Comparing Kriging and Radial Basis Function Surrogates

Samineh Bagheri[1], Wolfgang Konen[1], Thomas Bäck[2]

[1]Department of Computer Science
TH Köln (University of Applied Sciences)
Steinmüllerallee 1
51643 Gummersbach
E-Mail: {wolfgang.konen, samineh.bagheri}@th-koeln.de,

[2]Department of Computer Science
Leiden University, LIACS,
2333 CA Leiden, The Netherlands
Email: T.H.W.Baeck@liacs.leidenuniv.nl

Abstract

Radial basis function interpolation (RBF) and Kriging based on the Gaussian processes models (GPs) are great tools for multidimensional surrogate modeling. They can often deliver accurate models for complicated nonlinear functions. Although RBFs and GPs have very different origins, they share many fundamentals in practice. Gaussian processes not only provide a model but also an error estimate associated with the model which indicates the uncertainty of the model at different points. The uncertainty property determined by GPs is dependent on the distribution of the sample points in the input space and as expected the model error estimate is low where the distribution is dense and large where the distribution is sparse. Therefore, the uncertainty property can be determined regardless of the modeling technique.

In this work, we compare RBFs and GPs from the theoretical point of view. We show how to calculate the model uncertainty for any arbitrary kernel, e. g. cubic RBF, augmented cubic RBF and other types. Furthermore, we replace the Kriging model from the DiceKriging R package used in the SOCU framework [1] with a vectorized RBF model and report some preliminary results. We show that the new implementation of SOCU with RBF is faster than the older one. It delivers in many cases equal or even better optimization accuracy.

1 Introduction

Real-world optimization problems are often black-box and function evaluation can be very expensive.In order to handle such problems, different surrogate-assisted constrained optimization algorithms have been developed [1, 10, 8]. A common approach in surrogate-assisted optimization is the so-called *Expected Improvement* (EI) method based on Kriging models. EI was originally developed for unconstrained optimization, but approaches for constrained optimization have been proposed as well [1, 10], one of them being the SOCU algorithm [1].

Kriging has the big advantage of providing uncertainty information in surrogates, which is a necessary prerequisite for EI. But Kriging – at least in most currently available implementations – has also some disadvantages: We experienced in SOCU often crashes of the Kriging package if we did not introduce some form of regularization by setting a nonzero value to the noise variance parameter. This, however, leads in turn to less accurate models. Secondly, Kriging model calculations are often time-consuming, if either the dimensionality, the number of design points or the number of constraints becomes higher.

RBF surrogate models, which we used in other optimizers [4, 8], can be computed fast, in vectorized form, and robustly. They lack however the uncertainty information. The motivation for this paper is driven by the following research question: *Can we advise - by exploiting the analogies between RBF and GP - some form of RBF modeling which provides an uncertainty measure?* And if so, can we apply it to an EI-based optimization scheme like SOCU and in such arrive at a method having one or several of these desirable properties:

- Avoiding crashes without regularization
- Providing more accurate models
- Better computation time (e. g. through vectorization)
- More variety in radial basis kernels (parameter-free, augmented, ...)

The rest of this paper is organized as follows: In Section. 2, we describe Gaussian processes, radial basis function interpolation and their connection with each other. A brief description of the SOCU algorithm also appears in the same section. Section 3 describes the experimental setup and the test functions. We report our results and compare different versions of SOCU in Section 4. Section 5 concludes the paper.

Table 1: Commonly used kernel functions for GP and radial basis functions for RBF interpolation. $r = ||x_i - x_j||$

Name	GP	RBF
cubic	–	$\varphi(r) = r^3$
Gaussian	$\sigma e^{-\frac{r^2}{2\alpha^2}}$	$\varphi(r, \alpha) = e^{-\frac{r^2}{2\alpha^2}}$
multiquadric	–	$\varphi(r, \alpha) = \sqrt{1 - (\frac{r}{\alpha})^2}$
matern(3-2)	$\sigma(1 + \frac{\sqrt{3}r}{\alpha}) \exp(-\frac{\sqrt{3}r}{\alpha})$	–

2 Methods

2.1 Gaussian Processes

Gaussian processes (**GP**) – also known as Kriging – is a probabilistic modeling technique which applies Bayesian inferences over functions. Let us assume that an unknown function f is evaluated on a finite set of n arbitrary points $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ and $f_i = f(\mathbf{x}_i) = y_i$. The Gaussian processes method assumes that $p(f(\mathbf{x}_1), f(\mathbf{x}_2), \cdots, f(\mathbf{x}_n))$ belongs to a multivariate (jointly) Gaussian with a mean μ and covariance matrix $\mathbf{\Sigma}$:

$$\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \sim N \left(\begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} & \cdots & \Sigma_{1n} \\ \Sigma_{21} & \Sigma_{22} & \cdots & \Sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{n1} & \Sigma_{n2} & \cdots & \Sigma_{nn} \end{bmatrix} \right) \sim N(\mu, \mathbf{\Sigma}) \qquad (1)$$

where $\Sigma_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$. The covariance matrix contains the dependencies and similarities of random variables, in this case the $f(x_i)$. Suppose the unknown function f is smooth, then it is very likely that two points which are located very close to each other in the input space have very similar values in the output space too. This explains why the κ is also known as the similarity function and is often symmetric and a function of distances. Table 1 shows several commonly used kernel functions.

Suppose that we want to predict f_* the value of function f at a new point $\mathbf{x}_*$. The joint Gaussian distribution including the new point is

$$\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \\ f_* \end{bmatrix} \sim N\left(\begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \\ \mu_* \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} & \cdots & \Sigma_{1n} & \Sigma_{1*} \\ \Sigma_{21} & \Sigma_{22} & \cdots & \Sigma_{2n} & \Sigma_{1*} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \Sigma_{n1} & \Sigma_{n2} & \cdots & \Sigma_{nn} & \Sigma_{n*} \\ \Sigma_{*1} & \Sigma_{*2} & \cdots & \Sigma_{*n} & \Sigma_{**} \end{bmatrix} \right) \tag{2}$$

which can be summarized as follows

$$\begin{bmatrix} \mathbf{f} \\ f_* \end{bmatrix} \sim N\left(\begin{bmatrix} \mu \\ \mu_* \end{bmatrix}, \begin{bmatrix} \mathbf{K} & K_* \\ K_*^T & K_{**} \end{bmatrix} \right), \tag{3}$$

where $\mathbf{K} = \Sigma$ is the $n \times n$ matrix of Eq. (1), $K_* = \kappa(\mathbf{X}, x_*)$ is an $n \times 1$ vector and $K_{**} = \kappa(\mathbf{x}_*, \mathbf{x}_*)$ is a scalar and $\mathbf{f} = \{f_1, f_2, \cdots, f_n\}$ is an $n \times 1$ vector. $\mathbf{X} = (\mathbf{x}_1 \ldots \mathbf{x}_n)$ is the matrix of the data points. We look for the probability of f_* when the data $\mathbf{X}$, their corresponding values $\mathbf{f}$ and the new point $\mathbf{x}_*$ are given. Based on the conditional probability theorem [6] we can determine a distribution at every new point as follows:

$$p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{f}) = N(K_*^T \mathbf{K}^{-1} \mathbf{f}, K_{**} - K_*^T \mathbf{K}^{-1} K_*) = N(\mu_*, \Sigma_*), \tag{4}$$

where μ_* and Σ_* can be interpreted as the mean and the uncertainty of the Gaussian processes model, respectively. Fig. 1 shows an example of Gaussian process with a Gauss kernel function. Fig. 1-left illustrates several samples from the prior $p(\mathbf{f}|\mathbf{X})$ and Fig. 1-right shows samples from the posterior function $p(f_*|\mathbf{X}_*, \mathbf{X}, \mathbf{f})$. The dark black curve is the mean μ_* and the shaded area is showing the 90% confidence interval.

We can rewrite the mean as follows:

$$\begin{aligned} f_* &= K_*^T \mathbf{K}^{-1} \mathbf{f} \\ f_* &= K_*^T \underline{\theta} \\ f_* &= \sum_{i=1}^{n} \theta_i \kappa(\mathbf{x}_i, \mathbf{x}_*), \end{aligned} \tag{5}$$

Eq. (5) shows that the mean of the GP model can be determined as a linear summation of weighted kernel functions.

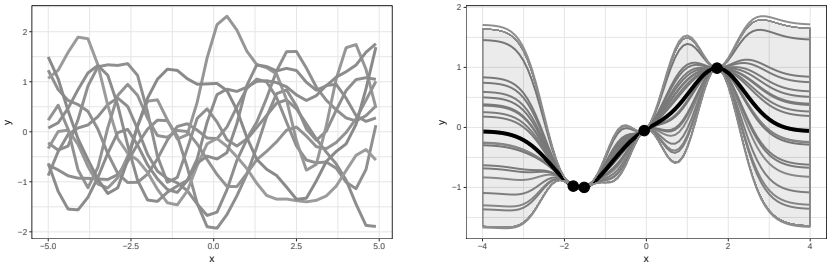

Figure 1: Left: prior. Right: posterior.

The uncertainty term in Eq. (4) can be rewritten as:

$$\Sigma_* \quad = \quad -K_*^T K^{-1} K_* + K_{**} \tag{6}$$

It is important to mention that the uncertainty of the model estimated by Eq. (6) is only a function of the distribution of points in the input space. Fig. 1 illustrates that the model uncertainty goes to zero at the given points and it becomes larger as the distance from the evaluated points increases.

2.2 Radial Basis Function Interpolation

Radial basis function (RBF) interpolation developed by Hardy in 1971 for cartography purposes was meant to be a suitable interpolation technique to model hills and valleys with a reasonably high local and global accuracy. RBF interpolation approximates a function by fitting a linear weighted combination of radial basis functions. A radial basis function is by definition any function $\varphi(||\cdot||)$ which is dependent on the distance of the points $\mathbf{x}$ from some fixed centers $\mathbf{c}$. It is common that all the evaluated points $\mathbf{x}_i$ are considered as centers.

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{n} \theta_i \phi(||\mathbf{x} - \mathbf{x}_i||) \tag{7}$$

In order to compute the weights θ_i we need to address the following linear system:

$$[\mathbf{\Phi}]\,[\theta] = [\mathbf{f}], \tag{8}$$

where $\mathbf{\Phi} \in \mathbb{R}^{n \times n}$: $\Phi_{ij} = \varphi(||\mathbf{x}_i - \mathbf{x}_j||), i, j = 1, \ldots, n$ and $\mathbf{f} = f_1, f_2, \cdots, f_n$. Therefore, the weights will be determined as following:

$$\theta = \mathbf{\Phi}^{-1}\mathbf{f} \tag{9}$$

Now that we have the weights θ, we can compute f_* at any point $\mathbf{x}_*$:

$$f_* = \sum_{i=1}^{n} \theta_i \phi(||\mathbf{x}_* - \mathbf{x}_i||) = \Phi_*^T \theta = \Phi_*^T \mathbf{\Phi}^{-1}\mathbf{f} \tag{10}$$

where $\Phi_*^T = [\varphi(||\mathbf{x}_* - \mathbf{x}_1||), \varphi(||\mathbf{x}_* - \mathbf{x}_2||), \cdots, \varphi(||\mathbf{x}_* - \mathbf{x}_n||)]$.

2.3 Augmented RBF

It is proven that $\mathbf{\Phi}$ in Eq. 8 is not guaranteed to be positive definite for all radial basis functions [5]. In order to assure that the Eq. 8 has a unique solution with all radial basis functions, Micchelli introduced augmented RBFs [5]. Augmented RBFs are actually RBF functions with a polynomial tail.

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{n} \theta_i \varphi(||\underline{x} - \underline{u}^{(i)}||) + p(\underline{x}), \quad \underline{x} \in \mathbb{R}^d, \tag{11}$$

where $p(\underline{x}) = \mu_0 + \underline{\mu_1}\,\underline{x} + \underline{\mu_2}\,\underline{x}^2 \cdots + \underline{\mu_k}\,\underline{x}^k$ is a k-th order polynomial in d variables with $kd + 1$ coefficients.

The augmented RBF model requires the solution of the following linear system of equations:

$$\begin{bmatrix} \mathbf{\Phi} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{0}_{(kd+1)\times(kd+1)} \end{bmatrix} \begin{bmatrix} \underline{\theta} \\ \underline{\mu}' \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ 0_{(kd+1)} \end{bmatrix} \tag{12}$$

Here, $\mathbf{P} \in \mathbb{R}^{n \times (kd+1)}$ is a matrix with $(1, \underline{x}_{(i)}, \underline{x}_{(i)}^2)$ in its ith row, $\mathbf{0}_{(kd+1)\times(kd+1)} \in \mathbb{R}^{(kd+1)\times(kd+1)}$ is a zero matrix, $0_{(kd+1)}$ is a vector of zeros. In this work we use the augmented cubic radial basis function with a second order polynomial tail.

2.4 GP vs. RBF

Although GP and RBF interpolation have two very different origins, the comparison of Eq. (10) and Eq. (5) shows that the mean of GP is identical to the RBF result, if the kernel function κ is identified with the basis function φ. On the other hand, GPs provide beside the prediction of the mean also a prediction of the model uncertainty Σ_* (Eq. (6)). Although radial basis functions by definition do not have any sort of uncertainty measure, we can determine the model uncertainty for any radial basis function in a similar way as GP does.

$$\Sigma_{rbf} = \varphi(||\mathbf{x}_* - \mathbf{x}_*||) - \Phi_*^T \boldsymbol{\Phi}^{-1} \Phi_*, \tag{13}$$

where $\varphi(||\mathbf{x}_* - \mathbf{x}_*||) = \varphi(0)$ is a scalar value.

Fig. 2 illustrates that RBF and Kriging with the same kernel type and parameters give almost the same results. The minimal differences in the first three columns are due to different matrix inversion techniques which the two implementations use. As it is shown in Fig. 2 the choice of kernel parameter has a large impact on the quality of the models. In this example, small values of α resulted in a very non-informative spiky model. The correct choice of parameter(s) is often a problem dependent challenging task. Kriging [9] tunes the kernel parameter(s) based on the maximum likelihood estimation (MLE) approach which its computational complexity is approximately $\mathcal{O}(\frac{1}{3}n^3 + \frac{1}{2}dn^2)$.

The kernel parameters for the RBF interpolation are often set manually. A recent work [2] suggests an online selection algorithm for choosing the best kernel type and parameters during an optimization process. In this work we use a parameter free radial basis function.

The plots in the last column of Fig. 2 are generated by the default configuration of RBF and Kriging. RBF's default configuration uses an augmented cubic kernel function with no need for parameter tuning. Kriging uses a Gauss kernel and the kernel parameters are assigned by MLE. We can see that the augmented cubic RBF model produces smaller errors than the default Kriging model (with MLE). Furthermore, we can observe that the Kriging model with MLE is not as good as Kriging with fixed parameters $\sigma = 1, \alpha = 10$ for the example shown in Fig. 2. Rasmussen and Williams [7] show that MLE cannot guarantee optimal estimation of kernel parameters, since the likelihood function can have multiple local optima. MLE can suffer from getting stuck in such a local optimum.

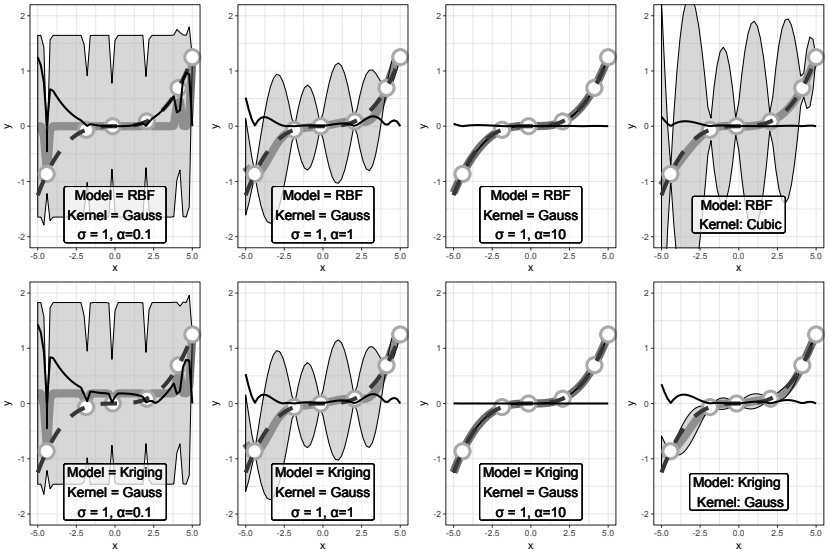

Figure 2: Comparing RBF and GP from the DiceKriging package in R. The examples in the first row are all generated by RBF and in the second row with GP. The plots in the last column are generated by the default configuration of RBF and GP. The dashed curve $f = x^3$ is the target curve to be modeled. The circles are the evaluated points. The solid thick curve is the delivered model. The solid thin curve is the model's absolute error. The gray areas indicate the 90% model uncertainty. For the same kernel function and same parameters, RBF and GP produce almost the same results. The minimal differences are due to different matrix inversion techniques used by the two implementations.

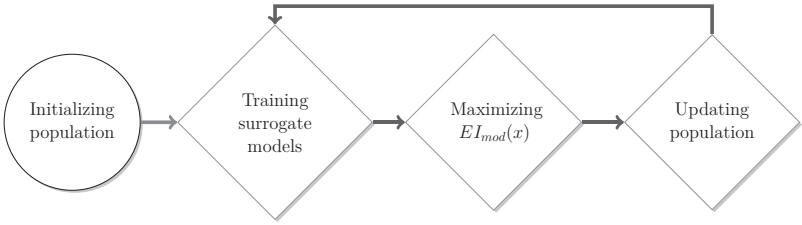

Figure 3: Main steps of the SOCU algorithm.

2.5 SOCU: Surrogate-Assisted Optimization encompassing Constraints and Uncertainties

The SOCU algorithm developed in [1] is a surrogate assisted constrained optimization method which makes use of Gaussian processes to model objective function and feasibility probability.

SOCU is a sequential EGO-based [3] optimization algorithm with four main steps as illustrated in Fig. 3. After generating an initial population of solutions, the objective and constraint function(s) are modeled. The next step is to generate a new infill point which maximizes the modified expected improvement:

$$EI_{mod}(x) = EI(x) \cdot F(x) = EI(x) \cdot \prod_{i=1}^{m} \min\Big(2P(g_i(x) < 0), 1\Big), \qquad (14)$$

where $EI(x)$ is the expected improvement of the objective function and $P(g_i(x) < 0)$ is the probability of the i-th constraint not being violated. The single new infill point is evaluated on the real functions and added to the population. The last three steps will be repeated as long as the budget is exhausted. More details about the computation of $EI_{mod}(x)$ can be found in [1].

3 Experimental Setup

In this work we compare two versions of the SOCU optimization framework, namely SOCU-Kriging and SOCU-RBF. SOCU-Kriging uses the Kriging model from R packages DiceKriging and DiceOptim, while SOCU-RBF uses our own implementation of RBF interpolation. The first

Table 2: Characteristics of the G-functions: d: dimension, ρ^*: feasibility rate (%), LI/NI: number of linear / nonlinear inequalities, a: number of constraints active at the optimum. We selected here only those G-functions without equality constraints.

Fct.	d	ρ^*	LI / NI	a
G01	13	0.0003%	9 / 0	6
G04	5	26.9217%	0 / 6	2
G06	2	0.0072%	0 / 2	2
G07	10	0.0000%	3 / 5	6
G08	2	0.8751%	0 / 2	0
G09	7	0.5207%	0 / 4	2
G10	8	0.0008%	3 / 3	6
G12	3	0.04819%	0 / 1	0
G24	2	0.44250%	0 / 2	2

major difference between the two versions of SOCU is the choice of kernel functions. SOCU-Kriging uses matern3-2 which is a relatively stable kernel function according to our initial experiments. SOCU-RBF makes use of cubic basis function which is a parameter-free kernel function.

DiceKriging uses a maximum likelihood estimation (MLE) algorithm to tune the two parameters of the matern3-2 kernel. The second difference is the numerical technique used in both versions for the required matrix inversion. The DiceKriging package [9] uses Cholesky decomposition but we found singular value decomposition used for RBF to be a more stable approach. In our experiments described in [1] we experienced frequent crashes of Kriging models. It was possible to cure this problem by using a non-zero regularization factor that can be assigned as the noise variance parameter. In this work we represent SOCU-Kriging results with two regularization factors of $\delta = \{10^{-3}, 10^{-4}\}$. The third major difference is an implementation detail which is the underlying reason for SOCU-RBF being much more time-efficient than SOCU-Kriging. The DiceKriging package does not support modeling several functions simultaneously which means that for a problem with m constraints, we have to run through a loop $m + 1$ times in each iteration, while SOCU-Kriging uses vectorization and performs training and prediction of all models within one pass. The main differences between SOCU-RBF and SOCU-Kriging are summarized in Table. 3.

Table 3: Differences between SOCU-Kriging and SOCU-RBF

	SOCU-Kriging	SOCU-RBF
kernel	matern3-2	cubic
parameter assignment	MLE	parameter free
matrix inversion	cholesky decomposition	svd
noise variance	0.001	0.0
vectorization	no	yes

We apply SOCU-Kriging and SOCU-RBF to the subset of G-problems having only inequality constraints (see Table 2). G02 is a scalable problem in its dimension d. We use here $d = 2$. For each algorithm we run 10 independent trials with different $n = 3d$ initial points. In order to optimize EI_{mod} we use Generalized Simulated Annealing (R package GenSA).

4 Results and Discussion

4.1 Performance on G-problems

Fig. 4 shows the optimization results over iterations for SOCU-Kriging and SOCU-RBF. For most of the problems, SOCU-RBF performs better than or comparable to SOCU-Kriging except for G12 and G24. G12 and G24 are the only problems where SOCU-RBF has a larger median error. However, several optimization runs for G12 conducted by SOCU-RBF perform better than the best runs of SOCU-Kriging.

4.2 Computational Time

Fig. 5 clearly shows that SOCU-Kriging is computationally more expensive than SOCU-RBF for all G-problems. SOCU-Kriging's computational time varies strongly in a range of (0.5-2.5) minutes per iteration for different G-problems, while SOCU-RBF's computational time per iteration is under 0.75 minutes regardless of the problem. The difference between computational time of SOCU-Kriging and SOCU-RBF is dependent on

the number of constraint functions. For example, the largest gap between SOCU-Kriging and SOCU-RBF appears to be for G01 and G07 which have 9 and 8 constraint functions, respectively. We can observe that solving G12 with one constraint has almost the same computational cost for SOCU-Kriging and for SOCU-RBF.

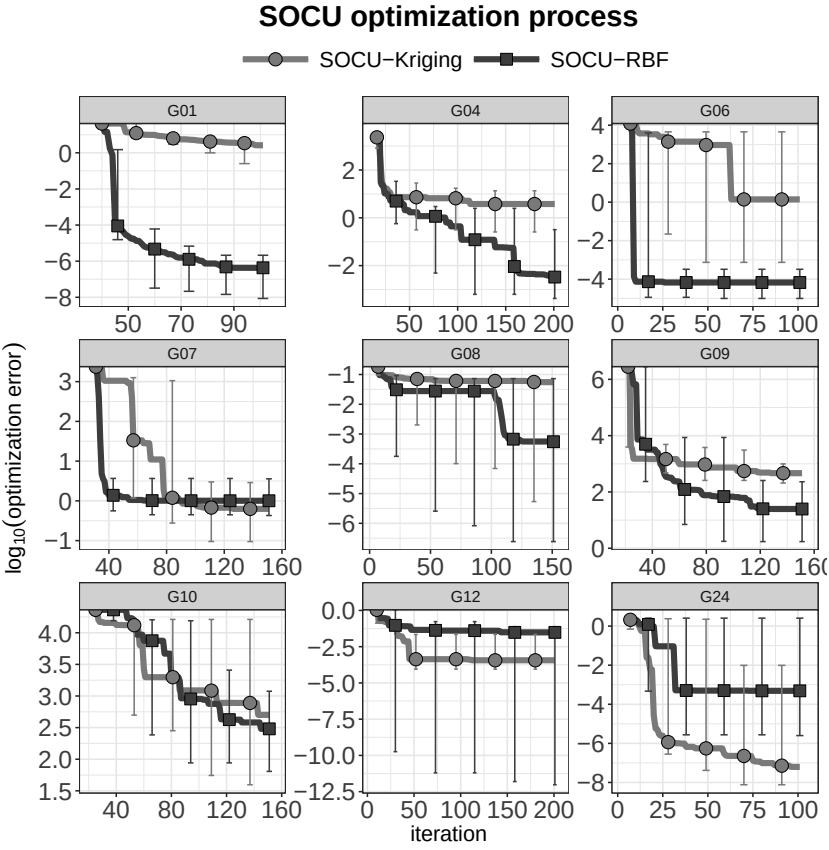

Figure 4: Comparing optimization performance of SOCU-Kriging and SOCU-RBF on G-problems. The curves are showing the median error out of 10 trials. The error bars indicate the best and the worst results out of 10 trials.

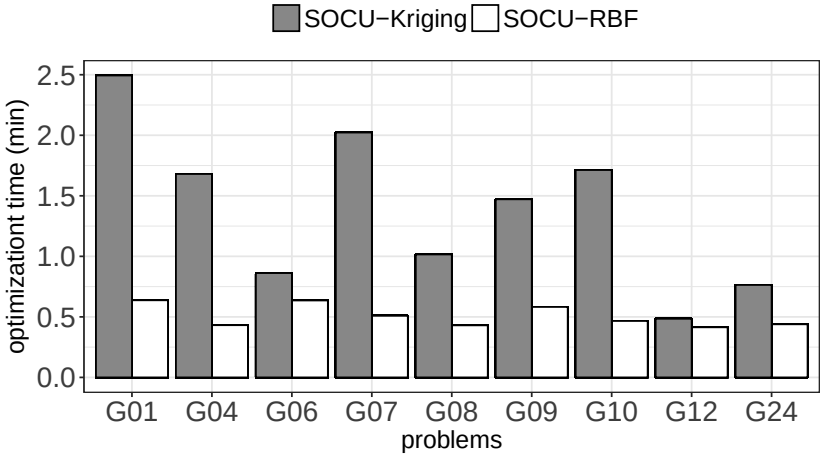

Figure 5: Average computational time required by SOCU-Kriging and SOCU-RBF to run one iteration of each G-problem.

4.3 Noise Variance

We have already shown that SOCU-RBF outperforms SOCU-Kriging in Fig. 4. The SOCU-Kriging algorithm used for generating Fig. 4 has a non-zero noise variance $\delta = 10^{-3}$. One possible reason behind the weaker performance of SOCU-Kriging in comparison to SOCU-RBF can be that SOCU-Kriging generates less accurate models due to the non-zero noise variance value. In order to investigate the impact of noise variance we applied the SOCU-Kriging framework to all G-problems in Table. 2 with two different noise variance values $\delta_1 = 10^{-3}$ and $\delta_2 = 10^{-4}$. SOCU-Kriging with the smaller noise variance $\delta_2 = 10^{-4}$ crashed on the G06 problem. Fig. 6 compares the SOCU-Kriging optimization results with two different noise variance values for all problems excluding G06.

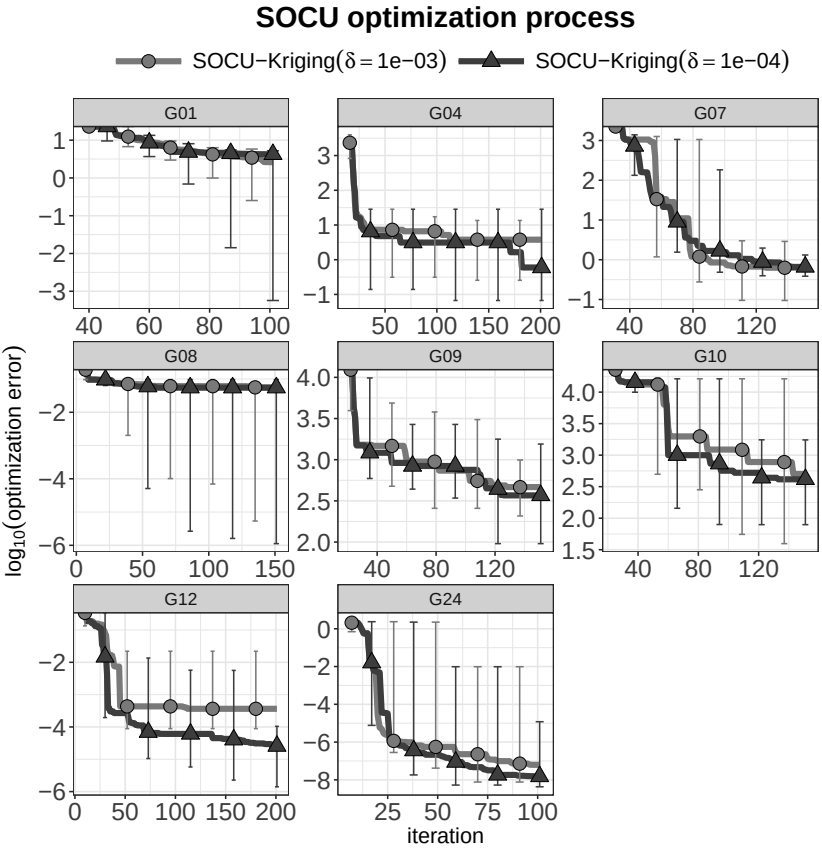

Figure 6: Comparing results of SOCU-Kriging with two different noise variance values δ. The curves are showing the median optimization error of the 10 independent trials for each algorithm. The error bars are indicating the best and worst case results.

Fig. 6 indicates that a smaller noise variance value can lead to a slightly better optimization results. For all the problems illustrated in Fig. 6 except G07, SOCU-Kriging $\delta = 10^{-4}$) has a smaller median or min. error. It is not possible to set the noise variance to zero because this would produce frequent crashes for SOCU-Kriging.

4.4 Model Accuracy

SOCU-Kriging and SOCU-RBF are only distinct in the modeling approach. Therefore, we hypothesize that the different optimization results observed in Fig. 4 are due to the different model quality. The performance of SOCU-Kriging and SOCU-RBF is significantly different especially on the G01 problem. In order to validate our hypothesis, we show the approximation error determined during the optimization process with various SOCU configurations in Fig. 7. As it is shown in Fig. 7, the approximation error of SOCU-RBF is significantly smaller than both SOCU-Kriging versions for objective and constraint functions. This shows that Kriging with matern3-2 kernel and optimized parameters through MLE cannot compete with our implementation of RBF with the parameter free augmented cubic kernel for G01. A large part of SOCU-RBF's better performance can probably be attributed to the augmented part. This advantage may depend on the type of the function to be modeled.

Comparing different versions of SOCU-Kriging in Fig. 7, we can also observe that SOCU-Kriging with a smaller noise variance $\delta = 10^{-4}$ has slightly smaller approximation error in the last iterations.

5 Conclusion

We explored in this work the similarities and differences between Kriging- and RBF-based surrogate models. As a new point from this comparison we could implement an uncertainty measure for RBFs which is needed for EI-optimization. RBFs allow a greater variety of kernel functions, notably in the form of augmented RBF variants introduced in Sec. 2.3. This helps to avoid crashes in the model-building process, which are otherwise encountered from time to time in Kriging modeling. RBF models have shown to provide a higher modeling accuracy and higher robustness (they do not produce crashes in all our experiments).

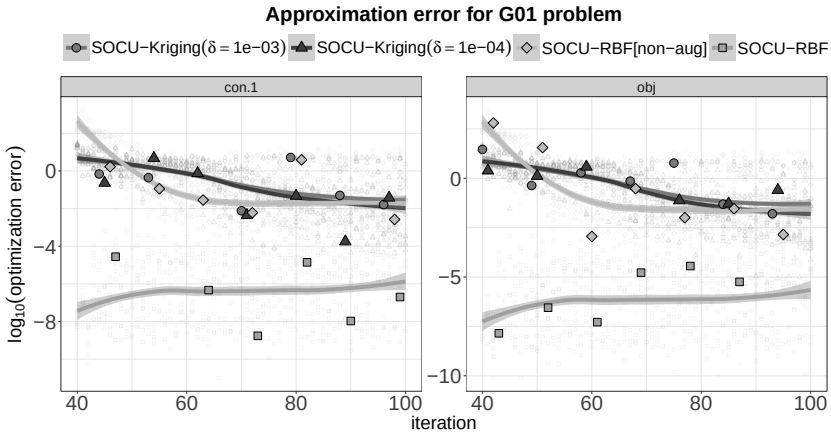

Figure 7: Approximation error for the objective and constraint functions of the G01 problem. G01 has 9 constraints but because of lack of space we just show the approximation error of the objective (a quadratic function) and one of its constraint functions (a linear function). Since all 9 constraints are of the same type, their approximation error curves looks similar. The approximation error is the error in predicting at the new infill point before this point is added to the population.

The new RBF surrogate models including uncertainties were tested on certain optimization benchmarks (a subset of the G-problems). The overall results were better, both in terms of solution quality and computational time. Probably a large part of the quality improvement may be attributed to the ability of augmented RBF models to include a polynomial tail. This may be a large or small advantage, depending on the type of functions to be modeled.

References

[1] Bagheri, S., Konen, W., Allmendinger, R., Branke, J., Deb, K., Fieldsend, J., Quagliarella, D., Sindhya, K.: Constraint handling in efficient global optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 673–680. GECCO '17, ACM, New York, NY, USA (2017)

[2] Bagheri, S., Konen, W., Bäck, T.: Online selection of surrogate models for constrained black-box optimization. In: 2016 IEEE Sym-

posium Series on Computational Intelligence (SSCI). pp. 1–8 (Dec 2016)

[3] Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. Journal of Global Optimization 13(4), 455–492 (Dec 1998)

[4] Koch, P., Bagheri, S., Foussette, C., Krause, P., Bäck, T., Konen, W.: Constrained optimization with a limited number of function evaluations. In: Hoffmann, F., Hüllermeier, E. (eds.) Proceedings of the 24th Workshop on Computational Intelligence. pp. 119–134. Universitätsverlag Karlsruhe (2014), young Author Award GMA-CI

[5] Micchelli, C.A.: Interpolation of scattered data: Distance matrices and conditionally positive definite functions. Constructive Approximation 2(1), 11–22 (Dec 1986)

[6] Murphy, K.P.: Machine Learning: A Probabilistic Perspective. The MIT Press (2012), pp. 118-121

[7] Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. the MIT Press (2006), pp. 114-117

[8] Regis, R.G.: Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. Engineering Optimization 46(2), 218–243 (2013)

[9] Roustant, O., Ginsbourger, D., Deville, Y.: DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by Kriging-based metamodeling and optimization. Journal of Statistical Software 21, 1–55 (2012)

[10] Schonlau, M., Welch, W.J., Jones, D.R.: Global versus local search in constrained optimization of computer models. In: Flournoy, N., et al. (eds.) New developments and applications in experimental design, Lecture Notes–Monograph Series, vol. 34, pp. 11–25. Institute of Mathematical Statistics, Hayward, CA (1998)