

## Washington University in St. Louis Washington University Open Scholarship

---

Engineering and Applied Science Theses &  
Dissertations

McKelvey School of Engineering

---


Winter 12-2015

# Applying Bayesian Machine Learning Methods to Theoretical Surface Science

Shane Carr

*Washington University in St. Louis*

Follow this and additional works at: [https://openscholarship.wustl.edu/eng\\_etds](https://openscholarship.wustl.edu/eng_etds)

 Part of the [Applied Statistics Commons](#), [Artificial Intelligence and Robotics Commons](#), [Catalysis and Reaction Engineering Commons](#), [Computational Engineering Commons](#), and the [Materials Chemistry Commons](#)

---

### Recommended Citation

Carr, Shane, "Applying Bayesian Machine Learning Methods to Theoretical Surface Science" (2015). *Engineering and Applied Science Theses & Dissertations*. 122.

[https://openscholarship.wustl.edu/eng\\_etds/122](https://openscholarship.wustl.edu/eng_etds/122)

This Thesis is brought to you for free and open access by the McKelvey School of Engineering at Washington University Open Scholarship. It has been accepted for inclusion in Engineering and Applied Science Theses & Dissertations by an authorized administrator of Washington University Open Scholarship. For more information, please contact [digital@wumail.wustl.edu](mailto:digital@wumail.wustl.edu).

Washington University in St. Louis  
School of Engineering and Applied Science  
Department of Computer Science and Engineering

Thesis Examination Committee:  
Cynthia Lo  
Roman Garnett  
Michael Brent

Applying Bayesian Machine Learning Methods to Theoretical Surface Science

by

Shane Frederic F. Carr

A thesis presented to the Graduate School of Arts and Sciences  
of Washington University in partial fulfillment of the  
requirements for the degree of

Master of Science

December 2015  
St. Louis, Missouri

copyright by  
Shane Frederic F. Carr  
2015

# Contents

List of Figures . . . . .	iv
Acknowledgments . . . . .	v
Abstract . . . . .	vi
Preface . . . . .	vii
<b>1 Introduction . . . . .</b>	<b>1</b>
1.1 The Problem . . . . .	1
1.2 Parameterizing the Objective Function . . . . .	3
1.2.1 Six Basic Parameters . . . . .	3
1.2.2 Additional Parameters . . . . .	4
<b>2 Estimating Potential Energies . . . . .</b>	<b>7</b>
2.1 Background: Density Functional Theory . . . . .	7
2.1.1 Numerical Methods for DFT . . . . .	8
2.2 Performing Potential Energy Calculations . . . . .	9
2.2.1 Study on SCF Convergence . . . . .	10
2.2.2 Predicting Potential Energies using Bayes' Rule . . . . .	10
2.2.3 Choosing the Number of Iterations . . . . .	14
2.3 Choosing Mode and Basis Set . . . . .	16
<b>3 Searching for Active Site . . . . .</b>	<b>18</b>
3.1 Background: Bayesian Optimization . . . . .	18
3.1.1 The Algorithm . . . . .	19
3.1.2 Kernels . . . . .	20
3.2 Modeling the Objective Function . . . . .	21
3.2.1 Setting the Hyperparameters . . . . .	22
3.2.2 Handling Periodic Variables . . . . .	25
3.3 Bayesian Optimization Demo . . . . .	27
<b>4 Results and Discussion . . . . .</b>	<b>29</b>
4.1 Background: State of the Art . . . . .	29
4.1.1 Constrained Minima Hopping . . . . .	29

4.1.2	Differential Evolution . . . . .	31
4.2	Results . . . . .	31
4.2.1	Runtime Comparison . . . . .	32
4.3	Discussion . . . . .	34
4.3.1	Hyperparameters: BO versus DE . . . . .	34
4.3.2	SCF Loop and Constrained Minima Hopping . . . . .	34
4.3.3	Parallelization . . . . .	35
<b>5</b>	<b>Conclusions . . . . .</b>	<b>37</b>
5.1	Future Work . . . . .	37
<b>Appendix A: Details on Experimental Setup . . . . .</b>		<b>39</b>
A.1	Surface Preparation . . . . .	39
A.2	Parameter Space and Objective Function . . . . .	40
A.3	Bayesian Optimization . . . . .	40
<b>Appendix B: Hyperparameter Optimization . . . . .</b>		<b>41</b>
B.1	Maximizing Likelihood . . . . .	41
B.2	Considerations with Automatic Optimization . . . . .	42
<b>References . . . . .</b>		<b>43</b>
<b>Vita . . . . .</b>		<b>47</b>

# List of Figures

1.1	Project flowchart . . . . .	2
1.2	Euler angles . . . . .	4
1.3	Example atomic configuration . . . . .	5
2.1	SCF potential energy traces . . . . .	10
2.2	Distributions over $\Delta_1 \equiv E_\infty - O_1$ . . . . .	11
2.3	Predictions of $E_\infty$ by iteratively applying Bayes' Rule . . . . .	13
2.4	Standard deviations of $\Delta_i$ over each iteration . . . . .	15
2.5	Comparison of initial basis energies versus converged energies . . . . .	15
2.6	Comparison of potential energy measures . . . . .	17
3.1	A Gaussian process . . . . .	19
3.2	Length scales for CO on Fe <sub>2</sub> O <sub>3</sub> . . . . .	23
3.3	Periodic unit cells . . . . .	25
3.4	Four methods for enforcing periodicity . . . . .	26
3.5	Runtime comparison of periodic kernels . . . . .	27
3.6	Demonstration of Bayesian Optimization . . . . .	28
4.1	Differential Evolution hyperparameter settings . . . . .	31
4.2	The global minimum for CO on Fe <sub>2</sub> O <sub>3</sub> . . . . .	33
4.3	Runtime of my routine and state-of-the-art routines . . . . .	33
4.4	SCF force traces . . . . .	35
A.1	Bayesian Optimization hyperparameter settings . . . . .	40
B.1	GPy optimized length scales . . . . .	42

# Acknowledgments

I would like to thank my research advisors, Dr. Cynthia Lo and Dr. Roman Garnett. Dr. Lo has been a mentor for me since I first joined her catalysis research group in my sophomore year. Dr. Garnett came along more recently, but he has taught me a great deal in our eight months of working together.

I also really want to thank all of the talented students in my research group, especially Alireza, Ahmed, Tola, Eddie, Wei, and Stephen. You have helped me make this thesis better every step of the way. I hope that my work proves helpful to your research.

This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1053575 [1].

This work used several Python libraries, including GPy for Gaussian processes [2], GPAW for density functional theory calculations [3, 4, 5], ASE for atomic manipulations [6], and SciPy for numerical routines [7]. Much of my work used the IPython graphical user interface to Python [8]. Most graphs were generated using the Python interface to Matplotlib [9].

Shane Frederic F. Carr

*Washington University in St. Louis*  
*December 2015*

## ABSTRACT OF THE THESIS

Applying Bayesian Machine Learning Methods to Theoretical Surface Science

by

Shane Frederic F. Carr

Master of Science in Computer Science

Washington University in St. Louis, December 2015

Research Advisors: Dr. Roman Garnett and Dr. Cynthia Lo

Machine learning is a rapidly evolving field in computer science with increasingly many applications to other domains. In this thesis, I present a Bayesian machine learning approach to solving a problem in theoretical surface science: calculating the preferred active site on a catalyst surface for a given adsorbate molecule. I formulate the problem as a low-dimensional objective function. I show how the objective function can be approximated into a certain confidence interval using just one iteration of the self-consistent field (SCF) loop in density functional theory (DFT). I then use Bayesian optimization to perform a global search for the solution. My approach outperforms the current state-of-the-art method, constrained minima hopping, for CO on ferric oxide by a factor of 75 to 1. This thesis is the first documented application of Bayesian optimization to surface science.

*Keywords:* Adsorption; Bayesian optimization; Catalysis; Density functional theory; Gaussian processes; Periodic kernels.



# Preface

When I first entered WashU as a freshman in 2011, I was interested in many different areas: computer science, chemistry, engineering, nanotechnology. Through my four years as an undergraduate, I ended up studying all of those fields, and here I am now, finishing my MS in Computer Science.

This thesis is an interdisciplinary project applying techniques in computer science to a problem in chemical engineering. To the computer scientist, this thesis is a real-life application of an art largely confined to theoretical literature. To the chemical engineer, this thesis is a novel method to solve a practical problem.

I assume that the reader has a basic understanding of computer algorithms, statistics, chemistry, and materials science. I try my best to explain domain-specific jargon whenever I use it for the first time.

*Shane Frederic F. Carr*  
*December 2015*

# Chapter 1

## Introduction

Surfaces are everywhere. The chair you are sitting on is a surface. The paper or screen you're reading off right now is a surface. The film formed between olive oil and balsamic vinegar in salad dressing is a surface.

*Surface science* is the study of surfaces like these. It considers their mechanical and chemical properties, how they form, and how they interact with the outside environment. In *theoretical surface science*, we answer these questions using models based on quantum mechanics and Newtonian physics.

This chapter does two things. First, it introduces the problem in theoretical surface science that this thesis will show how to solve. Second, it demonstrates how the problem can be formulated as a low-dimensional objective function. The low-dimensional objective function is the basis for Chapter 2, which shows how we can evaluate the function, and Chapter 3, which shows how we can minimize it.

### 1.1 The Problem

The field of surface science is closely related to *catalysis*, the study of the kinetics of chemical reactions. Breakthroughs in catalysis have given rise to more efficient engines, lower-cost chemicals, and carbon sequestration technologies.

When a molecule in a chemical reaction interacts with a catalyst, it does so on the surface of the catalyst. The process of a molecule binding to a surface is called *adsorption*.

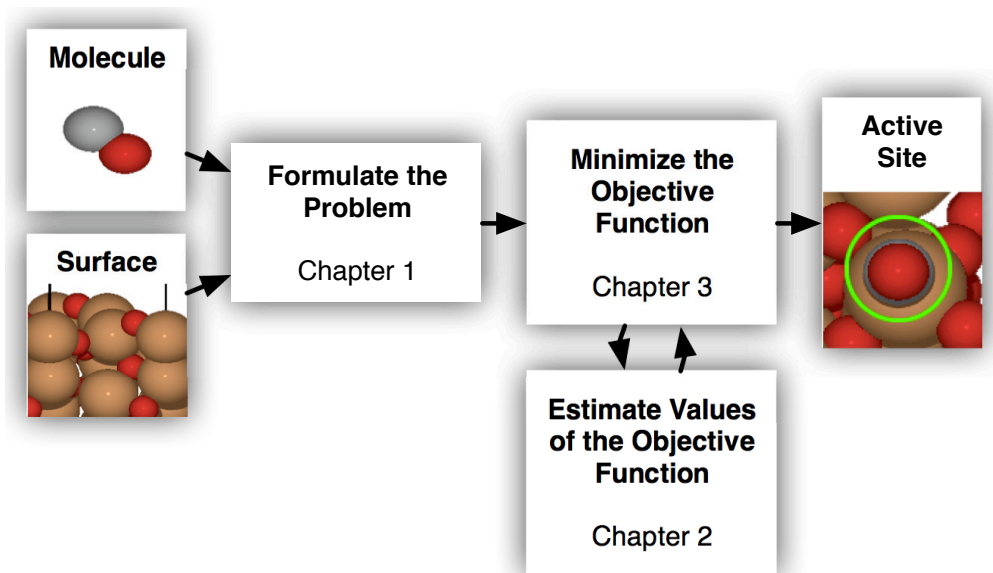


Figure 1.1: Flowchart illustrating the relationship between the different parts of this thesis.

A molecule could potentially adsorb at any given point on the surface. However, some areas of the surface are better and more stable than others. We call these areas the *active sites*. The laws of thermodynamics tell us that the best active site has the lowest *potential energy*.

In order to characterize a catalyst, we are often interested in calculating the *adsorption energy* of a molecule on the catalyst's surface. The adsorption energy is the difference between the potential energy of the molecule-surface system (with the molecule bound to the active site) and the potential energies of the molecule and surface by themselves.

This gives rise to an important question in surface science. **Given a surface and a molecule, what is the optimal active site for the molecule on the surface?**

In theoretical surface science, we can look at this as an optimization problem. **What configuration of a molecule on a surface gives rise to the lowest potential energy?**

In this thesis, I present a novel approach to solving this question, one that uses methods from Bayesian machine learning. This chapter formulates the problem as a low-dimensional objective function. In Chapter 2, I show how we can make cheap estimates of the objective function. In Chapter 3, I explain Bayesian optimization and show how we can use it to minimize the objective function. Chapter 4 shows how my solution stacks up against

the state-of-the-art solutions. Finally, in Chapter 5, I give some conclusions and recommendations for future work. Figure 1.1 illustrates the connection between these different chapters.

## 1.2 Parameterizing the Objective Function

Classical methods for minimizing the potential energy of a chemical system involve many dimensions: in particular, the three spatial coordinates of each atom. However, as an adsorbate molecule becomes larger, so does the number of dimensions in the optimization problem. Even a relatively small molecule, like dimethyl ether ( $\text{CH}_3\text{OCH}_3$ ), would have 27 dimensions.

In order to maintain the identity of the molecule, some methods, such as constrained minima hopping, add *constraints* to the minimization problem [10]. However, instead of adding constraints, we can preserve the molecular identity by reducing the dimensionality. The only requirement is that we should be able to closely approximate any possible molecule-surface configuration.

### 1.2.1 Six Basic Parameters

First, we need to specify the location of the center of the molecule along the surface plane. We can do this using two parameters,  $x$  and  $y$ . These two parameters are periodic across the boundaries of the unit cell.<sup>1</sup> I always take  $x$  and  $y$  to be fractional coordinates in the given direction, such that  $x, y \in [0, 1)$ .

Second, we need to specify how “close” the molecule is to the surface. We can do this with another parameter, which I will call  $z$ .

Finally, we need to specify the orientation of the molecule. The orientation of a rigid body in space is fully specified by three parameters,  $\phi$ ,  $\theta$ , and  $\psi$ , commonly known as the *Euler angles* [11]. I always take  $\phi$ ,  $\theta$ , and  $\psi$  to be measured in radians.  $\phi$  and  $\psi$  are periodic with

---

<sup>1</sup>The term *unit cell* refers to a section of the surface, usually shaped like a parallelogram, that infinitely repeats itself to form the entire surface.

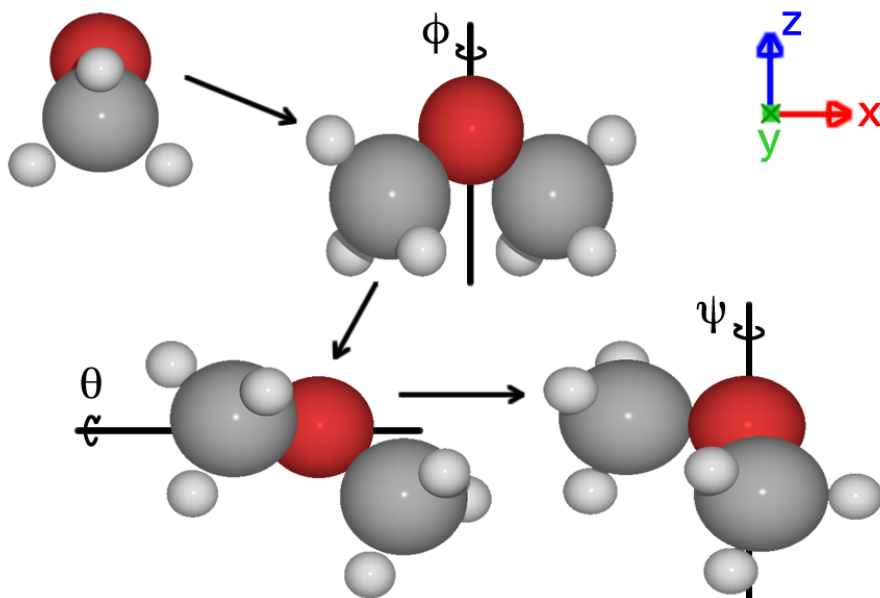


Figure 1.2: The three Euler angles fully specify the orientation of a body in space. This image shows the process of rotating a dimethyl ether molecule by  $\pi/3$  radians about each axis.

period  $2\pi$ .  $\theta$  is also periodic, with period  $\pi$ , but it alternates with mirror images of itself. See Figure 1.2 for an illustration of the Euler angles.

Molecules that are symmetric about an axis, like CO and CO<sub>2</sub>, need only two Euler coordinates rather than three. Single-atom adsorbates do not need any Euler coordinates.

We can therefore approximate the configuration of a molecule on a catalyst surface with a total of six parameters, and sometimes fewer. An example configuration is shown in Figure 1.3.

## 1.2.2 Additional Parameters

The six parameters of Section 1.2.1 neglect structural changes of the molecule and of the surface. I consider two types of structural changes: *free structural parameters* and *deformation by adsorption*.

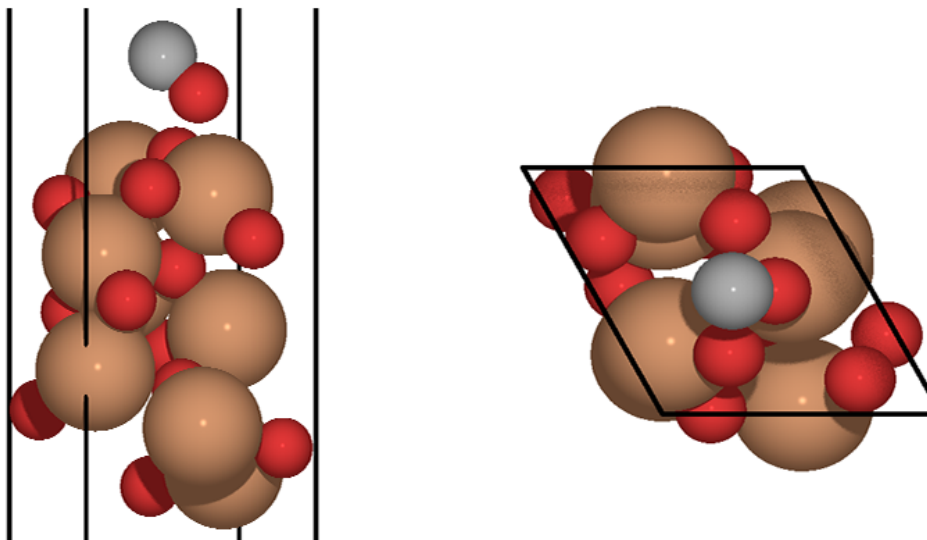


Figure 1.3: An example configuration of carbon monoxide (CO) on the 001 surface of ferric oxide ( $\text{Fe}_2\text{O}_3$ ). The red atoms are oxygen; brown is iron; and gray is carbon. The C atom in CO is positioned at  $\langle x, y \rangle = \langle 0.5, 0.5 \rangle$ , and it is  $z = 2.2\text{\AA}$  above the surface. The molecule is rotated with Euler coordinates  $\langle \theta, \psi \rangle = \langle \frac{\pi}{4}, \frac{3\pi}{2} \rangle$ . The symmetry of the CO molecule nullifies the Euler coordinate  $\phi$ . The surface is periodic across the boundaries of the unit cell, shown in black. For more detail, refer to Appendix A.

### Free Structural Parameters

Consider a molecule like ethane,  $\text{C}_2\text{H}_6$ . In addition to the three Euler angles, ethane needs one more parameter to fully define its orientation: the internal angle of rotation about the C–C bond. This parameter is periodic, and it could be added to our objective function. For this thesis, I did not study any molecules with free structural parameters.

### Deformation by Adsorption

It is usually the case that the the interaction of the molecule with the surface causes one or the other to deform in a way that they would not normally deform in free space. For example, in carbon monoxide, the length of the C–O bond may increase or decrease, and the binding of CO to the catalyst may cause the atoms on the surface to move.

These changes are difficult to parameterize. Depending on the molecule, some number of parameters could potentially be added to correspond to common ways that the molecule could deform. The length of the C–O bond would be one. The number of parameters is going to increase dramatically, though, for more complex molecules.

Fortunately, my results indicate that deformation by adsorption can be neglected for the optimization problem at hand, at least for a CO molecule on an Fe<sub>2</sub>O<sub>3</sub> surface. If desired, the user may perform an additional relaxation step after my algorithm returns an active site.<sup>2</sup> It is unclear whether or not this simplification translates over to other molecules on other surfaces.

An interesting area for further research would be coming up with sensible parameters to represent deformation by adsorption.

---

<sup>2</sup>A *relaxation step* means converging the full-dimensional system (three coordinates for each atom) to a local potential energy minimum, typically using a classical optimization algorithm like L-BFGS [12].

# Chapter 2

## Estimating Potential Energies

Chapter 1 introduced the problem and showed how we can formulate it as a low-dimensional objective function. The goal of this chapter is to demonstrate how we can make efficient estimates of the objective function.

Section 2.1 provides theoretical background on computational chemistry, and in particular on density functional theory (DFT). Section 2.2 shows how we can improve the performance of DFT given our well-defined parameters. Finally, Section 2.3 compares the choices we have for estimating the potential energy.

Throughout this and the next chapter, I will be studying a CO adsorbate on an  $\text{Fe}_2\text{O}_3$  surface (001 plane). Details of how I set up my calculations can be found in Appendix A.

### 2.1 Background: Density Functional Theory

*Computational chemistry* is the idea of simulating and computing properties of systems at the nanoscale using quantum mechanics and Newtonian physics. Computational chemistry is applied in many fields of study, from chemical engineering to biomedicine to climatology. It is the basis behind simulations of transport through carbon nanotubes, protein folding, and anthropogenic ozone depletion. [13]

At its core, all of computational chemistry revolves around Schrödinger's equation:



$$E\Psi = \hat{H}\Psi \tag{2.1}$$

Solving Equation 2.1 outright is not feasible, so in practice, we make assumptions and use models to aid in the computation.

*Density functional theory* (DFT) is one such model, based around the notion that potential and kinetic energies can be computed as a function of the electronic density [13]. The governing equation of DFT is [14]:

$$E[\rho(\vec{r})] = T_S[\rho(\vec{r})] + J[\rho(\vec{r})] + E_{XC}[\rho(\vec{r})] + \int \rho(\vec{r})V_{ext}d\vec{r} \tag{2.2}$$

where  $T_S$  is the kinetic energy,  $J$  is the classical Coulombic energy,  $E_{XC}$  is the *exchange and correlation energy*, and  $V_{ext}$  is the external potential. All of the terms in the governing equation are a function of the electronic density  $\rho$ , which in turn is a function of the electron coordinates  $\vec{r}$ ; thus, we have “density functional” theory.

DFT is a popular method used by researchers, with over 18,000 publications in 2015 making use of the method, according to Web of Science [15]. DFT’s popularity largely stems from its ability to handle systems with complex electronic interactions, such as semiconductors.

### 2.1.1 Numerical Methods for DFT

Thanks to much work in this field over the last two decades, researchers seldom need to implement Equation 2.2 on their own. Tools such as GPAW [3, 4, 5] give the user a much higher level of abstraction.

Internally, GPAW solves the DFT governing equation using numerical methods. It starts by making an educated initial guess, and then it runs an iterative procedure to converge the parameters to their true values. This iterative procedure is commonly known as the *self-consistent field* (SCF) loop. Performing an SCF iteration is computationally expensive, with classical techniques scaling as  $O(N^3)$  with the number of atoms and electrons [16].

## GPAW Modes and Basis Sets

There are several methods, or “modes,” that are used for computing the values of the wave function through space. GPAW gives the user three choices:

1. Finite-Difference (FD)
2. Linear Combination of Atomic Orbitals (LCAO)
3. Plane-Waves (PW)

In addition, GPAW gives the user several methods for coming up with the initial guess, or “basis set.” I consider two of them:

1. Pseudo partial waves (PPW)
2. Double-zeta potential (DZP)

I compare these different modes and basis sets in Section 2.3.

## 2.2 Performing Potential Energy Calculations

Given a list of nuclear coordinates from our objective function (see Section 1.2), we could just use a library like GPAW to get the potential energy, and call it a day. However, as mentioned in Section 2.1.1, performing the DFT calculations is expensive, so we’d like to do something more clever.

Between all of the different potential energy calculations we need to perform, the changes to the system are small. The five or six atoms in the adsorbate molecule will move, but all the atoms in the catalyst surface remain the same. **Can we take advantage of the similarity between calculations to improve on DFT’s performance?**

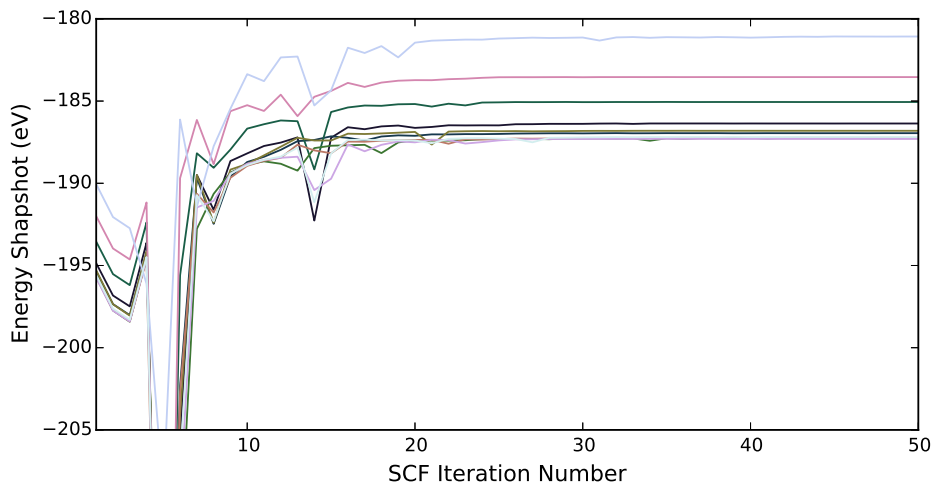


Figure 2.1: Values of the potential energy as GPAW runs an SCF loop for several different configurations, using finite difference (FD) mode with pseudo partial wave (PPW, default) basis set. At iteration 5, the traces all “dip” to approximately -240 eV.

### 2.2.1 Study on SCF Convergence

As explained in Section 2.1.1, libraries like GPAW typically use an iterative SCF loop to numerically converge the governing equation and obtain the potential energy of the system.

Suppose that, at every step of iteration, we wrote down what the potential energy of the system would be if we were to stop the iteration right there. As we let the number of iterations approach infinity, we get a trace that looks like Figure 2.1. It turns out that for most configurations of the molecule on the surface, the potential energy values follow a shared pathway through the convergence process.

We can take advantage of this fact to produce early estimates of the final converged potential energy values. I use a Bayesian approach, explained below.

### 2.2.2 Predicting Potential Energies using Bayes’ Rule

Let  $E_\infty$  be a random variable representing the final potential energy if we were to let SCF run all the way to convergence. Let  $O_i$  (for “observation”) be a value of the potential energy

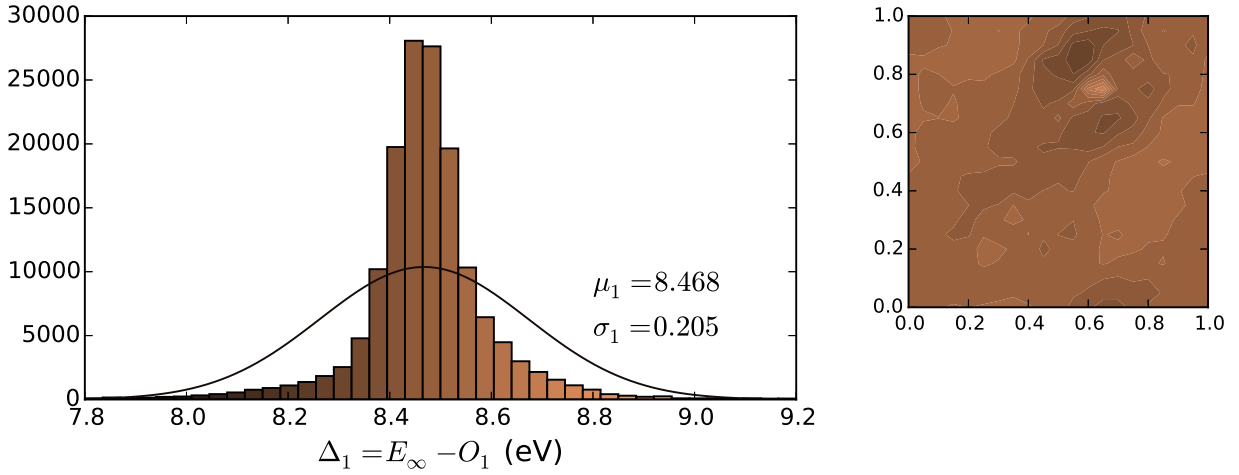


Figure 2.2: *Left*: Histogram of  $\Delta_1$  values for the CO on  $\text{Fe}_2\text{O}_3$  system, based on 154,631 full traces corresponding to a grid of parameters ranging over  $1.5\text{\AA} \leq z \leq 2.5\text{\AA}$  and all values of the periodic variables. All traces shown converged to within  $10^{-3}$  eV. *Right*: Mean of all  $\Delta_1$  values projected into the  $xy$ -plane, using the color scale defined in the histogram.

trace corresponding to iteration  $i$ . Let  $D$  (for “data”) be a matrix containing several full traces for different molecule configurations. Using Bayes’ Rule, we can say that:

$$Pr(E_\infty|O_i, D) \propto Pr(O_i|E_\infty, D)Pr(E_\infty|D) \quad (2.3)$$

We need to come up with a form for the likelihood term,  $Pr(O_i|E_\infty, D)$ . I define  $\Delta_i$  to be a random variable corresponding to the difference between an observation at iteration  $i$  and  $E_\infty$ . That is:

$$\Delta_i \equiv E_\infty - O_i \quad (2.4)$$

Figure 2.2 shows a plot of  $\Delta_1$  values for CO on  $\text{Fe}_2\text{O}_3$ . The other  $\Delta_i$ ’s have a similar behavior. We can therefore, with training data in hand, fit a Gaussian distribution over each  $\Delta_i$ .<sup>3</sup> Symbolically, we represent the Gaussian distribution as:

<sup>3</sup>The standard deviations of the  $\Delta_i$  values for  $1 \leq i \leq 30$  are shown in Figure 2.4.

$$Pr(\Delta_i|D) = \mathcal{N}(\Delta_i; \mu_i, \sigma_i^2) \quad (2.5)$$

where  $\mathcal{N}$  is the normal probability density function (normal PDF). The likelihood for Equation 2.3 can then take the form:

$$Pr(O_i|E_\infty, D) = E_\infty - \mathcal{N}(O_i; \mu_i, \sigma_i^2) \quad (2.6)$$

$$= \mathcal{N}(O_i; E_\infty - \mu_i, \sigma_i^2) \quad (2.7)$$

$$= \mathcal{N}(E_\infty; O_i + \mu_i, \sigma_i^2) \quad (2.8)$$

This solution isn't foolproof because there is a correlation between the  $\Delta_i$ 's and particular parameter values, illustrated in Figure 2.2. However, the correlation is weak enough that I have chosen to neglect it for this thesis. Studying the  $\Delta_i$ 's *bias function* would be a good topic for further research.

I choose the prior probability in Equation 2.3 to be the *conjugate prior* of our likelihood, fitted over the training data. In this case, we simply need another Gaussian distribution:

$$Pr(E_\infty|D) = \mathcal{N}(\mu_0, \sigma_0^2) \quad (2.9)$$

We can now express the posterior distribution  $Pr(E_\infty|O_i, D)$  in closed form [17]:

$$Pr(E_\infty|O_i, D) = \mathcal{N}(\mu_p, \sigma_p^2) \quad (2.10)$$

$$\mu_p = \left( \frac{\mu_0}{\sigma_0^2} + \frac{O_i + \mu_i}{\sigma_i^2} \right) / \left( \frac{1}{\sigma_0^2} + \frac{1}{\sigma_i^2} \right) \quad (2.11)$$

$$\sigma_p^2 = 1 / \left( \frac{1}{\sigma_0^2} + \frac{1}{\sigma_i^2} \right) \quad (2.12)$$

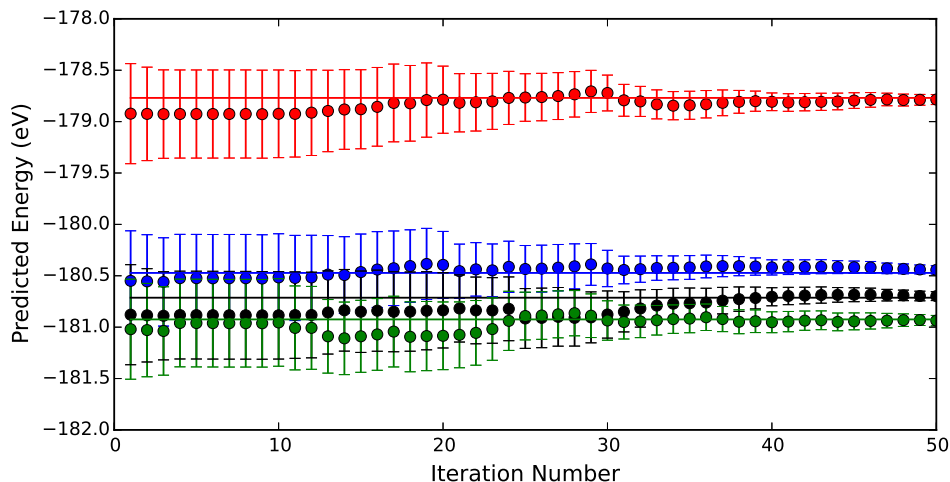


Figure 2.3: Illustration of how the confidence interval over  $E_\infty$  converges as we collect more observations from the trace. Each color corresponds to a different configuration of CO on  $\text{Fe}_2\text{O}_3$ . The error bars correspond to two standard deviations in each direction.<sup>4</sup>

Observe that a smaller  $\sigma_i$  results in a smaller  $\sigma_p$  and causes  $O_i + \mu_i$  to have a greater influence on the value of  $\mu_p$ . This corresponds to the idea that a confident measurement (low  $\sigma_i$ ) should give us more information and result in a confident prediction.

We can take this one step further. By taking the naive Bayes assumption that all observations are independent from one another, we can iteratively update our posterior belief on  $E_\infty$  whenever we get a new SCF observation: the  $\mu_p$  and  $\sigma_p^2$  terms from the previous observation become the  $\mu_0$  and  $\sigma_0^2$  terms for the new observation. If desired, we can continue collecting SCF observations until our confidence interval is sufficiently small. Figure 2.3 shows an example with error bars collapsing as we add more observations from the SCF trace.

In practice, we often encounter outliers in the SCF traces. In order to prevent outliers from biasing  $E_\infty$ , we want to ensure that the variance for the likelihood,  $\sigma_i^2$ , is sufficiently large. I do this by adding two additional terms to the likelihood variance:

$$\hat{\sigma}_i^2 = \sigma_i^2 + \sigma_n^2 + \sigma_{h,i}^2 \quad (2.13)$$

---

<sup>4</sup>The likelihood was trained on a set of four complete traces, which are different from the ones in this figure. The solid horizontal lines correspond to the  $E_\infty$  for each configuration. A constant noise of 0.05 eV was added to the variance of the likelihood functions, in addition to the heuristic noise discussed in the text.

$\sigma_n^2$  is a constant noise term, which I have typically set by hand.  $\sigma_{h,i}^2$  is a heuristic noise term defined as follows:

$$\sigma_{h,i}^2 \equiv \frac{h}{4} \sum_{j=i-3}^i |O_i - O_j| \quad (2.14)$$

where  $h$  is an additional hyperparameter controlling the amount of noise that is added to the likelihood from this term; I set  $h = \frac{1}{2}$ . If  $i \leq 3$ , we only sum over the available observations, and adjust the constant 4 accordingly.

The idea behind this expression in Equation 2.14 is that when we encounter an outlier, we expect the outlier to be very different from the observations that came before the outlier. We therefore take the average difference between the new observation and several previous observations, and multiply it by our hyperparameter  $h$  before adding it to the likelihood noise.

### 2.2.3 Choosing the Number of Iterations

Figure 2.3 showed how we can iteratively apply observations to converge our confidence intervals. However, observe that the error bars do not significantly shrink between iterations 1 and 20.

Figure 2.4 shows us why. Recall that the amount of information we gain from a particular observation is inversely proportional to  $\sigma_i$ , or  $\Delta_i$ 's standard deviation.  $\sigma_i$  is low at the first iteration ( $\sigma_1 = 0.205$ ), and the next iteration that beats the first iteration isn't until iteration 19 ( $\sigma_{19} = 0.154$ ). This raises a question: is it sufficient to use just a single iteration?

Figure 2.5 shows several slices out of the objective function for CO on  $\text{Fe}_2\text{O}_3$ . The top row shows the initial potential energy measurements transformed via Equation 2.10, and the bottom row shows the converged potential energy measurements. We can see right away that the top row models the bottom row very well.

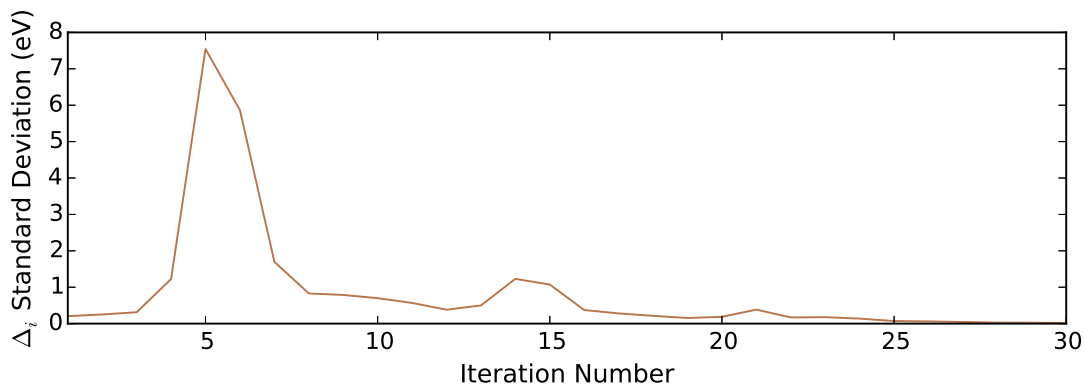


Figure 2.4: Standard deviations of the  $\Delta_i$  distributions for each iteration from 1 to 30, based on the same data as Figure 2.2.

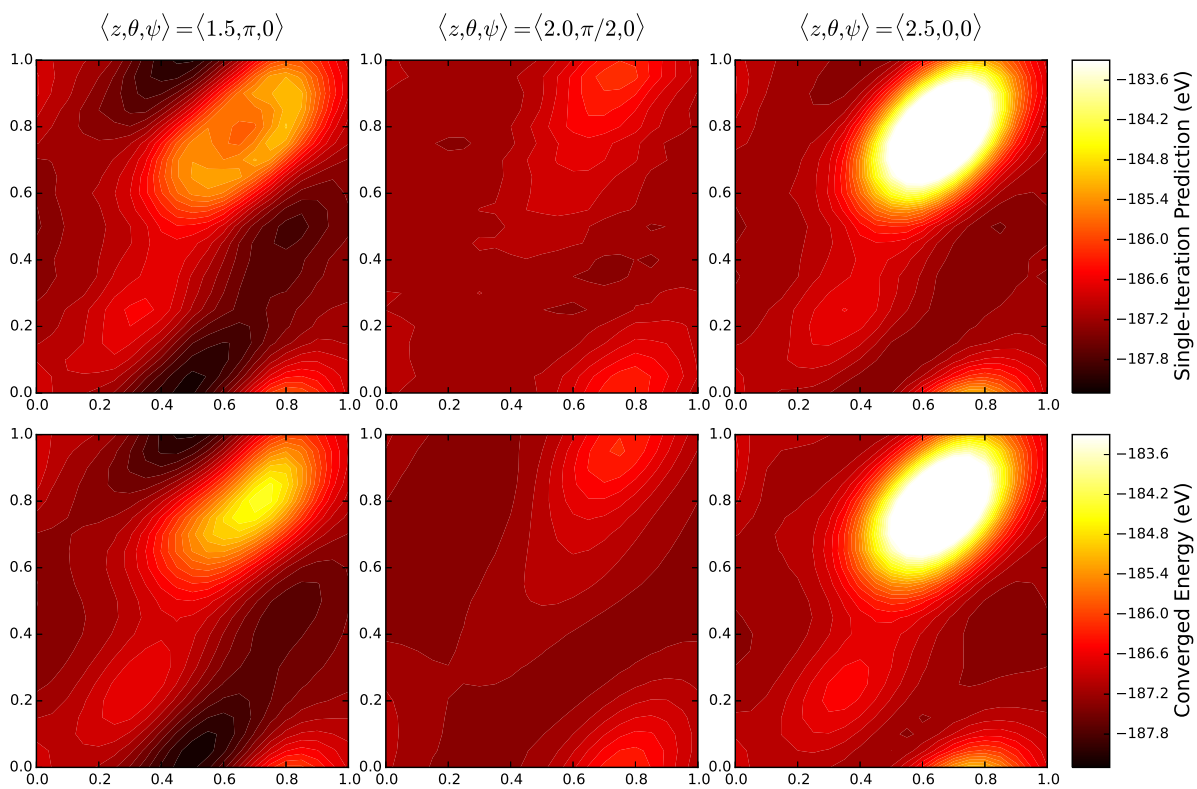


Figure 2.5: Three configurations of CO on  $\text{Fe}_2\text{O}_3$  using two potential energy measures, taking slices along the  $xy$ -plane. Observe how the initial energy closely models the converged energy. The global minimum is at the center of the dark area in the leftmost column.



## Approximating the Single-Iteration Posterior Distribution

A close approximation to the Bayes' prediction conditioned on a single observation, i.e., Equation 2.10, is to simply take the observation and add the mean of the delta distribution:

$$Pr(E_\infty|O_1, D) \approx \mathcal{N}(O_1 + \mu_1; \sigma_1^2) \quad (2.15)$$

One can choose to use Equation 2.15 instead of Equation 2.10 when performing the optimization step of Chapter 3.

## 2.3 Choosing Mode and Basis Set

Section 2.2 showed how we can estimate the final potential energy of a system by running just a single SCF iteration. The next question we need to ask is, how do GPAW's mode and basis set affect the outcomes?

In Section 2.1.1, I listed the modes and basis sets that we can plug into GPAW. I will consider four combinations. Note that the PW mode uses its own initialization strategy:

1. FD with PPW (default)
2. FD with DZP
3. LCAO with DZP
4. PW

Figure 2.6 shows how each of these combinations model four different points in the parameter space. I plotted a total of eight potential energy measures, including the single-iteration energy (based on the approximation in Equation 2.15) and the converged energy for each combination of mode and basis set.

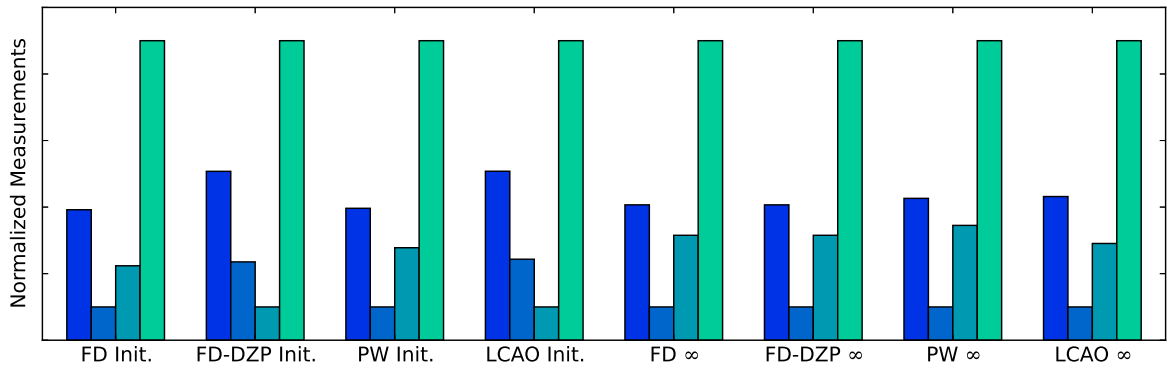


Figure 2.6: Eight measures of the potential energy for CO on  $\text{Fe}_2\text{O}_3$ . The four measures on the left are based off a single SCF iteration, while the four measures on the right correspond to SCF run to convergence. The four bars correspond to four different molecule configurations. To facilitate comparison, each set of measurements was normalized linearly to have the same minimum and maximum values.

As expected, the four fully-converged measures agree with one another. Of the single-iteration measures, the first and third (FD-PPW and PW) are the two that best agree with those converged measures. For the remainder of this thesis, I will be using a single-iteration FD-PPW for all calculations.

# Chapter 3

## Searching for Active Site

Chapter 1 presented a well-defined objective function we can use for optimization. The objective function has the following properties:

1. Slices from the objective function appear to be rather **smooth**.
2. Some dimensions of the objective function are **periodic**, while others are not.
3. Even with the improvements of Chapter 2, the objective function is **expensive**.
4. However, Chapter 2 gives us **confidence intervals** over the objective function.

In this chapter, I present how we can use *Bayesian optimization* [18] (BO) to efficiently minimize the objective function. In Section 3.1, I introduce theory behind BO. Then, in Section 3.2, I discuss how we can make the best use of BO. Finally, in Section 3.3, I give a visualization of BO.

### 3.1 Background: Bayesian Optimization

Bayesian optimization is a method for finding the global minimum of an objective function  $f(\vec{x})$  over some bounded set of parameters  $\mathcal{X}$ . It was first documented in 1978 [19], but it remained largely confined to theoretical literature until the past decade, when it has found several practical applications in the machine learning movement [20, 21]. Compared with other global optimization algorithms, BO has an advantage when the objective function is

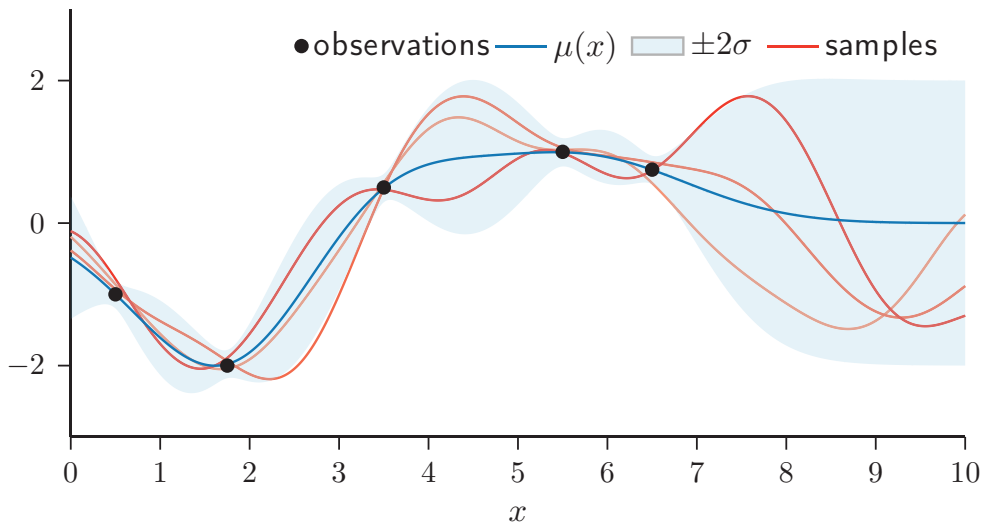


Figure 3.1: A Gaussian process is a Bayesian approach to nonlinear regression. In addition to point estimates, a GP provides us with confidence intervals over the function values. In this figure, a GP is fitted to a 1-D function with several noisy observations. [25]

expensive (more expensive than the optimization routine) and lacks a well-defined mathematical representation [20]. In the last two years, BO has begun seeing practical applications outside of machine learning, in areas such as robotics [22], bioengineering [23], and mechanical engineering [24]. However, to the best of my knowledge, this thesis is the first documented application of BO to the field of theoretical surface science.

### 3.1.1 The Algorithm

The Bayesian optimization algorithm<sup>5</sup> starts by considering the set  $D$  of previous observations of the function. It takes those observations and uses a kernel  $K$  to fit a *Gaussian process* (GP) to  $D$ . (See Figure 3.1 for more information about GPs and Section 3.1.2 for more information about kernels.)

The next step of BO is to decide what data point it should request from the objective function. It does this by way of an *acquisition function*. Typically, an acquisition function will return one of two types of points: for tuning a known local minimum (which we call

<sup>5</sup>The Bayesian optimization algorithm I'm talking about here is not to be confused with the mostly unrelated algorithm having the same name published by Martin Pelikan in 2002.

*exploitation*), or for considering a new area of the objective function space (which we call *exploration*). The classical acquisition function is the *maximum expected improvement*, defined as follows [26]:

$$x^* \equiv \arg \max_x \mathbb{E}[u(x) | x, D] \text{ for } u(x) = \max(0, y_{min} - y(x)) \quad (3.1)$$

$$= \arg \max_x \int_{-\infty}^{y_{min}} (y_{min} - \hat{y}) \mathcal{N}(\hat{y}; \mu_x, K_{xx}) d\hat{y} \quad (3.2)$$

$$= \arg \max_x (f_{min} - \mu_x) \Phi(f_{min}; \mu_x, K_{xx}) + K_{xx} \mathcal{N}(f_{min}; \mu_x, K_{xx}) \quad (3.3)$$

where  $y_{min}$  is the smallest observation of the objective function so far,  $y(x)$  is the objective function,  $\mu_x$  is the mean function of  $x$  (typically a constant),  $K_{xx}$  is the kernel function of  $x$ ,  $\Phi$  is the normal cumulative density function (normal CDF), and  $\mathcal{N}$  is the normal probability density function (normal PDF). Although the maximum expected improvement can be expressed in the closed form of Equation 3.3, it is itself a non-convex function with many local minima. In practice, one solves Equation 3.3 using their choice of classical optimization algorithm.

With  $x^*$  in hand, BO calls the expensive, black-box objective function to acquire  $y^*$ . The pair  $(x^*, y^*)$  is added to  $D$ , and BO moves on to the next iteration. BO will continue exploring and exploiting the function space until the user is happy with the result.

Note that BO is completely deterministic, with two exceptions: the choice of initial data points, and the maximization of the expected improvement in the case of a tie. If one uses a deterministic solution for solving those two problems, then BO will give the same result in the same number of iterations every time.

### 3.1.2 Kernels

A *kernel*, or *covariance function*, is a function that takes two points and returns a measure of the correlation between them. The classical kernel function is the squared exponential:

$$K^{SE}(x, y) = K_{xy}^{SE} \equiv \sigma^2 \exp\left(-\frac{\|x - y\|^2}{2l^2}\right) \quad (3.4)$$

where we call  $\sigma^2$  the *variance* and  $l$  the *length scale*. One can think of the squared exponential kernel as fitting a normal distribution over every point in  $D$ , and combining the results to obtain a nonlinear fit to the function. Note that the smaller the length scale  $l$ , the closer two points need to be to one another in order to be correlated.

It turns out that the squared exponential kernel is not the best choice for most real-life problems, so in practice, one typically uses an alternative, such as the *Matérn* kernel [27]. The Matérn kernel is related to the squared exponential kernel, but it is more resilient to outliers and rough edges in the data. It adds an additional hyperparameter  $\nu$ , which relates to the smoothness of the function. As  $\nu \rightarrow \infty$ , the function is modeled as being more and more smooth, and Matérn converges to the squared exponential.  $\nu$  typically takes on a value of either  $\frac{3}{2}$  or  $\frac{5}{2}$  for computational reasons.

The variance and length scale of the squared exponential and Matérn kernels are examples of what we call *hyperparameters*. These hyperparameters need to be specified in order for Bayesian optimization to work. It is possible to set the hyperparameters by solving yet another optimization problem. However, if we already know something about the topology of the objective function, it's better to set the hyperparameters ahead of time.

## 3.2 Modeling the Objective Function

Bayesian optimization is a powerful tool, but in order to work best, it needs information about the objective function. This section explains how we can model the function from Chapter 2 for use in BO.

### 3.2.1 Setting the Hyperparameters

With a Matérn kernel, BO requires the following hyperparameters:

1. Likelihood noise,  $\sigma_l^2$
2. Prior mean,  $\mu_0$
3. Matérn variance,  $\sigma_K^2$
4. Matérn length scale in each dimension,  $l_d$

Below, I present a systematic approach for setting hyperparameters 1-3 using training data, and some suggestions for setting hyperparameter 4.

The training data for hyperparameters 1-3 can be acquired by evaluating several, say 5-10, coordinates drawn uniformly at random from the parameter space, which can be the same points used to “train” the Chapter 2 objective function. I found it desirable to throw away the highest 25% of initial random energy measurements in order to bias the parameters toward the lower regions of the function space.

The actual values I used in my experiments for Chapter 4 can be found in Appendix A.

#### Likelihood Noise

The parameter  $\sigma_l^2$  corresponds to the estimated confidence interval of a particular observation to the true value at that point. If we use Equation 2.10 to give our potential energy estimates, then Equation 2.12 gives the value we can use for  $\sigma_l^2$ . If we instead use the approximation in Equation 2.15, then we can set  $\sigma_l^2$  to be the variance of the Gaussian distribution fitted over the  $\Delta_1$  data, like the one shown in Figure 2.2.

Figure 3.2: Length scales for CO on Fe<sub>2</sub>O<sub>3</sub>. Recall that due to the symmetry of the CO molecule, the Euler coordinate  $\phi$  has no effect, and we are effectively dealing with a 5-D system.

Parameter	x	y	z	$\theta$	$\psi$
Length Scale	0.1	0.1	0.5	$\pi/8$	$\pi/4$

### Prior Mean and Matérn Variance

The parameters  $\mu_0$  and  $\sigma_K^2$  correspond to the value and variance of the GP, respectively, in the absence of any nearby observations. To set these parameters, we can fit a Gaussian distribution over the evaluations of our objective function based on our training data, and then use the parameters of that Gaussian as  $\mu_0$  and  $\sigma_K^2$ .

### Length Scales

Properly setting the length scales is much more difficult. Although there do exist methods to learn the length scales from data, they typically require a large quantity of function evaluations, and I found by trial and error that manually tuning the hyperparameters can sometimes result in better BO performance. Read Appendix B for more information on this point. The settings listed in Figure 3.2 are the result of my hand-tuning.

Of course, these length scales will not work for all other chemical systems. I suggest two practical methods for setting the length scales. They are both good areas for future research.

**Heuristics** I expect that the three spatial coordinates will largely depend on the catalyst identity, while the three Euler coordinates will largely depend on the molecule identity. Some questions to consider would be, how does the spacing of atoms on the surface affect the length scales in the  $x$  and  $y$  directions? How does the shape of the molecule affect the length scales in the Euler coordinates?



**Similar Functions** When learning the length scales, we do not necessarily need to use our objective function; we could instead use some other function that behaves in a similar way. A question worth asking would be, could we fit the GP length scales to a cheap classical potential like Lennard-Jones [28], and then switch over to DFT for the optimization step? (See Appendix B for more information on automatically learning hyperparameters.)

**Digression: More about Length Scales**

Once the length scales are set, they give a deeper understanding of the topology of the objective function.

For example, using a squared exponential or the Matérn kernel, a particular data point typically influences only those parts of the function that are at most 2-3 length scales away. For CO on Fe<sub>2</sub>O<sub>3</sub>, we have a total of 10 length scales in the  $x$  and  $y$  directions, an arbitrary number in the  $z$  direction (say 2, from 1.5 Å to 2.5 Å), and 8 each in the  $\theta$  and  $\psi$  directions. This means that our parameter space has  $10 \times 10 \times 2 \times 8 \times 8 = 12,800$  unit hypercubes. If each point influences a volume up to 2 length scales away, we can model it as a hypersphere with radius  $r$  in the parameter space. The volume of a hypersphere in  $n$ -dimensional space is given by:

$$V = \frac{2\pi^{n/2}r^n}{n\Gamma(n/2)} \tag{3.5}$$

For  $n=5$  and  $r=2$ ,  $V = 330.73$  unit volume. If our objective were to characterize the whole of the hyperspace, then we would need to make on the order of  $12800/330.73 = 38.7$  function evaluations. In practice, BO is going to need to make more observations than this theoretical bound, in order to acquire enough detail about the areas where it found a minimum.

The other way of thinking about length scales is to say, given a particular point in space, it is only *influenced* by those points that are within 2 or 3 length scales. This fact will come in handy in the following section.

### 3.2.2 Handling Periodic Variables

It would be nice to enforce the periodicity of our variables at the kernel level. That is still an active area of research, however, and there are no out-of-the-box solutions for enforcing periodicity over more than one dimension.

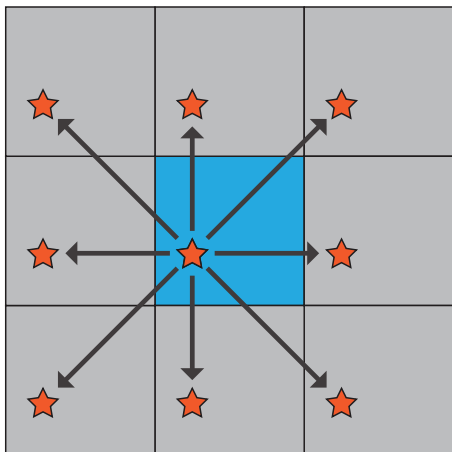


Figure 3.3: Points can be copied to adjacent unit cells to emulate periodicity.

Instead, what we can do is to model the system as an infinite grid of points, where each point is copied across to all adjacent cells. For example, in two periodic dimensions, we could copy all points in the first unit cell to the 8 adjacent cells, illustrated in Figure 3.3.

It should be rather obvious that this solution does not scale well. For example, in our system we typically have 4-5 active periodic dimensions  $d$ . Including the origin cell, the number of times we need to copy a point is given by

$$N(d) = 3^d \tag{3.6}$$

So when  $d = 5$ , we need 243 copies of every point. Since the time complexity of a GP prediction scales as  $O(n^3)$  to accommodating a matrix inversion, we get  $T(n, d) = O((3^d n)^3) = O(27^d n^3)$ , which quickly becomes prohibitively expensive.

However, we can be clever. Since a particular point is only influenced by other points that are within 2 or 3 length scales, we can search only for those points within a fixed radius of a query point  $p$  in linear time, construct an interim GP trained only on those points, and then make our prediction. If we assume that the number of points in a particular ball of the hypersphere is bounded by some small constant, we've effectively reduced the query runtime to  $O(3^d n)$ .

We don't have to stop there, though. Finding the set of points within a certain radius is a classic problem in computational geometry, and one for which there are efficient solutions.

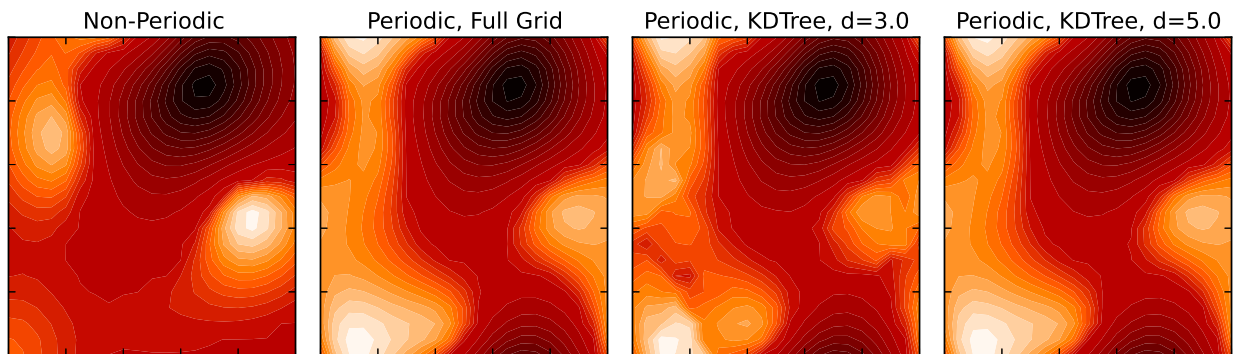


Figure 3.4: Plots of a GP fitted over a toy 2-D function with 8 observations.

We can use a  $k$ - $d$  tree to perform the search query in only  $O(\log n)$  time, at the expense of incurring a nominal  $O(\log n)$  penalty when adding points to the tree [29, 30]. This gives us a time complexity of  $O(\log(3^d n)) = O(d + \log n)$  for evaluating a point in our GP.

Figure 3.4 illustrates these different approaches on a toy data set. On the left is the standard Matérn kernel, without any information about periodicities. In the second panel (“Full Grid”), we copy points to the 8 adjacent cells, without any additional optimizations. In the third panel, we use a  $k$ - $d$  tree to grab only those points that are within 3 length scales; we get a nice fit, although there are a few details that are not the same as in Full Grid. The fourth panel shows a  $k$ - $d$  tree searching for points within 5 length scales; this contour plot is now virtually indistinguishable from the Full Grid one.

I ran some benchmarks to get an idea of how these different methods would actually perform.<sup>6</sup> My results are in Figure 3.5. Up to around 100 points, the Full Grid method is faster than the  $k$ - $d$  tree; this is most likely due to the fact that it can take advantage of machine-level optimizations and caching since all query points are referenced against the same GP. However, as the number of points grows to 200 or 400, the  $k$ - $d$  tree outperforms Full Grid. The  $k$ - $d$  tree with shorter length scale (and fewer points per evaluation) slightly outperforms the one with longer length scale. The non-periodic kernel is still the fastest option in all four cases, since it has the fewest points and it can use machine-level optimizations.

<sup>6</sup>Each test was run five times; the bar height corresponds to the mean runtime, and the error bars indicate the best and worst runtimes. The points used to condition the GP were sampled uniformly at random from the parameter space. The same set of points was used for each method.

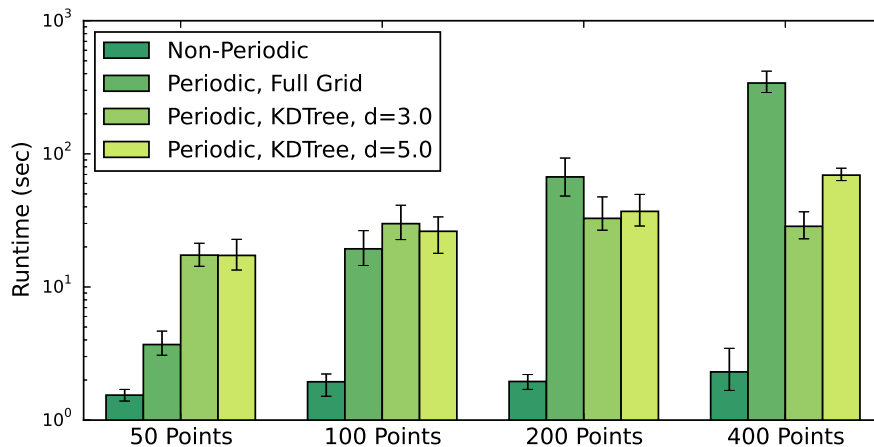


Figure 3.5: Time to compute the max expected improvement for the CO-on-Fe<sub>2</sub>O<sub>3</sub> data set using the four methods from Figure 3.4. Note the log scale.

### 3.3 Bayesian Optimization Demo

It is difficult to visualize Bayesian optimization in six dimensions, so I instead picked a 2-D “toy function.” The function is an  $xy$  slice of the Lennard-Jones potential for the CO-on-Fe<sub>2</sub>O<sub>3</sub> system at  $\langle z, \theta, \psi \rangle = \langle 1.5, \pi, 0 \rangle$ . The Lennard-Jones potential fails to correctly model this system, but it shares some properties with the DFT strategy, so it is a handy tool for quick test runs. I used the Full Grid procedure with the same length scale parameters as the DFT system. In Chapter 4, I use DFT (one iteration) as my objective function.

The result is shown in Figure 3.6. BO manages to converge on the global minimum in just a few function evaluations.

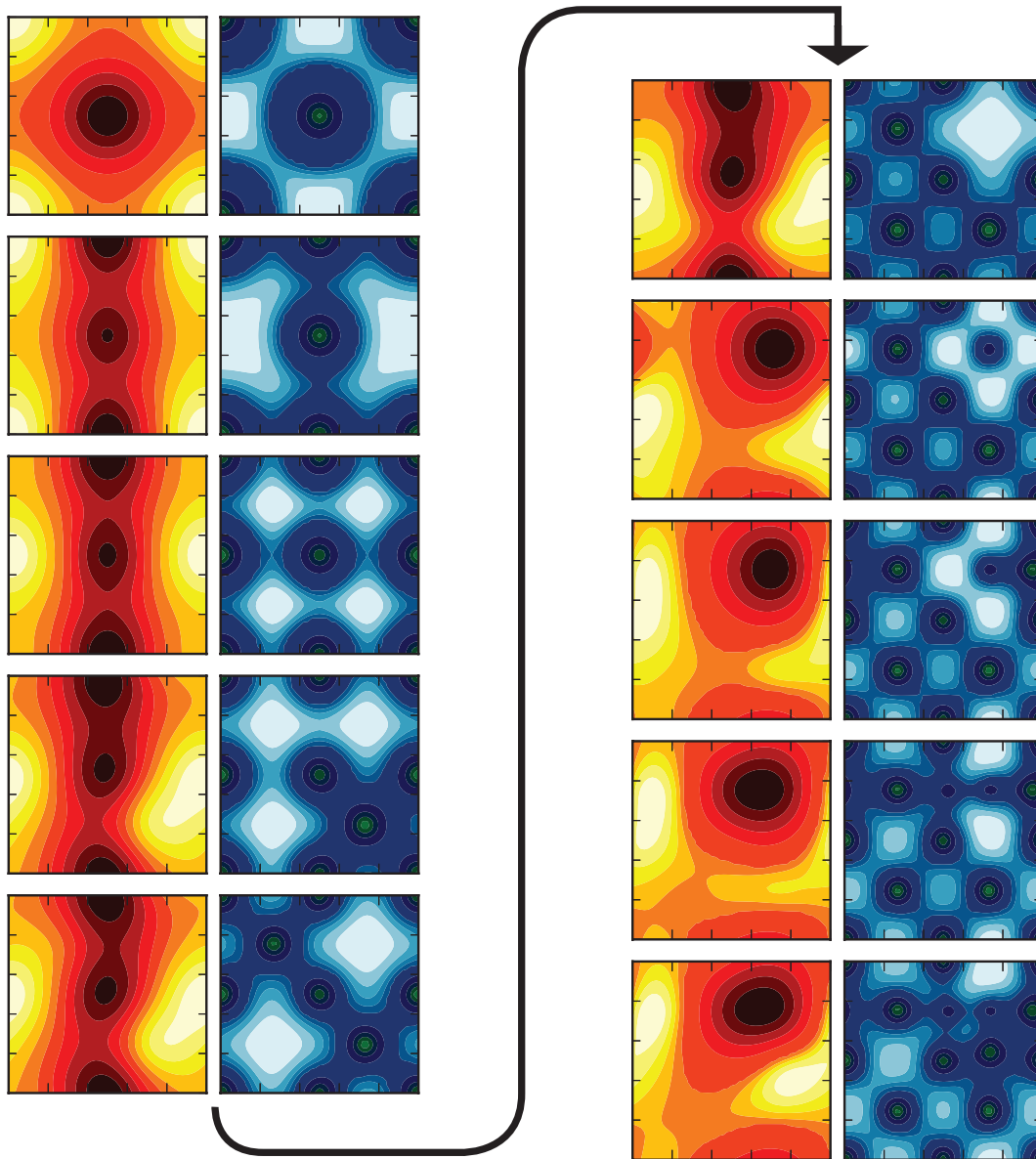


Figure 3.6: Bayesian optimization running on a toy function in two dimensions. You can read the diagram as a “film strip,” in which the yellow cell on the left is the mean of the current GP, and the blue cell on the right is the expected improvement. It starts in the upper-left pane with two initial observations, one at the corner and one at the center. At each step, BO picks a point that has maximum expected improvement (bright areas in the blue cells), evaluates it, and adds it to the GP of the next row. BO is in “exploration” mode in panels 1-6, and when it finds a good point in panel 7, it switches to “exploitation” mode.

# Chapter 4

## Results and Discussion

Chapters 2 and 3 discussed the theoretical justifications for my approach to identifying the active site on a catalyst surface. This chapter shows the results of running my algorithm and compares them to two other state-of-the-art routines.

Section 4.1 introduces the two state-of-the-art routines. Section 4.2 presents a comparison of my routine with the state-of-the-art routines. Finally, Section 4.3 has some discussion about the results.

### 4.1 Background: State of the Art

Before showing the results of my algorithm, I want to introduce two state-of-the-art routines that could also be used to solve this problem. The first is a method that was developed specifically for the molecule-on-surface problem studied in this thesis. The second is a general method for solving global optimization problems.

#### 4.1.1 Constrained Minima Hopping

*Constrained minima hopping* (CMH) is a routine for optimizing adsorbate-surface structures, documented by Andrew Peterson in 2014 [10]. It builds on an earlier routine called *minima hopping* proposed by Stefan Goedecker in 2004 [31]. These routines are not to be confused

with *basin hopping*, an older Monte Carlo method for relaxing chemical structures proposed by David Wales in 1997 [32].

At a high level, minima hopping iteratively “shakes” a system by adding kinetic energy, and then lets the system relax to a local minimum.<sup>7</sup> Every time the system relaxes, if the atoms represent a new configuration, that configuration is added to a “minima list.” (If they don’t represent a new configuration, the temperature is increased even more to shake the atoms farther away.) This procedure is repeated several times, until the user is happy with the result. Peterson describes it as follows:

[Minima hopping] starts with an atomic configuration that has been optimized to any local minimum in potential energy. ... The atoms are then thermalized, or given kinetic energy in a Maxwell-Boltzmann distribution corresponding to a specified initial temperature  $T_0$ , and allowed to evolve in a microcanonical (NVE) molecular dynamics simulation. As the molecular dynamics evolves, the potential energy of the simulation is monitored; after  $n_{\text{MDmin}}$  path minima are encountered, the molecular dynamics simulation is stopped and the atoms are optimized to their nearest local minimum-energy structure. ([10])

Peterson then points out a problem with classical minima hopping: that a multi-atom adsorbate molecule could potentially lose its identity, with the individual atoms “hopping” to different spots on the surface. CMH solves this problem by adding a *Hookean constraint*, which enforces the identity of the adsorbate molecule and prevents the simulation from breaking its bonds.

CMH can use any choice of method for relaxing the system and determining the potential energy. In my experiment, I let CMH use GPAW to evaluate the energies and forces and to run the local optimizations. I have to let GPAW run all the way to convergence, for reasons discussed in Section 4.3.2.

---

<sup>7</sup>By default, CMH performs the local optimization over the three spatial coordinates for each atom in the molecule using L-BFGS with a line search mechanism, keeping the surface fixed.

Figure 4.1: Hyperparameter settings for Differential Evolution.

Parameter	Pop Size	Tol	Mutation	Recombination
<b>Default Settings</b>	30	0.01	0.5 to 1	0.7
<b>Better Settings</b>	5	0.01	0.5 to 1	0.9

### 4.1.2 Differential Evolution

In contrast to CMH, *differential evolution* (DE), first documented by Rainer Storn in 1997, is a general routine for the bounded global optimization of a generic objective function [33]. DE is a type of *evolutionary strategy* algorithm, which is related to the more well-known class of algorithms known as *genetic algorithms*.

DE starts with a fixed number of solution candidates, which should be reasonably well-distributed over the parameter space. It obtains results from those candidates, and at each step, performs a “mutation” process in which the next generation of candidates are more likely to inherit properties from the “best” candidates in the previous generation.

Like BO, DE has hyperparameters needing to be set. I hand-tuned the DE hyperparameters to the values that gave the algorithm the best performance while still succeeding to find the global minimum in at least 75% of runs. I listed the default settings and my customized settings in Figure 4.1.<sup>8</sup>

In my experiment, I give DE the same 1-iteration objective function as I give to BO. I used the “best1bin” strategy from the implementation in SciPy [7].

## 4.2 Results

To compare my algorithm with the two approaches from Section 4.1, I ran five different routines to optimize the configuration of CO on Fe<sub>2</sub>O<sub>3</sub>:

---

<sup>8</sup>The “default” settings are those in the SciPy implementation of DE as of SciPy version 0.15, with an increased population size of 30.



1. Constrained minima hopping with its default parameters and a full GPAW calculator.
2. Differential evolution with the default parameters according to Figure 4.1.
3. Differential evolution with my custom parameters according to Figure 4.1.
4. Bayesian optimization with  $k$ -d tree periodic variables and the hyperparameter settings listed in Appendix A.
5. Bayesian optimization with non-periodic variables and the hyperparameter settings listed in Appendix A.

All five routines converged, and they all identified the configuration shown in Figure 4.2, corresponding to about -187.51 eV. More details on the experimental setup can be found in Appendix A.

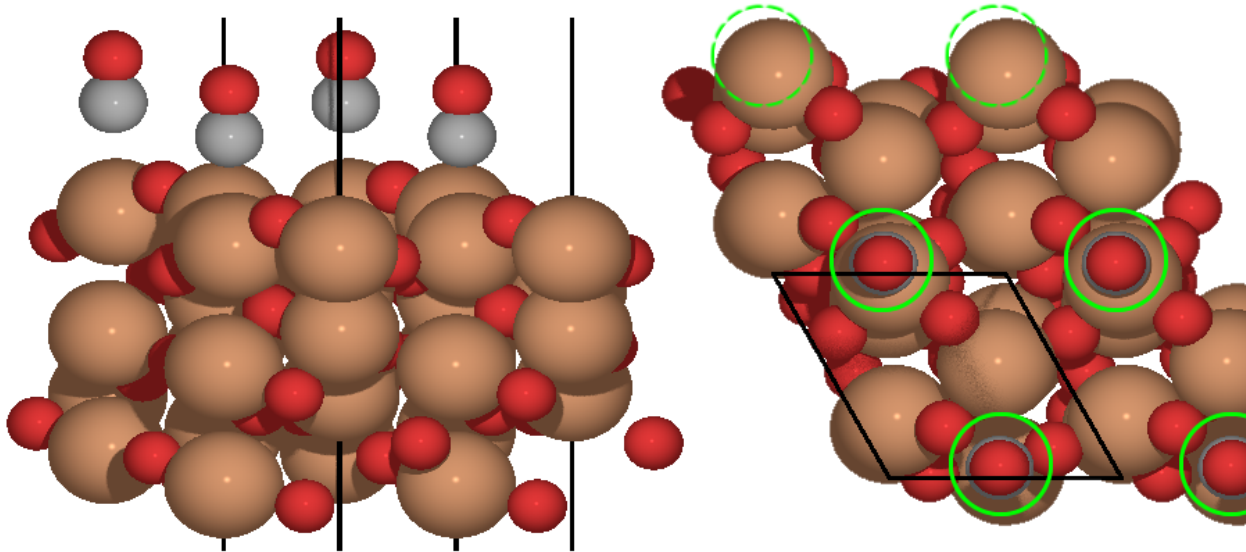
### 4.2.1 Runtime Comparison

What sets the three methods apart from one another is the amount of runtime required in order to find the minimum.

Since the routines have radically different termination conditions, it would not be fair to simply consider when they terminate. Instead, I calculate the runtime in the following manner, which I call “time to global minimum.”

- **Constrained Minima Hopping:** Time to the acceptance of the global minimum into the minima list.
- **Differential Evolution and Bayesian Optimization:** Time to the *first evaluation* of the objective function to within  $10^{-1}$  eV of the global minimum. (This means that all function evaluations up until the “time to global minimum” were greater than  $10^{-1}$  eV higher than the global minimum.)

The results are shown in Figure 4.3. On average, my routine outperforms differential evolution by a factor of 11:1, and my routine outperforms constrained minima hopping by a factor of 75:1.



Parameter	x	y	z (Å)	$\theta$	$\psi$
Value	0.51	0.03	1.5	$\pi$	n/a

Figure 4.2: All three routines identify the above configuration for CO on  $\text{Fe}_2\text{O}_3$ . The black lines in the figures indicate the unit cell; the unit cell is copied twice for visualization purposes. The green circles indicate the positions of CO. The dotted green circles show where CO would be if the unit cell were copied one more time in the  $y$  direction. When  $\theta = 0$  or  $\theta = \pi$ , the Euler angle  $\psi$  does not have an effect.

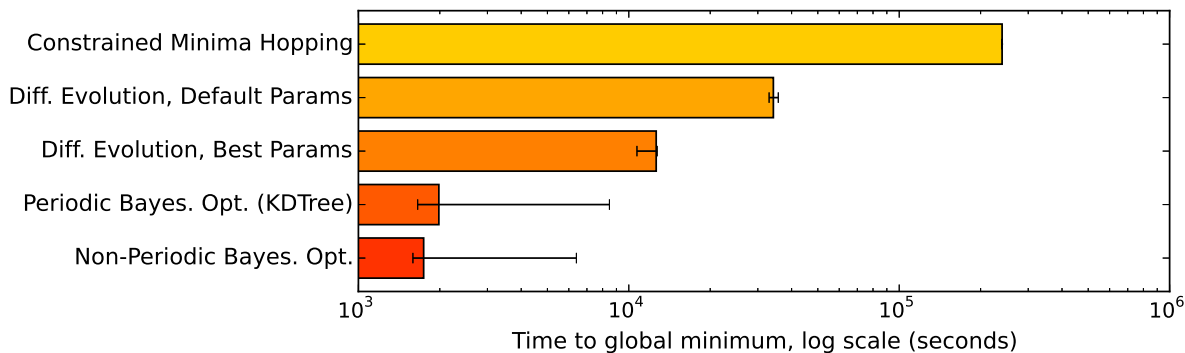


Figure 4.3: Comparison of the running time of my routine and the two state-of-the-art routines. For DE and BO, the error bars represent the minimum and maximum times to find the global minimum out of several runs.

## 4.3 Discussion

### 4.3.1 Hyperparameters: BO versus DE

In Section 3.2.1, I discussed my methodology for setting the hyperparameters for BO, and similarly for DE in Section 4.1.2.

Although both algorithms have hyperparameters needing to be specified, the hyperparameters of BO arise from the topology of the objective function, while the hyperparameters of DE arise from the inner workings of the genetic algorithm (relating to ideas such as the quantity and spacing of local minima).

This can be seen as an additional advantage for BO over DE, since the methodology in Section 3.2.1 gives us enough information to set BO's hyperparameters with a single pre-processing step.

### 4.3.2 SCF Loop and Constrained Minima Hopping

An acute reader will argue that my comparison against CMH is not fair, because I let CMH run the SCF loop in GPAW all the way to convergence, while BO and DE stop the SCF loop after only one iteration.

The reason I have to do it this way comes from the fact that CMH performs a *local optimization* using L-BFGS, which requires information about the forces acting on each atom. (Force is the negative gradient of energy with respect to distance.) Although GPAW provides information about the atomic forces, the method described in Section 2.2.2 for predicting the final energy given just a few SCF iterations does not work as well for forces. Figure 4.4 shows SCF traces for the x-force on the O atom in CO for four different configurations. There is no correlation strong enough to enable us to predict with high accuracy the converged value of the force by running only a few iterations.<sup>9</sup> Compare this to Figure 2.1, which shows that the traces of the energies are highly correlated.

---

<sup>9</sup>When the forces are projected into spherical coordinates, there is still no strong correlation.

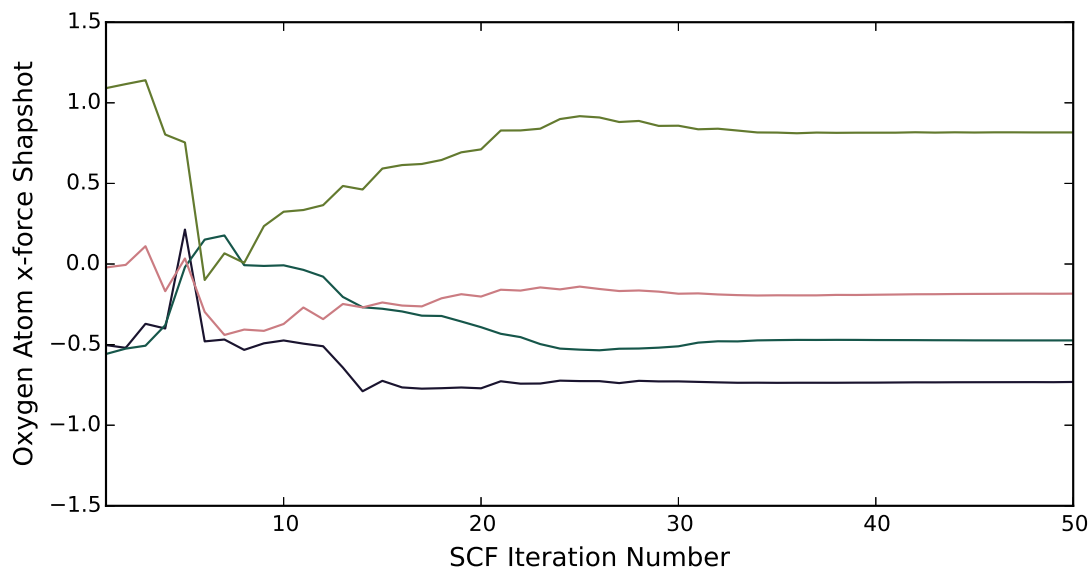


Figure 4.4: Values of the x-force on the oxygen atom in CO as GPAW runs an SCF loop for four different configurations, using finite difference (FD) mode with pseudo partial wave (PPW, default) basis set.

Despite this disadvantage, I attempted to run CMH using my Bayesian model applied to the forces as well as the energies, conditioning the two values on 25 iterations of SCF. This attempt failed because the local optimizer, L-BFGS with a line search mechanism, would not converge with such noisy force evaluations.

There has recently been research into alternative methods for obtaining atomic forces for DFT; one uses a physics-based approach to improve the initial guess [34], and another uses a machine learning approach to predict forces based on many more training simulations [35]. It would be interesting to see how these methods would affect the runtime of CMH.

### 4.3.3 Parallelization

Multicore environments are now the standard for performing computational chemistry calculations. All three of the routines in this chapter can take advantage of search parallelization. It would be interesting to see how implementing search parallelization would affect the overall runtime of the three routines.

**Bayesian Optimization** Classical BO is a serial algorithm, evaluating the point returned by the maximum expected improvement. However, there has been some research into methods to improve the parallelism of BO. One approach is to have multiple calculations running in parallel, and whenever one of the calculations finishes, pick the maximum expected improvement based both on the known points as well as on a belief over the “fantasy” points that have yet to finish calculating [21].

**Constrained Minima Hopping** CMH is also a serial algorithm, repeatedly running MD and relaxation steps. However, Peterson suggests sharing a common list of known minima between several searches running in parallel [10].

**Differential Evolution** DE, in contrast to BO and CMH, is designed to be able to take advantage of multiprocessor environments. At each step of evolution, all of the candidates can be evaluated in parallel. For optimal performance, the number of candidates can be set to be a multiple of the number of processors.

# Chapter 5

## Conclusions

This thesis has documented a novel approach to identifying the active site of a molecule on a surface.

There are numerous applications for such a tool. For example, one can perform high-throughput screening for catalysts by inspecting properties such as the adsorption energy, a calculation that can be automated with the help of the methods in this thesis.

The code behind this tool will soon be released as a module to the Atomic Simulation Environment (ASE).

### 5.1 Future Work

I leave the reader with a few open questions, which would be good candidates for future work in this field.

**Hyperparameters** In my opinion, the part of this thesis that deserves the most additional work would be in the setting of the hyperparameters. In Section 3.2.1 and Appendix B, I gave a number of strategies and heuristics for setting the hyperparameters, but none of these approaches have been thoroughly tested or compared to one another.

**Kernels** The approach in Section 3.2.2 involving  $k$ -d trees to enforce periodicity worked for this application, but a much more general and elegant solution would be to incorporate periodicity into the kernel directly. In addition, relationships between the parameters, such as between  $\theta$  and  $\psi$  (see Appendix B), should ideally be incorporated into the kernel as well.

**Additional Parameters** The six parameters of Section 1.2.1 worked well for CO on  $\text{Fe}_2\text{O}_3$ , but deformation by adsorption may be less negligible for larger systems. It would be worthwhile to come up with a handful of parameters that can accurately represent deformation by adsorption.

**Parallelization** In Section 4.3.3, I discussed approaches to utilize multicore architectures in my routine as well as the state-of-the-art routines. It would be interesting to see how utilizing multiple cores would affect the wall clock times of the various routines.

**Bias Function** Figure 2.2 showed how there is a limited, but existent, correlation between  $\Delta_1$  and location on the surface. It could help improve the model if we were able to characterize this “bias function,” that is,  $Pr(\Delta_1|x, y, D)$ . One observation, based only on Figure 2.2 and ones like it, is that areas with higher energy tend to have higher  $\Delta_1$ ’s, and vice-versa. It would be interesting to see if we could take that information and improve our  $E_\infty$  predictions.

# Appendix A

## Details on Experimental Setup

Throughout this paper, I have focused my study on CO on Fe<sub>2</sub>O<sub>3</sub>.

All of my atomic manipulations were performed with the Atomic Simulation Environment (ASE) [6], a popular library written in Python.

### A.1 Surface Preparation

I obtained a crystal structure for Fe<sub>2</sub>O<sub>3</sub> from Materials Project. I used the  $R\bar{3}C$  spacegroup, corresponding to material ID “mp-24972” [36, 37].

I used ASE to cut the 001-surface from the crystal structure, with two layers of atoms and 15 Å of vacuum.

I then performed a structure relaxation. I did so using regular GPAW and L-BFGS without any fancy optimizations. I let the surface relax until the forces on every atom were less than 0.05 eV/Å. This relaxation process took 7 hours and 41 minutes.

The relaxed Fe<sub>2</sub>O<sub>3</sub> surface was used as a basis for all further calculations, including Bayesian optimization, differential evolution, and constrained minima hopping.



Figure A.1: Hyperparameter settings for Bayesian Optimization (all units are in eV).

Parameter	Likelihood Noise	Prior Mean	Matérn Variance	Length Scales
Value	0.021	-186.965	1.2	See Figure 3.2

## A.2 Parameter Space and Objective Function

In my experiment,  $x$  and  $y$  were treated as fractional coordinates, while  $z$  has units of Å. All angles were measured in radians.

The CO molecule’s initial configuration had the oxygen atom 1.128 Å below the carbon atom along the  $z$  axis, chosen such that the first axis of rotation,  $\phi$ , did not have an effect. The center of the oxygen atom was always the reference point for the  $x$ ,  $y$ , and  $z$  parameters.

The objective function was evaluated according to Equation 2.15.

## A.3 Bayesian Optimization

I use differential evolution [33] to maximize the expected improvement (Equation 3.3). DE is stochastic, which is the reason why my runs of BO take different numbers of iterations to reach the minimum.

I always start with the same two initial points: one at the origin, and one at the center of the hyperspace.

I use a Matérn kernel with  $\nu = \frac{5}{2}$ . The other hyperparameters are listed in Figure A.1.

Rather than enforcing length scales on a kernel level, I scale all data points before adding them to the GP in order to make the kernel length scales all equal to unity. The advantage of this approach is that it makes the data set more amenable to  $k$ -d trees.

# Appendix B

## Hyperparameter Optimization

I mentioned in Section 3.2.1 that there exist methods for automatically setting the values of the hyperparameters for a Gaussian process, but that the automatic settings are sometimes not the best for BO. The goal of this short appendix is to explain what I mean by those statements.

### B.1 Maximizing Likelihood

One can look at setting hyperparameters as yet another optimization problem [38]. In this case, we have some number of data points, and what we want is the “best” hyperparameter settings for the GP.

To quantify what makes a “good” hyperparameter setting, we typically look at the *marginal likelihood* of the GP, corresponding to  $Pr(\vec{y}_D | X_D, \text{model})$ , where  $X_D$  and  $\vec{y}_D$  are the coordinates and values, respectively, of some set of known data points. The higher the marginal likelihood, the better the fit of the GP to the data. We can therefore find the values of the hyperparameters that maximize the marginal likelihood. (The implementation in GPy uses L-BFGS to minimize the negative log of the marginal likelihood [2].)

Figure B.1: Length scales estimated by three runs of GPy’s optimizer.

Parameter	x	y	z	$\theta$	$\psi$
<b>Run 1</b>	0.182	0.289	0.398	0.636	1.014
<b>Run 2</b>	0.108	0.184	0.444	0.705	3.142 <sup>a</sup>
<b>Run 3</b>	0.223	0.180	0.585	0.857	1.928

<sup>a</sup>This was the upper bound set on  $\psi$  during the optimization procedure.

## B.2 Considerations with Automatic Optimization

I ran the GPy optimizer several times on different sets of 2000 points from my objective function. I held all hyperparameters fixed except for the five shown in Figure B.1. The results of the first three runs are shown in the table. The points were drawn uniformly at random without replacement from a grid of function evaluations ranging over  $0 \leq x, y \leq 1$  in 0.05 increments,  $1.5 \leq z \leq 2.5$  in 0.25 Å increments,  $0 \leq \theta \leq \pi$  in  $\pi/8$  increments, and  $0 \leq \psi \leq 2\pi$  in  $\pi/4$  increments. (This is the same grid of over 150,000 points referenced elsewhere in this thesis.)

For  $x$ ,  $y$ , and  $z$ , I’m happy with the values returned by the optimizer. They are a bit higher than, but more-or-less in line with, my own settings in Figure 3.2.

However, the settings for the Euler coordinates, especially  $\psi$ , are problematic. It turns out that since CO is symmetric about an axis, the length scale for  $\psi$  is actually a function of  $\theta$ : when  $\theta = 0$  or  $\theta = \pi$ , then  $l_\psi = \infty$ , but when  $\theta = \pi/2$ , then  $l_\psi$  should be small.<sup>10</sup> This unusual relationship between  $\psi$  and  $\theta$  causes GPy to find an artificially large value for  $l_\psi$ .

The overestimated length scales would cause the GP to draw incorrect conclusions about certain areas of the parameter space, which may cause BO to fail to explore promising regions of the function. This would be a worthwhile area for future research.

<sup>10</sup>Since  $\phi$  rotates about the  $z$  axis, we set CO’s axis of symmetry to be that axis. The variable  $\psi$  rotates about the  $z'$  axis.  $z = z'$  when  $\theta = 0$  or  $\theta = \pi$ , and in those cases, the  $\psi$  coordinate has no effect.

# References

- [1] John T. XSEDE: Accelerating Scientific Discovery. 2014;16(5):62–74. Computing in Science and Engineering.
- [2] The GPy authors. GPy: A Gaussian process framework in python; 2012–2015. Available from: <https://github.com/SheffieldML/GPy>.
- [3] Mortensen JJ, Hansen LB, Jacobsen KW. Real-space grid implementation of the projector augmented wave method. Physical Review B. 2005;71(3):035109. PRB.
- [4] Enkovaara J, Rostgaard C, Mortensen JJ, Chen J, Duak M, Ferrighi L, et al. Electronic structure calculations with GPAW: a real-space implementation of the projector augmented-wave method. Journal of Physics: Condensed Matter. 2010;22(25):253202.
- [5] Larsen AH, Vanin M, Mortensen JJ, Thygesen KS, Jacobsen KW. Localized atomic basis set in the projector augmented wave method. Physical Review B. 2009;80(19):195112. PRB.
- [6] Bahn SR, Jacobsen KW. An object-oriented scripting interface to a legacy electronic structure code. Computing in Science & Engineering. 2002;4(3):56–66.
- [7] Jones EJ, Oliphant T, Peterson P, et al.. SciPy: Open source scientific tools for Python; 2001–. [Online; accessed 2015-12-19]. Available from: <http://www.scipy.org/>.
- [8] Prez F, Granger BE. IPython: a system for interactive scientific computing. Computing in Science & Engineering. 2007;9(3):21–29. Available from: <http://ipython.org/>.
- [9] Hunter JD. Matplotlib: A 2D graphics environment. Computing in science and engineering. 2007;9(3):90–95. Available from: <http://matplotlib.org/>.
- [10] Peterson AA. Global Optimization of AdsorbateSurface Structures While Preserving Molecular Identity. Topics in Catalysis. 2014;57(1-4):40–53.
- [11] Wikipedia. Euler angles — Wikipedia, The Free Encyclopedia; 2015. [Online; accessed 3-December-2015]. Available from: [https://en.wikipedia.org/w/index.php?title=Euler\\_angles&oldid=688367057](https://en.wikipedia.org/w/index.php?title=Euler_angles&oldid=688367057).
- [12] Liu D, Nocedal J. On the limited memory BFGS method for large scale optimization. Mathematical Programming. 1989;45(1-3):503–528.

- [13] Jensen F. Introduction to Computational Chemistry. Wiley; 2007.
- [14] Cuevas JC. Introduction to Density Functional Theory. Institut für Theoretische Festkörperphysik, Universität Karlsruhe; 2011.
- [15] Web of Science. Search Results. Thomson Reuters; 2015. Available from: <http://wokinfo.com/>.
- [16] Osei-Kuffuor D, Fattebert JL. Accurate and Scalable O(N) Algorithm for First-Principles Molecular-Dynamics Computations on Large Parallel Computers. Physical Review Letters. 2014;112(4):4.
- [17] Wikipedia. Conjugate prior — Wikipedia, The Free Encyclopedia; 2015. [Online; accessed 4-December-2015]. Available from: [https://en.wikipedia.org/w/index.php?title=Conjugate\\_prior&oldid=686670771](https://en.wikipedia.org/w/index.php?title=Conjugate_prior&oldid=686670771).
- [18] Jones DR, Schonlau M, Welch WJ. Efficient global optimization of expensive black-box functions. Journal of Global optimization. 1998;13(4):455–492.
- [19] Mockus J, Tiesis V, Zilinskas A. The application of Bayesian methods for seeking the extremum. Towards Global Optimization. 1978;2(117-129):2.
- [20] Brochu E, Cora VM, De Freitas N. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning; 2010. arXiv preprint, arXiv:1012.2599.
- [21] Snoek J, Larochelle H, Adams RP. Practical Bayesian optimization of machine learning algorithms. In: Advances in neural information processing systems; 2012. p. 2951–2959.
- [22] Heping C, Binbin L, Gravel D, Zhang G, Biao Z. Robot Learning for Complex Manufacturing Process. 2015 IEEE International Conference on Industrial Technology; 2015. Icit 2015 IEEE International Conference on Industrial Technology (ICIT) 17-19 March 2015 Seville, Spain.
- [23] Luna M, Martinez E. A Bayesian Approach to Run-to-Run Optimization of Animal Cell Bioreactors Using Probabilistic Tendency Models. Industrial & Engineering Chemistry Research. 2014;53(44):17252–17266. 1.
- [24] Sterling D, Sterling T, YuMing Z, Heping C. Welding parameter optimization based on Gaussian process regression Bayesian optimization algorithm. 2015 IEEE International Conference on Automation Science and Engineering (CASE). 2015;p. 1490–6. 2015 IEEE International Conference on Automation Science and Engineering (CASE) 24-28 Aug. 2015 Gothenburg, Sweden 0 978-1-4673-8183-3.

- [25] Garnett R. Gaussian Process Regression. Washington University in St. Louis; 2015. Available from: [http://www.cse.wustl.edu/~garnett/cse515t/files/lecture\\_notes/9.pdf](http://www.cse.wustl.edu/~garnett/cse515t/files/lecture_notes/9.pdf).
- [26] Garnett R. Bayesian Optimization. Washington University in St. Louis; 2015. Available from: [http://www.cse.wustl.edu/~garnett/cse515t/files/lecture\\_notes/12.pdf](http://www.cse.wustl.edu/~garnett/cse515t/files/lecture_notes/12.pdf).
- [27] Matérn B. Spatial variation. vol. 36. Springer Science & Business Media; 2013.
- [28] Jones JE. On the Determination of Molecular Fields. II. From the Equation of State of a Gas. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*. 1924;106(738):463–477.
- [29] Maneewongvatana S, Mount DM. Its okay to be skinny, if your friends are fat. In: *Center for Geometric Computing 4th Annual Workshop on Computational Geometry*. vol. 2; 1999. p. 1–8.
- [30] Bentley JL. Multidimensional binary search trees used for associative searching. *Communications of the ACM*. 1975;18(9):509–517.
- [31] Goedecker S. Minima hopping: An efficient search method for the global minimum of the potential energy surface of complex molecular systems. *The Journal of chemical physics*. 2004;120(21):9911–9917.
- [32] Wales DJ, Doye JP. Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms. *The Journal of Physical Chemistry A*. 1997;101(28):5111–5116.
- [33] Storn R, Price K. Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*. 1997;11(4):341–359.
- [34] Mhlbach AH, Vaucher AC, Reiher M. Accelerating Wave Function Convergence in Interactive Quantum Chemical Reactivity Studies; 2015. arXiv preprint, arXiv:1512.02111.
- [35] Botu V, Ramprasad R. A learning scheme to predict atomic forces and accelerate materials simulations; 2015.
- [36] Jain A, Ong SP, Hautier G, Chen W, Richards WD, Dacek S, et al. The Materials Project: A materials genome approach to accelerating materials innovation. *APL Materials*. 2013;1(1):011002.
- [37] Ong SP, Cholia S, Jain A, Brafman M, Gunter D, Ceder G, et al. The Materials Application Programming Interface (API): A simple, flexible and efficient API for materials data based on REpresentational State Transfer (REST) principles. *Computational Materials Science*. 2015;97:209–215.

- [38] Rasmussen CE, Williams CKI. Gaussian processes for machine learning. The MIT Press; 2006. Available from: <http://gaussianprocess.org/gpml/>.

# Vita

Shane Frederic F. Carr

**Degrees**            B.S. in Chemical Engineering, Summa Cum Laude, May 2015  
B.S. in Computer Science, Summa Cum Laude, May 2015  
Minor in Nanoscale Science and Engineering, May 2015  
M.S. in Computer Science, December 2015  
*Washington University in St. Louis*

**Honors and Awards**    Robert N. Varney Prize in Physics, September 2012  
Outstanding Junior in Computer Science, May 2014

**Professional and Honor Societies**    Tau Beta Pi  
Upsilon Pi Epsilon  
Sigma Xi  
American Institute of Chemical Engineers  
Association for Computing Machinery

December 2015



**Machine Learning in Surface Science, Carr, M.S. 2015**