Self-adjusting surrogate-assisted optimization techniques for expensive constrained black box problems

Bagheri, S.

Chapter 7

Radial Basis Function vs. Kriging Surrogates

7.1 Outline

Radial basis function interpolation (RBF) and Kriging based on the Gaussian processes models (GPs) are popular tools for surrogate modeling. These modeling techniques can deliver accurate models for complicated nonlinear functions, even if only a limited number of evaluated points are affordable. In this dissertation, two different surrogate-assisted optimizers, SACOBRA and SOCU are described. These frameworks use RBF and GPs, respectively. Their different performance in terms of computational time and optimization quality on various COPs motivates us to compare them.

Although RBFs and GPs have very different origins, they share many fundamentals in practice. Gaussian processes not only provide a model but also a stochastic error term which indicates an estimation for the model uncertainty at each point. The stochastic error term determined by GPs is dependent on the distribution of the sample points in the input space. As expected, the uncertainty of the model is low where the distribution is dense and large where the distribution is sparse.

In this chapter, we compare RBFs and GPs from the theoretical point of view. We show how to calculate the model error 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 [19] 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 and it delivers in many cases equal or even better optimization accuracy. The results and analysis in this chapter are mainly taken from [17].

The rest of this chapter is organized as follows: In Sec. 7.2 the main motivation is explained and a research question is formulated. Sec. 7.3 gives a brief overview

about the historical development of RBF and GP. In Sec. 7.4, we describe Gaussian processes, radial basis function interpolation and their connection to each other. Section 7.5 describes the experimental setup and the test functions. We compare different versions of SOCU in Section 7.6. Section 7.7 concludes this chapter and answers the research question.

7.2 Introduction

Up till now we have described two different surrogate-assisted constrained optimizers in Ch. 3 and Ch. 5. These algorithms are designed to handle time-expensive COPs in an efficient manner, as for the real-world applications often a limited number of function evaluations is affordable. SACOBRA described in Ch. 3, uses RBF interpolations as surrogates of objective and constraint functions. SOCU described in Ch. 5, however, uses another common approach in surrogate-assisted optimization, the so-called *Expected Improvement* (EI) method based on Kriging models aka Gaussian process. EI was originally developed for unconstrained optimization, but it got modified for constrained optimization as well [19, 159].

Kriging has the big advantage of providing uncertainty information for surrogates, which is necessary for determining EI. But Kriging – at least in most currently available implementations – has also some disadvantages: In Ch. 5 we experienced that SOCU often crashes if we do 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 are used in other optimizers [97, 141, 19], can be computed fast, in vectorized form, and robustly. They lack however the model uncertainty. The motivation for this chapter is driven by the following research question:

Q7.1 Can we determine an estimation for the model uncertainty for any arbitrary kernel, e.g. cubic RBF, augmented cubic RBF, ...?
In this chapter we exploit the analogies between RBF and GP to measure the model uncertainty for any arbitrary kernel. Furthermore, we investigate if we can apply the determined model uncertainty to an EI-based optimization scheme like SOCU and 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, ...)

7.3 Related Work

Radial basis function (RBF) interpolation was first developed by Hardy in 1971 for cartography purposes [79]. This technique was designed to model hills and valleys with a reasonably high local and global accuracy. Shortly after Hardy introduced the Multiquadric (MQ) RBFs in [79], he and many other researcher extended his work by applying and investigating RBF interpolations in various scientific disciplines [119, 170, 78].

RBF interpolation approximates a function by fitting a linear weighted combination of radial basis functions. Whilst many researchers were associated with investigating different effective radial basis functions like Gaussian, cubic, thin plate spline [50, 158], other researchers were working on the mathematical foundation and proof of nonsingularity of RBFs [112, 65].

Kriging is named after a South African statistician who made use of Gaussian stochastic processes to model the gold distribution in South Africa. D. G. Krige developed an algorithm in his master thesis to estimate a model and a measure of uncertainty for the model [103] based on the limited sampled information.

The mathematical foundation of Kriging was published about 10 years later by Matheron [110]. Bayesian optimization [117] and efficient global optimization (EGO) [90] based on the expected improvement concept are applications of Kriging in the field of black-box optimization.

Forrester et al. [63] briefly discussed the ties between GPs and RBF interpolation. They show that in some cases the mean value prediction by GP (without the model uncertainty) is equivalent to the RBF's prediction. Additionally, Emmerich in his PhD thesis [54] compares Kriging and RBF network (RBFN). Emmerich also shows similarities between RBF and GPs, however, the RBF interpolation does not provide any model uncertainty measure. He concludes that RBFs are more efficient because they skip the parameter tuning step which is often done for GPs by means of maximum likelihood estimation.

Table 7.1: Commonly used kernel functions for GP and radial basis functions for RBF interpolation. $r = ||\vec{x}_i - \vec{x}_j||$.

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

7.4 Methods

7.4.1 Gaussian Process Modeling

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 $\mathbf{X} = \{\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_n\}$ and $f_i = f(\vec{x}_i) = y_i$. The Gaussian processes method assumes that $p(f_1, f_2, \cdots, f_n)$ belongs to a multivariate (jointly) Gaussian with a mean $\vec{\mu}$ and covariance matrix $\mathbf{\Sigma}$:

$$\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \sim N\left(\begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} & \cdots & \Sigma_{1n} \\ \Sigma_{21} & \Sigma_{22} & \cdots & \Sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{n1} & \Sigma_{n2} & \cdots & \Sigma_{nn} \end{bmatrix} \right) \sim N\left(\vec{\mu}, \mathbf{\Sigma}\right) \tag{7.1}$$

where $\Sigma_{ij} = \kappa(\vec{x}_i, \vec{x}_j)$. The covariance matrix contains the dependencies and similarities of random variables, in this case the $f(\vec{x}_i)$. 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. The similarity function is often symmetric and a dependent on the distance between every two points $\kappa(\vec{x}_i, \vec{x}_j) = \kappa(||\vec{x}_i - \vec{x}_j||)$. Table 7.1 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

$$\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \\ f_* \end{bmatrix} \sim N \left(\begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \\ \mu_* \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} & \cdots & \Sigma_{1n} & \Sigma_{1*} \\ \Sigma_{21} & \Sigma_{22} & \cdots & \Sigma_{2n} & \Sigma_{1*} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \Sigma_{n1} & \Sigma_{n2} & \cdots & \Sigma_{nn} & \Sigma_{n*} \\ \Sigma_{*1} & \Sigma_{*2} & \cdots & \Sigma_{*n} & \Sigma_{**} \end{bmatrix} \right) \tag{7.2}$$

which can be summarized as follows

$$\begin{bmatrix} \vec{f} \\ f_* \end{bmatrix} \sim N \left(\begin{bmatrix} \vec{\mu} \\ \mu_* \end{bmatrix}, \begin{bmatrix} \mathbf{K} & \vec{K}_* \\ \vec{K}_*^T & K_{**} \end{bmatrix} \right), \tag{7.3}$$

where $\mathbf{K} = \Sigma$ is the $n \times n$ matrix of Eq. (7.1), $\vec{K}_* = \kappa(\mathbf{X}, \vec{x}_*)$ is an $n \times 1$ vector and $K_{**} = \kappa(\vec{x}_*, \vec{x}_*)$ is a scalar and $\vec{f} = \{f_1, f_2, \cdots, f_n\}$ is an $n \times 1$ vector. $\mathbf{X} = \{\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_n\}$ is the matrix of the data points. We look for the probability of f_* when the data $\mathbf{X}$, their corresponding values $\vec{f}$ and the new point $\vec{x}_*$ are given. Based on the conditional probability theorem [121] and conducting long lines of heavy algebra we can determine a distribution at every new point as follows:

$$p(f_*|\vec{x}_*, \mathbf{X}, \vec{f}) = N(\vec{K}_*^T \mathbf{K}^{-1} \vec{f}, K_{**} - \vec{K}_*^T \mathbf{K}^{-1} \vec{K}_*) = N(\mu_*, \Sigma_*), \tag{7.4}$$

where μ_* and Σ_* can be interpreted as the mean and the uncertainty of the Gaussian processes model, respectively. Fig. 7.1 shows an example of a Gaussian process model with a Gauss kernel function. Fig. 7.1-left illustrates several samples from the prior $p(\vec{f}|\mathbf{X})$ and Fig. 7.1-right shows samples from the posterior function $p(f_*|\mathbf{X}_*, \mathbf{X}, \vec{f})$. The dark black curve is the mean μ_* and the shaded area is showing the 90% confidence interval.

We can rewrite the prediction of the mean value as follows:

$$\begin{aligned} f_* &= \vec{K}_*^T \mathbf{K}^{-1} \vec{f} \\ f_* &= \vec{K}_*^T \vec{\theta} \\ f_* &= \sum_{i=1}^{n} \theta_i \kappa(\vec{x}_i, \vec{x}_*), \end{aligned} \tag{7.5}$$

Eq. (7.5) shows that the mean of the GP model can be determined as a linear summation of weighted kernel functions.

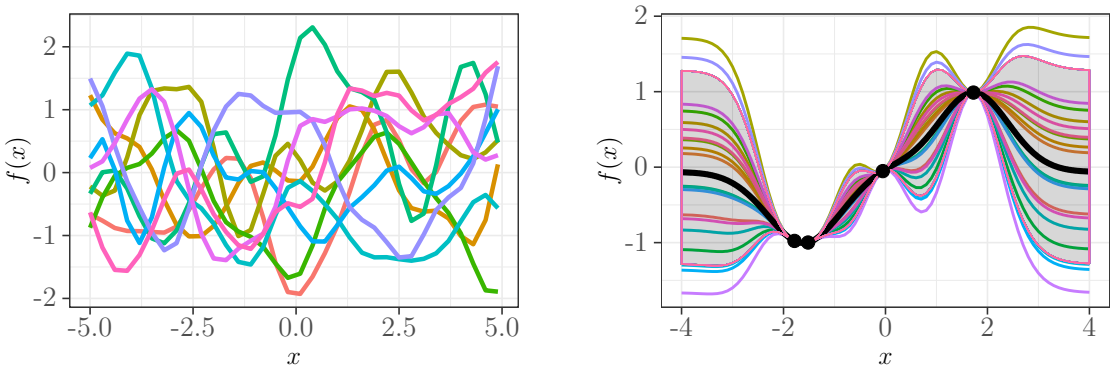

Figure 7.1: Left: prior function distribution using squared exponential kernel. Right: posterior function distribution given the evaluated points using squared exponential kernel.

The uncertainty term in Eq. (7.4) can be rewritten as:

$$\Sigma_* \quad = \quad -\vec{K}_*^T \mathbf{K}^{-1} \vec{K}_* + K_{**} \tag{7.6}$$

It is important to mention that the uncertainty of the model estimated by Eq. (7.6) is only a function of the distribution of points in the input space. Fig. 7.1 illustrates that the model uncertainty goes to zero at the given points and it becomes larger as the distance from the evaluated points increases.

The GP modeling technique is easily extendable for fitting noisy data. Assuming the presence of noise in the data, Eq. (7.5) changes to Eq.(7.7) as it is determined in [121]. This change also known as regularization trick introduces one hyperparameter σ_y.

$$f_* \quad = \quad \vec{K}_*^T \mathbf{K}_y^{-1} \vec{f}, \text{where} \quad \mathbf{K}_y \quad = \quad \mathbf{K} + \sigma_y^2 \mathbf{I} \tag{7.7}$$

The correct choice of hyperparameters for GPs, including the variance σ_f, the noise variance σ_y and the shape parameter α is a problem-dependent task. The common practice to estimate the hyperparameters for GPs is to select a set of parameters which maximizes the likelihood of $p(\vec{f}|\mathbf{X}, \sigma_f, \sigma_y, \alpha)$. To do so, one should maximize Eq. (7.8).

$$\log p(\vec{f}|\mathbf{X}, \sigma_f, \sigma_y, \alpha) = \log N(\vec{f}|0, \mathbf{K}_y) = -\frac{1}{2}\vec{f}\mathbf{K}_y^{-1}\vec{f} - \frac{1}{2}\log|\mathbf{K}_y| - \frac{N}{2}\log(2\pi) \tag{7.8}$$

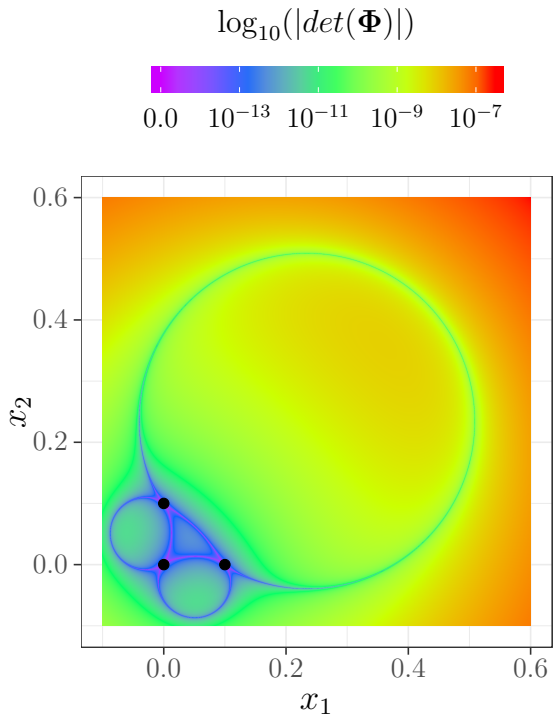

Figure 7.2: Showcasing possible ill-conditioned or singular $\mathbf{\Phi}$ for cubic RBF. The color gradient shows the determinant of the $\mathbf{\Phi}$ matrix at any point $\vec{x} = \{x_1, x_2\}$, when the matrix is built based on four points including the three black points and any arbitrary $\vec{x}$. The thin purple curved lines are where the determinant becomes exactly zero.

The likelihood function in Eq. (7.8) is not always a convex function and can have multiple local optima. Maximum likelihood estimation (MLE) can suffer from getting stuck in such a local optimum. An example for such a scenario is illustrated by Rasmussen and Williams [138].

7.4.2 Radial Basis Function Interpolation

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 $\vec{x}$ from some fixed centers $\vec{c}$. When RBFs are used for interpolation tasks, it is common that all the evaluated points $\vec{x}_i$

are considered as centers.

$$\hat{f}(\vec{x}) = \sum_{i=1}^{n} \theta_i \varphi((||\vec{x} - \vec{x}_i||) \tag{7.9}$$

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

$$[\mathbf{\Phi}] \left[\vec{\theta}\right] = [\vec{f}], \tag{7.10}$$

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

$$\vec{\theta} = \mathbf{\Phi}^{-1} \vec{f} \tag{7.11}$$

Now that we have the weight vector $\vec{\theta}$, we can compute f_* at any point $\vec{x_*}$:

$$f_* = \sum_{i=1}^{n} \theta_i \varphi(||\vec{x_*} - \vec{x}_i||) = \vec{\Phi}_*^T \vec{\theta} = \vec{\Phi}_*^T \mathbf{\Phi}^{-1} \vec{f} \tag{7.12}$$

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

Augmented RBF

It is proven that $\mathbf{\Phi}$ in Eq. 7.10 is not guaranteed to be positive definite for several radial basis functions [112] like cubic r^3. Fig. 7.2 showcases a possible scenario where the determinant of $\mathbf{\Phi}$ can be equal to zero. As shown in Fig. 7.2 the area with exact zero determinant is very small.

In order to assure that the Eq. 7.10 has a unique solution with all radial basis functions, Micchelli introduced augmented RBFs [112]. Augmented RBFs are actually RBF functions with a polynomial tail.

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{n} \theta_i \varphi(||\vec{x} - \vec{x}_i||) + p(\vec{x}), \quad \vec{x} \in \mathbb{R}^d, \tag{7.13}$$

where $p(\vec{x}) = \mu_0 + \vec{\mu_1}\, \vec{x} + \vec{\mu_2}\, \vec{x}^2 \cdots + \vec{\mu_k}\, \vec{x}^k$ 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:

$$\begin{bmatrix} \mathbf{\Phi} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{0}_{(kd+1)\times(kd+1)} \end{bmatrix} \begin{bmatrix} \vec{\theta} \\ \vec{\mu}\,' \end{bmatrix} = \begin{bmatrix} \vec{f} \\ 0_{(kd+1)} \end{bmatrix} \tag{7.14}$$

Here, $\mathbf{P} \in \mathbb{R}^{n \times (kd+1)}$ is a matrix with $(1, \vec{x}_{(i)}, \vec{x}^2_{(i)})$ 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.

7.4.3 GP vs. RBF

Although GP and RBF interpolation have two very different origins, the comparison of Eq. (7.12) and Eq. (7.5) shows that the mean of GP is identical to the RBF result, if the kernel function κ is identified with the basis function φ. In addition to the prediction of the mean, GPs determine a prediction of the model uncertainty Σ_* (Eq. (7.6)). 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.

$$\Sigma_{rbf} = \varphi(||\vec{x}_* - \vec{x}_*||) - \vec{\Phi}_*^T \mathbf{\Phi}^{-1} \vec{\Phi}_*, \tag{7.15}$$

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

Fig. 7.3 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 shown in Fig. 7.3 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 Kriging implementation in R from the DiceKriging package tunes the kernel parameter(s) based on the maximum likelihood estimation (MLE) approach which has a computational complexity of approximately $\mathcal{O}(\frac{1}{3}n^3 + \frac{1}{2}dn^2)$.

The kernel parameters for the RBF interpolation are often set manually. In Ch. 8, an online selection algorithm for choosing the best kernel type and parameters during an optimization process is suggested. In this chapter we use a parameter-free radial basis function.

The plots in the last column of Fig. 7.3 are generated with 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. 7.3.

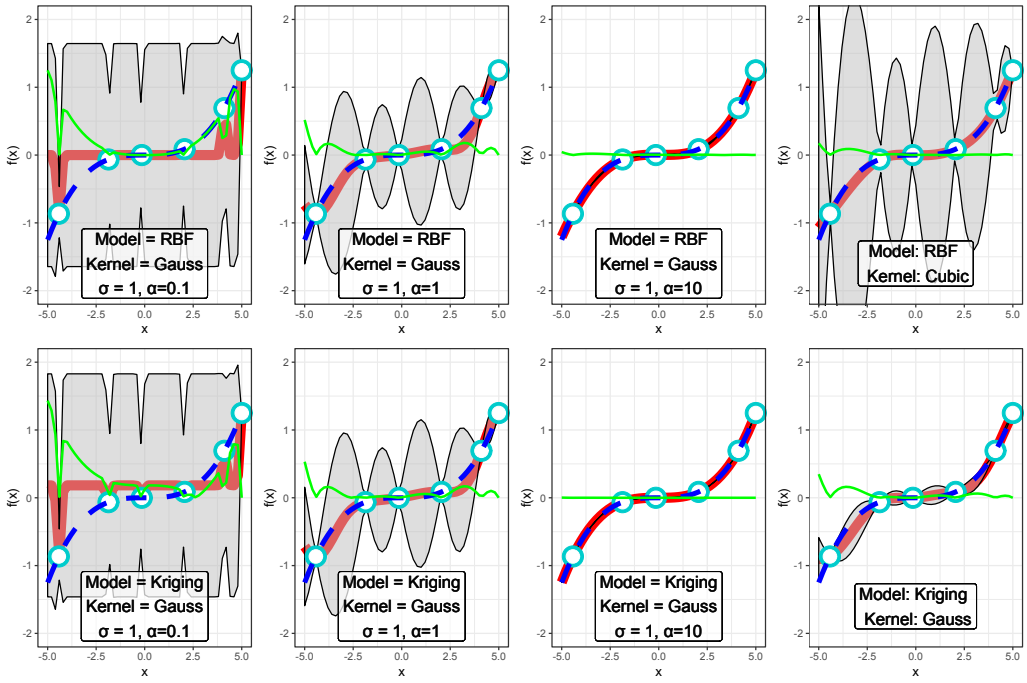

Figure 7.3: Comparing RBF and GP from the DiceKriging package in R. The examples in the first row are all generated by RBF and in the second row with GP. The dashed blue curve $f = x^3$ is the target curve to be modeled. The circles are the evaluated points. The thick red curve is the delivered model. The thin green 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. The plots in the last column are generated by the default configuration of RBF and GP.

7.5 Experimental Setup

In this chapter we compare two versions of the SOCU optimization framework namely SOCU-Kriging and SOCU-RBF. The SOCU algorithm developed in [19] and also described in Ch. 5 is a surrogate-assisted constrained optimization method. SOCU-Kriging uses the Kriging model from the R packages DiceKriging and DiceOptim, while SOCU-RBF uses our own implementation of RBF interpolation. The first major difference between the two versions of SOCU is the choice of the kernel functions. SOCU-Kriging uses matern3-2 which is a relatively stable kernel function according to our initial experiments. SOCU-RBF makes use of an augmented cubic

Table 7.2: 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. Here, we only selected 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

basis function with a second order polynomial tail 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 important difference between SOCU-RBF and SOCU-Kriging is the numerical approach used by them for the required matrix inversion. The DiceKriging package [149] uses Cholesky decomposition but we found singular value decomposition used for RBF to be a more stable approach. In our experiments described in [19] 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 chapter we present SOCU-Kriging results with two regularization factors of $\sigma_y = \{10^{-3}, 10^{-4}\}$. 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-RBF 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. 7.3.

We apply SOCU-Kriging and SOCU-RBF to the subset of G-problems having only inequality constraints (see Table 7.2). G02 is a scalable problem in its dimension d. We use here $d = 2$. For each algorithm we run 10 independent trials with

Table 7.3: 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

different initial population fo size $3 \cdot d$. In order to optimize EI_{mod} we use Generalized Simulated Annealing (R package GenSA).

7.6 Results and Discussion

Performance on G-problems

Fig. 7.4 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.

Computational Time

Fig. 7.5 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.

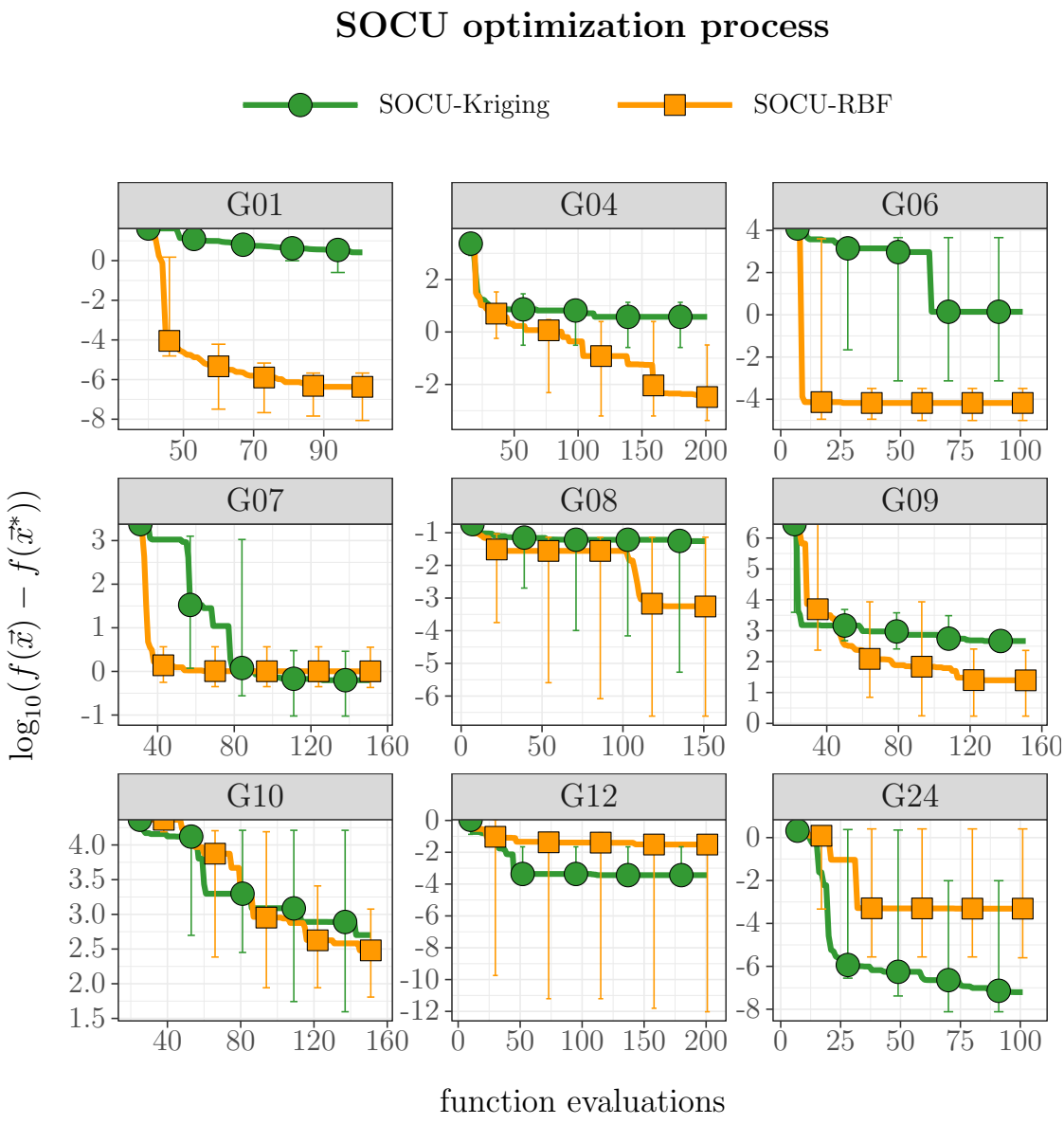

Figure 7.4: Comparing optimization performance of SOCU-Kriging and SOCU-RBF on G-problems. The curves are showing the median error out of 10 trials. The error bars indicate the best and the worst results out of 10 trials.

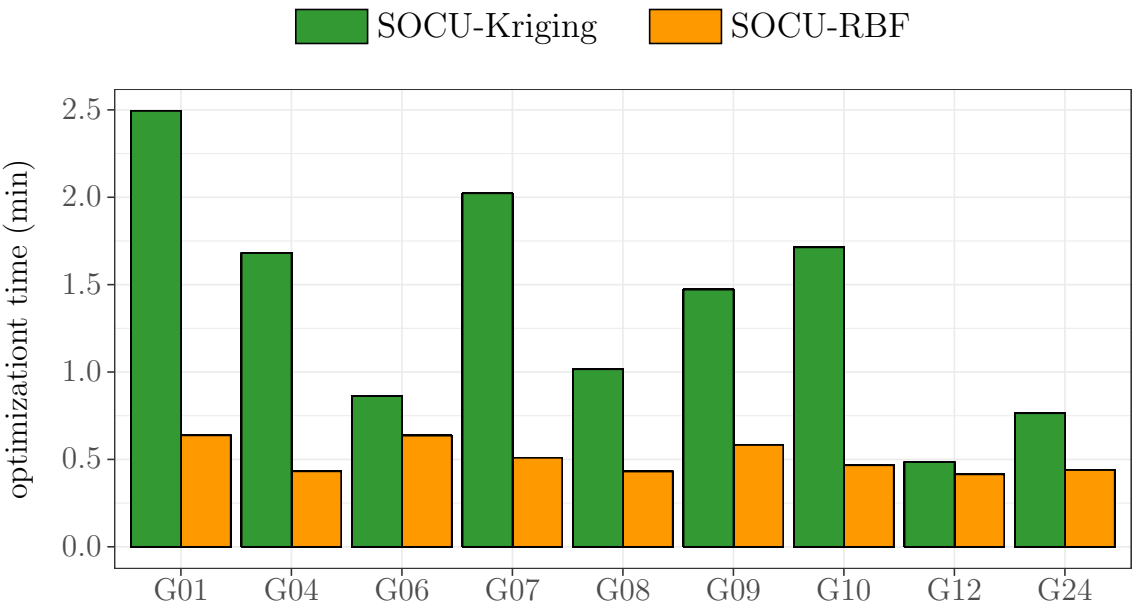

Figure 7.5: Average computational time in minutes, required by SOCU-Kriging and SOCU-RBF to run one iteration of each G-problem.

Noise Variance

We have already shown that SOCU-RBF outperforms SOCU-Kriging in Fig. 7.4. The SOCU-Kriging algorithm used for generating Fig. 7.4 has a non-zero noise variance $\sigma_y = 10^{-3}$. 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 the noise variance we applied the SOCU-Kriging framework to all G-problems in Table. 7.2 with two different noise variance values $\sigma_y = 10^{-3}$ and $\sigma_y = 10^{-4}$. SOCU-Kriging with the smaller noise variance $\sigma_y = 10^{-4}$ crashed on the G06 problem. Fig. 7.6 compares the SOCU-Kriging optimization results with two different noise variance values for all problems excluding G06.

Fig. 7.6 indicates that a smaller noise variance value can lead to slightly better optimization results. For all the problems illustrated in Fig. 7.6 except G07, SOCU-

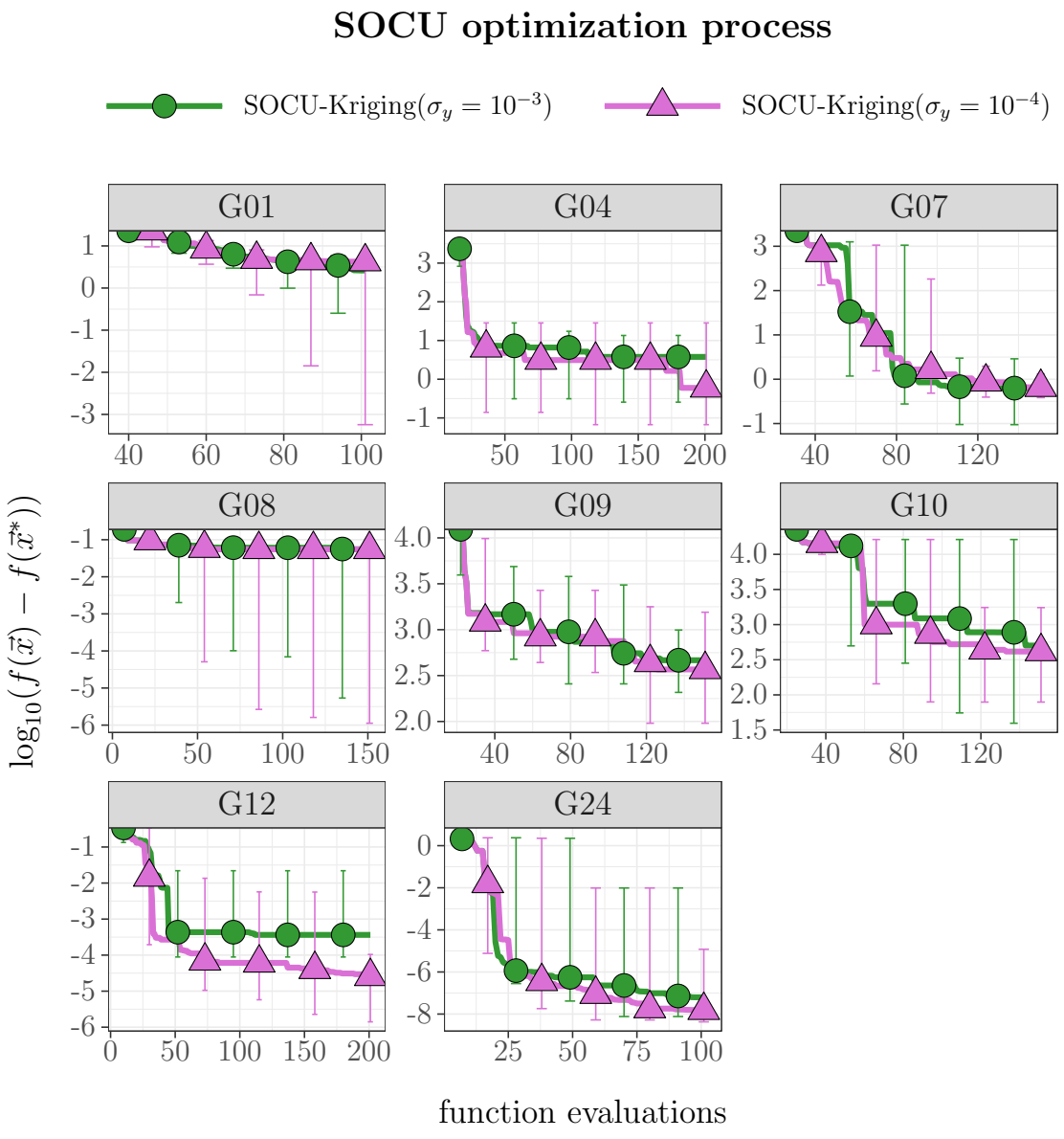

Figure 7.6: Comparing the results of SOCU-Kriging with two different noise variance values σ_y. The curves are showing the median optimization error of the 10 independent trials for each algorithm. The error bars are indicating the best and worst case results.

Kriging $\sigma_y = 10^{-4}$) 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.

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. 7.4 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. 7.7. As it is shown in Fig. 7.7, 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. 7.7, we can also observe that SOCU-Kriging with a smaller noise variance $\sigma_y = 10^{-4}$ has slightly smaller approximation error in the last iterations.

7.7 Conclusion

In this chapter we explored 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. 7.4. 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 any of our experiments).

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.

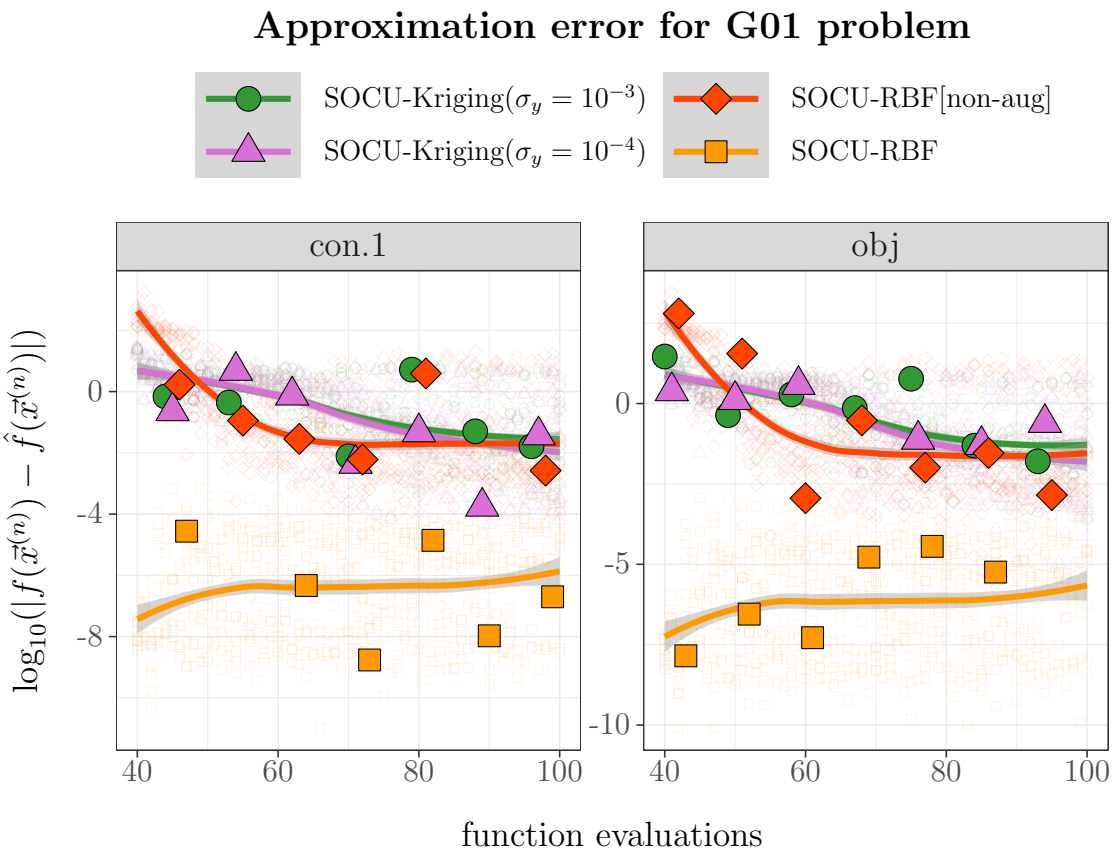

Figure 7.7: Approximation error for the objective and constraint functions of the G01 problem. G01 has 9 constraints but due to lack of space we just show the approximation error of the objective (a quadratic function) and one of its constraint functions (a linear function), see Appendix A. Since all 9 constraints are of the same type, their approximation error curves look similar. The approximation error is the error in predicting at the new infill point before this point is added to the population.