Supplementary Material: Noisy-Input Entropy Search for Efficient Robust Bayesian Optimization

Lukas P. Fröhlich[1,2] **Edgar D. Klenske**[1] **Julia Vinogradska**[1]

Christian Daniel[1]
[1]Bosch Center for Artificial Intelligence
Renningen, Germany

Melanie N. Zeilinger[2]
[2]ETH Zürich
Zürich, Switzerland

A Expectation Over Input Noise for Sparse Spectrum GP Samples

Consider a sampled function from a sparse spectrum Gaussian process (SSGP) of the form $\tilde{f}(\boldsymbol{x}) = \boldsymbol{a}^T \boldsymbol{\phi}_f(\boldsymbol{x})$. In this section, we solve the following integral,

$$\phi_{g,i}(\boldsymbol{x}) = \int \phi_{f,i}(\boldsymbol{x} + \boldsymbol{\xi}) p(\boldsymbol{\xi}) d\boldsymbol{\xi} \tag{1}$$

where $\phi_{f,i}(\boldsymbol{x})$ is the i-th component of $\boldsymbol{\phi}_f(\boldsymbol{x})$. $\phi_{g,i}(\boldsymbol{x})$ is the i-th component of the corresponding 'robust' sample of the form $\tilde{g}(\boldsymbol{x}) = \boldsymbol{a}^T \boldsymbol{\phi}_g(\boldsymbol{x})$. Note that the weights $\boldsymbol{a}$ are the same for both sampled functions, $\tilde{f}(\boldsymbol{x})$ and $\tilde{g}(\boldsymbol{x})$.

Eq. (1) requires the cross-correlation between function ϕ and p. Since p is a probability distribution (Gaussian in this case), it's complex conjugate is p itself and the cross-correlation theorem states that in this case the cross-correlation is equivalent to the convolution (Smith, 2007, Sec. 8.4). Thus, we can apply the convolution theorem, which states

$$(\phi_{f,i} * p)(\boldsymbol{x}) = \mathcal{F}^{-1}\left\{\mathcal{F}\left\{\phi_{f,i}\right\}\mathcal{F}\left\{p\right\}\right\},$$

or in words: a convolution in 'time' domain is the same as a multiplication in frequency domain. Before we apply this result, however, note that in the case of a separable filter window, we can apply the convolution in each dimension separately. The final integral we need to solve then becomes,

$$\int \cos(\omega_{i,k}(x_k + \xi) + \underbrace{\sum_{j \neq k} \omega_{i,j} x_j + c_i}_{b_k}) p(\xi_k) d\xi_k,$$

for $k = 1, \ldots, n$. We find the Fourier transforms of a shifted cosine with frequency $\omega_{i,k}$ and the univariate normal distribution, then multiply those and perform the inverse transform. We use the following standard Fourier transforms:

$$\mathcal{F}\left\{\cos(\omega_{i,k} x_k + b_k)\right\} = \sqrt{\frac{\pi}{2}}\left(\delta(\omega - \omega_{i,k}) + \delta(\omega + \omega_{i,k})\right)\exp\left(j\frac{b_k}{\omega_{i,k}}\omega\right),$$

and

$$\mathcal{F}\left\{\frac{1}{\sqrt{2\pi\sigma_{x,k}^2}}\exp\left(-\frac{x_k^2}{\sigma_{x,k}^2}\right)\right\} = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}\omega^2\sigma_{x,k}^2\right).$$

The inverse Fourier transform is given as

$$h(x) = \mathcal{F}^{-1}\left\{\hat{h}\right\}(x) = \int \hat{h}(\omega)\exp(j\omega x) d\omega$$

and plugging in the results from above gives

$$\begin{aligned}\phi_{g,i}(\boldsymbol{x}) &= (\phi_{f,i} * p)(\boldsymbol{x}) \\ &= \phi_{f,i}(\boldsymbol{x})\exp\Big(-\frac{1}{2}\sum_{j=1}^{d}\boldsymbol{w}_{i,j}^2\sigma_{x,j}^2\Big).\end{aligned}$$

Overall, filtering results in scaling of the basis functions.

B Details on EP-Approximation of the Conditional Predictive Distribution

We aim at finding $p(f(\boldsymbol{x})|\mathcal{D}_n, g^*)$, which is the predictive distribution for the latent function $f(\boldsymbol{x})$, i.e., the observable function, conditioned on the data $\mathcal{D}_n$ and as well as on a sample of the robust maximum value distribution $g_k^* \sim p(g^*|\mathcal{D}_n)$. We will denote all evaluated points as $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$ and the corresponding observed function values as $\boldsymbol{y} = [y_1, \ldots, y_n]$.

We start the derivation by rewriting the desired distribution as

$$\begin{aligned} p(f(\boldsymbol{x})|\mathcal{D}_n, g_k^*) &= \int p(f(\boldsymbol{x}), g(\boldsymbol{x})|\mathcal{D}_n, g_k^*) dg(\boldsymbol{x}) \\ &= \int p(f(\boldsymbol{x})|\mathcal{D}_n, g(\boldsymbol{x})) p(g(\boldsymbol{x})|\mathcal{D}_n, g_k^*) dg(\boldsymbol{x}). \quad (2) \end{aligned}$$

We compute this integral in 3 steps: First, we approximate $p(g(\boldsymbol{x})|\mathcal{D}_n, g_k^*)$ by a Gaussian distribution via expectation propagation (EP). Second, we compute $p(f(\boldsymbol{x})|g(\boldsymbol{x}), \mathcal{D}_n)$ by standard Gaussian process (GP) arithmetic. Third, we make use of the fact that the marginalization over a product of Gaussian can be computed in closed form.

Gaussian approximation to $p(g(\boldsymbol{x})|\mathcal{D}_n, g_k^*)$: We fit a Gaussian approximation to $p(g(\boldsymbol{x})|\mathcal{D}_n, g_k^*)$ as this enables us to compute the integral in Eq. (2) in closed form. This approximation itself is done in three steps, following along the lines of Hoffman and Ghahramani (2015) where they approximate $p(f(\boldsymbol{x})|\mathcal{D}_n, f^*)$. The key idea is that conditioning on the robust maximum value sample implies the constraint that $g(\boldsymbol{x}) \leq g^*$.

1. In a first step, we only incorporate the constraint $g(\boldsymbol{x}_i) \leq g_k^* \;\; \forall \;\; \boldsymbol{x}_i \in \mathcal{D}_n$ such that

   $$p(\boldsymbol{g}|\mathcal{D}_n, g_k^*) \propto p(\boldsymbol{g}|\mathcal{D}_n) \prod_{i=1}^{n} \mathbb{1}_{\{\boldsymbol{x}_i | g(\boldsymbol{x}_i) \leq g_k^*\}},$$

   where $\boldsymbol{g} = [g_1, \dots, g_n]$ denotes the latent function values of g evaluated at $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ and $\mathbb{1}_{\{\cdot\}}$ is the indicator function. The above distribution constitutes a multi-variate truncated normal distribution. There is no analytical solution for its moments. One common strategy is to approximate the moments using EP Herbrich (2005). In practice, EP converges quickly for this distribution. We denote the outcome as

   $$p(\boldsymbol{g}|\mathcal{D}_n, g_k^*) \approx \mathcal{N}(\boldsymbol{g}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1).$$

2. The next step is getting a predictive distribution from the (constrained) latent function values:

   $$p_0(g(\boldsymbol{x})|\mathcal{D}_n, g_k^*) = \int p(\boldsymbol{g}|\mathcal{D}_n, g_k^*) p(g(\boldsymbol{x})|\mathcal{D}_n, \boldsymbol{g}) d\boldsymbol{g}. \quad (3)$$

   For the first term we use the Gaussian approximation of the previous step and the second term is given by standard GP arithmetic:

   $$p(g(\boldsymbol{x})|\mathcal{D}_n, \boldsymbol{g}) = \mathcal{N}(g(\boldsymbol{x})|\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g),$$

   with

   $$\begin{aligned} \boldsymbol{\mu}_g &= [k_g(\boldsymbol{x}, X), k_{gf}(\boldsymbol{x}, X)] \\ &\qquad \begin{bmatrix} k_g(X,X) & k_{gf}(X,X) \\ k_{fg}(X,X) & k_f(X,X) + \sigma_\epsilon^2 \boldsymbol{I} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{y} \end{bmatrix} \\ &= [\boldsymbol{B}_1, \boldsymbol{B}_2] \begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{y} \end{bmatrix}, \end{aligned}$$

   and

   $$\boldsymbol{\Sigma}_g = k_g(\boldsymbol{x}, \boldsymbol{x}) - [\boldsymbol{B}_1, \boldsymbol{B}_2] \begin{bmatrix} k_g(\boldsymbol{x}, X) \\ k_{gf}(\boldsymbol{x}, X) \end{bmatrix}.$$

   Note that the integral in Eq. (3) is the marginalization over a product Gaussians where the mean of $p(g(\boldsymbol{x}|\mathcal{D}_n, \boldsymbol{g})$ is an affine transformation of $\boldsymbol{g}$. Integrals of this form occur often when dealing with Gaussian distributions, e.g., in the context of Kalman filtering, and can be solved analytically (see e.g., Schön and Lindsten (2011, Corollary 1)). We obtain

   $$p_0(g(\boldsymbol{x})|\mathcal{D}_n, g_k^*) \approx \mathcal{N}(g(\boldsymbol{x})|m_0, v_0)),$$

   with

   $$\begin{aligned} m_0(\boldsymbol{x}) &= \boldsymbol{B}_1 \boldsymbol{\mu}_1 + \boldsymbol{B}_2 \boldsymbol{y} \\ v_0(\boldsymbol{x}) &= \boldsymbol{\Sigma}_g + \boldsymbol{B}_1 \boldsymbol{\Sigma}_1 \boldsymbol{B}_1^T. \end{aligned}$$

3. Recall that in the first step we only enforced the constraints on the function values at the data points. Thus, we still need to integrate the constraint $g(\boldsymbol{x}) \leq g_k^* \;\; \forall \;\; \boldsymbol{x} \in \mathcal{X}$

   $$p(g(\boldsymbol{x})|\mathcal{D}_n, g_k^*) \propto \mathcal{N}(m_0, v_0) \mathbb{1}_{\{\boldsymbol{x}|g(\boldsymbol{x}) \leq g_k^*\}},$$

   where we again utilize a Gaussian approximation to this distribution. However, this is only a univariate truncated normal distribution and we can easily find the corresponding moments, such that

   $$p(g(\boldsymbol{x})|\mathcal{D}_n, g_k^*) \approx \mathcal{N}(g(\boldsymbol{x})|\hat{m}(\boldsymbol{x}), \hat{v}(\boldsymbol{x})), \quad (4)$$

   with mean and variance given as

   $$\begin{aligned} \hat{m}(\boldsymbol{x}) &= m_0(\boldsymbol{x}) - \sqrt{v_0(\boldsymbol{x})} r, \\ \hat{v}(\boldsymbol{x}) &= v_0(\boldsymbol{x}) - v_0(\boldsymbol{x}) r (r + \alpha), \end{aligned}$$

   where $\alpha = (g_k^* - m_0(\boldsymbol{x}))/\sqrt{v_0(\boldsymbol{x})}$ and $r = \varphi(\alpha)/\Phi(\alpha)$. As usual, $\varphi(\cdot)$ and $\Phi(\cdot)$ denote the PDF and CDF of the standard normal distribution, respectively.

GP arithmetic to find $p(f(\boldsymbol{x})|g(\boldsymbol{x}), \boldsymbol{y})$: Starting with the joint distribution of all involved variables

$$\begin{bmatrix} f(\boldsymbol{x}) \\ \boldsymbol{y} \\ g(\boldsymbol{x}) \end{bmatrix} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}),$$

$$\boldsymbol{K} = \begin{bmatrix} k_f(\boldsymbol{x}, \boldsymbol{x}) & k_f(\boldsymbol{x}, X) & k_{fg}(\boldsymbol{x}, \boldsymbol{x}) \\ k_f(X, \boldsymbol{x}) & k_f(X, X) + \sigma_n^2 I & k_{fg}(X, \boldsymbol{x}) \\ k_{gf}(\boldsymbol{x}, \boldsymbol{x}) & k_{gf}(\boldsymbol{x}, X) & k_g(\boldsymbol{x}, \boldsymbol{x}) \end{bmatrix},$$

we introduce $\boldsymbol{z} = [\boldsymbol{y}, g(\boldsymbol{x})]^T$ for notational convenience and rewrite the joint distribution as

$$\begin{bmatrix} f(\boldsymbol{x}) \\ \boldsymbol{z} \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} k_f(\boldsymbol{x},\boldsymbol{x}) & k_z(\boldsymbol{x},X)^T \\ k_z(\boldsymbol{x},X) & K_z(\boldsymbol{x},X) \end{bmatrix}\right). \tag{5}$$

Conditioning then gives

$$p(f(\boldsymbol{x})|\boldsymbol{z}) = \mathcal{N}\left(f(\boldsymbol{x})|\boldsymbol{\mu}_4, \boldsymbol{\Sigma}_4\right) \tag{6}$$
$$\boldsymbol{\mu}_4 = k_z(\boldsymbol{x},X)^T K_z(\boldsymbol{x},X)^{-1}\boldsymbol{z}$$
$$\boldsymbol{\Sigma}_4 = k_f(\boldsymbol{x},\boldsymbol{x}) - k_z(\boldsymbol{x},X)^T K_z(\boldsymbol{x},X)^{-1} k_z(\boldsymbol{x},X).$$

Let's rewrite the mean of Eq. (6) as follows

$$\boldsymbol{\mu}_4 = \underbrace{k_z(\boldsymbol{x},X)^T K_z(\boldsymbol{x},X)^{-1}}_{=[\boldsymbol{A}_1,\boldsymbol{A}_2]}\boldsymbol{z} = \boldsymbol{A}_1\boldsymbol{y} + \boldsymbol{A}_2 g(\boldsymbol{x}), \tag{7}$$

with $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ being of appropriate dimensions.

Solve the integral: Now that we have the explicit forms of the distributions in the integral, we make use of the results (4) and (6),

$$p(f(\boldsymbol{x})|\mathcal{D}_n, g_k^*) \tag{8}$$
$$= \int p(f(\boldsymbol{x})|\mathcal{D}_n, g(\boldsymbol{x}))p(g(\boldsymbol{x})|\mathcal{D}_n, g_k^*)dg(\boldsymbol{x}) \tag{9}$$
$$= \int \mathcal{N}\left(f(\boldsymbol{x})|\boldsymbol{A}_1\boldsymbol{y} + \boldsymbol{A}_2 g(\boldsymbol{x}), \boldsymbol{\Sigma}_4\right)$$
$$\mathcal{N}\left(g(\boldsymbol{x})|\hat{m}(\boldsymbol{x}), \hat{v}(\boldsymbol{x})\right)dg(\boldsymbol{x}). \tag{10}$$

This integral has the same form as Eq. (3) and can be solved in closed form as well (see (Schön and Lindsten, 2011, Corollary 1)). The final result is

$$p(f(\boldsymbol{x})|\mathcal{D}_n, g_k^*) \approx \mathcal{N}\left(f(\boldsymbol{x})|\tilde{m}(\boldsymbol{x}), \tilde{v}(\boldsymbol{x})\right) \tag{11}$$
$$\tilde{m}(\boldsymbol{x}) = \boldsymbol{A}_1\boldsymbol{y} + \boldsymbol{A}_2\hat{m}(\boldsymbol{x}) \tag{12}$$
$$\tilde{v}(\boldsymbol{x}) = \boldsymbol{\Sigma}_4 + \boldsymbol{A}_2\hat{v}(\boldsymbol{x})\boldsymbol{A}_2^T. \tag{13}$$

C Additional Results

C.1 Results for Hartmann (6-dim.)

In Sec. 4 of the main paper, we provide a comparison on several benchmark functions up to three dimensions in terms of the inference regret, $r_n = |g(\mathbf{x}_n^*) - g^*|$. For computing the regret, one requires the 'true' robust optimum value g*. This value is generally not known and has to be found numerically. In practice, we use the FFT over discrete signals to approximate the expectation in Equation (1). For the 3-dimensional Hartmann function, we use $n_{\text{FFT}} = 101$ evaluation points in each dimension to achieve high accuracy. However, in 6 dimensions this is computationally infeasible for the required accuracy. Thus, we compare the different acquisition functions just in terms of the estimated optimal robust value $g(\mathbf{x}_n^*)$, see Fig. 1. The input noise was set to $\boldsymbol{\Sigma}_x = 0.1^2\boldsymbol{I}$.

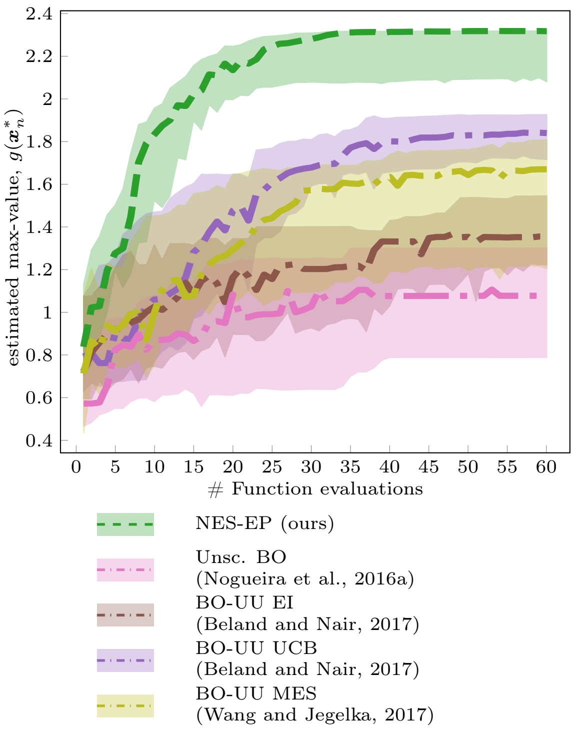


Figure 1: Estimated robust max-value $g(\boldsymbol{x}_n^*)$ for the 6-dimensional Hartmann function. We present the median (lines) and 25/75th percentiles (shaded areas) across 20 independent runs with 10 randomly sampled initial points.

C.2 Comparison of Computation Times

Table 1: Average compute time per BO iteration of different acquisition functions as needed for the within-model comparison. We report the mean (std) across the 50 different function samples. All units are in seconds. Timing experiments were run on an Intel Xeon CPU E5-1620 v4@3.50GHz.

Acquisition function	time [sec]
NES-RS (ours)	5.39 (0.23)
NES-EP (ours)	1.90 (0.60)
BO-UU UCB (Beland and Nair, 2017)	0.06 (0.03)
BO-UU EI (Beland and Nair, 2017)	0.71 (0.33)
Unsc. BO (Nogueira et al., 2016a)	0.15 (0.09)
Standard BO EI	0.07 (0.03)

C.3 Number of Max-Value Samples

In Section 3.1 we discuss how to approximate the expectation over robust maximum values by Monte Carlo sampling. Here, we explain the exact sampling procedure and subsequently present results of a within-model

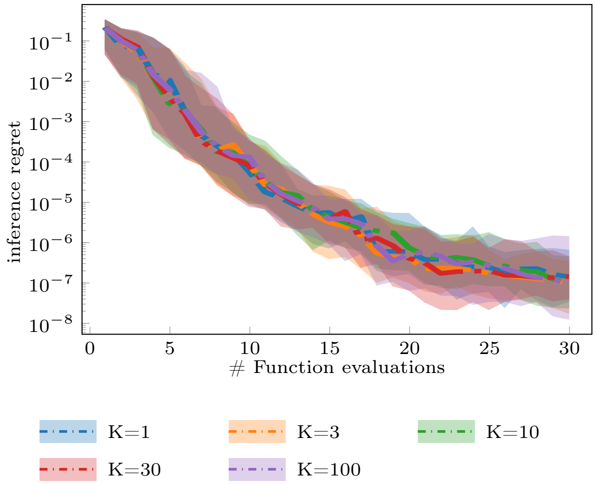


Figure 2: Within-model comparison in terms of the inference regret $r_n = |g(\boldsymbol{x}_n^*) - g^*|$ for different values of the hyperparameter K, i.e., the number of Monte-Carlo samples to approximate the expectation over robust max-values. As there is no significant difference in the performance, we used $K = 1$ for all experiments in the paper due to the lower computational cost.

comparison that investigates the effect of the number of robust max-value samples K on the final result.

Sampling Max-Values Note that the computation of the acquisition function scales linearly with the number K of Monte-Carlo samples. However, sampling the robust max-values only needs to be done once per Bayesian optimization (BO) iteration, while the acquisition function requires many evaluations during one BO iteration. Thus, it is advantageous to use as few Monte-Carlo samples as possible. The exact sampling procedure for K robust max-value samples is given as follows:

1. Sample 100 robust max-values as described in Section 3.1,
2. Create a regular grid between the 25th and 75th percentile with K points,
3. Draw the robust max-values from the sample distribution (step 1) corresponding to the percentiles of the regular grid (step 2).

The benefit of this procedure is that it makes the estimate of the expectation more robust w.r.t. the number of samples used.

Within-Model Comparison To investigate the effect of the number of Monte-Carlo samples K on the final performance, we perform a within-model comparison for NES-EP with $K = \{1, 3, 10, 30, 100\}$ samples. Results are presented in Fig. 2. Note that the performance is independent of the number of samples used to approximate the expectation. Thus, for the purpose of computational efficiency we use $K = 1$ for all experiments in the paper.

C.4 Unscented BO: Hyperparameter κ

The unscented transformation (Julier and Uhlmann, 2004) used for unscented BO (Nogueira et al., 2016a) is based on a weighted sum:

$$\bar{\boldsymbol{x}} = \mathbb{E}_{\boldsymbol{x}}\left[f(\boldsymbol{x})\right] \approx \sum_{i=0}^{2d} \omega^{(i)} f(\boldsymbol{x}^{(i)}), \tag{14}$$

with $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{x}|\boldsymbol{x}^0, \boldsymbol{\Sigma}_x)$. The so-called sigma points $\boldsymbol{x}^{(i)}$ are computed as

$$\begin{aligned} \boldsymbol{x}_+^{(i)} &= \boldsymbol{x}^0 + \left(\sqrt{(d+\kappa)\boldsymbol{\Sigma}_x}\right)_i, \quad \forall i = 1, \ldots, d \\ \boldsymbol{x}_-^{(i)} &= \boldsymbol{x}^0 - \left(\sqrt{(d+\kappa)\boldsymbol{\Sigma}_x}\right)_i, \quad \forall i = 1, \ldots, d, \end{aligned} \tag{15}$$

where $(\sqrt{\cdot})_i$ is the i-th column of the (elementwise) square root of the corresponding matrix. The weights $\omega^{(i)}$ to the corresponding sigma points are given by

$$\begin{aligned} \omega^0 &= \frac{k}{d+\kappa}, \\ \omega_+^{(i)} = \omega_-^{(i)} &= \frac{1}{2(d+\kappa)}, \quad \forall i = 1, \ldots, d. \end{aligned} \tag{16}$$

In the corresponding tech-report (Nogueira et al., 2016b) to the original paper (Nogueira et al., 2016a), the authors discuss the choice of optimal values for the hyperparameter k and suggest $\kappa = 0.0$ or $\kappa = -3.0$. For negative (integer) values of k, however, Eq. (16) leads to a division by zero if $d = -\kappa$. Thus, we decided against $\kappa = -3.0$ to be consistent across all experiments and objective functions. To find the best (non-negative) value for κ we performed a within-model comparison with different values for κ in the range between 0.0 and 2.0. Results are presented in Fig. 3. We found that for $\kappa = 1.0$, unscented BO showed the best performance and consequently also used $\kappa = 1.0$ for all experiments in the paper.

C.5 Synthetic Benchmark Functions - Distance to Robust Optimum

In the main part of this paper, we compare all methods with respect to the inference regret $r_n = |g(\boldsymbol{x}_n^*) - g^*|$. Depending on the objective's scale, the inference regret may be small although an entirely different optimum is found. Here, we present the results in terms of distance to the optimum $\|\boldsymbol{x}_n^* - \boldsymbol{x}^*\|$. See Sec. 4.1 for details on the objective functions and the evaluated methods.

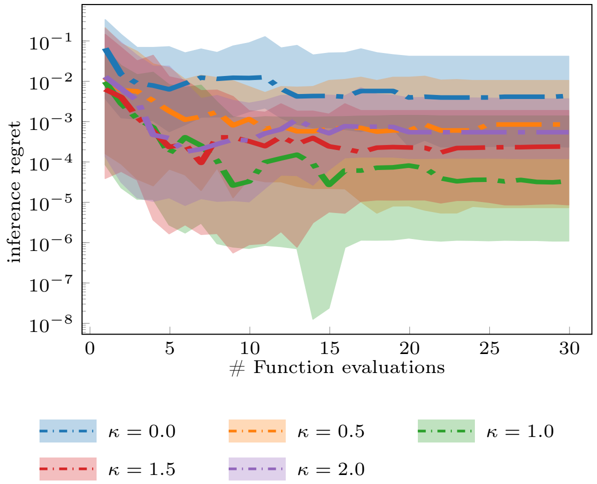


Figure 3: Within-model comparison in terms of the inference regret $r_n = |g(\boldsymbol{x}_n^*) - g^*|$ for different values of the hyperparameter K, i.e., the number of Monte-Carlo samples to approximate the expectation over robust max-values. As there is no significant difference in the performance, we used $K = 1$ for all experiments in the paper due to the lower computational cost.

D Synthetic Objective Functions

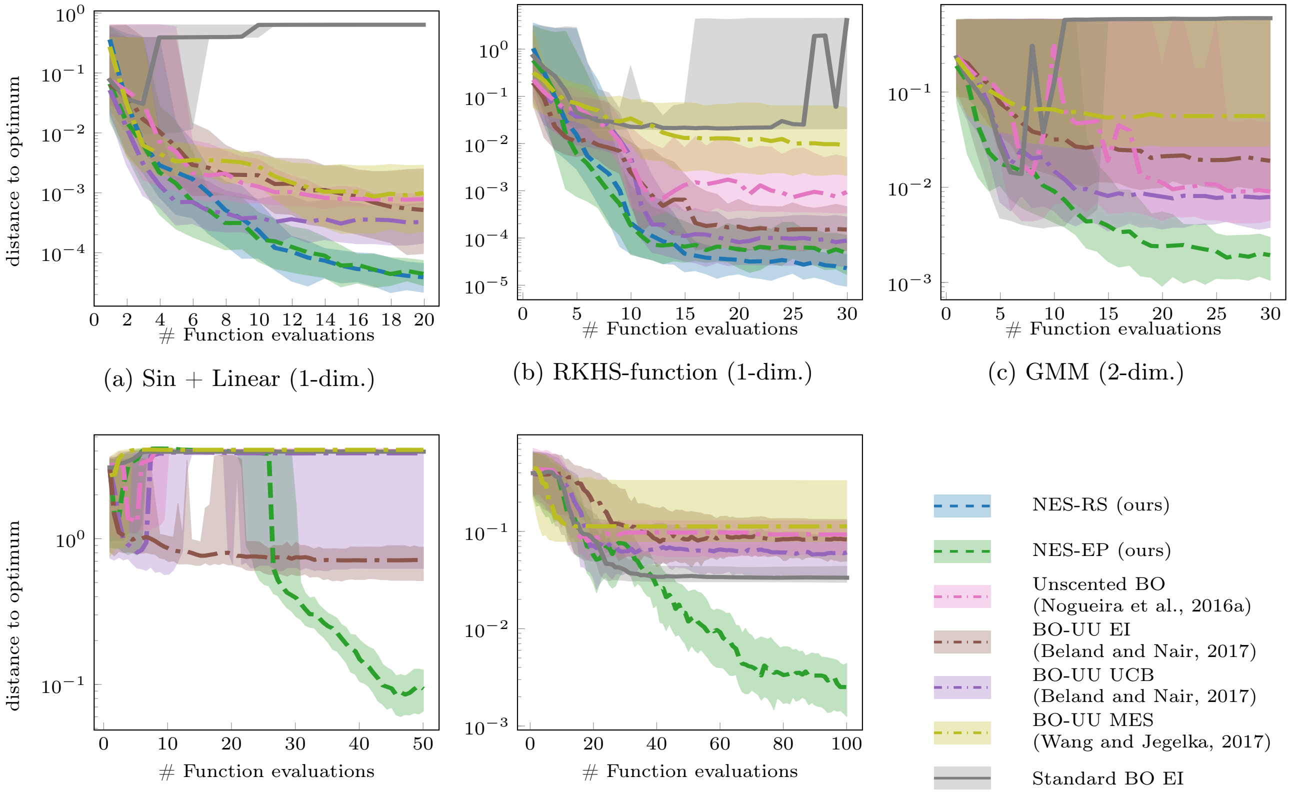
In this section, the 1- and 2-dimensional functions $f(\boldsymbol{x})$ of the synthetic benchmark problems are visualized. Furthermore, the robust counterparts $g(\boldsymbol{x})$ are depicted.

(a) $f(\boldsymbol{x}) = \sin(5\pi \boldsymbol{x}^2) + 0.5\boldsymbol{x}$, with $\boldsymbol{x} \in [0, 1]$ and $\boldsymbol{\Sigma}_x = 0.05^2$,

(b) RKHS-function (1-dim.) with $\boldsymbol{\Sigma}_x = 0.03^2$ from Assael et al. (2014), also used by Nogueira et al. (2016a),

(c) Gaussian mixture model (2-dim.) with $\boldsymbol{\Sigma}_x = 0.1^2\boldsymbol{I}$, also used by Nogueira et al. (2016a),

(d) Polynomial (2-dim.) with $\boldsymbol{\Sigma}_x = 0.6^2\boldsymbol{I}$ from Bertsimas et al. (2010), also used by Bogunovic et al. (2018). We chose the domain to be $\mathcal{X} = [-0.75, -0.25] \times [3.0, 4.2]$ and scaled/shifted the original objective $f(x)$ s.t. $\mathbb{E}[f(x)] = 0.0$ and $\mathbb{V}[f(x)] = 1.0$.



(a) Sin + Linear (1-dim.)

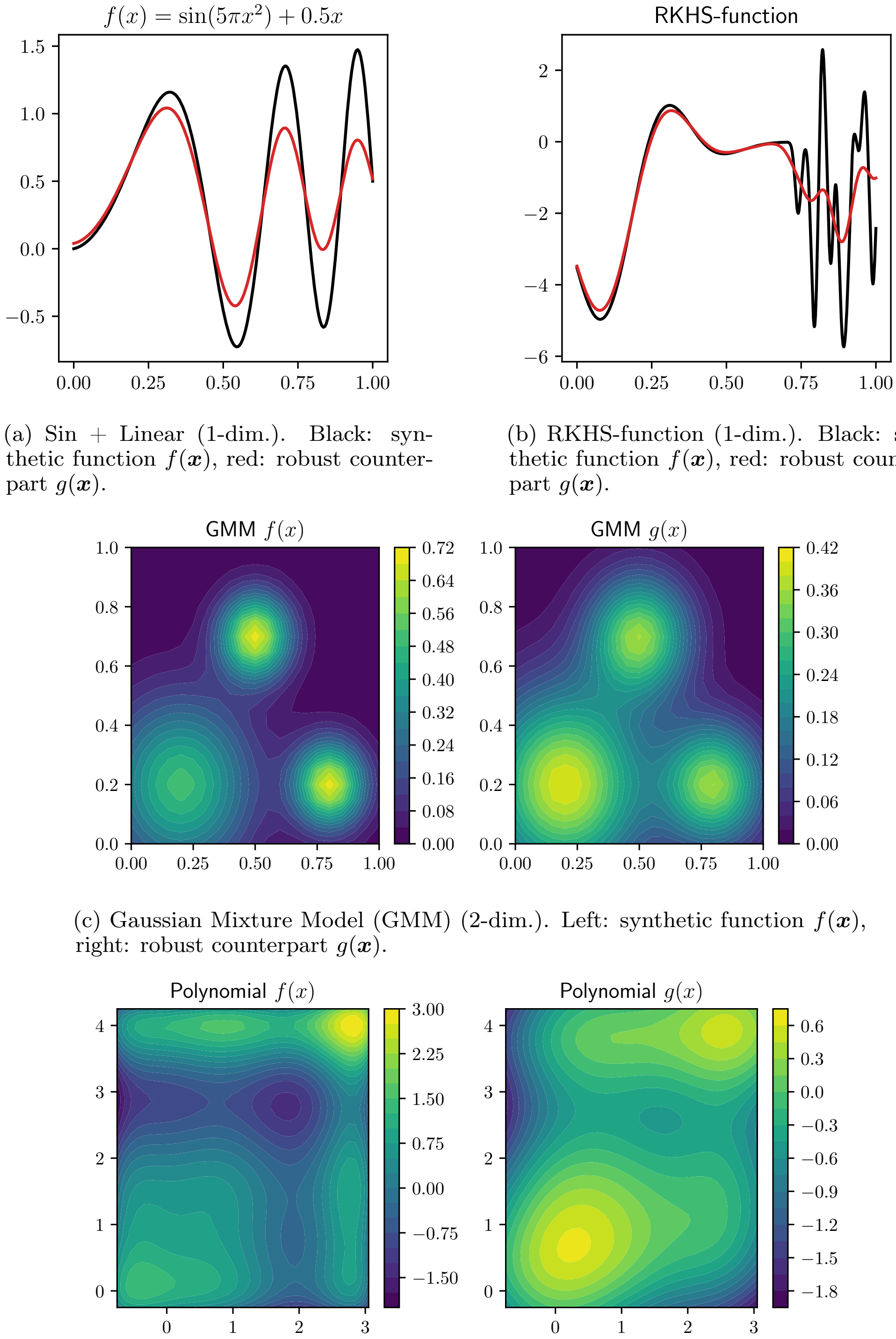
(b) RKHS-function (1-dim.)

(c) GMM (2-dim.)

(d) Polynomial (2-dim.)

(e) Hartmann (3-dim.)

Figure 4: Distance to optimum $\|\boldsymbol{x}_n^* - \boldsymbol{x}^*\|_2$ on synthetic benchmark problems. We present the median (lines) and 25/75th percentiles (shaded areas) across 100 independent runs with randomly sampled initial points.



(a) Sin + Linear (1-dim.). Black: synthetic function $f(\boldsymbol{x})$, red: robust counterpart $g(\boldsymbol{x})$.

(b) RKHS-function (1-dim.). Black: synthetic function $f(\boldsymbol{x})$, red: robust counterpart $g(\boldsymbol{x})$.

(c) Gaussian Mixture Model (GMM) (2-dim.). Left: synthetic function $f(\boldsymbol{x})$, right: robust counterpart $g(\boldsymbol{x})$.

(d) Polynomial (2-dim.). Left: synthetic function $f(\boldsymbol{x})$, right: robust counterpart $g(\boldsymbol{x})$.

Figure 5: Visualization of synthetic benchmark functions $f(\boldsymbol{x})$ with the robust counterpart $g(\boldsymbol{x})$.

References

John-Alexander M. Assael, Ziyu Wang, Bobak Shahriari, and Nando de Freitas. Heteroscedastic treed Bayesian optimisation. *arXiv preprint:1410.7172*, 2014.

Justin J. Beland and Prasanth B. Nair. Bayesian optimization under uncertainty. *NIPS Workshop on Bayesian Optimization*, 2017.

Dimitris Bertsimas, Omid Nohadani, and Kwong Meng Teo. Robust optimization for unconstrained simulation-based problems. *Operations Research*, 58(1):161–178, 2010.

Ilija Bogunovic, Jonathan Scarlett, Stefanie Jegelka, and Volkan Cevher. Adversarially robust optimization with Gaussian processes. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5765–5775, 2018.

Ralf Herbrich. On Gaussian expectation propagation. Technical report, Microsoft Research Cambridge, 2005.

Matthew W. Hoffman and Zoubin Ghahramani. Output-space predictive entropy search for flexible global optimization. In *NIPS Workshop on Bayesian Optimization*, 2015.

Simon J. Julier and Jeffrey K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

José Nogueira, Ruben Martinez-Cantin, Alexandre Bernardino, and Lorenzo Jamone. Unscented Bayesian optimization for safe robot grasping. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1967–1972, 2016a.

José Nogueira, Ruben Martinez-Cantin, Alexandre Bernardino, and Lorenzo Jamone. Unscented Bayesian optimization for safe robot grasping. *arXiv preprint arXiv:1603.02038*, 2016b.

Thomas B. Schön and Fredrik Lindsten. Manipulating the multivariate Gaussian density. Technical report, Linköping University, 2011.

Julius Orion Smith. *Mathematics of the discrete Fourier transform (DFT): with audio applications*. Julius Smith, 2007.

Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient Bayesian optimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3627–3635, 2017.