Emulators in Deterministic Computer Modelling

Witold Wolski Std_NR 160223367

2018-04-06

Contents

1 Summary

Evaluating computationally intensive models at thousands of input parameter combinations, which is necessary to perform parameter sensitivity analysis or to calibrate those models, can be prohibitive. Therefore, fast to evaluate emulators might help achieve different types of input-output analysis.

We are asked to test if interpolating Gaussian Processes (GP) can be used to build an emulator. Given 100, 8-dimensional inputs and the corresponding outputs of a deterministic computer model, we have to fit an interpolating GP and do it R (R Core Team 2017). To measure how well the emulator approximates the simulator, we use a test dataset consisting of 100 8-dimensional input tuples, different from those in the training dataset, and the corresponding responses obtained by evaluating the full computer model.

Furthermore, we are asked to compare the performance of the GP with a multidimensional linear interpolation (MLI) model which is implemented in Matlab using the function `interpn` and was fitted using a different training dataset containing 3^8 points, with inputs placed on a regular grid. This dataset might give us only limited insights into the performance of GP's, since we do not expect a MLI to fit complex response surfaces in high dimensions well with only 3 support points in each dimension.

Therefore, to better understand the power of a GP as well as the complexity of the response surface (since no information about it is given), we also will fit a linear model, and include it in our analysis.

2 Introduction

A good emulator is a fast to evaluate function f which given inputs $X_1, \dots X_N$ generates in case of deterministic models exactly (with no uncertainty) the outputs $Y_1, \dots, Y_N$. Secondly, it should in a sensible way interpolate and extrapolate the outputs Y for inputs X, not in the training set and ideally also give a measure of the uncertainty of these predictions. A sensible way to perform such inter and extrapolation is to require that the function f is smooth. That means that if two close to each other inputs X and X' are evaluated, also the outputs Y and Y' should be relatively close.

Gaussian processes are a reasonable choice to produce functions with these properties (O'Hagan 2006). However, the parameters of the Kernel, e.g., a multivariate Gaussian, which determines how the emulator behaves between the design points, need to be specified. They either can be provided as input parameters if known or need to be established. There are different approaches used to assess the model hyperparameters, e.g., MLE or leave one out (LOO) (MacDonald, Ranjan, and Chipman 2015, Roustant, Ginsbourger, and Deville (2012)). One has to be aware, however, that these parameters estimation methods might be computationally intensive for large designs (many input-output points in the training dataset) since they might require multiple matrix inversions. Furthermore, identifiability issues might occur, which occur when design points are not close enough relative to the spatial correlation length Dupuy, Helbert, and Franco (2015). The second problem can, for instance, be addressed by constraining the values of the covariance or by using a penalized MLE(Roustant, Ginsbourger, and Deville 2012). A further aspect of fitting a GP is that different covariance functions, not only Gaussian can be used to specify the behavior of the responses, e.g., Matern, Exponential or power exponential kernels can be used (Roustant, Ginsbourger, and Deville 2012).

Furthermore, a GP can be fitted either to the data or the residues of a linear model. The GP is then used to interpolate the residues of the linear model (O'Hagan 2006,Roustant, Ginsbourger, and Deville (2012)). To enable the specification of the linear model to be fitted the DiceKrigging package used here allows passing a formula, in the same syntax as the `lm` function in the package stats, to the model fitting function.

The Figure 1 A, B shows how the choice of the user specified covariance parameter influences the smoothness of the curve and shape of 95 confidence interval (CI). Figure 1 C shows how removing a linear trend changes the function, especially where it is extrapolated compared with Figure 1 B (where the GP is on top of a linear trend). Furthermore, Figure 1 D shows the fit and CI when hyperparameter estimation is made using MLE method. Figure 1 E shows a Kriging model with a Matern covariance function instead of Gaussian covariance. And finally, Figure 1 F illustrates the identifiability problem. Here the MLE estimation of the hyperparameters would have produced a zero covariance if we did not constrain the MLE ($\theta > 0.01$).

3 Description of the data

The data available are one training dataset and one test dataset each consisting of 100, 8-dimensional, design points labeled $X1, X2, \dots X8$ and the corresponding simulator output `Y`. The test data also has a column `estimate` with the results of the MLI (multidimensional linear interpolator).

To build a good emulator the training points X should cover the entire input parameter space. How to best distribute those points is the subject of active research, since this is key for the performance of the emulator. Many approaches to generate optimal, space-filling designs were developed and implemented in R (see Table 4).

To understand the design of the dataset provided we will use metrics which measure the quality of the design. For instance the *coverage* criterion measures if a design is close to a regular mesh (for a regular mesh, coverage = 0, smaller better). The *meshRatio* is the ratio between the largest minimum distance and the smallest minimum distance, hence for a regular the mesh ratio = 1. Finally, the *mindist* criterion returns the smallest distance between two points. Larger *mindist* indicates that the points are spread throughout the experimental domain. We measured the quality of the training and test dataset design using these criteria (Dupuy, Helbert, and Franco 2015) and compare them with random and maximum entropy designs (see Table 2). We also recreated the design used to train the MLI and include it in the evaluation (`grep3pow8`).

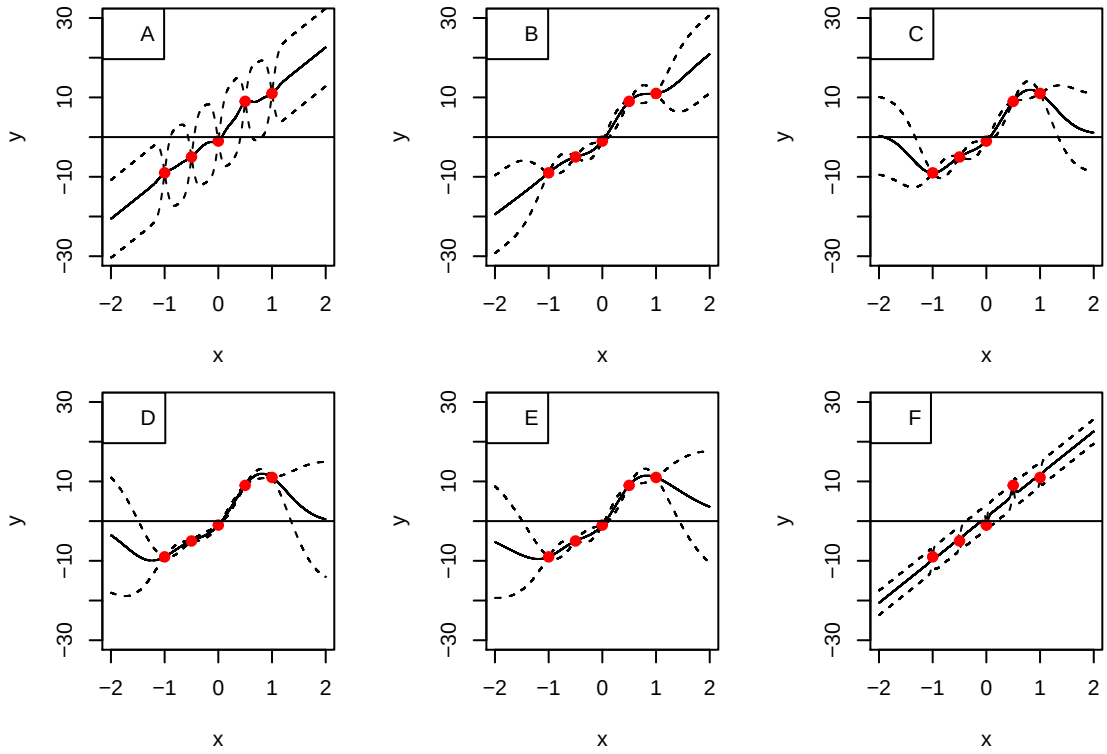

Figure 1: GP's fitted using the function DiceKrigin::km function. red points - training points. Black line - mean of the gaussian process, dashed lines - lower and upper bounds of the 95 % CI computed at newd ata. A: covtype=Gauss, cov=0.1, var=5^2, formula=~x, B: covtype=Gauss, cov=0.4, var=5^2, formula=~x, C: covtype=Gauss, cov=0.4, var=5^2, formula=~1, D: covtype=Gauss, cov=NULL, var=NULL, formula=~1, E: covtype=matern5_2, cov=NULL, var=NULL, formula=~1, F: covtype=Gauss, cov=NULL, var=NULL, formula=~1

Table 1: coverage, meshRatio and mindist computed for test, training a random and a maximum entropy design.

	grid3pow8	train	test	random	dmax
coverage	0.0	0.2097919	0.2057439	0.2092943	0.0630764
meshRatio	1.0	3.1844909	2.8342800	3.4470178	1.4477038
mindist	0.5	0.2422416	0.2923662	0.2310780	0.5921749

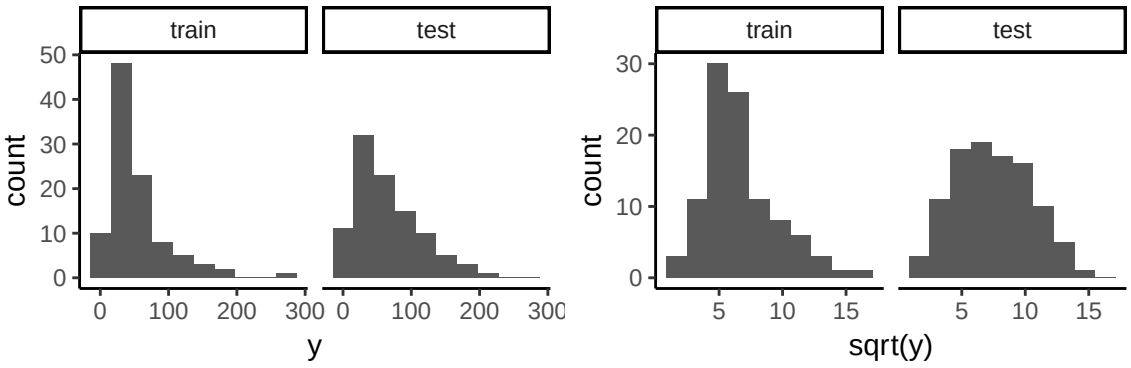

Figure 2: Histogram of simulated responses for training and test dataset

Table 2: Models fitted, km - GaussKriging::km, NA- not applicable, ?Y transformation of responses unknown.

	modelName	method	design_size	response	formula	covariance
6	linear_model	stats::lm	100	sqrt(y)	~. + I(X1..X8^2) + .^2	NA
1	Gauss_SqrtY_Const	km	100	sqrt(y)	~1	Gauss
2	Gauss_SqrtY	km	100	sqrt(y)	~.	Guass
3	Gauss_Y	km	100	sqrt(y)	~.	Gauss
4	Matern_SqrtY	km	100	Y	~.	Matern(2,5)
5	linear_Interpolation	interpn	6561	?Y	NA	NA

The 3^8 grid used to train the MLI performs best for the measures shown in table 1, but is worst design if it comes to discrepancy criteria (See Table 7) which describe how strongly a design deviates from a perfectly uniform one (Dupuy, Helbert, and Franco 2015). The regular grid projected on one dimension allows only for sampling at three distinct locations. We can conclude that the design used for the training and test data likely is not optimized but reasonable, and potentially much better than the uniform 3^8 grid.

Figure 2 show the distribution of the simulator responses for the train and test data. We observe that the distribution of the responses for the test dataset differs from that for the training dataset which also isn't ideal. We also see that square root transformation of the responses makes their distribution more normal which might simplify the modeling.

4 Methods

There are many R-packages for Gaussian Process modeling, e.g., *mlegp* (Dancik and Dorman 2008), *tgp* (R. Gramacy and Taddy 2010), *plgp* (R. B. Gramacy 2014), *gpfit* (MacDonald, Ranjan, and Chipman 2015). They differ concerning the methodology used for estimating hyperparameters, or in the variety of available covariance functions. We use here the DiceKriging package (Roustant, Ginsbourger, and Deville 2012) (see DiceKriging-CRAN) to fit the GP. Furthermore, the R package DiceEval (Dupuy, Helbert, and Franco 2015) is used to measure the quality of the fitted models.

5 Results

We are going to fit five different models: 4 GP's with various parametrizations, one linear model to better understand the response surface and we compare them with MLI results. All the models are summarised in table 2.

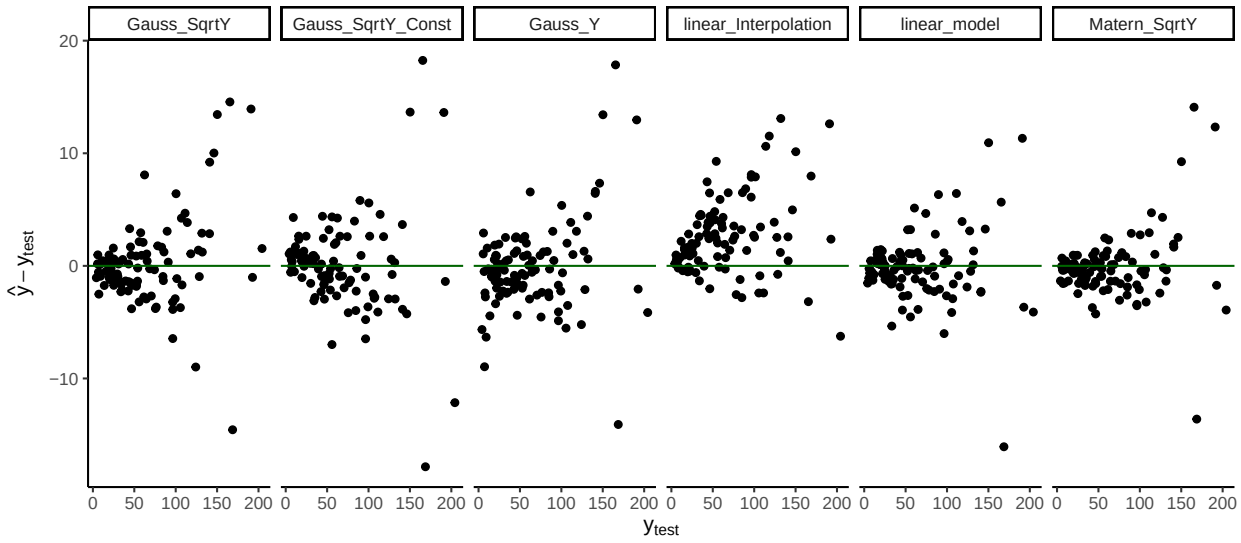

Figure 3: Altman bland plot showing residues versus y for the four models fitted to the simulator output.

Table 3: Evaluation results for all emulators.

	linear_model	Gauss_SqrtY	Matern_SqrtY	Gauss_Y	Gauss_SqrtY_Const	linear_Inter
R2	0.9954870	0.9931273	0.9959864	0.9925237	0.9920867	0.9918102
RMSE	3.1798928	3.9241312	2.9987877	4.0928300	4.2107535	4.2836893
MAE	1.9830081	2.4362368	1.7375020	2.7290325	2.5997081	3.1341870
RMA	0.3377079	0.3060313	0.2961805	0.3752053	0.3835331	0.2751189

Figure 3 shows the residues for all fitted models and the provided MLI results. For all the parametrizations the estimates of the covariances obtained by MLE were similar (Appendix Table 5). The Appendix also shows the summary of the linear model (Appendix Table 6).

5.1 Model validation and testing

We use four different measures of the goodness of fit of our model: 1) the coefficient of determination (*R2* - larger better), 2) root mean square error (*RMSE*), 3) mean absolute error (*MAE*), and 4) relative maximum absolute error (*RMA*) (Dupuy, Helbert, and Franco 2015). For more details see Appendix.

We evaluate how models are predicting the response for the test dataset and compare the model prediction with the actual response of the simulator. Table 3 shows the scores obtained for each model while Figures 4 plots those score to simplify the interpretation.

6 General discussion

Gaussian processes (GP) can be used to build emulators of high-dimensional computer models (many input parameters), also with complicated although ideally smooth response surfaces. They then can be used to calibrate that simulator (Palomo, Paulo, and García-Donato 2015), propose new design points at which to evaluate the simulator (Roustant, Ginsbourger, and Deville 2012) or perform sensitivity analysis(Dancik and Dorman 2008).

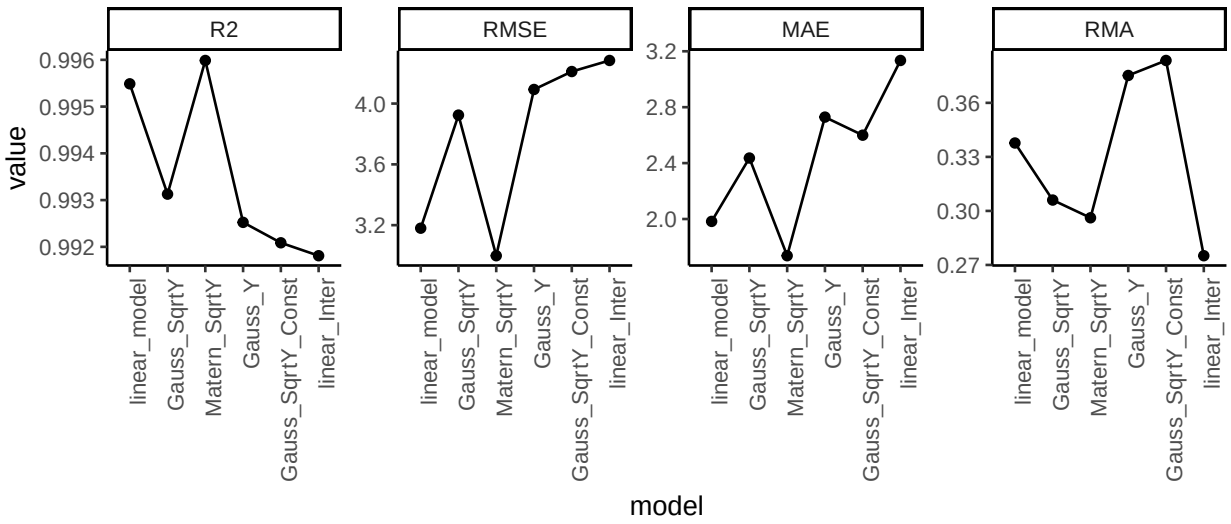

Figure 4: Plots of the 4 performance scores all emulators.

Building a good model (emulator) of the simulator based on a GP can be challenging. Figure 1 illustrates that GP's have many hyperparameters which need to be chosen, estimated or tuned. It gets even more difficult when the covariance of the outputs changes, or the response surface is not smooth. A further difficulty already discussed is that estimating those hyperparameters can be computationally challenging especially for large designs.

The GP's, fitted using the relatively small training dataset, make an excellent prediction for this particular dataset and are comparable with those obtained using the MLI trained on 60 times larger training dataset. Therefore, no surprise, GPs can be used to replace the MLI. One clear disadvantage of the MLI is that it has to be fitted using a regular grid which in high dimensions is worse than a random design. The performance of the GP can be improved by taking more care designing the points at which the simulator is evaluated.

Some questions emerged in course fo the analysis. We showed that a linear model outperforms some of the fitted GP's (See Table 3, and Figure 4) and does have only a few significant coefficients (see Table 6). Also, the parametrization of the GP does not have a significant impact on the quality of predictions and the covariances for all dimensions are relatively large and of the same order (see Table 5) indicating that indeed the function is smooth. It seems that the provided data has a response surface which is rather easy to estimate. Therefore, it would be interesting to test the GP with other more challenging datasets and benchmark it against other more sensible types of emulators than the MLI. Even more interesting and conclusive it would be to examine the performance of a GP in a real application, e.g., when calibrating a computer model.

I will provide this document as a Rmarkdown file on https://github.com/wolski/MAS6006Proj2, after the submission deadline.

Table 4: R packaes for creating designs.

design	r_package
Latin hypercube design	lhs
low discrepancy sequences	randomtoolbox, fOptions
maximum entropy designs	DiceDesign::dmaxDesign
strauss designs DiceDesign	DiceDesign::straussDesig

Table 5: Estimated covariances for the 4 GP models fitted.

name	Gauss	Gauss_SqrtY	Matern_SqrtY	Gauss_SqrtY_Const
X1	0.56	0.66	0.81	0.97
X2	1.96	1.96	1.96	1.96
X3	1.95	1.95	1.95	1.95
X4	1.93	1.93	1.52	1.93
X5	1.99	1.99	1.99	1.99
X6	1.81	1.55	1.99	1.99
X7	1.68	1.29	1.93	1.93
X8	0.45	0.35	0.60	0.62

7 Appendix

7.1 R packages for creating and evaluating designs

7.2 Estimated covariances

7.3 Linear model summary

7.4 Measures of goodness of fit.

$$R2 = 1 - \frac{\sum(y-\hat{y})^2}{\sum(y-\bar{y})^2} \quad RMSE = \sqrt{\frac{1}{n}\sum(y-\hat{y})^2}$$

$$MAE = \frac{1}{n}\sum|y-\hat{y}| \qquad RMA = \frac{\max|y-\hat{y}|}{\sigma_y}$$

where $\hat{y}$ are the predicted response and y is the true response.

7.5 Discrepancy criteria

The Lp discrepancy compares a distribution of points to the perfectly uniform design.

Table 6: Linear model summary.

term	estimate	std.error	statistic	p.value
(Intercept)	1.1471296	0.3655424	3.1381577	0.0027319
X1	5.2767441	0.3549041	14.8680849	0.0000000
X2	0.3091690	0.3595173	0.8599557	0.3935465
X3	0.7634562	0.4286677	1.7809977	0.0804354
X4	-0.2979235	0.4902340	-0.6077170	0.5458764
X5	-0.1710235	0.4868425	-0.3512913	0.7267120
X6	0.1956792	0.5154664	0.3796159	0.7056924
X7	0.1210183	0.4465591	0.2710018	0.7874042
X8	6.6880787	0.4955449	13.4964132	0.0000000
I(X1^2)	-0.4093085	0.2622423	-1.5608025	0.1243065
I(X2^2)	0.0893174	0.2764615	0.3230736	0.7478646
I(X3^2)	-0.1744464	0.2701838	-0.6456583	0.5211854
I(X4^2)	0.3775885	0.2900557	1.3017792	0.1984179
I(X5^2)	-0.0709847	0.2740086	-0.2590600	0.7965561
I(X6^2)	-0.1737260	0.2763421	-0.6286627	0.5321722
I(X7^2)	0.1764287	0.2597905	0.6791189	0.4999106
I(X8^2)	-2.6929056	0.2900355	-9.2847442	0.0000000
X1:X2	0.2061028	0.2600382	0.7925865	0.4314244
X1:X3	-0.3066218	0.2598635	-1.1799341	0.2431048
X1:X4	1.3261082	0.2716581	4.8815344	0.0000094
X1:X5	-0.0368874	0.2699037	-0.1366687	0.8917917
X1:X6	-1.4951103	0.2426993	-6.1603406	0.0000001
X1:X7	-1.3827155	0.2325417	-5.9460977	0.0000002
X1:X8	6.2833246	0.2431131	25.8452791	0.0000000
X2:X3	-0.0361261	0.2246377	-0.1608193	0.8728249
X2:X4	0.4700824	0.2462898	1.9086556	0.0615304
X2:X5	0.0063741	0.2353943	0.0270783	0.9784953
X2:X6	-0.1696043	0.2820181	-0.6013952	0.5500477
X2:X7	-0.7566745	0.2502228	-3.0240031	0.0037858
X2:X8	-0.4064867	0.2455901	-1.6551427	0.1035915
X3:X4	-0.5713907	0.2612612	-2.1870474	0.0330093
X3:X5	-0.4175902	0.2760225	-1.5128845	0.1360344
X3:X6	-0.0119915	0.2356192	-0.0508934	0.9595949
X3:X7	0.1593795	0.2384341	0.6684425	0.5066470
X3:X8	0.0037104	0.2426969	0.0152882	0.9878576
X4:X5	-0.3818483	0.2580788	-1.4795805	0.1446906
X4:X6	0.2575117	0.2940154	0.8758441	0.3849240
X4:X7	-0.0859901	0.2704321	-0.3179728	0.7517098
X4:X8	1.1553091	0.2396604	4.8206092	0.0000117
X5:X6	0.1141848	0.2978837	0.3833201	0.7029601
X5:X7	0.4979530	0.2630867	1.8927331	0.0636604
X5:X8	0.4837157	0.2996221	1.6144193	0.1121597
X6:X7	-0.2428742	0.2296466	-1.0575996	0.2948611
X6:X8	-1.4215805	0.2444225	-5.8160781	0.0000003
X7:X8	-1.2716129	0.2432316	-5.2279927	0.0000027

Table 7: Discrepancy criteria computed using 'DiceDesign:discrepancyCriteria'

	train	test	random	dmax
DisC2	0.2236048	0.1849059	0.1833877	0.2975794
DisL2	0.0000991	0.0000715	0.0000736	0.0000510
DisL2star	0.0052663	0.0052361	0.0067379	0.0061646
DisM2	0.4020719	0.3562331	0.3635660	0.5499128
DisS2	1.5869488	1.5124409	1.5041011	1.7293469
DisW2	4.4864757	4.4854248	4.4847686	4.5185922

References

Dancik, Garrett M, and Karin S Dorman. 2008. "mlegp: Statistical Analysis for Computer Models of Biological Systems Using R." *Bioinformatics* 24 (17): 1966.

Dupuy, Delphine, Céline Helbert, and Jessica Franco. 2015. "DiceDesign and Diceeval: Two R Packages for Design and Analysis of Computer Experiments." *Journal of Statistical Software, Articles* 65 (11): 1–38. doi:10.18637/jss.v065.i11.

Gramacy, Robert B. 2014. *Plgp: Particle Learning of Gaussian Processes*. https://CRAN.R-project.org/package=plgp.

Gramacy, Robert, and Matthew Taddy. 2010. "Categorical Inputs, Sensitivity Analysis, Optimization and Importance Tempering with Tgp Version 2, an R Package for Treed Gaussian Process Models." *Journal of Statistical Software, Articles* 33 (6): 1–48. doi:10.18637/jss.v033.i06.

MacDonald, Blake, Pritam Ranjan, and Hugh Chipman. 2015. "GPfit: An R Package for Fitting a Gaussian Process Model to Deterministic Simulator Outputs." *Journal of Statistical Software, Articles* 64 (12): 1–23. doi:10.18637/jss.v064.i12.

O'Hagan, Anthony. 2006. "Bayesian Analysis of Computer Code Outputs: A Tutorial." *Reliability Engineering & System Safety* 91 (10-11). Elsevier: 1290–1300.

Palomo, Jesús, Rui Paulo, and Gonzalo García-Donato. 2015. "SAVE: An R Package for the Statistical Analysis of Computer Models." *Journal of Statistical Software, Articles* 64 (13): 1–23. doi:10.18637/jss.v064.i13.

R Core Team. 2017. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Roustant, Olivier, David Ginsbourger, and Yves Deville. 2012. "DiceKriging, Diceoptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization." *Journal of Statistical Software, Articles* 51 (1): 1–55. doi:10.18637/jss.v051.i01.