J Glob Optim (2012) 53:699–736
DOI 10.1007/s10898-011-9732-z

An experimental methodology for response surface optimization methods

Daniel J. Lizotte · Russell Greiner · Dale Schuurmans

Received: 4 January 2010 / Accepted: 19 May 2011 / Published online: 14 June 2011

Abstract Response surface methods, and global optimization techniques in general, are typically evaluated using a small number of standard synthetic test problems, in the hope that these are a good surrogate for real-world problems. We introduce a new, more rigorous methodology for evaluating global optimization techniques that is based on generating thousands of test functions and then evaluating algorithm performance on each one. The test functions are generated by sampling from a Gaussian process, which allows us to create a set of test functions that are interesting and diverse. They will have different numbers of modes, different maxima, etc., and yet they will be similar to each other in overall structure and level of difficulty. This approach allows for a much richer empirical evaluation of methods that is capable of revealing insights that would not be gained using a small set of test functions. To facilitate the development of large empirical studies for evaluating response surface methods, we introduce a dimension-independent measure of average test problem difficulty, and we introduce acquisition criteria that are invariant to vertical shifting and scaling of the objective function. We also use our experimental methodology to conduct a large empirical study of response surface methods. We investigate the influence of three properties—parameter estimation, exploration level, and gradient information—on the performance of response surface methods.

Keywords Global optimization · Response surface · Surrogate model

D. J. Lizotte (✉)
University of Michigan, Ann Arbor, MI, USA
e-mail: danjl@umich.edu

R. Greiner · D. Schuurmans
University of Alberta, Edmonton, Canada

R. Greiner
e-mail: greiner@ualberta.ca

D. Schuurmans
e-mail: dale@cs.ualberta.ca

List of symbols

f	The function to be optimized. $f : \mathcal{X} \subseteq \mathbb{R}^d \to \mathbb{R}$
$\mathcal{X}$	The domain of f
x, z, w	Points in the domain of f
x_j	The jth coordinate of the point x
$\|\|x\|\|$	The Euclidean norm of x
$\boldsymbol{x}$	A list of points $x^1, x^2, ..., x^n$
$x^{1:k}$	A sublist of points $x^1, x^2, ..., x^k$
$f(\boldsymbol{x})$, or $\boldsymbol{f}$	A list of function values $f(x^1), f(x^2), ..., f(x^n)$
$k(x, z)$	The kernel function between the points x and z
$k(\boldsymbol{x}, z)$	A column vector whose ith element is $k(x^i, z)$
$k(x, \boldsymbol{z})$	A row vector whose ith element is $k(x, z^i)$
$k(\boldsymbol{x}, \boldsymbol{z}) = \mathrm{K}$	A kernel matrix whose (i, j) th element is $k(x^i, z^j)$
$\mu(x)$	The prior mean function
σ_f^2	The "signal" or "process" variance of a Gaussian process
σ_n^2	The noise variance of a Gaussian process
ℓ_i	The characteristic length-scale along the ith dimension
L	A diagonal length-scale matrix
X	A random variable
$p(X = x)$	Probability (or density) of the event $X = x$
$\mathrm{E}[X]$	The expectation of X
$\mathrm{Cov}(X, Y)$	The covariance between X and Y: $\mathrm{Cov}(X, Y) = E[(X-E(X)) \cdot (Y-E[Y])]$
$(a)_+, [a]_+$	$\max(a, 0)$
I	The identity matrix
$\mathbb{R}$	The set of real numbers
$\Psi(x)$	The standard Gaussian right tail probability $p(X > x),\ X \sim \mathcal{N}(0, 1)$
EEC	Expected Euler Characteristic
MEI	Maximum Expected Improvement
MPI	Maximum Probability of Improvement
ILN	Independent Log Normal
BFGS	Broyden-Fletcher-Goldfarb-Shanno (hill-climbing method)

1 Introduction

Response surface methods (Jones 2001) belong to a class of global optimization algorithms that retain *all* accumulated data about the objective function and use all of it to determine where next to evaluate the objective. They are particularly appropriate for problems where the objective is multimodal and expensive to evaluate, which is common in engineering applications; response surface methods are thus extremely important in the development and analysis of complex, real-world, deployed systems.

In contrast to local optimization techniques like quasi-Newton and conjugate gradient methods, response surface methods have many free parameters that must be supplied by the user to tailor the method to the optimization problem at hand. Here we are considering "parameters" in the most general sense: a parameter could be a real number that controls the amount of exploration an algorithm does, or it could be a choice among different model classes for the response surface. Some of these parameters appear in the models of the objective function, and some appear in the acquisition criteria that are used to determine where next to observe the objective function. Guidance on how to set these parameters to achieve good performance can be gleaned from empirical studies of response surface methods, and indeed this is a sensible approach since the kind of theoretical analysis that is feasible for local optimization methods is not feasible in the more complex response surface setting. Thus, response surface methods, and global optimization techniques in general, have typically been evaluated using case-study type analyses of real-world problems, and by analysis of performance on sets of synthetic test problems.

We wholeheartedly endorse the study of response surface optimization on real-world problems (Li and Sudjianto 2005; Huang et al. 2006; Lizotte et al. 2007). Such studies reveal strengths and weaknesses of response surface methods in precisely the domains where we want good performance. Unfortunately, these studies can be high-cost in terms of time or money or both from the point of view of a researcher who is more interested in studying the performance of optimization methods rather than the substantive outcome of the study. This can make it difficult to use real-world studies for algorithm development.

Therefore, research on response surface methods uses standard synthetic test problems (Dixon and Szegö 1978) to evaluate methods instead, in the hope that these are a good surrogate for expensive, real-world problems. To date, however, many algorithm evaluations have used only a small number (i.e. fewer than twenty) hand-constructed test functions, although there is work on expanding this number (Khare et al. 2003; Deb et al. 2005). While hand-constructed functions have proven useful for ensuring that response surface methods show reasonable behaviour, we are concerned that such a small set is not sufficient for selecting appropriate parameters for a given method, nor is it sufficient for rigorously comparing different methods. Of course these functions are designed to have properties like non-concavity and multi-modality that are representative of real problems, and we would certainly not expect good methods to perform poorly on them; however, meta-optimization of optimization methods to perform well on such a limited test suite invites "overfitting" the methods to problems that have no practical importance.

Ideally, we would like to use a large number of real-world problems to evaluate and study global optimization methods, but as we noted above, this is impractical. Failing this, however, it is crucial to at least move toward the use of a much richer set of synthetic test problems in order to achieve meaningful progress. We therefore introduce a new methodology for evaluating global optimization techniques that is based on generating thousands of test functions and then evaluating algorithm performance on each one. The test functions are generated by sampling from a Gaussian process, which allows us to create a set of test functions that are interesting and diverse—they will have different numbers of modes, different maxima, etc. Despite this variety, the test functions will be similar to each other in overall structure and level of difficulty, which are attributes that can be controlled by the experimenter. This approach allows for a much richer empirical evaluation of methods that is capable of revealing insights that would not be gained using a very small set of test functions.

To facilitate the development of large empirical studies for evaluating response surface methods, we introduce two new ideas:

– We introduce a dimension-independent measure of average test problem difficulty based on an interpretation of the expected Euler characteristic of excursion sets of functions drawn from a Gaussian process, and present a polynomial time algorithm for computing this measure. This allows us to generate test problems that have different dimensionality and different "bumpiness", but have similar difficulty on average. This also allows us to express a preference for simple functions when estimating response surface model parameters.

- We introduce acquisition criteria (i.e. infill sample criteria) that are invariant to vertical shifting and scaling of the objective function. This makes the conclusions of our empirical study more general, since we would obtain the same results for all shifted and scaled versions of our test functions.

Using these ideas, we present the first large empirical study of response surface methods and investigate the influence of three properties—parameter estimation, exploration level, and gradient information—on the performance of response surface methods:

- We show how model parameter estimation based on maximum a posteriori (MAP) model selection can improve performance over maximum likelihood estimation when we have limited objective function data. Along the way, we show that an initial Latin hypercube design may be unnecessary if we are only to collect a small number of points, and we show that a more accurate response surface model results in better optimization performance, which is an important but unverified tacit assumption in the field.
- We show how the amount of exploration present in the acquisition criterion affects performance, and suggest appropriate settings for the exploration parameter of two acquisition criteria. We find that, using our invariant criteria, the same amount of exploration works well in a wide variety of problems with different dimensions, length scales, and difficulties.
- We show how the use of gradient information in response surface methods can improve performance using standard acquisition criteria. We show that with proper tuning, the expected improvement criterion performs better than a state-of-the-art local search method, in terms of accuracy achieved after a fixed number of function evaluations.

While we base these conclusions on results from large set of test problems, we of course cannot claim that they hold in general for all optimization problems. The types of test problems we consider here are motivated by our previous work in robot gait optimization (Lizotte et al. 2007). In particular, we focus on global optimization of deterministic functions whose domains are box-constrained subsets of $\mathbb{R}^d$. Furthermore, we are interested in problems where the total number of objective evaluations is limited by a budget constraint, that is, we are interested in how well an optimization procedure performs after a fixed number of objective evaluations rather than how many evaluations it takes to achieve a specified error tolerance. The budgeted setting makes intuitive sense when the cost of evaluating the objective is orders of magnitude larger than the cost of determining the next acquisition point, and is particularly salient when one has six hours in which to learn a good robot walking gait before a soccer match (Chernova and Veloso 2004). Although our particular empirical study is geared toward our problems of interest, the ideas behind the design of the study can be adapted to other scenarios as well, as we discuss in Sect. 6.

We begin in Sect. 2 by reviewing the main ideas behind response surface optimization, and we describe the viewpoint we will use in our work. Section 3 introduces a notion of problem difficulty based on results from the geometry of random fields, and illustrates how two common acquisition criteria can be made shift- and scale-invariant. Section 4 describes the specific choices we made for our empirical study, and Sect. 5 reports the results of our investigation. Finally, we summarize and discuss advantages and limitations of our approach in Sect. 6.

2 Probabilistic response surface optimization

Throughout this work, we will only consider probabilistic response surface methods, i.e. methods whose response surface model provides both a prediction of the value of the objective along with a measure of the uncertainty in that prediction. A probabilistic response surface method for optimization has two main components: The probabilistic generative *model* that describes the objective function, and the *acquisition criterion*[1] function that determines how to choose the domain point where we next evaluate the objective, based on the predictions and uncertainty of the model. This potentially gives response surface methods a major advantage: One hopes that an end-user can improve algorithm performance by modeling the objective function well, as opposed to describing a solution algorithm well, which is much more difficult for a non-expert.

Work on probabilistic response surface methods has been steady over the past 40 years (Kushner 1964; Perttunen et al. 1989; Stuckman 1989; Mockus 1989; Sacks et al. 1989; Elder 1992; Cox and John 1995; Schonlau 1997; Jones and Schonlau 1998; Gutmann 2001; Sasena 2002; Santner et al. 2003; Boyle 2006; Regis 2007; Lizotte 2008). Jones' *Taxonomy* (2001) provides a useful overview, and the review by Floudas and Gounaris (2009) lists more recent work. All of the response surface models used in the aforementioned works, whether based on Brownian motion models, thin-plate splines, radial basis functions, or Kriging, can be interpreted as *Gaussian process regression* models (Rasmussen and Williams 2006 d). This Bayesian viewpoint assumes a Gaussian process prior over possible objective functions, and computes the posterior model using Bayes's rule and the observed data. The acquisition criteria used with these models are then functions of the mean and variance of the posterior process. This framework is described in detail below.

2.1 Gaussian processes

A *Gaussian process* (GP) (Rasmussen 2004) is a collection[2] of random variables $\{F_{x^1}, F_{x^2}, \dots\}$ for which any finite subset of the variables has a joint multivariate Gaussian distribution. The variables are indexed by elements x of a set $\mathcal{X}$, the domain of the process. We will restrict our attention to $\mathcal{X} \subset \mathbb{R}^d$, but this is not necessary; $\mathcal{X}$ could be the space of integers or trees or strings, for example. For any finite length vector of indices $\boldsymbol{x} = [x^1, x^2, \dots, x^n]^T$, we have a corresponding vector $\boldsymbol{F_x} = [F_{x^1}, F_{x^2}, \dots, F_{x^n}]^T$ of random variables that have a joint multivariate Gaussian (or *normal*) distribution,

$$\boldsymbol{F_x} \sim \mathcal{N}\{\mu_0(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x})\}, \tag{1}$$

where the elements of $\mu_0(\boldsymbol{x})$ are given by a prior mean function, and k is the *kernel* or covariance function (Shawe-Taylor and Cristiani 2004). The kernel takes two indices x^i and x^j, and gives the covariance between their corresponding variables F_{x^i} and F_{x^j}. Given vectors of indices $\boldsymbol{x}$ and $\boldsymbol{z}$, the function k returns the matrix of covariances between all pairs of variables where the first in the pair comes from F_{x^i} and the second from F_{z^j}. The result of $k(\boldsymbol{x}, \boldsymbol{x})$ must be an $n \times n$, symmetric positive semi-definite matrix for any $\boldsymbol{x}$ in order for k to be a valid kernel (Shawe-Taylor and Cristiani 2004). Note that each F_{x^i} is marginally Gaussian, with mean $\mu_0(x^i)$ and variance $k(x^i, x^i)$.

[1] This function is given various names in previous work, including the 'infill sample criterion' and the 'acquisition criterion' (Sasena 2002).

[2] All of our GPs will have an uncountable number of variables, but we will abuse notation and "list" them occasionally.

2.2 Gaussian process regression

Suppose we have a function $f(x)$ that we would like to model. Furthermore, suppose that we cannot observe f directly, but that we can observe a random variable F_x that is indexed by the domain of f and whose *expected value* is f, i.e., $\forall x \in \mathcal{X},\ \mathrm{E}[F_x] = f(x)$. In particular, we assume that our prior belief about the function f conforms to a Gaussian process with prior mean μ_0 and kernel k, as described above, and that the observed variable F_x is an observation of $f(x)$ that has been corrupted by zero-mean, i.i.d. Gaussian noise, i.e., $F_x = f(x) + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$. Hence, $f(x)$ is a hidden variable whose posterior distribution we can infer after observing samples of F_x at various locations in the domain. The resulting inference is called Gaussian process regression.

Let $\boldsymbol{x}$ be the list of domain points and $\boldsymbol{f}$ be a vector of corresponding real-valued observations. We want to compute the posterior distribution at some new point $z \in \mathcal{X}$. This distribution will be Gaussian with mean and variance given by

$$\mu(F_z|\boldsymbol{F_x} = \boldsymbol{f}) = \mu_0(z) + k(z, \boldsymbol{x})(k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_\varepsilon^2 \mathrm{I})^{-1} (\boldsymbol{f} - \mu_0(\boldsymbol{x})) \tag{2}$$

$$\sigma^2(F_z|\boldsymbol{F_x} = \boldsymbol{f}) = k(z, z) - k(z, \boldsymbol{x})(k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_\varepsilon^2 \mathrm{I})^{-1} k(\boldsymbol{x}, z). \tag{3}$$

Note that the inverse applies to the kernel matrix of observed domain points (plus a diagonal term), and so can be computed once and used to evaluate the posterior at many points in the domain.[3]

Gaussian process regression is a generalization of least squares linear regression that allows for more complex regression functions, and provides information about the uncertainty of the regression model at different domain points.[4] The posterior mean function $\mu(F_z|\boldsymbol{F_x} = \boldsymbol{f})$ consists of the prior mean function $\mu_0(z)$, which is frequently taken to be constant, plus a term involving the kernel function between the query point z and each observed data point. The second part of the second term, $(k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_\varepsilon^2 \mathrm{I})^{-1} (\boldsymbol{f} - \mu_0(\boldsymbol{x}))$, is a weight vector independent of z; therefore, the posterior mean is a weighted sum of kernel functions between each observed data point and the query point. In other words, the posterior mean function is a linear combination of n copies of the kernel function, each one centered at an observed data point. From this construction, one can see that the shape of the difference between the posterior mean function and the prior mean function is governed by k, the kernel function. Depending on the choice of kernel and parametric form for the prior mean function, one can recover models used in "Kriging" (Bakker 2000) in geostatistics, as well as the thin-plate spline and Brownian motion models.

The specification of a Gaussian Process prior involves choosing the prior mean function $\mu_0(x)$ and the covariance function or kernel $k(x, x)$. These functions are usually chosen to have a particular parametric form and then the parameters of k and μ_0 are chosen to optimize some function of the data, typically the likelihood. We will choose μ_0 from the set of constant functions, and we will examine two classes of kernel function to choose k. It is common to

[3] In practice, the matrix will be Cholesky factored instead of inverted. Additionally, methods have been developed (Csató and Opper 2001; Lawrence et al. 2003) for reducing this computational burden that could be directly applied to response surface optimization.

[4] For $\mathcal{X} \subset \mathbb{R}^k$ and $k(x_i, x_j) = x_i \cdot x_j$, this formulation is equivalent to linear ridge regression with regularizer σ_ε^2.

choose k to be a function of the weighted difference between points, i.e.[5] $k(x, z) = k(r(x, z))$ where r is the vector given by

$$r_i(x_i, z_i) = \frac{x_i - z_i}{\ell_i}, \ \ell_i > 0, \quad i = 1 \dots d. \tag{4}$$

The ℓ_i parameters introduced here are referred to as *length scales*. Covariance functions defined in terms of r typically assign a larger covariance to pairs of points with small $||r||$, and small covariance to pairs with large $||r||$, that is, domain points that are near to each other will have associated random variables that are strongly correlated. This expresses a belief that underlying function f is smooth. We allow the amount of smoothness of each dimension to be different by assigning a length scale ℓ_i to each one. The length scales give us some idea of the importance of each dimension for optimization: If ℓ_i is very large, then we expect that moving along the ith dimension will not have a large effect on f.

One common choice that gives large covariance to nearby points is to define k to be a "squared exponential" (SE) or Gaussian kernel.

$$k(r) = \sigma_f^2 \cdot \mathrm{e}^{-\frac{||r||^2}{2}} \tag{5}$$

Another commonly used kernel is the Matérn kernel (Rasmussen and Williams 2006c), which consists of an exponential kernel multiplied by a polynomial. This class of kernels can be parameterized by the same length scales ℓ_i as the SE kernel, along with an additional real parameter $\nu > 0$. For half-integer values of ν, the kernel has a simple form. Two commonly-used values of ν are

$$k_{\nu=3/2}(r) = \sigma_f^2 \cdot \left(1 + \sqrt{3}||r||\right) \cdot \mathrm{e}^{-\sqrt{3}||r||} \tag{6}$$

$$k_{\nu=5/2}(r) = \sigma_f^2 \cdot \left(1 + \sqrt{5}||r|| + \frac{5}{3}||r||^2\right) \cdot \mathrm{e}^{-\sqrt{5}||r||} \tag{7}$$

As $\nu \to \infty$ the Matérn kernel function converges uniformly to the squared exponential kernel (Rasmussen and Williams 2006c).

2.3 Maximum likelihood parameter estimation

When there is a lack of *a priori* knowledge about how to select a kernel or its parameters, it is common to use a maximum likelihood criterion to select the functional form of k and estimate any free parameters k may have.[6] The likelihood function for a Gaussian process with prior mean function $\mu_0(\boldsymbol{x})$ and kernel k is given by

$$\log p(F_{\boldsymbol{x}} = \boldsymbol{f}) = -\frac{1}{2}(\boldsymbol{f} - \boldsymbol{\mu}_0)^{\mathsf{T}} \, \mathrm{K}^{-1} \, (\boldsymbol{f} - \boldsymbol{\mu}_0) - \frac{1}{2} \log |\mathrm{K}| - \frac{n}{2} \log 2\pi \tag{8}$$

where $\mathrm{K} = k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_\varepsilon^2 \mathrm{I}$. In the above equation, $\boldsymbol{x}$, $\boldsymbol{f}$, and n are uniquely determined by the data. The quantities σ_ε^2 and $\boldsymbol{\mu}_0$ and the ℓ_i and σ_f^2 that parameterize k, are free model parameters, which are chosen to maximize (8). (In our work we will assume deterministic observations and therefore fix σ_ε to a very small positive value for numerical stability.) In Sect. 5.1 we will examine modifications to (8) that reflect prior beliefs about kernel parameters.

[5] We make the following notational abuse throughout: If k has two parameters, the parameters are domain points. If k has one parameter, that parameter is a weighted difference of two domain points.

[6] Note that cross validation can also be used for model selection in this way; see (Rasmussen and Williams 2006b) for further details.

We want our optimization procedure to be invariant to additive scaling, which we describe in Sect. 3.2. To this end, we will always choose the best *constant* $\boldsymbol{\mu}_0$ for our data and substitute it into the likelihood equation, effectively removing $\boldsymbol{\mu}_0$ as a parameter by optimizing it out to maximize likelihood. We can find this most likely constant $\boldsymbol{\mu}_0$ by defining $\mu_0(\boldsymbol{x}) = \mu_c$, implying $\boldsymbol{\mu}_0 = \mathbf{1} \cdot \mu_c$. We then take the derivative with respect to μ_c and equate it to zero, which gives the maximum likelihood constant prior mean

$$\boldsymbol{\mu}_0 = \boldsymbol{\mu}_* = \left(\frac{\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\boldsymbol{f}}{\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\mathbf{1}}\right)\mathbf{1}$$

where $\mathbf{1}$ is the vector of all ones. Furthermore, since

$$\begin{aligned}(\boldsymbol{f}-\boldsymbol{\mu}_*)^\mathsf{T}\mathrm{K}^{-1}(\boldsymbol{f}-\boldsymbol{\mu}_*) &= \boldsymbol{f}^\mathsf{T}\mathrm{K}^{-1}\boldsymbol{f} - 2\boldsymbol{f}^\mathsf{T}\mathrm{K}^{-1}\boldsymbol{\mu}_* + \boldsymbol{\mu}_*{}^\mathsf{T}\mathrm{K}^{-1}\boldsymbol{\mu}_*\\ &= \boldsymbol{f}^\mathsf{T}\mathrm{K}^{-1}\boldsymbol{f} - 2\boldsymbol{f}^\mathsf{T}\mathrm{K}^{-1}\mathbf{1}\left(\frac{\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\boldsymbol{f}}{\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\mathbf{1}}\right) + \mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\mathbf{1}\left(\frac{\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\boldsymbol{f}}{\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\mathbf{1}}\right)^2\\ &= \boldsymbol{f}^\mathsf{T}\mathrm{K}^{-1}\boldsymbol{f} + \frac{-2\boldsymbol{f}^\mathsf{T}\mathrm{K}^{-1}\mathbf{1}\left(\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\boldsymbol{f}\right) + (\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\boldsymbol{f})^2}{\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\mathbf{1}}\\ &= \boldsymbol{f}^\mathsf{T}\mathrm{K}^{-1}\boldsymbol{f} - \frac{(\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\boldsymbol{f})^2}{\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\mathbf{1}}\end{aligned}$$

it follows that if we use $\boldsymbol{\mu}_*$ for the mean, the likelihood function can be written

$$\log\, p(F_{\boldsymbol{x}} = \boldsymbol{f}) = -\frac{1}{2}\boldsymbol{f}^\mathsf{T}\mathrm{K}^{-1}\boldsymbol{f} - \frac{1}{2}\log|\mathrm{K}| - \frac{n}{2}\log 2\pi - \frac{1}{2}\frac{(\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\boldsymbol{f})^2}{\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\mathbf{1}} \tag{9}$$

The first three terms of this expression represent the likelihood assuming $\boldsymbol{\mu}_0 = \mathbf{0}$, and the last term accounts for our maximizing to find $\boldsymbol{\mu}_*$. Therefore, when we take a partial derivative with respect to a kernel parameter θ, we can use the usual form (Rasmussen and Williams 2006b) of the likelihood gradient plus an extra term to account for our optimization of $\boldsymbol{\mu}$. The derivative of the last term above is

$$\begin{aligned}&\frac{\partial}{\partial\theta}\frac{(\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\boldsymbol{f})^2}{\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\mathbf{1}}\\ &\quad= \frac{-2(\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\mathbf{1})(\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\boldsymbol{f})(\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\frac{\partial\mathrm{K}}{\partial\theta}\mathrm{K}^{-1}\boldsymbol{f}) + (\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\boldsymbol{f})^2(\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\frac{\partial\mathrm{K}}{\partial\theta}\mathrm{K}^{-1}\mathbf{1})}{(\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\mathbf{1})^2}\\ &\quad= \frac{1}{(\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\mathbf{1})^2}\cdot\\ &\qquad\times\left(-2(\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\mathbf{1})(\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\boldsymbol{f})\ \boldsymbol{f}^\mathsf{T}\mathrm{K}^{-1} + (\mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\boldsymbol{f})^2\ \mathbf{1}^\mathsf{T}\mathrm{K}^{-1}\right)\cdot\ \frac{\partial\mathrm{K}}{\partial\theta}\cdot(\mathrm{K}^{-1}\mathbf{1})\end{aligned}$$

so if we define $\boldsymbol{\alpha} = \mathrm{K}^{-1}\boldsymbol{f}$, and $\boldsymbol{\gamma} = \mathrm{K}^{-1}\mathbf{1}$, the derivative of the likelihood function is

$$\frac{\partial}{\partial\theta}\log\, p(F_{\boldsymbol{x}} = \boldsymbol{f}) = \frac{1}{2}\,\mathrm{tr}\left[(\boldsymbol{\alpha\alpha}^T - \mathrm{K}^{-1})\frac{\partial \mathrm{K}}{\partial\theta}\right] - \frac{1}{2}\frac{1}{(\boldsymbol{\gamma}^{\mathsf{T}}\mathbf{1})^2}\left(-2(\boldsymbol{\gamma}^{\mathsf{T}}\mathbf{1})(\boldsymbol{\alpha}^{\mathsf{T}}\mathbf{1})\,\boldsymbol{\alpha}^{\mathsf{T}} + (\boldsymbol{\alpha}^{\mathsf{T}}\mathbf{1})^2\,\boldsymbol{\gamma}^{\mathsf{T}}\right)\cdot\frac{\partial \mathrm{K}}{\partial\theta}\cdot(\boldsymbol{\gamma}) \quad (10)$$

Note that we only use kernels for which $\partial k(x,z)/\partial\theta$ exists for all x and z in the domain of f, and for all θ we are interested in.

2.4 Gaussian processes as generative models

As seen in Eq. (1), *a priori*, $\boldsymbol{F_x}$ is multivariate Gaussian with mean given by the prior mean function and variance given by the kernel. Thus, for any specified set of domain points $\boldsymbol{x}$, we may draw a sample from $\boldsymbol{F_x}$. If every point z in the domain is sufficiently close to one of the $\boldsymbol{x}$, then the value of F_z is almost completely determined, since it will be very highly correlated with nearby F_x and its posterior variance will be very low. Thus we can sample a function from the Gaussian process prior by sampling the vector $\boldsymbol{F_x}$, conditioning on the resulting data, and using the posterior mean to interpolate for $z \notin \boldsymbol{x}$. By repeatedly drawing different $\boldsymbol{F_x}$ for sufficiently large sets $\boldsymbol{x}$, we can sample different functions from the Gaussian process prior.

If the prior mean is taken to be zero, the properties of the sampled functions will be determined entirely by the kernel. The ℓ_i in a kernel control how quickly a function sampled from the GP varies along each dimension: Sampled functions will vary quickly along dimensions with small ℓ, and will vary slowly along dimensions with large ℓ. Figure 1 illustrates this using a 2D example. The functional form of the kernel also determines other measures of smoothness. For example, the squared exponential kernel produces processes that are infinitely mean-square differentiable, whereas the Matérn kernel with $\nu = 3/2$ produces processes that are mean-square differentiable exactly once.

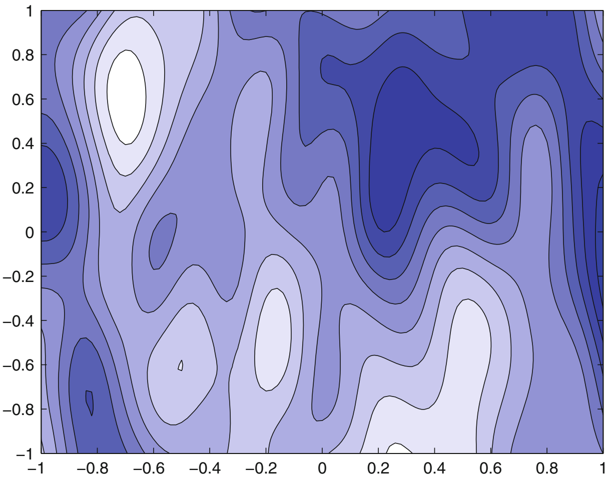


Fig. 1 Contour plot of a function sampled from a Gaussian process over $\mathbb{R}^2$ with $\mu_0(z) = 0$ and squared exponential kernel with horizontal length scale $\ell_1 = 0.13$, and vertical length scale $\ell_2 = 0.41$. This function was constructed by drawing 500 domain points $\boldsymbol{x}$ uniformly randomly, drawing a vector $\boldsymbol{f}$ from the multivariate normal $\mathcal{N}\{0, k(\boldsymbol{x},\boldsymbol{x})\}$, and plotting $\mu(F_z|\boldsymbol{F_f} = \boldsymbol{f})$. Note how the function varies more rapidly along the horizontal dimension than along the vertical dimension

Our empirical study will evaluate the performance of response surface methods on sampled functions from different Gaussian process priors with different dimensionality, length scales, and kernel family.

3 A measure of problem difficulty and invariant acquisition criteria

We now give two extensions to response surface optimization that are useful for developing large empirical studies. First we review the expected Euler characteristic of a Gaussian process, give a polynomial time algorithm for computing it, and offer an interpretation that is relevant for assessing the difficulty of optimization problems drawn from a GP. We then describe how the two most common acquisition criteria can be modified so that the resulting points chosen by them are invariant to vertical shifting and scaling of the objective function.

3.1 The difficulty of a global optimization problem

Common test problems used in research on response-surface methods fall into two common types of problems, the "mountain-range" problems and the "needle-in-a-haystack" problems. Mountain-range problems have several local optima whose basins of attraction are similarly sized, and whose *function values* are not far from the value at the global optimum. For this type of problem, good performance can be achieved by finding one of the local optima whose *value* is near the global optimum. Needle-in-a-haystack problems have a global optimum whose basin of attraction is very small, and whose value is much more extreme than the rest of the function. For this type of problem, performance will be poor unless our algorithm correctly determines the location of the "needle"—a scenario we feel is not representative of most real-world problems. (On these problems, nothing short of exhaustive search will perform well.) Depending on the kernel, functions sampled from a Gaussian process can have either of these characteristics: If the domain is too large or length scales are too small or the dimension is too high, then we will produce needle-in-a-haystack test functions that are hopeless for any reasonable optimization method. We must quantify this difficulty to ensure that the test functions we generate are reasonable surrogates for real-world problems.

Therefore, we present a property of Gaussian processes of arbitrary dimension that gives insight into whether functions sampled from a GP have the mountain-range or the needle-in-a-haystack properties. To do this, we review what is known about the *excursion sets* of functions sampled from GPs, which are sets $\mathcal{A}_u(f) = \{x : f(x) \geq u\}$. An example of an excursion set in two dimensions is shown in Fig. 2. For a smooth function f with a unique global maximum, the excursion set $\mathcal{A}_u$ for $u = \sup_{\mathcal{X}} f - \delta$ for some small enough $\delta > 0$ will consist of a connected, nearly ellipsoid-shaped closed set of points containing $x^* = \operatorname{argmax}_{\mathcal{X}} f(x)$. To see this, note that by Taylor's theorem, if f is twice differentiable at the point x^*, the behavior of f near x^* will be very nearly quadratic within a small enough radius, and that the level sets of a quadratic function are ellipsoids. As we decrease u, the size of $\mathcal{A}_u$ will grow, and new components will appear around other local maxima that u crosses on its descent. The excursion set $\mathcal{A}_u$ will become more and more connected until it consists of most of the domain $\mathcal{X}$, but with holes surrounding the locations of extreme local minima.

We now consider what we can tell about a function from looking at its excursion sets to try to give an idea of their potential in studying extrema. First, it is easy to see that the number of connected components of $\mathcal{A}_u$ gives a lower bound on the number of local maxima that exceed level u. Second, if we somehow knew that a Gaussian process would produce either a single local maximum above u or none at all, then the number of components of $\mathcal{A}_u$ for any

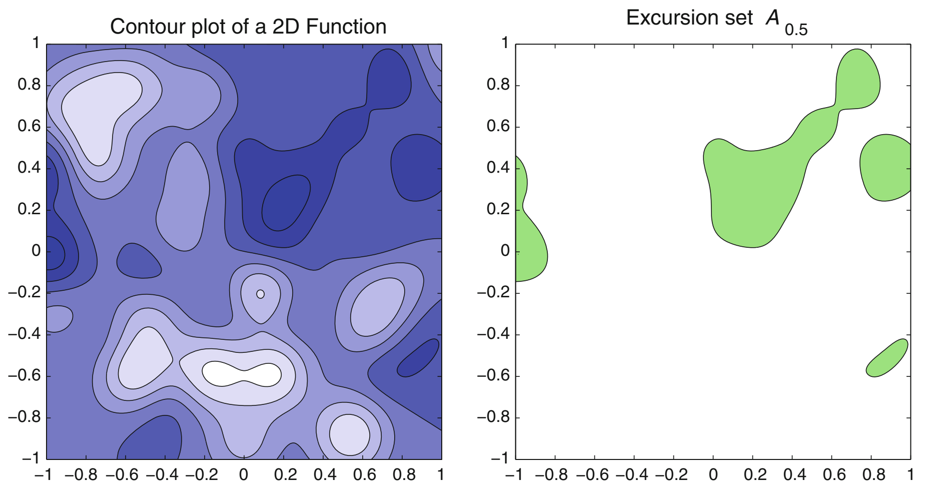


Fig. 2 Example of an excursion set for a two-dimensional function f. The *shaded area* in the graph on the *right* is the excursion set $\mathcal{A}_{0.5}$—the set of domain points where $f > 0.5$—has four components, none of which have holes. Therefore $\chi(\mathcal{A}_{0.5}) = 4$ for this function

sampled function will be zero or one, and the expected number of components of $\mathcal{A}_u$ would be equal to $p(\sup_{x\in\mathcal{X}} F_x \geq u)$, which we call the *excursion probability* of F above u. It turns out that the best asymptotic approximations known for excursion probabilities (and they are very good for moderate to large u/σ_f, i.e. $u/\sigma_f > 1.5$) are related to counting connected components of excursion sets (Adler and Taylor 2007).

The number of connected components of a closed set is an example of a topological invariant. That is, the quantity is the same for all sets that are *homeomorphic*—sets between which there exists a continuous, one-to-one, and onto mapping. Less formally, a topological invariant property of two sets will be the same if one set can be smoothly deformed into the other, like a doughnut and a coffee cup, but not a doughnut and a bowling ball.[7] The *Euler characteristic* $\chi(\mathcal{S})$ of a set $\mathcal{S}$ is the most widely known topological invariant.[8] Precisely defining the Euler characteristic is a lengthy process, so we instead give a few of its properties that are relevant to our discussion.

$$\chi(\mathcal{B}_d) = 1 \quad \text{where } \mathcal{B}_d \text{ is the unit ball in } \mathbb{R}^d$$
$$\chi(\mathcal{U} \cup \mathcal{V}) = \chi(\mathcal{U}) + \chi(\mathcal{V}) \qquad \text{if } \mathcal{U} \cap \mathcal{V} = \emptyset$$

Any set that is homeomorphic to $\mathcal{B}_d$, for example the ellipsoid-shaped components of excursion sets around local maxima that we mentioned above, counts for 1 when computing the Euler characteristic. Furthermore, the characteristic is additive, so if $\mathcal{A}_u$ consists of some number of disjoint components that are each homeomorphic to $\mathcal{B}_d$, then the $\chi(\mathcal{A}_u)$ will be the number of components of $\mathcal{A}_u$. We have of course left out all of the sets which are not homeomorphic to $\mathcal{B}_d$; here are a few examples of these:

[7] The problem here is continuity—to map the bowling ball back to the doughnut, there will have to be points that are nearby on the bowling ball that map to points that are far apart on the original doughnut, and vice versa. This is caused by the hole in the doughnut (Or by the absence of a hole through the bowling ball).

[8] Along with *genus*, which is defined for surfaces. The orientable genus g of a surface can be defined as $g = 1 - \chi/2$.

$$\chi(\mathcal{K}_{d,h}) = 1 + h \cdot (-1)^d$$
$$\chi(\bar{\mathcal{K}}_{d,h}) = 1 - h$$

Here, $\mathcal{K}_{d,h}$ is $\mathcal{B}_d$ with h non-intersecting d-dimensional holes drilled through it, and $\bar{\mathcal{K}}_{d,h}$ is $\mathcal{B}_d$ with h "handles" attached to it. Envisioning a d-dimensional sphere with d-dimensional handles attached to it is taxing at best, but the important thing to note is that $\chi(\mathcal{S})$ is *not* simply the number of connected components of $\mathcal{S}$.

We now come to the single most important formula governing the extrema of Gaussian processes.

Theorem 1 *Adler and Taylor (2007). For a stationary Gaussian process with zero mean, we have*

$$\mathrm{E}[\chi(\mathcal{A}_u)] = e^{-u^2/2\sigma_f^2} \sum_{k=1}^{d} \sum_{\mathcal{J} \in \mathcal{E}_k} \frac{\mu_L(\mathcal{J}) |\Lambda_{\mathcal{J}}|^{1/2}}{(2\pi)^{(k+1)/2} \sigma_f^k} H_{k-1}\left(\frac{u}{\sigma_f}\right) + \Psi\left(\frac{u}{\sigma_f}\right) \tag{11}$$

where the process is over the set $\mathcal{X}$, a rectangle in $\mathbb{R}^d$ with one vertex at the origin. The $\mathcal{E}_k$ are collections of the k-dimensional facets of the boundary of $\mathcal{X}$ that contain the origin, of which there are $\binom{d}{k}$. For example, if $\mathcal{X} \subset \mathbb{R}^3$, then $\mathcal{E}_1$ would be the three line segments that border $\mathcal{X}$ and touch the origin, $\mathcal{E}_2$ would be the three 2-dimensional rectangles that border $\mathcal{X}$ and touch the origin, and $\mathcal{E}_3$ would be $\mathcal{X}$ itself. In the sum, $\mathcal{J}$ ranges over these $\binom{d}{k}$ facets. $\Lambda_{\mathcal{J}}$ is the matrix of spectral moments of the process when restricted to the facet $\mathcal{J}$, whose elements are given by $\lambda_{ij} = \left.\frac{\partial^2 k(r)}{\partial r_i \partial r_j}\right|_{r=0}$. σ_f is the process variance. $\mu_L(\mathcal{J})$ is the (k-dimensional) Lebesgue measure of the facet $\mathcal{J}$, and $\Psi(x)$ is the standard Gaussian right tail probability $p(X > x)$, $X \sim \mathcal{N}(0, 1)$. $H_{k-1}(\cdot)$ is the $k-1$st "probabilist's" Hermite polynomial, defined in "Appendix B".

The preceding formula is important not only because it is interesting to know the expected Euler characteristic of excursion sets, but because it provides the state-of-the-art approximation to excursion probabilities of Gaussian processes:

$$|p\{\sup F(x) \geq u\} - \mathrm{E}[\chi(\mathcal{A}_u)]| < \mathcal{O}\left(e^{-\alpha u^2/2\sigma_f^2}\right) \tag{12}$$

for $\alpha > 1$. Furthermore, it provides insight in to how the spectral moments—and therefore the length-scales—of a process influence its supremum distribution. To see this more clearly, we give a more specialized version of Eq. (11).

Corollary 1 *For an axis-scaled isotropic Gaussian process over the hyper-rectangle $\mathcal{X} = [0, w_1] \times [0, w_2] \times \cdots \times [0, w_d]$, if we let $\mathcal{D}(\mathcal{J})$ represent the set of indices of the k axes that are included in a boundary set (i.e. facet) $\mathcal{J} \in \mathcal{E}_k$, the expected Euler characteristic is given by*

$$\mathrm{E}[\chi(\mathcal{A}_u)] = e^{-u^2/2\sigma_f^2} \sum_{k=1}^{d} \sum_{\mathcal{J} \in \mathcal{E}_k} \frac{\left[\prod_{i \in \mathcal{D}(\mathcal{J})} w_i\right] \cdot \left[\prod_{i \in \mathcal{D}(\mathcal{J})} \sqrt{\lambda_{ii}}\right]}{(2\pi)^{(k+1)/2} \sigma_f^k} H_{k-1}\left(\frac{u}{\sigma_f}\right) + \Psi\left(\frac{u}{\sigma_f}\right) \tag{13}$$

Proof This corollary uses two main properties: First, for an axis-scaled isotropic process, since $\lambda_{ij:i \neq j} = 0$, we have $|\Lambda_{\mathcal{J}}|^{1/2} = \prod_{i \in \mathcal{D}(\mathcal{J})} \sqrt{\lambda_{ii}}$. Furthermore, since the domain is a hyper-rectangle, the k-dimensional Lebesgue measure of each facet $\mathcal{J}$ is given by $\mu_L(\mathcal{J}) = \prod_{i \in \mathcal{D}(\mathcal{J})} w_i$. $\Psi(x)$ is the standard Gaussian right tail probability $p(X > x)$, $X \sim \mathcal{N}(0, 1)$. □

Recall from (11) that there are $\binom{d}{k}$ such boundary sets $\mathcal{J}$ for each $k \in \{1, 2, ..., d\}$. Therefore, a naïve expansion of the sum in (13) would involve $\sum_{i=1}^{d} \binom{d}{k} = 2^d$ terms, making use of this approximation prohibitively expensive. We now give a novel re-formulation of (11) that takes $\mathcal{O}(d^2)$ time to evaluate.

Theorem 2 *For an axis-scaled isotropic Gaussian process as defined in Corollary 1, the expected Euler characteristic is also given by*

$$\mathrm{E}[\chi(\mathcal{A}_u)] = e^{-u^2/2\sigma_f^2} \sum_{k=1}^{d} \frac{S_k}{(2\pi)^{(k+1)/2}\sigma_f^k} H_{k-1}\left(\frac{u}{\sigma_f}\right) + \Psi\left(\frac{u}{\sigma_f}\right) \tag{14}$$

where

$$S_k = \frac{\sum_{j=1}^{k}(-1)^{j+1} S_{k-j} \cdot S^{[j]}}{k} \tag{15}$$

$$S_0 = 1 \tag{16}$$

$$S^{[j]} = \sum_{i=1}^{d} (w_i\sqrt{\lambda_{ii}})^j \tag{17}$$

This alternate factorization, in terms of S, generates the necessary sums for each dimension k in $\mathcal{O}(d)$ time, for a total run time of $\mathcal{O}(d^2)$. Each of σ_f^2, w_i, λ_{ii}, H and Ψ are defined as in Corollary 1. Note that the λ_{ii} only enter into the computation via each $S^{[j]}$.

Proof See “Appendix A”.

The value S_k is the evaluation of the kth *elementary symmetric polynomial* in d variables, at the point $(w_1\sqrt{\lambda_{11}}, w_2\sqrt{\lambda_{22}}, ..., w_d\sqrt{\lambda_{dd}})$. These polynomials are symmetric in all d variables, which implies that the ordering of the axes of the domain does not matter—so long as the $w_i\sqrt{\lambda_{ii}}$ remain the same, the excursion sets of sampled functions will have the same expected Euler characteristics. They are also linear in each w_i and $\sqrt{\lambda_{ii}}$, since there are no exponents greater than one. This means that if we fix u, σ_f, and w_i, then $\mathrm{E}[\chi(\mathcal{A}_u)] = c_1 \cdot \sqrt{\lambda_{ii}} + c_0$ for some constants c_1 and c_0.

For example, suppose we had a zero-mean Gaussian process with covariance given by the Matérn kernel with $\nu = 3/2$ as defined in Eq. (6). To use Theorem 2, we need to know σ_f^2, which is part of the definition of the process, the w_i, which are the widths of each axis of the domain, and the λ_{ii} which are derived from the kernel function. Recall that for this Matérn kernel,

$$k_{\nu=3/2}(r) = \sigma_f^2 \cdot \left(1 + \sqrt{3}||r||\right) \cdot \mathrm{e}^{-\sqrt{3}||r||}$$

where

$$r_i(x_i, z_i) = \frac{x_i - z_i}{\ell_i}, \quad i = 1 \ldots d$$

From the definition of the second spectral moments, we have

$$\lambda_{ii} = -\left.\frac{\partial^2 k(r)}{\partial r_i^2}\right|_{r=0} = -\frac{\partial^2}{\partial r_i^2}\left[\sigma_f^2 \cdot \left(1 + \sqrt{3}||r||\right) \cdot \mathrm{e}^{-\sqrt{3}||r||}\right]\Bigg|_{r=0} = \sigma_f^2 \cdot \frac{3}{\ell_i^2} \tag{18}$$

From this, we see that the spectral moments λ_{ii} depend only on the process variance and the length scales. (Incidentally, if we had used a squared exponential kernel for k instead, we would have $\lambda_{ii} = \sigma_f^2/\ell_i^2$.) This shows that decreasing the length scale along dimension i increases the spectral moment for that dimension, pushing more power into higher frequencies relative to the other dimensions, and making dimension i "bumpier" than the others. Using σ_f, w_i, and λ_{ii} with Theorem 2, we can compute $E[\chi(\mathcal{A}_u)]$ for this kernel with arbitrary length scales and any value of u. Furthermore, since $E[\chi(\mathcal{A}_u)]$ is linear in each of the λ_{ii}, it is a smooth function with respect to ℓ_i and σ_f^2, and derivatives with respect to these quantities are easily computed using the chain rule. We will require these derivatives later when we use $E[\chi(\mathcal{A}_u)]$ in a prior over the ℓ_i that favours simple functions.

Theorem 2 allows the expected Euler characteristic to be used for the first time in a computational setting when working with Gaussian processes over hyper-rectangles with more than a few dimensions. We now discuss the interpretation of the Expected Euler Characteristic (EEC) of excursion sets of a GP in relation to the difficulty of optimizing test objectives sampled from that GP.

3.1.1 *Interpreting the expected euler characteristic*

The Euler characteristic $\chi(\mathcal{A})$ of a set $\mathcal{A}$ is equal to the number of connected components in $\mathcal{A}$ if these components are each homeomorphic to the d-dimensional ball. The excursion set $\mathcal{A}_u$ of a function above level u consists of all points in the domain where the function exceeds u. For sufficiently large u, the excursion sets become the union of small disjoint convex sets surrounding the local maxima of the function in question and therefore the Euler characteristic counts the number of local maxima above u. We propose the expected Euler characteristic of excursion sets as a measure of the difficulty of optimizing functions drawn from a particular Gaussian process model—the higher the $\mathrm{E}[\chi(\mathcal{A}_u)]$ for large u, the more difficult the optimization problems. Most practitioners would agree that functions with many local maxima are difficult to optimize, assuming no special structure is known about the problem. In addition to this argument, the expected Euler characteristic allows us to say something about the "mountain-range" and "needle-in-a-haystack" properties discussed earlier.

To see how EEC influences difficulty, as a thought experiment we consider the performance of a naïve algorithm that evaluates any objective at the same fixed set of n domain points and returns the best result. If we test this algorithm for $n = 1$ on functions drawn from a GP, then *a priori*, the probability that the evaluation exceeds $3.0\sigma_f$ is the Gaussian right tail probability $P(X > 3.0)$, $X \sim \mathcal{N}(0, 1)$, which we write as $\Psi(3.0) = 0.0013$. If for $n = 30$, the probability that the best result is greater than $3.0\sigma_f$ is less than $1 - (1 - \Psi(3.0))^{30} = 0.0383$, and the probability the best result is greater than $2.0\sigma_f$ is less than $1 - (1 - \Psi(2.0))^{30} = 0.4986$. These simple bounds hold regardless of the correlation structure of the process induced by the kernel (they assume independence) and therefore they hold regardless of the underlying domain $\mathcal{X}$ and the dimension of the problem.

Suppose we knew that $\mathrm{E}[\chi(\mathcal{A}_{3.0\sigma_f})] = 1.0$, meaning that there is a very high probability that *somewhere* in the domain we could achieve a value of $3.0\sigma_f$ or greater—the needle in the haystack. The fixed strategy sampling 30 points achieves this level with only small probability, and furthermore misses the mark by at least one standard deviation more than half of the time. Thus, on average over sampled functions, performance will be poor.

On the other hand, suppose $\mathrm{E}[\chi(\mathcal{A}_{3.0\sigma_f})]$ is very low, say around 0.01. Then the average performance of the fixed strategy should be much better, because it is unlikely that in any given sampled function there exists a needle in the haystack somewhere in the domain.

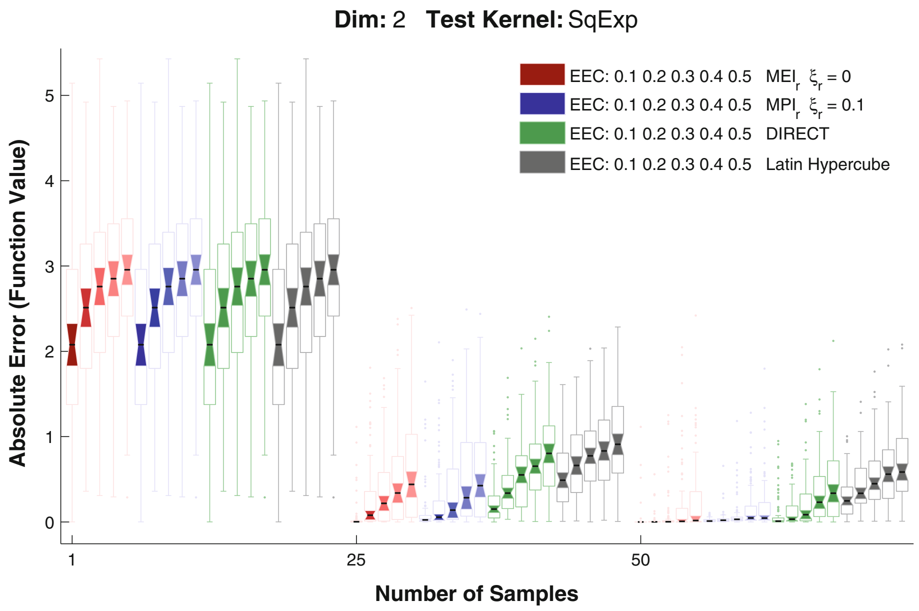


Fig. 3 Performance of two GP-based optimization methods, the DIRECT algorithm, and a random Latin hypercube allocation scheme on test problems with varying values of E[$\chi(\mathcal{A}_{3.0\sigma_f})$]. Each test problem was drawn from a 2D GP prior using a squared exponential kernel with length scales set so as to generate problems with a set value of E[$\chi(\mathcal{A}_{3.0\sigma_f})$]. Each group of 5 boxplots gives the absolute error of a particular method after 1, 25, or 50 acquisitions for each level of EEC. The left-to-right order of the groups matches the top-to-bottom order of the legend. Note that as EEC increases, problems become more difficult and the methods all show a decrease in performance (i.e. an increase in median absolute error)

Figure 3 shows performance of four optimization methods on a suite of test problems ranging in difficulty from E[$\chi(\mathcal{A}_{3.0\sigma_f})] = 0.1$ to E[$\chi(\mathcal{A}_{3.0\sigma_f})] = 0.5$. As expected, we see a decrease in performance with increasing E[$\chi(\mathcal{A}_{3.0\sigma_f})$]; in our examples this holds true for two GP-based optimization methods, the DIRECT method (Jones et al. 1993), and Latin hypercube sampling.

For our empirical study, we are interested in a moderate difficulty level when examining the effect of different dimensions and length scale ratios on performance; therefore, we have chosen to use GPs with E[$\chi(\mathcal{A}_{3.0\sigma_f})] = 0.2$. Figure 4 shows two examples of these functions. Setting E[$\chi(\mathcal{A}_{3.0\sigma_f})] = 0.2$ means that in approximately 1/5 of our sampled test functions there will be a rather extreme maximum that is unlikely to be found by the fixed strategy. At this level of difficulty, on average approximately 30 samples are needed by the GP-based methods to achieve an absolute error less than 0.01. To explore how problem difficulty in terms of EEC interacts with the exploration parameter ξ_r (defined below), we will also consider test functions drawn from a 2D Gaussian process with EEC ranging from 0.1 to 0.5.

3.2 Invariant acquisition criteria

The choice of acquisition criterion used to select where next to evaluate the objective clearly has an enormous impact on response surface method performance. Here, we present a

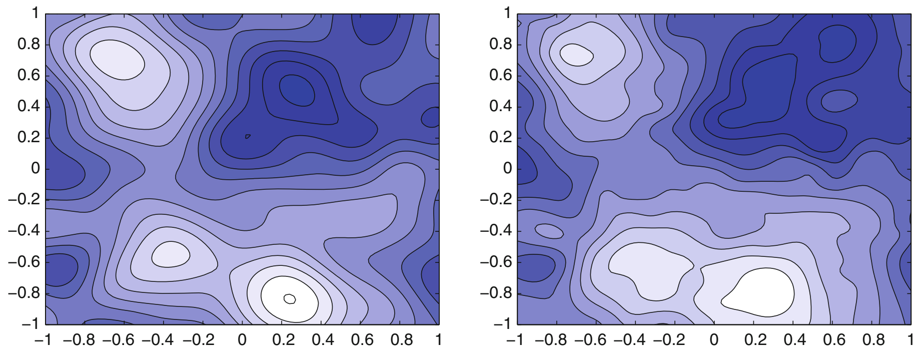


Fig. 4 Two example test functions, one drawn from a GP with a squared exponential kernel, and the other from a GP with a Matérn kernel

modification to the two most common criteria that renders them invariant to shifting and scaling of the objective function. This is both intuitively appealing, since should not care whether our test function is $f(x)$ or $af(x)+b$, and useful for our empirical study, since we can immediately generalize our conclusions to functions that are shifted and scaled versions of our test functions.

There are several choices for the acquisition criterion for Gaussian process optimization, and development and analysis of criteria is ongoing (Srinivas et al. 2010). We have chosen to examine two popular acquisition criteria here: The Maximum Expected Improvement (MEI) criterion (Mockus 1989), which selects the point at the maximum of the Expected Improvement (EI) function,

$$\begin{aligned} \mathrm{EI}(x,\xi) &= \mathrm{E}[(F_x - (\mu_{\max}+\xi))_+] \qquad (19)\\ &= (\mu(x) - (\mu_{\max}+\xi))\Phi\left(\frac{\mu(x)-(\mu_{\max}+\xi)}{\sigma(x)}\right) \qquad (20)\\ &\quad +\sigma(x)\phi\left(\frac{\mu(x)-(\mu_{\max}+\xi)}{\sigma(x)}\right) \qquad (21)\end{aligned}$$

and the Maximum Probability of Improvement (MPI) criterion (Kushner 1964) which selects the point at the maximum of the Probability of Improvement (PI) function,

$$\begin{aligned} \mathrm{PI}(x,\xi) &= P[F_x \geq \mu_{\max}+\xi] \qquad (22)\\ &= 1-\Phi\left(\frac{\mu(x)-(\mu_{\max}+\xi)}{\sigma(x)}\right) \qquad (23)\\ &\equiv \left(\frac{\mu(x)-(\mu_{\max}+\xi)}{\sigma(x)}\right) \qquad (24)\end{aligned}$$

We use $\equiv$ to indicate that the function given by Eq. (24) is equivalent to the MPI criterion, since removing the Φ and negating is a monotonic transformation and therefore the maxima and minima do not change location. Recall that $\mu(x)$ and $\sigma(x)$ are the posterior mean and standard deviation of F_x, and $\mu_{\max} = \max_x \mu(x)$.

Both of these criteria prefer points with higher posterior mean and higher posterior variance, but the tradeoff between these two quantities is qualitatively different for the different criteria. This is tradeoff is controlled by the parameter $\xi > 0$. For PI, the tradeoff is straightforward; for a fixed σ, PI increases linearly in μ. For fixed μ, it increases like $-1/\sigma$ since the numerator in (24) is always negative by definition. This brings up an important point: PI is bounded above by 0, and PI $= 0$ is achieved only when $\xi = 0$ and $\mu(x) = \mu_{\max}$, regardless of the value of $\sigma(x)$. It is easy to see therefore that we can can make MPI purely "greedy" by setting $\xi = 0$. In this case, using MPI will always acquire the maximum point of the posterior mean. This is undesirable as it can lead to the method getting stuck at a local optimum of the objective.

On the other hand, EI is bounded *below* by 0 by definition, and MEI $= 0$ is only attained when $\xi = 0$, $\mu(x) = \mu_{\max}$ *and* $\sigma(x) = 0$—that is, at the current maximum posterior mean in the noise-free setting. Therefore MEI avoids this point and is not "greedy" in this sense even when $\xi = 0$. Furthermore, the tradeoff MEI makes between μ and σ is more complicated than that of MPI, although the marginal preference for higher μ and σ is maintained.

We wish to make our optimization procedure invariant to any additive shift, motivated by the knowledge that such scaling does not change the location of optima of the objective function. We achieve this by choosing the maximum likelihood constant mean: When we do this, the EI and PI functions become invariant to any additive constant in the data, which we will show shortly. We present modifications to the MEI and MPI criteria that make them invariant to multiplicative scaling also.

Recall the likelihood function we are using is given by

$$\log\, p(F_{\boldsymbol{x}} = \boldsymbol{f}) = -\frac{1}{2}(\boldsymbol{f} - \boldsymbol{\mu}_*)^{\mathsf{T}}\mathrm{K}^{-1}(\boldsymbol{f} - \boldsymbol{\mu}_*) - \frac{1}{2}\log|\mathrm{K}| - \frac{n}{2}\log 2\pi$$
$$\boldsymbol{\mu}_* = \left(\frac{\mathbf{1}^{\mathsf{T}}\mathrm{K}^{-1}\boldsymbol{f}}{\mathbf{1}^{\mathsf{T}}\mathrm{K}^{-1}\mathbf{1}}\right)\mathbf{1}$$

and note that if we shift the data by a constant c, we have

$$\boldsymbol{\mu}_*[c] = \left(\frac{\mathbf{1}^{\mathsf{T}}\mathrm{K}^{-1}(\boldsymbol{f} + c\cdot\mathbf{1})}{\mathbf{1}^{\mathsf{T}}\mathrm{K}^{-1}\mathbf{1}}\right)\mathbf{1} \;=\; \left(\frac{\mathbf{1}^{\mathsf{T}}\mathrm{K}^{-1}\boldsymbol{f}}{\mathbf{1}^{\mathsf{T}}\mathrm{K}^{-1}\mathbf{1}} + c\right)\mathbf{1} \;=\; \boldsymbol{\mu}_* + c\cdot\mathbf{1} \tag{25}$$

therefore shifting the data by a constant c shifts the maximum likelihood mean by c, which means that $(\boldsymbol{f} - \boldsymbol{\mu}_*)$ remains the same, as does the rest of the likelihood function. Therefore if we are maximizing likelihood and we shift the data additively, estimates of all the other parameters remain the same, and the posterior mean

$$\mu(x) = \mu^*(x) + k(x, \boldsymbol{z})\;\mathrm{K}^{-1}\;(\boldsymbol{f} - \boldsymbol{\mu}_*)$$

is also shifted by c. The posterior variance function, which does not involve the observed data but only the locations of observations, does not change. Now, note that

$$\mathcal{Z} = \frac{\mu(x) - (\mu_{\max} + \xi)}{\sigma(x)}$$

It is clear now that if we shift the data by c then $\mu(x)$ and $\mu_{\max}$ also shift by c, so we have

$$\begin{aligned}\mathcal{Z}[c] &= \frac{\mu(x) + c - (\mu_{\max} + c + \xi)}{\sigma(x)} \\ &= \frac{\mu(x) - (\mu_{\max} + \xi)}{\sigma(x)} \\ &= \mathcal{Z} \end{aligned} \tag{26}$$

so $\mathcal{Z}$ (and PI) are unchanged by an additive shift in the data. It is also immediately clear that the shifted expected improvement criterion

$$\mathrm{EI}[c] = (\mu(x) + c - (\mu_{\max} + c + \xi))\Phi(\mathcal{Z}) + \sigma(x)\phi(\mathcal{Z}) \tag{27}$$
$$= (\mu(x) - (\mu_{\max} + \xi))\Phi(\mathcal{Z}) + \sigma(x)\phi(\mathcal{Z}) \tag{28}$$
$$= \mathrm{EI}[c] \tag{29}$$

is unaffected. Therefore a global optimization strategy based on a maximum likelihood estimate of the constant mean that uses either of these criteria will select the same points to acquire regardless of any additive shift in the data.

We now consider the effect of a multiplicative scaling $s > 0$ of the data. First, it is easy to see what happens to the original $\boldsymbol{\mu}_*$:

$$\boldsymbol{\mu}_*[s] = \left(\frac{\mathbf{1}^{\mathsf{T}}\mathrm{K}^{-1}(s \cdot \boldsymbol{f})}{\mathbf{1}^{\mathsf{T}}\mathrm{K}^{-1}\mathbf{1}}\right)\mathbf{1} = s \cdot \left(\frac{\mathbf{1}^{\mathsf{T}}\mathrm{K}^{-1}\boldsymbol{f}}{\mathbf{1}^{\mathsf{T}}\mathrm{K}^{-1}\mathbf{1}}\right)\mathbf{1} = s \cdot \boldsymbol{\mu}_* \tag{30}$$

Unsurprisingly, $\boldsymbol{\mu}_*$ is scaled by the amount s. Next we address what happens to the maximum likelihood estimate of the signal variance σ_f^2, which we denote $\hat{\sigma}_f^2$. For convenience, we define the following quantities:

$$\begin{aligned}\rho(x, z) &= \frac{1}{\sigma_f^2}k(x, z) \\ \mathrm{R} &= \frac{1}{\sigma_f^2}k(\boldsymbol{x}, \boldsymbol{x}) = \frac{1}{\sigma_f^2}\mathrm{K}\end{aligned}$$

Note that k depends linearly on σ_f, so ρ does not depend on σ_f. (The kernel k gives the covariance between domain points, and ρ gives the correlation.) Using these definitions, we can write the likelihood function as

$$\log\, p(F_{\boldsymbol{x}} = \boldsymbol{f}) = -\frac{1}{2\sigma_f^2}(\boldsymbol{f} - \boldsymbol{\mu}_*)^{\mathsf{T}}\mathrm{R}^{-1}(\boldsymbol{f} - \boldsymbol{\mu}_*) - \frac{n}{2}\log\sigma_f^2 - \frac{1}{2}\log|\mathrm{R}| - \frac{n}{2}\log 2\pi$$

and, by taking derivatives with respect to σ_f^2 and equating to zero, we can find the maximum likelihood estimate

$$\hat{\sigma}_f^2 = \frac{1}{n}(\boldsymbol{f} - \boldsymbol{\mu}_*)^{\mathsf{T}}\mathrm{R}^{-1}(\boldsymbol{f} - \boldsymbol{\mu}_*) \tag{31}$$

We can substitute this back into the likelihood function to get

$$\log\, p(F_{\boldsymbol{x}} = \boldsymbol{f}) = -\frac{n}{2} - \frac{n}{2}\log\hat{\sigma}_f^2 - \frac{1}{2}\log|\mathrm{R}| - \frac{n}{2}\log 2\pi$$

If we scale the data by a factor s, we have

$$\hat{\sigma}_f^2[s] = \frac{1}{n}(s \cdot \boldsymbol{f} - s \cdot \boldsymbol{\mu}_*)^{\mathsf{T}}\mathrm{R}^{-1}(s \cdot \boldsymbol{f} - s \cdot \boldsymbol{\mu}_*) \tag{32}$$

$$= \frac{1}{n}s^2 \cdot (\boldsymbol{f} - \boldsymbol{\mu}_*)^{\mathsf{T}}\mathrm{R}^{-1}(\boldsymbol{f} - \boldsymbol{\mu}_*) \tag{33}$$

$$= \frac{1}{n}s^2 \cdot \hat{\sigma}_f^2 \tag{34}$$

implying that

$$\begin{aligned}\log\, p(F_{\boldsymbol{x}} = \boldsymbol{f}) &= -\frac{n}{2} - \frac{n}{2}\hat{\sigma}_f^2[s] - \frac{1}{2}\log|\mathrm{R}| - \frac{n}{2}\log 2\pi \\ &= -\frac{n}{2} - \frac{n}{2}\log\frac{s^2 \cdot \hat{\sigma}_f^2}{n} - \frac{1}{2}\log|\mathrm{R}| - \frac{n}{2}\log 2\pi \\ &= -\frac{n}{2} - \frac{n}{2}\log\hat{\sigma}_f^2 - \frac{1}{2}\log|\mathrm{R}| - \frac{n}{2}\log 2\pi - \frac{n}{2}\log s^2 + \frac{n}{2}\log n\end{aligned}$$

and since $-\frac{n}{2}\log s^2$ and $\frac{n}{2}\log n$ are constant with respect to all model parameters, this objective equivalent to the un-scaled objective. This means that when we scale the data, the *only* effect is to scale $\hat{\sigma}_f^2$ and $\boldsymbol{\mu}_*$, since the other kernel parameters (i.e. length scales) only influence the $-\frac{1}{2}\log|\mathrm{R}|$ term, which is unaffected by scaling. In other words, the length scales estimated from the data vector $\boldsymbol{f}$ will be identical to those estimated from data vector $s \cdot \boldsymbol{f}$.

We now turn our attention back to the acquisition criteria, starting with the quantity $\mathcal{Z}$, which can be written

$$\begin{aligned}\mathcal{Z} &= \frac{\mu(x) - (\mu_{\max} + \xi)}{\sigma(x)} \\ &= \frac{(\hat{\sigma}_f^2\rho(x, z))\frac{1}{\hat{\sigma}_f^2}\mathrm{R}^{-1}(\boldsymbol{f} - \boldsymbol{\mu}_*) - (\mu_{\max} + \xi)}{\sqrt{\hat{\sigma}_f^2\rho(x, x) - (\hat{\sigma}_f^2\rho(x, z))\frac{1}{\hat{\sigma}_f^2}\mathrm{R}^{-1}(\hat{\sigma}_f^2\rho(z, x))}} \\ &= \frac{1}{\hat{\sigma}_f} \cdot \frac{\rho(x, z)\mathrm{R}^{-1}(\boldsymbol{f} - \boldsymbol{\mu}_*) - (\mu_{\max} + \xi)}{\sqrt{\rho(x, x) - \rho(x, z)\mathrm{R}^{-1}\rho(z, x)}} \qquad (35)\end{aligned}$$

We arrive at Eq. (35) by substituting the Gaussian process formulae in for $\mu(x)$ and $\sigma(x)$. This illustrates an important point: If the data are scaled by an amount s, then

$$\begin{aligned}\mathcal{Z}[s] &= \frac{1}{s \cdot \hat{\sigma}_f} \cdot \frac{s \cdot \rho(x, z)\mathrm{R}^{-1}(\boldsymbol{f} - \boldsymbol{\mu}_*) - (s \cdot \mu_{\max} + \xi)}{\sqrt{\rho(x, x) - \rho(x, z)\mathrm{R}^{-1}\rho(z, x)}} \\ &= \frac{1}{\hat{\sigma}_f} \cdot \frac{\rho(x, z)\mathrm{R}^{-1}(\boldsymbol{f} - \boldsymbol{\mu}_*) - (\mu_{\max} + \xi/s)}{\sqrt{\rho(x, x) - \rho(x, z)\mathrm{R}^{-1}\rho(z, x)}}\end{aligned}$$

Because of the ξ in the numerator, $\mathcal{Z}$ is *not* invariant to scaling. For example, scaling by a factor $s < 1$ is equivalent to increasing ξ by a factor of $1/s$, meaning that the MPI and MEI criteria will shift in focus toward points of higher variance, even though such a scaling would not change the location of the optimum of the objective function.

We now present a new alternative to $\mathcal{Z}$ that is scale invariant. Consider the following modified objective, denoted $\mathcal{Z}_r$. The subscript r indicates that the objective is "relative" to the scale of the objective function.

$$\begin{aligned}\mathcal{Z}_r &= \frac{\mu(x) - (\mu_{\max} + \hat{\sigma}_f \cdot \xi)}{\sigma(x)} \\ &= \frac{1}{\hat{\sigma}_f} \cdot \frac{\rho(x,z)\mathrm{R}^{-1}(\boldsymbol{f} - \boldsymbol{\mu}_*) - (\mu_{\max} + \hat{\sigma}_f \cdot \xi_r)}{\sqrt{\rho(x,x) - \rho(x,z)\mathrm{R}^{-1}\rho(z,x)}} \end{aligned} \tag{36}$$

Suppose we now scale the data $\boldsymbol{f}$ by an amount s. Then this new objective becomes

$$\begin{aligned}\mathcal{Z}[s] &= \frac{1}{s \cdot \hat{\sigma}_f} \cdot \frac{s \cdot \rho(x,z)\mathrm{R}^{-1}(\boldsymbol{f} - \boldsymbol{\mu}_*) - (s \cdot \mu_{\max} + s \cdot \hat{\sigma}_f \cdot \xi_r)}{\sqrt{\rho(x,x) - \rho(x,z)\mathrm{R}^{-1}\rho(z,x)}} \\ &= \frac{1}{\hat{\sigma}_f} \cdot \frac{\rho(x,z)\mathrm{R}^{-1}(\boldsymbol{f} - \boldsymbol{\mu}_*) - (\mu_{\max} + \hat{\sigma}_f \cdot \xi_r)}{\sqrt{\rho(x,x) - \rho(x,z)\mathrm{R}^{-1}\rho(z,x)}}\end{aligned}$$

therefore changing the scale of the data does not affect the quantity $\mathcal{Z}_r$.

Furthermore, if we modify MEI in a similar way, we obtain

$$\mathrm{EI}_r = (\rho(x,z)\mathrm{R}^{-1}(\boldsymbol{f} - \boldsymbol{\mu}_*) - (\mu_{\max} + \hat{\sigma}_f \cdot \xi_r))\Phi(\mathcal{Z}_r) + \sigma(x)\phi(\mathcal{Z}_r) \tag{37}$$

$$\begin{aligned}&= (\rho(x,z)\mathrm{R}^{-1}(\boldsymbol{f} - \boldsymbol{\mu}_*) - (\mu_{\max} + \hat{\sigma}_f \cdot \xi_r))\Phi(\mathcal{Z}_r) \\ &\quad + (\hat{\sigma}_f \cdot \sqrt{\rho(x,x) - \rho(x,z)\mathrm{R}^{-1}\rho(z,x)})\phi(\mathcal{Z}_r)\end{aligned} \tag{38}$$

and again if we scale the data by s and maximize likelihood, we get

$$\begin{aligned}\mathrm{EI}_r[s] &= (s \cdot \rho(x,z)\mathrm{R}^{-1}(\boldsymbol{f} - \boldsymbol{\mu}_*) - (s \cdot \mu_{\max} + s \cdot \hat{\sigma}_f \cdot \xi_r))\Phi(\mathcal{Z}_r) \\ &\quad + (s \cdot \hat{\sigma}_f \cdot \sqrt{\rho(x,x) - \rho(x,z)\mathrm{R}^{-1}\rho(z,x)})\phi(\mathcal{Z}_r)\end{aligned} \tag{39}$$

$$\begin{aligned}&= s \cdot [(\rho(x,z)\mathrm{R}^{-1}(\boldsymbol{f} - \boldsymbol{\mu}_*)) - (\mu_{\max} + \hat{\sigma}_f \cdot \xi_r)\Phi(\mathcal{Z}_r) \\ &\quad + (\hat{\sigma}_f \cdot \sqrt{\rho(x,x) - \rho(x,z)\mathrm{R}^{-1}\rho(z,x)})\phi(\mathcal{Z}_r)]\end{aligned} \tag{40}$$

$$= s \cdot \mathrm{EI}_r \tag{41}$$

Consequently, while EI_r is not invariant to scaling, the effect of scaling the data by s is simply to scale the EI_r function by the same amount. Therefore the optima of EI_r remain the same, and any strategy relying on maximum likelihood estimates $\boldsymbol{\mu}_*$ and $\hat{\sigma}_f$ and using the EI_r criterion to select points will select the same points regardless of any multiplicative scaling of the objective function. Furthermore, the PI_r and EI_r functions are also invariant to additive shifts by the same arguments used for PI and EI, simply by re-writing $\xi = \hat{\sigma}_f^2 \cdot \xi_r$.

3.2.1 Heuristics for criterion optimization

To use either of these criteria, we must maximize them over the domain of f, and both EI and PI can be complicated, multi-modal functions. For processes with many dimensions, it will be impossible to guarantee finding the global maximum, and various methods have been tried for optimizing surrogates effectively (Bardenet and Kegl 2010). Nevertheless, we can take advantage of properties of these functions to at least find points where the criteria are large, if not maximal.

We note that both criteria can be written as functions of

$$\mathcal{Z} = \frac{\mu(x) - (\mu_{\max} + \xi)}{\sigma(x)} \tag{42}$$

With this definition we have,

$$\mathrm{EI} = (\mu(x) - (\mu_{\max} + \xi))\Phi(\mathcal{Z}) + \sigma(x)\phi(\mathcal{Z}) \tag{43}$$

$$\mathrm{PI} = \mathcal{Z} \tag{44}$$

Each of these functions has properties that make them difficult to maximize in certain situations.

For areas where $\mathcal{Z}$ is small, EI and its gradient become numerically zero because Φ and ϕ fall off exponentially as we decrease $\mathcal{Z}$. In Matlab 2010a, for example, these functions[9] are numerically zero for $\mathcal{Z} \leq -40$. This causes problem when optimizing EI by hill-climbing, since if we start at a point that is very poor then we cannot make any progress.

For PI, the problem is the opposite. Particularly when ξ is small, maxima of $\mathcal{Z}$ can become very flat since it is bounded *above* by 0. From a practical standpoint, this problem is less severe than the problem of maximizing if we fail to accurately find the domain point that maximizes PI, we can definitely make progress from poor points because the function and its gradient are not bounded below, thus allowing us to find a search direction that improves PI from such points.

In view of these properties, we use the following heuristic for finding the MPI point: Since it is easy to make progress from poor points, we evaluate PI at a small number of uniform random domain points (we used 100) and run a quasi-Newton hill-climber from the best of these (we used 5). There are certainly more intelligent and complicated heuristics that could be used here; however we have found that even with this simple approach the PI criterion performs well in experiments; see Sect. 5 for details.

Since EI is sometimes difficult to maximize from very poor points, as described above, we modify the heuristic slightly for this criterion. Again we use a small number of uniform random domain points (again 100) and run a quasi-Newton hill-climber from the best of these (again 5). However, we also always run the hill-climber from the point found by MPI, since this typically has a high $\mathcal{Z}$ and therefore EI has a nonzero gradient, allowing us to improve the point further.

4 Experimental setup

Our experiments are constructed as follows: For each experiment, we fix a *test model*, which is a Gaussian process with fixed parameters and kernel from which we draw multiple *test functions*.

4.1 Models

For the test models, we consider Gaussian processes over a rectangle $\mathcal{X} = [-1, 1]^d$, each with a zero constant prior mean $\mu_0(\boldsymbol{x}) = 0$ and unit signal variance $\sigma_f^2 = 1.0$. We do not vary these parameters because the criteria we are assessing, MEI_r and MPI_r, are invariant to additive shifts and multiplicative scaling. Therefore our test model distribution is completely determined by the kernel of the GP and its length scale parameters.

[9] They are named `normcdf` and `normpdf`, respectively, in this software.

4.1.1 Kernels

The kernels we consider are the squared exponential kernel and its relative, the Matérn kernel with $\nu = 3/2$. (We will henceforth refer to this simply as the "Matérn" kernel.) The main difference between these two kernels is in their smoothness. Two test function examples—one drawn from a process with each type of kernel—are shown in Fig. 4.

The squared exponential kernel produces processes that are mean square continuous and infinitely mean square differentiable, whereas the Matérn kernel with $\nu = 3/2$ produces processes that are mean square continuous but mean square differentiable only once. As such, they represent extremes of this kind of "average smoothness" among processes with twice-differentiable posterior mean and variance functions. (Though the smoothness of the *processes* produced by these two kernels differs in this way, all posterior means and variances are twice differentiable, since these are expressed as weighted sums of kernel functions, and both of these kernel functions are twice differentiable.) Twice-differentiable posterior means and variances are required for our approach, since we use hill-climbing to optimize the MEI_r and MPI_r.

Using EEC as our criterion for selecting test models allows us to generate models that have different length scale configurations—even different dimensions—but that are of similar difficulty. We have chosen to use test kernels with $\mathrm{E}[\chi(\mathcal{A}_{3.0})] = 0.2$. (All EEC values mentioned in this work are in fact $\mathrm{E}[\chi(\mathcal{A}_{3.0})]$, so we will use "EEC" to mean $\mathrm{E}[\chi(\mathcal{A}_{3.0})]$, even though it could be more general.)

Each test function is drawn by choosing 500 uniformly randomly drawn points in $[-1, 1]^d$, sampling function values for these points, and then treating them as observed values of an underlying function. The posterior mean given these function values is the test function. All experiments that have the same kernel and length scales use the same set of 500 test functions.

For each sampled function, we run each algorithm for 30 acquisition steps, starting from the origin.[10] GP-based methods that take a parameter ξ_r have this fixed for the duration of the experiment. We use 6 different test kernels, which generate a total of 3,000 test functions. Each of these is optimized using the MEI_r and MPI_r criteria at 5 different levels of ξ_r each, and using at least 4 different methods for estimating kernel parameters (described below.) This resulted in more than 120,000 optimization test runs.

Performance is measured by absolute error, assuming each algorithm reports the best function value it has observed so far. The true maximum of the test function is estimated by hill-climbing using Newton's method (i.e. with the true Hessian of the test function) starting from the location of the best of the test function's 500 observed function values.

We use test models with different dimensionality, kernel choice, and length scale in our experiments, as described in Table 1. With this set of models, we have tried to cover several different scenarios, some where all dimensions are equally important some where they differ, some with the smooth squared exponential kernel and some with the rougher Matérn kernel. When comparing these characteristics, all of the models have an expected Euler characteristic of 0.2, in an attempt to keep their complexity approximately the same even though their length scales, kernel, and dimension may differ. We also examine the effect of varying EEC between 0.1 and 0.5 when drawing test functions from a prior based on the 2D Squared Exponential Kernel.

[10] Before seeing data, none of these algorithms has a prior preference on where to acquire points. Therefore the first point of each run is at the origin in order to reduce variance across methods.

Table 1 Dimensionality, kernel choice, and log length scales for test models

Dimensions	Kernel	Log length scales
2	Squared exponential	−1.4917, −1.4917
2	Squared exponential	−2.0524, −0.9018
2	Matérn	−0.9424, −0.9424
2	Matérn	−1.5031, −0.3525
8	Squared exponential	−0.3739, −0.3739, −0.3739, 3.0000, 3.0000...
32	Squared exponential	−0.1408, −0.1408, −0.1408, 4.0000, 4.0000...

5 Empirical investigations

We now present the results of our empirical study. We investigate the impact of three factors: the effect of different kernel parameter estimation methods, the effect of exploration as controlled by ξ_r, and the effect of providing gradient information.

5.1 Model parameter estimation

As we noted in Sect. 2.3, kernel parameters are commonly estimated by maximizing the log likelihood of the data. However, researchers have noted problems caused by using maximum likelihood to determine the length scales of Gaussian processes when there are few data points (Jones 2001). Indeed, we will see that optimization procedures that use maximum likelihood model selection can perform *significantly worse* than a uniform random strategy when data are sparse. In this situation, the likelihood function can be very flat along certain dimensions, or worse yet, it can be monotonically increasing in certain directions, causing length scales to head toward zero or infinity. Besides hurting performance, this can also produce a kernel matrix that is numerically ill-conditioned and difficult to work with.

To ensure that maximum likelihood estimation succeeds, previous approaches to response surface optimization (Jones and Schonlau 1998; Sasena 2002) have included an initial pre-acquisition of points in a Latin hypercube design. When using such a design, $10 \cdot d$ points are acquired using Latin hypercube (LHC) sampling before the Gaussian process model and acquisition criterion are used to choose observation points. This ensures there is enough data for the maximum likelihood procedure to perform adequately. This initial design is *not* in any way dependent on incoming data. In such a design, n points are chosen by first dividing each dimension into n bins. For each dimension, the bins are permuted and dimension i of point j is drawn uniformly from within the jth bin in the ith permutation.

We will see from our experiments that this strategy is not necessary to ensure reasonable model fitting, and we will attempt to avoid the expense of pre-acquisition altogether, achieving much better performance over the first $10 \cdot d$ points acquired. To accomplish this, we assume that, *a priori*, one believes that all the variables for a given problem may have some importance; if we did not hold this prior belief, it seems we would not have included those variables in the first place. (Note that it is frequently natural to believe that only a subset of the variables are important, particularly in high dimensions, which motivates techniques like variable selection. However, *a priori*, we typically do not know *which* subset this is.) Our approach is similar in spirit to the penalized kriging models of Li and Sudjianto (2005) in the field of computer experiments, however their work is focused on accurately estimating parameters, and we are concerned primarily with achieving good optimization performance.

We investigate two types of priors that encode the belief that, a priori, all input variables might influence the value of the objective. One prior has a simple preference for moderate length scales, while the other is based on a dimension-independent measure of mean problem difficulty. Note that under these priors, the invariant properties of MPI_r and MEI_r still hold.

5.1.1 MAP-based parameter estimation

We now present two possibilities for priors on model parameters, one that prefers shorter length scales but does not take into account dimensionality, and another that prefers easier problems as measured by the EEC.

Independent Log-Normal Prior This prior is designed simply to prevent estimated length-scales from being very large or very small. For the independent log-normal (ILN) prior, we place a normal distribution with mean 0 and standard deviation 10 on the *logarithm* of each of the length-scales. This is a very vague prior; for example it asserts that there is a 95% probability of finding a length-scale between 7.18×10^{-8} and 1.39×10^{7}. We simply add the logs of these probabilities to the likelihood function (8) to obtain our new objective

$$\log\, p(F_{\mathbf{x}} = \boldsymbol{f}) = -\frac{1}{2}(\boldsymbol{f} - \boldsymbol{\mu}_*)^{\mathsf{T}}\, \mathrm{K}^{-1}\, (\boldsymbol{f} - \boldsymbol{\mu}_*) - \frac{1}{2}\log|\mathrm{K}| - \frac{n}{2}\log 2\pi + \sum_{i=1}^{d}\left(\frac{-\log(\ell_i)^2}{2\cdot 10^2} - \log 10\sqrt{2\pi}\right) \tag{45}$$

Recall that the ℓ_i in Eq. (45) are the length scales from the kernel as defined in Eqs. (4, 5) and (6). These appear explicitly in (45) in the prior part, and implicitly in K. In practice we work with $\log(\ell_i)$ for convenience, which makes the derivatives simpler and allows us to use an unconstrained optimizer. To compute the gradient of (45) with respect to the log length scales, we therefore simply add the gradient of the likelihood from Eq. (10) to the gradient of the last term of (45), which is

$$\frac{\partial}{\partial \log(\ell_j)}\left[\sum_{i=1}^{d}\left(\frac{-\log(\ell_i)^2}{2\cdot 10^2} - \log 10\sqrt{2\pi}\right)\right] = \frac{-\log(\ell_j)}{10^2}$$

We will need this gradient for computing MAP estimates of the ℓ_i, discussed below.

Expected Euler Characteristic Prior This prior is based on the expected Euler characteristic (EEC) of excursion sets drawn from the model GP. The motivation for the expected Euler characteristic prior is based on the desire for a prior that considers "simple" functions more likely *a priori*. This is somewhat true of the independent log normal prior discussed above, but only for functions of few dimensions. For example, a two-dimensional process with log length-scale parameters $\log(\boldsymbol{\ell}) = [0, 0]$ and a ten-dimensional process with parameters $\log(\boldsymbol{\ell}) = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$ are both at the mode of the independent log-normal prior described above, and are both therefore most likely *a priori*. However, the two-dimensional process is much simpler, both intuitively and as measured by expected Euler characteristic. For the two-dimensional process, $\mathrm{E}[\chi(\mathcal{A}_{3.0})] = 0.0070$, whereas for the ten-dimensional process $\mathrm{E}[\chi(\mathcal{A}_{3.0})] = 1.0769$. Figure 5 shows that the difference between these two scenarios is striking.

Our EEC-based prior places most of its mass between these two extremes of complexity. To achieve this, we use a normal prior over $\mathrm{E}[\chi(\mathcal{A}_{3.0})]$ that places about 97% of its mass

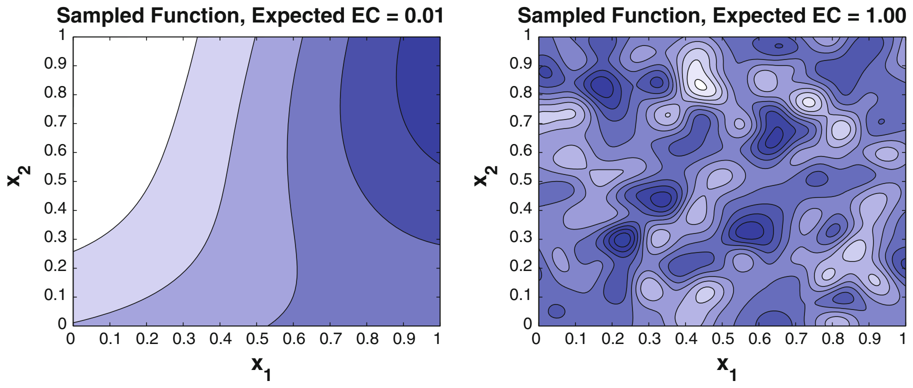


Fig. 5 Contour plots of two sampled functions, one from a GP with $\mathrm{E}[\chi(\mathcal{A}_{3.0})] = 0.01$ and the other with $\mathrm{E}[\chi(\mathcal{A}_{3.0})] = 1.0$. For both, $\mu(x) = 0$ and $\sigma_f^2 = 1.0$

between 0.0 and 0.5 by using parameters $\mu_{\mathrm{eec}} = 0.175$ and $\sigma_{\mathrm{eec}} = 0.0917$. Again, we simply add this penalty to the likelihood function to get

$$\log\ p(F_{\boldsymbol{x}} = \boldsymbol{f}) = -\frac{1}{2}(\boldsymbol{f} - \boldsymbol{\mu}_*)^{\mathsf{T}}\ \mathrm{K}^{-1}\ (\boldsymbol{f} - \boldsymbol{\mu}_*) - \frac{1}{2}\log|\mathrm{K}| - \frac{n}{2}\log 2\pi \qquad (46)$$
$$-\frac{(\mathrm{E}[\chi(\mathcal{A}_{3.0})] - \mu_{\mathrm{eec}})^2}{2\sigma_{\mathrm{eec}}^2} - \log \sigma_{\mathrm{eec}}\sqrt{2\pi}$$

and again, the length scales from the kernels defined in Eqs. 5 and 6 will allow us to compute both the log likelihood and its gradient, as well as $\mathrm{E}[\chi(\mathcal{A}_{3.0})]$ and its gradient.

Figure 6 illustrates both the ILN prior and the EEC prior. While both priors prefer log length-scales near the origin, the EEC prior is also willing to allow one length scale to become shorter if the other is made longer. This is because if we fix the number of dimensions of the process, there are sub-manifolds of length-scales that produce processes with the same EEC. The level curves of the EEC prior in Fig. 6 show some examples of these sub-manifolds for two-dimensional processes; in d dimensions, these manifolds will be $d-1$ dimensional.

These two objective functions give us a mechanism for estimating the relevant kernel parameters (i.e. length scales) from data. To do so, we simply maximize (45) or (46) with respect to the ℓ_i, and then use the resulting parameters in our posterior computations. To maximize these criteria, we use an unconstrained quasi-Newton method, starting from the mode of the prior. Specifically, we use the BFGS method as implemented in Matlab 2010a. The method is provided with the value and gradient of the objective function, which we compute analytically as shown above. Examples of the impact of these priors on the estimated posterior mean function are shown in Fig. 6. One can see that both of these priors produce reasonable length scales when there are only a few data points.

5.1.2 The effect of parameter priors on performance

A summary of the performance of these different parameter estimation methods is shown in Figs. 7, 8 and 9. Each chart compares the performance of MEI_r, MPI_r and Latin hypercube

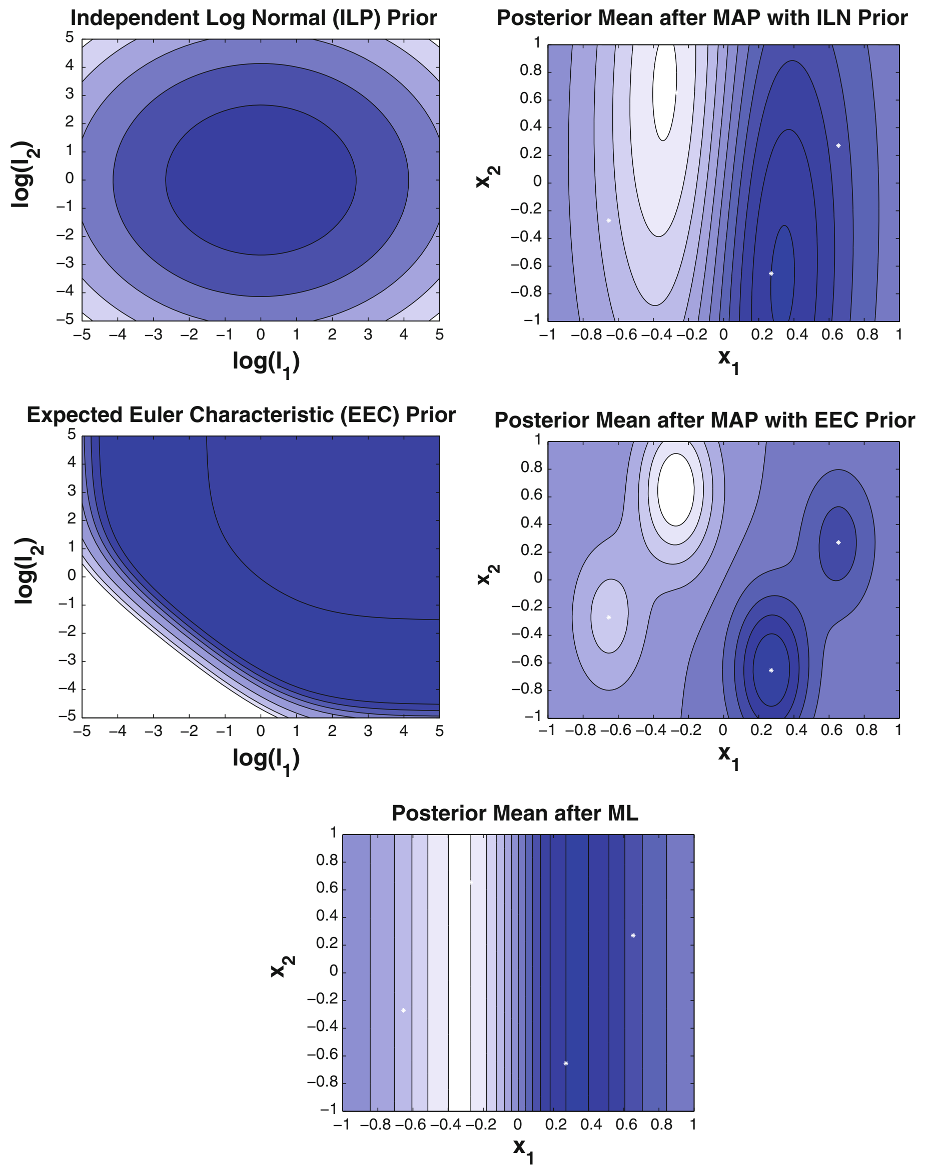
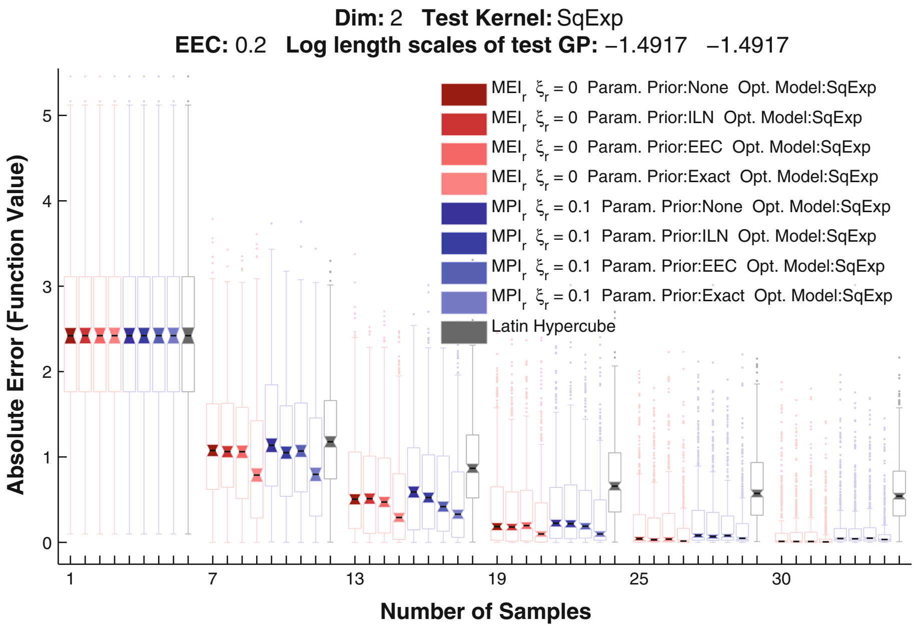


Fig. 6 Contour plots of the independent log-normal prior and the expected Euler characteristic prior used in this work, along with plots of the resulting posterior means. Data points are indicated in the posterior mean plots by small *white* $*$. From the lower-left heading clockwise, the function values are -0.5, -1.0, 0.5, 1.0 in all cases. The last plot shows the posterior mean after maximum likelihood estimation. Note that the vertical length scale has effectively gone to infinity in this case, meaning that the resulting model explains the variability in the function values using only the horizontal dimension. The ILN or EEC priors penalize having very large length scales, and so produce moderate length scales for both dimensions

sampling after different numbers of acquisitions. The boxplots describe the distribution of the absolute error achieved by each method combination of prior and acquisition criterion. The “Prior:None” case corresponds to maximum likelihood estimation, and the “Prior:Exact”



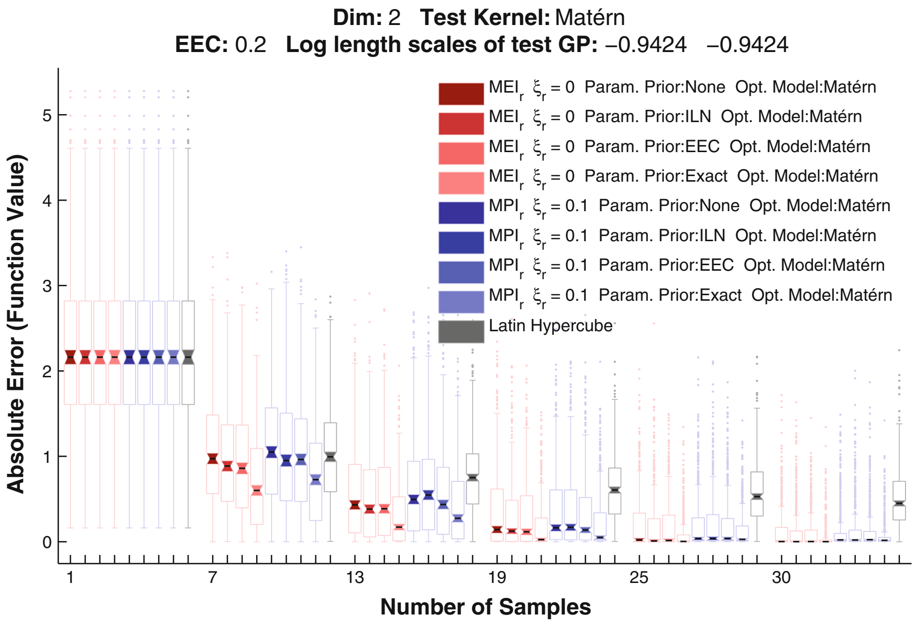
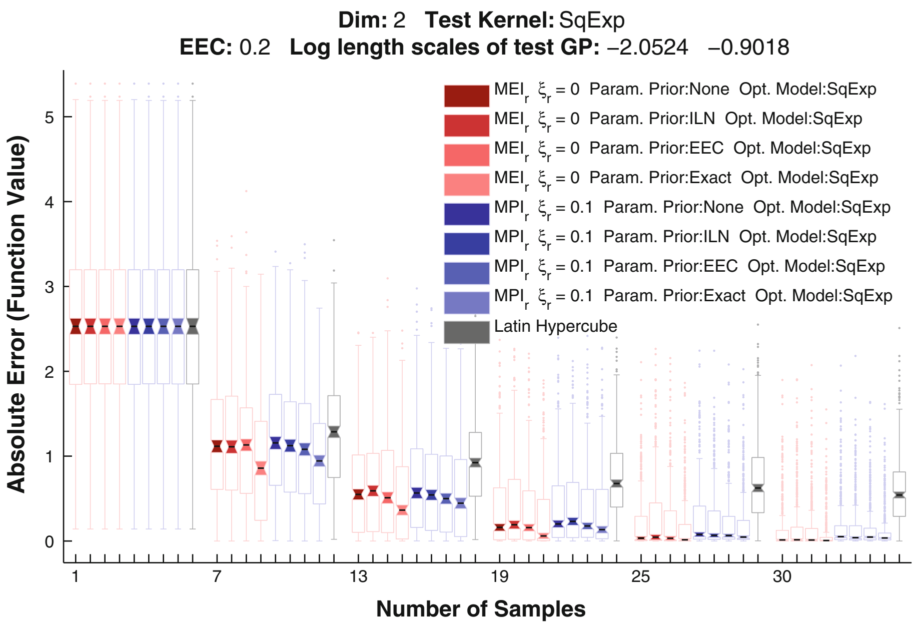


Fig. 7 Performance of MEIr and MPIr using MAP estimation with different parameter priors. Groups of boxplots show absolute error after 1, 7, 13, 19, 25, and 30 objective function evaluations. Within each group, the left-to-right order of the boxes matches the top-to-bottom order of the legend

case assumes the uses the correct kernel parameters throughout. To simplify, we will show only $\xi_r = 0$ for MEI$_r$ and $\xi_r = 0.1$ for MPI$_r$ in our comparisons. (We will see later that these are good values in all of the situations we tested).



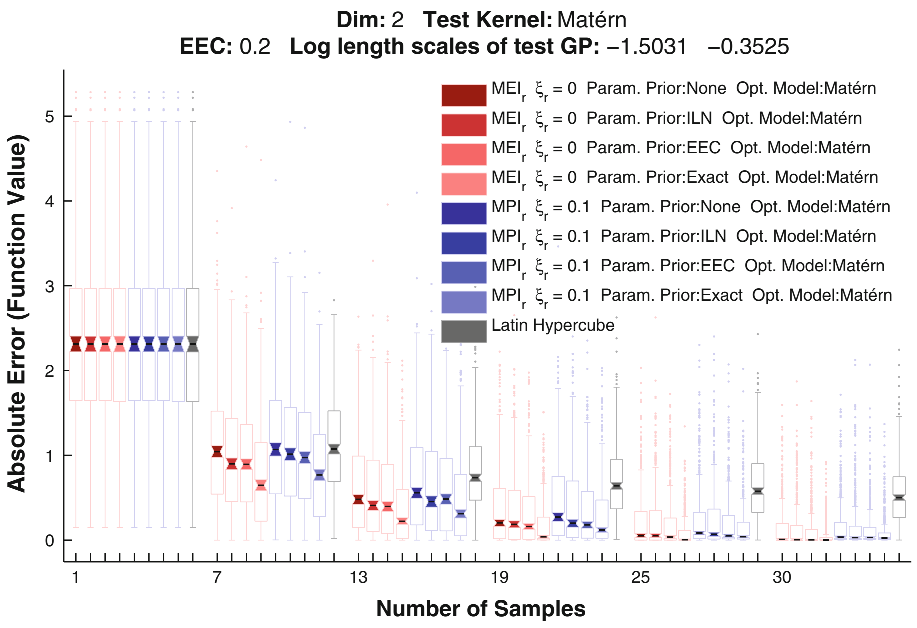
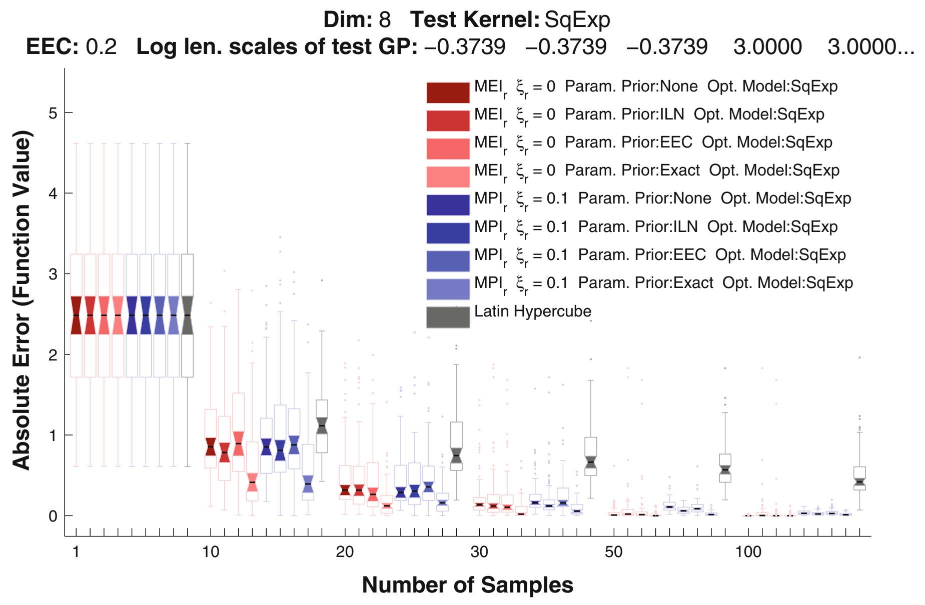


Fig. 8 Performance of MEIr and MPIr using MAP estimation with different parameter priors. Groups of boxplots show absolute error after 1, 7, 13, 19, 25, and 30 objective function evaluations. Within each group, the left-to-right order of the boxes matches the top-to-bottom order of the legend

First, we note that in all cases, knowing the correct kernel parameters gives the best performance, and there are varying degrees of performance loss associated with having to estimate them depending on the situation. However, as the number of acquisitions increases,



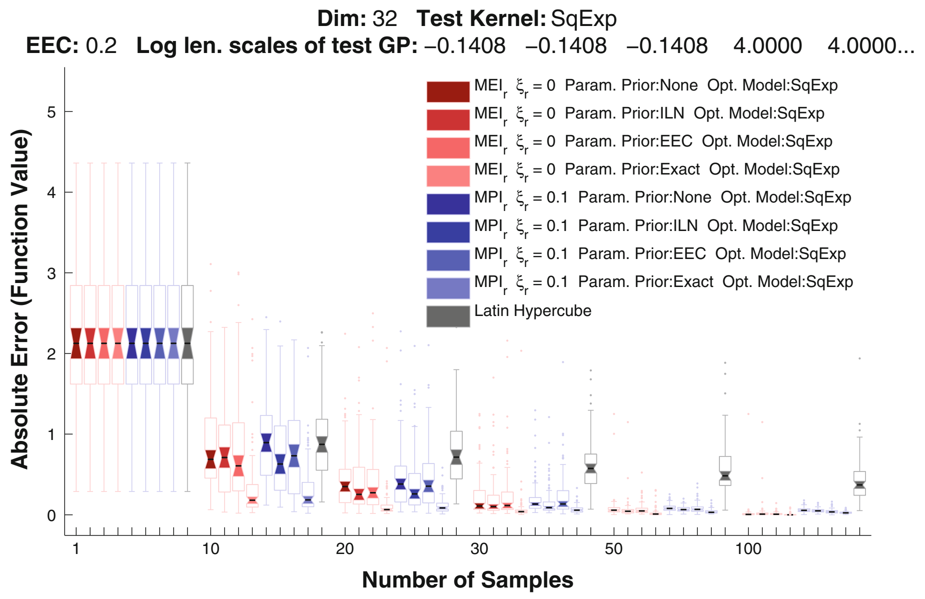


Fig. 9 Performance of MEI$_r$ and MPI$_r$ using MAP estimation with different parameter priors. Groups of boxplots show absolute error after 1, 7, 13, 19, 25, and 30 objective function evaluations. Within each group, the left-to-right order of the boxes matches the top-to-bottom order of the legend

performance improves to near that achieved by using the the correct model parameters, i.e. when we have "exact" priors. Also, in most cases, performance earlier on is no worse than acquisition on a random Latin hypercube, and can be much better. This is significant because in previous work Latin hypercube acquisition is commonly used for the first $10 \cdot d$ acquired points *before* acquiring using MPI or MEI begins. Our results show that even using maximum likelihood (i.e. with no prior on parameters) from the very beginning, we can achieve better performance in fewer function evaluations. Note however that we cannot say what the effect of an initial LHC is on much longer sampling runs intended to acquire a very precise optimal value—it may be that in this case an initial LHC design is preferable.

Furthermore, there are cases where maximum likelihood estimation can perform as poorly as randomly choosing domain point. We hypothesize that this is because some of our test models have quite short length scales, since we have found that maximum likelihood estimation tends to generate very long length scales. Introducing an ILN or EEC prior on length scales can mitigate this problem. Performance using the EEC prior is similar to using ILN, being slightly worse or slightly better depending on the situation. In most cases, including the high and low dimensional settings, performance when using MAP estimation with ILN priors is not significantly worse than using no priors at all, and is significantly better during initial sampling in the Matérn kernel example of Fig. 8. Using the ILN or EEC prior when estimating length scales seems to be a "safe" choice.

5.2 The effect of exploration

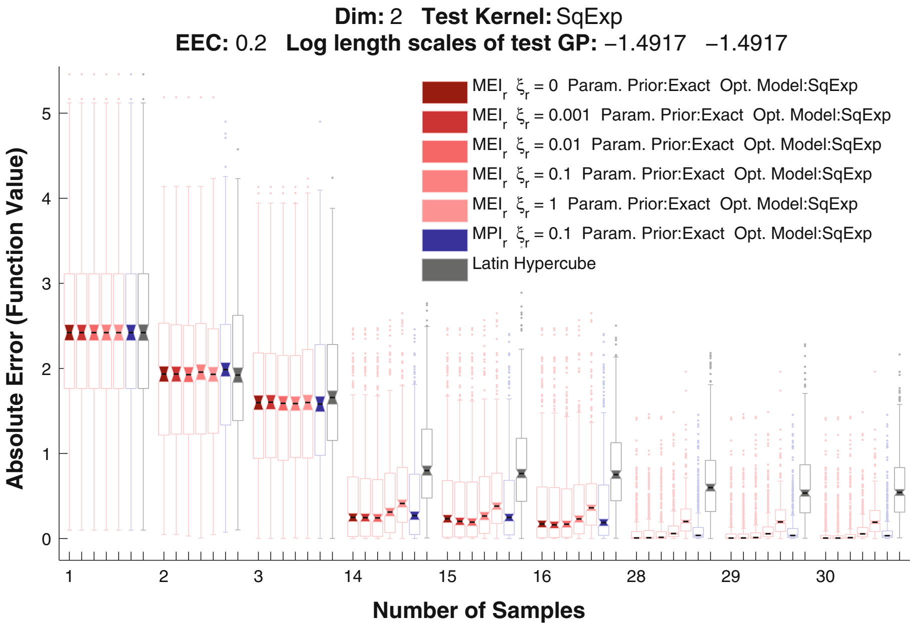
We now turn our attention to the effect of ξ_r on optimization performance. We evaluate the performance of MEI_r and MPI_r on each of the test models at a variety of ξ_r values, with parameters coming from one of four possible estimation strategies. We give a brief summary of the results rather than presenting them explicitly. There is a striking uniformity among all of these results: In all cases, using MEI_r with $\xi_r = 0$ is never significantly worse than the choice that gives the best median performance. However, we noted that the optimal value of ξ_r for the MPI_r criterion seems to vary between 0.1 and 1.0 depending on how many acquisitions are made, though $\xi_r = 0.1$ worked reasonably well in all cases.

As an example of this trend, Fig. 10 shows the results from one test model for the case where we do not do any parameter estimation, but fix all the kernel parameters to their correct values. The first three, middle three, and last three acquisitions from five different experiments are shown. The performance of independently acquiring points using a Latin hypercube is shown for comparison.

The observation that the best value of ξ_r for MEI_r is independent of the number of acquisitions runs completely counter to conventional wisdom, which would claim that larger values of ξ_r would perform better early on and smaller values would perform better later. Indeed, varying the exploration parameter according to the number of samples acquired been shown to be useful in some cases (Schonlau 1997; Sasena 2002); however, these conclusions were drawn using a small number of test functions. Our evidence is certainly not proof that scheduling ξ_r would not improve performance, particularly for MPI_r where there seems to be a slight need to explore more early on; nevertheless, it does not suggest that there are significant average performance gains to be made by scheduling ξ_r. Furthermore, the performance of MEI_r and MPI_r using the best ξ_r for each are similar; therefore we believe that in most cases they are both reasonable choices.

5.2.1 Effect of varying problem difficulty

Consistent good performance at fixed ξ_r appears to hold even as problem difficulty changes. In Figs. 11 and 12, we see that whether we draw functions from a prior with EEC $= 0.1$ or EEC $= 0.5$, the optimal amount of exploration as governed by ξ_r remains basically the same, for both MEI_r and MPI_r. We have found this to be true for EEC as large as 1.0, and



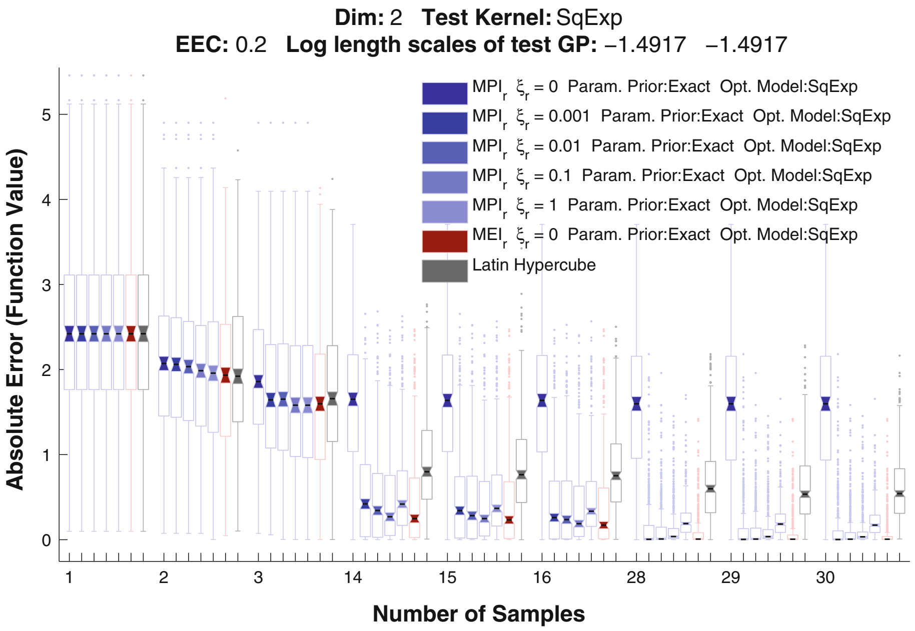
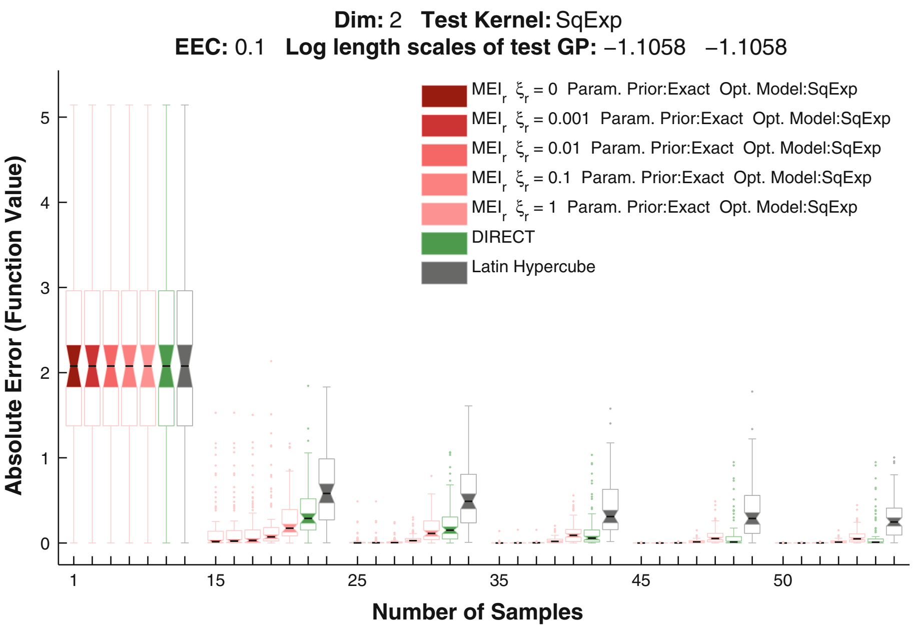


Fig. 10 Effect of ξ_r on the performance of MEI and MPI when priors are correct and fixed. For MEI, using $\xi_r = 0$ is never significantly worse than the choice that gives the best median performance. For MPI, after 30 acquisitions, using $\xi_r = 0.1$ provides the best performance. In some cases, however, performance is improved by using larger ξ_r for smaller numbers of function evaluations. Within each group of boxplots, the left-to-right order of the boxes matches the top-to-bottom order of the legend



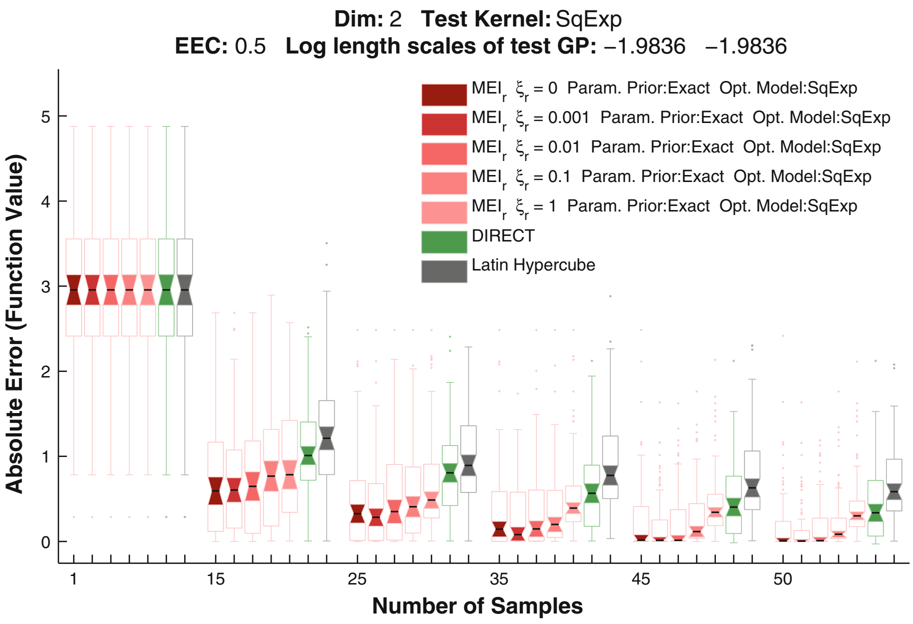
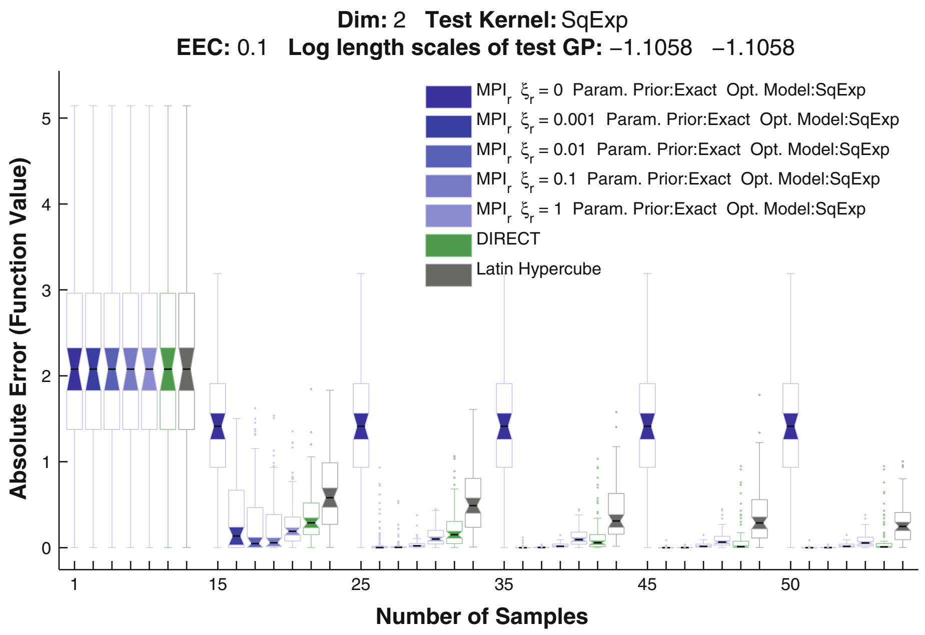


Fig. 11 Effect of ξ_r for on the performance of MEI when priors are correct and fixed for different problem difficulties: The *top graph* shows results on problems with EEC $= 0.1$, and the *bottom* shows results on problems with EEC $= 0.5$. For MEI, using $\xi_r = 0$ is never significantly worse than the choice that gives the best median performance. Within each group of boxplots, the left-to-right order of the boxes matches the top-to-bottom order of the legend



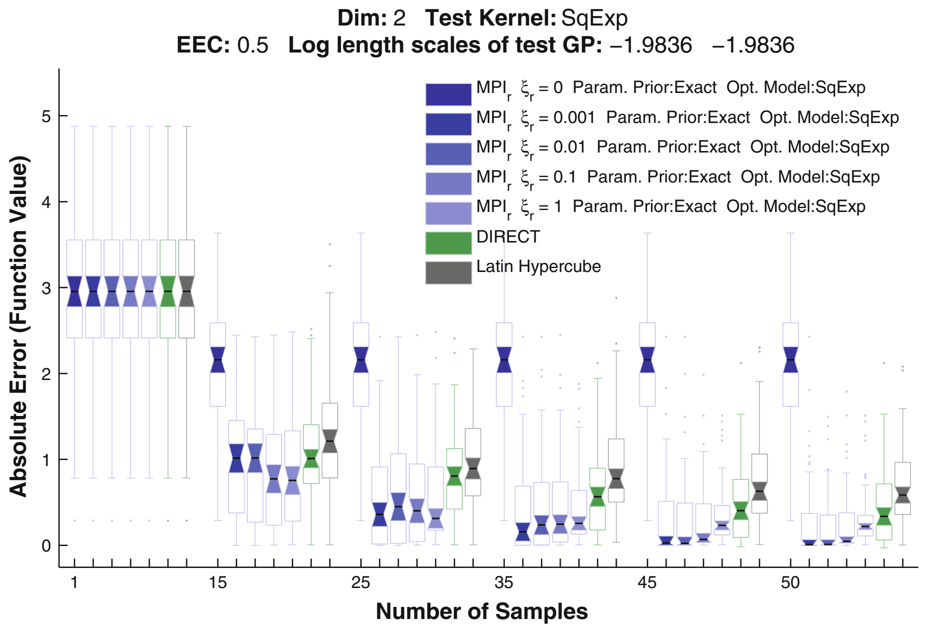


Fig. 12 Effect of ξ_r for on the performance of MPI when priors are correct and fixed for different problem difficulties: The *top graph* shows results on problems with EEC $= 0.1$, and the *bottom* shows results on problems with EEC $= 0.5$. Within each group of boxplots, the left-to-right order of the boxes matches the top-to-bottom order of the legend. For MPI, using $\xi_r = 0.1$ is never significantly worse than the choice that gives the best median performance

for all four possible priors on length scale we considered, i.e. the ILN prior, the EEC prior, no prior (maximum likelihood) and the 'exact' prior where parameters are assumed known.

5.3 Gradient information

All the experiments discussed so far assume that we can only observe the value of $f(x)$ at the points we choose. However, the underlying Gaussian process models we use can condition on gradient information when such information is available, because the gradient observations are jointly Gaussian with the function evaluations and with each other. Thus the optimization procedure remains the same, but each time we acquire the objective $f(x)$, we also acquire its gradient vector $\nabla f(x)$ at the domain point x. The $d+1$ resulting values are all incorporated in to our response surface model before acquiring the next point; this results in a kernel matrix of size $n \cdot (d+1) \times n \cdot (d+1)$ after acquiring n points. This additional information results in a higher fidelity model and improves optimization performance. For details, we refer the reader to an overview by Rasmussen and Williams (2006c), and an example in the response-surface optimization context by Leary et al. (2004). We can use all of the same techniques for estimating kernel parameters and for choosing acquisition points that we used previously, augment the model with gradient information, and explore the effects of different ξ_r and different parameter priors.

Using gradient information in this way, we again found that the best fixed ξ_r for the MEI$_r$ criterion does not vary significantly regardless of the number of acquisitions, and again we use a fixed value of $\xi_r = 0$ for all of our comparisons, which is the same as the value we chose for our function-value-only experiments. However, we found that when using gradient information, the MPI$_r$ criterion seems to have more variable performance depending on the test kernel and the method of estimating parameters. The best choice of ξ_r ranges from 0.01 to 1.0. Despite the apparent sensitivity of MPI$_r$, it appears that MEI$_r$ performs well at $\xi_r = 0$ regardless of test model or parameters, and when estimating parameters performs slightly better with an EEC prior than with an ILN prior or none at all. Boxplots of the performance of MEI$_r$ using one of the test kernels are shown in Fig. 13.

As another point of comparison, we include data on the performance of using a hill-climber with random restarts *applied directly to the objective function.*[11] This is an optimization strategy that is very common in practice when gradient information is available. To make a comparison with our response-surface approach, we restrict the random-restart hillclimber to 30 function-plus-gradient observations. The procedure is as follows: Starting from the origin, the BFGS hill-climbing algorithm is run using the objective function $f(x)$ as input. If the hill-climber converges to a local optimum before its allotted function-plus-gradient acquisitions are expended, it is restarted at a uniformly randomly drawn point in the domain. This continues until 30 function-plus-gradient acquisitions have been made. The reported value for BFGS after k acquisitions is the best observed value up to that point. Performance of this approach is denoted BFGS$_{\text{RESTARTS}}$ and is shown in green box plots.

The performance of MEI$_r$ on test functions is better than BFGS after only a few function evaluations, and is soon very far ahead: After 15 function evaluations in both test cases of 500 functions each, 75% of the values found by MEI$_r$ are better than the median performance of BFGS$_{\text{RESTARTS}}$. This is a stunning improvement over a method that is widely considered state-of-the-art, albeit at a significantly higher computational overhead—recall that our method actually calls BFGS multiple times as a subroutine. Nonetheless, for

[11] Recall we also use a hill-climber as a subroutine in our response-surface method to optimize the acquisition criterion, but it does not access $f(x)$ directly.

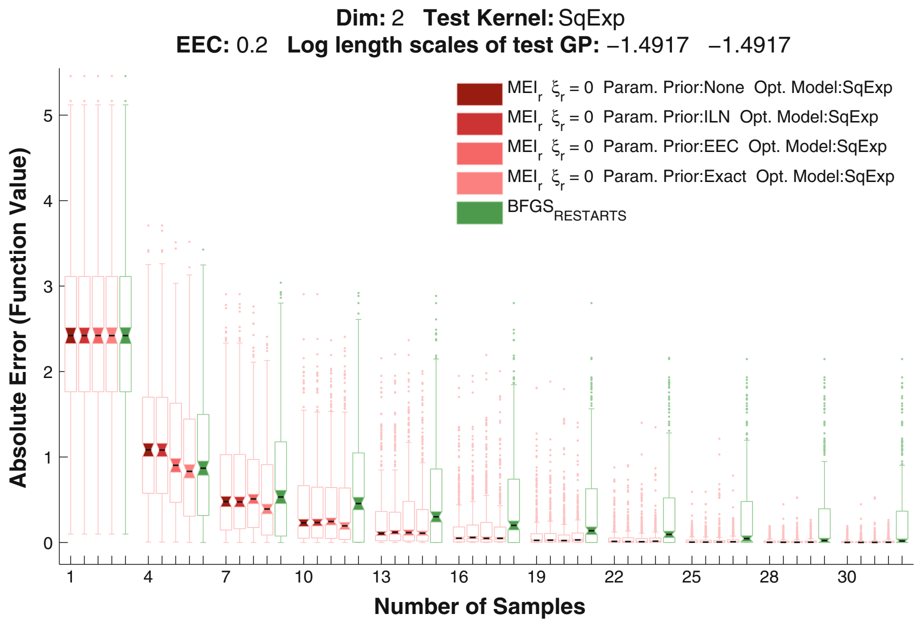


Fig. 13 Performance of MEI_r and MPI_r with gradient information, with performance of the BFGS method with random restarts for comparison. Within each group of boxplots, the left-to-right order of the boxes matches the top-to-bottom order of the legend

expensive functions, expending the extra effort in order to make fewer function evaluations may be an attractive tradeoff.

6 Conclusion

We have argued for a more rigorous and informative approach to the evaluation of response surface methods by introducing an experimental methodology that uses a much larger set of test functions than previous work. We have shown how to generate test functions that are interesting and yet tractable, and we have shown how to modify commonly-used acquisition criteria to help make the conclusions drawn from these experiments more widely applicable.

We have also presented results from the first study of this kind, and have made three surprising discoveries:

- If the optimization budget is limited and only a small number of points (i.e. $< 10 \cdot d$) are to be acquired, an initial Latin hypercube acquisition is unnecessary. Response surface optimization achieves performance exceeding that of LHC acquisition in fewer than the $10 \cdot d$ function evaluations typically allocated to the initial LHC sampling. It is advisable to use an ILN or other prior to ensure that reasonable length scales are estimated when data are few.
- Performance for MEI_r is good at $\xi_r = 0$, and performance of MPI_r is good at $\xi_r = 0.1$. This is largely independent of the number of samples acquired by the methods, the dimension of the test problems, and the difficulty of the test problems as measured by EEC.
- When using gradient information, MEI_r is preferable to MPI_r, since it is more robust to different ξ_r and models. Here again, $\xi_r = 0$ was effective for MEI_r. In terms of number of

function evaluations, response surface methods perform better than state-of-the-art local search combined with random restarts.

Our test functions are non-concave and multi-modal, much like the standard synthetic test functions that have been used in the past, and yet are much more diverse due to their sheer numbers. However, we acknowledge that our conclusions are limited by our choice of optimization setting. We have restricted ourselves to deterministic functions whose domains are box-constrained subsets of $\mathbb{R}^d$. Some may feel that the test problems we chose are not representative of the real-world problems they are interested in.

For example, one might wish to impose general constraints of the form $g(x) < 0$ on the solution, where $g(x)$ is also modelled using a Gaussian process (Schonlau 1997; Audet et al. 2000; Sasena 2002). In this case, we would suggest placing a prior on g and then sampling a constraint function g for each test problem just as we sampled an objective f for each problem. Note that the feasible set $\{x : g(x) < 0\}$ is precisely the excursion set of the function $-g(x)$ at level $u = 0$, $\mathcal{A}_0(-g)$ in our notation. However, it is not clear that the Euler characteristic of $\mathcal{A}_0(-g)$ is a good measure of the difficulty of solving a problem subject to $g(x) < 0$, since it is related to the number of connected components of $\mathcal{A}_0(-g)$ and says nothing about the boundary $\partial\mathcal{A}_0(-g)$. Intuitively, one would expect optimizing subject to a constraint set that only has one connected component might still be very difficult if its boundary is very complex. Thus going forward it will be necessary to identify what priors on g generate constraint functions that have an appropriate level of test problem difficulty.

The Gaussian process model is very flexible, and it is likely that among the many choices of kernel and prior mean, a researcher can find a process that generates functions she or he finds interesting. Our experimental methodology is more flexible and rigorous than the prevailing approach, and will allow future work to more completely explore the potential of response surface methods for global optimization.

Acknowledgements Daniel J. Lizotte, Dale Schuurmans and Russell Greiner were supported by the Natural Sciences and Engineering Research Council of Canada. Daniel Lizotte was also supported by National Institutes of Health grant R01 MH080015.

Appendix A: Proof of theorem 2

Proof First, note that the denominator inside the inner sum of (13) does not depend on $\mathcal{J}$, and so can immediately be factored out. This leaves us with the task of computing

$$\sum_{\mathcal{J}\in\mathcal{E}_k}\left(\left[\prod_{i\in\mathcal{D}(\mathcal{J})}\sqrt{\lambda_{ii}}\right]\cdot\left[\prod_{i\in\mathcal{D}(\mathcal{J})}w_i\right]\right)=\sum_{\mathcal{J}\in\mathcal{E}_k}\left(\prod_{i\in\mathcal{D}(\mathcal{J})}w_i\sqrt{\lambda_{ii}}\right)=\sum_{\mathcal{J}\in\mathcal{E}_k}\left(\prod_{i\in\mathcal{D}(\mathcal{J})}q_i\right)$$

for each k, where $q_i = w_i\sqrt{\lambda_{ii}}$. The sum is over all subsets of size k of the index set $\{1, 2, ..., d\}$, so for each k the sum has $\binom{d}{k}$ terms that cover all possible monomials of degree k formed from $\{q_1, q_2, ..., q_d\}$ that have *no exponent greater than 1*.

The sequence S_k constructs these sums as follows: at the jth step in the sum in (15) to construct S_k, all monomials of degree j composed of j distinct variables are generated, along with some extra monomials that have a $j + 1$ in the exponent. These are then cancelled by part of the $(j + 1)$st term, but this in turn introduces some monomials with a $j + 2$ in the exponent, which are cancelled by the term after that, etc. Finally for $j = k$, we remove the last of the extra terms by adding in $(-1)^{k+1}S^{[k]}$. The amount of computation can be reduced by a constant factor by noting that S_d can be computed in linear time—it is simply $\prod_{i=1}^{d} w_i\sqrt{\lambda_{ii}}$. In practice, we also use the following sequence

$$S_k = \frac{\sum_{j=k}^{d-1}(-1)^{j-k} S_{j+1} \cdot S^{[-(j-k+1)]}}{d-k}$$
$$S_d = \prod_{i=1}^{d} w_i \sqrt{\lambda_{ii}}$$
$$S^{[-j]} = \sum_{i=1}^{d} (w_i \sqrt{\lambda_{ii}})^{-j}$$

which begins with S_d and successively removes variables from each monomial, thus introducing terms with extra negative powers, which are removed in a manner analogous to the previous sequence. We can save some time by using both sequences and "meeting in the middle"—we use the first sequence to compute $S_0 \ldots S_{d/2}$ and the second to compute $S_{d/2+1} \ldots S_d$. $\square$

Appendix B: Probabilist's Hermite polynomials

We define $H_k(x)$ to be the kth "probabilist's" Hermite polynomial. Some properties of $H_k(x)$ include:

$$H_k(x) = (-1)^k \cdot e^{x^2/2} \cdot \frac{d^k}{dx^k} e^{-x^2/2}$$
$$H_0(x) = 1$$
$$H_1(x) = x$$
$$H_{k+1}(x) = xH_k(x) - kH_{k-1}(x)$$

References

Adler, R.J., Taylor, J.E.: Random Fields and Geometry. Springer, Berlin (2007)

Audet, C., Dennis, J.J.E., Moore, D., Booker, A., Frank, P.: A surrogate-model-based method for constrained optimization. In: Proceedings of the 8th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization. Paper No. AIAA-2000-4891 (2000)

Bakker, T.M.D.: Design Optimization With Kriging Models. Delft University Press, Delft (2000)

Bardenet, R., Kegl, B.: Surrogating the surrogate: accelerating Gaussian-process-based global optimization with a mixture cross-entropy algorithm. In: Proceedings of the 27th Annual International Conference on Machine Learning (2010)

Boyle, P.: Gaussian processes for regression and optimisation. PhD thesis, Victoria University of Wellington (2006)

Chernova, S., Veloso, M.: An evolutionary approach to gait learning for four-legged robots. In: Intelligent Robots and Systems (2004)

Cox, D.D., John, S. SDO: a statistical method for global optimization. In: Multidisciplinary design optimization (Hampton, VA, 1995), pp. 315–329. SIAM, Philadelphia, PA (1997)

Csató, L., Opper, M.: Sparse representation for Gaussian process models. In: Leen, T.K., Dietterich, T.G., Tresp, V. (eds.) Neural Information Processing Systems, vol. 13, The MIT Press, Cambridge, MA (2001)

Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multiobjective optimization. In: Jain, L., Wu, X., Abraham, A., Goldberg, R. (eds.) Evolutionary Multiobjective Optimization., pp. 105–145. Springer, Berlin (2005)

Dixon, L., Szegö, G. (eds.): Towards Global Optimisation, Chapter The global optimisation problem: an introduction, vol. 2, pp. 1–15. North-Holland, Amsterdam (1978)

Elder J.F.: Global $\mathbb{R}^d$ optimization when probes are expensive: the GROPE algorithm. In: Proceedings IEEE International Conference on Systems, Man, and Cybernetics, Chicago, Illinois (1992)

Floudas, C.A., Gounaris, C.E.: A review of recent advances in global optimization. J. Glob. Optim. **45**, 3–38 (2009)

Gutmann, H.-M.: A radial basis function method for global optimization. J. Glob. Optim. **19**, 201–227 (2001)

Huang, D., Allen, T.T., Notz, W.I., Zeng, N.: Global optimization of stochastic black-box systems via sequential kriging meta-models. J. Glob. Optim. **34**, 441–466 (2006)

Jones, D., Perttunen, C., Stuckman, B.: Lipschitzian optimization without the lipschitz constant. Optim. Theory Appl. **79**(1), 157–181 (1993)

Jones, D.R.: A taxonomy of global optimization methods based on response surfaces. J. Glob. Optim. **21**, 345–383 (2001)

Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. J. Glob. Optim. **13**, 455–492 (1998)

Khare, V., Yao, X., Deb, K.: Performance scaling of multi-objective evolutionary algorithms. In: Evolutionary Multi-Criterion Optimization, volume 2632/2003 of Lecture Notes in Computing Science. Springer, Berlin (2003)

Kushner, H.: A new method of locating the maximum of an arbitrary multipeak curve in the presence of noise. J. Basic Eng. **86**, 97–106 (1964)

Lawrence, N., Seeger, M., Herbrich, R. : Fast sparse gaussian process methods: the informative vector machine. In: Becker, S.T.S., Obermayer, K. (eds.) Advances in Neural Information Processing Systems, vol. 15, pp. 609–616. MIT Press, Cambridge, MA (2003)

Leary, S.J., Bhaskar, A., Keane, A.J.: A derivative based surrogate model for approximating and optimizing the output of an expensive computer simulation. J. Glob. Optim. **30**, 39–58 (2004)

Li, R., Sudjianto, A.: Analysis of computer experiments using penalized likelihood in Gaussian kriging models. Technometrics **47**, 111–120 (2005)

Lizotte, D.J.: Practical Bayesian optimization. PhD thesis, University of Alberta, Alberta (2008)

Lizotte, D.J., Wang, T., Bowling, M., Schuurmans, D.: Automatic gait optimization with Gaussian process regression. In: Proceedings of the International Joint Conference on Artificial Intelligence (2007)

MATLAB: Version 7.10.0 (R2010a). The MathWorks Inc., Natick, MA (2010)

Mockus, J.: Bayesian Approach to Global Optimization. Kluwer Academic Publishers, Dordrecht (1989)

Perttunen, C. D.: A nonparametric global optimization method using the rank transformation. In: Proceedings of the IEEE Conference on Systems, Man, and Cybernetics, vol. 1, pp. 888–893 (2007)

Rasmussen, C.E.: Gaussian processes in machine learning. In: Bousquet, O., Luxburg, U.von , Rätsch, G. (eds.) Advanced Lectures in Machine Learning: ML Summer Schools 2003, Springer, Berlin (2004)

Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning, chapter 9.4 Derivative Observations. MIT Press, Cambridge (2006a)

Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning, chapter 5 Model Selection and Adaptation of Hyperparameters. MIT Press, Cambridge (2006b)

Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning, chapter 4.2 Examples of Covariance Functions. MIT Press, Cambridge (2006c)

Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning, chapter 2 Regression. MIT Press, Cambridge (2006d)

Regis, C.A.S.R.G.: Improved strategies for radial basis function methods for global optimization. J. Glob. Optim. **37**(1), 113–135 (2007)

Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Stat. Sci. **4**,409–435 (1989)

Santner, T.J., Williams, B.J., Notz, W.I.: The Design and Analysis of Computer Experiments. Springer, Berlin (2003)

Sasena, M.J.: Flexibility and efficiency enhancements for constrained global design optimization with Kriging approximations. PhD thesis, University of Michigan, Ann Arbor (2002)

Schonlau, M.: Computer experiments and global optimization. PhD thesis, University of Waterloo, Waterloo (1997)

Shawe-Taylor, J., Cristiani, N.: Kernel methods for pattern analysis, chapter Properties of Kernels. MIT Press, Cambridge (2004)

Srinivas, N., Krause, A., Kakade, S., Seeger, M.: Gaussian process optimization in the bandit setting: No regret and experimental design. In: Proceedings of the 27th Annual International Conference on Machine Learning (2010)

Stuckman, B.: A global search method for optimizing nonlinear systems. In: Proceedings of the IEEE Conference on Systems, Man, and Cybernetics, vol. 1, pp 965–977 (1989)