Gradient Observations in Global Structure Search with Bayesian Optimization

Master's Thesis in Physics
Faculty of Natural Sciences and Technology
Åbo Akademi University
October 2022

Mikael Granit

Supervisors

Joakim Löfgren
Department of Applied Physics
Aalto University

Torbjörn Björkman
Faculty of Natural Sciences and Technology
Åbo Akademi University

Milica Todorović
Department of Mechanical and Materials Engineering
University of Turku

Abstract

Knowledge of the microscopic atomic structure of a material gives insight into its macroscopic properties, which facilitates tailoring of the material for specific functionalities. Potential obstacles encountered during direct experimentation can be avoided using computational simulations to model atomic structures. The space of possible atomic configurations can be mapped onto a potential energy surface, with lower energies indicating more stable configurations. This surface can be explored to determine stable states of the material. However, the exploration typically demands a high number of configuration samples. The Bayesian optimization structure search (BOSS) method alleviates the exploration by keeping the number of samples required to a minimum.

Atomic simulations often calculate gradients in addition to energies of atomic systems, which means data points that are richer in information content. Therefore, including gradient observations should theoretically further reduce the number of configuration samples required by the BOSS method to find stable states. The goal of my research was to implement the functionality of including gradient observations into the BOSS method, and to investigate the effect the functionality has on the exploration. The implementation involved the writing of relevant mathematical concepts into software, specifically two Python packages. The effect the functionality has on the exploration was examined by applying the augmented BOSS method on two computational experiments that both modeled different atomic systems.

The results from the computational experiments suggest that including gradient observations reduces the amount of data required by the BOSS method, by as much as approximately 66 % in some cases. However, the inclusion of gradient observations make certain matrix operations scale as $\mathcal{O}(N^3D^3)$, for N data points and D dimensions. This resulted in a slower process in some cases, even though the amount of data required was reduced. Therefore, there is a compromise between amount of data required and time taken by the BOSS method. The inclusion of gradient observations will likely be most beneficial for slower simulations—such as those of higher fidelity.

Table of Contents

1 Introduction

1.1 Materials Science Simulations

Many fields of research, such as modern medicine or renewable energy, depend on increasingly complex devices. The functionalities of a device stem from the macroscopic properties—e.g., elasticity, electrical conductivity or heat capacity—of the material that composes the device. The macroscopic properties of a material are conditioned by its microscopic atomic configurations. Knowledge of the atomic structure of a material is therefore required when tailoring it for a specific functionality in a complex device.

How the material is expected to behave in its natural (relaxed) state is key in understanding how to engineer the material in a rational manner. The natural state of a material is represented by a particular atomic configuration. Finding this atomic configuration is one of the main purposes of **structure search**. Structure search can be performed with experimentation, but it does pose challenges in certain scenarios. While examining material surfaces experimentally is relatively straightforward, probing buried interfaces in bulk material can be problematic. The experimental resolution can also be insufficient for an accurate description of the atomic structure.

An alternative approach to structure search is provided by computational materials simulations. In a simulation, models of the atomic structure can be adjusted and then investigated without the concern for experimental cost. In many cases, a simulation will save time compared to an experiment. Different types of simulations will, however, incur varying amounts of computational costs, depending on the accuracy of the model. The quantum mechanical modeling method **density functional theory** (DFT) accurately describes the microscopic interactions that determine atomic structures for many systems [1]. DFT derives its accuracy from its quantum mechanical basis, yet carries moderate computational cost compared to other higher-level wave function-based methods [2].

A DFT simulation takes a model of the atomic structure as input and returns a variety of outputs describing the atomic system. One of these outputs is the energy of the system, with lower energies indicating more stability. The phase space of possible atomic configurations can be mapped onto a **potential energy surface** (PES) (an example surface is illustrated in fig. 1). Structure search, in the context of materials simulations, focuses on exploring this surface in order to find the atomic configuration with the lowest energy, i.e., the **global minimum**. The global minimum indicates the natural state of the material, and is found through the process of **global optimization**.

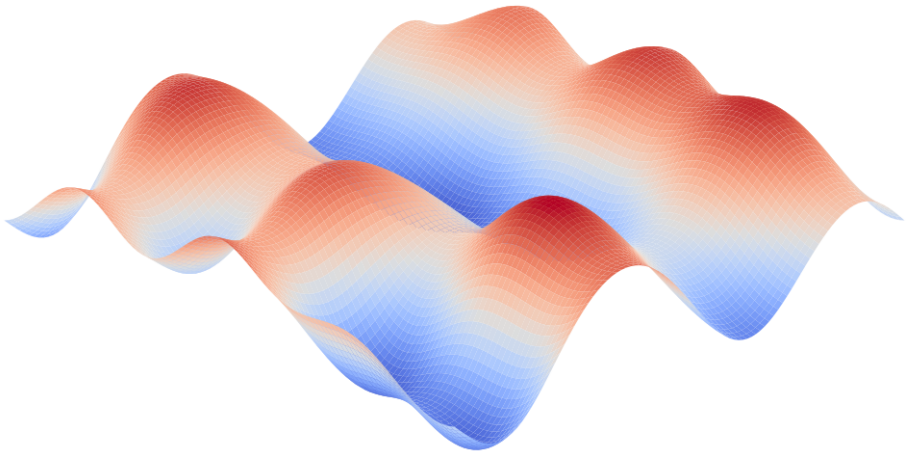


Fig. 1: A potential energy surface over two-dimensional phase space. The energy of the system is represented as the height of the surface. The surface contains several minima and maxima, with the global minimum indicating the most stable state of the system. The surface depicted is the PES of a conformer search of an alanine molecule, described in section 2.3.

1.2 Global Optimization for Structure Search

Finding the extrema of a function can be formulated as an optimization problem. Both maxima and minima can be found by minimizing the **objective function** (the function that is being optimized), as maximizing the function $f(x)$ is equivalent to minimizing the function $g(x) = -f(x)$. Because of this equivalency, I will from this point forward refer to optimization as a process solely concerned with finding minima.

Local optimization is the process of finding the local minima that is closest to some starting point. The location of the minimum is approached by descending the objective function in the direction of the negative gradient (the gradient points in the direction of steepest ascent). The magnitude of the gradient will decrease during the descent, and will tend to zero at the minimum.

Global optimization finds the local minimum with the lowest function value. With structure search, the optimization takes place in the phase space of possible configurations. Global optimization for structure search finds the lowest function value within this phase space, indicating the most stable structure configuration. There exists no general solution for finding the global minimum directly [3], so all local minima within the phase space need to be determined and compared. A key criterion for global optimization is then that the phase space needs to be explored thoroughly, so that no local minimum is left out of the comparison. This entails many function evaluations required for both exploration and gradient descent.

The high number of function evaluations required presents a problem when structure search is done with DFT simulations. With DFT, a function evaluation entails a simulation and the resulting data point is the energy of the given atomic configuration. While the computational cost of an individual DFT simulation is modest, the global optimization process becomes computationally prohibitive when many simulations are run in aggregate. Conventional phase space exploration methods, such as minima hopping [4], Monte Carlo methods [5], or metadynamics [6], typically require thousands of data points [7].

An approach to global optimization for structure search that minimizes the number of data points required is therefore of interest. One such approach is the **Bayesian optimization structure search** (BOSS) method [1]. BOSS employs **Bayesian optimization** (BO), which is an active learning machine learning method that facilitates global optimization by building a surrogate model of the objective function that is iteratively updated with data through smart sampling (see fig. 2 for an example process). The surrogate model is fit to (potentially noisy) data with **Gaussian process regression** (GPR), and allows for inference of the objective function across the entire phase space. The model will return an estimate of the function value $f(x)$ at point x, along with an uncertainty in its prediction. The uncertainty will vanish at the data points, and rise in unexplored areas of phase space. The data points with which the model is updated are chosen at locations in phase space that are determined by an **acquisition function**. The acquisition function specifies the location that would give the surrogate model the most additional information about the objective function, therefore leading to a better fit. For example, the acquisition function can prioritize areas of high uncertainty, as areas close to already known function values yield less new information.

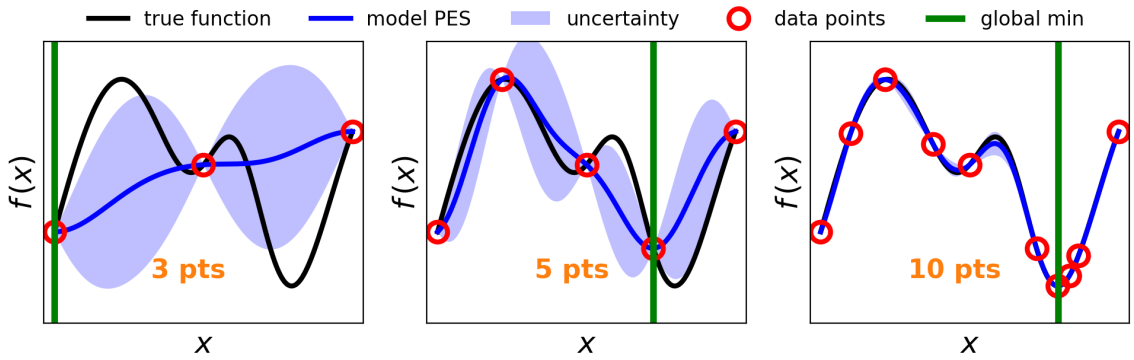


Fig. 2: A BO example process of a simple 1D function. The surrogate model is updated as new evaluations are added to the data set. The global minimum can be estimated by minimizing the model, and the estimation will approach the true global minimum as the model becomes refined with more data.

When structure search using DFT simulations is combined with BO, the PES is emulated by the surrogate model. Inferring energy values with the model is considerably less expensive than DFT simulations. Therefore, the model can be optimized for an estimate of the global minimum using conventional phase space exploration methods, without the concern for computational cost. The more data that the model is fit to, the better an emulator of the PES it will be, which means a better estimate of the minimum. The BO process can be initialized with a small data set, which is then iteratively expanded with data points chosen by the acquisition function. As the data set expands, the estimate will approach the true value. When the estimate is deemed an acceptable approximation of the true value, the process is said to have **converged** and can be discontinued. Thus, the BOSS method finds the global minimum of a PES with a minimal number of data points.

The number of data points required for convergence typically increases with dimensionality. As GPR is reliant on matrix calculations, BOSS does not necessarily scale well for complex systems with many degrees of freedom. This can be alleviated with a "building-block" approach, by treating particular groupings of atoms as rigid objects (such as aromatic rings or functional groups). The degrees of freedom can then be constrained by searching the translational or rotational phase space of the single object, or "block". When an approximate location of the global minimum has been established, the constraint can be relaxed and the minimum can be determined through simple gradient descent.

In this work, I employed the BOSS method with the Python package `aalto-boss` [1], available for download on the PyPI repository and authored by Milica Todorović (University of Turku) and Patrick Rinke (Aalto University) in collaboration with Jukka Corander (University of Helsinki/University of Oslo) and Michael Gutmann (University of Edinburgh). The BOSS package is under continuous development at Aalto University by the Computational Electronic Structure Theory (CEST) group and at University of Turku.

1.3 Research Objectives

In addition to the total energy of an atomic configuration, the forces acting on each atom can be calculated using DFT simulations, without any meaningful increase in computational cost. The combination of forces describes the gradient of the energy landscape, i.e., how the landscape changes. Incorporating this information into the surrogate model of BO should theoretically make the model a better emulator of the objective function, which consequently should reduce the number of data points required for convergence by BOSS, ultimately resulting in a more efficient process.

My research for this thesis focused on investigating the extent to which including gradient observations in BOSS improves efficiency. For this purpose, the gradient information was incorporated into the BOSS method through computational implementation of relevant mathematical concepts into Python code packages. The majority of the implementation took place in a dependency that aalto-boss utilizes for GPR: `GPy` [8], developed by the Sheffield machine learning group. With the implementation finalized, I tested the augmented code on structure search problems in order to measure the effect gradient observations has on efficiency. These aspects of the research were defined as three main objectives:

1. **Implementation of gradient observations in GPR**
   This implementation involved expanding the GPy code by writing functions that allows for the incorporation of gradient observations into the surrogate model. This was built on the code framework for gradient observations already established for GPy by Eero Siivola [9]. The main part of this objective involved generalizing the dimensionality of the framework.

2. **Implementation of augmented surrogate model in BOSS**
   With the GPR framework of GPy augmented, a new surrogate model class that could include gradient observations was implemented for aalto-boss.

3. **Study of structure search experiments**
   In order to investigate the effect of including gradient observations in BOSS, two computational structure search experiments were conducted. BOSS was performed both with and without gradient observations, so that the results could be compared.

In this thesis, I will first present the theoretical framework behind the BOSS method and its constituent parts in chapter 2. This is followed by a description of the computational implementation of gradient information into BOSS, along with more detailed descriptions of relevant code packages, in chapter 3. The results from the structure search experiments will be presented and discussed in chapter 4. Finally, I will present my conclusion for this research in chapter 5.

2 Bayesian Optimization Structure Search

In this chapter, I will present the theoretical framework behind the BOSS method. BO is the algorithmic foundation on which the BOSS method is based and involves two main components: a surrogate model of the objective function that is fit with GPR, and an acquisition function that determines the sampling locations in phase space. GPR will be described in detail in section 2.1, followed by the acquisition function in section 2.2. The combination of BO with structure search is described in section 2.3, along with introductions to the structure search problems that I have studied for the sake of this research.

2.1 Gaussian Process Regression

The surrogate model of the objective function is fit with GPR, which is a Bayesian statistical modeling method. Bayesian statistical methods combine prior information or beliefs about the distribution of a random variable with observed instances of that variable. BO receives its name from this Bayesian statistical approach.

We can encode our presuppositions about the problem—such as smoothness or symmetry—onto a **prior probability distribution** (from now on, I will simply refer to this distribution as the **prior**). The prior is a distribution over functions, which means a sample drawn from the distribution (see fig. 3) is a random function represented as a vector of function values. The random functions are our candidates for the objective function f, with some random functions being more likely than others depending on our prior beliefs. The prior is considered to be a multivariate normal distribution, defined by mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$:

$$f \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{1}$$

Given some observed data at training locations $\mathbf{X}$ with function values $\mathbf{f} = f(\mathbf{X})$, the prior can be conditioned on the data set $(\mathbf{X}, \mathbf{f})$ (described by the **likelihood**) in order to produce the **posterior probability distribution** (henceforth referred to as the **posterior**). Specifically, the posterior is proportional to the product of the prior and the likelihood, and is described by Bayes' theorem:

$$\textbf{posterior} \propto \textbf{likelihood} \times \textbf{prior}, \quad P(M|D) \propto P(D|M)P(M), \tag{2}$$

where $P(M|D)$ is the posterior probability of a model M given data D. An example of a prior being conditioned on data in order to produce a posterior is illustrated in fig. 3.

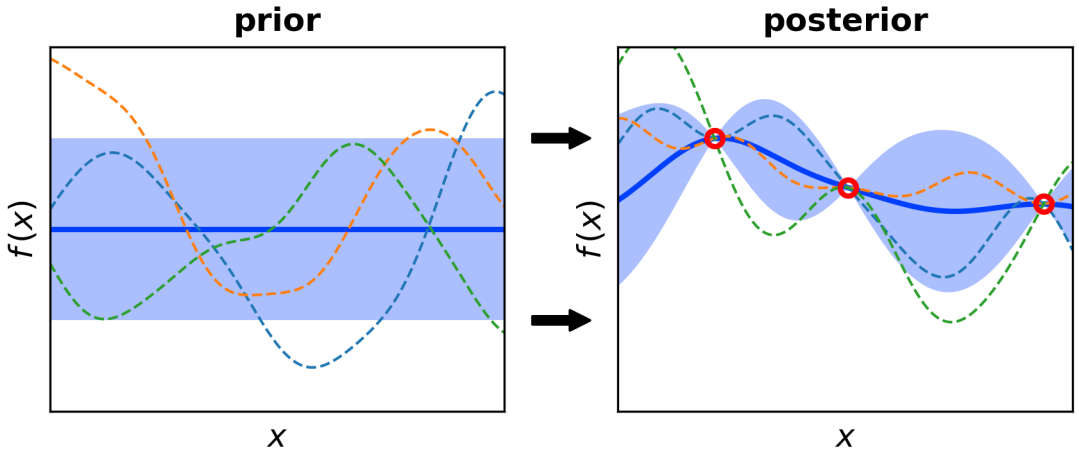


Fig. 3: An example of a prior being conditioned on data in order to produce a posterior. The prior is uninformative, which means the mean vector is flat with values of zero everywhere. Three random function samples are drawn from the prior distribution and are shown as dashed lines. The prior is then conditioned on three data points (shown in red) in order to produce the posterior on the right. Three samples are also drawn from the posterior distribution, and they all pass through the data points. In both plots, the mean vector of each distribution is drawn as the solid blue line with one standard deviation above and below the mean as the blue area.

If we have no prior knowledge to encode, and we do not wish to bias the posterior result, then we assume a flat mean function of zeroes everywhere: $\boldsymbol{\mu} = \mathbf{0}$. The prior is then defined simply by the covariance matrix $\boldsymbol{\Sigma}$:

$$f \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}). \tag{3}$$

The covariance matrix is constructed by a covariance function K, or **kernel**. The kernel can encode the assumption that two points in the input space should have a larger positive correlation the closer they are to each other. This assumption elevates the probability of random function samples drawn from both the prior and the posterior being smooth, as can be seen in fig. 3 (the "smoothness" in this case is a result of the kernel being infinitely differentiable). Just how smooth the samples will be will depend on a hyperparameter of a kernel. Kernels and hyperparameters will be described in more detail in section 2.1.1.

With a data set of observed values $(\mathbf{X}, \mathbf{f})$, function value predictions $\mathbf{f}_*$ can be inferred at new locations $\mathbf{X}_*$. The joint distribution of known values $\mathbf{f}$ and predictions $\mathbf{f}_*$ is itself a normal multivariate distribution:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^\top & \mathbf{K}_{**} \end{bmatrix}\right), \tag{4}$$

where $\mathbf{K}_* = K(\mathbf{X}, \mathbf{X}_*)$ denotes the $n \times n_*$ covariance matrix of n training points and n_* test points, and similarly for $\mathbf{K} = K(\mathbf{X}, \mathbf{X})$ and $\mathbf{K}_{**} = K(\mathbf{X}_*, \mathbf{X}_*)$. The joint posterior is described by [10]:

$$\mathbf{f}_* | \mathbf{X}_*, \mathbf{X}, \mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*), \tag{5}$$

with mean vector $\boldsymbol{\mu}_*$ and covariance matrix $\boldsymbol{\Sigma}_*$:

$$\boldsymbol{\mu}_* = \mathbf{K}_*{}^\top \mathbf{K}^{-1} \mathbf{f} \tag{6}$$

$$\boldsymbol{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_*{}^\top \mathbf{K}^{-1} \mathbf{K}_*. \tag{7}$$

Thus far, only noise-free observations of f have been considered. It is useful to include the possibility of noisy observations $y = f(x) + \varepsilon$ with independent Gaussian noise ε with variance σ_f^2. Noisy observations can be expected in some simulation circumstances (and in real-world measurements in particular), and can increase numerical stability in matrix calculations. With the inclusion of noisy observations, the joint distribution of observations $\mathbf{y}$ and predictions $\mathbf{f}_*$ becomes:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}_y & \mathbf{K}_* \\ \mathbf{K}_*^\top & \mathbf{K}_{**} \end{bmatrix}\right), \tag{8}$$

where $\mathbf{K}_y = K(\mathbf{X}, \mathbf{X}) + \sigma_f^2 \mathbf{I}$, with the noise added to the diagonal of the noise-free covariance matrix. The posterior mean and covariance is then described as:

$$\boldsymbol{\mu}_* = \mathbf{K}_*{}^\top \mathbf{K}_y^{-1} \mathbf{y} \tag{9}$$

$$\boldsymbol{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_*{}^\top \mathbf{K}_y^{-1} \mathbf{K}_*. \tag{10}$$

These are the key predictive equations for GPR, and they are used in building the surrogate model for the BO process. For our purposes the covariance between predictions is not useful information. The variance in predictions $\boldsymbol{\nu}_* = \mathbb{V}[\mathbf{f}_*]$ is sufficient. Therefore, we compute the diagonal of the covariance matrix $\boldsymbol{\nu}_* = \mathrm{diag}(\boldsymbol{\Sigma}_*)$.

To compute $\mathbf{K}_y$, it is typically faster to compute equations (9) and (10) by solving a linear system of equations using a Cholesky decomposition [11]. If a matrix $\mathbf{K}_y$ of size $n \times n$ is symmetric and positive semi-definite, it can be decomposed into a product of a lower triangular matrix L and its transpose as $\mathbf{K}_y = LL^\top$. The calculation of the Cholesky factor L takes time $n^3/6$, and enables computing equations (9) and (10) by solving triangular systems by forward and backward substitutions which require only $n^2/2$ operations [10]. The positive-definiteness of the covariance matrix is ensured by the covariance functions, and is discussed in the next section.

2.1.1 Covariance Functions and Hyperparameters

Each element of the covariance matrix is determined by a covariance function K, or **kernel**. Kernels typically encode the belief that two points $\mathbf{x}$ and $\mathbf{x}'$ in phase space should have a stronger correlation the smaller the distance $||\mathbf{x}-\mathbf{x}'||$ is between them. Kernels must be positive semi-definite functions in order to be valid covariance functions [10].

Different kernels are suitable for different problems. In materials experiments, we expect physical quantities to be smooth and continuous. For this research, I have used two types of kernels that are applicable to materials problems. The first one, used in fig. 3, is the **radial basis function** (RBF) kernel:

$$K(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2l^2}\right), \tag{11}$$

where the kernel variance σ^2 and lengthscale l are hyperparameters. The kernel variance σ^2 determines the range of the model output. Prior knowledge of or belief about the domain of the objective function can be incorporated into the kernel variance hyperparameter. The lengthscale l determines how quickly the model output can vary in its output. Examples of varying lengthscales are shown in fig. 4.

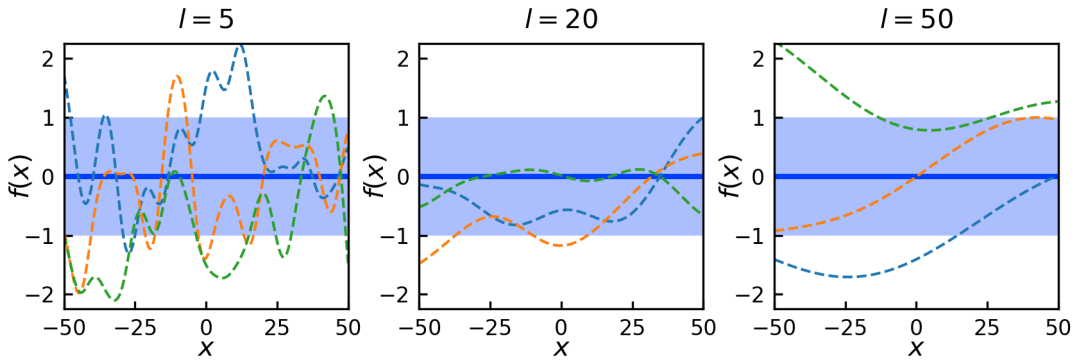


Fig. 4: Three examples of a 1D prior distribution with samples drawn. The priors are all defined by covariance matrices constructed with an RBF kernel of variance $\sigma^2 = 1$ but varying lengthscales. A lower lengthscale will reduce the correlation between two points a specific distance apart, allowing for random function samples to take sharper turns.

The kernel in equation (11) describes the correlation between data points in phase space with only one lengthscale. For non-isotropic systems with varying lengthscales per dimension, it can be useful to define a kernel for dimension d:

$$k_d(\mathbf{x}, \mathbf{x}') = \sigma_d^2 \exp\left(-\frac{|\mathrm{x}_d - \mathrm{x}'_d|^2}{2l_d^2}\right). \tag{12}$$

One-dimensional kernels can then be multiplied together into a **product** kernel:

$$K(\mathbf{x}, \mathbf{x}') = \prod_{d=1}^{D} k_d(\mathbf{x}, \mathbf{x}'). \tag{13}$$

The product kernel meets the criteria needed for constructing the covariance matrix. Note that the individual kernel variances σ_d^2 will always be multiplied together into one product. This means that the kernel variance will remain a single hyperparameter when using a product kernel, in contrast to the varying lengthscales.

Another kernel that is useful when the input range is periodic is the **standard periodic** (StdP) kernel:

$$k_d(\mathbf{x}, \mathbf{x}') = \sigma_d^2 \exp\left(-\frac{\sin^2\left(\frac{\pi}{T_d}|\mathrm{x}_d - \mathrm{x'}_d|\right)}{2l_d^2} \right), \tag{14}$$

where T_d is the period of the input dimension d. If a problem is only periodic in one particular dimension, a product kernel can combine both RBF and StdP kernels.

For a set of data, there are a multitude of possible models with different hyperparameters. Certain hyperparameters—such as the period of an input dimension—could be known and then be defined as fixed, while others can be optimized. The optimization of hyperparameters $\boldsymbol{\theta}$ ensures the best fit for the surrogate model to N data points. This is done by maximizing the **log marginal likelihood**:

$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^\top \mathbf{K}_y^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}_y| - \frac{N}{2}\log 2\pi. \tag{15}$$

2.1.2 Gradient Observations in GPR

So far, only observations of function values y have been considered for GPR. The topic of this research is the inclusion of gradient observations in BOSS, and consequently, in GPR. Fitting the surrogate model with gradient observations provides it with more information, which means less data is required for the model to emulate the objective function (see fig. 5).

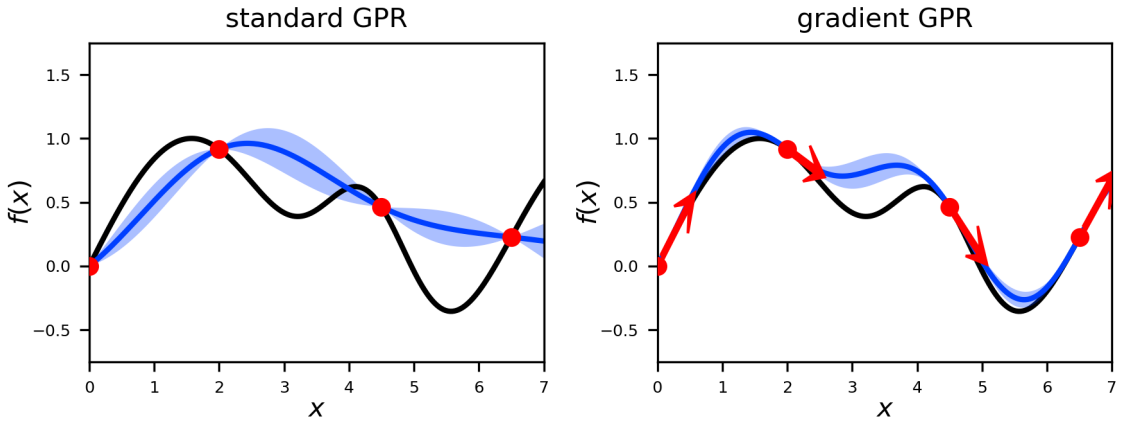


Fig. 5: A 1D comparison between two different GPR model fits. In the left panel, a model has been fit to four data points that contain just the function values. In the right panel, a model has been fit to four data points that contain both the function value and gradients. The right model is a better fit with the same number of data points, as the posterior mean of the model must fit to the function values and gradients.

Each component of the gradient, i.e., each partial derivative of the function, can be considered a separate output channel for GPR. The joint distribution in equation (8) can be expanded to include gradient observations $\nabla\mathbf{y}$ and gradient predictions $\nabla\mathbf{f}_*$:

$$\begin{bmatrix} \mathbf{y} \\ \nabla\mathbf{y} \\ \mathbf{f}_* \\ \nabla\mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \boldsymbol{\Gamma}_y & \boldsymbol{\Gamma}_* \\ \boldsymbol{\Gamma}_*^\top & \boldsymbol{\Gamma}_{**} \end{bmatrix}\right), \tag{16}$$

where $\boldsymbol{\Gamma}$ is the covariance matrix of multiple output channels. $\boldsymbol{\Gamma}$ collects the covariance of all individual channels, but also includes the cross-covariance between channels. We therefore need covariances between function values and partial derivatives, and covariances between partial derivatives of different dimensions:

$$\operatorname{cov}\left(f(\mathbf{x}), \frac{\partial f(\mathbf{x}')}{\partial \mathrm{x}'_i}\right) = \frac{\partial K(\mathbf{x}, \mathbf{x}')}{\partial \mathrm{x}'_i}, \quad \operatorname{cov}\left(\frac{\partial f(\mathbf{x})}{\partial \mathrm{x}_i}, \frac{\partial f(\mathbf{x}')}{\partial \mathrm{x}'_j}\right) = \frac{\partial^2 K(\mathbf{x}, \mathbf{x}')}{\partial \mathrm{x}_i \partial \mathrm{x}'_j}, \tag{17}$$

where i, j are dimensions. Replacing $\mathbf{K}$ with $\boldsymbol{\Gamma}$ in the predictive equations of (9) and (10) will yield:

$$\begin{bmatrix} \boldsymbol{\mu}_* \\ \boldsymbol{\mu}_*^{\nabla} \end{bmatrix} = {\boldsymbol{\Gamma}_*}^\top \boldsymbol{\Gamma}_y^{-1} \begin{bmatrix} \mathbf{y} \\ \nabla\mathbf{y} \end{bmatrix} \tag{18}$$

$$\begin{bmatrix} \boldsymbol{\nu}_* \\ \boldsymbol{\nu}_*^{\nabla} \end{bmatrix} = \operatorname{diag}\left(\boldsymbol{\Gamma}_{**} - {\boldsymbol{\Gamma}_*}^\top \boldsymbol{\Gamma}_y^{-1} \boldsymbol{\Gamma}_*\right), \tag{19}$$

where $\boldsymbol{\mu}_*^{\boldsymbol{\nabla}}$ and $\boldsymbol{\nu}_*^{\boldsymbol{\nabla}}$ are the means and variances for the partial derivative posteriors. The multi-output covariance matrix $\boldsymbol{\Gamma}$ of two D-dimensional datasets $\mathbf{X}$ and $\mathbf{X}'$, containing N and N' data points respectively, is:

$$\boldsymbol{\Gamma}(\mathbf{X},\mathbf{X}') = \begin{bmatrix} K(\mathbf{X},\mathbf{X}') & \frac{\partial}{\partial x'_1}K(\mathbf{X},\mathbf{X}') & \dots & \frac{\partial}{\partial x'_D}K(\mathbf{X},\mathbf{X}') \\ \frac{\partial}{\partial x_1}K(\mathbf{X},\mathbf{X}') & \frac{\partial^2}{\partial x_1 \partial x'_1}K(\mathbf{X},\mathbf{X}') & \dots & \frac{\partial^2}{\partial x_1 \partial x'_D}K(\mathbf{X},\mathbf{X}') \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial x_D}K(\mathbf{X},\mathbf{X}') & \frac{\partial^2}{\partial x_D \partial x'_1}K(\mathbf{X},\mathbf{X}') & \dots & \frac{\partial^2}{\partial x_D \partial x'_D}K(\mathbf{X},\mathbf{X}') \end{bmatrix}, \tag{20}$$

and is of size $N(1+D)\times N'(1+D)$. This presents a scalability issue for large datasets of high dimensionality. While standard GPR inference takes $O(N^3)$ time, training and inference with gradient observations scale as $O(N^3D^3)$ [12]. I will return to this topic of poor scaling and what it means for the efficiency of BOSS when I discuss the results of chapter 4.

2.2 The Acquisition Function

After a model is fit to a number of data points, we need to know which atomic configuration to evaluate next in order to gain as much information about the objective function as possible. This configuration can be found with the acquisition function, which takes the posterior mean and variance as inputs. There are different types of acquisition functions—such as expected improvement [13], gradient descent [14], and entropy search [15]—but in this research I have used the **exploratory lower confidence bound** (ELCB) acquisition function.

ELCB, among other acquisition functions, balances *exploration* and *exploitation*. Exploration means that the BO process searches areas where the model is uncertain in its prediction, i.e., areas with high posterior variance. This aspect of the acquisition function ensures an inherent space-filling search process. Exploitation searches areas of low predicted energy, i.e., areas with low posterior mean, increasing model accuracy where the global minimum is more probable to be found. These attributes can be quantified with the function $a(x)$:

$$a(x) = \mu_*(x) - \kappa\,\sigma_*(x), \tag{21}$$

where $\mu_*(x)$ and $\sigma_*(x)$ are the posterior mean and standard deviation of the uncertainty. The exploratory weight κ is described by [16]:

$$\kappa = \sqrt{2\log\left(\frac{N^{D/2+2}\pi^2}{3\epsilon}\right)}, \tag{22}$$

where N is the number of observed data points, D is the dimensionality of the problem, and ϵ is a small constant (chosen as $\epsilon = 0.1$ in BOSS). According to Gutmann and Corander [16], the exploratory weight should increase with iteration number—i.e., size of the collected data set—so that the BO process will not get stuck in a local minimum. The location of the next evaluation x_{next} is at the global minimum of the acquisition function:

$$x_{next} = \operatorname{argmin} a(x), \tag{23}$$

and can be computed by conventional global minimization methods, as the inputs of the acquisition function are the predictions of the surrogate model which are relatively inexpensive to evaluate.

2.3 Structure Search Application

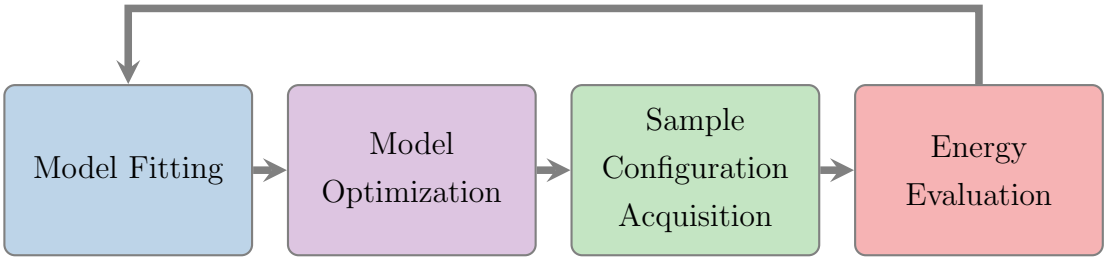


Fig. 6: The BOSS iteration cycle. The surrogate model is fit to energy values using GPR. It is then optimized for an estimate of the global minimum. A sample configuration that would best improve the model is generated by the acquisition function. The energy value of this configuration is then evaluated by the objective function and added to the total data set, which the model is then fit to, and the process repeats.

BO can be applied to a structure search problem in order to determine the atomic configuration with the lowest total energy—i.e., the most stable state—in what is referred to as BOSS. The configuration with the lowest energy is represented as the global minimum of the PES. The surrogate model of BO is fit to known energy values in order to emulate the PES, and can then be optimized for an estimate of the most stable state. If the estimate is inadequate, and more the model requires more data in order to produce a better estimate, the acquisition function will generate a sample configuration that will provide the model with the highest amount of information. The energy for this configuration is evaluated with the objective function—e.g., a simulation—and added to the total data set. The model is then fit to this expanded data set and optimized for a new estimate, and the process repeats (see fig. 6).

2.3.1 Convergence

The process can be performed for a set number of iterations, or it can be discontinued when the estimate of the most stable state is deemed to be an adequate approximation of the true energy value. One method of determining the adequacy of the approximation is to track the change in value for consecutive estimations. Once the change has stayed below a threshold for a number of iterations, the process is be said to have converged and can be discontinued. The threshold and number of iterations is defined by the user.

For my research, I studied structure search problems where the most stable state was known. Therefore, I could use a different method for determining convergence. The definition of convergence, that will be utilized to measure the efficiency of BOSS for the results presented in chapter 4, is for this research as follows:

BOSS has converged for a threshold of ε in i iterations, if the absolute difference between the global minimum prediction $\hat{\mu}$ and the true global minimum f_{min} has stayed below $|\,\hat{\mu} - f_{min}\,| < \varepsilon$ for iterations $\{i,\, i{+}1,\, ...,\, i{+}n\}$

For all structure search problems, I use $n = 10$ and three convergence thresholds of decreasing orders of magnitude: $\varepsilon \in \{10^{-1}, 10^{-2}, 10^{-3}\}$. However, there is a primary threshold per problem that is deemed as sufficiently accurate.

2.3.2 Degrees of Freedom

An atomic configuration consisting of N atoms, each with Cartesian position (x, y, z), will have $3N$ independent degrees of freedom. High-dimensional energy surfaces can become excessively convoluted, which requires many data points for exploration. In the case of including gradient observations, matrix calculations scale poorly with high dimensionality, as discussed in section 2.1.2. This issue can be alleviated with a "building-block" approximation approach. If allowed by chemical rules, certain groupings of atoms—such as aromatic rings or functional groups—can be treated as rigid objects, or "blocks". The degrees of freedom can then be constrained to the translational and/or rotational motions of the blocks. This approach produces an approximation of the PES with reduced dimensionality, whose global minimum can be found with BOSS. When the approximate global minimum configuration has been determined, the constraints on the degrees of freedom can be relaxed, and the true minimum can be found with gradient descent.

The building-block approach is applicable to the structure search problems that I have studied for this research, and they will be introduced in the next sections.

2.3.3 Alanine Conformer Search

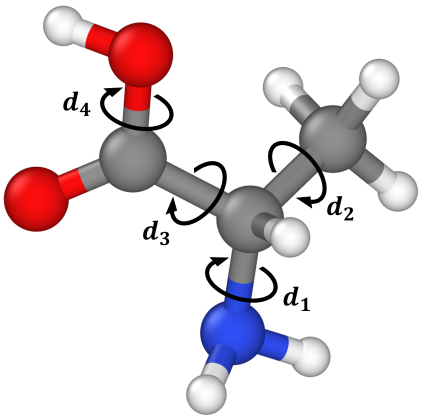


Fig. 7: An alanine molecule. Four dihedral angles determine the rotations of functional groups. Grey atoms correspond to carbon C, white atoms to hydrogen H, red atoms to oxygen O, and blue atoms to nitrogen N.

The objective of the alanine conformer search is to find the most stable state of an alanine molecule (see fig. 7). The molecule consists of 13 atoms in total. However, there are four identifiable functional groups that allows for the application of the building-block approach. The rotations of these functional groups are determined by four dihedral angles. An atomic configuration of the molecule can therefore be represented as a state vector of the four dihedral angles. Thus, the problem has been reduced to four dimensions.

The alanine molecule is computationally simulated on the IT Center for Science (CSC) Puhti supercomputer [17], using a force field simulation method. The total energy of the molecule is computed using AMBER [18] code, with a general AMBER force field (GAFF). The potential energy between a pair of atoms is described by the functional form [19]:

$$\begin{aligned} V(\mathbf{r}_i) = &\sum_{bonds} K_r(r - r_{eq})^2 + \sum_{angles} K_\theta(\theta - \theta_{eq})^2 \\ &+ \sum_{dihedrals} \frac{V_n}{2}\left[1 + \cos(n\phi - \gamma)\right] + \sum_{i<j}\left[\frac{A_{ij}}{R_{ij}^{12}} - \frac{B_{ij}}{R_{ij}^{6}} + \frac{q_i q_j}{\epsilon R_{ij}}\right], \end{aligned} \tag{24}$$

where r_{eq} and θ_{eq} are equilibrium structural parameters, K_r, K_q, V_n are force constants, n is multiplicity and γ is the phase angle for torsional angle parameters.

Because data from a previous BOSS study of the same problem was available, a surrogate model could be fit to this data and be used as an **emulator function**. I used this emulator function to evaluate energies and gradients. This way a local installation of AMBER could be avoided and the tests could be performed quickly.

2.3.4 Benzene Adsorption Search

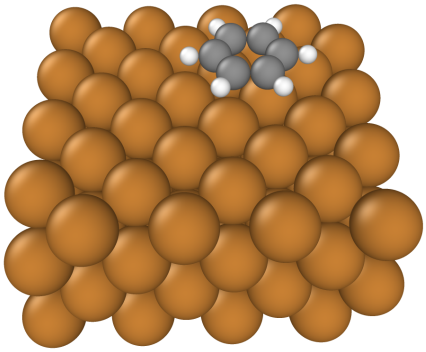


Fig. 8: A benzene molecule above a Cu(100) surface. Grey atoms correspond to carbon C, white atoms to hydrogen H, and orange atoms to copper Cu.

The objective of the benzene adsorption search is to find the most stable position of a benzene adsorbed to a copper surface. The building-block approach can be applied by treating the benzene molecule as a rigid object, and the copper surface as fixed. In order to further reduce the problem to three dimensions, I do not consider the rotation of the benzene molecule, as the tilt of the molecule is known from a previous study. The most stable position can then be represented as the cartesian position of the center of mass of the benzene molecule, that minimizes the total energy of the system consisting of both the molecule and the surface.

The system is simulated on the CSC Puhti supercomputer, using DFT [20]. To employ DFT, we have used FHI-aims code [21]. With DFT, the ground-state solution for a many-body system can be approximated using only the electron density distribution $n(\mathbf{r})$. Properties of a system can be acquired from the total energy functional, expressed in DFT as:

$$E[n] = T_s[n] + \int d\mathbf{r}\, V_{ext}(\mathbf{r})\, n(\mathbf{r}) + E_H[n] + E_{xc}[n], \tag{25}$$

where T_s is the kinetic energy, $V_{ext}(\mathbf{r})$ is the external potential, E_H is the Hartree (or Coulomb) energy and E_{xc} is the exchange-correlation energy. The first three terms in eq. (25) can be calculated exactly, but the exchange-correlation energy needs to be approximated. For this approximation, we have used the generalized gradient approximation (GGA) by Perdew, Burke, and Ernzerhof (1996) (PBE) [22], and the dispersion corrections of Tkatchenko and Scheffler (2009) [23].

3 Computational Implementation

In this chapter, I will describe the computational implementation I have performed in order to investigate the role gradient observations have in BOSS. The implementation was split into two tasks, each in a different Python package. The first task, described in section 3.1, was to incorporate gradient information into the surrogate model. The second task, described in 3.2, was to integrate the augmented surrogate model into BOSS. I describe how the full implementation was validated at the end of this chapter, in section 3.3.

3.1 Implementation of Gradient Observations in GPR

The objective of this implementation was to incorporate the mathematical theory of gradient observations in GPR, described in section 2.1.2, for the dependency that BOSS utilizes for GPR: GPy [8]. This Python package creates and fits the surrogate model that can then be used to make predictions. The model can be created with a wide selection of possible kernels. For the purposes of this research, I have focused solely on RBF (eq. 12) and StdP (eq. 14) kernels, along with the product kernel (eq. 13) that can combine kernels of different dimensions.

The principal object of focus for this implementation is the multi-output covariance matrix $\mathbf{\Gamma}$ (eq. 20), with which the surrogate model is augmented with gradient information. The $\mathbf{\Gamma}$ matrix can be subdivided into its constituent parts as:

- Multi-output covariance matrix $\mathbf{\Gamma}$
  - Product kernel
    - RBF kernel
    - StdP kernel

For the surrogate model to make predictions (see eqs. 18 and 19), the $\mathbf{\Gamma}$ matrix requires first- and second-order derivatives of its constituent kernels. These derivatives were implemented as functions in work done by Eero Siivola [9], but generalization was needed for N dimensions. This mainly involved debugging of relevant functions, and was performed in a bottom-up approach by first focusing on the constituent kernels, and then moving up towards the $\mathbf{\Gamma}$ matrix. Minimization of the surrogate model means calculating the gradient of the posterior mean and variance, which involves differentiating the $\mathbf{\Gamma}$ matrix. This differentiation requires third-order derivatives of the constituent kernels, which I implemented as functions.

3.2 Implementation of Gradient Observations in BOSS

With the implementation in GPR of the previous section, the surrogate model can be fit with gradient observations. In this section, I describe how this augmented surrogate model is incorporated into the BOSS process. First, I describe the BOSS Python package in detail, followed by my implementation.

3.2.1 Standard BOSS

BOSS [1] is under continuous development at Aalto University by the Computational Electronic Structure Theory (CEST) group and at University of Turku. BOSS is written in Python 3, is distributed as a PyPi package, and can be run via a command-line interface or directly in Python scripts.

BOSS facilitates structure search by building surrogate models, refining the models iteratively through smart sampling, and inferring global minima. External data, from the objective function to be minimized, is incorporated into BOSS by a Python interface, called the "user function". The properties of the BOSS algorithm—such as the number of iterations, convergence thresholds, kernel variants, and acquisition types—are set by the user in an input file. BOSS relies on the GPy package for GPR and kernels. Some essential algorithm settings can be determined automatically by the program if not defined by the user.

BOSS will track key metrics—such as data acquisitions, model hyperparameters and global minimum predictions. These metrics will be stored and written to output files during the optimization, and are returned to the user when the process is completed. If the optimization is inadvertently interrupted or deliberately halted by the user, it can be restarted at a later time from the iteration at which the process was stopped. A completed optimization can also be restarted for additional iterations if required. BOSS also provides the user the ability to study the full history of the optimization process through post-processing. Options for post-processing include plotting the evolution of key metrics throughout the optimization, and reconstructing and visualizing surrogate models at a specific iteration step.

BOSS Internal Structure

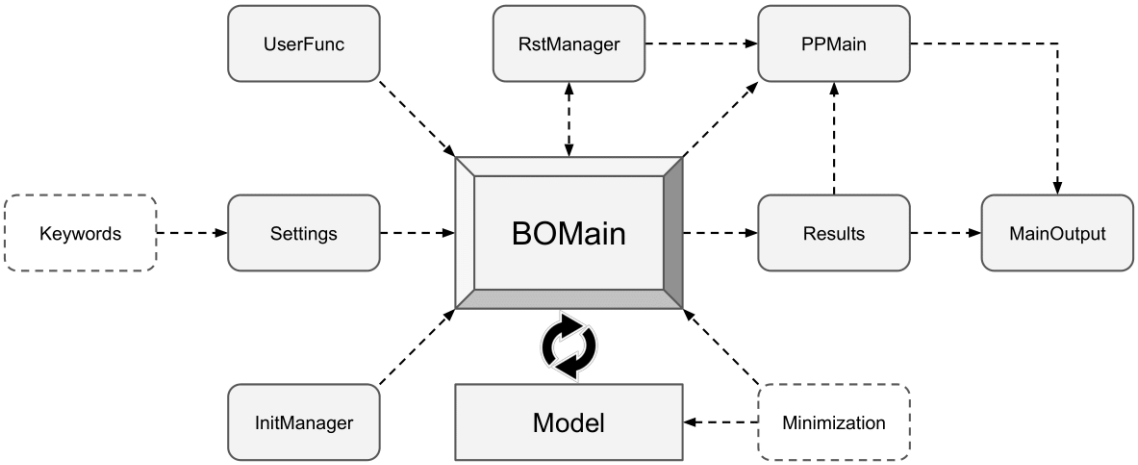


Fig. 9: A diagram of the internal structure of BOSS that shows its constituent modules and how they relate to each other.

I will now describe the internal structure and modules of BOSS in more detail. The BOSS algorithm has many parts and utilizes the object-oriented programming of Python. A diagram of the BOSS structure can be seen in Fig. 9. The BOSS algorithm uses the following modules to facilitate the structure search process:

- The **BOMain** object is the main program for BOSS. It initializes and contains all other objects. The BO loop is implemented in BOMain.
- The **UserFunc** object is the source of the external data from the objective function. The object ensures that the data shapes of the inputs and outputs are consistent.
- The **Model** object contains the surrogate model and all functionalities it needs, like predictions and hyperparameter optimization. The BOSS model class is a wrapper for the GPy model class that performs GPR and makes predictions.
- **Minimization** is a utility applied when minimizing the model and the acquisition function. The model is minimized to predict the global minimum. The acquisition function is minimized in order to determine the location of the next evaluation.
- The **Settings** contain all the algorithm settings that determine the optimization conditions.

- ☐ The **InitManager** determines the initial data point locations for the optimization, typically chosen at random across the phase space. The optimization can also be started with specific data points chosen by the user. The initial data points are determined before the main loop starts.

- ☐ The **RstManager** implements reading and writing to the `boss.rst` output file. If an optimization was interrupted, the RstManager will give BOMain the data required in order to restart the optimization from the interruption point.

- ☐ The **MainOutput** writes to the main `boss.out` output file during the BOSS run. The output file contains information about every iteration step—such as data point acquisition, global minimum prediction, and iteration time.

- ☐ The **Results** object stores data, such as acquisitions and model hyperparameters, during the optimization. When the optimization is complete, the Results object can be used by other parts of BOSS if they require the data. The Results object can also be returned directly to the user if BOSS is run in a Python environment.

- ☐ The **PPMain** object performs post-processing for BOSS runs. Post-processing options including plotting of key metrics and visualizing surrogate models.

3.2.2 Implementation of Augmented Surrogate Model

I incorporated the augmented surrogate model into BOSS by creating a new model class. This new model class was built on a copy of the standard model class, ensuring that both classes had the same essential functionalities, such as fitting with GPR and making predictions. The augmented model class performs these functionalities with gradient observations included, and is enabled with a keyword in the input file.

Along with the new model class, minor changes were made in other modules:

- **BOMain**: Added functionality for main program to utilize gradient data.
- **UserFunc**: The user function expects the objective function to calculate gradients along with energies if the relevant keyword is enabled.
- **Settings**: Added keyword for enabling observations of gradients.
- Added gradient data to the **RstManager**, **MainOutput**, and **Results** object.

3.3 Code Validation

The code was validated during the implementation in a number of ways. The individual kernels of GPy were first verified to produce the correct output. With the bottom-up approach, I checked that the $\mathbf{\Gamma}$ matrix was ultimately calculated correctly, through printout statements and debugging. A number of simple optimization test functions were then analyzed with both the standard and the augmented GPR fit. Measuring the root-mean-squared error between predicted values and true function values, the augmented GPR fit performed better than its standard counterpart. The augmented surrogate model was then incorporated into BOSS, and then tested on a simple 1D function to confirm that all changes made to the program work as intended.

The 1D function that was tested is written as:

$$f(x) = \sin(x) + 1.5\exp\left(-(x-4.3)^2\right), \tag{26}$$

with global minimum $f_{min} \approx -0.36$. BOSS can be used to make a prediction $\hat{\mu}$ for the global minimum each iteration. I performed the test with both the standard variant of BOSS, and the augmented variant with gradient observations included. The results are illustrated in fig. 10 below:

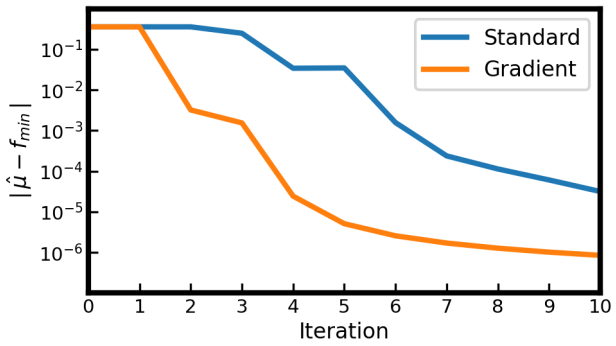


Fig. 10: Results from testing BOSS on a simple 1D function. The absolute distance between the global minimum prediction $\hat{\mu}$ and the true minimum f_{min} is plotted as a function of iteration. The y-scale is logarithmic. Results that include gradients observations are plotted in orange.

The results are depicted as the absolute distance between the global minimum prediction and the true minimum, with a logarithmic scale. The results confirmed that the code works as intended, as both BOSS variants make predictions that tend towards the true minimum as data increases. The results also suggest that including gradients observations accelerates the process, but this will be investigated further in the next chapter.

4 Structure Search Results

In this chapter, I will present, examine and discuss the results from the two computational structure search studies introduced in section 2.3 that have been studied to investigate the role including gradient information has on the efficiency of BOSS. Throughout this chapter, I will refer to **gradient BOSS** as the augmented BOSS process where gradient observations are included, and **standard BOSS** as the process where the observations are left out. The primary metrics with which I examine the results will be the amount of data (i.e., number of iterations) and time required for convergence, as defined in section 2.3.1. Results will be presented with three convergence thresholds of decreasing orders of magnitude. However, each study has a primary threshold that is deemed as sufficiently accurate and not excessively constrained. The time required for convergence will be examined solely based on this primary threshold.

For each study, I will first describe the setup and design of the test, followed by results and a short discussion. First, the alanine conformer search study will be presented in section 4.1, followed by the benzene adsorption study in section 4.2.

4.1 Alanine Conformer Search

4.1.1 Test Design

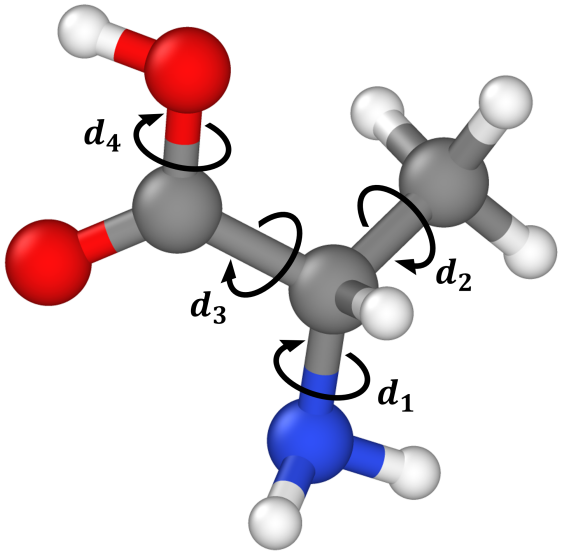


Fig. 11: A model of the alanine $C_3H_7NO_2$ molecule that is the subject of this structure search study. The four dihedral angles depicted determine the rotation of the functional groups. Grey atoms correspond to carbon C, white atoms to hydrogen H, red atoms to oxygen O, and blue atoms to nitrogen N.

The purpose of the alanine conformer search study is to find the atomic configuration of an alanine molecule with the lowest energy, indicating the most stable state of the molecule. The dimensionality of the search space is reduced from the collection of Cartesian positions for every atom, to the rotational phase space of four functional groups that also describe the full range of possible atomic configurations, given that the atoms retain fixed positions within the functional groups. The rotation of the functional groups is determined by dihedral angles, illustrated in fig. 11.

In order to investigate how gradient BOSS scales with dimensionality, the study is split into two separate studies with different dimensions. In one study, two angles are kept fixed and two are allowed to vary. I refer to this study as the **2D** alanine conformer search study. In the other study, all four angles are allowed to vary, and I refer to this study as the **4D** alanine conformer search study.

The energy of the molecule is computed by an emulator function, which is a converged GPR fitted surrogate model from a previous BOSS run, where the energies were calculated using an AMBER force field, described in section 2.3.3, with energy units of kcal/mol. The primary convergence threshold for alanine conformer search is 1×10^{-1} kcal/mol.

parameter	2D	4D
variables	d_1, d_4	d_1, d_2, d_3, d_4
bounds [deg.]	[-50, 310], [-50, 310]	[-50, 310], [-50, 70], [-50, 310], [-50, 310]
range [kcal/mol]	[0, 50]	[0, 50]
kernels	StdP (x2)	StdP (x4)
noise variance	1×10^{-6}	1×10^{-6}
initial points	2	2
iterations	50	200

Table 1: Test parameters for each case study (2D or 4D alanine conformer search).

The parameters for each test are listed above in table 1. For the 2D study, angles d_2 and d_3 are fixed to 60°, while for the 4D study all angles vary. All angles are rotated a full 360 degrees, with the exception of d_2, which has 120 degree rotational symmetry. For each study, a product of standard periodic (StdP) kernels per dimension is used. As a prior on the kernel variance, the energy value of the global minimum is expected to lie between 0 and 50 kcal/mol. I allow for a fixed gaussian noise variance of 1×10^{-6} in function evaluations. The search is initialized with two data points, chosen from random locations across search space. The process is then performed for 50 iterations in the 2D study and 200 iterations in the 4D study. As the search process is stochastic, it is repeated ten times for both standard and gradient BOSS for statistical purposes.

4.1.2 Results

Global Minimum Predictions

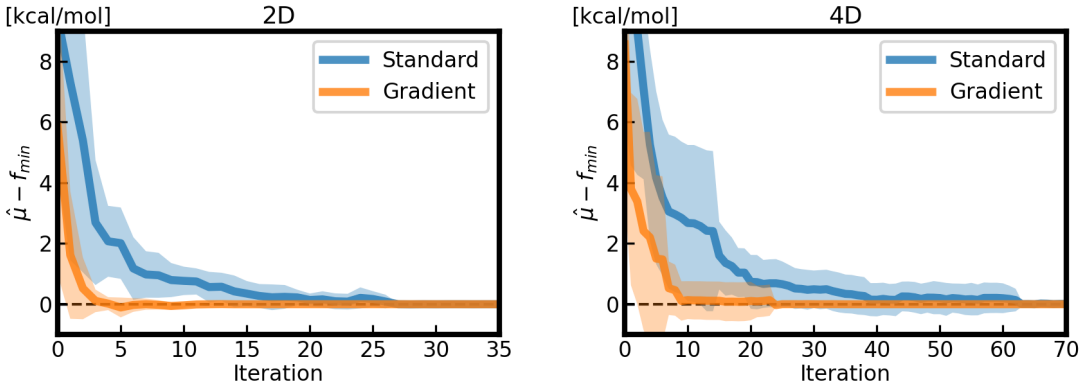


Fig. 12: Mean global minimum predictions $\hat{\mu}$ as a function of iteration, for alanine conformer search. The predictions are averaged over ten runs for each BOSS variant (standard and gradient). The mean predictions are plotted as full lines, with one standard deviation above and below the mean plotted as the lighter area surrounding the lines.

The global minimum predictions $\hat{\mu}$ are plotted above in fig. 12, with results from both the 2D and 4D studies. The predictions are obtained through global minimization of the surrogate model. The results from the ten runs of each BOSS variant (standard and gradient) are combined by averaging the predictions. The y-scale of the graph is shifted by the known true global minimum f_{min}, so the predictions should ideally converge to zero as the number of iterations increases. The same results are plotted with a logarithmic scaling in fig. 13, to more closely observe the behaviour of the predictions with increasing iteration.

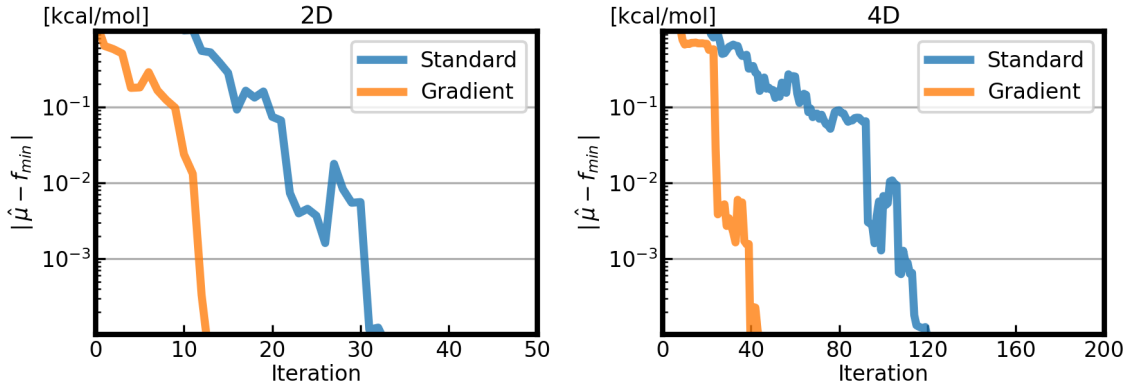


Fig. 13: Mean absolute difference between global minimum predictions $\hat{\mu}$ and the known true global minimum f_{min}, per iteration, for alanine conformer search. The predictions are averaged over ten runs for each BOSS variant (standard and gradient). The y-scale is logarithmic.

Convergence

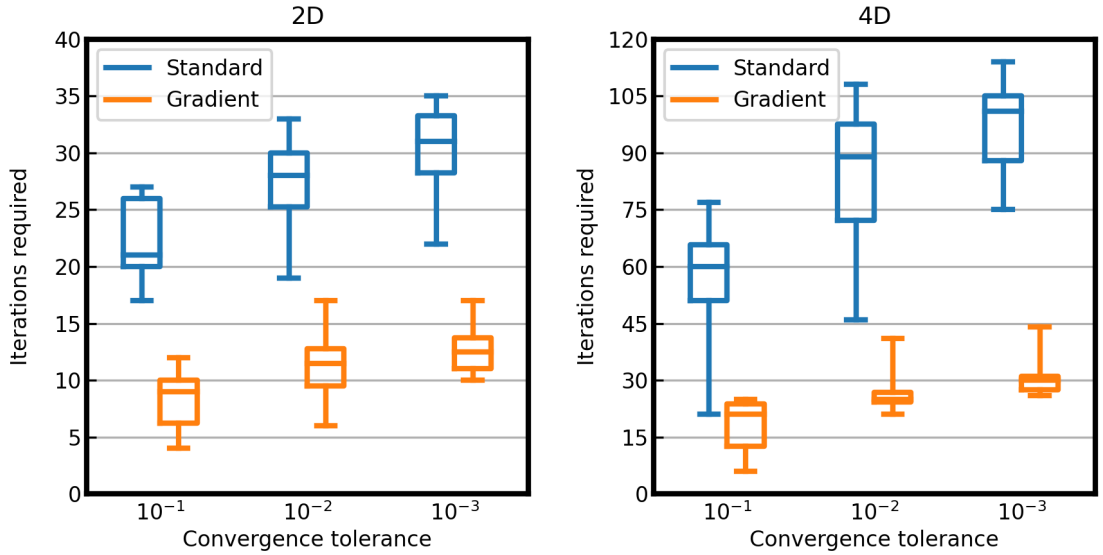


Fig. 14: Box plot of iterations required for convergence within a specific tolerance ε, for alanine conformer search. Each box contains data for ten runs of each BOSS variant (standard or gradient).

With the convergence definition of section 2.3.1, we can determine how quickly each BOSS run converges. The fewer number of iterations required for convergence N is, the quicker a BOSS run has converged. I have chosen three levels of convergence thresholds for investigating how the accuracy of the models increase with iterations. Fig. 14 displays statistics from both studies (2D and 4D), for each type of BOSS variant (standard or gradient). Table 2 displays the mean number of iterations required $\bar{N}$ for each BOSS variant and study, given a convergence criterion ε. The primary convergence threshold of 1×10^{-1} kcal/mol is bolded.

	2D			4D		
ε	$\bar{N}_s$	$\bar{N}_g$	$\bar{N}_g/\bar{N}_s$	$\bar{N}_s$	$\bar{N}_g$	$\bar{N}_g/\bar{N}_s$
10^{-1}	**22.2**	**8.3**	**0.37**	**54.9**	**18.3**	**0.33**
10^{-2}	27.0	11.2	0.41	84.8	26.8	0.32
10^{-3}	30.3	12.5	0.41	96.6	31.4	0.33

Table 2: Mean number of iterations required $\bar{N}$ for a given convergence tolerance ε. The number of iterations is averaged over ten runs of each BOSS variant (standard or gradient) and study (2D or 4D alanine conformer search). The primary convergence threshold is bolded.

Model Visualization

A way of monitoring that the BOSS process is working correctly is to not only monitor the global minimum prediction, but to also investigate the surrogate model and how it evolves during the process. By making predictions for a grid of locations across phase space, the models can be visualized. This is what I will be illustrating in this section for the 2D alanine conformer search. As a reference, the PES of the true function is illustrated to the right in fig. 15. The true global minimum location is marked by the star.

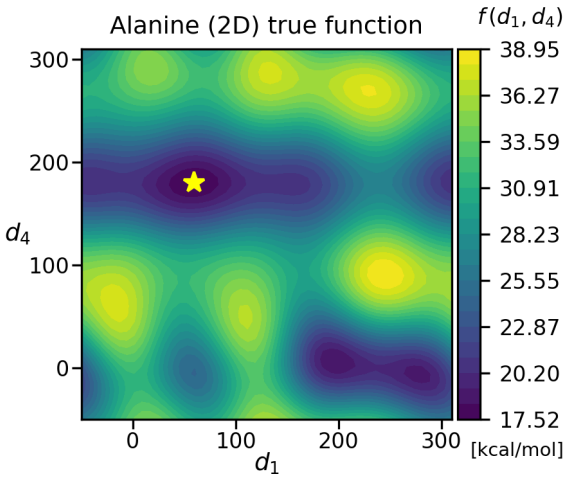


Fig. 15: True function of alanine (2D) conformer search. The star denotes the location of the global minimum.

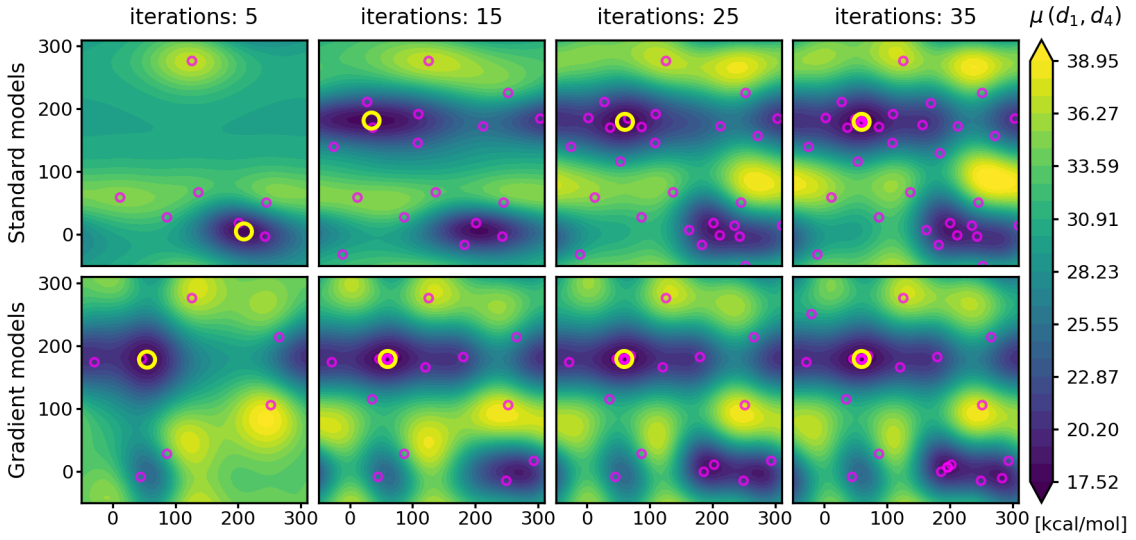


Fig. 16: A sequence of surrogate models, with BOSS iterations increasing to the right. Purple markers are data acquisition locations. The large yellow marker is the location of the global minimum prediction.

Depicted in fig. 16 above are surrogate models at different iterations in the BOSS process, for one particular pair of BOSS variants. The top and bottom rows illustrates standard and gradient models, respectively. The locations of data acquisitions and the global minimum prediction are also depicted. The color scale for the surrogate models is the same for the true function in fig. 15. The scaling does, however, accommodate for predictions that are outside the true range of function values.

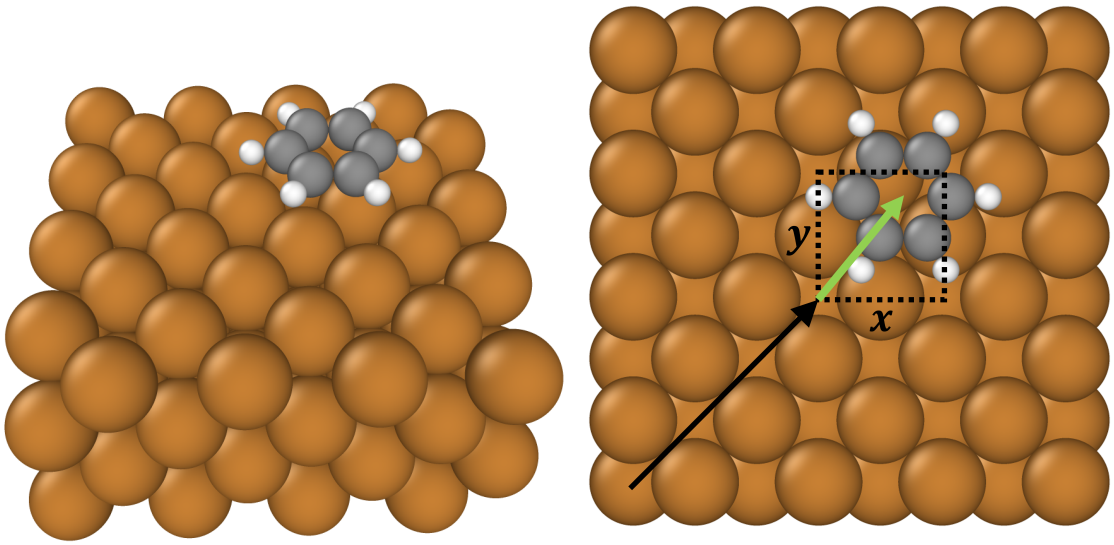
Discussion

The results indicate that global minimum predictions made by gradient BOSS have values that are closer to the true global minimum than standard BOSS, when less data is available (i.e., when fewer iterations have been performed). This can be observed in in figures 12 and 13, as gradient BOSS predictions tend towards the true minimum more rapidly than standard BOSS predictions. The box plots of fig. 14 clearly illustrate that including gradient observations reduces the number of iterations required for any convergence of any threshold. The height of the boxes suggest that standard BOSS runs are more inconsistent in the number of iterations required for convergence. This could be due to the set of random points used to initialize each pair of runs. Standard BOSS could be more vulnerable to a suboptimal set of initial points, i.e., random sets that would produce inadequate surrogate models of the objective function. Including gradient observations produces surrogate models that more closely resemble the objective function for any set of points, so gradient BOSS should be less vulnerable to a suboptimal set of initial points, which means less variance in iterations required for convergence across multiple runs. Table 2 shows how including gradient observations cuts the number of iterations required for convergence by over one half in for all convergence thresholds in the 2D study. In the 4D study, iterations required are cut by two thirds.

The model visualizations of fig. 16 illustrate how surrogate models of gradient BOSS more closely resemble the true function of fig. 15 than standard BOSS models, with the same amount of data. At five iterations, the gradient BOSS surrogate model has found the correct location for the global minimum, while the standard BOSS model has made its prediction at an incorrect local minimum. With less data, the landscape of the standard BOSS model is quite a poor predictor of the true function, while the landscape of the gradient model is a better indicator of the function's peaks and valleys.

4.2 Benzene Adsorption Search

4.2.1 Test Design



(a) The simulation cell. **(b)** A top-down view perspective.

Fig. 17: Figure (a) depicts the simulation cell used in the benzene adsorption search study. Figure (b) shows a top-down view of the simulation cell, with the smallest repeating surface unit outlined as the dashed black square. The bounds for the lateral position of the molecule (green vector) is defined as the dimensions of the smallest repeating surface unit. In order to search a surface unit that is not located at the edge of the simulation cell, a lateral offset (black vector) is applied to the lateral coordinates of the molecule.

The benzene adsorption study involves determining the position of a benzene molecule that has the lowest energy above a Cu(100) surface. The tilt of the molecule is known from a previous study. The position with the lowest energy is represented as the global energy minimum of the combined system of the molecule and the surface. The total energy is calculated with a DFT simulation (described in section 2.3.4), with units of eV. The primary convergence threshold is 1×10^{-2} eV, as we determine it to be sufficiently accurate. For a model of the combined system of the molecule and the surface, I used a model that was computed in a previous study, that had a determined simulation cell size and surface thickness. In order to constrain the structure search to three dimensions, the benzene molecule is treated as a rigid object with position vector (x, y, z) (defined as the center of mass of the molecule) above the fixed surface. The tilt and rotation of the molecule is set as fixed for the same purpose. The molecule is positioned with the Python package `ase` [24].

The model is comprised of a slab of copper atoms that is four atomic layers thick, and a benzene molecule that is positioned above the surface. The simulation cell is periodic in x and y, and wide enough so that the benzene molecule would not interact with any of its periodic counterparts. As x and y are periodic variables, the search is constrained to within the smallest repeating surface unit (see fig. 17 (b)). The bounds for x and y are defined as the dimensions of this smallest surface unit. While not strictly necessary, a lateral offset is applied to the position of the molecule so that the search is within a surface unit that is not located at the edge of the simulation cell. A vertical offset is also applied to the z-position of the molecule in order to define the z bounds as the height of the molecule above the surface.

parameter	
variables	x, y, z
bounds [Å]	[0, 3.632], [0, 3.632], [2.0, 3.5]
range [eV]	[-2, 0]
kernels	StdP (x2), RBF (x1)
noise variance	1×10^{-6}
initial points	5
iterations	65

Table 3: Test parameters for benzene adsorption search.

The BOSS process is started with five initial data points, chosen at random across phase space. The process is then continued for 65 iterations, as knowledge of the previous study states that the global minimum should be found within this number of iterations. The width of the smallest repeating surface unit is 3.632 Å, and the global minimum is expected to be found at 2.0 and 3.5 Å above the surface. The lowest energy value is expected to be between -2 and 0 Å. As x and y are periodic, a standard periodic kernel is chosen for each of those dimensions, while an RBF kernel is chosen for the z dimension. These kernels are then multiplied together into a product kernel. I allow for a fixed Gaussian noise variance of 1×10^{-6} in function evaluations. As the five initial data points are chosen randomly and the process is stochastic, the structure search is repeated only three times—as the simulation is more computationally expensive—for each BOSS variant (standard and gradient) for statistical purposes. Each pair of variants are initialized with the same set of random data points.

4.2.2 Results

Global Minimum Predictions

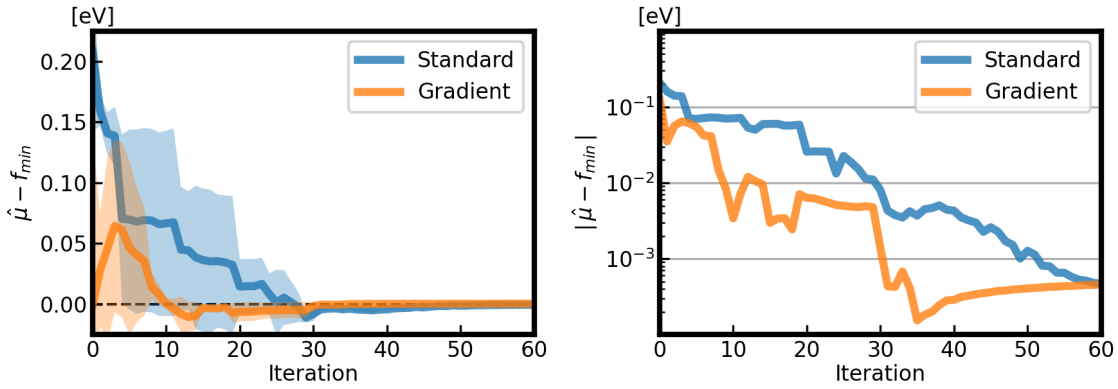


Fig. 18: Mean global minimum predictions $\hat{\mu}$ as a function of iterations, for benzene adsorption search. Both graphs contain the same data, but the graph on the right has a logarithmic y-scale. Both graphs are shifted by the known true global minimum f_{min}. The means are averaged across three runs for each BOSS variant (standard and gradient). The variance in predictions is depicted in the left graph as one standard deviation above and below the mean.

The global minimum predictions $\hat{\mu}$ for benzene adsorption search are depicted in fig. 18 above. The predictions are obtained through global optimization of the surrogate model. The results from the three runs of each BOSS variant (standard and gradient) are combined by averaging the predictions. Table 4 displays the mean number of iterations required $\bar{N}$ given a convergence criterion ε, for each BOSS variant. The primary convergence threshold of 1×10^{-2} eV is bolded.

Convergence

ε	$\bar{N}_s$	$\bar{N}_g$	$\bar{N}_g/\bar{N}_s$
10^{-1}	9.0	2.7	0.30
10^{-2}	**30.0**	**16.3**	**0.54**
10^{-3}	51.7	24.7	0.48

Table 4: Mean number of iterations required $\bar{N}$ for a given convergence tolerance ε. The number of iterations is averaged over three runs of each BOSS variant (standard or gradient). The primary convergence threshold is bolded.

Discussion

The results for the benzene adsorption search study suggest that the global minimum predictions are closer to the true global minimum when gradient observations are included. The mean number of iterations required in table 4 implies that, for the main convergence threshold $1 \times 10^{-2}\,\mathrm{eV}$, including gradient observations cut the iterations required for convergence by almost one half. The very low number of iterations required for a convergence threshold of $1 \times 10^{-1}\,\mathrm{eV}$, compared to other thresholds, could be explained by higher number of initial points used in the benzene adsorption search study.

4.3 Computational Cost Analysis

The results so far have indicated that including gradient observations in BOSS does reduce the number of iterations required for convergence. However, due to larger matrices, an iteration of gradient BOSS is more computationally expensive than an iteration of standard BOSS. In this section, I will investigate the actual time saved, or not saved, by including gradient observations.

One iteration of BOSS consists of four main operations that each take a different amount of time to perform. The time taken for each of these operations was recorded during the testing of the two material problems whose results have been presented so far. The four main operations whose durations compound into the total time for one iteration are:

- **Function evaluation**
  Evaluating the objective function, e.g., a simulation.
- **Surrogate model fitting**
  Fitting the surrogate model to the expanded data set. This includes optimizing the model to new hyperparameters (see eq. 15).
- **Global minimization of surrogate model**
  Optimizing the surrogate model in order to procure a prediction for the global minimum.
- **Locating next point of evaluation**
  Optimizing the acquisition function in order to determine the location of the function evaluation for the following iteration.

The function evaluation is independent of the BOSS process, and the time it takes depends on the type of simulation chosen. Furthermore, many atomic simulations include force—in addition to energy—calculations without any significant increase in computational cost. Therefore, any difference in time taken between standard BOSS and gradient BOSS to reach convergence is expected to be observed only in the latter three operations listed above. All three of these operations involve using the expanded covariance matrix $\mathbf{\Gamma}$ of eq. (20) in the gradient BOSS case, which means more computationally expensive matrix calculations. I will refer to the time taken to perform all these three operations as "**BOSS time**".

To examine timing differences between standard BOSS and gradient BOSS, I average the time taken per operation, for all BOSS runs of each variant. I then take the ratio $R_{g/s}$ of the time taken for gradient BOSS to perform the operation, compared to how long it takes standard BOSS to perform the operation. The ratios for alanine conformer search are illustrated in fig. 19 below:

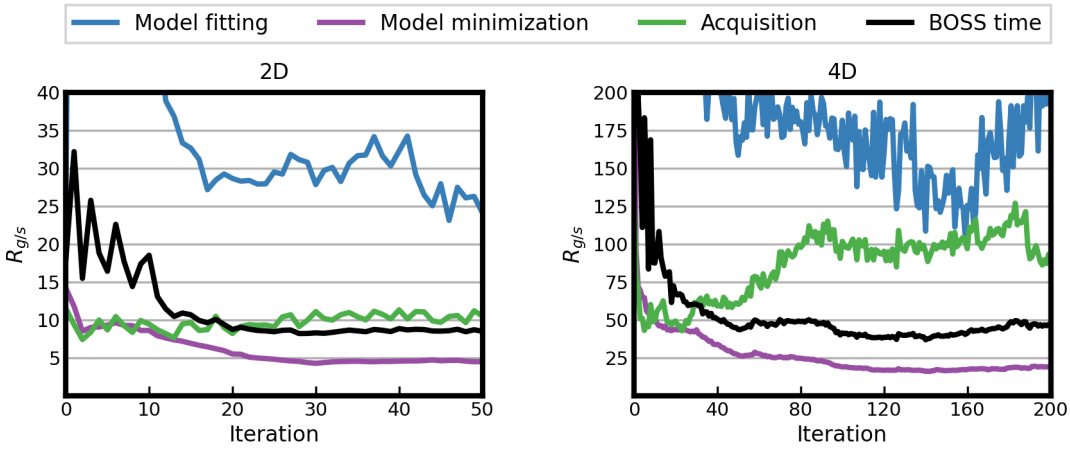


Fig. 19: Ratios of time taken to perform an operation, per iteration, for alanine conformer search (both 2D and 4D). The BOSS time is the sum of the duration of each operation.

The plotted ratios demonstrate how including gradient observations carries an increased computational cost. The duration taken by fitting the surrogate model with GPR is what is scaled by the largest factor. When the surrogate model is fit with a smaller amount of data (lower end of number of iterations), the time taken to fit the model can vary a lot, which can lead to a large ratio. With more data, the time taken for GPR fitting for gradient BOSS becomes more consistent. This variation in time taken by GPR fitting at the lower end of iterations can be seen in the ratio for total BOSS time. For 2D, the BOSS time tends towards being ~ 9 times slower when including gradient observations. For 4D, it tends towards being ~ 50 times slower.

By including gradient observations, the BOSS time per iteration is essentially increased by at least one order of magnitude. Whether gradient information will accelerate the BOSS process depends on the reduction in number of iterations required for convergence, and the time taken by the function evaluation. For very slow evaluations, the BOSS time per iteration will be a small fraction, so the factor increase will not be as significant.

To determine the evaluation time for which including gradient observations accelerates the process, I calculate the BOSS time taken until convergence for every run, and add a variable t_{eval} to every iteration required. By taking the ratio $R_{g/s}$ of time taken for convergence for gradient BOSS to standard BOSS, as a function of t_{eval}, the crossover evaluation time can be ascertained.

Alanine Conformer Search

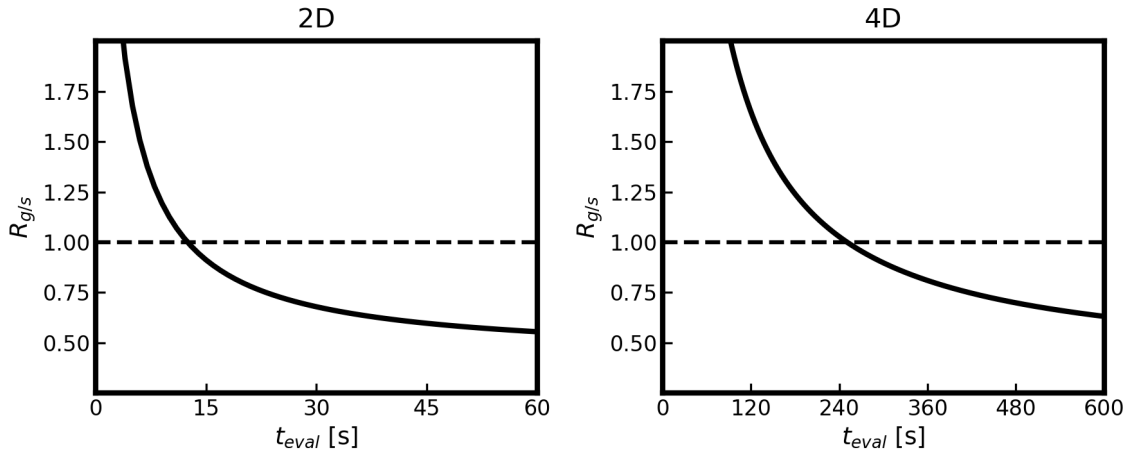


Fig. 20: Ratio of mean convergence time of gradient BOSS to standard BOSS $R_{g/s}$, as a function of evaluation time t_{eval}, for alanine conformer search. The convergence threshold is 1×10^{-1} kcal/mol. Gradient BOSS is more efficient than standard BOSS at t_{eval} when the ratio is below 1.

Graphed in fig. 20 above are the ratios $R_{g/s}$ of time required until convergence between gradient and standard BOSS, as a function of objective function evaluation time t_{eval}, for alanine conformer search. The convergence threshold is 1×10^{-1} kcal/mol. The crossover evaluation time is roughly ~ 13 seconds for 2D, and ~ 250 seconds for 4D. The energies and forces for the alanine conformer search studies were calculated using an emulator function based on AMBER code. The average time taken for an AMBER calculation of the properties of the alanine molecule is ~ 2 seconds [25]. This suggests that, for the alanine conformer search, including gradient observations does **not** improve efficiency.

Benzene Adsorption Search

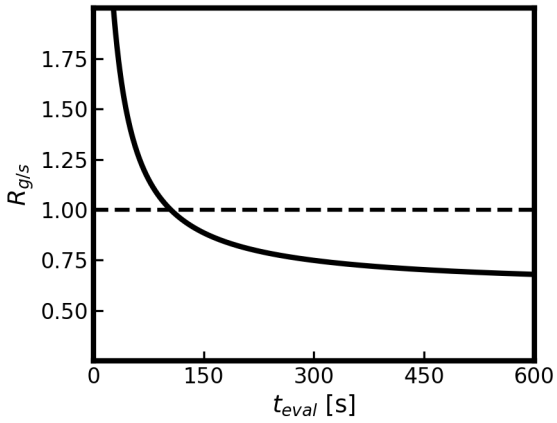


Fig. 21: Ratio of mean convergence time of gradient BOSS to standard BOSS $R_{g/s}$, as a function of evaluation time t_{eval}, for benzene adsorption search. The convergence threshold is 1×10^{-2} eV. Gradient BOSS becomes more efficient than standard BOSS at t_{eval} where the ratio passes 1.

For benzene adsorption search, the convergence threshold is 1×10^{-2} eV. The crossover time, where gradient BOSS becomes more efficient, is roughly $\sim$ 100 seconds. The energies and forces for the benzene adsorption search study were calculated using DFT simulations (described in section 2.3.4), with a mean function evaluation time of $\sim$ 755 seconds. This is well beyond the crossover time. Therefore, gradient BOSS is more efficient than standard BOSS in the benzene adsorption study. With a mean evaluation time of 755 seconds, standard BOSS converges, on average, in 7 h 22 min. Gradient BOSS, on the other hand, converges, on average, in 4 h 54 min, which means a time save of 2 h 28 min.

4.4 Discussion

From the results, two main points can be concluded:

- **Gradient observations reduce the number of iterations required for BOSS**
- **Including gradient observations substantially increases the BOSS time taken per iteration**

The number of iterations required for convergence being reduced is synonymous with requiring less data for global optimization. The alanine conformer search results suggest that the reduction in data required scales with dimensionality. Table 2 indicates that, by including gradient information, the 2D study required roughly 60 % fewer iterations for convergence, and the 4D study required roughly 66 % fewer iterations. However, the problem is that including gradient observations results in larger matrices, which leads to more expensive matrix calculations required by GPR. This increase in computational cost for matrix calculations scale with dimensions D and number of data points N as $\mathcal{O}(N^3D^3)$. Therefore, for some problems including gradient observations would impair the global optimization process, and for others it would be alleviated. For problems of very high dimensionality, including gradient observations would almost certainly make the process computationally prohibitive. Reducing the dimensionality of the problem as much as possible—with the building-block approach, for example—would greatly improve the efficiency gain provided by including gradient observations.

For the problems I studied, the deciding factors of whether gradient observations accelerated the process was the evaluation time of the computational simulations. For the alanine conformer search study a classical force field simulation was used, which has lower fidelity than the quantum mechanical DFT simulation used for the benzene adsorption study. The mean evaluation time of the force field simulation was a small fraction of the total time of one iteration when gradients were included. This means that the time per iteration was dominated by the cumbersome matrix calculations, which ultimately led to an inefficient global optimization process. For the benzene adsorption search study however, because of the quantum mechanical nature of the function evaluations, the time per iteration was dominated by the evaluation time itself. Now the matrix calculations of gradient observations were only a fraction of the total iteration time, allowing for a more efficient global optimization process.

The main issue with gradient observations is the poor scaling of $\mathcal{O}(N^3D^3)$ for matrix calculations. Further research in the acceleration of BOSS using gradient observations could be focused on reducing this detrimental scaling. Various strategies of addressing this issue have been proposed, such as the use of stochastic variational approximations [12], iterative solvers using fast matrix-vector multiplications together with pivoted Cholesky preconditioning [26], and Gram matrix decomposition [27]. The utilization of sparse matrices [28], or the parallel processing of graphical processing units (GPUs) [29], can increase the efficiency of matrix calculations in general.

5 Conclusion

BOSS facilitates structure search by reducing the amount of data required in order to find stable atomic structures, which accelerates the development of complex devices. Simulations that calculate forces in addition to energies yield data points with high information content, so utilizing these simulations to their fullest extent is of interest if one wanted to accelerate structure search further. In this research, I have implemented the functionality for including gradient observations in BOSS, and investigated the effect the functionality has on the process.

The results imply that the inclusion of gradient observations does reduce the amount of data required for the structure search process. However, the time taken per iteration of BOSS is increased significantly, with poor scaling for increasing dimensionality. This trade-off is something that will have to be taken into account when choosing to include gradient observations. The determining factor that decides whether observing gradients will increase efficiency is the evaluation time of the simulation. I suggest including gradient observations in BOSS when using quantum-mechanical simulations, such as DFT, for problems that allow for approximation by the building-block approach, thereby reducing dimensionality. For situations where the main concern is the amount of data available—not time taken to perform structure search—including gradient observations will likely be beneficial.

In conclusion, gradient observations in global structure search with Bayesian optimization presents an interesting avenue of research in materials science. In this thesis, I have presented the advantages and shortcomings of this functionality, and noted how such shortcomings might be alleviated. I hope that my implementation and analysis facilitates any future research into this topic.

Summary in Swedish - Svensk sammanfattning

Gradient observationer i global struktursökning med Bayesisk optimisering

Moderna apparater är ofta beroende av funktionaliteter som härstammar från de material som utgör apparaten. Ett materials funktionaliteter beror på dess makroskopiska egenskaper, t.ex elasticitet, elektrisk ledningsförmåga eller värmekapacitet. Källan till dessa makroskopiska egenskaper är materialets mikroskopiska atomkonfigurationer. Kunskap om ett materials atomära struktur krävs därför när man vill forma materialet för en specifik funktionalitet.

Det är viktigt att förstå hur materialet förväntas bete sig i sitt naturliga tillstånd, för att möjliggöra effektiv användning av materialet. Det naturliga tillståndet representeras av en viss atomkonfiguration. Att finna detta tillstånd är ett av huvudsyftena med struktursökning. Struktursökning kan utföras experimentellt, men det innebär utmaningar i vissa situationer. Att undersöka ytor av material experimentellt är relativt enkelt, medan nedgrävda gränssnitt i bulkmaterial kan vara svåråtkomliga. Experiment kan också sakna tillräcklig noggrannhet för en korrekt beskrivning av atomstrukturen.

En alternativ metod for struktursökning är användningen av datorsimuleringar. En simulering kan bygga en modell i atomär detalj, vars struktur kan justeras och sedan undersökas utan oro för experimentkostnaden. I många fall kommer en simulering även att spara tid jämfört med ett experiment. Olika typer av simuleringar kommer dock att medföra olika belopp av beräkningskostnader, beroende på modellens noggrannhet. Den kvantmekaniska modelleringsmetoden täthetsfunktionalteori ("density functional theory", DFT) beskriver väl de mikroskopiska interaktionerna som bestämmer atomstrukturer för många system [1]. DFT får sin noggrannhet från sin kvantmekaniska grund, men bär ändå en måttlig beräkningskostnad jämfört med andra kvantmekaniska metoder.

En DFT-simulering tar en atomkonfiguration och returnerar ett antal attribut som beskriver systemet. Ett av dessa attribut är systemets energi. Lägre energier indikerar ett system med högre stabilitet. Fasrummet för möjliga atomkonfigurationer kan kartläggas på en potentiell energiyta ("potential energy surface", PES). Struktursökning, i sammanhang med simuleringar, fokuserar på att utforska denna yta för att hitta atomära konfigurationer av intresse. På den potentiella energiytan kan det finnas flera lokala extremer, dvs. punkter där gradienten av ytan är noll. Minima på ytan indikerar stabila konfigurationer, och det minimum med det lägsta

energivärdet indikerar systemets mest stabila och naturliga tillstånd. Detta globala minimum hittas med global optimering.

Struktursökning använder global optimering för att hitta det globala minimumet, alltså det mest stabila tillståndet för systemet. Optimeringen sker i fasrummet för möjliga konfigurationer. Det finns ingen generell lösning för att hitta det globala minimumet direkt, så alla lokala minimum inom fasrummet måste bestämmas och jämföras. Ett kriterium för global optimering är då att fasrummet måste utforskas grundligt, så att inget lokalt minimum lämnas utanför jämförelsen. Detta medför många funktionsutvärderingar.

Det höga antalet funktionsutvärderingar som krävs utgör ett problem när struktursökning görs med DFT-simuleringar. Med DFT innebär en funktionsutvärdering en simulering och den resulterande datapunkten är ett energivärde som representerar en atomär konfiguration. Medan beräkningskostnaden för en individuell DFT-simulering är låg, blir den globala optimeringsprocessen beräkningsmässigt hindrande när många simuleringar körs sammantaget. Konventionella metoder för fasutforskning av rymd, såsom minimahoppning [4], Monte Carlo-metoder [5] eller metadynamik [6], kräver vanligtvis tusentals datapunkter [7].

Ett tillvägagångsätt för global optimering för struktursökning som minimerar antalet datapunkter som krävs är därför av intresse. Ett sådant tillvägagångssätt är bayesisk optimering struktursökning ("Bayesian Optimization Structure Search", BOSS) [1]. BOSS är en maskininlärningsmetod som underlättar struktursökning genom att välja konfigurationer för funktionsutvärderingar som påskyndar processen. BOSS bygger en surrogatmodell som lämpas till målfunktionen med funktionsutvärderingar. Med surrogatmodellen kan hela energiytan uppskattas. Modellen är stokatisk, vilket betyder att den returnerar ett medelvärde för sin uppskattning förutom en osäkerhet i sin förutsägelse. Osäkerheten minskar ju närmare uppskattningen är de datapunkter som modellen har lämpats enligt. Modellen kräver inte hög beräkningskostnad för att utvärdera, så konventionella globala optimiseringsmetoder kan tillämpas.

BOSS är en aktiv inlärningsprocess, dvs. datamängden byggs upp medan sökningen fortskrider. Surrogatmodellen uppdateras kontinuerligt genom att den anpassas till mer data som väljs av en s.k. ackvisitionfunktion ("acquisition function"). Modellen kan sökas efter en uppskattning av det globala minimumet varje gång modellen uppdateras. Så småningom har hela fasrummet utforskats, och uppskattningen för det globala minimumet kommer inte att variera per iteration. Då sägs det att processen har konvergerat, och det sanna globala minimumet och den stabila konfigurationen har blivit hittade.

DFT-simuleringar kan beräkna krafter som verkar på varje atom utöver konfigurationens energi. Krafterna är gradienten av energilandskapet, och ger information om hur landskapet förändras i närheten av datapunken. Att inkludera denna information i surrogatmodellen bör göra modellen mer korrekt i sina förutsägelser. En mer korrekt modell skulle kräva färre datapunkter för struktursökningsprocessen.

I detta arbete har jag implementerat denna gradientinformation i BOSS. För att avgöra giltigheten av förbättringen i BOSS som gradientinformationen tillför, har jag testat både normala och förbättrade processen på materiella problem. Jag beskriver teorin bakom implementeringen i detalj, dessutom resultaten från testen.

References

[1] Milica Todorović et al. "Bayesian Inference of Atomistic Structure in Functional Materials". In: *npj Computational Materials* 5.1 (1 Mar. 18, 2019), pp. 1–7. ISSN: 2057-3960. DOI: 10.1038/s41524-019-0175-2.

[2] J. Behler. "Representing Potential Energy Surfaces by High-Dimensional Neural Network Potentials". In: *Journal of Physics: Condensed Matter* 26.18 (Apr. 2014), p. 183001. ISSN: 0953-8984. DOI: 10.1088/0953-8984/26/18/183001.

[3] Aimo Törn and A. Zhilinskas. *Global Optimization*. Lecture Notes in Computer Science 350. Berlin ; New York: Springer-Verlag, 1989. 255 pp. ISBN: 978-3-540-50871-7 978-0-387-50871-9.

[4] Stefan Goedecker. "Minima Hopping: An Efficient Search Method for the Global Minimum of the Potential Energy Surface of Complex Molecular Systems". In: *The Journal of Chemical Physics* 120.21 (May 11, 2004), p. 9911. ISSN: 0021-9606. DOI: 10.1063/1.1724816.

[5] Fengyu Li et al. "Improved Stability of Water Clusters (H2O)30–48: A Monte Carlo Search Coupled with DFT Computations". In: *Theoretical Chemistry Accounts* 131.3 (Mar. 2, 2012), p. 1163. ISSN: 1432-2234. DOI: 10.1007/s00214-012-1163-5.

[6] Alessandro Laio and Michele Parrinello. "Escaping Free-Energy Minima". In: *Proceedings of the National Academy of Sciences* 99.20 (Oct. 2002), pp. 12562–12566. DOI: 10.1073/pnas.202427399.

[7] Jari Järvi, Patrick Rinke, and Milica Todorović. "Detecting Stable Adsorbates of (1S)-Camphor on Cu(111) with Bayesian Optimization". In: *Beilstein Journal of Nanotechnology* 11.1 (Oct. 19, 2020), pp. 1577–1589. ISSN: 2190-4286. DOI: 10.3762/bjnano.11.140.

[8] GPy. *GPy: A Gaussian process framework in python*. http://github.com/SheffieldML/GPy.

[9] Eero Siivola. *GPYgradients*. https://github.com/esiivola/GPYgradients.

[10] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. Cambridge, Mass: MIT Press, 2006. 248 pp. ISBN: 978-0-262-18253-9.

[11] Peter I. Frazier. *A Tutorial on Bayesian Optimization*. July 8, 2018. DOI: 10.48550/arXiv.1807.02811. arXiv: 1807.02811 [cs, math, stat].

[12] Misha Padidar et al. *Scaling Gaussian Processes with Derivative Information Using Variational Inference*. July 8, 2021. DOI: 10.48550/arXiv.2107.04061. arXiv: 2107.04061 [cs, stat].

[13] Donald R Jones, Matthias Schonlau, and William J Welch. "Efficient Global Optimization of Expensive Black-Box Functions". In: *Journal of Global Optimization* 13 (June 1998), pp. 455–492.

[14] Peter Frazier, Warren Powell, and Savas Dayanik. "The Knowledge-Gradient Policy for Correlated Normal Beliefs". In: *INFORMS Journal on Computing* 21.4 (Nov. 2009), pp. 599–613. ISSN: 1091-9856, 1526-5528. DOI: 10.1287/ijoc.1080.0314.

[15] Philipp Hennig and Christian J Schuler. "Entropy Search for Information-Efficient Global Optimization". In: *Journal of Machine Learning Research* 13 (June 2012), pp. 1809–1837.

[16] Michael U Gutmann and Jukka Corander. "Bayesian Optimization for Likelihood-Free Inference of Simulator-Based Statistical Models". In: *Journal of Machine Learning Research* 17 (Aug. 2016), pp. 1–47.

[17] IT Center for Science. https://www.csc.fi/en/home.

[18] AMBER. http://ambermd.org/.

[19] AMBER: GAFF. http://ambermd.org/antechamber/gaff.html.

[20] R. O. Jones. "Density Functional Theory: Its Origins, Rise to Prominence, and Future". In: *Reviews of Modern Physics* 87.3 (Aug. 25, 2015), pp. 897–923. DOI: 10.1103/RevModPhys.87.897.

[21] Volker Blum et al. *Ab initio molecular simulations with numeric atom-centered orbitals*. 2009. DOI: http://dx.doi.org/10.1016/j.cpc.2009.06.022.

[22] John P. Perdew, Kieron Burke, and Matthias Ernzerhof. "Generalized Gradient Approximation Made Simple". In: *Physical Review Letters* 77.18 (Oct. 28, 1996), pp. 3865–3868. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.77.3865.

[23] Alexandre Tkatchenko and Matthias Scheffler. "Accurate Molecular Van Der Waals Interactions from Ground-State Electron Density and Free-Atom Reference Data". In: *Physical Review Letters* 102.7 (Feb. 20, 2009), p. 073005. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.102.073005.

[24] Ask Hjorth Larsen et al. "The atomic simulation environment—a Python library for working with atoms". In: *Journal of Physics: Condensed Matter* 29.27 (2017), p. 273002.

[25] Manuel Kuchelmeister. “Multi-Fidelity Bayesian Machine Learning for Global Optimization”. MA thesis. 2022. ISBN: 9781817731059. DOI: 10.18419/opus-12396.

[26] David Eriksson et al. *Scaling Gaussian Process Regression with Derivatives*. Oct. 29, 2018. DOI: 10.48550/arXiv.1810.12283. arXiv: 1810.12283 [cs, stat].

[27] Filip de Roos, Alexandra Gessner, and Philipp Hennig. “High-Dimensional Gaussian Process Inference with Derivatives”. In: *Proceedings of the 38th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, July 1, 2021, pp. 2535–2545.

[28] Raphael Yuster. “Fast Sparse Matrix Multiplication”. In: *ACM Transactions on Algorithms* 1 (1 July 2005), pp. 2–13.

[29] K. Fatahalian, J. Sugerman, and P. Hanrahan. *Understanding the Efficiency of GPU Algorithms for Matrix-Matrix Multiplication*. 2004.