A sequential Monte Carlo approach to Thompson sampling for Bayesian optimization

Hildo Bijl, Thomas B. Schönh, Jan-Willem van Wingerden, and Michel Verhaegen

Abstract—Bayesian optimization through Gaussian process regression is an effective method of optimizing an unknown function for which every measurement is expensive. It approximates the objective function and then recommends a new measurement point to try out. This recommendation is usually selected by optimizing a given acquisition function. After a sufficient number of measurements, a recommendation about the maximum is made. However, a key realization is that the maximum of a Gaussian process is not a deterministic point, but a random variable with a distribution of its own. This distribution cannot be calculated analytically. Our main contribution is a sequential Monte Carlo approach towards approximating this maximum distribution. Subsequently, by taking samples from this distribution, we enable Thompson sampling to be applied to (armed-bandit) optimization problems with a continuous input space. All this is done without requiring the optimization of a nonlinear acquisition function. Experiments have shown that the resulting optimization method has a competitive performance at keeping the cumulative regret limited.

Index Terms—Gaussian processes, optimization methods, particle methods, model-free, controller tuning.

I. INTRODUCTION

Consider the problem of maximizing a continuous nonlinear reward function $f(x)$ (or equivalently minimizing a cost function) over a compact set $X$. In the case where $f(x)$ can be easily evaluated, where derivative data is available and where the function is convex (or concave), the solution is relatively straightforward, as is for instance discussed by [1].

However, we will consider the case where convexity and derivative data are not known. In addition, every function evaluation is expensive and we can only obtain noisy measurements of the function. How can we now, through a data-driven approach, optimize the function?

A. Two different problem set-ups

The main idea is to try out certain inputs $x_1, \ldots, x_n$. After selecting a so-called try-out input $x_k$, we feed it into the function and obtain a noisy measurement $y_k = f(x_k) + \varepsilon$, with $\varepsilon = \mathcal{N}(0, \sigma^2)$. We then use all measurements obtained so far to make a Bayesian approximation of the function $f(x)$, based on which we choose the next try-out input $x_{k+1}$. As such, this problem is known as Bayesian optimization. In particular, we can approximate $f(x)$ through Gaussian process regression (see [2]). This gives us a mean $\mu(x)$ and a standard deviation $\sigma(x)$ of our estimate for $f(x)$. The resulting optimization method is also known as Gaussian process optimization. Bayesian methods like Gaussian process regression are known to efficiently deal with data, requiring only little data to make relatively accurate approximations. This makes these techniques suitable for a data-driven approach to problems in which data is expensive.

The main question is how to choose the try-out inputs $x_k$. There are actually two different problem set-ups here. In the first problem set-up, after performing all $n$ measurements, we have to give a recommendation $\hat{x}^*$ of what we believe is the true optimum $x^*$. The difference $f^* - f^*$ between the corresponding function values $f^* \equiv f(x^*)$ and $f^* \equiv f(\hat{x}^*)$ is known as the error or the instantaneous regret. As such, this problem set-up is known as the error minimization set-up or also as the probabilistic global optimization problem. It is useful in applications like sensor placement [3] and controller tuning in damage-free environments [4], [5]. These are all applications in which every try-out input (every experiment) has the same high set-up cost.

In the second problem set-up, our aim is instead to maximize the sum of all the rewards $f(x_1), \ldots, f(x_n)$, which is known as the value $V$. Equivalently, we could also minimize the (cumulative) regret

$$\sum_{k=1}^n (f^* - f(x_k)) = nf^* - V. \quad (1)$$

This set-up is known as the regret minimization set-up or also as the continuous armed bandit problem. It is useful in applications like advertisement optimization [6] or controller tuning for damage minimization (see Section III-D). These are applications where the reward or cost of an experiment actually depends on the result of the experiment. Because our applications fall in the latter category, we will focus on the regret minimization set-up in this paper. However, with the set-ups being similar, we also take error minimization strategies into account.

B. Existing error minimization methods

Several Bayesian optimization methods already exist. Good overviews are given by [7], [8], [9], though we will provide a brief summary here. The recurring theme is that, when selecting the next input $x_k$, we optimize some kind of Acquisition Function (AF). In literature, the discussion mainly concerns selecting and tuning an AF.

The first to suggest the Probability of Improvement (PI) acquisition function was [10]. This function is defined as
PI(x) = P(f(x) ≥ y_+), where P(A) denotes the probability of event A to occur and y_+ denotes the highest value of the observation obtained so far. This was expanded by [11] and [12] to the form PI(x) = P(f(x) ≥ y_+ + ξ), with ξ a tuning parameter trading off between exploration (high ξ) and exploitation (zero ξ).

Later on, [13] suggested an AF which also takes the magnitude of the potential improvement into account. It is known as the Expected Improvement (EI) acquisition function EI(x) = E[\max(0, f(x) − y_+)]. Similar methods were used by others. For instance, multi-step lookahead was added by [3], a trust region to ensure small changes to the tried inputs x by others. For instance, multi-step lookahead was added by [3], and [12] to the form PI introduced by [8]. An analysis was performed by [15].

PI was used by [14], and an additional exploration/exploitation parameter ξ similar to the one used in the PI AF was introduced by [8]. An analysis was performed by [15].

Alternatively, [16] suggested the Upper Confidence Bound (UCB) acquisition function UCB(x) = μ(x) + κσ(x). Here the parameter κ determines the amount of exploration/exploitation, with high values resulting in more exploration. Often κ = 2 is used. The extreme case of κ = 0 is also known as the Expected Value (EV) acquisition function EV(x) = μ(x). It applies only exploitation, so it is not very useful by itself. Methods to determine the value of κ optimizing regret bounds were studied by [17].

A significant shift in focus was made through the introduction of entropy search. This method was first developed by [18], although [19] independently set up a similar method and introduced the name entropy search. The method was subsequently developed further as predictive entropy search by [20]. The main idea here is to look at the so-called maximum distribution: the probability p_{max}(x) ≡ P(x = x^*) that a certain point x equals the (unknown) optimum x^*, or for continuous problems the corresponding probability density.

We then focus on the relative entropy (the Kullback-Leibler divergence) of the maximum distribution compared to the flat probability density function over X. Initially this relative entropy is zero, but the more information we gain, the higher this relative entropy becomes. As such, we want to pick the try-out point x_k which is expected to increase the relative entropy the most.

At the same time, portfolio methods were developed with the aim to optimally use a whole assortment (a portfolio) of acquisition functions. These methods were introduced by [21], using results from [22], [23] and subsequently expanded on by [24], who suggested to use the change in entropy as criterion to select recommendations.

C. Existing regret minimization methods

Nearly all the above methods have focused on the error minimization set-up, where the focus was on gaining as much information as possible. The most notable exception is the UCB AF which is mostly used for the regret minimization set-up, although the PI and EI AFs can, with the right parameters, also be applied to it.

When it comes to the regret minimization set-up, most of the research has focused on the case where the number of possible inputs x is finite. It is then known as the armed bandit problem and has been analyzed by for instance [25], [26], [27].

One of the more promising acquisition methods for the armed bandit problem is Thompson sampling. This method was first suggested by [28] and has more recently been analyzed by [29], [30]. It is fundamentally different from other methods, because it does not use an acquisition function. Instead, we select an input point x as the next try-out point x_k with probability equal to the probability that x is the optimal input x^*. This is equivalent to sampling x_k from the maximum distribution p_{max}(x). Generally this distribution is not known though. When only finitely many different input points x are possible, the solution is to consider the vector f ≡ f(X) of all possible function outputs. Using Bayesian methods, we approximate f as a random variable. As such, we can take a sample \tilde{f} from this random variable, find for which input point x this sample has its maximum, and subsequently use that input x as the next try-out point x_k.

This method has proven to work well when the number of input points is finite. When there are infinitely many possible input points, like in continuous problems, it is impossible to sample from f. This means that a new method to sample from the maximum distribution p_{max}(x) is needed. However, in literature this maximum distribution is not studied much at all. The idea of it was noted (but not evaluated) by [7]. The maximum distribution was calculated by [18] through a brute force method. An expansion to this was developed by [19], who used a method from [31] to approximate the minimum distribution. Though the approximation method used was reasonably accurate, it had a runtime of \mathcal{O}(n^4), making it infeasible to apply to most problems. An alternative method was described by [20] who approximated function samples of a Gaussian process through a finite number of basis functions and then optimized these function samples to generate samples from the maximum distribution. Though effective, this method requires solving a nonlinear optimization problem for each sample, which is computationally expensive and subject to the risk of finding only a local optimum.

D. Paper contribution and outline

The main contribution of this paper is a Sequential Monte Carlo (SMC) approach to approximate the maximum distribution. The same SMC method can then be used to sample from this maximum distribution. This enables us to set up an efficient Bayesian optimization algorithm with Thompson sampling for problems with a continuous input space, which is highly suitable for regret minimization problems. To the best of our knowledge, such an approach has not been applied before in literature and it is hence our main contribution.

In this paper we start by presenting the Monte Carlo maximum distribution algorithm for approximating the distribution of the maximum in Section II. We also analyze it and examine how we can use it to apply Thompson sampling. Experimental results are presented in Section III with conclusions and recommendations given in Section IV.

II. FINDING THE MAXIMUM DISTRIBUTION

In this section we will set up an algorithm to find/approximate the distribution of the maximum of a Gaus-
A Gaussian process and its maximum

Consider a function \( f(x) \). We assume that we have taken \( n \) measurements \( y_i = f(x_{m_i}) + \varepsilon \), where \( \varepsilon \sim N(0, \sigma_n^2) \) is Gaussian white noise. We merge all the measurement input points \( x_{m_i} \) into a set \( X_m \) and all the measured output values \( y_i \) into a vector \( y \).

Now suppose that we want to predict the (noiseless) function values \( f_* = f(X_*) \) at a given set of trial input points \( X_* \). In this case we can use the GP regression equation from \([2]\). We use a mean function \( m(x) \) and a covariance function \( k(x, x') \), and we shorten \( m(X_0) \) to \( m_a \) and \( k(X_0, X_0) \) to \( K_{ab} \) for any sensible subscripts \( a \) and \( b \). We then have

\[
\begin{align*}
  f_* &\sim N(\mu_*, \Sigma_*), \\
  \mu_* &= m_* + K_{sm} (K_{mm} + \Sigma_n)^{-1} (y - m_m), \\
  \Sigma_* &= K_{ss} - K_{sm} (K_{mm} + \Sigma_n)^{-1} K_{ms}.
\end{align*}
\]

A Gaussian process can be seen as a distribution over functions. That is, we can take samples of \( f_* \) and plot those as if they are functions, as is for instance done in Figure 1. These sample functions generally have their maximums at different locations \( x^* \). This implies that \( x^* \) is a random variable, and hence has a distribution \( p_{\text{max}}(x) \). An example of this is shown in Figure 2.

The distribution \( p_{\text{max}}(x) \) cannot be analytically calculated, but it can be approximated through various methods. The most obvious one is through brute force: for a finite number of trial input points \( X_* \), we take a large number of samples \( f_* \), for each of these samples we find the location of the maximum, and through a histogram we determine the distribution of \( x^* \). This method is far from ideal as it is computationally very intensive, even for low-dimensional functions, but it is guaranteed to converge to the true maximum distribution.

For larger problems the brute force method is too computationally intensive, motivating the need for a way of approximating the maximum distribution. Methods to do so already exist, like those used by [19] and [20]. However, these methods are all also computationally intensive for larger problems, and so a different way to approximate \( p_{\text{max}}(x) \) would be beneficial.

B. Approximating the maximum distribution

We propose a Sequential Monte Carlo (SMC) approach to finding the maximum distribution \( p_{\text{max}}(x) \). For background on SMC samplers, see [32], [33].

The main idea is that we have \( n_p \) so-called particles at positions \( x^1, \ldots, x^{n_p} \). Each of the particles has a corresponding weight \( w^1, \ldots, w^{n_p} \). Based on these particles and their weights, the maximum distribution is approximated through kernel density estimation as

\[
p_{\text{max}}(x) = \frac{\sum_{i=1}^{n_p} w^i k_x(x, x^i)}{\sum_{i=1}^{n_p} w^i}, \tag{3}
\]

with \( k_x(x, x') \) some manually chosen kernel. It is common to make use of a squared exponential kernel with a small length scale.

We initially distribute these particles \( x^i \) at random positions across the input space. That is, we sample the particles \( x^i \) from the flat distribution \( q(x) = c \). Note that, because we have assumed that the input space \( X \) is compact, the constant \( c \) is nonzero.

To learn more about the position of the maximum, we will challenge existing particles. To challenge some existing particle \( x^i \), we first sample a random challenger particle \( x^c \) from a proposal distribution \( q'(x) \). We then set up the joint distribution

\[
\begin{align*}
  f(x^i) &\sim N(\mu, \Sigma), \\
  f(x^c) &\sim N(\mu, \Sigma), \\
  \Sigma &= \Sigma(x^i, x^i) + \Sigma(x^i, x^c) + \Sigma(x^c, x^c),
\end{align*}
\]

and subsequently generate a sample \( [\hat{f}^i \hat{f}^c]^T \) from it. If \( f^c > \hat{f}^i \), then the challenger has ‘beaten’ the current particle and it takes its place. Otherwise the challenger particle is discarded.

The challenger particle also has a weight associated with it. In SMC methods this weight is usually given by

\[
w^i_c = \frac{q(x^c)}{q'(x^c)}. \tag{5}
\]

However, to speed up convergence, we use a proposal distribution \( q'(x) \) based on the ideas of mixture importance sampling and defensive importance sampling. Specifically, we use

\[
q'(x) = \alpha p_{\text{max}}(x) + (1 - \alpha)q(x). \tag{6}
\]

Here, \( \alpha \) is manually chosen (often roughly \( \frac{1}{2} \)) and \( p_{\text{max}}(x) \) is calculated according to the mixture proposal distribution \([3]\).

To generate a challenger particle \( x^c \), we hence randomly (according to the particle weights) select one of the particles \( x^i \). Then, in a part \( \alpha \) of the cases, we sample \( x^c \) from \( k_x(x, x^i) \); while in the remaining \( (1 - \alpha) \) part of the cases, we sample \( x^c \) from \( q(x) \). If we sample our challenger particles in this way, it is computationally more efficient to instead use the weight

\[
w^i_c = \frac{q(x^c)}{\alpha k_x(x^c, x^i) + (1 - \alpha)q(x^c)}. \tag{7}
\]

Based on this set-up, we will challenge every existing particle once. This is called one round of challenges. Afterwards, we apply systematic resampling to make sure all particles have the same weight again. We repeat this until the distribution of particles has mostly converged.

We call the resulting algorithm the Monte Carlo Maximum Distribution (MCMD) algorithm. Pseudo-code for it is given in Algorithm 1.

C. Analysing the limit distribution

The distribution of the particles converges to a limit distribution. But does this limit distribution equal the true maximum distribution? We can answer this question in the special case when \( a = 0 \), when we consider the discrete case, where there are finitely many possible input points \( x_1, \ldots, x_n \) and when we use the kernel \( k_x(x, x') = \delta(x - x') \), with \( \delta(\ldots) \) the delta function.
Data: A known Gaussian process, user-defined parameters $n, \alpha$ and a kernel $k(x, x')$.

Result: An approximate distribution $p_{\text{max}}(x)$ of the optimal input $x^*$, given through (3).

Initialization:

\[
\text{for } i \leftarrow 1 \text{ to } n_p \text{ do } \\
\quad \text{Sample } x^i \text{ from } q(x). \text{ Assign } w^i = 1. \\
\text{end}
\]

Iteration:

\[
\text{repeat } \\
\quad \text{Apply systematic resampling to all particles.} \\
\quad \text{for } i \leftarrow 1 \text{ to } n_p \text{ do } \\
\quad \quad \text{Select a random particle } x^j. \\
\quad \quad \text{if we select a challenger based on } x^j \text{ (probability } \alpha) \text{ then } \\
\quad \quad \quad \text{Sample a challenger particle } x^c_{ij} \text{ from the kernel } k_e(x, x^j). \\
\quad \quad \text{else } \\
\quad \quad \quad \text{Sample a challenger particle } x^c_{ij} \text{ from the flat distribution } q(x). \\
\quad \quad \text{end} \\
\quad \quad \text{Sample a vector } [\hat{f}^i, \hat{f}^c_{ij}]^T \text{ based on } [4]. \\
\quad \quad \text{if } \hat{f}^c_{ij} > \hat{f}^i \text{ then } \\
\quad \quad \quad \text{Replace particle } x^i \text{ by its challenger } x^c_{ij}. \\
\quad \quad \quad \text{Set the new weight } w^i \text{ according to } (7). \\
\quad \text{end} \\
\text{end} \\
\text{until } a \text{ sufficient number of rounds has passed; } \\
\text{end}
\]

Algorithm 1: The Monte Carlo maximum distribution algorithm. Self-normalized importance sampling, mixture importance sampling and systematic resampling are incorporated.

Consider the given Gaussian process. Let’s denote the probability $P(f(x_i) > f(x_j))$, based on the data in this Gaussian process, as $P_{ij}$. Here we have

\[
P_{ij} = \Phi\left(\frac{\mu(x_i) - \mu(x_j)}{\sqrt{\Sigma(x_i, x_i) + \Sigma(x_j, x_j) - 2\Sigma(x_i, x_j)}}\right),
\]

where $\Phi(\ldots)$ is the standard Gaussian cumulative density function. Through this expression we find the matrix $P$ element-wise. Additionally, we write the part of the particles which will eventually be connected to the input $x_i$ as $p_i$. We hence have, per definition, $p_i = p_{\text{max}}(x_i)$. In this case, the resulting vector $p$ can be shown to satisfy

\[
(P - \text{diag}(1_nP))p = 0,
\]

where $1_n$ is an $n \times n$ matrix filled with ones, and $\text{diag}(\ldots)$ is the function which sets all non-diagonal elements to zero. If we also use the fact that the sum of all probabilities $1^T p$ equals 1, we can find the maximum distribution $p$ for this discrete problem.

Since we can analytically calculate this limit distribution of the particles, we already know that this distribution is not equal to the true maximum distribution, since that cannot be calculated analytically. The example from Figure 2 does show that they are reasonably close. The limit distribution is generally less peaked than the true maximum distribution, which means it contains less information about where the maximum is (lower relative entropy) but overall its predictions are accurate. Furthermore, the difference will decrease as the variance present within the Gaussian process decreases.

D. Applying the MCMD algorithm for Thompson sampling

We can now use the MCMD algorithm to apply Thompson sampling in a Gaussian process optimization setting. To do so, we sample an input point $x$ from the approximated maximum distribution $p_{\text{max}}(x)$ whenever we need to perform a new measurement.

The downside of this method is that samples are not drawn from the true maximum distribution, but only from an approximation of it. However, the upside is that this approximation can be obtained by making simple comparisons between function values. No large matrix equations need to be solved or nonlinear function optimizations need to be performed, providing a significant advantage over other methods that approximate the maximum distribution.

Whether the resulting algorithm is effective at optimizing a function with as little regret as possible is discussed next.

III. RESULTS

Here we show the results of the presented algorithms. First we study how the MCMD algorithm works for a fixed one-dimensional Gaussian process. Then we apply it through Thompson sampling to the same problem, expand to a two-dimensional problem and finally apply it to a real-life application. Code related to the experiments can (after August 2016) be found through [34].

A. Execution of the MCMD algorithm

Consider the function

\[
f(x) = \cos(3x) - \frac{1}{9}x^2 + \frac{1}{6}x.
\]

From this function, we take 20 noisy measurements, at random locations in the interval $[-3, 3]$, with $\sigma_n = 0.3$ as standard deviation of the white noise. We then apply GP regression with a squared exponential covariance function with predetermined hyperparameters. The subsequent GP approximating these measurements is shown in Figure 1.

We can apply the MCMD algorithm to find the maximum distribution of this Gaussian process. This distribution $p_{\text{max}}(x)$, during successive challenge rounds of the algorithm with $\alpha = \frac{1}{2}$, is shown in Figure 2. Here we see that the algorithm has mostly converged to the limit distribution after $n_r = 10$ rounds of challenges.

B. Application to an optimization problem

We will now apply the newly developed method for Thompson sampling to Bayesian optimization. We will compare it with the UCB, the PI and the EI acquisition functions. After some manual tuning, their parameters were set to $\kappa = 2$...
and $\xi = 0.1$. To optimize these acquisition functions, we use a multi-start optimization method, because otherwise we occasionally wind up with a local optimum of the acquisition function, resulting in a detrimental performance. We do not compare our results with entropy search or portfolio methods, because they are designed for the error minimization set-up.

The first problem we apply these methods to is the optimization of the function $f(x)$ of (10). We use $n = 50$ input points $x_1, \ldots, x_n$ and look at the obtained regret. To keep the memory and runtime requirements of the GP regression algorithm somewhat limited, given the large number of experiments that will be run, we will apply the FITC approximation described in [35], implemented in an online fashion according to [36]. As inducing input points, we use the chosen input points, but only when they are not within a distance $d_{iu}$ (decreasing from 0.3 to 0.02 during the execution of the algorithm) of any already existing inducing input point. For simplicity the hyperparameters are assumed known and are hence fixed to reasonable values. Naturally, it is also possible to learn hyperparameters on-the-go as well, using the techniques described by [2].

The result is shown in Figure 3. In this particular case, it seems that Thompson sampling and the PI AF applied mostly exploitation: they have a better performance on the short term. On the other hand, the UCB and EI acquisition functions apply more exploration: the cost of quickly exploring is higher, but because the optimum is found sooner, it can also be exploited sooner.

It should also be noted that all algorithms occasionally wind up in the wrong optimum (near $x = 2$) every now and then. This can be seen from the fact that the regret graph does not level out. For this particular problem, the UCB AF seems to be the best at avoiding the local optima, but it still falls for them every now and then. As noted earlier, only Thompson sampling has the guarantee to escape local optimums given infinitely many measurements.

C. Extension to a two-dimensional problem

Next, we apply the optimization methods to a two-dimensional problem. We will minimize the well-known Branin function from (among others) [37]. Or equivalently, we maximize the negative Branin function,

$$f(x_1, x_2) = - \left( x_2 - \frac{5x_1^2}{4\pi^2} + \frac{5x_1}{\pi} - 6 \right)^2 - 10 \left( 1 - \frac{1}{8\pi} \right) \cos(x_1) - 10, \quad (11)$$

where $x_1 \in [-5, 10]$ and $x_2 \in [0, 15]$. This function is shown in Figure 4. We can find analytically that the optimums occur at $(-\pi, 10\frac{1}{8\pi})$, $(\pi, 10\frac{9}{8\pi})$, and $(3\pi, 10\frac{9}{8\pi})$, all with value $-\frac{5}{4\pi}$.
The performance of the various optimization methods, averaged out over fifty full runs, is shown in Figure 5. Here we see that Thompson sampling now performs significantly better at keeping the regret small compared to the UCB ($\kappa = 2$), the PI and the EI ($\xi = 2$) acquisition functions. We can find the reason behind this, if we look at which try-out points the various algorithms select. When we do (not shown here), we see that all acquisition functions often try out points at the border of the input space, while Thompson sampling does not. In particular, the acquisition functions (nearly) always try out all four corners of the input space, including the very detrimental point $(-5, 0)$. It is this habit which makes these acquisition functions perform worse on this specific problem.

Other than this, it is also interesting to note that in all examined runs, all optimization methods find either two or three of the optimums. So while multiple optimums are always found, it does regularly happen that one of the three optimums is not found. All of the methods have shown to be susceptible to this. In addition, the three acquisition functions have a slightly lower average recommendation error than Thompson sampling, but since all optimization methods find various optimums, the difference is negligible. On the flip side, an advantage of using the MCMD algorithm is that it can provide us with the posterior distribution of the maximum, given all the measurement data. An example of this is shown in Figure 6.

Fig. 4. The Branin function, defined by (11).

D. Optimizing a wind turbine controller

Finally we test our algorithm on an application: using data-based controller tuning for load mitigation within a wind turbine. More specifically, we use a linearized version of the so-called TURBU model, described by [38]. TURBU is a fully integrated wind turbine design and analysis tool. It deals with aerodynamics, hydrodynamics, structural dynamics and control of modern three bladed wind turbines, and as such gives very similar results as an actual real-life wind turbine.

We will consider the case where trailing edge flaps have been added to the turbine blades. These flaps should then be used to reduce the vibration loads within the blades. To do so, the Root Bending Moment (RBM) of the blades is used as input to the control system.

To determine the effectiveness of the controller, we look at two quantities. The first is the Damage Equivalent Load (DEL – see [39]). The idea here is that the blades are subject to lots of vibrations, some with large magnitudes and some with small magnitudes. For fatigue damage, large oscillations are much more significant. To take this into account, we look at which $1$ Hz sinusoidal load would result in the same fatigue damage as all measured oscillations put together. To accomplish this, the RBM signal is separated into individual oscillations using the rainflow algorithm [40]. We then use Miner’s rule [41], applying a Wöhler exponent of $m = 11$ for the glass fiber composite blades [42], to come up with an equivalent $1$ Hz load.

The second quantity to optimize is the mean rate of change of the input signal. The reason here is that the lifetime of bearings is often expressed in the number of revolutions, or equivalently in the angular distance traveled, and dividing this
distance traveled by the time passed will result in the mean rate of change of the flap angle. The eventual performance score for a controller will now be a linearly weighted sum of these two parameters, where a lower score is evidently better.

As controller, we apply a proportional controller. That is, we take the RBM in the fixed reference frame (so after applying a Coleman transformation – see [43]) and feed the resulting signal, multiplied by a constant gain, to the blade flaps. Since the wind turbine has three blades, there are three gains which we can apply. The first of these, the collective flap mode, interferes with the power control of the turbine, and hence we will ignore this one and only tune the gains of the tilt and yaw modes. Very low gains (in the order of $10^{-8}$) will result in an inactive controller which does not reduce the RBM, while very high gains (in the order of $10^{-5}$) will react to every small bit of turbulence, resulting in an overly aggressive controller with a highly varying input signal. Both are suboptimal, and the optimal controller will have gains somewhere between these two extreme values.

To learn more about the actual score function, we can apply a brute force method – just applying 500 random controller settings – and apply GP regression. This gives us Figure 7. Naturally, this is not possible in real life as it would cause unnecessary damage to the wind turbine. It does tell us, however, that the score function is mostly convex and that there does not seem to exist any local optima. One result of this is that our manual tuning of the acquisition function parameters resulted in the low values $\kappa = 1$ and $\xi = 0.005$, which shows that this problem is mainly one of exploitation.

The results from the experiments, which are remarkably similar to earlier experiments, are shown in Figure 8. They once more show that Thompson sampling has a competitive performance at keeping the regret limited. A similar experiment, though with far fewer measurements, has been performed on a scaled wind turbine placed in a wind tunnel, and the results there were similar as well. See [44] (after August 2016) for further details on this wind tunnel test.

IV. CONCLUSIONS AND RECOMMENDATIONS

We have introduced the MCMD algorithm which uses particles to approximate the distribution of the maximum of a Gaussian process. This particle approximation can then be used to set up a Bayesian optimization method using Thompson sampling. Such optimization methods are suitable for tuning the parameters of systems with large amounts of uncertainty in an online data-based way. As an example, we have tuned the controller gains of a wind turbine simulation to reduce the fatigue load, using performance data that was obtained during the operation of the wind turbine.

The main advantage of Thompson sampling with the MCMD algorithm is that it does not require the optimization of a nonlinear function to select the next try-out point. In addition, it has shown to have a competitive performance at keeping the cumulative regret limited. However, we cannot conclude that Thompson sampling, or any other optimization method, works better than its competitors. Which method works well depends on a variety of factors, like how much the method has been fine-tuned to the specific function that is being optimized, as well as which function is being optimized in the first place. It is for this very reason that any claim that a Bayesian optimization works better than its competitors may be accepted only after very careful scrutiny.

There are still many ways in which the MCMD algorithm can be improved. One particular suggestion would be to use multiple challenger points in every single challenge. If we are dealing with a discrete function $f(x)$ with $n$ possible different input points, and if we use $n$ different challenger points every time, then we are guaranteed to wind up with the true maximum distribution. So this improvement is likely to bring the limit distribution discussed in Section II-C closer to the true maximum distribution. Whether it does is left as a subject for future research.
ACKNOWLEDGMENT

This research is supported by the Dutch Technology Foundation STW, which is part of the Netherlands Organisation for Scientific Research (NWO), and which is partly funded by the Ministry of Economic Affairs (Project number: 12173, SMART-WIND). The work was also supported by the Swedish research Council (VR) via the project Probabilistic modeling of dynamical systems (Contract number: 621-2013-5524).

REFERENCES

[1] S. Boyd and L. Vandenberghe, Convex Optimization. Cambridge University Press, 2004.
[2] C. E. Rasmussen and C. K. Williams, Gaussian Processes for Machine Learning. MIT Press, 2006.
[3] M. Osborne, “Bayesian Gaussian processes for sequential prediction, optimisation and quadrature,” Ph.D. dissertation, University of Oxford, 2010.
[4] D. Lizotte, T. Wang, M. Bowling, and D. Schuurmans, “Automatic gait optimisation with Gaussian process regression,” in Proceedings of the 20th International Joint Conference on Artificial Intelligence, 2007, pp. 944–949.
[5] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe, “Automatic LQR tuning based on Gaussian process global optimization,” in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) 2016, IEEE, May 2016.
[6] S. Pandey and C. Olston, “Handling advertisements of unknown quality in search advertising,” in Advances in Neural Information Processing Systems. MIT Press, 2007, vol. 19, pp. 1065–1072.
[7] D. J. Lizotte, “Practical Bayesian optimization,” Ph.D. dissertation, University of Alberta, 2008.
[8] E. Brochu, V. M. Cora, and N. de Freitas, “A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning,” University of British Columbia, Tech. Rep., 2010.
[9] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, “Taking the human out of the loop: A review of Bayesian optimization,” Proceedings of the IEEE, vol. 104, no. 1, pp. 148–175, January 2016.
[10] H. Kushner, “A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise,” Journal of Basic Engineering, vol. 86, no. 1, pp. 97–106, 1964.
[11] A. Torn and A. Zilinskas, Global Optimization. Springer-Verlag New York, Inc., 1989.
[12] D. R. Jones, “A taxonomy of global optimization methods based on response surfaces,” Journal of Global Optimization, vol. 21, no. 4, pp. 345–383, 2001.
[13] J. Mockus, V. Tiesis, and A. Zilinskas, The application of Bayesian methods for seeking the extremum. Elsevier, Amsterdam, 1978.
[14] J. Park and K. H. Law, “Bayesian Ascent (BA): A data-driven optimization scheme for real-time control with application to wind farm power maximization,” IEEE Transactions on Control Systems Technology, November 2015.
[15] E. Vazquez and J. Bect, “Convergence properties of the expected improvement algorithm with fixed mean and covariance functions,” Journal of Statistical Planning and Inference, vol. 140, no. 11, pp. 3088–3095, 2010.
[16] D. D. Cox and S. John, “SDO: A statistical method for global optimization,” in Multidisciplinary Design Optimization: State-of-the-Art, 1997, pp. 315–329.
[17] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, “Information-theoretic regret bounds for Gaussian process optimization in the bandit setting,” IEEE Transactions on Information Theory, vol. 58, no. 5, pp. 3250 – 3265, May 2012.
[18] J. Villemonteix, E. Vazquez, and E. Walter, “An informational approach to the global optimization of expensive-to-evaluate functions,” Journal of Global Optimization, vol. 43, no. 2, pp. 373–389, March 2009.
[19] P. Hennig and C. J. Schuler, “Entropy search for information-efficient global optimization,” Journal of Machine Learning Research, vol. 13, pp. 1809–1837, 2012.
[20] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani, “Predictive entropy search for efficient global optimization of black-box functions,” in Advances in Neural Information Processing Systems 27. Curran Associates, Inc., 2014.
[21] M. Hoffman, E. Brochu, and N. de Freitas, “Portfolio allocation for Bayesian optimization,” in Uncertainty in Artificial Intelligence (UAI), 2011, pp. 327–336.
[22] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, “Gambling in a rigged casino: The adversarial multi-armed bandit problem,” in Proceedings of the 36th Annual Symposium on Foundations of Computer Science, 1995, pp. 322–331.
[23] K. Chaudhuri, Y. Freund, and D. J. Hsu, “A parameter-free hedging algorithm,” in Advances in Neural Information Processing Systems, 2009, vol. 22, pp. 297–305.
[24] B. Shahriari, Z. Wang, M. W. Hoffman, A. Bouchard-Côté, and N. de Freitas, “An entropy search portfolio for Bayesian optimization,” University of Oxford, Tech. Rep., 2014.
[25] R. D. Kleinberg, “Nearly tight bounds for the continuum-armed bandit problem,” in Advances in Neural Information Processing Systems 17. MIT Press, 2004, pp. 697–704.
[26] S. Grünewälder, J.-Y. Audibert, M. Opper, and J. Shawe-Taylor, “Regret bounds for Gaussian process bandit problems,” in Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS), 2010.
[27] N. de Freitas, A. Smola, and M. Zoghi, “Regret bounds for deterministic gaussian process bandits,” arXiv.org, Tech. Rep., 2012.
[28] W. R. Thompson, “On the likelihood that one unknown probability exceeds another in view of the evidence of two samples,” Biometrika, vol. 25, no. 3/4, pp. 285–294, 1933.
[29] O. Chapelle and L. Li, “An empirical evaluation of Thompson sampling,” in Advances in Neural Information Processing Systems. Curran Associates, Inc., 2011, vol. 24, pp. 2249–2257.
[30] S. Agrawal and N. Goyal, “Analysis of Thompson sampling for the multi-armed bandit problem,” in JMLR Workshop and Conference Proceedings, vol. 23, 2012, pp. 39.1–39.26.
[31] T. P. Minka, “Expectation propagation for approximate bayesian inference,” in Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence, 2001.
[32] P. Del Moral, A. Doucet, and A. Jasra, “Sequential Monte Carlo samplers,” Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 68, no. 3, pp. 411–436, 2006.
[33] A. B. Owen, Monte Carlo theory, methods and examples, 2013.
[34] H. Bijl, “Gaussian progression regression techniques source code,” 2016. [Online]. Available: https://github.com/HildoBijl/GPR
[35] J. Q. Candela and C. E. Rasmussen, “A unifying view of sparse approximate Gaussian process regression,” Journal of Machine Learning Research, vol. 6, pp. 1939–1959, 2005.
[36] H. Bijl, J.-W. van Wingerden, T. B. Schön, and M. Verhaegen, “Online sparse Gaussian process regression using FITC and PTC approximations,” in Proceedings of the IFAC symposium on System Identification, SYSID. Beijing, China, October 2015.
[37] L. Dixon and G. Szęgo, “The global optimisation problem: an introduction,” in Towards global optimization, L. Dixon and G. Szęgo, Eds. North-Holland Publishing, 1978, vol. 2, pp. 1–15.
[38] T. van Engelen and H. Braam, “TURBU Offshore: Computer program for frequency domain analysis of horizontal axis offshore wind turbines – Implementation,” ECN, Tech. Rep. Report ECN-C-04-079, 2004.
[39] G. Freebury and W. Musial, “Determining equivalent damage loading for full-scale wind turbine blade fatigue tests,” in Proceedings of the 19th American Society of Mechanical Engineers (ASME) Wind Energy Symposium, Reno, Nevada, 2000.
[40] A. Nieslony, “Determination of fragments of multiaxial service loading strongly influencing the fatigue of machine components,” Mechanical Systems and Signal Processing, vol. 23, no. 8, pp. 2712–2721, November 2009.
[41] R. H. Wirsching, T. L. Paez, and K. Ortiz, Random Vibrations, Theory and Practice, John Wiley & Sons, Inc., 1995.
[42] F. Savenije and J. Peering, “Aero-acoustic simulation of offshore wind turbines in the frequency domain.” Energy research centre ECN, The Netherlands, Tech. Rep. Report ECN-E-09-060, 2009.
[43] E. van Solingen and J. W. van Wingerden, “Linear individual pitch control design for two-bladed wind turbines,” Wind Energy, vol. 18, pp. 677–697, 2015.
[44] H. Bijl, “Gaussian process regression techniques,” Ph.D. dissertation, Delft University of Technology, 2016.
Hildo Bijl has received the bachelor's degree in Aerospace Engineering Cum Laude in 2009 at the Delft University of Technology, The Netherlands. In 2012 he completed the Dynamics and Control of Aerospace Vehicles MSc track with the highest grade possible. As part of his MSc, he did an internship at the NASA Ames research center in California in 2010-2011. For his thesis work he was awarded the PEGASUS prize for best Aerospace Engineering graduate of Europe. Since 2012 he has been a Ph.D. candidate at the Delft Center for Systems and Control. In 2015 he has been a visiting researcher at the Information Technology department at Uppsala University, Sweden. His main research interests include machine learning and Gaussian process regression. Next to this, he has an interest in improving education using novel online technologies.

Thomas B. Schön is Professor of the Chair of Automatic Control in the Department of Information Technology at Uppsala University. He received the Ph.D. degree in Automatic Control in Feb. 2006, the MSc degree in Applied Physics and Electrical Engineering in Sep. 2001, the BSc degree in Business Administration and Economics in Jan. 2001, all from Linköping University. He has held visiting positions with the University of Cambridge (UK), the University of Newcastle (Australia) and Universidad Técnica Federico Santa María (Valparaíso, Chile). He received the Arnberg prize by the Royal Swedish Academy of Sciences in 2016. He is a senior member of the IEEE and an Associate Editor of Automatica. He was awarded the Automatica Best Paper Prize in 2014, and in 2013 he received the best Ph.D. thesis award by The European Association for Signal Processing. He received the best teacher award at the Institute of Technology, Linköping University in 2009. His main research interest is nonlinear inference problems, especially within the context of dynamical systems, solved using probabilistic methods.

Jan-Willem van Wingerden was born on December 9, 1980 in Ridderkerk, The Netherlands. In 1999 he started his study in Mechanical Engineering at the Delft University of Technology, where he graduated Cum Laude in December 2004 at the Control Engineering group, nowadays part of the Delft Center for Systems and Control (DCSC). His graduation project was carried out at Philips Applied Technologies in Eindhoven. In January 2005, he started his Ph.D. project entitled: ‘Smart’ dynamic rotor control for large offshore wind turbines within the Delft Center for Systems and Control and the Delft University Wind Energy Research Institute. In 2008, he received his Ph.D. degree Cum Laude from the Delft Center of Systems and Control, Delft University of Technology. In 2010 he was a visiting scientist at the National Renewable Energy Lab. (USA). Currently he is an associate professor at the Delft University of Technology. His main research interests include system identification, robust control, and control and identification of wind turbines/farms.

Michel Verhaegen received the Engineering degree in Aeronautics from Delft University of Technology, Delft, The Netherlands, in 1982, and the Ph.D. degree in Applied Sciences from the Catholic University Leuven, Leuven, Belgium, in 1985. From 1985 to 1994, he was a Research Fellow at the U.S. National Research Council (NRC) and the Dutch Academy of Arts and Sciences. During 1994-1999, he was an Associate Professor with the Control Laboratory, Delft University of Technology, and became a Full Professor at the Faculty of Applied Physics, University of Twente, The Netherlands, in 1999. Since 2001, he has been with the Delft Center for Systems and Control, Delft University of Technology. His main research directions include system identification, distributed and fault-tolerant control, and data-driven controller design methodologies. Application areas include control for high resolution imaging and wind energy.