or Gaussian (Default)), according to the smoothness of the function of interest. We refer to [9] for more details.
All covariance parameters (including the variance one) are estimated by

maximum likelihood (see [7]). However, the number of parameters equals $\sum_{i=1}^{L}(1 + |C_i|)$, and can be large. Hence, a useful option is to constrain the parameters in one clique to have a common value. We call such cliques *isotropic*.
It is often a good idea in practice to consider isotropic cliques, and several examples are provided in [7]. To illustrate its usage in `fanovaGraph`, let us consider again the 6-dimensional function f of Section 2. Looking at the estimated FANOVA graph, it seems reasonable to consider an isotropic kernel for the clique $C_1 = \{1, 3, 5\}$, since both total effects and total interaction indices are similar in between the variables of the clique. Hence, we add the constraint $\theta_{C_1,1} = \theta_{C_1,3} = \theta_{C_1,5}$ in equation 5, which gives only 6 parameters to estimate instead of 8. This change is done in `kmAdditive` by setting the argument `iso`, which is a vector of Boolean whose `TRUE` coordinates indicates the isotropic cliques. Looking again at the cliques order (see Section 2):

```
> print(Cliques)
[[1]]
[1] 1 3 5

[[2]]
[1] 2 4 6
```

we see that C_1 is indeed the first clique in the list. Hence, to make C_1 isotropic and leave C_2 unchanged, we must choose:

```
> iso <- c(TRUE, FALSE)
```

Parameter estimation is then done by writing:

```
> param <- kmAdditive(x, y, cl = Cliques, covtype = "gauss",
+           iso = iso)
```

The accuracy of the corresponding Kriging metamodel can be compared to the one based on an anisotropic block-additive kernel (see Section 2). The RMSE value shows that the predictive power looks similar, with the benefit of reducing the number of parameters.

```
[1] 0.006851003
```

Actually, for this example, the predictive power is also very good when the two cliques are assumed to be isotropic, which reduces the number of parameters to 4 only :

```
> iso <- c(TRUE, TRUE)
> param <- kmAdditive(x, y, cl = Cliques, covtype = "gauss",
+           iso = iso)
> y_block_iso <- predictAdditive(xpred, x, y, param,
+                 cl = Cliques, iso = iso)
> sqrt(mean((y_block_iso[, 1] - y_exact)^2))
[1] 0.006463344
```

Finally generating samples from block-additive kernels for simulation purposes can be done by the function `simAdd`. Here for a given kernel the Kriging covariance matrix R at desired data points $\mathbf{x}$ of size n_{sim} is computed together with its Cholesky decomposition $R = C^T C$. A simulation with mean μ and covariance matrix R is then obtained by

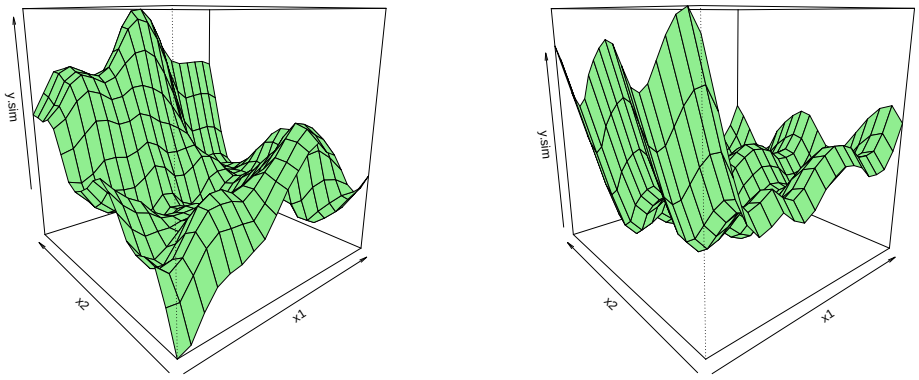
$$y = \mu + C^T \varepsilon$$

with $\varepsilon_i \sim \mathcal{N}(0,1),\ i = 1, \ldots, n_{\text{sim}}$ independent random numbers.

In the following code two simulations are generated to compare graphically a standard tensor-product kernel and an additive kernel (see Figure 5).

```
> x1 <- x2 <- seq(-1, 1, , 20)
> x.grid <- expand.grid(x1, x2)
> parameter <- list(list(alpha = 1, theta = c(0.5, 0.5)))
> y.sim <- simAdditive(x.grid, mu = 0, parameter, covtype = "matern5_2",
+      cl = list(1:2))
> persp(x1, x2, matrix(y.sim, 20), theta = -40, col = "lightgreen",
+      zlab = "y.sim")

> parameter <- list(list(alpha = 0.5, theta = 0.5),
+      list(alpha = 0.5, theta = 0.5))
> y.sim <- simAdditive(x.grid, mu = 0, parameter, covtype = "matern5_2",
+      cl = list(1, 2))
> persp(x1, x2, matrix(y.sim, 20), theta = -40, col = "lightgreen",
+      zlab = "y.sim")
```



(a) Tensor-product kernel (b) Additive kernel

Figure 5: Simulations from Kriging kernels.

4. Thresholding decision

4.1. The problem

Errors in the initial metamodel and in the index estimation usually pertubate the FANOVA graph, so that inactive indices are not necessarily zero. Therefore a thresholding cut is necessary to separate high from low indices and thus find inactive interactions. This separation can easily be done within the package by applying the `threshold` function to the graph object with a given cut value `delta`. See below for a graphical support to find a suitable value. By default, the cut is performed at the scaled indices. This can be changed through the argument `scaled=FALSE`. Here we perform a threshold cut at $\delta = 0.01$, which sets all indices that are smaller then δ to zero.

```
> g.cut <- threshold(g, delta = 0.01)
> head(g.cut$tii)
        totalInt  Std.Error      lower      upper
X1*X2 0.00000000 0.00000000 0.00000000 0.00000000
X1*X3 0.04117455 0.00311034 0.03507840 0.04727071
X1*X4 0.00000000 0.00000000 0.00000000 0.00000000
X1*X5 0.05533783 0.00464957 0.04622484 0.06445082
```

```
X1*X6 0.00000000 0.00000000 0.00000000 0.00000000
X2*X3 0.00000000 0.00000000 0.00000000 0.00000000
```

4.2. Decision plot

Decision plots are a popular way to help the user find data dependent values. We present a decision plot called *delta jump plot* that helps finding a suitable threshold value δ (see Figure 6).

```
> plotDeltaJumps(g)
```

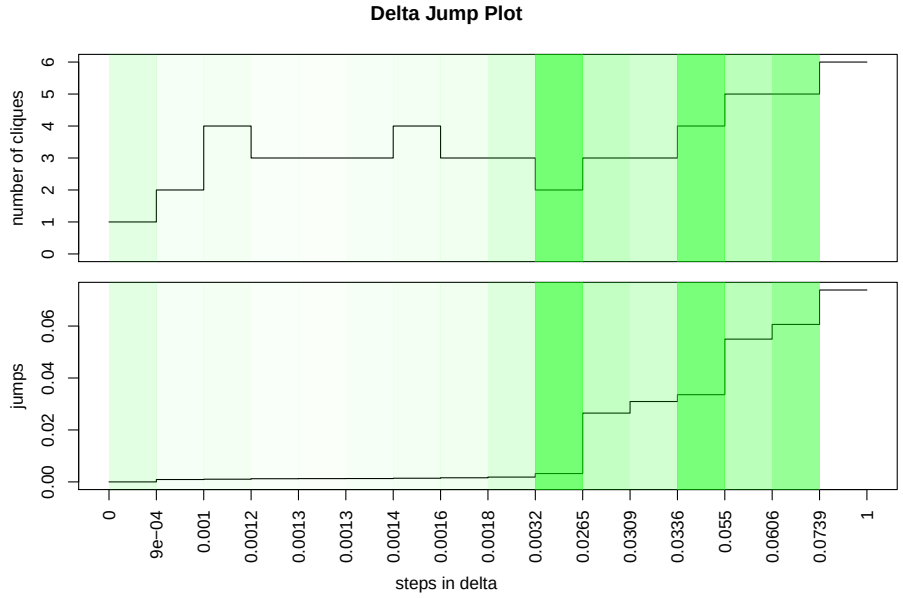


Figure 6: Delta jump plot.

On the x-axis it shows all relevant threshold values, i.e. the ordered values of indices in the FANOVA graph in equal distances. In the bottom part of the graph, the real distances of the indices are shown, so that high jumps point to big differences between successive indices. In a moderately pertubated graph, the difference between inactive and active indices should be high and thus be revealed by a jump. High jumps are additionally highlighted by shading of the relevant interval, the darker the shading the higher the jump. Values of delta that lie inside dark intervals indicate good choices for the threshold.

Simultaneously the upper part of the plot shows the number of cliques a threshold would lead to. A small number of cliques might be preferable in order to obtain a clearer structure and less parameters to estimate. Figure 6 reveals that in this case a suitable δ lies between 0.0021 and 0.025, because the jump to right of this interval is the highest and the number of cliques is very low.

5. `fanovaGraph` with further initial metamodels

As mentioned in Section 2 the metamodel is not restricted to Kriging but can be any prediction model, chosen suitable for the underlying problem. Here we show how further fitted models can easily be analyzed by `fanovaGraph`. Using the same data as before, we exemplary fit a Neural Network model and a Support Vector Machine (SVM) regression model, using standard R functions. First, for Neural Networks the output has to be scaled to lie between 0 and 1.

```
> data <- data.frame(y,x)
> data$y <- (data$y-min(data$y))/(max(data$y)-min(data$y))
```

Now we fit the models and construct a prediction function wrapper `wrapp`, that can be used for both functions.

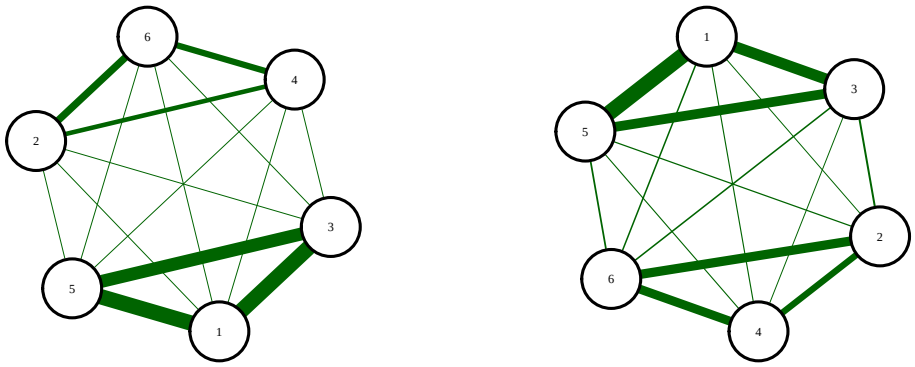
```
> library(nnet)
> library(e1071)
> nnet.mod <- nnet(y ~ ., data=data, size=2)
> svm.mod = svm(y ~ ., data=data)
> wrapp <- function(newdata, object) {
+    colnames(newdata) <- c("X1","X2","X3","X4","X5","X6")
+    as.numeric(predict(object, newdata = newdata))
+  }
```

From the prediction functions the FANOVA graph can be estimated and plotted as before (see Figure 7). Both metamodels discovered the interaction structure of function f very good, though not as good as Kriging. But of course the choice of the metamodel always depends on the situation.

```
> nnet.g <- estimateGraph(f.mat = wrapp, d = d, n.tot = 30000,
+    q.arg = list(min = domain[1], max = domain[2]), object = nnet.mod)
```

```
> svm.g <- estimateGraph(f.mat = wrapp, d = d, n.tot = 30000,
+    q.arg = list(min = domain[1], max = domain[2]), object = svm.mod)
> plot(nnet.g, plot.i1=FALSE)


> plot(svm.g, plot.i1=FALSE)
```



(a) Neural Network (b) SVM regression

Figure 7: Estimated FANOVA graph for two other metamodels on the 6D function f.

6. Conclusion

The FANOVA graph is a useful tool for visualization of interaction structures of various functions, and can especially be used for Kriging model improvement via block-additive decomposition. The R package `fanovaGraph` provides a user friendly implementation of FANOVA graphs and their application including estimation, plotting, block-additive Kriging analysis and threshold decision graphs.

One point for further improvement of the package is the metamodel error. Kriging models with second order polynomial trend have proven to better catch the interaction structure than standard Kriging models. Other Kriging kernels can be taken into consideration like FANOVA kernels, where each

interaction is modelled explicitely. Also iterative repetition of model fitting and graph fitting could be a possible alternative. Furthermore, other fields of application of the block-additive structure can be implemented like dimension reduction and optimization problems, where special clique structures are necessary.

Acknowledgements

This paper is based on investigations of the collaborative research centre SFB 708, project C3. The authors would like to thank the Deutsche Forschungsgemeinschaft (DFG) for funding. We also thank Sonja Kuhnt, Simon Neumaerker and Stefan Hess for their helpful support.

References

[1] Rob Carnell. lhs: Latin Hypercube Samples, 2012. R package version 0.10.

[2] K. Fang, R. Li, and A. Sudjianto. Design and modeling for computer experiments. Computer science and data analysis series. Chapman & Hall/CRC, Boca Raton and London, 2006.

[3] J. Fruth, T. Muehlenstaedt, and O. Roustant. fanovaGraph: Building Kriging models from FANOVA graphs, 2012. R package version 1.3.

[4] J. Fruth, O. Roustant, and S. Kuhnt. Total interaction index: A variance-based sensitivity index for interaction screening. Submitted, +2012.

[5] G. Hooker. Discovering additive structure in black box functions. In Proceedings of KDD 2004, pages 575–580. ACM DL, 2004.

[6] R. Liu and A.B. Owen. Estimating mean dimensionality of analysis of variance decompositions. Journal of the American Statistical Association, 101(474):712–721, 2006.

[7] T. Muehlenstaedt, O. Roustant, L. Carraro, and S. Kuhnt. Data-driven Kriging models based on FANOVA-decomposition. Statistics & Computing, 22(3):723–728, 2012.

[8] G. Pujol. sensitivity: Sensitivity Analysis, 2012. R package version 1.4-1.

[9] O. Roustant, D. Ginsbourger, and Y. Deville. DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by kriging-based metamodelling and optimization. In revision for the Journal of Statistical Software, +2012.

[10] O. Roustant, D. Ginsbourger, and Y. Deville. DiceKriging: Kriging methods for computer experiments, 2012. R package version 1.3.3.

[11] A. Saltelli, K. Chan, and E.M. Scott. Sensitivity analysis. Wiley series in probability and statistics. Wiley, Chichester, 2000.