MICROSTRUCTURE SENSITIVE DESIGN

A Dissertation

by

RICHARD ANDREW COUPERTHWAITE

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Raymundo Arróyave
Committee Members,	Ibrahim Karaman
	Ankit Srivastava
	Douglas Allaire
Head of Department,	Ibrahim Karaman

August 2021

Materials Science & Engineering

ABSTRACT

Microstructure sensitive design (MSD) has become an essential component of Integrated Computational Materials Engineering (ICME). There are effectively two components to MSD, multi-scale, microstructure sensitive materials models and design frameworks.

The models used in materials science show much variety, from atomistic Density Functional Theory models to Finite Element mechanical or Phase Field models and beyond. However, all these models attempt to construct Process-Structure-Property-Performance relationships that allow for predicting material microstructure, properties, and performance from either microstructure descriptors or material processing parameters.

Design frameworks can take many forms. However, for the current work, the aim is to use a Reification-Fusion framework. This Framework uses multiple information sources (i.e., models) and fuses them after estimating the model correlation using a process called Reification. This fused model is then used to optimize the material properties in a Bayesian Optimization framework.

One of the more recent developments in the materials science community has been the building of high-throughput experimental methods. These methods have typically relied on thin-film approaches. However, more recently, additive manufacturing methods have started to play a more prominent role. One of the significant challenges is how to use design methods to guide high-throughput experimentation by making batch predictions. A recent batch Bayesian optimization approach has shown promise in this regard.

This work details the construction, testing, and application of a novel design framework. The Batch Reification/Fusion Optimization (BAREFOOT) Framework combines the Reification/fusion and Batch Bayesian Optimization approaches. The Framework is developed in Python and can conduct optimizations using both the individual approaches and the combined approach. The optimization can be paired with computational models or experimental tests as the target source that is being optimized. Furthermore, the results show that all the approaches in the Framework are capable of reducing the time required to optimize the property of interest and provide significant

benefit to the materials design community.

As such, the BAREFOOT Framework has been proven to be a flexible tool that can be used easily in materials design approaches to speed up material design and development. Despite this, much work can be done to bolster the options available in the Framework further.

DEDICATION

To my wife Marina and Daughter Hannah, who made many sacrifices in order for me to complete my degree.

ACKNOWLEDGMENTS

I would like to credit my advisory committee, Dr. Arróyave, Dr. Karaman, Dr. Srivatastava and Dr. Allaire, who gave me the space to learn and make mistakes, but who provided guidance when needed, and correction when I strayed too far from the right path.

I would like to thank the non-faculty members of the DEMS project group, Fathima, Abhilash, Jaylen, and Danial who provided much feedback on the ideas and work conducted for my research.

I would also like to thank the members of Dr. Arróyave's Research Group who helped guide and shape my understanding of many of the topics in this research, and provided valuable feedback on the work that I was doing.

Portions of this research were conducted with the advanced computing resources provided by Texas A&M High Performance Research Computing.

Finally, my wife and daughter, who moved to a whole new country to support this endeavor. Their love and support has been invaluable.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis (or) dissertation committee consisting of Professor Raymundo Arroyave, Professor Ibrahim Karaman and Professor Ankit Srivastava of the Department of Materials Science and Engineering and Professor Douglas Allaire of the J. Mike Walker '66 Department of Mechanical Engineering.

The Representative Volume Element (RVE) model and data used in Sections 2 and 3 were provided by Dr. Ankit Srivastava and Mr. Abhilash Molkeri.

The KKR-CPA Density Functional Theory (DFT) model used in Section 4 was provided by Dr. Prashant Singh (AMES Laboratory) and Dr. Duane Johnson (Iowa State University).

All other work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was supported by a Graduate Research Assistantship from Texas A&M University and a Bursary from Mintek.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION AND LITERATURE REVIEW

Accelerated Materials Design is a goal that both Integrated Computational Materials Engineering (ICME) [1] and initiatives like the Materials Genome Initiative (MGI) [2] are focused on achieving. As stated in the name, the aim is to accelerate materials development and drastically reduce the time to develop novel or improved materials. One of the key concepts proposed as a means to facilitate Accelerated Materials Design is the development of Process-Structure-Property-Performance (PSPP) relationships [3] within materials science. This approach acknowledges that materials can be considered systems where the processing experienced by a material determines the microstructure formed. The microstructure affects the properties and, ultimately, the performance of the material. The full utilization of PSPP relationships in design frameworks leads directly to the concept of microstructure-sensitive design.

Microstructure sensitive design is a broad concept that can encompass many approaches. Some examples of this are the work by Tallman et al. [4] who used crystal plasticity models to determine the properties of regions of the microstructure based on composition and structure, and then using this knowledge, built a reduced-order model for predicting macroscopic mechanical properties. Another example is the work of Parthasarathy et al. [5] who used thermodynamic-based calculations to determine the microstructure at different locations in a complex geometry component and then used this information to build a mechanical model based on the different properties at different locations in the component. These examples show how using knowledge of the microstructure features in the material or component facilitates the building of more robust and accurate predictors of material properties. There appear to be two main focuses of microstructure-sensitive design. The first is to focus on designing the microstructural features directly. The second is to design the processing parameters that will produce a specific microstructure.

However, a distinction must be made between multi-scale modeling and microstructure sensitive design [6]. As shown in Figure 1.1 [3] multi-scale modeling is part of the deductive process of determining properties and performance metrics for materials from given processing or structure

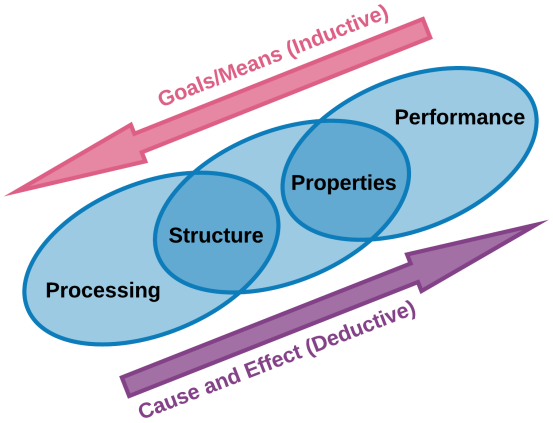


Figure 1.1: Schematic representation of the Process-Structure-Property-Performance paradigm in Materials Science and how it relates to deductive materials modeling and inductive design approaches. Figure adapted from work by Olson [3].

information. However, the concept of multi-scale modeling [7] is more complex, and exists in conjunction with the venn diagram shown in Figure 1.1. In contrast microstructure sensitive design is the inductive process of determining which structure or processing parameters will provide the desired material performance. However, despite ensuring that there is a distinction between these two approaches, they are inextricably linked. It is not possible to affect microstructure-sensitive design without multi-scale models.

The concept of multi-scale modeling acknowledges that materials consist of a complex hierarchy of features at different length scales [7]. Multi-scale modeling encompasses models such as Density Functional Theory (DFT) at the quantum level, up to Finite Element mechanical models that are capable of continuum scale models of material properties and performance. Furthermore, multi-scale modeling also considers the bridging of different length scales to obtain more accurate representations of material properties [7]. The focus on multi-scale modeling has evolved to produce a body of knowledge that contains large numbers of models at different length scales that are capable of predicting many material properties. However, it has also resulted in many models capable of predicting the same material properties. For example, we can predict phase stability

using DFT at atomic scale, or we could use the CALPHAD approach at continuum scale. Or we can use a finite element model to predict the mechanical properties of a material at micro-scale, or we can use one of several empirical models that predict the mechanical properties at continuum scale. This wealth of models is one of the reasons why the Reification approach in particular is so attractive for materials design.

To achieve the design component of microstructure sensitive design, it is necessary to incorporate materials modeling into design frameworks. The field of design optimization is quite large and includes techniques such as Non-linear programming [8], Genetic Algorithms [9], Simulated Annealing [10], Particle Swarm [11], and Bayesian Optimization [12]. In the current work, the focus is on the use of a Bayesian Optimization framework. The design framework in question is the design framework presented by Ghoreishi et al. [13], [14]. The Framework employs a model fusion technique called Reification [15] to combine the predictions of multiple models by estimating the correlation between the models as well as between the individual models and the ground truth. The net result is a fused model that is significantly more accurate than any of the individual models. After the construction of the fused model, the Framework selects the specific information source (i.e., model) to query next as well as the location in the materials design space (i.e., phase constitution) to explore within a Bayesian Optimization (BO) framework.

There are several advantages of using Bayesian Optimization Frameworks, firstly it is possible to consider many distinct sources of uncertainty. And many of the approaches to uncertainty propagation in Materials Design [16]–[20] use Bayesian Approaches in order to account for uncertainty. One of the more common approaches is the method proposed by Kennedy and O'Hagan [21], and this approach is used in the current work to account for uncertainty in some of the surrogate models generated. The second advantage is that Bayesian Optimization frameworks consider the model as a black-box function, which means that the actual models used can be as complex or simple as desired/available. The only consideration is that exceptionally complex models may take too long to compute, and so the optimization process will take a prohibitively long time to complete.

In recent years, the wider materials science community has also explored the use of high-

throughput experiments[22]–[25] or simulations[26], mostly based on DFT calculations, to accelerate the exploration of materials spaces. High throughput experimental methods tend to involve thin-film combinatorial libraries [27]. Although very recently, using additive manufacturing platforms to parallelize the synthesis of alloys [23] has emerged as a potential alternative. While optical and electrical properties are most easily measured in a high-throughput fashion [23], recent approaches have shown that it is possible to rapidly measure other material properties such as composition [24], [25], microstructure [24], [25], hardness [25], and even transformation temperature of shape memory alloy thin films [22]. These high-throughput approaches, while highly advantageous, suffer from the fact that they tend to be open-loop, one-shot approaches. Such approaches do not integrate a principled approach to use the information gained from the parallel exploration of the materials space to decide what to do next after taking the first information-gathering step.

There are numerous proposals for how to approach batch optimization in a Bayesian optimization framework. Some of the more common approaches involve a multi-step look ahead [28], [29] where the batch is created by sequentially adding the predictions from the surrogate model and predicting a new best point. Another approach has been to maximize the variance of the predictions [30], [31], while a third approach attempts to extract multiple peaks from the same acquisition function [32]. A more recent approach by Joy et al. [33] instead calls for the parallel query of the 'black-box' to be modeled at multiple locations to better understand the underlying shape of the objective function (e.g., materials design space) to be emulated or modeled. This approach is simple to understand and implement and addresses one of the fundamental limitations of Bayesian Optimization (BO) approaches: one fundamental assumption in BO is that querying 'black boxes' is costly. Thus it is likely that very little information will be available when starting a BO sequence, which in turn means that the relative smoothness of the objective function and the characteristic length scales connecting it with the input space are not known. Under such data scarcity, it is not rational or practical to attempt to find optimal hyperparameter sets. Since the notion of optimality will always depend on the ground truth data at our disposal at any given time. Under these circumstances, a more rational approach would be to assume that every length scale or hyperparameter

set is possible a priori and work through the BO cycle accordingly.

The integration of high-throughput materials testing and computational modeling is one of the objectives identified in the position paper by McDowell and Kalidindi [34]. The literature contains a relatively significant amount of work on developing optimization frameworks specifically aimed at material optimization. A small selection of these are CAMEO [35] which uses Bayesian Optimization to predict the next-best points to evaluate and first attempts to expand the knowledge of the system (explores the design space) before optimizing for a microstructure specific property. GOLEM is a robust optimization framework [36] where the aim was to ensure that the optimization provides a robust result in the presence of uncertainty in the outputs of the models. The OLYMPUS framework [37] was set up to provide a range of different optimization approaches, assuming that a single optimization approach may not be optimal in all cases. Most of these optimization frameworks are configured for single-objective optimization. However, multi-objective frameworks are available, with Chimera being one of the more prominent options. However, the Chimera framework does not construct the Pareto front explicitly, relying on achievement scalarizing functions (ASFs) to convert the multi-objective into a single objective. There still exists a need for multi-objective optimization that aims to construct the Pareto front explicitly.

Another area for potential development is the use of autonomous experimentation to optimize material properties. Autonomous experimentation is an area of research that has opened up very recently. Part of the reason for this is that integrating computational models and experimental processes is a formidable challenge [35]. However, some approaches have been proposed that demonstrate the integration of optimization and experimentation. The first example is the work by Gongora et al. [38] who coupled a Bayesian Optimization approach with compression testing of components produced by 3D printing. The weakness of this approach is that while the testing was capable of batch production of components for testing, the Bayesian Optimization was not. However, the results showed that the combined experimental and optimization approach could efficiently optimize the components' structure to obtain the desired properties.

Another example is the Noack et al. [39] who considered the evaluation of XRD diffraction

patterns in block copolymers. In this case, the experimental approach was very limited since they had pre-made samples, and the optimization was required to select one of these pre-made samples. The requirement for pre-making the samples for the optimization brings into stark relief where the challenge is for these full-loop approaches. It is relatively straightforward to automate the data capture and analysis of samples. However, automating the production of samples is incredibly challenging, even for thin-film approaches [39] where it is necessary to create composition spreads before the analysis and optimization. This limitation could be answered in the Additive Manufacturing (AM) space since specific AM instruments allow individual element powder feeding to the deposition process [40]. However, this will always be limited by how many powder feeders can be attached to a single machine.

The current work is an effort to fill some of the gaps in the material optimization approaches expanded on above. This is achieved by combining the Batch Bayesian Optimization, and Reification/Fusion approaches into a single framework capable of using each technique separately and integrating with computational or experimental methods to optimize materials. The Framework developed has been named BAREFOOT. Direct integration with almost any computational method is possible (integration with a Finite Element Micromechanical Model and a DFT model has been successful). Additionally, the Framework provides the possibility of integration with online and offline experimental approaches. The Framework testing was as extensive as possible to determine the performance under different conditions and define standard operating parameters. As identified by Häse et al. [41] a single optimization approach is not ideal in all cases, and the results obtained in the current work demonstrate this point. Despite this, the BAREFOOT Framework is a flexible approach to optimization that can be quickly and easily set up for new applications.

The remainder of this document covers the construction of the Framework and what options are available for fine-tuning the optimization. After this, results from testing of the framework performance under different parameter combinations are presented. In the final section on the Framework, the results from applying the Framework to optimize the Bulk Modulus of alloys in a refractory High Entropy Alloy design space are presented, and how the Framework can train the

Framework reduced-order models on the fly. The final section of the current work presents an approach for adding uncertainty in the surrogate modeling of Thermo-Calc™data and demonstrates the building and accuracy of the surrogate models using this approach.

2. DEVELOPMENT AND VALIDATION OF THE BAREFOOT FRAMEWORK CODE

2.1 Framework Approach

2.1.1 Optimization approaches

The Batch Reification/Fusion Optimization (BAREFOOT) Framework is an open-source Python code that is available on Github [42]. This chapter covers the construction and general operation of the Framework code. Chapters 3 and 4 contain more detailed descriptions of the methods involved in the Framework. The Framework uses Bayesian Optimization (BO) approaches, and there are currently three optimization methods available,

1. Reification/Fusion Optimization: This optimization approach uses the method proposed by Thomison and Allaire [15] that uses several reduced-order models to build a fused model of the ground truth experiment or model. This fusion approach uses a sequential BO approach for optimization.

2. Batch Bayesian Optimization: the Batch BO approach follows the method put forward by Joy et al. [33]. The approach uses the uncertainty in the Gaussian Process model hyperparameters to build multiple Gaussian Process models and predict a batch of next-best points to evaluate.

3. Barefoot: the Barefoot approach combines the above two approaches. This approach builds the fused model from the reduced-order models and applies the Batch BO approach to the fused model.

Having all three of these approaches available in the Framework provides a significant degree of flexibility when applying the Framework. To further expand on that flexibility, since BO relies heavily on evaluating acquisition functions, we have implemented several acquisition function approaches. All of these acquisition functions are available for all three optimization approaches.

2.1.2 Acquisition Functions

In Bayesian Optimization, the most commonly used functions appear to be Expected Improvement [12], [43], Probability of Improvement [44], and Upper (or Lower) confidence Bound [45], [46]. The less frequently used ones are Knowledge Gradient[47], Thompson Sampling[48], and the GP Hedge Portfolio approach [49]. Many BO approaches use Greedy acquisition functions in contrast to another acquisition function. The main reason for this is that Greedy Algorithms are very exploitative, which often means the optimization does not perform well. However, we include it as a potential acquisition function in this work since the Batch approach is naturally exploratory. This contrast balances the performance of the Framework when using the Greedy Acquisition function. The following discussion provides a brief description of each of the acquisition functions.

2.1.2.1 Knowledge Gradient:

The Knowledge Gradient (KG) acquisition function is defined according to the work by Frazier et al. [47]. In summary, the method requires a set of M distinct alternative points in the fused model input space. At these points we evaluate the posterior predictive mean, μ_x^n, and variance, $(\sigma_x^n)^2$ of the fused model. The n superscript denotes the iteration number. KG is then defined as:

$$\nu^{KG} = \max_{x^n \in \{1,\dots,M\}} \mathbb{E}_n \left[\left(\max_{x'} \mu_{x'}^{n+1} \right) - \left(\max_{x'} \mu_{x'}^{n} \right) \right] \tag{2.1}$$

where $\mathbb{E}_n$ is the conditional expected value given the knowledge at iteration n and μ_x^{n+1} is the predicted mean at step $n+1$ using a Bayesian look-ahead approach. The Bayesian look ahead approach estimates μ_x^{n+1} conditional on μ_x^n and $(\sigma_x^n)^2$. This is acheived by defining the precision of the posterior predictive distribution as $\beta_x^n = (\sigma_x^n)^{-2}$. The conditional variance for the look ahead step is then defined by:

$$\tilde{\sigma}\left(\beta_x^n\right) = \sqrt{\left(\beta_x^n\right)^{-1} - \left(\beta_x^n + \beta^\epsilon\right)^{-1}} \tag{2.2}$$

where β^{ϵ} is the measurement precision and is generally assumed to be constant. Using the definition above, the look ahead mean is defined as:

$$\mu_x^{n+1} = \mu^n + \tilde{\sigma}\left(\beta_x^n\right) Z e_x \tag{2.3}$$

where Z is the standard normal distribution and e_x is a vector in $\mathbb{R}^M$ with all components zero except for component x. For a complete description of the method and algorithm implemented in the current work, refer to the work by Frazier et al. [47].

2.1.2.2 Expected Improvement

The Expected Improvement algorithm evaluates the Expected Improvement of all test points where the Improvement is the magnitude of the increase in the predicted mean over the previous known maximum at the test point ($I(x) = \{0, f_{t+1}(x) - f(x^+)\}$). The Improvement is maintained as a positive value by specifying that if the predicted mean is less than the previous known maximum, the Improvement is set to 0. The analytical solution for Normal distributions as defined by Močkus et al. [12] and Jones et al. [43],

$$\mathrm{EI}(x) = \begin{cases} d\Phi(d/\sigma(x) + \sigma(x)\phi(d/\sigma(x)) & \text{if } \sigma(x) > 0 \\ 0 & \text{if } \sigma(x) = 0 \end{cases} \tag{2.4}$$

where $d = \mu(x) - \mu^+ - \xi$, Φ is the standard Normal cumulative distribution function (CDF) and ϕ is the standard Normal probability distribution function (PDF). And ξ is a parameter to avoid the acquisition function from being too heavily exploitative

2.1.2.3 Probability of Improvement

The Probability of Improvement acquisition function is closely related to the Expected Improvement acquisition function. However, this approach aims to estimate the probability that a test point has a higher value than the previously known maximum. One of the disadvantages of this technique is that it is highly exploitative and can get stuck exploiting a local maximum for a long time before exploring further. Therefore, ξ is included as a parameter to force the acquisition

function to be more exploratory. A higher value of ξ will lead to more exploration, while a lower value will result in more exploitation. The equation for the Probability of Improvement calculation is,

$$\text{PI}(x) = P((f(x) \geq \mu^+ + \xi) = \Phi\left(\frac{\mu(x) - \mu^+ - \xi}{\sigma(x)}\right) \tag{2.5}$$

where Φ is the standard Normal cumulative distribution function (CDF).

2.1.2.4 Upper Confidence Bound

Cox and John [45], [46] defined an acquisition function for minimization problems called Lower Confidence Bound. By changing a sign, it is possible to configure the function for maximization problems, and in this case, the acquisition function is referred to as the Upper Confidence Bound,

$$\text{UCB}(x) = \mu(x) + \sqrt{\nu\beta_t}\sigma(x) \tag{2.6}$$

where ν is a constant and β_t is a parameter that varies with the progression of the iteration. While there appear to be several ways to define the parameter, the value in this work is $\beta_t = log(k(t^2)\pi^2)/6$ where 'k' is the number of test points, and 't' is the iteration.

2.1.2.5 Thompson Sampling

Thompson Sampling is a method that was proposed by Thompson in 1933 [48]. Despite this, the acquisition function has only really found favor more recently. The approach is relatively simple, and Algorithm 1 shows the implementation. The basic principle is to take the posterior predictive distribution of the GP model and sample out of the Normal distribution at a selection of test points. The test point with the highest sampled value is the next-best point to evaluate.

2.1.2.6 Greedy Optimization

Greedy optimization is often the baseline when comparing other acquisition functions. In this respect, it is often not used in optimization approaches since it tends to be almost exclusively ex-

Algorithm 1 Thompson Sampling Acquistion Function

for t=1,2,... do **do**
  # Sample Posterior Predictive Fused Model
  for k=1,...,K do **do**
    $\hat{\theta}_k \sim \mathcal{N}(\mu_k, \sigma_k)$
  end for
  # Select action
  $x_t \leftarrow \text{argmax}_k \hat{\theta}_k$
end for

ploitative. However, with the Barefoot approach, this is tempered slightly by the use of multiple hyperparameters. Therefore, we maintain the Greedy Optimization as one of the acquisition function options. This acquisition function merely defines the next best point as the point with the largest predicted mean,

$$x_t = \text{argmax}_k \mu_k \tag{2.7}$$

2.1.2.7 GP-Hedge Portfolio Optimization

The GP-Hedge Portfolio Optimization approach assumes that no single acquisition function will be best suited to all optimization problems. Moreover, the optimal acquisition function could change during the optimization. Therefore, the GP-Hedge approach evaluates all possible acquisition functions at all steps and obtains the next-best predictions from each. It then chooses which set of next-best points to use by considering a ratio of the Gains of each acquisition function. Algorithm 2 shows the implementation of the approach.

One of the significant challenges with using the GP-Hedge approach is that it requires calculating all the acquisition functions. These evaluations can take a significant amount of time. Currently, the Framework does not conduct these evaluations in parallel. Therefore, one approach to alleviate the time cost is implementing a parallel evaluation of the acquisition functions. However, since each acquisition function evaluation uses parallel processing when doing the calculations, it will require some modification to implement the parallel evaluation in the GP-Hedge method.

Algorithm 2 GP-Hedge Portfolio Optimization Approach

for t=1,2,... do **do**
  Evaluate all acquisition functions: $x_t^i = \text{argmax}_x u_i\left(X|\mathcal{D}_{1:t-1}\right)$
  Select $x_t = x_t^j$ with max probability: $p_t(j) = exp\left(\eta g_{t-1}^j\right) / \sum_{l=1}^k exp\left(\eta g_{t-1}^l\right)$
  Sample objective function: $y_t = f(x_t) + \varepsilon_t$
  Augment the data: $\mathcal{D}_{1:t} = \{\mathcal{D}_{1:t-1}, (x_t, y_t)\}$
  Calculate rewards $r_t^i = \mu(x_t^i)$ from updated GP
  Update gains: $g_t^i = g_{t-1}^i + r_t^i$
end for

2.1.2.8 Multi-Objective Acquisition Function

All the acquisition function approaches above are for single objective optimization. The Framework is also equipped with the ability to do multi-objective optimizations. In the case of a multi-objective optimization, the Expected Hypervolume Improvement (EHVI) [50], [51] acquisition function is selected automatically. The formula for EHVI is defined as,

$$\mathbb{E}\left[\mathcal{H}_{\mathcal{I}}(\mathbf{y})\right] = \int_U \mathbb{P}(\mathbf{y} \prec \mathbf{y}')\mathrm{d}\mathbf{y}' \tag{2.8}$$

where $\mathbf{y}$ is a vector of all objective values, $\mathbb{P}(\mathbf{y} \prec \mathbf{y}')$ is the probability that $\mathbf{y}$ is dominated by $\mathbf{y}'$ and U is the dominated hypervolume. For GP models with normal distributions there is a closed form solution for the probability ($\mathbb{P}$). This solution is,

$$\mathbb{P}(\mathbf{y} \prec \mathbf{y}') = \prod_{i=1}^{m} \Phi\left(\frac{y_i' - \mu_i}{\sigma_i}\right) \tag{2.9}$$

where m is the number of objectives, μ, and σ are the mean and standard deviation of the Normal distribution, and Φ is the standard Normal CDF.

2.2 Framework Components

2.2.1 Gaussian Process Model Implementation

The barefoot Framework currently uses Gaussian Process models as the surrogate models for the Bayesian Optimization. The motivation for building the Framework using Gaussian Process

models was that they are the most common class of surrogate models used in Bayesian Optimization. However, the framework code is not limited to using only Gaussian Process models. The only requirement from the Framework is that the surrogate model must provide a covariance matrix and a mean as the output. This requirement exists since all the acquisition functions implemented in the Framework require the covariance matrix, the variance (diagonal of the covariance matrix), or the standard deviation (square root of the covariance matrix diagonal). Listing 2.1 shows the basic structure of the GP model class used in the Framework.

While we do not show the details of the code for the GP model class in Listing 2.1, the current implementation of the Framework utilizes the George.py module for the Gaussian Process models. This choice has an advantage in that the George.py model objects are picklable, enabling the use of this module with Python's multiprocessing module. The GP model class serves two primary purposes. The first purpose is to reduce unnecessary code by creating a single line of code for constructing the GP models in the main framework code. The second purpose for the GP model class is to assist with updating or training the Gaussian Process model. While the comments in Listing 2.1 indicate what each class method aims to achieve, it is helpful to explain these in a little bit more detail.

To aid in the description of the methods, it is first necessary to provide a basic description of a Gaussian process model. There are numerous sources in literature that provide the derivation of Gaussian Process models, however, for the purposes of this work, we will simply state that a Gaussian Process is defined as,

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}), C(\boldsymbol{x}_i, \boldsymbol{x}_j)), \tag{2.10}$$

where μ is the mean function and $C(\cdot, \cdot)$ is the covariance function. Many different covariance functions exist. In the BAREFOOT Framework, it is possible to use three different covariance functions. These are the Squared Exponential function and two forms of the Matérn class of covariance functions, namely where $\nu = 3/2$ and $\nu = 5/2$. The Squared Exponential covariance function is defined for 'h' dimensions by,

$$C(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \, exp\left(-\frac{1}{2}\sum_{h=1}^{n}, \left[\frac{(x_{i,h} - x_{j,h})}{l_h}\right]^2\right). \tag{2.11}$$

The general Matérn class of covariance functions is defined by Equation 3.4. However, there are closed form solutions for values of $\nu = 3/2$ and $\nu = 5/2$ that are commonly used as covariance functions. These two covariance functions are shown in Equations 2.13 and 2.14.

$$C(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \left[\frac{2^{1-v}}{\Gamma(v)}\left(\frac{\sqrt{2v}(x_i - x_j)}{l}\right)^v K_v\left(\frac{\sqrt{2v}(x_i - x_j)}{l}\right)\right]. \tag{2.12}$$

$$C(\mathbf{x}_{i,h}, \mathbf{x}_j) = \sigma_f^2 \sum_{h=1}^{n}\left(1 + \frac{\sqrt{3}(x_{i,h} - x_{j,h})}{l_h}\right) exp\left(-\frac{\sqrt{3}(x_{i,h} - x_{j,h})}{l_h}\right). \tag{2.13}$$

$$C(\mathbf{x}_{i,h}, \mathbf{x}_j) = \sigma_f^2 \sum_{h=1}^{n}\left(1 + \frac{\sqrt{5}(x_{i,h} - x_{j,h})}{l_h} + \frac{5(x_{i,h} - x_{j,h})^2}{3l_h^2}\right) \times exp\left(-\frac{\sqrt{5}(x_{i,h} - x_{j,h})}{l_h}\right). \tag{2.14}$$

In all the covariance functions, l_h is the characteristic length scale, σ_f^2 is the signal variance, and n is the number of dimensions. The characteristic length scale determines the smoothness of the GP model. The signal variance is typically defined as the range of possible objective values that the GP model needs to fit. These are two of the main parameters that need to be provided to the GP model class when constructing the Gaussian Process and are the input parameters (`l_param` and `sigma_f`) in the `__init__` function call shown in Listing 2.1. The other parameter that is required is the noise variance (`sigma_n`) which is a parameter that is commonly referred to as the nugget and serves the purpose of stabilizing the calculation of the inverse covariance matrix.

Several more covariance functions can eventually be implemented in the Framework. However, since these three covariance functions are all commonly used, and all have the same number of hyperparameters, it was chosen only to implement these covariance functions.

```
class gp_model:
   def __init__(self, x_train, y_train, l_param, sigma_f, sigma_n, n_dim,
       kern, mean=0):
      # Initialization Code
   def create_kernel(self):
      # Sets up the kernels according to the process defined in the George.py
         module
   def create_gp(self):
      # Create and train the GP model object
   def predict_cov(self, x_pred):
      # Obtain the mean and full covariance matrix for the test points
   def predict_var(self, x_pred):
      # Obtain the mean and diagonal of the covariance matrix for the test
         points
   def update(self, new_x_data, new_y_data, new_y_err, err_per_point):
      # Add the new data to the GP and retrain the GP model object
   def log_likelihood(self):
      # Return the log likelihood value for the training data
   def get_hyper_params(self):
      # Obtain the hyperparameter values from the GP object
   def hp_optimize(self, meth="L-BFGS-B", update=False):
      # Algorithm to optimize the GP hyperparameters using a gradient descent
         approach
```

Listing 2.1: Outline of the Gaussian Process wrapper function

One of the additional reasons for using the wrapper function shown in Listing 2.1 is that the implementation of the covariance functions in the George.py module is not identical to the conventional formulation shown in Equations 2.11 to 2.14. In the George.py module, the characteristic length scale (l_h) parameter and the signal variance (σ_f) are not squared. This provides a potential challenge for comparing results with different implementations. The wrapper function also corrects these parameters before using them to ensure that the parameter implementation is consistent with other modules.

2.2.2 Reification/Fusion Implemenation

The Reification/fusion approach is explained in detail in the work by Thomison and Allaire. However, we present a summary here. The Reification/fusion approach relies on reduced-order models, typically simple empirical models with some predictive capability. These reduced-order models could differ significantly from the Ground Truth. Usually, an experimental evaluation or

a highly accurate finite element model is the Ground Truth. No matter what the Ground Truth is, it is generally significantly more expensive to evaluate than the reduced-order models. Therefore the Reification/fusion approach aims to create a fused model using the data from the reduced-order models to be a more accurate representation of the Ground Truth. In order to achieve this, we need to quantify the relative accuracy of each of the reduced-order models.

The approach for defining the accuracy of the models is shown in Figure 2.1. As can be seen in part (a) of Figure 2.1, the Truth Function has been evaluated at four different points, while the reduced-order model is evaluated at six. In the next stage, we fit a surrogate model to the reduced-order model and then calculate the discrepancy (δ_i) at each of the evaluated points from the Truth Function (Figure 2.1 (c)). A separate surrogate model fits these discrepancy values to predict the discrepancy over the entire domain. We add the predicted discrepancy to the standard deviation of the reduced-order surrogate model. The resulting uncertainty bounds ($\sigma_{GP} + \sigma_d$) from the total uncertainty encompasses all of the Truth Function points (Figure 2.1 (c)).

Using this information, we define the total uncertainty of the model as a combination of the standard deviation calculated for the Gaussian Process model prediction (σ^i_{GP}) and the discrepancy as calculated from the difference between the information source and the Truth Function (σ^i_d). Therefore,

$$\sigma^i = \sigma^i_{GP} + \sigma^i_d \tag{2.15}$$

where the superscript refers to the reduced-order model. This total uncertainty is used as the standard deviation of the models in the Reification and Fusion approach.

Under the assumption that two models need to be fused (and without a loss of generalizability), the model fusion approach aims to generate a fused model that can be represented by the equation:

$$y = k_1(x^*)f_1(x^*) + k_2(x^*)f_2(x^*) \tag{2.16}$$

where $k_1(x^*)$ and $k_2(x^*)$ are real-valued scalar quantities subject to $k_1(x^*) + k_2(x^*) = 1$. While the two models, $f_1(x)$ and $f_2(x)$, are assumed to estimate the quantity of interest (y) with some

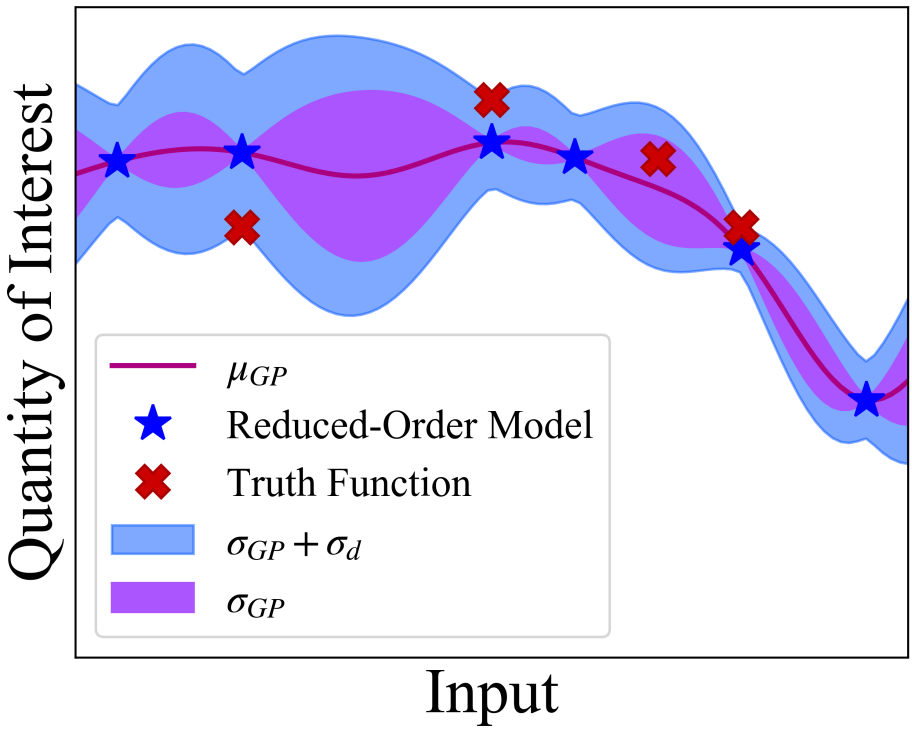


Figure 2.1: Schematic representation of how the accuracy of the models are evaluated and quantified

total uncertainty (δ_i),

$$y = f_1(x) = \bar{f}_1(x) + \delta_1(x), \tag{2.17}$$

$$y = f_2(x) = \bar{f}_2(x) + \delta_2(x), \tag{2.18}$$

where $\bar{f}_1(x)$ is the mean prediction and the model uncertainties $\delta_i(x)$ are assumed to be normally distributed with, $\delta_1(x) \sim \mathcal{N}(0, \sigma_1^2)$ and $\delta_2(x) \sim \mathcal{N}(0, \sigma_2^2)$. And by using this assumption it is possible to use the approach outlined by Winkler [52] to define the fused mean,

$$\mathbb{E}[y] = \frac{(\sigma_2^2 - \bar{\rho}\sigma_1\sigma_2)\bar{f}_1(x^*) + (\sigma_1^2 - \bar{\rho}\sigma_1\sigma_2)\bar{f}_2(x^*)}{\sigma_1^2 + \sigma_2^2 - 2\bar{\rho}\sigma_1\sigma_2} \tag{2.19}$$

and variance

$$\text{Var}(y) = \frac{(1-\bar{\rho}^2)\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2-2\bar{\rho}\sigma_1\sigma_2} \tag{2.20}$$

However, we require the correlation coefficient to use this approach. Since the correlation between two models is generally not known, we use an approach called Reification [15] to estimate the correlation between the models. First, we "reify" model 1 which means that we assume that model 1 is the Truth Function. And we can calculate the Pearson correlation coefficient (ρ) using:

$$\rho_1(x^*) = \frac{\sigma_1^2}{\sigma_1\sigma_2} = \frac{\sigma_1}{\sqrt{(\bar{f}_1(x^*)-\bar{f}_2(x^*))^2+\sigma_1^2}}, \tag{2.21}$$

where the subscript on the coefficient indicates which model has been reified. Repeating this for each pair of models allows us to calculate the average correlation between the models,

$$\bar{\rho}(\boldsymbol{x}^*) = \frac{\sigma_2^2}{\sigma_1^2+\sigma_2^2}\rho_1(\boldsymbol{x}^*) + \frac{\sigma_1^2}{\sigma_1^2+\sigma_2^2}\rho_2(\boldsymbol{x}^*). \tag{2.22}$$

We use this average correlation when calculating the fused mean and variance.

The function used to calculate the fused mean and variance is shown in Listing 2.3. Initially, this function used a series of nested loops. This was not a very efficient approach. Changing to the code shown in Listing 2.3 it was possible to decrease the Framework's iteration time by about 40%. The previously published results used the old approach for calculating the fused mean and variance, and even then, the Framework was able to optimize faster than a conventional Bayesian approach. With this change, those results would be even more significant. The critical parameter calculated is the `alpha` value, which is the inverse of the correlation matrix. Using this matrix, we can calculate the fused mean and variance with a simple matrix multiplication for the mean and by taking the sum of the unit-wise inversion of the matrix.

2.2.3 Batch Optimization

The Batch Bayesian Optimization approach implemented in the Framework is the method proposed by Joy et al. [33]. This basis of the approach is the premise that it is rarely possible to

```
class model_reification():
   def __init__(self, x_train, y_train, l_param, sigma_f, sigma_n, model_mean,
             model_std, l_param_err, sigma_f_err, sigma_n_err,
             x_true, y_true, num_models, num_dim, kernel):
      # initialize the model_reification object
   def create_gps(self):
      # Construct the GP models for each of the reduced order models
   def create_error_models(self):
      # Construct GP models for the discprency of each reduced order model
   def create_fused_GP(self, x_test, l_param, sigma_f, sigma_n, kernel):
      # Use the Reification function and calculate the fused mean and
      # variance then construct a GP model
   def update_GP(self, new_x, new_y, model_index):
      # Update the data in one of the reduced order GP models
   def update_truth(self, new_x, new_y):
      # Update the Ground Truth Data used to calculate the discrepancy
      # models and recalculate the discrepancy for each model
   def predict_low_order(self, x_predict, index):
      # Provide mean and variance predictions from one of the
      # reduced order GP models
   def predict_fused_GP(self, x_predict):
      # Provide the mean and covariance matrix (diagonal values only)
      # for the fused GP model
```

Listing 2.2: Outline of the Reification/fusion Class used in the BAREFOOT Framework

```
def reification(y,sig):
   yM = np.tile(y, (len(y),1,1)).transpose()
   sigM = np.tile(sig, (len(y),1,1)).transpose()
   unoM = np.tile(np.diag(np.ones(len(y))), (3,1,1))
   zeroM = np.abs(unoM-1)
   yMT = np.transpose(yM, (0,2,1))
   sigMT = np.transpose(sigM, (0,2,1))

   factor1 = (sigM/(sigM+sigMT))*np.sqrt(sigM)/np.sqrt((yM-yMT)**2 + sigM)
   factor2 = (sigMT/(sigM+sigMT))*np.sqrt(sigMT)/np.sqrt((yMT-yM)**2 + sigMT)
   factor3 = zeroM*(np.sqrt(sigM*sigMT))
   factor4 = unoM*sigM

   alpha = np.linalg.pinv((factor1 + factor2)*factor3 + factor4)
   w = (np.sum(alpha,1)/np.sum(alpha))
   mean = np.sum(w@y, axis = 0)
   var = 1/np.sum(alpha,(1,2))
   return mean, var
```

Listing 2.3: The Reification function used in the BAREFOOT Framework

determine the hyperparameters for the Gaussian Process surrogate models used in the optimization. Therefore, it is better to sample potential hyperparameters and generate many instances of the surrogate models. By doing this, the process never makes any assumption on the shape of the black-box function. The left subfigure of Figure 2.2 demonstrates how sampling the hyperparameter space leads to different model shapes. Using a Bayesian Optimization approach, we can then define the next best point to query by evaluating an acquisition function for each of these surrogate models, which is demonstrated in the right sub-figure in Figure 2.2. This process provides us with the next-best prediction for each set of hyperparameters used. We then cluster these results based on the batch size of the process. In other words, if we want a batch of 10 predictions, then we cluster the results from the acquisition function into 10 batches. To do this, we use the k-medoids algorithm.

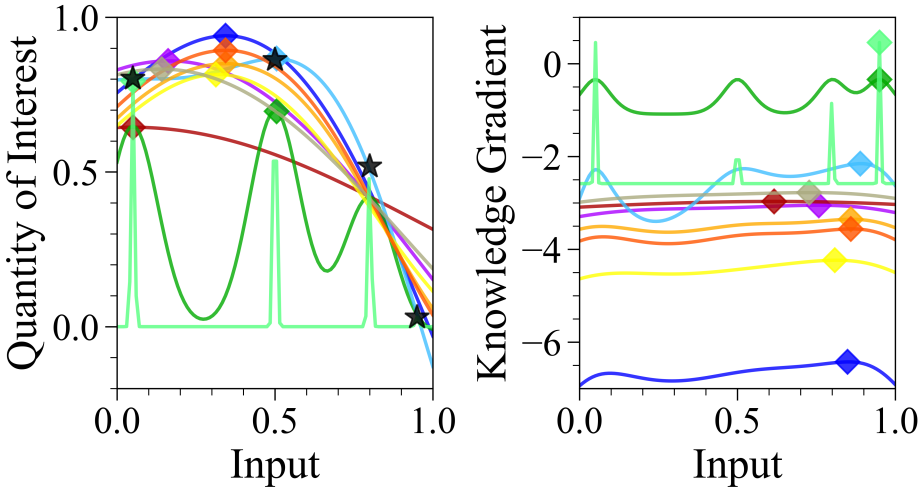


Figure 2.2: Demonstration of the underlying principle of the Batch Bayesian Optimization approach showing, on the left, the different shapes achieved by using different hyperparameters for the surrogate models. The right figure shows how these different surrogate models translate into the acquisition function space (in this case using the Knowledge Gradient acquisition function)

We choose to use the k-medoids algorithm instead of the more commonly used k-means since k-medoids will choose one of the existing points as the central point of the cluster, rather than

calculating the exact center of the cluster, which k-means will do. The primary motivation behind this is that we have already tested all the existing points for possible violation of constraints. So we know that it is possible to evaluate any of these points. Using the centroid defined by k-means would require extra steps to determine whether the centroid violates constraints and could cause the batch of points to be incomplete if any of the centroids violate the constraints.

The framework code provides the options of changing four parameters related to the Batch Optimization. These are the batch size, the number of hyperparameter sets, and the upper and lower bounds of the hyperparameter sampling space.

The batch size parameter is self-explanatory. While the number of hyperparameter sets is straightforward to understand, it will be helpful to define how the Framework chooses the sets. First, we sample linearly for every order of magnitude in the range between the upper and lower bounds of the hyperparameters. Breaking down the range into smaller sections ensures that we sample all possible ranges. After defining these linearly spaced parameters, we randomly combine them to construct the hyperparameter sets.

2.3 Framework Implementation

A schematic showing how the framework operations are structure is shown in Figure 2.3. In this overview, the background colors used indicate different sections of the code. The green section involves queries to the true models and the construction of the GPs. The light blue section is for the batch optimization related to the reduced-order models, which the light purple section is for the batch optimization related to the Ground Truth. Later sections will provide more detail on the distinction between these different calculations.

This flow diagram is the overview of the Barefoot approach. However, as already noted, the Framework has been developed to enable three different optimization approaches. These will be referred to as the Batch and Reification approaches. The Batch approach is effectively only the purple section of the flow diagram in Figure 2.3, with the fused model replaced by a single GP surrogate model of the Ground Truth Model. In contrast, the Reification approach is almost identical to Figure 2.3 with a batch size of 1, and instead of clustering the results, the algorithm

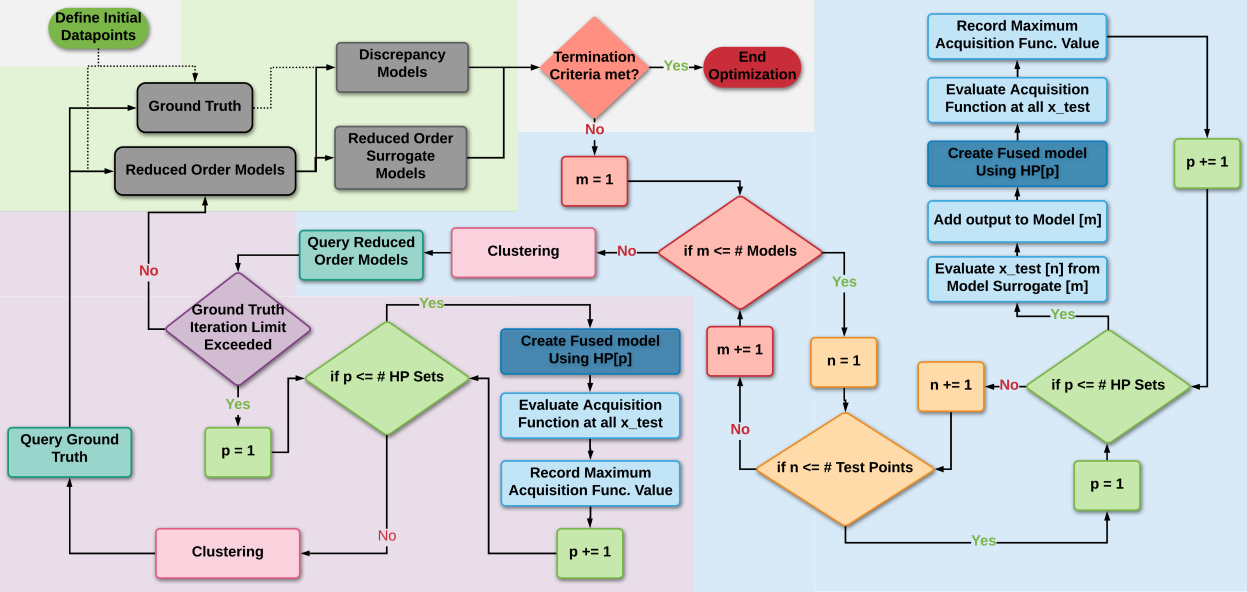


Figure 2.3: A schematic representation of the BAREFOOT Framework.

selects the results with the maximum acquisition function value as the next-best point.

2.3.1 Python Class Structure

We chose to implement the BAREFOOT Framework as a Python Class. This ensures that the code is as modular as possible. Having this modular structure is a considerable advantage in terms of upgrading the functionality of the Framework. An outline of the structure for the Python Class is shown in Listing 2.4. There are two stages to the initialization of the Framework.

The first is the parameters provided to the class `__init__` function. The parameters used at the initial setup determine the general framework parameters. In contrast, the `initialize parameters` function handles the setup of parameters more directly related to the actual calculations in the Framework. As can be observed, almost all the parameters have default values. These are parameters determined as potential best standard values from extensive testing of the Framework. The results from this testing can be found in Chapters 3 and 4.

It will be helpful to define at least some of the input parameters to get an idea of how the input parameters determine the Framework's functioning. The parameters used when initializing the Framework are explained below.

```
class barefoot():
   def __init__(self, ROMModelList=[], TruthModel=[], calcInitData=True,
             initDataPathorNum=[], multiNode=0, workingDir=".",
             calculationName="Calculation", nDim=1, input_resolution=5,
             restore_calc=False, updateROMafterTM=False,
             externalTM=False, acquisitionFunc="KG", A=[], b=[],
             Aeq=[], beq=[], lb=[], ub=[], func=[], keepSubRunning=True,
             verbose=False, sampleScheme="LHS", tmSampleOpt="Greedy",
             logname="BAREFOOT"):

   def initialize_parameters(self, modelParam, covFunc="M32", iterLimit=100,
                        sampleCount=50, hpCount=100, batchSize=5,
                        tmIter=1e6, totalBudget=1e16, tmBudget=1e16,
                        upperBound=1, lowBound=0.0001, fusedPoints=5):

   def run_optimization(self):
      # This function runs the optimization
```

Listing 2.4: The barefoot class public functions

1. calcInitData: The initial data can be calculated from the models directly or uploaded from a file. This parameter toggles which option is used.

2. initDataPathorNum: This parameter works in conjunction with the previous one. If the Framework calculates the initial data, this must be a list of integers that indicate how many times to evaluate each model. If the initial data has already been calculated, this parameter is the path string that points to the initial data file.

3. multiNode: The Framework operates in two modes. More details will be provided later. However, the simple description is that multiprocessing in the Python standard library is limited to a single node on a high-performance computing cluster. The Framework submits and runs subprocesses to avoid this limitation, and this parameter determines how many subprocesses to use.

4. restore_calc: There will be times when it is desirable to continue the calculations after the Framework has terminated or when the Framework encounters an error. We will want to be able to restore the Framework with as little data loss as possible. As such, the Framework

saves its state at the end of each iteration. This parameter allows the calculation to be restored from the last saved state.

5. updateROMafterTM: If the reduced-order models can be trained when there is additional data available from the Ground Truth, this parameter will enable this functionality. It will require the models to be formulated in a specific manner, but this will be discussed in more detail later.

6. externalTM: Integrating the Framework with an experimental Ground Truth, or a computational Ground Truth that is not available on the same resources as the Framework, requires the next-best points to be output to a file. Setting this parameter to True enables this feature and ensures that the Framework shuts down to save computational resources. The Framework is then restarted using the restore_calc functionality explained already when the Truth Model evaluations are completed.

7. acquisitionFunc & tmacqFunc: Each optimization (reduced-order model or Ground Truth) in the Framework uses an acquisition function, and it is possible to toggle these two separately.

8. A, b, Aeq, beq, ub, lb, func: These are the constraints on the input space. They follow the standard definitions of inequality constraints (A, b), equality constraints (Aeq, beq), input bounds (ub, lb), and custom function constraints (func).

9. keepSubRunning: The subprocesses generated when running on multiple nodes are used while calculating the acquisition functions, but not while querying the actual functions. As a result, there is a chance that there will be a significant waste of resources if these subprocesses are left running while doing long-running Ground Truth Calculations. Therefore, it is possible to close these subprocesses while querying the Ground Truth to avoid this.

The parameters set in the second stage of the initialization are explained in more detail below. As seen, these set more of the parameters for the actual calculation.

1. covFunc: All the GPs in the Framework need to use a covariance function. This parameter defines which covariance function is used. The current options available are the Squared Exponential ("SE"), Matérn($\nu = 3/2$) ("M32"), and Matérn($\nu = 5/2$) ("M52") covariance functions.

2. iterLimit & totalBudget: There are two methods to providing termination criteria for the Framework. The first is to set an iteration limit, and the second is to set a budget limit.

3. sampleCount: For the reduced-order model optimization step, it is necessary to define how many sample points to use. These are the number of points in the design space at which the acquisition function is evaluated. The standard method for finding these points is Latin Hypercube sampling. However, this can be changed to grid-based or custom sampling approaches.

4. hpCount: This parameter determines the number of parameter sets. Together with the sampleCount parameter, these two values will determine how many calculations are required for each reduced-order model evaluation step.

5. tmIter & tmBudget: As with the termination criteria, there are both iteration and cost-based limits for when the Framework will evaluate the Ground Truth.

6. upperBound & lowBound: These two parameters determine the upper and lower bounds of the hyperparameters. Since the inputs are transformed to the unit hypercube, the upper bound should not be increased significantly above 1. While in our testings of the Framework, we have found that a lower bound of 0.001-0.0001 is usually a reasonable lower limit.

7. fusedPoints: When constructing the fused model, we use a LHS of the design space and evaluate the fused mean and variance at each of these points. This parameter determines how fine the sampling of the design space is and should be adjusted lower as the dimensionality of the problem increases.

```
def model(x_input):
    # Sample of a 2D model
    x[:,0] = x[:,0]*max_input1-min_input1
    x[:,1] = x[:,1]*max_input2-min_input2
    # Query Model
    return model_output #1D vector
```

Listing 2.5: Sample model function demonstrating the inputs for a two-dimensional problem.

2.3.2 Models and Model Parameters

How the different models should be added to the Framework and the structure of the model parameter dictionary deserve a much more substantial description since both need to be provided in specific ways. We will discuss the models first.

The Framework sends a matrix of input values to each model, both reduced order and Ground Truth. As such, the functions for each model need to accept a single matrix input (Listing 2.5). As can be seen, the inputs provided are values from the unit hypercube, so it is necessary to convert them to the values required by the model. The function's output needs to be a single dimension vector with results corresponding to the index of the inputs. The reduced-order models and the Ground Truth model are added to the Framework separately. The reduced-order models are added as elements in a Python list, while the Ground Truth model is added on its own.

Having defined how the models need to be structure, we can focus on the GP model parameters input for the Framework. This input is a Python dictionary that has several required values. This dictionary provides the GP hyperparameters for all the reduced-order model GPs and the discrepancy model GPs for the Reification object. All the parameters are shown in Listing 2.6. There are three sets of hyperparameters for each of the sets of GPs (designated with subscripts "l", "sf", and "sn"), which are the characteristic length scale, the signal variance, and the noise variance. A set of parameters hold the mean and standard deviation for each of the reduced-order models. The mean and variance convert the model output to the standard Normal distribution, which can improve performance. If the mean and variance are unknown, they can be set to zero mean and

```
# Sample model parameter dictionary for an optimization
# involving three reduced order models in a 2D system
modelParam = {'model_l': [[0.1,0.1], [0.1,0.1], [0.1,0.1]],
           'model_sf': [1,1,1],
           'model_sn': [0.01, 0.01, 0.01],
           'means': [0,0,0],
           'std': [1,1,1],
           'err_l': [[0.1,0.1], [0.1,0.1], [0.1,0.1]],
           'err_sf': [1,1,1],
           'err_sn': [0.1, 0.1, 0.1],
           'costs': [0.9, 1.1, 5, 5000]}
```

Listing 2.6: Example model parameter dictionary required by the BAREFOOT framework.

unit standard deviation. All of these parameters require a list with an entry for each reduced-order model. The final entry in the dictionary is a list of the costs of each model. This information is used in two ways; firstly, the acquisition function evaluations are adjusted by the cost of each model to take advantage of cheaper models. The second use is for calculating the total cost of the process.

As already mentioned, this is a required input. It is also one of the only parameters that do not have a default value. If the hyperparameters for each of the reduced-order models are known, then they can be used. However, the example shown in Listing 2.6 shows a reasonable set of values to use if the information for the reduced-order models is not known. Several of the tests conducted on the Framework have used these default values for the GP hyperparameters, and the Framework's performance has still been significantly better than a conventional sequential Bayesian Optimization. One important aspect to note here is that the batch optimization approach of considering a distribution of hyperparameters only applies to the Fused GP, where these parameters apply to the model GPs.

2.3.3 Single Node vs Multi-Node Approaches

After discussing the Framework's general structure and required inputs, it is necessary to expand on how the optimization is applied in the single node and multi-node optimization approaches. The Framework uses the `concurrent.futures` approach from the Python Standard Library for parallel processing of the calculations. This approach has a significant limitation in that

it is only possible to utilize the cores available on a single node in a high-performance computing cluster. Running the code on a single node works admirably when the problem's dimensionality is low. Or when the number of samples, hyperparameter sets, and reduced-order models is small. The advantage of this approach is that it only requires the user to set up a single batch file to run the code.

However, if the calculation takes too long, it was considered worthwhile to expand the calculations to multiple nodes. The approach implemented in the Framework currently is a custom approach. While there are ongoing efforts to shift this aspect of the Framework over to a module such as "Ray" this has not been successfully implemented yet. However, despite this, there are some advantages to the approach implemented in the Framework. Currently, the subprocess jobs stay active until closed, which means that once the subprocesses are started, there is no longer a need to wait for processes to start. In the initial testing of submitting each job as needed, the result was some unacceptable delays while waiting for jobs to start. The downside of this approach is that the user must provide the Framework with the required batch files for submitting the subprocess jobs. This is relatively straightforward, and an example of how this should be achieved is shown in Listing 2.7.

In this example, the batch file is for a high-performance cluster using the LSF batch system. The information comes as two separate shell files. This was a shortcoming of not knowing how to submit batch files from Python directly. However, this is something that will be rectified in future updates to the code.

Something that must be noted when using the multi-node approach. Very few calculations are completed on the main node when using this approach, so it is possible to reduce the number of cores required for the main node. However, it must still be noted that as the Framework progresses, the memory requirement does increase quite significantly, especially for large batch sizes. So it is still necessary to ensure that there is enough memory available on the main node. In addition, the Framework is currently only set up to use this multi-node approach when the Reification approach is used. In other words, pure Batch calculations will only work as a single-node calculation even

```
substr1 = ['#!/bin/bash',
        '##NECESSARY JOB SPECIFICATIONS',
        '#BSUB -J sub{0}',
        '#BSUB -P *Project Numbder*',
        '#BSUB -L /bin/bash',
        '#BSUB -W 01:00',
        '#BSUB -n 20',
        '#BSUB -R "span[ptile=20]"',
        '#BSUB -R "rusage[mem=2560]"',
        '#BSUB -M 2560',
        '#BSUB -o LSFOut/Out.%J',
        '#BSUB -e LSFOut/Err.%J',
        'cd $SCRATCH/BAREFOOT',
        'module load PyTorch/1.1.0-foss-2019a-Python-3.7.2',
        'source venv/bin/activate',
        'cd barefootTest',
        'python subProcess.py {0}']

substr2 = ['#!/bin/bash',
        'cd /scratch/user/username/BAREFOOT/barefootTest/subprocess',
        'bsub < {0}.sh']

with open("data/processStrings", 'wb') as f:
  dump(['\n'.join(substr1), '\n'.join(substr2)], f)
```

Listing 2.7: Example code for constructing the job files shell script files used in the multi-node approach.

if the multi-node option is set.

2.3.4 Multi-objective Optimization

Currently, the Framework is configured for two-objective optimization. The development of n-objective optimization is underway. However, it has not been successfully implemented at this point. The multi-objective optimization approach uses many of the same methods that the single objective approach uses. However, there are a few changes that are necessitated by having multiple objectives.

When constructing either the Reification object (Barefoot and Reification Only approaches) or the GP Model object (Batch Only approach), an object is required for each objective. Therefore, rather than constructing a single Reification object, we now create two. These objects are then updated separately during the optimization. A more robust approach that will be investigated is to

have these objects built around multi-output GPs. This would be straightforward to implement in the Batch Only method. However, the Reification object code would need to be adapted to handle such a situation in the Reification only and Barefoot methods.

The second change is that every time the Truth Function is evaluated, it is necessary to re-evaluate the Pareto Front. In addition to calculating the Pareto Front, we also add it to the outputs of the Framework. Furthermore, since the Framework now has two objectives that need to be tracked, we modify the output files slightly to include both objectives.

2.4 Framework Ouputs

The final topic for discussion is the framework output. There are several outputs from the Framework, and these are saved in at least two different locations in the BAREFOOT directory. The first set is the results from the iterations. These results are saved in two pandas DataFrames. There is a folder labeled "results" in the BAREFOOT directory, and the results of the Framework are saved here. Each run of the Framework will create a results folder that has the calculation name. Subsequent framework calculations with the same calculation name will overwrite this folder, so care must be taken to save data at the end of a calculation or to change the calculation names each time. As already mentioned, the results in this directory are two pandas DataFrames. These DataFrames are saved twice, as a pickle of the DataFrame object and as a 'CSV' for human readability. The first DataFrame saved is evaluatedPoints (Table 2.1) which holds all the information on which inputs have been evaluated from each model and what the output from that model is. The second data frame is iterationData (Table 2.2) which has the information on the maximum ground truth found, the computational cost for each iteration, and the number of model calls for each model.

In addition to these DataFrames, the entire framework state is saved at the end of each iteration. This is done by pickling the barefoot object. This pickle file is saved in the "data" directory in the BAREFOOT directory.

In addition to the outputs discussed above, the Framework uses the logging module to output information on the Framework's progress. An example of the normal logging output is shown in

	Model Index	Iteration	y	x0	x1	x2	x3
0	0.0	-1.0	4.448	0.188	0.886	0.87	0.182
1	0.0	-1.0	4.442	0.699	0.94	0.068	0.111
2	1.0	-1.0	3.725	0.666	0.408	0.192	0.803
3	1.0	-1.0	3.583	0.24	0.894	0.791	0.001
4	2.0	-1.0	33.255	0.765	0.555	0.075	0.188
5	2.0	-1.0	36.250	0.402	0.031	0.685	0.543
6	-1.0	-1.0	22.3415	0.899	0.016	0.154	0.527
7	-1.0	-1.0	3.535	0.471	0.891	0.522	0.064
8	1.0	0.0	9.143	0.10574	0.40441	0.44789	0.07863
9	1.0	0.0	2.035	0.2514	0.88052	0.56246	0.48145
10	1.0	0.0	11.206	0.64209	0.12887	0.04872	0.7271
11	0.0	1.0	4.005	0.09048	0.94353	0.41923	0.40858
12	1.0	1.0	1.362	0.76124	0.82767	0.72192	0.54378
13	1.0	1.0	3.180	0.09048	0.94353	0.41923	0.40858
14	2.0	2.0	1.341	0.6476	0.90067	0.25648	0.84428
15	0.0	2.0	1.332	0.6476	0.90067	0.25648	0.84428
16	1.0	2.0	2.256	0.16678	0.96561	0.31769	0.78021
17	0.0	3.0	1.300	0.94204	0.9959	0.45024	0.11732
18	2.0	3.0	1.300	0.94204	0.9959	0.45024	0.11732
19	1.0	3.0	1.314	0.94204	0.9959	0.45024	0.11732
20	1.0	4.0	1.692	0.20801	0.99995	0.22914	0.0347
21	0.0	4.0	1.330	0.79293	0.89271	0.48804	0.93257
22	2.0	4.0	44.988	0.20801	0.99995	0.22914	0.0347
23	2.0	5.0	1.308	0.75676	0.96471	0.74903	0.90069
24	0.0	5.0	1.411	0.70908	0.69235	0.62636	0.44692
25	1.0	5.0	1.339	0.67306	0.90585	0.5968	0.74547
26	-1.0	5.0	29.497	0.13889	0.35789	0.12318	0.68573
27	-1.0	5.0	23.680	0.63383	0.03459	0.2762	0.30311
28	-1.0	5.0	22.392	0.21793	0.35725	0.50141	0.11357

Table 2.1: Example contents of the Evaluated Points Dataframe file showing the results from a calculation involving a 4D problem with 3 reduced order models (model index 0-2) and a ground truth model (model index -1) that each had 2 initial values (iteration -1 data). The optimization had a batch size of 3.

Listing 2.8. It is also possible to output debugging level logs by setting the “verbose” value in the Framework inputs to “True”

	Iteration	Max Found	Calculation Time	Truth Model	ROM 0	ROM 1	ROM 2
0	-1.0	22.341	0.085	2.0	2.0	2.0	2.0
1	0.0	22.341	149.955	2.0	2.0	5.0	2.0
2	1.0	22.341	144.315	2.0	3.0	7.0	2.0
3	2.0	22.341	154.436	2.0	4.0	8.0	3.0
4	3.0	22.341	154.847	2.0	5.0	9.0	4.0
5	4.0	22.341	155.492	2.0	6.0	10.0	5.0
6	5.0	29.497	166.032	5.0	7.0	11.0	6.0

Table 2.2: Example output from the iteration data dataframe from a calculation involving a 4D problem with 3 reduced order models. The optimization had a batch size of 3.

2.5 Validation of BAREFOOT Framework methods

2.5.1 Reification Code Validation

To validate that the Reification code developed in Python is achieving the correct results, we compared the results with those obtained by Ghoreishi et al. [13]. This test uses three empirical reduced-order models and a Finite Element Truth Model to predict the normalized strain hardening rate ($1/\tau(\mathrm{d}\tau/\mathrm{d}\epsilon_{pl})$ of a dual-phase microstructure. In this particular test case, the input is a single dimension. This input is the phase fraction of the Hard phase in the material. The three reduced-order models are empirical models that make different assumptions about the stress and strain distribution in the two phases. The simplifying assumptions are that the stress is the same in the two phases (isostress), the strain is the same in the two phases (isostrain) and that the mechanical work is the same in the two phases (isowork).

Using the exact inputs and GP model hyperparameters from work by Ghoreishi et al. [13] we constructed all the surrogate models related to the Reification approach. The output of the fused model from the Python code and the Matlab code used by Ghoreishi et al. is shown in Figure 2.4. As can be observed, the results are very similar. There is a slight difference in the fused model in the middle of the design space. However, this difference is not large enough to cause significant errors.

```
... - INFO - ###########################################################
... - INFO - #                                                         #
... - INFO - #   Start BAREFOOT Framework Initialization               #
... - INFO - #                                                         #
... - INFO - ###########################################################
... - INFO - Initialization Completed
... - INFO - Reification Object Initialized. Ready for Calculations
... - INFO - Start BAREFOOT Framework Calculation
... - INFO - ###########################################################
... - INFO - #         Start Iteration : 0000          #
... - INFO - ###########################################################
... - INFO - Start Acquisition Function Evaluations for 3000 Parameter Sets
... - INFO - Acquisition Function Evaluations Completed
... - INFO - Clustering of Acquisition Function Evaluations Completed
... - INFO - Start ROM Function Evaluations | 3 Calculations
... - INFO - ROM Function Evaluations Completed
... - INFO - Dataframes Pickled and Dumped to Results Directory
... - INFO - Calculation State Saved
... - INFO - Iteration 0 Completed Successfully
...
... - INFO - ###########################################################
... - INFO - #         Start Iteration : 0005          #
... - INFO - ###########################################################
... - INFO - Start Acquisition Function Evaluations for 3000 Parameter Sets
... - INFO - Acquisition Function Evaluations Completed
... - INFO - Clustering of Acquisition Function Evaluations Completed
... - INFO - Start ROM Function Evaluations | 3 Calculations
... - INFO - ROM Function Evaluations Completed
... - INFO - Start Truth Model Evaluations
... - INFO - Start Max Value Calculations | 50 Sets
... - INFO - Max Value Calculations Completed
... - INFO - Start Truth Model Evaluations | 3 Sets
... - INFO - Truth Model Evaluations Completed
... - INFO - Dataframes Pickled and Dumped to Results Directory
... - INFO - Calculation State Saved
... - INFO - Iteration 5 Completed Successfully
...
... - INFO - ###########################################################
... - INFO - #                                                         #
... - INFO - #  Iteration or Budget Limit Met or Exceeded             #
... - INFO - #      BAREFOOT Calculation Completed                     #
... - INFO - #                                                         #
... - INFO - ###########################################################
```

Listing 2.8: Edited output from the logging module that shows the information provided during calculations for an iteration that involves a Ground Truth evaluation and an iteration that does not.

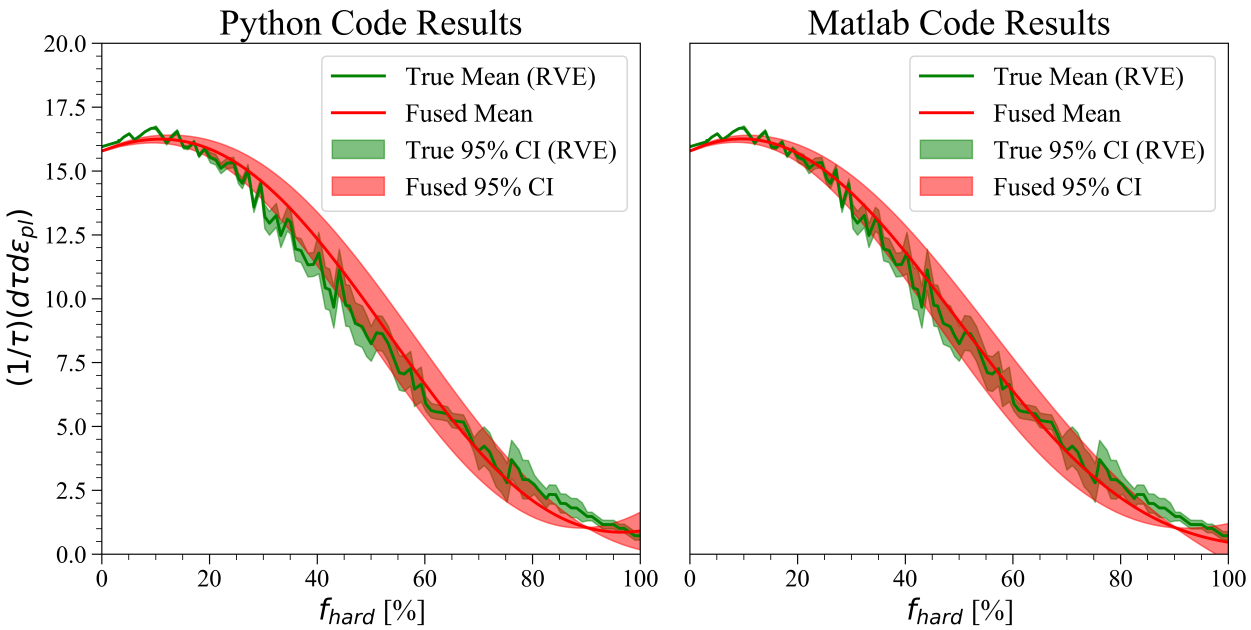


Figure 2.4: Comparison of the Reification Results obtained from the Matlab code used in [13], and the Python code used in the BAREFOOT Framework.

2.5.2 Validation of single objective approaches

In order to test the different single objective approaches available in the Framework we considered a test function based on the eggholder function,

$$f(x) = |x_1| sin\left(5x_1\right) + |x_2| sin\left(6x_2\right) \tag{2.23}$$

which is defined for $x_1, x_2 \in [-\pi, \pi]$. The model is shown in Figure 2.5. As can be observed, the function has many local optima, however, the global maximum is $f(x) = 5.719076$ at $x = [2.841, -2.889]$.

Using a Fourier series expansion, we defined three models to be used as reduced-order models for the Reification-based approaches 2.6. These models include up to the third-order terms from the Fourier series expansion.

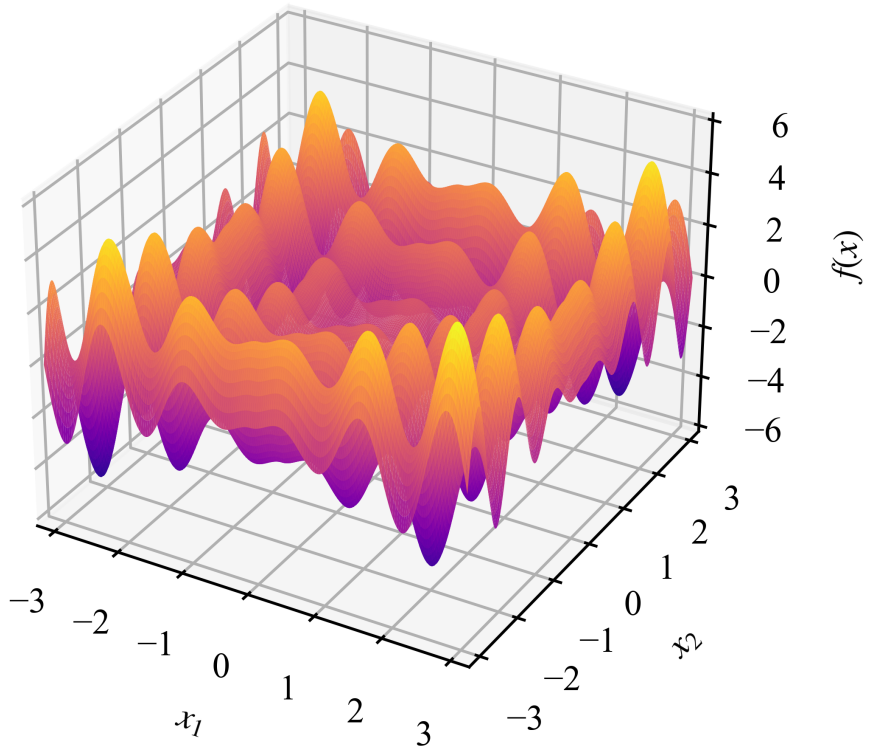
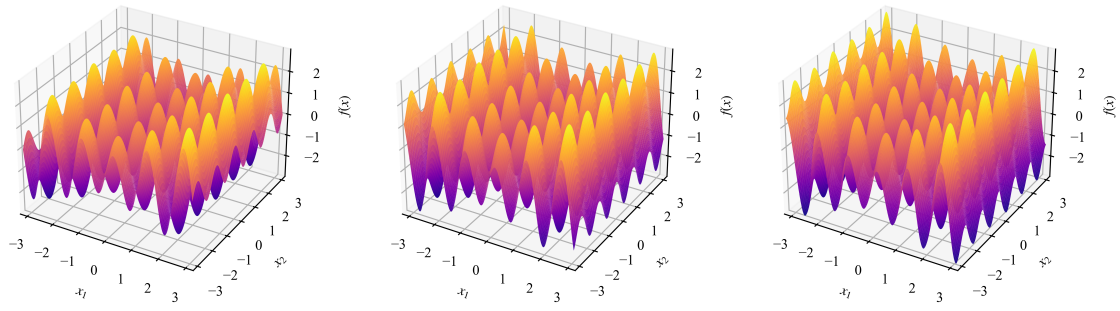


Figure 2.5: Plot of the sample test function used in the accuracy test



(a) (b) (c)

Figure 2.6: Representation of the three reduced order models used in the testing of the single objective Framework Approaches

For comparing the results we use the Gap metric,

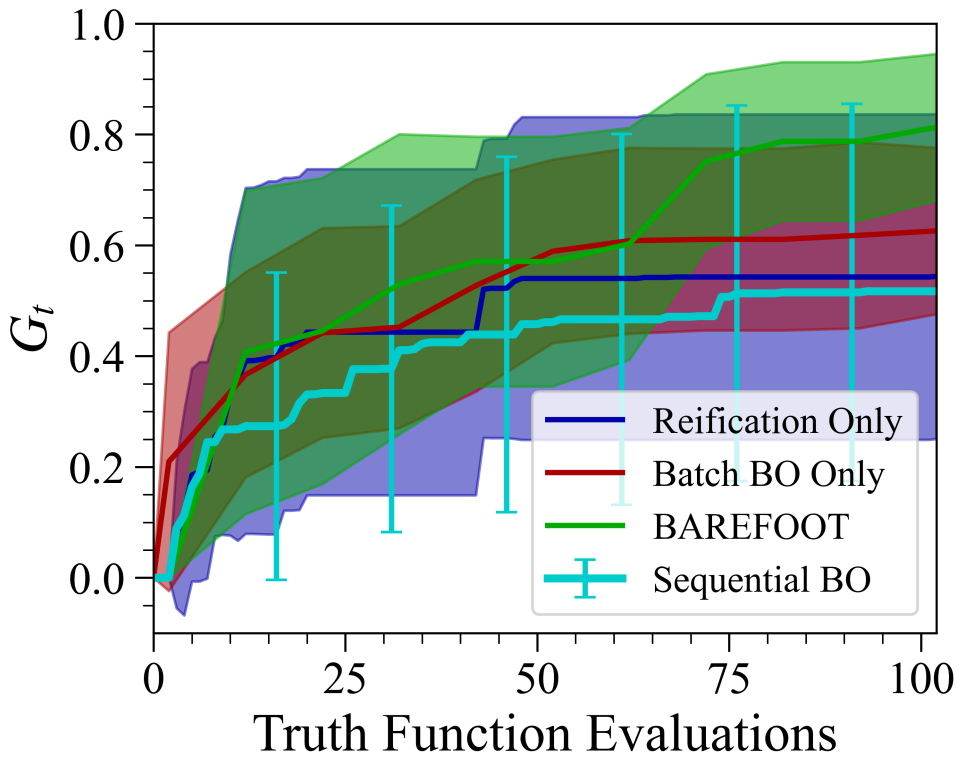


Figure 2.7: Results for running the BAREFOOT Framework calculations for Single Objective optimization. The results show the mean and approximate 95% confidence interval of the Gap Metric for 10 calculations for each approach.

$$G_t = \frac{y^+ - y^*}{y^{max} - y^*} \tag{2.24}$$

where y^+ is the current best value from the optimization, y^* is the best value from the Truth Function from the initial sampling before the optimization starts, and y^{max} is the global maximum of the function. Using this measure gives us an idea of how much information is gained by each of the approaches as a function of Truth Model evaluation, Figure 2.7.

As can be observed in Figure 2.7 all three single-objective methods perform relatively well. However, we can see that the Barefoot method does provide a better result than the other two methods. In addition to this, we draw attention to the fact that the batch size is 10. So while all three methods have the same number of Truth Function evaluations, the Reification Only method

will take approximately ten times longer than the Barefoot and Batch Only methods (assuming that the Truth Function evaluations dominate the time taken). This reduction in time is the significant advantage of the batch-based approaches.

2.5.3 Validation of the Multi-Objective Optimization approach in BAREFOOT

One of the most recent developments in BAREFOOT is the possibility of multi-objective optimization. At this point, the multi-objective option is only available for a two-objective optimization. The functionality for expanding the framework to any number of objectives is under development but isn't available yet. The multi-objective functionality has been tested against one of the most common multi-objective test functions, namely the Poloni's function [53]. The Poloni function is a two-dimensional function defined by the dual set of equations,

$$\begin{aligned} f_1(x_1, x_2) &= 1 + (A_1 - B_1(x_1, x_2))^2 + (A_2 - B_2(x_1, x_2))^2 \\ f_2(x_1, x_2) &= (x_1 + 3)^2 + (x_2 + 1)^2 \end{aligned} \tag{2.25}$$

where,

$$\begin{aligned} &-\pi \leq x_1, x_2 \leq \pi \\ &A_1 = 0.5\sin(1) - 2\cos(1) + \sin(2) - 1.5\cos(2) \\ &A_2 = 1.5\sin(1) - \cos(1) + 2\sin(2) - 0.5\cos(2) \\ &B_1(x_1, x_2) = 0.5\sin(x_1) - 2\cos(x_1) + \sin(x_2) - 1.5\cos(x_2) \\ &B_2(x_1, x_2) = 1.5\sin(x_1) - \cos(x_1) + 2\sin(x_2) - 0.5\cos(x_2) \end{aligned}$$

The two objectives in Equation 2.5.3 are minimized. Using a Latin Hypercube sampling of

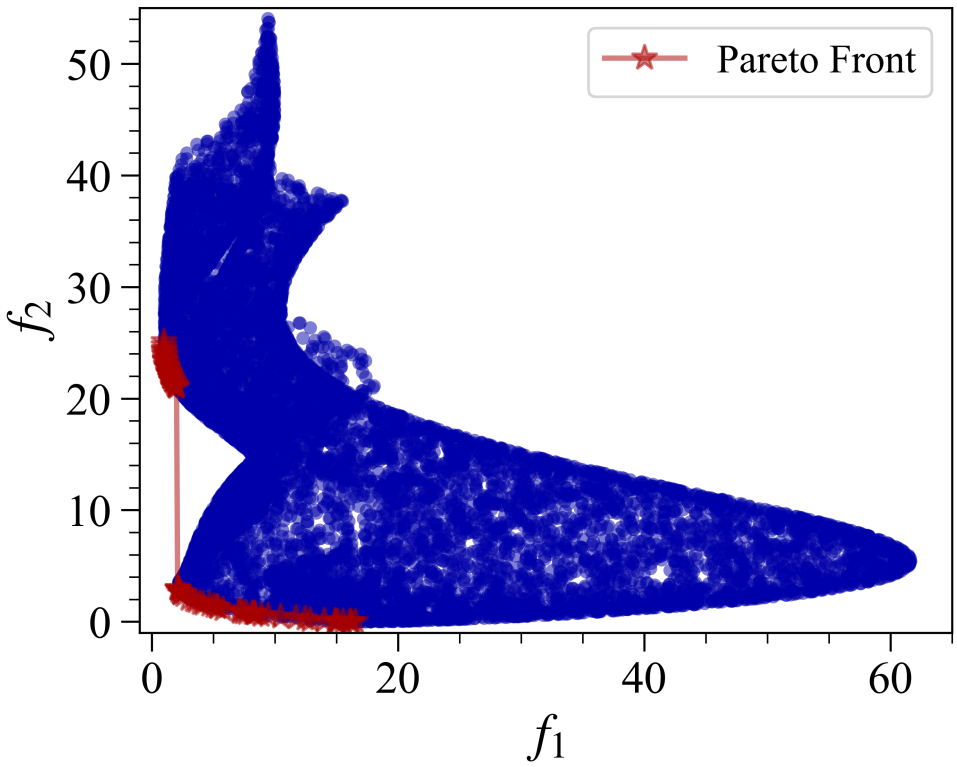


Figure 2.8: Plot showing the general results and calculated Pareto Front for the two objectives of the Poloni two-objective Standard Test Function.

the design space to obtain 10,000 points, an illustration of the possible range of values for the two objectives is found and shown in Figure 2.8. Using the 10,000 points, we evaluated the Pareto Front and found the non-dominated points in the set of 10,000. These points are plotted as the red stars in Figure 2.8.

To test the multi-objective optimization in the BAREFOOT Framework, we use all three approaches, Reification only, Batch Only, and BAREFOOT. For the Reification only approach, which requires reduced-order models, we created two reduced-order models by modifying the Poloni Functions parameters. This is an ad hoc approach and is not ideal. A more structured approach will be presented later to demonstrate how the ad hoc approach can be detrimental to the Reification approach. All approaches were set to have the same parameters and were run to obtain the same number of Truth Model evaluations.

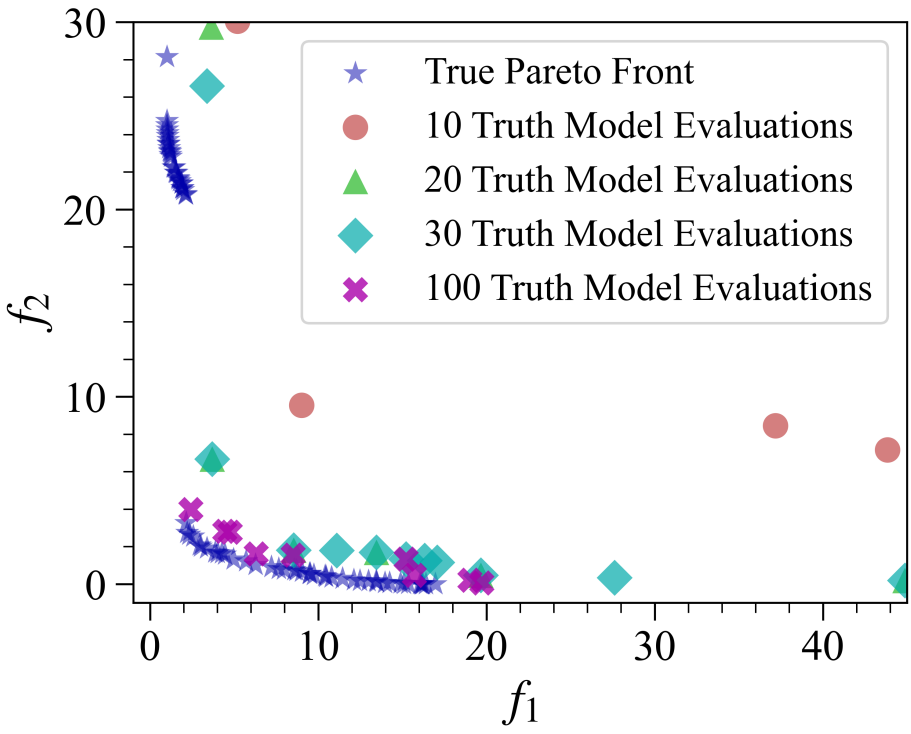


Figure 2.9: Results from a Multi-objective calculation in the BAREFOOT Framework that utilized the Barefoot Approach. The plots show how the Pareto Front develops as the iterations increase.

In Figure 2.9 we can observe that the Barefoot method is optimizing and starting to approach the Best Known Pareto Front. However, there is still room for improvement since the results have not captured the part of the Pareto Front in the top left corner. The hypothesis for why this is happening is that the quality of the reduced-order models has an effect on the Reification part of the approach. While the Reification approach can account for the model discrepancy, the approach works better when this discrepancy smaller. When the reduced-order models are less accurate, the approach still works but is not as efficient.

The results from using the Batch Only method are very good, Figure 2.10. After 100 Truth Model evaluations, the Batch Only approach has identified much of the Best Known Pareto Front. This approach works better since it uses only the Truth Model and not an ad hoc construction of reduced-order models. This contrast is beneficial when comparing the results in Figure 2.10 with the results from the Reification only approach in Figure 2.11. In Figure 2.11 we can observe that

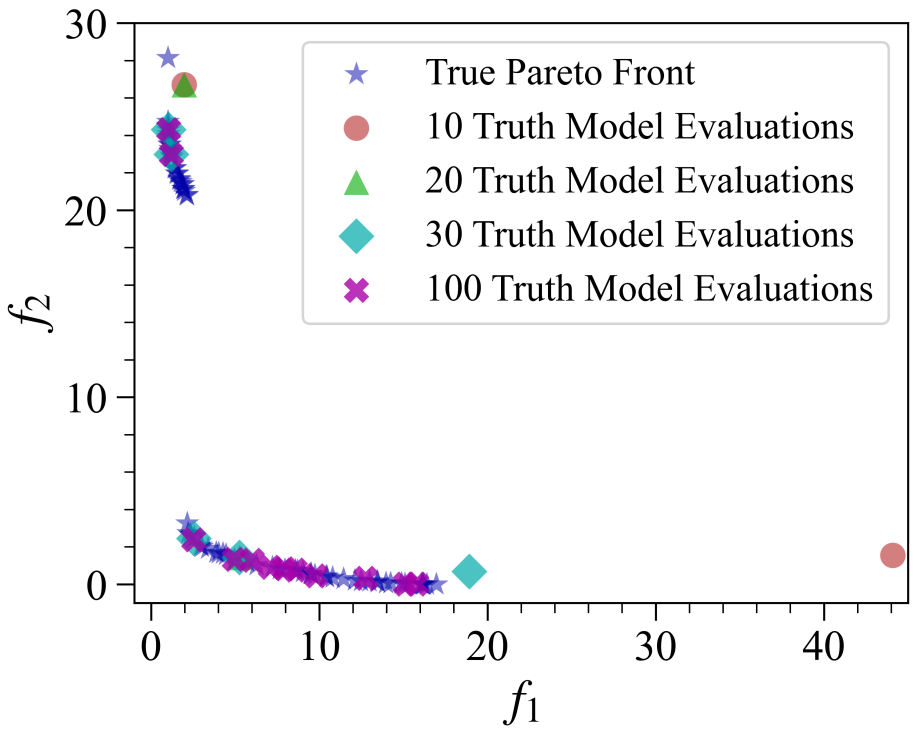


Figure 2.10: Results from a Multi-objective calculation in the BAREFOOT Framework that utilized the Batch Only Approach. The plots show how the Pareto Front develops as the iterations increase.

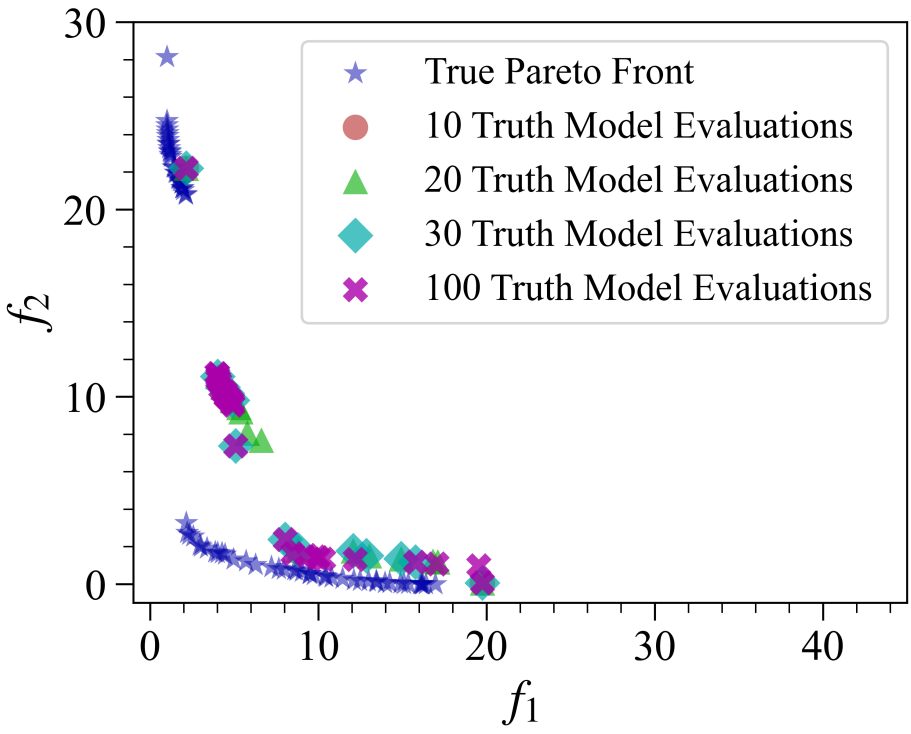


Figure 2.11: Results from a Multi-objective calculation in the BAREFOOT Framework that utilized the Reification Only Approach. The plots show how the Pareto Front develops as the iterations increase.

optimization has improved the Pareto Front slightly in the 100 Truth Model evaluations. There is significantly less improvement than seen in the Batch Only approach.

2.6 Conclusion and Planned Development

The BAREFOOT Framework has proven to be a valuable tool for the optimization and design of materials. Despite the significant changes that have been implemented since the beginning of this Framework's development, there is still plenty of room for development. One of the most important developments is the expansion of the Framework to n-objective multi-objective optimization approaches. This is of critical importance since most material problems aim to optimize at least two properties of the material. There will be a challenge when implementing this kind of improvement since the multi-objective approaches are typically more computationally costly than a single objective, which could significantly increase the computational cost of the Framework.

Therefore, implementing the multi-objective approaches must be coupled with verifying that all approaches are implemented to limit the computational cost as much as possible. Other avenues for the advancement of the code are to implement additional approaches such as active subspace optimization.

3. MATERIALS DESIGN THROUGH BATCH BAYESIAN OPTIMIZATION WITH MULTI-SOURCE INFORMATION FUSION*

3.1 Introduction

Integrated Computational Materials Engineering (ICME) [1] calls for the integration of various computational tools (validated against experiments) to establish quantitative process-structure-property-performance (PSPP) relationships. Inverting these relationships can accelerate the design of materials—under the key assumption that simulations are faster and cheaper than experiments. However, there are still significant challenges to this approach.

Despite the assumption that simulations are cheaper than experiments, a major drawback of ICME implementations is the considerable computational cost associated with evaluating PSPP chains. This has recently been addressed through the deployment of Bayesian Optimization (BO) to efficiently balance the exploration and exploitation of materials design spaces [54], [55].

Furthermore, most ICME frameworks tend to assume that there is a single information source (i.e., model) per linkage along the PSPP chain. In recent work [13], [14], however, we have shown that this is an unnecessary limitation, as the combination of multiple information sources–each containing *at least some* useful information about the problem space—always results in significant improvements in the efficiency of ICME-based alloy design schemes.

A limitation shared by modern (i.e., BO-based) and more traditional ICME frameworks is that the vast majority of them query PSPP relationships in a sequential manner (i.e., one-at-a-time). In computational settings, the sequential exploration of materials spaces is far from effective, given the availability of high-performance research computing (HPRC) facilities that make high-throughput materials simulations relatively straightforward. When it comes to experimental

*Reprinted with permission from "Materials Design Through Batch Bayesian Optimization with Multisource Information Fusion," by R. Couperthwaite, A. Molkeri, D. Khatamsaz, A. Srivastava, D. Allaire, and R. Arroyave, JOM, Oct. 2020, doi: 10.1007/s11837-020-04396-x. Copyright 2020 The Minerals, Metals and Materials Society

materials science, there has been sustained growth in the number of synthesis and characterization approaches amenable to parallelization. Indeed, the Materials Genome Initiative (MGI) [2] stimulated the development of high-throughput (i.e., combinatorial) experimental [22]–[25] or computational [26] schemes as a way to accelerate the exploration of materials design spaces.

High throughput experimental methods tend to involve thin-film [27] combinatorial libraries although, very recently, additive manufacturing platforms have been used for parallel synthesis of alloys [23]. While optical and electrical properties are most easily measured in a high-throughput fashion [23], recent approaches have shown that it is possible to rapidly measure other material properties such as composition and microstructure [24], [25], hardness [25], and even transformation temperature of shape memory alloy thin films [22]. These high-throughput approaches, while highly advantageous, suffer from the fact that they tend to be open-loop, one-shot approaches, as they lack principled policies to integrate the information gained from the high-throughput exploration to decide *what to do next once the first information-gathering step has been taken.*

There are significant opportunities to further improve BO-based ICME approaches [13], [14] by incorporating the ability to query the materials design space in a parallel fashion. This would combine the advantages of ICME (i.e., closed loops) and combinatorial materials science while addressing their common limitations (i.e., agnosticism regarding resource constraints). We note, however, that such an approach would also significantly benefit exclusively experimental combinatorial materials science efforts, including recently proposed concepts, such as self-driven laboratories [56]—as well as their computational counterparts [54]— that so far are implemented using sequential BO schemes.

The challenge of exploring and exploiting design spaces in a *parallel and optimal* fashion can be set as a general problem of **Batch Bayesian Optimization**. The key challenge to BBO is how to carry out such balanced exploration/exploitation in parallel while maintaining optimality throughout the process. Some common approaches to BBO include multi-step look ahead policies [28], [29] where the batch is created by sequentially adding the predictions from the surrogate model and predicting a new best point. Another approach considered adding queries that maximise

the variance after each sequential addition from the surrogate model [30], [31]. A third approach attempts to extract multiple peaks from the same acquisition function by removing peaks that have already been identified [32].

A more recent approach by Joy *et al*. [33] considers a slightly more intuitive approach: in sparsely sampled high-dimensional problem spaces it is too risky to place too much confidence on the tuning of the hyperparameters of the surrogate model, as the latter will depend heavily on the data captured thus far. Instead, Joy *et al*. assume that the hyperparameters can take any possible value (within reasonably set bounds) and proceeds to carry out BO over all the surrogate models that result from sampling the hyperparameter space. The predictions from this batch of BO optimizations are clustered according to the number of samples that will be evaluated in the next step, as will be described below.

The current work combines the approaches of Ghoreishi *et al*. [13], [14] for multi-information source BO and Joy *et al*. [33] for Batch BO into a single framework. Additionally, a thermodynamic model connecting chemistry and processing conditions to microstructure phase constitution is connected to the microstructural mechanics models to establish a (chemistry) processing-structure-property chain. The framework is demonstrated using the same four micromechanical models used in [13], namely the isowork, isostress, isostrain reduced-order models, as well as, a finite element representative volume element (RVE) micromechanical model, connecting the microstructure of a dual-phase steel to its mechanical response. We start by presenting each of the elements of the framework and proceed to evaluate its performance under different policies for continuation/termination of the optimization loop.

3.2 Methods

The design objective of the current work is the maximization of the normalized strain hardening rate ($1/\tau(\mathrm{d}\tau/\mathrm{d}\varepsilon_{pl})$) of dual-phase high-strength steel. This parameter was chosen as it is an indication of ductility and formability, with higher values indicating better ductility and formability. The current work considers the optimization of a dual-phase (martensite-ferrite) composed of Fe, C, Mn, and Si. The material is considered to undergo a single-stage intercritical annealing heat

treatment followed by rapid quenching. For simplification, the only parameters optimized are the concentration of C (wt%) and the intercritical annealing temperature. The Mn and Si compositions are kept constant in the current work. The ranges for the two parameters and the composition of the Mn and Si are shown in Table 3.1.

Table 3.1: The optimization approach was conducted on a dual-phase steel alloyed with C, Mn and Si. The aim was to optimize the carbon content and intercritical annealing temperatures in the range shown to obtain a maximum in the normalized strain hardening rate.

T_{IA} [°C]	X_C [wt%]	X_{Mn} [wt%]	X_{Si} [wt%]
650 - 850	0 - 1	0.328	0.283

The first part of this section deals with descriptions of the individual computational tools that are being used in both of these methods. These descriptions are not a detailed analysis of the methods, and interested readers are directed to the various references should further information on the methods be required. Later in this section, we explain how these computational tools are combined into the current framework. After constructing the framework, the optimization process is tested using three different case studies, describing the parameters and termination criteria used in each of the case studies.

3.2.1 Computational Tools

3.2.1.1 Gaussian Process

One of the major ingredients in BO is a surrogate model capable of predicting the outcome of experiments yet to be carried out, as well as, the uncertainty associated to these predictions [57]. In BO problems, such predictive models tend to be constructed out of Gaussian processes (GPs) due to their underlying mathematical properties (including smoothness, controllable modeled correlation among observed points, etc.). A GP is a non-parametric statistical model that defines a stochastic process $f(\boldsymbol{x})$, where all the finite distributions of the model are assumed to be multivariate normal. Using this definition, the joint probability distribution of the outputs from the stochastic process

may be modeled as an n-dimensional multivariate normal distribution for any finite set of inputs $\mathbb{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$,

$$p(f(\boldsymbol{x}_1), ..., f(\boldsymbol{x}_n)) \sim \mathcal{N}_n(\boldsymbol{\mu}, \boldsymbol{C}). \tag{3.1}$$

where $\boldsymbol{\mu}$ is the mean vector and $\boldsymbol{C}$ is the covariance function. The mean and covariance are defined by a mean function $\mu(\cdot)$ and a covariance function $C(\cdot, \cdot)$ with the following properties,

$$\mu(\boldsymbol{x}_i) = \boldsymbol{\mu}_i = \mathbb{E}\left[f(\boldsymbol{x}_i)\right]. \tag{3.2}$$

$$C(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{C}_{i,j} = cov\left[f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)\right]. \tag{3.3}$$

From the above definitions, we formally define a Gaussian Process as $f(\cdot) \sim GP(\mu, C)$. A more detailed explanation of this kind of stochastic process is provided in the work by Rasmussen and Williams [58].

The covariance function of the GP captures the degree of correlation between two different locations in the input space. The ability to make explicit inferences (through well-defined covariances) about the degree to which observations are correlated is one of the reasons why GPs tend to be the model class of choice in BO—inherent to BO is the assumption that current information about the state of a system can be used to infer yet-to-be-observed states. Due to the difference in scales (e.g., temperature and compositions) between the inputs of the current approach, the inputs were re-scaled to the $[0, 1]$ interval. Therefore, the notion of space is abstract and the spatial dependence denotes a metric representing the distance between two points in a mathematical space. In the current work, the Matérn class of covariance functions was used since they are generally more robust when the smoothness of the data is not known [59]:

$$C(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \left[\frac{2^{1-v}}{\Gamma(v)}\left(\frac{\sqrt{2v}(x_i - x_j)}{l}\right)^v K_v\left(\frac{\sqrt{2v}(x_i - x_j)}{l}\right)\right]. \tag{3.4}$$

The Matérn class of covariance functions is defined by Equation 3.4, where σ_f^2 is referred to as the signal variance, l is the characteristic length scale and v is a parameter that determines the shape of the function. However, it is more common to define the function by specifying a value for v. The function has a closed form solution for values of $v = r + 1/2$ where $r \in \mathbb{Z}^+$. One of the more common values is $v = 5/2$ [58]. This choice reduces the covariance function to the form shown in Equation 3.5.

$$C(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \left(1 + \frac{\sqrt{5}(x_i - x_j)}{l} + \frac{5(x_i - x_j)^2}{3l^2}\right) \exp\left(-\frac{\sqrt{5}(x_i - x_j)}{l}\right). \tag{3.5}$$

3.2.1.2 Reification

A key ingredient of the present framework is the simultaneous consideration of multiple information sources at once and their fusion to achieve better, unbiased, predictions that take advantage of all the useful information provided by each model individually [14]. Information fusion requires the quantification of the statistical correlations among the different information sources and between the sources and the ground truth. The *Reification* method developed by Thomison and Allaire [15] estimates the model correlations by sequentially elevating each model at a time as 'truth' (i.e., the model is 'reified'), followed by the computation of the statistical correlation between this reified model and the other sources.

Assuming that we have two models, $f_1(x)$ and $f_2(x)$, that can both estimate the quantity of interest (y) with some discrepancy,

$$y = f_1(x) = \bar{f}_1(x) + \delta_1(x), \tag{3.6}$$

$$y = f_2(x) = \bar{f}_2(x) + \delta_2(x), \tag{3.7}$$

where $\bar{f}_1(x)$ is the mean prediction and the model discrepancies $\delta_i(x)$ are assumed to be normally distributed with, $\delta_1(x) \sim \mathcal{N}(0, \sigma_1^2)$ and $\delta_2(x) \sim \mathcal{N}(0, \sigma_2^2)$.

Using this information we reify model 1 and then calculate the error of each model. Since

model 1 has been reified, the standard deviation of model one ($\tilde{f}_1(x^*)$) at a single point in the design space (x^*) is defined simply by the model discrepancy as:

$$\tilde{f}_1(x^*) = f_1(x^*) - \bar{f}_1(x^*) = \delta_1(x^*), \tag{3.8}$$

and the error for model 2, with respect to model 1, is defined by:

$$\tilde{f}_2(x^*) = f_2(x^*) - \bar{f}_2(x^*) \tag{3.9}$$

$$= \bar{f}_1(x^*) - \bar{f}_2(x^*) + \delta_1(x^*). \tag{3.10}$$

To calculate the correlation it is necessary to calculate both the mean squared errors and the covariance. Using the errors above, the mean squared errors are defined by:

$$\mathbb{E}[\tilde{f}_1(x^*)^2] = \mathbb{E}[\delta_1(x^*)] = \sigma_1^2, \tag{3.11}$$

$$\mathbb{E}[\tilde{f}_2(x^*)^2] = \mathbb{E}[(\bar{f}_1(x^*) - \bar{f}_2(x^*))^2] + \mathbb{E}[\delta_1(x^*)] \tag{3.12}$$

$$= (\bar{f}_1(x^*) - \bar{f}_2(x^*))^2 + \sigma_1^2, \tag{3.13}$$

while the covariance is given by:

$$\mathbb{E}[\tilde{f}_1(x^*)\tilde{f}_2(x^*)] = \sigma_1^2. \tag{3.14}$$

The Pearson correlation coefficient (ρ) can then be calculated,

$$\rho_1(x^*) = \frac{\sigma_1^2}{\sigma_1\sigma_2} = \frac{\sigma_1}{\sqrt{(\bar{f}_1(x^*) - \bar{f}_2(x^*))^2 + \sigma_1^2}}, \tag{3.15}$$

where the subscript on the coefficient indicates which model has been reified. This process is repeated for the other model to obtain the value of $\rho_2(x^*)$. When more than two models are used,

the correlation coefficients are calculated for every pair of models. The average correlation ($\bar{\rho}$) is used in the model fusion approach and is calculated using the following:

$$\bar{\rho}(\boldsymbol{x}^*) = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2}\rho_1(\boldsymbol{x}^*) + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\rho_2(\boldsymbol{x}^*). \tag{3.16}$$

Theoretically, this Reification approach can be expanded to any number of models. However, there are practical limits to how many models can be considered, based on the available computational resources available as well as the time necessary to compute all the relevant pairwise correlations. In most cases it is unlikely that such a computational limit can be reached since the number of models/sources corresponding to every linkage of the PSPP chain is likely to be generally modest.

3.2.1.3 Model Fusion

Given Equations 3.6 and 3.7 for the two models that estimate the quantity of interest (y), the fused model can be represented by the equation:

$$y = k_1(x^*)f_1(x^*) + k_2(x^*)f_2(x^*) \tag{3.17}$$

where $k_1(x^*)$ and $k_2(x^*)$ are real-valued scalar quantities subject to $k_1(x^*) + k_2(x^*) = 1$. By assuming that both the models have a normal distribution given by $f_1(x^*) \sim \mathcal{N}(\bar{f}_1(x^*), \sigma_1^2)$, and $f_2(x^*) \sim \mathcal{N}(\bar{f}_2(x^*), \sigma_2^2)$, it is possible to solve Equation 3.17 for $k_1(x^*)$ and $k_2(x^*)$ by solving the minimization problem:

$$\min_{\mathbf{k}} \ \mathbf{k}^T\Sigma\mathbf{k} \ \text{ subject to } \ k_1 + k_2 = 1 \tag{3.18}$$

where $\mathbf{k} = [k_1, k_2]^T$ and

$$\Sigma = \begin{bmatrix} \mathbb{E}[\tilde{f}_1(x^*)^2] & \mathbb{E}[\tilde{f}_1(x^*)\tilde{f}_2(x^*)] \\ \mathbb{E}[\tilde{f}_2(x^*)\tilde{f}_1(x^*)] & \mathbb{E}[\tilde{f}_2(x^*)^2] \end{bmatrix} = \begin{bmatrix} \sigma_1^2 & \bar{\rho}\sigma_1\sigma_2 \\ \bar{\rho}\sigma_2\sigma_1 & \sigma_2^2 \end{bmatrix}. \tag{3.19}$$

The covariance matrix, Σ, requires the correlation coefficient, ρ. This is approximated using the Reification approach outlined previously that provides $\bar{\rho}$ as an estimate of this quantity. The solution of this minimization problem defines a fused model for y that has a mean defined by:

$$\mathbb{E}(y) = \frac{(\sigma_2^2 - \bar{\rho}\sigma_1\sigma_2)\bar{f}_1(x^*) + (\sigma_1^2 - \bar{\rho}\sigma_1\sigma_2)\bar{f}_2(x^*)}{\sigma_1^2 + \sigma_2^2 - 2\bar{\rho}\sigma_1\sigma_2} \tag{3.20}$$

and variance

$$\text{Var}(y) = \frac{(1 - \bar{\rho}^2)\sigma_1^2\sigma_2^2}{\sigma_1^2 + \sigma_2^2 - 2\bar{\rho}\sigma_1\sigma_2} \tag{3.21}$$

The proof and full derivation of these equations are found in the work by Winkler [52]. While it is not a considered in the current work , it is worthwhile to note that this Reification fusion approach could be used to estimate the impact of parameter uncertainty on the BO itself by constructing multiple models with different parameters, using the Reification approach to weigh the importance of each of the models, relative to the 'ground truth'.

3.2.1.4 Knowledge Gradient

The second [57] of any BO approach is the acquisition function or policy that is used to select the next experiment (or simulation/observation) to make, given the data acquired thus far, as well as the underlying model (i.e., GP with hyperparameters) used to represent the problem space. In BO, there are a large number of acquisition functions that can be used, including Probability of Improvement (PI) [44], Expected Improvement (EI) [12], [43], Upper Confidence Bound (UCB) [60] and Knowledge Gradient (KG) [47]. In this work, we have selected the KG as it tends to be better suited to potentially noisy problem spaces [61], although it should be pointed out that KG is considerably more expensive to compute than other acquisition functions, including EI, PI, and UCB.

For the calculation of the Knowledge Gradient, we define a set of M distinct alternative points in the fused model input space and evaluate the mean, μ_x^n, and variance, $(\sigma_x^n)^2$, using the posterior predictive distribution of the fused model. The n superscript denotes the iteration number. KG is

then defined as:

$$\nu^{KG} = \max_{x^n \in \{1,\ldots,M\}} \mathbb{E}_n \left[\left(\max_{x'} \mu_{x'}^{n+1} \right) - \left(\max_{x'} \mu_{x'}^{n} \right) \right] \tag{3.22}$$

where $\mathbb{E}_n$ is the conditional expectation with respect to what is known after the first n iterations and μ_x^{n+1} is the Bayesian look-ahead prediction of the mean at step $n+1$. The Knowledge Gradient approach uses a Bayesian look ahead approach to estimate μ_x^{n+1} conditional on μ_x^n and $(\sigma_x^n)^2$. This is done by first defining the precision of the posterior predictive distribution as $\beta_x^n = (\sigma_x^n)^{-2}$. According to the work by Frazier *et al.* [47], the conditional variance for the look ahead step is defined by:

$$\tilde{\sigma}\left(\beta_x^n\right) = \sqrt{\left(\beta_x^n\right)^{-1} - \left(\beta_x^n + \beta^{\epsilon}\right)^{-1}} \tag{3.23}$$

where β^{ϵ} is the measurement precision and is generally assumed to be constant over the entire input space. Then, the look ahead mean is defined as:

$$\mu_x^{n+1} = \mu^n + \tilde{\sigma}\left(\beta_x^n\right) Z e_x \tag{3.24}$$

where Z is the standard normal distribution and e_x is a vector in $\mathbb{R}^M$ with all components zero except for component x. For a full description of the method and algorithm implemented in the current work refer to the work by Frazier *et al.* [47].

3.2.1.5 Batch Bayesian Optimization

Given the formulation for the GPs covariance function shown in Equation 3.5, the hyperparameters are σ_f, σ_n and l. These three hyperparameters and the available data determine the shape of the GP. The characteristic length scale, l, will possibly have the greatest effect, but the other two hyperparameters also play a role. Usually, the hyperparameters are determined by minimizing the log-marginal likelihood of the GP, given the data. This is typically done by either gradient-based optimization approaches or BO methods [58]. Unfortunately, when faced with relatively sparse

high-dimensional input spaces, the optimized values of the hyperparameters may be extremely dependent on the data already available and it is thus too risky to make such definite inferences about the covariance structure of the entire problem space, and to use this assumed correlation to evaluate the BO acquisition policy.

Joy *et al*. [33] propose that under data-sparse conditions, rather than selecting single values for each of the hyperparameters, it is advisable instead to sample a wide range of hyperparameters (within reasonable bounds), thereby making no assumption with regards to the shape of the underlying objective function and on the degree of correlation between points in the design space. It follows that each set of hyperparameters sampled through this framework would result in different predictions as to the location of the next best point to query given the current knowledge of the system and the acquisition function used:

$$\boldsymbol{x}_{1:n} = \underset{\boldsymbol{x}\in\chi}{\operatorname{argmax}}\ \nu^{KG}(\boldsymbol{x}|GP(\boldsymbol{D}_0, \boldsymbol{\theta}_{1:n})), \tag{3.25}$$

where the acquisition function, in this case, is the Knowledge Gradient (as defined above), $\boldsymbol{D}_0$ is the data available at the start of the iteration, and n is the number of sets of hyperparameter values ($\boldsymbol{\theta}_{1:n}$).

After acquiring all of these predicted 'best design points' it is then possible to cluster them into the number of clusters (K) required by the size of the batch processing step. This is done using a k-medoids approach, which clusters the samples to minimize the total distance between the samples and the selected medoids. The number of medoids (K) is defined by the size of the batch available to query the problem space. The difference between this approach and a k-means approach is that the medoids are samples in the dataset rather than the arbitrary centroids predicted by k-means that may not necessarily exist in the data acquired thus far. This clustering approach defines K points that can be queried from the information source, *in parallel*. For a more in-depth discussion of the technique, please refer to Appendix A.

3.2.1.6 Mechanical Models

Three reduced-order models and one finite element micromechanical model are used in the current work. The reduced-order models represent different approaches to homogenize the response of a composite microstructure based on different assumptions on the nature of the interactions/-coupling among the constituent phases in the composite:

1. An isostrain model where the strain is assumed to be the same in both phases [62].
2. An isostress model where the assumption is made that the stress is homogenous throughout the composite [63].
3. An isowork model where the mechanical work in the two phases is assumed to be the same [64].

The 'ground truth' in the current work corresponds to the simulation of the deformation behavior of a representative volume element (RVE) representation of the dual-phase microstructure through the use of Finite Elements. All models include isotropic hardening that followed Ludwik's Power Law [65]. The strength of the two phases was dependent on the composition based on the assumption that only carbon affected the martensite strength, while manganese and silicon affected the ferrite strength. Further details of these models can be found in Appendix A.

For comparison, the output of each of the low order models is compared with the output from the RVE model (Figure 3.1). In the optimization calculations, a surrogate model was used in place of the true RVE model to speed up the calculations in the framework.

3.2.2 Current Approach

The previous section provides details on the methods applied in the current work. This section will explain the overall algorithmic approach used in the current work. A schematic showing the general flow of the framework is shown in Figure 3.2.

The first step in the current approach is to define the hyperparameter sets to be used for generating the *fused model GP*. These hyperparameter sets are constructed by using a Latin-hypercube

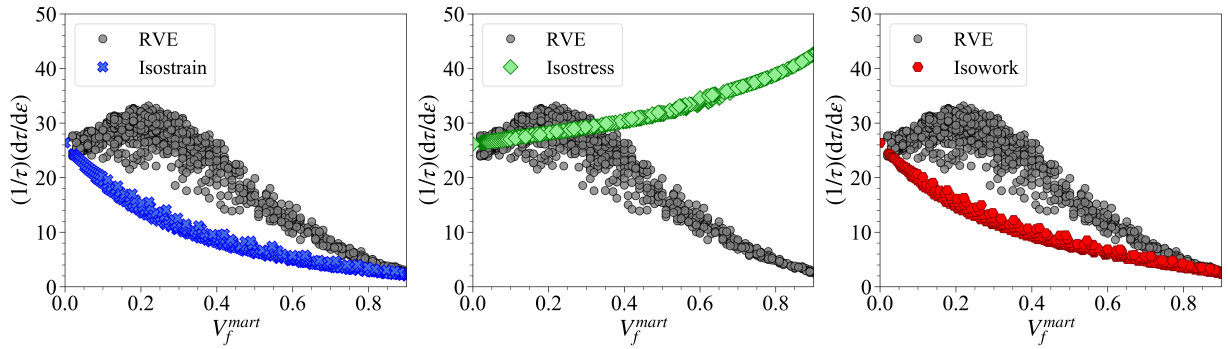


Figure 3.1: Comparison of the outputs of the three reduced-order models and the RVE finite element model, a) isostrain b) isostress c) isowork.

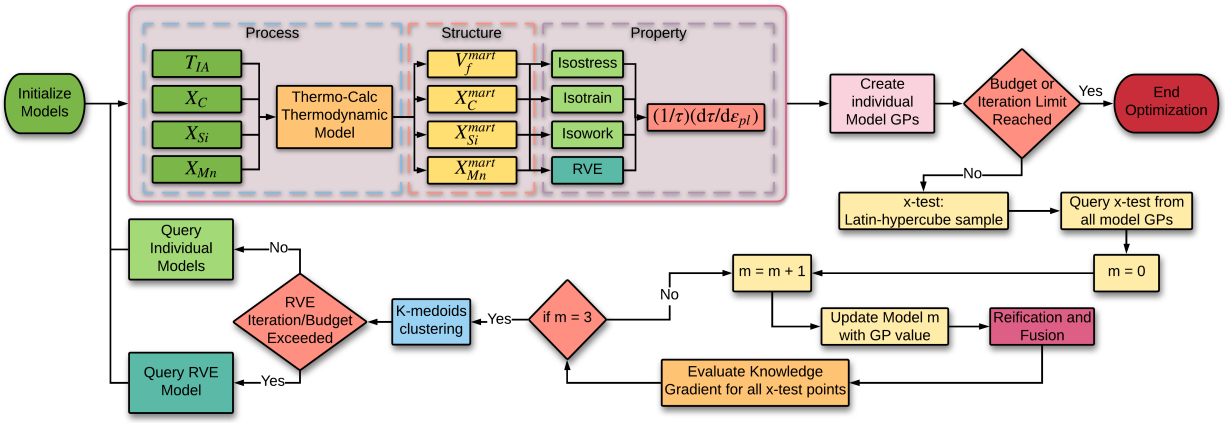


Figure 3.2: Schematic overview of the method applied in the current work

sampling in the bounds of the hyperparameter space. In the current work this space is defined as $l_h \in [0.01, 20]$, and $\sigma_f \in [0.01, 100]$. The noise variance hyperparameter was set to a constant value of $\sigma_n^2 = 0.1$. A total of 500 different hyperparameter sets were defined, and these were kept constant throughout the optimization process. After defining the hyperparameter sets, we select two random points within the design space. These two points are queried from all three reduced-order models and the RVE model as initial data.

In the following discussion, the reduced-order models will be collectively referred to as the object $\boldsymbol{Models}$. Where each reduced-order model is indicated by an index in the set $[1, 2, 3]$. For

example, $\boldsymbol{Models}[\ \mathbf{1}\ ]$ refers to the isostrain reduced-order model. These model objects contain the X and Y data that has been evaluated for that particular model. Using this data, it is possible to construct a GP for that model. These GPs are denoted as $\boldsymbol{Models}[\ \cdot\ ]\boldsymbol{\_GP}$. In the current work, we assume that the hyperparameters defining the GPs of these models are known *a priori* as their extremely low computational cost makes their full evaluation over the design space highly feasible.

At the start of each iteration, a set of $\boldsymbol{x}_{test}$ vectors are defined using Latin-hypercube sampling of the input space. The number of samples generated is increased after every iteration that calls the ground truth model. This ensures that, as the optimization progresses, the process is capable of finding finer spaced points. The design space was limited to contain only points with a volume fraction of martensite less than 0.9, and a Random Forest (RF) classifier was trained to remove test points that did not meet this criterion.

Algorithm 3

Input: $Models$, x_{test}
Output: {max $KG(Models, HP, x_{test})$, arg max $KG(Models, HP, x_{test})$}

1: **for** i=1,2,3 **do**
2: **for** j=1,2,...,length(x_{test}) **do**
3: $y = Models[\ i\ ]\_GP(x_{test}[\ j\ ])$
4: Update $Models[\ i\ ]$ with $(x_{test}[\ j\ ], y)$
5: **for** k=1,2,...,hp_count **do**
6: Estimate model Correlation (*Reification*)
7: Fuse $Models \rightarrow (x_{fused}, y_{fused})$
8: Build $Fused\_GP(\sigma_n^k, l_1^k, l_2^k, x_{fused}, y_{fused})$
9: Evaluate $KG(Fused\_GP(x_{test}))$
10: **end for**
11: **end for**
12: **end for**

After defining the test points to be used, Algorithm 3 is used to calculate the Knowledge Gradient for each combination of hyperparameters, test point, and model. The outermost loop (line 1) runs the full set of calculations for each of the 3 reduced-order models, while the next loop (line

2) runs the calculation for each of the x_{test} values defined at the beginning of the iteration. The final loop (line 5) is used to obtain results using different combinations of hyperparameters. This creates a matrix of results that have the maximum knowledge gradient, reduced-order model index, $\boldsymbol{fused\_GP}$ hyperparameter index, and x_{test} index.

Algorithm 4

1: $def:\ \boldsymbol{x}_{init}$
2: $calc:\ y = Models(\boldsymbol{x}_{init})$
3: $calc:\ y = RVE(\boldsymbol{x}_{init})$
4: **for** i = 1,2,...,n_{iter} **do**
5: $[\nu^{KG},\ \arg(\nu^{KG}),\ Model]$:= Algorithm 1
6: $\boldsymbol{x}_{medoids}$ = K-Medoids Clustering $[\nu^{KG},\ \arg(\nu^{KG}),\ Model]$
7: **if** Iteration/budget $\geqslant$ Limit **then**
8: $calc:\ y = RVE(\boldsymbol{x}_{medoids})$
9: update RVE GP
10: **else**
11: $calc:\ y = Models(\boldsymbol{x}_{medoids})$
12: update Model GPs
13: **end if**
14: **end for**

Algorithm 4 shows the entire iteration process, and as shown in Line 6, the next stage in the process is to cluster the output from Algorithm 3. The clustering is done in 3-dimensional space defined by the Knowledge Gradient value, the model index, and the index of the x_{test} value. This is done to increase the distance between the points to be queried as much as possible and to reduce the likelihood that the process will only select a single model at every iteration. The final stage of the iteration involves calling the models. At this stage, a decision is made on whether to call the RVE model or not. If the conditions are not met for calling the RVE model, the medoids are used to query the reduced-order models. Since each medoid contains a reduced-order model index and an x_{test} index, these are used to query the corresponding model and test point. If the conditions have been met for calling the RVE model, then all x_{test} points contained in the medoids are queried from the RVE model.

Each model has an individual cost (measured in computer clock units) associated with doing a single calculation, and for the current work, the cost of querying a larger batch size is considered to be the cumulative time of completing that number of calculations from that model. In other words, the current work does not consider any discount for using larger batch sizes. In many experimental or computational situations, it would be likely that there would be a discount for using a larger batch size, so this assumption is potentially a conservative one.

In addition to the model calculation costs, there is an iteration cost (again calculated as the computer clock time) associated with calculating and updating the Gaussian Processes, as well as calculating the Knowledge Gradient. This cost and the individual model costs are used to calculate the total cost of the process. In contrast to the multiplication of the cost of model calculations, the time for the calculations is considered to be constant no matter what the batch size. The justification behind this is that all the calculations are done in parallel. In the event of multiple models being called in a single iteration, the calculation time is considered to be the time cost of the longest-running model.

One of the challenges of using this kind of approach is that there is no single fused model to use for the predictions of the maximum normalized strain hardening rate. Therefore, the maximum value predicted by the optimization is taken as the maximum normalized strain hardening rate found from calculations of the RVE model.

3.2.3 Optimization Case Studies

We considered three optimization case studies in the present work. These case studies change the utility function used as well as the conditions under which the ground truth is called and provide different termination criteria for the optimization process. As already mentioned, in these optimization case studies, the cost is considered to be the computation clock time of the models:

- *No Cost Constrained (NCC) Optimization*: The no-cost optimization used the knowledge gradient as the acquisition function with no adjustments and all queries to the RVE model were iteration based. After 25 iterations of updating the low order models, the next iteration

would update the RVE model. This iteration limit was chosen arbitrarily, and should not be taken as an optimum setting. Finally, the optimization process was terminated after 200 iterations.

- *Cost Constrained and Iteration Controlled Ground Truth Query (CC-IC) Optimization*: In this optimization scenario, the cost (computational clock time) of the low order models is considered when calculating the acquisition function. The cost-adjusted acquisition function is defined as,

$$\nu_{cost}^{KG} = \frac{\nu^{KG}}{\text{model cost}} \tag{3.26}$$

where ν^{KG} is the knowledge gradient value and model cost is the cost of the model in question. Additionally, a cost-based termination criterion is also included. This will stop the process from continuing once a total budget has been exceeded. This is to emulate a scenario where a project has a budget limit. The costs that contribute to this limit are both the cost of running the models as well as the cost (computational clock time) of calculating the next best points to query. This approach also queries the RVE after 25 iterations of updating the low order models. Additionally, this approach was also run with iteration limits of 10 and 50 for the calling of the RVE. In total 15 of the calculations were completed for this comparison.

- *Cost Constrained and Cost Controlled Ground Truth Query (CC-CC) Optimization*: The final approach uses the cost-adjusted acquisition function but considers two budget constraints. The first is that when the cost of iterations exceeds the RVE Budget amount, the RVE Model will be called. After the RVE model is called, the RVE Budget is replenished. The second constraint is that the process is terminated if the total cost exceeds the Total Budget. In both cases, the model and process costs are considered when calculating the cost of an iteration. Again, all costs are assumed to be the computational clock time.

- *Sequential Bayesian Optimization* A conventional sequential Bayesian Optimization of the RVE Surrogate model only was conducted to enable the assessment of whether the framework in the current work is performing better. This optimization was done by initializing a GP using the same initial RVE data as used for the Batch Optimization approach and then using the Knowledge Gradient evaluated at 500 samples from the design space to determine the next best point to evaluate. The samples were obtained by Latin hypercube sampling and the number of samples was incremented by 1 with each iteration. This calculation was completed for all the initial datasets that were used for the batch optimization and the results were averaged.

3.3 Results and Discussion

The current work aimed to maximize the normalized strain hardening rate of a dual-phase steel. This was done by optimizing the carbon content (wt%) and intercritical annealing temperature. For the analysis of the results, a normalized strain hardening rate of 30 or greater is considered an optimum result. The results presented show the maximum normalized strain hardening rate found from the RVE model compared against the number of iterations, computational cost, and time for the optimization. The shaded regions of the plots indicate the 95% confidence interval calculated from the results of 20 optimization calculations for each batch size. All the batch optimization results are compared with the sequential Bayesian Optimization of the RVE surrogate model only.

The first result shows the maximum normalized strain hardening rate found from the RVE model against the number of iterations of the optimization routine, Figure 3.3. In all three cases, it can be seen that the larger batch sizes (batch sizes 5 and 7) lead to faster optimization of the normalized strain hardening rate. In both the cost-constrained cases, these large batch sizes end quickly since the increased number of calls to the ground truth function exhaust the available budget quicker. However, when considering the comparison with the sequential optimization of the RVE surrogate model, only the batch size of 7 performs as well as the sequential optimization of the RVE. While these results show a benefit for using larger batch sizes, the number of iterations for the optimization is not necessarily the most useful comparison that can be used.

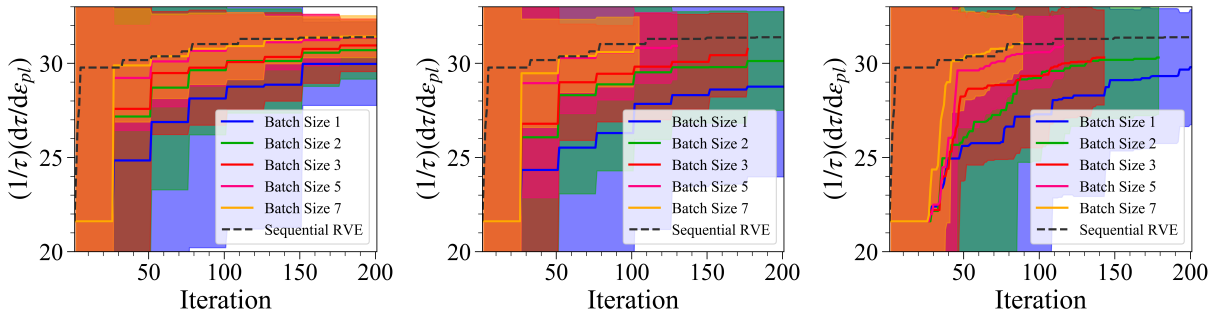


Figure 3.3: Maximum normalized strain hardening rate achieved from RVE calculations as a function of the number of iterations of the optimization process for (a) No Cost Constraint, (b) Cost Constrained - Iteration Controlled and (c) Cost Constrained - Cost Controlled optimization cases. The shaded regions of the plots indicate the 95% confidence interval calculated from the results of 20 optimization calculations for each batch size and the mean from the sequential RVE optimization calculations is also shown.

When the maximum value of the RVE calculations is compared with the total cost of the optimization (Figure 3.4), the downside of using the large batch sizes is observed. In these figures, we can see that, *at the beginning of the optimization*, the larger batch sizes result in a much larger cost much more quickly and that the lower batch sizes start optimizing at lower costs. However, considering the results after all approaches have called the RVE model at least once, the larger batch sizes are still able to achieve higher normalized strain hardening rates at a lower computational cost. This is particularly true when comparing the results with the sequential optimization of the RVE only. This shows that while there may be an advantage to using sequential optimization in terms of the number of iterations required, the cost of the optimization can be decreased by using the framework developed in this work.

The final consideration was how the maximum RVE value found changed with the time taken for the optimization (Figure 3.5). Here we can see that the large batch sizes manage to attain higher values significantly faster in real-time. This is the case, especially in the cost-constrained approaches. This could be due to the cost-constrained acquisition function favoring the cheaper (faster) models. The comparison with the sequential model shows the real significant advantage of the current approach, as it can obtain an optimum value significantly faster than the sequential

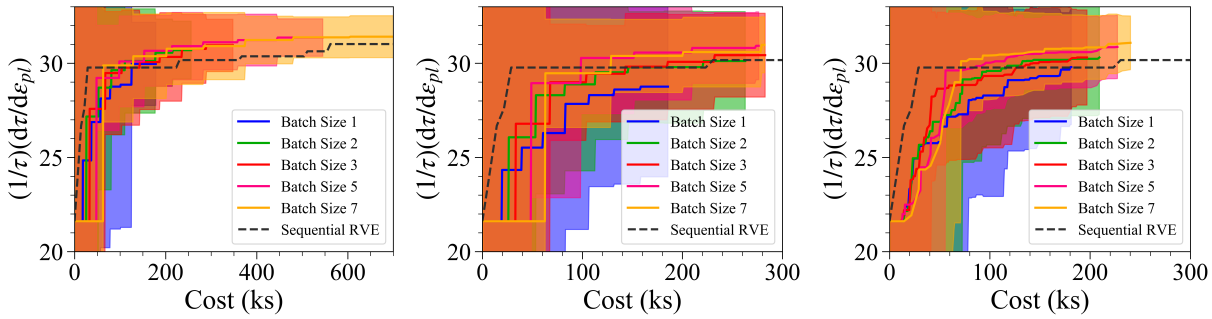


Figure 3.4: Maximum normalized strain hardening rate achieved from RVE calculations as a function of the total cost of the optimization process for (a) No Cost Constraint, (b) Cost Constrained - Iteration Controlled and (c) Cost Constrained - Cost Controlled optimization cases. The shaded regions of the plots indicate the 95% confidence interval calculated from the results of 20 optimization calculations for each batch size and the mean from the sequential RVE optimization calculations is also shown.

optimization.

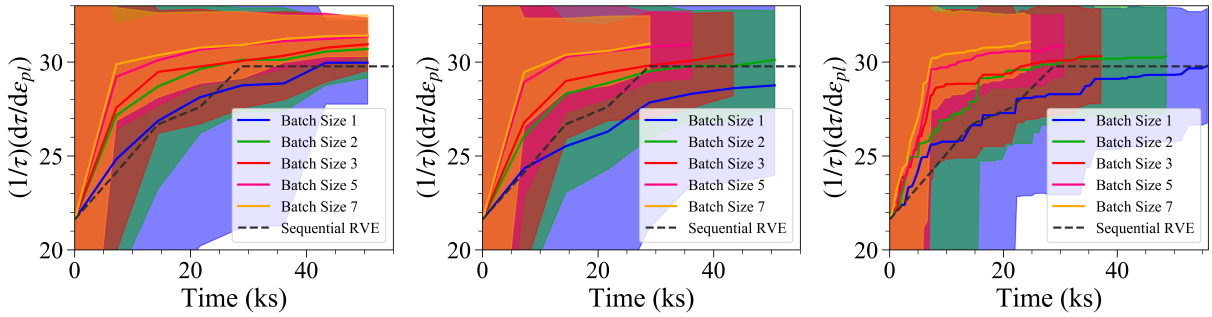


Figure 3.5: Maximum normalized strain hardening rate achieved from RVE calculations as a function of the total time of the optimization process for (a) No Cost Constraint, (b) Cost Constrained - Iteration Controlled and (c) Cost Constrained - Cost Controlled optimization cases. The shaded regions of the plots indicate the 95% confidence interval calculated from the results of 20 optimization calculations for each batch size and the mean from the sequential RVE optimization calculations is also shown.

It is of interest to compare the performance of our proposed BBO approach to that of a conventional BO, carried out by exclusively querying the ground truth—i.e., the RVE-based finite

element simulations. This is shown in Figure 3.6, which compares a sequential approach without model fusion to our model fusion-based approach with batch sizes 1 and 7. As in other cases, we include the uncertainty bounds resulting from running the optimization framework over the design space, multiple times. One noticeable aspect of this figure is the extremely large variance in the performance of the sequential BO approach, even at the latest stages of the optimization approach. This implies that there is considerable risk in employing such an approach as it seems to be highly dependent on the initial conditions—i.e., data—of the optimization process. The figure also shows that our BBO approach, with a batch size 7 results in a dramatic decrease *of an order of magnitude* in the time necessary to find the global optimum, with *much lower levels of variance*. This latter result is significant as BBO appears to be considerably less dependent on initial conditions, providing strong performance guarantees, at much faster rates.

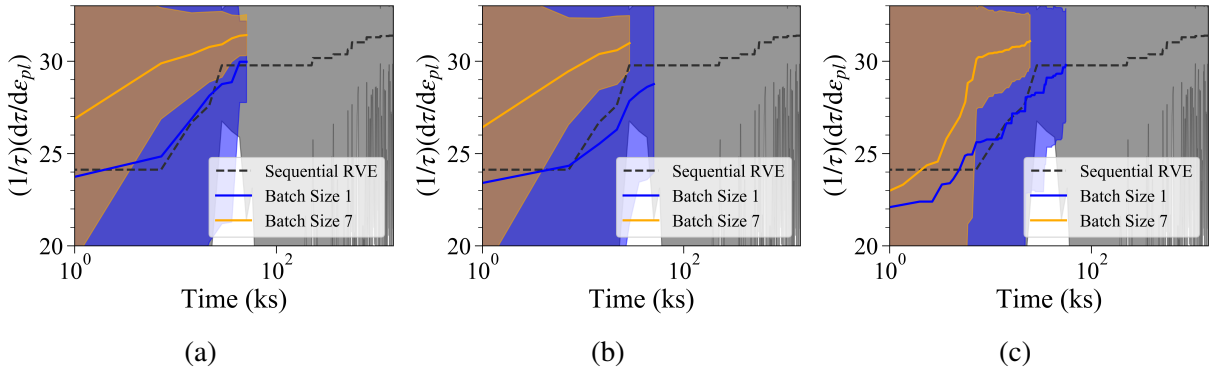


Figure 3.6: Maximum normalized strain hardening rate achieved from RVE-based sequential BO compared to BBO with batch sizes of 1 and 7, as a function of the total time of the optimization process for (a) No Cost Constraint, (b) Cost Constrained - Iteration Controlled and (c) Cost Constrained - Cost Controlled optimization cases. The shaded regions of the plots indicate the 95% confidence interval calculated from the results of 20 optimization calculations for each batch size

As an example of the potentially complex interplay between all the parameters of the framework, we consider the results of using different iteration limits for calling the RVE model in the CC-IC approach. The full results for all batch sizes are contained in Appendix A, however, presented here is a comparison of the maximum RVE values found at the termination of the opti-

mization procedure, Figure 3.7. To obtain these results, the CC-IC approach was calculated for 15 unique initial datasets. This was repeated for each of the three iteration limits. The data in the plot shows the mean and 95% confidence interval calculated for these 15 calculations for each iteration limit.

The first trend that can be noticed is that there is a general increase in the maximum value of the normalized strain hardening rate correlated to the increase in batch size. This happens with all iteration limits. When we consider the final iteration number in Figure 3.7a we can see that in general, as the batch size increases the final iteration number decreases. This is a result observed previously and is linked to the increased cost of the larger batch size. What we can also observe by looking at each of the 5 groups is that the final iteration number is negatively correlated with the iteration limit. This result makes intuitive sense since a lower iteration limit will mean more RVE calculations which will result in a higher cost. Thus, the budget limit will be reached faster.

However, if we consider the total cost effects in Figure 3.7b, the cost of using the lower iteration limit remains fairly consistent. However, the difference between the costs of the optimizations with different iteration limits decreases as batch size increases. If we couple this observation with the results in Figure 3.7c we can observe that while the cost is staying fairly constant for the 10 iteration limit case, the time taken for the optimization is decreasing rapidly as the batch size increases. The decrease in the time taken for the optimization is not as evident in the 25 and 50 iteration limit cases, however, this could be due to the 200 iteration limit placed on the optimization.

All these results indicate that there is a fairly complex correlation between the batch size and the optimum iteration limit for calling the RVE model. However, as has been noted, it is not likely that the optimum combination will be possible to determine when considering more costly functions, but the correlation appears to be that for larger batch sizes, smaller iteration limits for calling the “ground truth" decreases the time taken for the optimization process. In addition, while the smaller iteration limit does increase the cost compared to a larger iteration limit, this difference decreases with batch size.

The assumption that the cost of batch BBO scales linearly with the batch size, relative to an

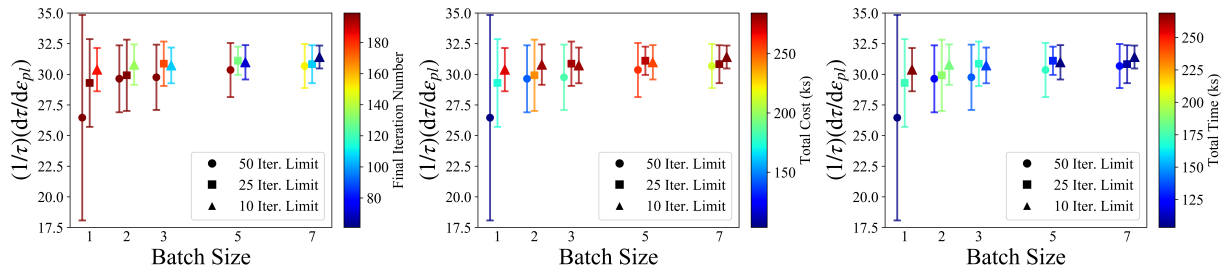


Figure 3.7: Comparison of final iteration, total cost and total time for the optimization approach in the current work when changing the iteration limit for calling the RVE model. The results shown have coloring indicating a) Final Iteration Number, b) Total Cost, & c) Total Time. The results show the mean result at the end of the optimization using 15 sets of unique initial conditions, and the error bar shows the 95% confidence interval for the final prediction.

exclusively sequential approach is the most conservative one that can be made. This cost scaling is likely to operate mostly when a design space is being explored using exclusively computer simulations—the cost (in computer-time) depends linearly on the number of *parallel* simulations. However, in real-world experimental approaches, high-throughput, or parallel, processing often has a reduced cost per experiment compared to sequential counterparts. These economies of scale arise from the simplification of ancillary activities associated with batch experimental operations—there are many activities for which the cost is the same regardless of the batch size. For example, we can consider the case of combinatorial synthesis through batch arc melting, in which it is necessary to load the feedstock and evacuate the chamber before melting, and wait for the samples to cool down before removal. Each of those steps will be shorter, per-sample, the more samples are simultaneously melted in a single run, with the ultimate cost per synthesized specimen being lower.

From the results shown above, the fact that the framework is capable of achieving better results at a lower cost under the conservative cost model that assumes no economies of scale indicates that this approach could have significant effects on the process cost for developing new materials or optimizing existing materials. The benefits of such Batch BO frameworks may indeed be much more evident in experimental settings than in computational ones. A benefit that is perhaps more

difficult to quantify but that can potentially be even more significant is the beneficial impact on the net present value of any materials development campaign: the faster that one arrives at a potential optimal materials solution the greater the value that one can extract out of such a development effort. To reiterate this point: even if the total cost of a BBO-based materials development campaign were the same as that of a sequential approach, arriving at the optimal solution in much shorter times is extremely beneficial. Coupling a design framework like that developed in the current work with high-throughput experiments has the possibility of further reducing the time and cost of materials development.

We note that in this work we tuned the hyperparameters for the GPs used to emulate the reduced-order models *a priori*. The cost of evaluating these models is orders of magnitude [14] lower than that of evaluating the RVE and it is thus practical to exhaustively explore the input space of these models before the model fusion BBO is carried out. In cases in which the cost of querying even the 'cheap' information sources is non-negligible, it may be necessary to modify our proposed framework. For example, we could implement the BBO routine in two stages: one (perhaps at much larger batch sizes) for fitting the reduced-order models themselves, followed by a second application of BBO for the fused model.

3.4 Summary and Concluding Remarks

The results from the current work show significant promise in the use of Batch Bayesian Optimization frameworks within an ICME methodology for materials design. Most notably, the results showed that using larger batch sizes resulted in the quantity of interest being optimized in a shorter time and at a lower cost than when using smaller batch sizes. This confirms an intuitive understanding that by adding more information on each iteration we can gain a better knowledge of the system under optimization in much shorter times.

Here, we implemented a model fusion-based BBO approach and applied it successfully to the optimization—through linked computational PSPP relationships—of the 'formability' of dual-phase steels by tailoring the chemistry and processing conditions. The results indicate that using batch optimization can greatly decrease the time and cost of the process while simultaneously

reducing the uncertainty of the predictions. This has important implications for ICME-enabled design of materials as well as for exclusively-experimental materials discovery and design.

We provided further arguments for the benefits of this approach by pointing out that arriving at the answer faster than when using sequential approaches may supersede any consideration of the cost associated with the (computational or experimental) querying of the materials space. The reduction in time necessary to complete the alloy development process would have a very positive impact on the net present value (NPV) of the development effort, minimizing risks while maximizing the potential future benefits of deploying a material in a specific technology.

We note that there is still much work that can be done to improve the framework, particularly to make it more applicable when the hyperparameters of the reduced-order model and the fused model GPs are not known. The authors do acknowledge that the current results might not be generalizable to all applications of the framework. Therefore, work is being conducted to test the framework using standardized test functions. This will allow for full benchmarking of the results from this framework. In addition to this, while we have demonstrated the effect of changing the iteration limit, there are numerous other framework parameters (for example, the acquisition function, GP hyperparameter ranges and covariance function) that have not yet been tested to ascertain their effect on the optimization process.

In fact, while in this work we have carried BBO over the hyperparameter space with a fixed covariance structure (i.e., Matérn kernel) and acquisition function (i.e., Knowledge Gradient), it may be possible to extend this approach over the model as well as the acquisition function space. This would follow the spirit of the current BBO approach, following the premise that, at the beginning of an optimization, it is not certain what type of covariance structure or even what type of policy is most effective for a given problem space.

4. BATCH REIFICATION/FUSION OPTIMIZATION (BAREFOOT) FRAMEWORK

4.1 Introduction

Integrated computational materials engineering (ICME) [1] calls for the integration of materials modeling across scales to aid in materials-by-design. One of the significant challenges in the materials design process is the size of the potential design space. The design space's size means that any design approach must inevitably operate with a comparatively small amount of data.

To this end, Bayesian optimization techniques are robust, especially in the absence of large amounts of initial data, [38], [54]. Work by Talapatra et al. [54] even indicated that Bayesian optimization frameworks might work better in the absence of initial data. As such, Bayesian-based methods are an ideal candidate for the building of ICME frameworks. One of the challenges in ICME that is not addressed significantly is integrating experimental results into the optimization framework. The focus in much of the ICME literature is to use experimental results for validation, and verification [66] of the computational models. This is a critical role for experimental results since we acknowledge that any computational approach deviates from experimental results due to simplifying assumptions used in the computational method. However, in previous work by Ghoreishi et al. [13], [14] it has been demonstrated that it is possible to build frameworks capable of incorporating experimental results directly into the optimization approach.

In experimental materials science, one of the more recent developments has been high-throughput experimental approaches. These approaches initially focused on thin-film methods [27] since it was possible to make different material combinations easily. However, recent improvements in Additive Manufacturing techniques have opened the possibility of high-throughput [23] or batch [38] experimental approaches targeting bulk materials. In these approaches, and particularly the thin-film analysis, the experiments have usually been designed using combinatorial methods [22]–[25], [27], [38] to explore as much of the design space as possible. While high-throughput and batch approaches provide results significantly faster than conventional experimental methods, there

is a need to guide the high-throughput analysis in a way that reduces testing of areas in the design space that are of little value.

Several approaches to Batch Bayesian Optimization exist [28]–[32] that would bridge this gap and provide batches of recommendations from a Bayesian Optimization approach. However, one of the more promising approaches, due to its simplicity, is the approach proposed by Joy et al. [33]. In a Bayesian approach, particularly when using Gaussian Processes (GPs) as the surrogate models, the choice of the hyperparameters for the covariance function determines the shape of the surrogate model and hence assumes the underlying function shape. The batch optimization approach proposed by Joy et al. avoids this by sampling the hyperparameter space extensively. This avoids assuming the shape of the Function. However, each set of hyperparameters will also have a different maximum in either the surrogate model or the acquisition function evaluated on the surrogate model. Clustering these predictions for the maximum of either the surrogate or acquisition function allows for a batch of predictions to be generated from the process.

From the discussion above, two issues need to be addressed: the integration of experimental results with computational methods, and secondly, the prediction of a batch of next-best points. The Framework presented in the current work addresses both of these issues by combining the Reification/Fusion approach presented in the work by Ghoreishi [13], [14], and the batch optimization approach proposed by Joy et al. [33] into a single framework. A framework was built in Python to achieve the combined objective of reification/fusion and batch optimization. This Framework is flexible and can be implemented with any number of reduced-order models (this will be limited by the amount of memory available). The Framework can also conduct the Batch Bayesian Optimization approach on a single model if the parameters are set correctly.

An example of this Framework being used has already been presented in our previous work [67]. The purpose of the current work is to present the Framework in more detail, particularly to delve into the values for the parameters of the Framework and demonstrate the operation of the Framework using two different objective functions. We start this explanation of the Framework by expanding on each of the Framework's techniques and demonstrating how the implementations are

coded. The final sections of this work discuss the framework parameters that can be manipulated, with results demonstrating how these parameters can affect the Framework's operation.

4.2 Framework Description

This work aims to show how the Batch Reification/Fusion Optimization (BAREFOOT) Framework has been constructed. The Framework is shown schematically in Figure 2.3. To summarize, the Framework starts by initializing the Truth Function and reduced-order models. These models are not required to be initialized with the same input data, although the current work tends to use this approach as a simplification. Using this initial data, two sets of models are created. Firstly, GP surrogate models of the reduced-order models are built. These are used to model the response of each of these models over the entire design space. In addition to these, a discrepancy model for each of the reduced-order models is created. This will be discussed in more detail later.

After this initialization stage, the batch optimization approach is started. This optimization is a combination of the reification/fusion approach developed by Thomison and Allaire and the Batch Bayesian Optimization approach developed by Joy et al. [33]. While each of these approaches, and the combined approach, will be discussed in more detail later, a summary of the approach is as follows. A fused model is built for multiple combinations of different hyperparameters (which can be GP hyperparameters or framework parameters) and an acquisition function evaluated. The acquisition function used most commonly in this work is the Knowledge Gradient [47]. However, any acquisition function used with conventional Bayesian Optimization approaches can be used.

For each combination of hyperparameters, the acquisition function's value is recorded, and then these results are clustered. The number of clusters formed is the batch size of the process, and for the current Framework, it is the maximum number of Truth Function queries that can be achieved in parallel. The clustering algorithm used in the Framework is the k-medoids algorithm, which finds the data point closest to the cluster's weighted center. After the medoids are defined, the models are queried, and the reduced-order surrogate and discrepancy models are updated before checking whether the termination criteria for the optimization have been met.

This concludes the brief overview of the entire process. The next sections will explain each of

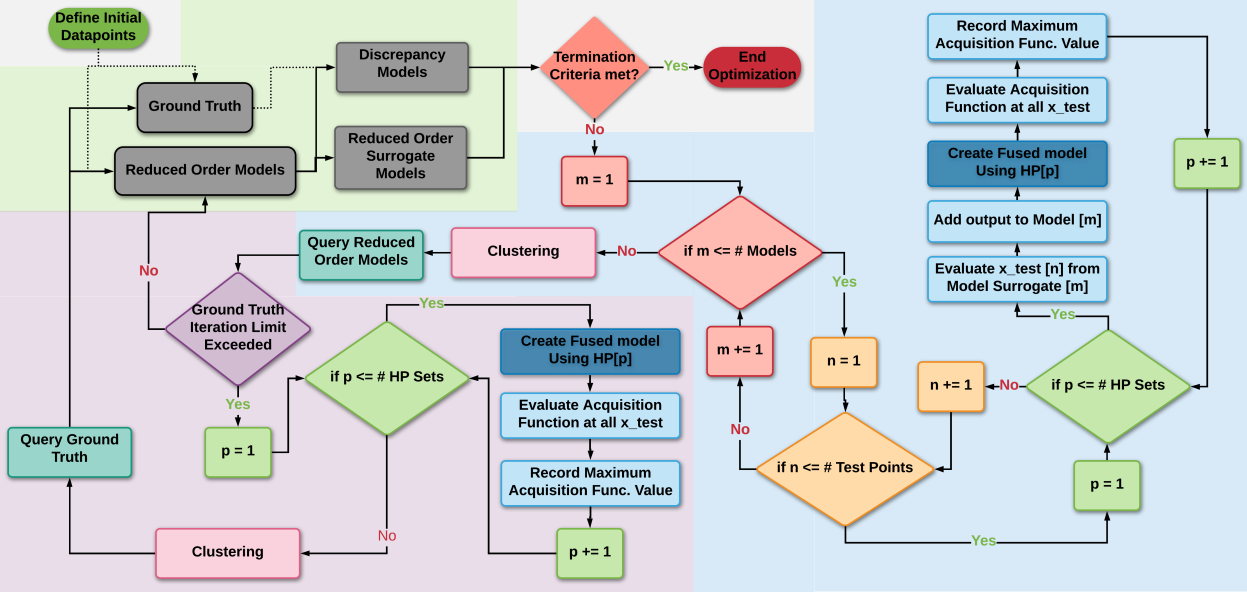


Figure 4.1: Schematic of the BAREFOOT framework

the components of the Framework. These more detailed descriptions of the framework components will follow the sequence shown in the schematic.

4.2.1 Models

The BAREFOOT Framework is designed to maximize a quantity of interest using multiple reduced-order models and a single ground truth. In this case, the ground truth is the most accurate model or experiment for the quantity of interest. Much of the framework testing has been done using a finite element model as the ground truth. However, the Framework can just as easily be applied to an experimental ground truth. In the remainder of this work, we will refer to the Truth Function, where the use of Function indicates that the ground truth can be either experimental testing or a high accuracy computational model.

In practice, we expect that queries to the Truth Function will be very costly, both in terms of the time it takes to obtain the result and the monetary cost of the evaluation. The Framework utilizes two or more reduced-order models that are cheaper than the Truth Function to alleviate this cost.

The reduced-order models chosen should model the quantity of interest with the same inputs as the Truth Function. These reduced-order models, by their definition, will have lower fidelity

than the Truth Function. And in this work, fidelity is defined as the model's accuracy over the entire design space. Using this definition, we acknowledge that some reduced-order models may be significantly more accurate in some regions of the design space than in others. However, a high fidelity model will be considered a model that closely models the maximum of the Truth Function.

The first stage of the Framework is to initialize the models. In this case, we assume that we evaluate each reduced-order model and the Truth Function a small number of times to obtain initial data. Therefore, we can either utilize a random sampling of the design space or existing knowledge of the system, particularly if there are existing results from the Truth Function. The knowledge already gained about the system could be a significant factor in how the optimization framework proceeds. We will discuss this in further detail later in this work.

4.2.2 Surrogate Modeling and Discrepancy Models

After obtaining the initial data from all the reduced-order models and the Truth Function, the first stage of the Framework is to define surrogate and discrepancy models for each of the reduced-order models. The surrogate models of the reduced-order models are required by the Bayesian optimization approach used in the Framework. The discrepancy models measure how closely the reduced-order models match the results of the Truth Function.

Bayesian optimization relies heavily on the construction of surrogate models. The aim is to predict the model's mean, with a degree of uncertainty associated with the prediction, in areas where the model has not been evaluated yet. There are several options for surrogate models. However, Gaussian Process models are considered one of the most common. This is due to their flexibility and ease of use. In addition, the outputs of Gaussian Process models are normally distributed, which simplifies the mathematics of some of the other aspects of the Framework, and thus are the chosen surrogate model here. However, this is not a requirement of the Framework.

4.2.3 Gaussian Process Surrogate Modeling

In work by Rasmussen and Williams [58], a Gaussian Process is defined as a collection of random variables, any finite number of which have a joint Gaussian distribution. By defining

a mean function ($\mu(\cdot)$) and a covariance function ($C(\cdot,\cdot)$), it is possible to specify a Gaussian process completely, and for this purpose, we define them as,

$$\mu(\boldsymbol{x}) = \mathbb{E}\left[f(\boldsymbol{x})\right]. \tag{4.1}$$

$$C(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathbb{E}\left[(f(\boldsymbol{x}_i) - \mu(\boldsymbol{x}_i))(f(\boldsymbol{x}_j) - \mu(\boldsymbol{x}_j))\right]. \tag{4.2}$$

Therefore, we can write the Gaussian Process as,

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}), C(\boldsymbol{x}_i, \boldsymbol{x}_j)). \tag{4.3}$$

A more detailed explanation of this kind of stochastic process is provided in Rasmussen and Williams's work [58].

4.2.4 Model Discrepancy

In the current Framework, the aim is to utilize multiple models, and we assume that each of these models will have a different level of fidelity. In this context, we define fidelity as how accurate the model is, or in other words, how accurately the model predicts the Truth Function value. We measure the model fidelity by defining the model discrepancy, which is the difference between the reduced-order model and the Truth Function. Since we do not know the actual Truth Function values for the entire design space, we can never know the model fidelity over the entire design space. However, we estimate the model fidelity by measuring the discrepancy between the reduced-order model and the Truth Function, *where the Truth Function has been evaluated*. Using this data, we fit a GP model to the discrepancy of the model.

The approach for defining the discrepancy of the models is shown in Figure 4.2. As shown in part (a) of Figure 4.2, the Truth Function has been evaluated at four different points, while the reduced-order model is evaluated at six. In the next stage, we fit a surrogate model to the reduced-

order model and then calculate the discrepancy (δ_i) at each of the points that have been evaluated for the Truth Function (Figure 4.2 (c)). A separate surrogate model is fit to these discrepancy values to predict the discrepancy over the entire domain, and the predicted discrepancy is added to the standard deviation of the reduced-order surrogate model. The resulting uncertainty bounds ($\sigma_{GP} + \sigma_d$) from the total uncertainty encompasses all of the Truth Function points (Figure 4.2 (c)).

Using this information, we define a total uncertainty of the model as a combination of the standard deviation calculated for the Gaussian Process model prediction (σ^i_{GP}) and the discrepancy as calculated from the difference between the information source and the Truth Function (σ^i_d). Therefore,

$$\sigma^i = \sigma^i_{GP} + \sigma^i_d \tag{4.4}$$

where the superscript refers to the reduced-order model. We use this total uncertainty for the standard deviation of the models in the Reification and Fusion approach.

4.2.5 Model Reification and Fusion

The model fusion approach aims to generate a fused model that without loss of generality, assuming two models are to be fused, can be represented by the equation:

$$y = k_1(x^*)f_1(x^*) + k_2(x^*)f_2(x^*) \tag{4.5}$$

where $k_1(x^*)$ and $k_2(x^*)$ are real-valued scalar quantities subject to $k_1(x^*) + k_2(x^*) = 1$. While the two models, $f_1(x)$ and $f_2(x)$, are assumed to estimate the quantity of interest (y) with some total uncertainty (δ_i),

$$y = f_1(x) = \bar{f}_1(x) + \delta_1(x), \tag{4.6}$$

$$y = f_2(x) = \bar{f}_2(x) + \delta_2(x), \tag{4.7}$$

where $\bar{f}_1(x)$ is the mean prediction and the model uncertainties $\delta_i(x)$ are assumed to be normally

distributed with, $\delta_1(x) \sim \mathcal{N}(0, \sigma_1^2)$ and $\delta_2(x) \sim \mathcal{N}(0, \sigma_2^2)$.

By assuming that both the models have a normal distribution given by $f_1(x^*) \sim \mathcal{N}(\bar{f}_1(x^*), \sigma_1^2)$, and $f_2(x^*) \sim \mathcal{N}(\bar{f}_2(x^*), \sigma_2^2)$, it is possible to solve Equation 4.5 for $k_1(x^*)$ and $k_2(x^*)$ by solving the minimization problem:

$$\min_{\mathbf{k}} \ \mathbf{k}^T \Sigma \mathbf{k} \ \text{ subject to } \ k_1 + k_2 = 1 \tag{4.8}$$

where $\mathbf{k} = [k_1, k_2]^T$ and

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \bar{\rho}\sigma_1\sigma_2 \\ \bar{\rho}\sigma_2\sigma_1 & \sigma_2^2 \end{bmatrix}. \tag{4.9}$$

The covariance matrix, Σ, requires the correlation coefficient, ρ. The purpose of the Reification approach is to estimate this correlation coefficient. In the current work we use the approach outlined by Winkler [52] to define the solution to the minimization problem in Equation 4.8 and define a fused model for y that has a mean defined by:

$$\mathbb{E}[y] = \frac{(\sigma_2^2 - \bar{\rho}\sigma_1\sigma_2)\bar{f}_1(x^*) + (\sigma_1^2 - \bar{\rho}\sigma_1\sigma_2)\bar{f}_2(x^*)}{\sigma_1^2 + \sigma_2^2 - 2\bar{\rho}\sigma_1\sigma_2} \tag{4.10}$$

and variance

$$\text{Var}(y) = \frac{(1 - \bar{\rho}^2)\sigma_1^2\sigma_2^2}{\sigma_1^2 + \sigma_2^2 - 2\bar{\rho}\sigma_1\sigma_2}. \tag{4.11}$$

However, this solution requires the correlation coefficient. Since it is generally impossible to know the true correlation between models, we use an approach called reification [15] to estimate the correlation between the models. First, we "reify" model 1, which means that we assume that model 1 is the Truth Function. Since model 1 has been reified, the standard deviation of model 1 ($\tilde{f}_1(x^*)$) at a single point in the design space (x^*) is defined simply by the model uncertainty as:

$$\tilde{f}_1(x^*) = f_1(x^*) - \bar{f}_1(x^*) = \delta_1(x^*), \tag{4.12}$$

and the error of model 2, with respect to model 1, can be defined as:

$$\tilde{f}_2(x^*) = f_2(x^*) - \bar{f}_2(x^*) \tag{4.13}$$

$$= \bar{f}_1(x^*) - \bar{f}_2(x^*) + \delta_1(x^*). \tag{4.14}$$

Using these two errors, we then calculate the mean squared error:

$$\mathbb{E}[\tilde{f}_1(x^*)^2] = \mathbb{E}[\delta_1(x^*)] = \sigma_1^2, \tag{4.15}$$

$$\mathbb{E}[\tilde{f}_2(x^*)^2] = \mathbb{E}[(\bar{f}_1(x^*) - \bar{f}_2(x^*))^2] + \mathbb{E}[\delta_1(x^*)] \tag{4.16}$$

$$= (\bar{f}_1(x^*) - \bar{f}_2(x^*))^2 + \sigma_1^2, \tag{4.17}$$

and the covariance:

$$\mathbb{E}[\tilde{f}_1(x^*)\tilde{f}_2(x^*)] = \sigma_1^2. \tag{4.18}$$

The Pearson correlation coefficient (ρ) is then calculated using:

$$\rho_1(x^*) = \frac{\sigma_1^2}{\sigma_1\sigma_2} = \frac{\sigma_1}{\sqrt{(\bar{f}_1(x^*) - \bar{f}_2(x^*))^2 + \sigma_1^2}}, \tag{4.19}$$

where the subscript on the coefficient indicates which model has been reified. This process is repeated until every reduced order model has been reified and the Pearson coefficient for every pair of reduced order models has been calculated. The variance weighted average correlation ($\bar{\rho}$) is used in the model fusion approach and is calculated using the following:

$$\bar{\rho}(\boldsymbol{x}^*) = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2}\rho_1(\boldsymbol{x}^*) + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\rho_2(\boldsymbol{x}^*). \tag{4.20}$$

While the reification/fusion approach can, theoretically, be expanded to any number of models,

there are practical limits to how many models can be considered. These limits will be determined by the available computational resources and the time required to compute the pairwise correlations. However, since the number of reduced-order models is likely to be modest in most practical cases, this computational limit will likely not be reached.

4.2.6 Batch Bayesian Optimization

The batch optimization approach proposed by Joy et al. [33] has shown significant promise for use with the reification/fusion approach above. Since the fusion process produces a surrogate model of the fused model, the method is well suited to a Bayesian Optimization scheme. One of the most significant challenges in BO is the definition of the hyperparameters for the surrogate models used. The Batch Bayesian Optimization approach removes this challenge by sampling the hyperparameters from a defined range rather than specifying single values.

Figure 4.3 demonstrates how this batch approach is achieved. Given a small set of training points and a small sampling of hyperparameter values, we can generate multiple Gaussian Process models. Each of these GP models has a slightly different maximum. In addition to this, each model has a slightly different location for the maximum acquisition function value (in this case, the Knowledge Gradient [47]). The method proposed by Joy et al. [33] for batch optimization calculates the location of the maximum acquisition function for many different combinations of hyperparameters. It then clusters the locations to provide a batch of predicted next-best points.

4.2.7 Batch Reification/Fusion Approach

The Framework uses a combination of the Batch Bayesian Optimization and the Reification/-Fusion approaches. The BAREFOOT Framework varies three parameters when conducting batch calculations. The first is the index of the test point at which the reduced-order model GP is evaluated. The second is the reduced-order model index. The final parameter is the index of the hyperparameter set. The parameters and how they are used for the batch approach are shown in Algorithm 15.

The loops on lines 1 and 2 of Algorithm 1 show the steps associated with the Reification/Fusion

Algorithm 5

Input: $\chi = [Models, Hyperparameters, x_{test}]$
Output: $\{\max KG(\chi), \arg\max KG(\chi)\}$

1: **for** i=1,2,...,length($Models$) **do**
2: **for** j=1,2,...,length(x_{test}) **do**
3: $y = Models[\,i\,]\_GP(x_{test}[\,j\,])$
4: Update $Models[\,i\,]$ with $(x_{test}[\,j\,], y)$
5: **for** k=1,2,...,hp_count **do**
6: Estimate model Correlation (***Reification***)
7: Fuse $Models \rightarrow (x_{fused}, y_{fused})$
8: Build $Fused\_GP(\sigma_n^k, l^k, x_{fused}, y_{fused})$
9: Evaluate $KG(Fused\_GP(x_{test}))$
10: **end for**
11: **end for**
12: **end for**

approach. This approach takes each test point and evaluates it from one of the reduced-order model GPs. We then temporarily update that GP with the evaluated point as a training point. Using this temporarily updated GP, it is possible to calculate the fused mean and variance using the Reification/Fusion approach. We then repeat this process for each of the reduced-order models and each test point until we have evaluated every combination of the test point and reduced-order model. The difference between the standard reification/fusion approach and the BBO approach is the loop at line 5. Instead of using a single set of hyperparameters for constructing the fused model GP from the calculated fused mean and variance, we iterate through all the hyperparameter sets and construct a new GP for each set. For each of these GPs, we evaluate the acquisition function at all test points. At the end of this combined approach, a matrix of outputs is acquired. This matrix contains the maximum Knowledge Gradient value, the test point index (for the test point used to update the reduced-order model GP) associated with this value, the reduced-order model index, and the hyperparameter set index. The next step in the process is to cluster this data.

4.2.8 Clustering

Since the Batch Optimization approach suggests the next-best query for each of the hyperparameter sets used, it is necessary to narrow down the number of next-best queries. To do this, we

use a k-medoids [68] clustering approach. The reason for using k-medoids and not k-means [69] is that k-means clustering provides the weighted center of each cluster, which may or may not be an evaluated point. In contrast, k-medoids clustering provides the evaluated point that is the center of the cluster. Additionally, depending on the constraints in the optimization problem, the centroid obtained by k-means could violate the constraints while we know that the evaluated points do not. Therefore, using k-medoids ensures that no additional computation is required and that we don't violate any constraints.

The current work does not cluster on the acquisition function value only. The clustering is done considering the acquisition function, the reduced-order model index, and the input values. By doing this, we aim to reduce the chance that the same input value is selected multiple times in a single batch. The aim is also to diversify the queries between the models since the Framework has shown a tendency to choose a single model regularly. These measures do not eradicate the occurrence of these situations. However, they are successful in reducing them.

The k-medoids method assumes that there is a set of objects that can be denoted as $X = \{x_1, x_2, \ldots, x_n\}$. Further, the distance between objects x_i and x_j is defined as $d(i, j)$. Each cluster will be defined by a single representative object (medoid). Therefore, a set of representative objects is defined by $Y = \{y_1, y_2, \ldots, y_n\}$ where y_i is a one-zero type object that takes on a value if one of the objects is selected as a medoid, and zero if not selected.

The second set of one-zero type objects is defined by variables z_{ij}, which indicates whether object x_j has been assigned to the cluster with medoid y_i. The k-medoids approach aims to partition the objects in X into clusters to solve the minimization problem;

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} d(i, j) z_{ij}, \tag{4.21}$$

subject to the following constraints,

$$\sum_{i=1}^{n} z_{ij} = 1 \qquad j = 1, 2, \ldots, n, \tag{4.22}$$

$$z_{ij} \leq y_i \qquad i, j = 1, 2, \ldots, n, \tag{4.23}$$

$$\sum_{i=1}^{n} y_i = k \qquad k = \text{no. of clusters}, \tag{4.24}$$

$$y_i, z_{ij} \in (0, 1) \qquad i, j = 1, 2, \ldots, n. \tag{4.25}$$

where the constraints are given in Equations 4.22 and 4.25 ensure that each object can only exist in a single cluster, while the constraint in Equation 4.23 ensures that objects can only be assigned to a medoid if that medoid exists. Finally, Equation 4.24 ensures that the number of clusters is correct.

4.2.9 Model Updating

There are two separate update steps. The first is when only the reduced-order models are evaluated, and the second when the Truth Function is evaluated. Each step is described below.

4.2.9.1 Reduced-order Model updating

The clustering step provides results that contain a value in the input space and the model associated with that value. The reduced-order model's update step takes each cluster's medoid and evaluates the reduced-order model at that medoid's input values.

4.2.9.2 Truth Function updating

The process's batch size is determined by how many parallel queries of the Truth Function can be evaluated. As such, the number of medoids will match exactly the number of evaluations of the Truth Function that can be run in parallel. Therefore, when updating the Truth Function, all the medoids are used to evaluate the Truth Function. This is done by extracting the input space values from the medoid and then using those inputs to query the Truth Function. However, further testing of the Framework has allowed for developing another method for querying the Truth Function. This second approach relies more significantly on the information contained in the fused models.

This second approach takes each of the hyperparameter sets, constructs the fused model, and then evaluates the fused model at "x" points in the design space. The points are defined by Latin hypercube sampling. Out of these "x" points, the one with the maximum value from the fused model is recorded with the input index. These two values are then clustered using the k-medoids approach, and the input values for evaluating the Truth Function are chosen. The reason for this changed approach is two-fold. Firstly, the number of test samples for the batch optimization is limited, and so the ability to thoroughly test the design space is equally as limited. Secondly, the original approach does not leverage the knowledge contained in the fused models. This new approach answers both questions, and because the points are clustered using the input value index, it is not purely exploitative. In other words, because we consider the input space index in the clustering step, the process will find maximum values in the fused models that are also relatively separated in space.

4.3 Framework Testing

Following our previous work where we demonstrated the use of the Framework, we did further testing to determine how the various framework parameters would affect the optimization process's outcome. While we did some testing in our initial work, this was also repeated to remove the iteration limit applied when we first evaluated the Framework.

4.3.1 Models used for testing

In the testing of the Framework, we used two sets of models. The first set [67] uses several models for predicting the mechanical response of a dual-phase microstructure. The second set of models was defined using a standard test function called the "Three Hump Camel". Using two different test sets aims to demonstrate that the Framework can operate more generally and that the results obtained are not specific to the test set used.

4.3.1.1 Dual-phase steel mechanical response models

This test set consists of four models. A surrogate model built on data from a representative volume element (RVE) model and three reduced-order mechanical models. In this test set, the

RVE Surrogate model is considered the Truth Function. We used the RVE calculation's surrogate to speed up the tests since the actual RVE calculations take several hours to complete. The reduced-order micromechanical models are as follows:

- *Isostrain:* The isostrain model assumes that the strain in both phases of the two-phase materials is the same.
- *Isostress:* The isostress model assumes that the stress is the same in both phases of the material.
- *Isowork:* The isowork model assumes that the mechanical work is the same for both phases of the material.

In all four models the aim is to predict the normalized strain hardening rate ($(1/\tau)(\mathrm{d}\tau/\mathrm{d}\epsilon_{pl})$), where τ is the mechanical stress and ϵ_{pl} is the plastic strain. This parameter is chosen in our design problem since it is an indicator of the formability of the dual-phase steel.

4.3.1.2 Three Hump Camel

The Three Hump Camel function is a standard test function, with maximum $f(0,0) = 0$, that is used for benchmarking of optimization approaches. This test function was selected because it is considered a relatively complex functionto maximize, but also because it is a two-dimensional problem that allows for easier visualization of the results. The function is defined as,

$$f(x_1, x_2) = -2x_1^2 + 1.05x_1^4 - \frac{x_1^6}{6} - x_1x_2 - x_2^2 \tag{4.26}$$

In the previous test set, the more accurate RVE model had existing reduced-order mechanical models used as the low fidelity sources. In the case of the Three Hump Camel function, there are no such reduced-order functions. As such, we constructed a selection of five reduced-order models out of Eq. 4.26. These equations are shown in the Appendix B. These models approximate the Three Hump Camel function to some degree, with some passing exactly through its true maximum. However, all of them contain maxima at locations in the design space that differ from the

true maxima of the Three Hump Camel function. This means that if any of these reduced-order models were used to make predictions in isolation, the location of the maximum found would not correspond with the location of the true maximum of the Truth Function.

4.3.2 Framework Parameters Available

We can adjust many parameters in the BAREFOOT Framework. The following list of parameters shows all the tested parameters in the two tests done in the current work. A short description of each parameter is provided. These do not constitute all the parameters that can be adjusted. However, this list does include all the significant parameters.

1. *Sample Count:* This parameter determines the number of points sampled from the design space for evaluating the acquisition function.

2. *Hyperparameter (HP) Count:* This parameter determines the number of hyperparameter sets generated by the Framework.

3. *Batch Size:* The number of evaluations that we can do in parallel for any of the functions. In the current Framework, this is set to the largest parallel query for the Truth Function.

4. *Truth Function (TF) Iteration Limit:* This is the number of iterations that must be completed, calling the reduced-order models, before the Truth Function is evaluated.

5. *Number of Fused Points:* When building the fused model, the fused mean and variance are evaluated at points uniformly sampled from the design space. This parameter determines the number of points per dimension at which the design space is sampled. In other words, the total number of samples will be this parameter raised to the power of the number of dimensions.

6. *Truth Function (TF) Cost:* Evaluating the Truth Function will incur some level of cost. This parameter defines that cost.

7. *Reduced-order Model (ROM) Cost:* This parameter is a vector containing the cost (computational or otherwise) for calling the reduced-order models. In the case of the mechanical model problem, this cost was defined by the computational time required for evaluating that model. However, in the Three Hump Camel problem, a base cost was defined for each of the models, and then this base cost was adjusted by an additional factor. This is referenced as ROM Cost Factor in the table of parameter values.

8. *Number of Reduced-order Models:* This parameter is used to define the number of reduced-order models that are used in the Framework. This was used to test the effect of changing the number of reduced-order models.

9. *Total Iteration Limit:* There are two approaches to terminating the Framework. This parameter is the first and limits the number of iterations that the Framework can run. When this amount is exceeded, the Framework terminates.

10. *Total Budget Limit:* This parameter is the second termination criteria for the Framework. This provides a total cost limit to the Framework. When this amount is *exceeded*, the Framework terminates. As a result of this decision, the total cost can exceed this budget amount, which might have implications when planning the optimization in a budget-constrained approach.

11. *Truth Function Budget Limit:* An alternate way of defining how long the Framework runs before calling the Truth Function is to provide a cost limit. This parameter defines the total cost of running the Framework (reduced-order model calls and calculation of the fused model and acquisition function) that must be expended before the Truth Function is called.

12. *Number of Initial Datapoints:* This parameter determines how many data points are used to initialize all the models. In the current implementation, this is constant for all models. However, in practice, this parameter could be ignored to allow all the models to be initialized with existing data for the model.

13. *Hyperparameter Lower Bound:* The minimum value that the hyperparameter can take. In the current iteration of the Framework, this specifically refers to the signal variance and length scale hyperparameters for the covariance function.

14. *Hyperparameter Upper Bound:* The maximum value that the hyperparameter can take.

4.3.3 Framework Tests

The testing of the mechanical property models in the Framework was slightly more restricted than that used for the Three Hump Camel test function. The parameters and the values that we tested for the mechanical property model optimization are shown in Table 4.1.

Table 4.1: Parameter values used for the Mechanical Model Function Test

Parameter	Values
Sample Size	10; 30; 50; 70
HP Count	100; 300; 500; 700; 900
Batch Size	1; 5; 10; 15
TF Iteration Limit	10; 25; 50; 100; 200
HP Lower Bound	0.1; 0.01; 0.001; 0.0001
HP Upper Bound	1; 10; 100; 1,000; 10,000
Model Cost Ratio	10^{-6}; 10^{-4}; 10^{-2}; 10^{-1}; 1
Number of initial Data	1; 2; 5; 10; 20; 50

The testing of parameters with the Three Hump Camel function was more extensive. In addition to testing single parameters, we also did two-parameter tests to investigate the correlation between specific parameters. These two-way tests use the same levels of the parameters in Table 4.2. In these tests, all the parameters were limited to three values to decrease the number of calculations required for the testing and allow for easier visualization of the results.

For each of these tests, we use multiple initial conditions to measure the distribution of the response from the optimization framework. Therefore, we require a measure to compare responses that have a mean and variance. This kind of analysis is well defined in the field of economics in

Table 4.2: Parameter Values Used for the Three Hump Camel Function Test

Parameter	Values
Sample Size	10; 50; 100
HP Count	100; 500; 1000
Batch Size	5; 15; 50
TF Iteration Limit	10; 50; 100
Fused Points	3; 10; 20
ROM Cost Factor	10^{-2}; 10^{-4}; 10^{-8}
Number of Models	3; 4; 5
Covariance Function	SE; M32; M52
TF Cost	5k; 20k; 50k
Two-way Tests	
Covariance Function & Fused Points	
Sample Size & HP Count	
HP Count & Batch Size	
Batch Size & TF Iteration Limit	
TF Iteration Limit & TF Cost	
TF Cost & ROM Cost Factor	

the form of Utility Function Theory. Following the description of this approach in the work by Sargent, [70] we define the Expected Utility as,

$$EU(x) = -exp\left(-\lambda\left(\mu(x) - \frac{\lambda\sigma^2}{2}\right)\right) \tag{4.27}$$

This expected utility allows us to measure which output is preferable considering a given risk aversion ($\lambda \in \mathbb{R}^+$). The Expected Utility defined in this way is always risk-averse to some degree and will always tend to favor results with lower variance. However, as λ increases, the risk aversion increases, and so the results of the Expected utility will more strongly favor those results that have the lowest variance. As a result, the risk aversion can be modified easily by changing a single parameter, making this a simple way to compare the results from the different tests.

The Framework contains several stochastic steps, the formulation of the test points is done using Latin Hypercube sampling, for instance. Therefore, there is likely to be variance in the Framework's performance related to these stochastic processes. To test this, we ran the Framework 5 times with the same input values and framework parameters. While conducting this test, we

tested a change to the Framework to determine whether it would improve the performance. The only difference was to update the method used to determine the next-best points to query from the Truth Function. In the first set of calculations, as with most of the tests in this work, the points to be queried from the Truth Function were found using the Knowledge Gradient approach on the test sample points. In the new approach, potential next-best points were found by finding the fused models' maximum. Since this new approach is less computationally costly than calculating the Knowledge Gradient, we could use more test samples, which results in a finer search of the space. Once the fused model's maximum value for each set of hyperparameters has been determined, these values are clustered using the k-medoids approach, and the batch of points to evaluate are selected.

4.4 Results and Discussion

4.4.1 Expected Utility representation of Results

As mentioned above, the results from the Framework testing are considered to be normally distributed with a mean and variance. Therefore, comparing the results from the different tests presents a challenge. This challenge has been met to some degree by the use of the Expected Utility. However, consideration needs to be made for the value of λ. As demonstrated in Figure 4.4 two probability distributions can be compared using the term $\mu(x) - \frac{\lambda\sigma^2}{2}$, with the maximum value of this term indicating the more desirable result. As shown in the right-hand plot of Figure 4.4, as λ increases, the distribution with the larger mean but larger variance becomes the less desirable option due to the high variance.

Unfortunately, other than the guidance that λ must be positive, the exact value required to differentiate between two different cases is a function of the mean and variance. For the current work, a value of $\lambda = 0.1$ was used. While the form of the expected utility used in the current work ensures that the decision-maker is always risk-averse, the value of $\lambda = 0.1$ was a relatively risk-neutral value that did not penalize the variance significantly. Choosing a different value for λ has the potential to change the analysis of the results significantly. In Figure 4.10, we demonstrate

the effect of increasing λ.

4.4.2 Bayesian Optimization Comparison

In both the mechanical and Three Hump Camel case studies in the current work, we compared the Framework's performance with a traditional Bayesian Optimization approach. In both cases, this approach used a single Gaussian Process surrogate model fit to data from the Truth Function. The number of initial results used was the same, and the same number of test points were used. These test points were also calculated using the same method as the BAREFOOT Framework used. For the Gaussian Process models, the hyperparameters were set so that all characteristic length scales (l) were 0.1 and the noise variance (σ_n) was 0.1 for the Three Hump Camel case study and 0.05 for the mechanical models' case study. The signal variance (σ_f) was set to 10 in the Three Hump Camel case study and 1 for the mechanical models' case study. Finally, for the mechanical model case study, we used the squared exponential covariance function, and for the Three Hump Camel case study, we used the Matern ($\nu = 5/2$) covariance function.

The method used for this pure Bayesian Optimization approach was to construct the GP surrogate model, and then we evaluated the Knowledge Gradient at each of the test points. As with the Framework, after each call to the Truth Function, the number of test points is incremented to ensure that a finer search of the design space is achieved as the optimization progresses. The next-best query is defined using the Knowledge Gradient and evaluated from the Truth Function.

In both case studies, the pure Bayesian Optimization results are compared with the results from the BAREFOOT Framework, where the cost and time of the pure Bayesian Optimization approach are calculated using the Truth Function cost for each of the case studies.

4.4.3 Mechanical Model Test Set

The results from all the parameter tests are included in the Appendix B. A selection of these results is presented here. The results shown in Figure 4.5(a) are the mean and 95% confidence interval from 5 separate calculations for each parameter set. The results shown in Figure 4.5(b) are the same data but converted to the Expected Utility value. We did this to demonstrate how

the expected utility results compare to the mean and confidence interval results. It is easier to observe the trends in the expected utility results. From this point on, all results will show only the expected utility. These results demonstrate how Truth Function Iteration Limit affects the overall optimization as a function of the time taken for the optimization. From Figure 4.5 we can observe that the iteration limit for calling the Truth Function is a significant parameter, and these results show that lower limits aid the optimization process.

The results for the upper bound tests for the hyperparameters in Figure 4.6 show that increasing the upper bound much beyond the length of the design space tends to decrease the effectiveness of the Framework. However, this effect is negligible. Therefore, since all the inputs in the Framework are transformed to the unit hypercube, it is unnecessary to have a maximum hyperparameter significantly above 1. The opposite was true of the lower bound of the hyperparameters (results shown in Appendix B). A boundary value of 0.0001 appears to help the Framework find better values, and so, it would be beneficial to set the lower bound at quite a low value. Changing the amount of initial data used appears to have minimal effect on the Framework's performance. It can be quite clearly observed that small numbers of initial points do not hinder the Framework's performance at all. This result confirms a finding in other work that shows that Bayesian optimization, in general, appears to work best when starting with very little data. This does not mean that having more data at the start of the optimization is a hindrance; it just means that if that data is not available, it is not necessary to obtain it before starting the optimization.

4.4.4 Three Hump Camel Test Set

The black dashed line shows the performance of a pure Bayesian Optimization approach for comparison. We can make several observations from this data. Firstly, Figure 4.7 shows that increasing the test sample count decreases the performance of the Framework. This is a slightly non-intuitive result since we would expect that sampling the design space more extensively would result in better performance. This could be influenced by using a wide range of hyperparameters to generate multiple fused models. The results for changing the hyperparameter count confirm the results seen in the previous results that a moderate number of hyperparameter sets is the best for

this Framework.

Interestingly, the results for the Truth Function iteration limit in Figure 4.8 show that the iteration limit of 50 performs worse than both the 10 or 100 iteration limits. This non-monotonic behavior is slightly strange but might indicate that there is more interaction happening in the Framework. However, we can observe that increasing the batch size does increase the efficiency of the Framework.

The results of the two-way interaction tests of parameters provided some beneficial results. While there is no clear correlation between the two-way test sample and hyperparameter count tests, these results shed more light on the results seen in the single parameter tests. In these tests, the 100 Test Sample cases outperformed the other cases, which is slightly contrary to the trend seen in the previous results. However, there is confirmation that the higher hyperparameter counts do not provide significant improvement. Some of the difference between these sets of results is possibly due to the Framework's stochastic nature. One of the strongest correlations in the two-parameter tests is with the Batch Size and the Truth Function iteration limit, as shown in Figure 4.9. This shows that as the iteration limit decreases, larger batch sizes are favored. This makes sense since as the iteration limit decreases, there are fewer iterations to obtain information from the reduced-order models. So by increasing the Batch Size, we can gain more information at each iteration.

These results from the Three-hump Camel tests show that while the BAREFOOT Framework can outperform a sequential Bayesian optimization, the Framework is not always capable of the improvements seen in the mechanical model problem. The hypothesis for why this occurs is that the accuracy, or link, between the reduced-order models and the Truth Function has a significant effect on the optimization. In the mechanical model problem, the reduced-order models are linked to the Truth Function through the physics that define the models. In particular, the response of the Truth Function is known to be some combination of the simplified models used as reduced-order models. In contrast, the reduced-order models in the Three-hump Camel test have no real connection to the Truth Function. A small trial of this hypothesis was done using a test function

and Fourier Series expansions of the Function. The details of this test are shown in Appendix B.

In an attempt to measure the variability of the Framework, we ran an additional test. In this case, we ran the same set of framework parameters and initial conditions five times. If the Framework were deterministic, each of these calculations would give the same results. However, as can be seen in Figure 4.10 this is not the case. Two sets of results are presented in this figure. The first is the results from 5 calculations using the Framework as described above (old structure). The second result is from a modification of the framework structure in an attempt to (a) reduce the variance in the result, (b) utilize the information contained in the fused model more thoroughly, and (c) provide a much finer query of the design space when evaluating the Truth Function. The change in the Framework is purely in determining the points for querying from the Truth Function. A set of 5000 test points are sampled from the design space by Latin hypercube sampling in the new structure. Then for each of the hyperparameter sets in the Framework, the fused model is constructed, and the point with the maximum objective value is stored. These points are clustered using the k-medoids approach to form the required batch size. As seen from the results, the new structure shows a much smaller variance and a higher maximum. We also present how the λ parameter in the Expected Utility function modifies the results as an additional comparison of these results. When using the lowest value of $\lambda = 0.1$ (low risk-aversion), there is almost no difference between the performance of the two framework structures since both results lie within the confidence intervals of the largest uncertainty bound. However, if we consider a more risk-averse decision-maker ($\lambda = 1.0$), then the difference between the two framework structures becomes quite significant, with the newer structure performing significantly better.

4.5 Conclusions and Future Work

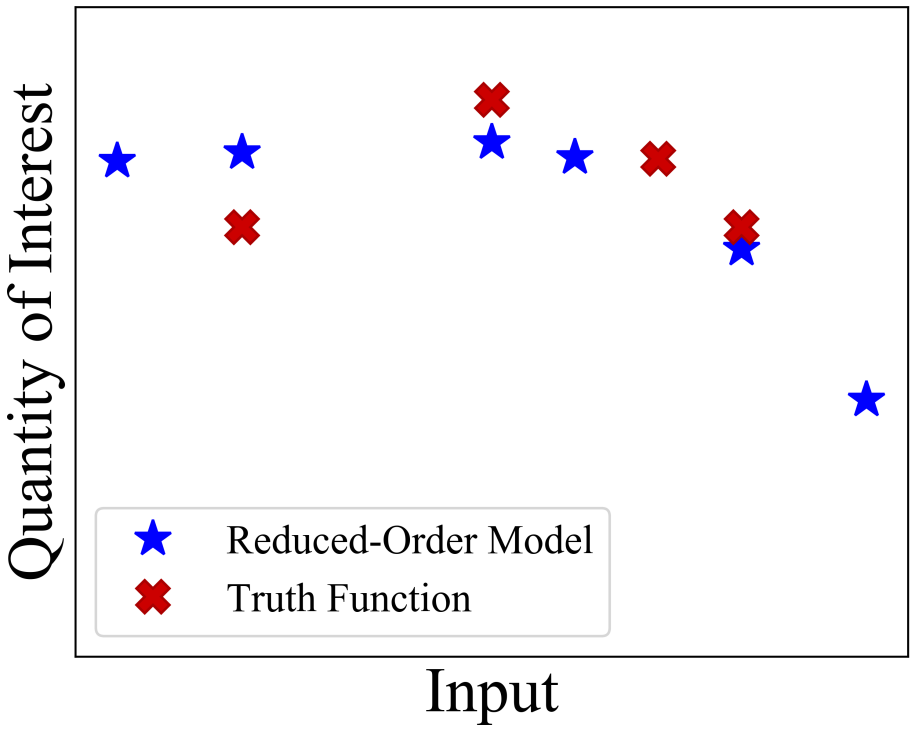
In this work, we have presented results from comprehensive testing of our batch optimization framework. The results demonstrate that the Framework performs satisfactorily when optimizing two different functions. In addition to testing the Framework against multiple functions, we have also evaluated how changing some of the most prominent parameters in the Framework affects the Framework's operation and result. These results have indicated that specific parameters are of

significantly greater importance than others. It is also possible to suggest typical values for at least some of the parameters to aid with analyzing future situations.

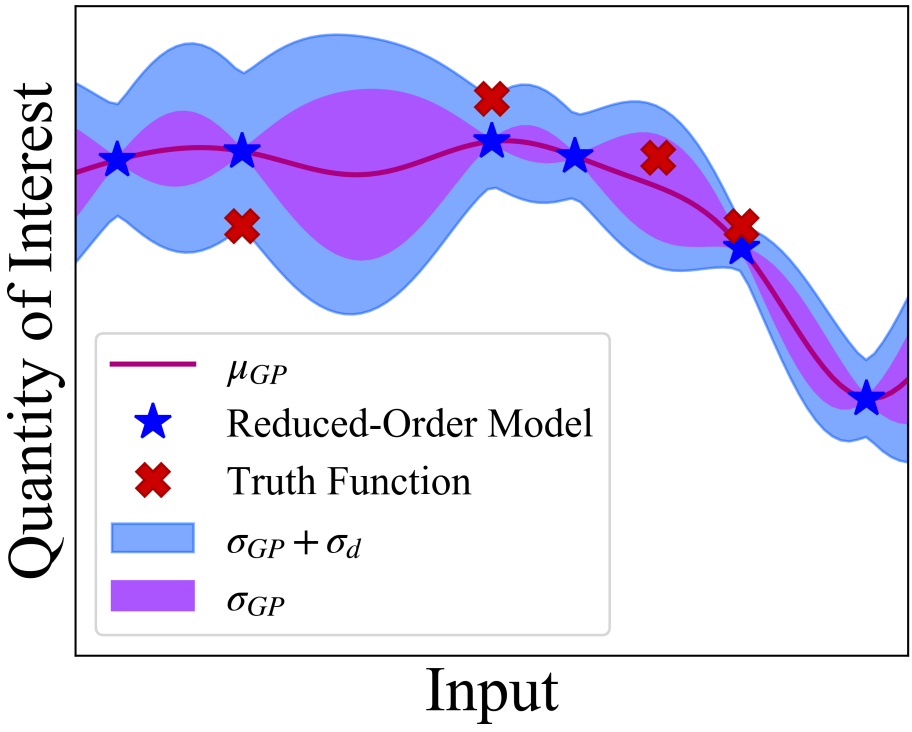
These results also indicate that the modified approach to identifying the next-best points for the Truth Function provides less variance than the earlier approach. Therefore, this approach will be the one used in the Framework.

We can still develop the Framework further in several ways. Firstly, it would be worth considering including the acquisition function as a randomly assigned hyperparameter for each iteration. Secondly, and more importantly, it is necessary to test this Framework on a higher dimension problem to determine how much this will affect the performance. In addition to these two tests, it would also be interesting to ascertain whether the use of the k-medoids clustering is required. Part of the reasoning behind this additional test would be that the justification for k-medoids clustering given previously is most applicable when the dataset has already been generated. It is necessary to define the clusters by an evaluated point. However, in the current work, the medoids are evaluated from the true models after the clustering is completed. As such, there should be no real barrier to using the k-means approach (As noted previously, if there are a large number of constraints on the optimization problem, it would be better to use the k-medoids approach). K-means and k-medoids potentially differ in the computational time and the robustness of each approach to outliers. So it would be worthwhile to investigate which approach works better in the current Framework.

We tested the effect of reduced-order models of different accuracy in a preliminary case study to determine whether the accuracy of the reduced-order models affects the performance of the optimization. These results indicate that there is potential for models that significantly deviate from the Truth Function to influence the results. However, this effect still needs to be studied further, and we should also investigate the implication of this result for using generic machine learning models as the reduced-order models.

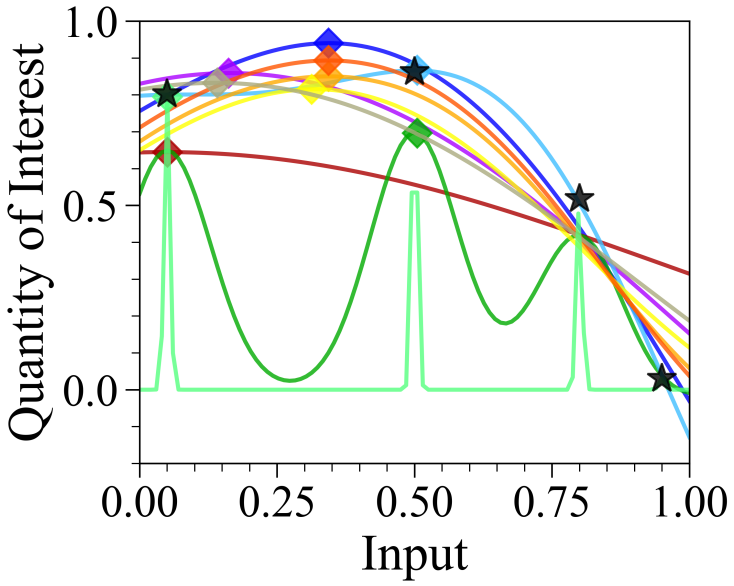


(a)

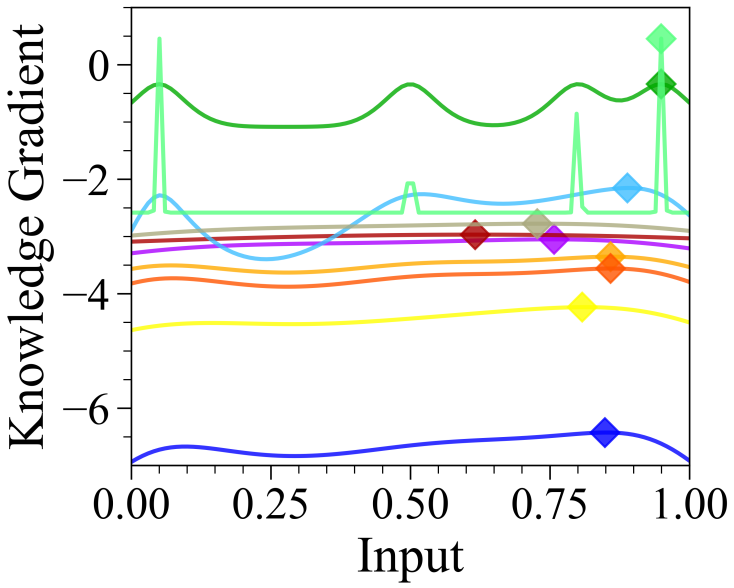


(b)

Figure 4.2: Schematic representation of how the surrogate and discrepancy models are constructed



(a)



(b)

Figure 4.3: Gaussian Process models fit to a set of four training points (stars) using different values of the noise variance, signal variance, and length scale hyperparameter. Diamond marker indicates the location of the maximum of each Gaussian Process model in the left plot. The right-hand plot shows the Knowledge Gradient calculated for each GP model with the maximum indicated by a diamond marker.

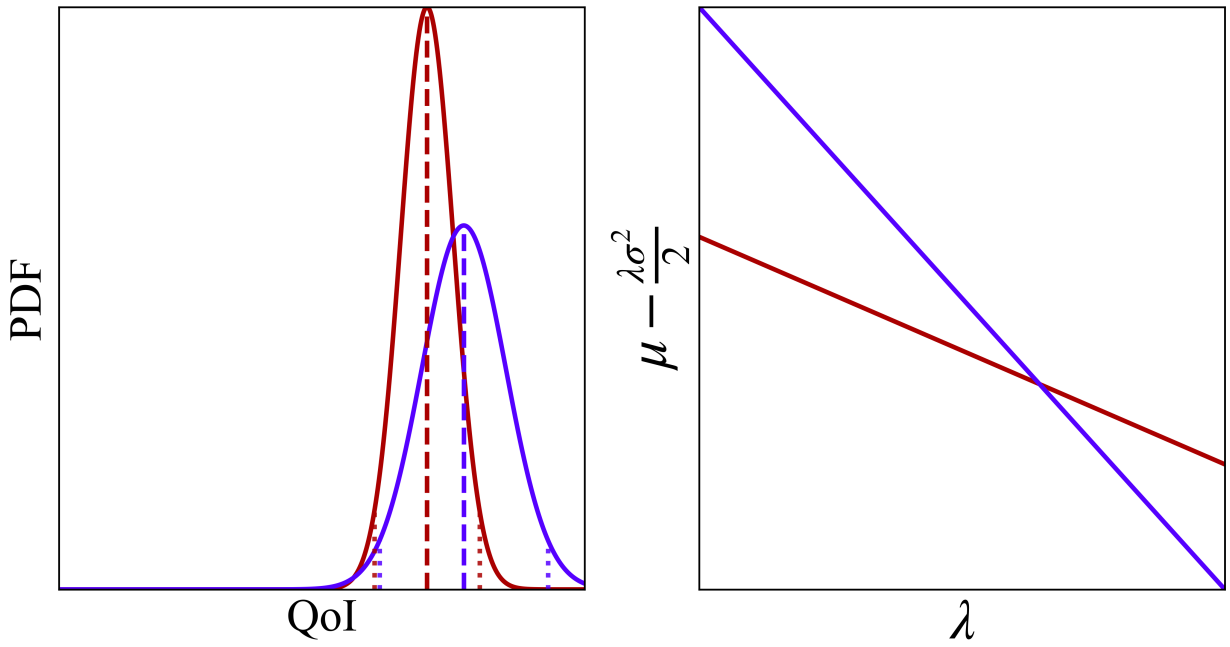
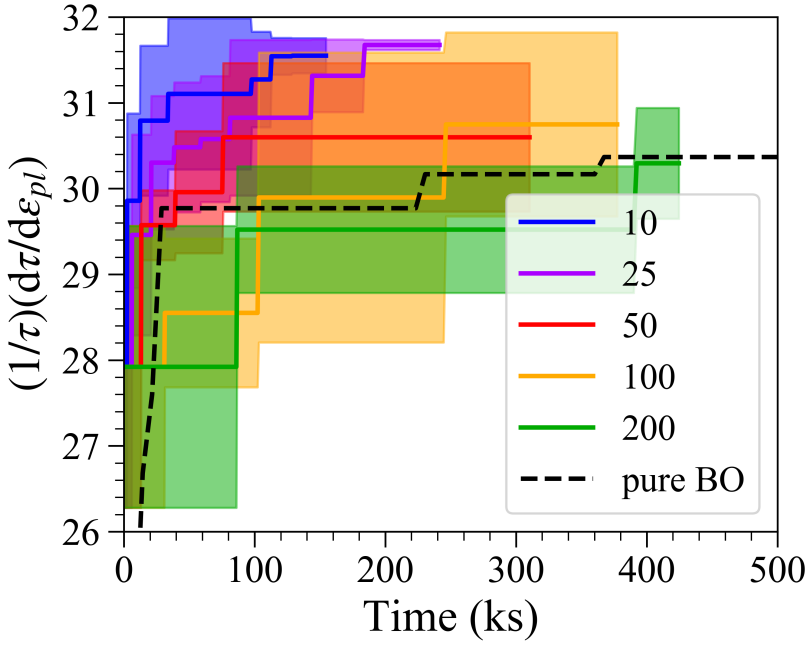
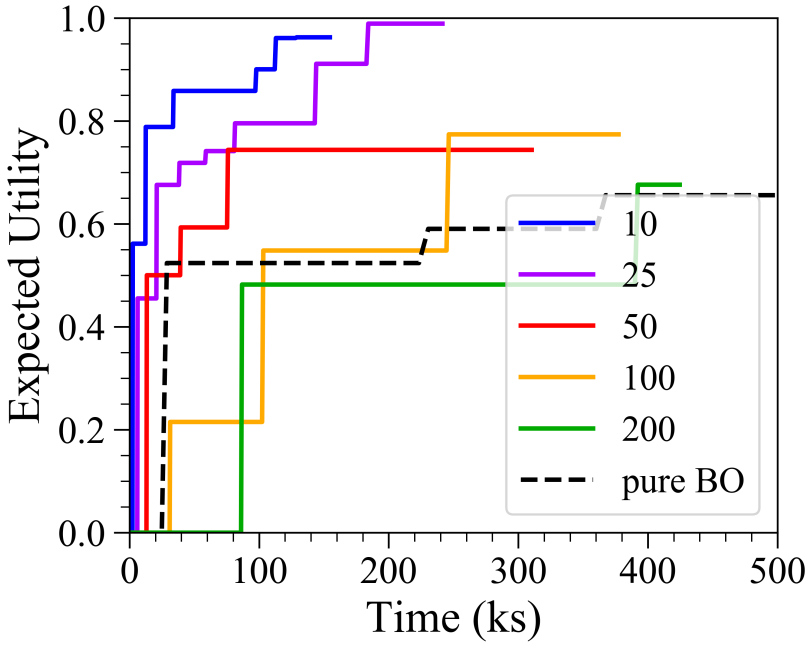


Figure 4.4: Comparison of Expected Utility for two distributions as a function of the risk aversion parameter λ

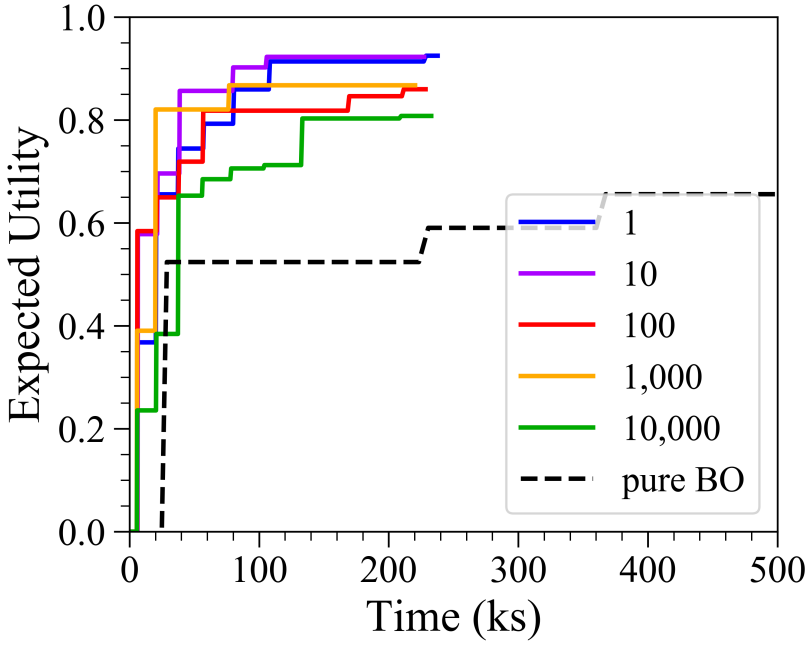


(a)

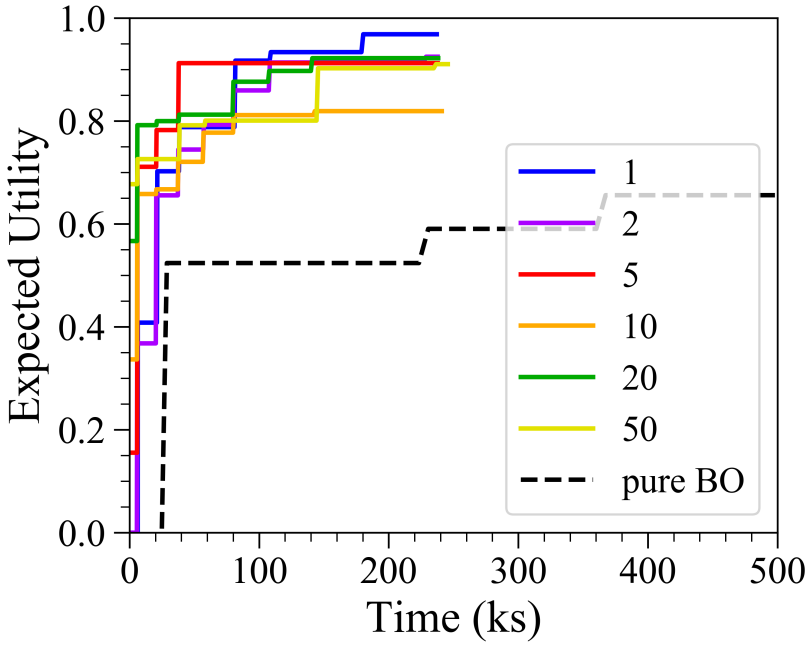


(b)

Figure 4.5: Comparison of framework performance when optimizing the Mechanical Models for different Truth Function Iteration Limits showing (a) mean and variance (b) Expected Utility

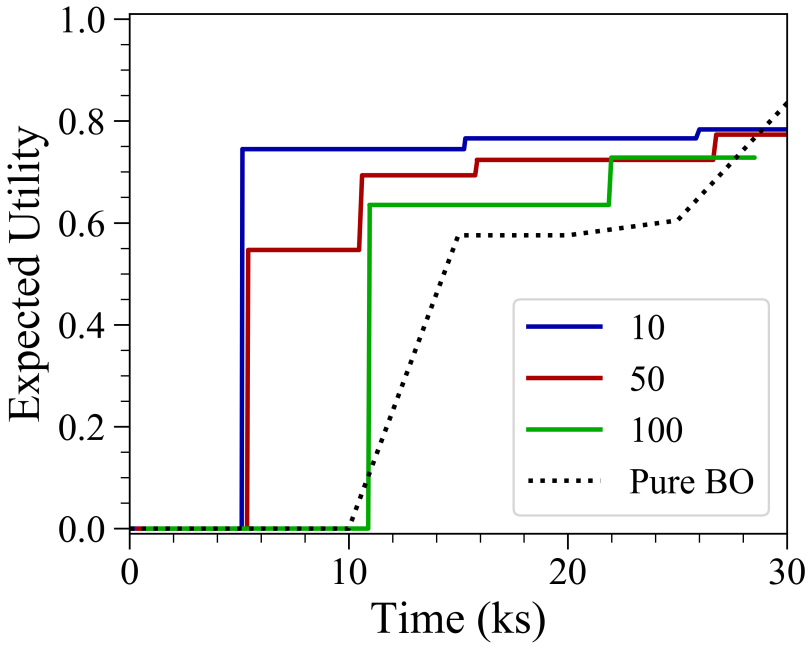


(a)

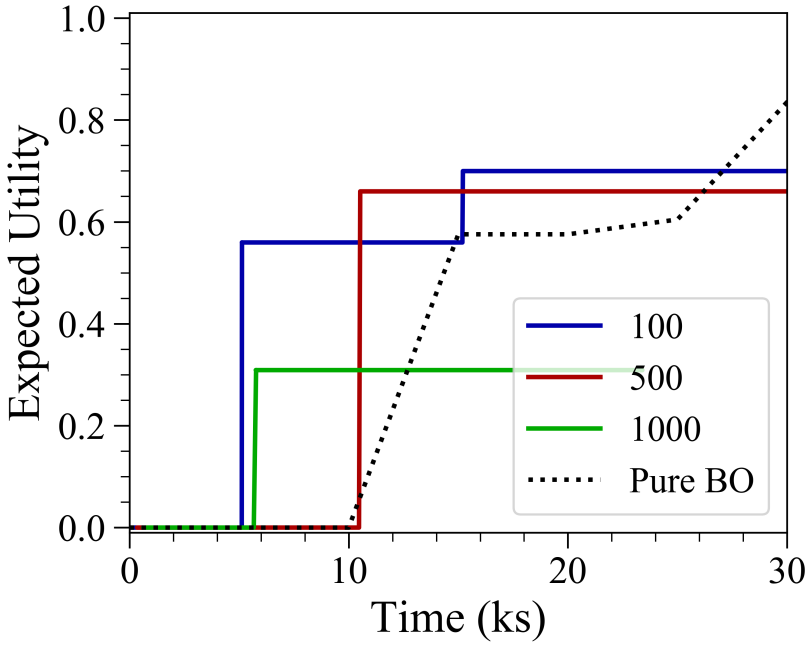


(b)

Figure 4.6: Comparison of framework performance when optimizing the Mechanical Models for single parameter tests (a) Hyperparameter Upper Bound (b) Number of Initial Data Points

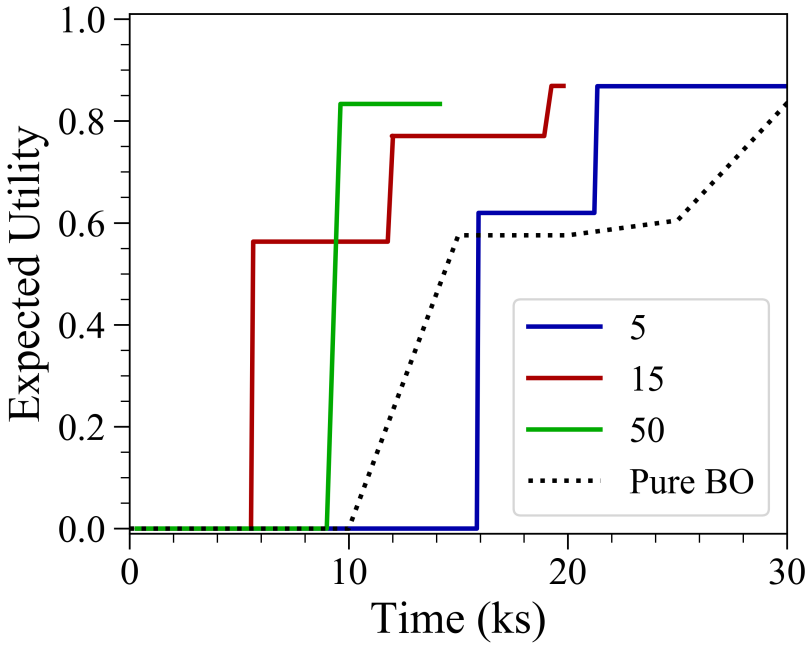


(a)

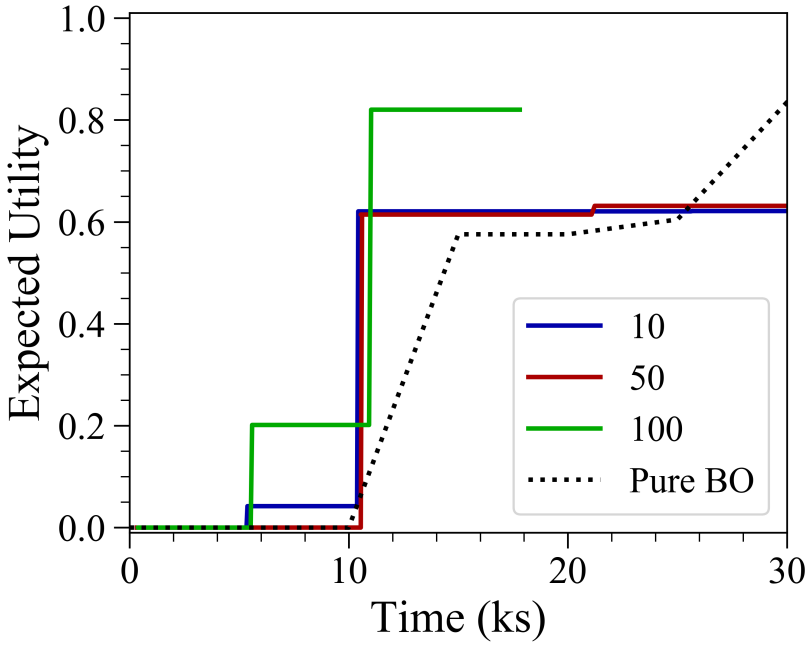


(b)

Figure 4.7: Comparison of framework performance when optimizing the Three Hump Camel Function for single parameter tests (a) Test Sample Count (b) Hyperparameter Count (c) Batch Size.

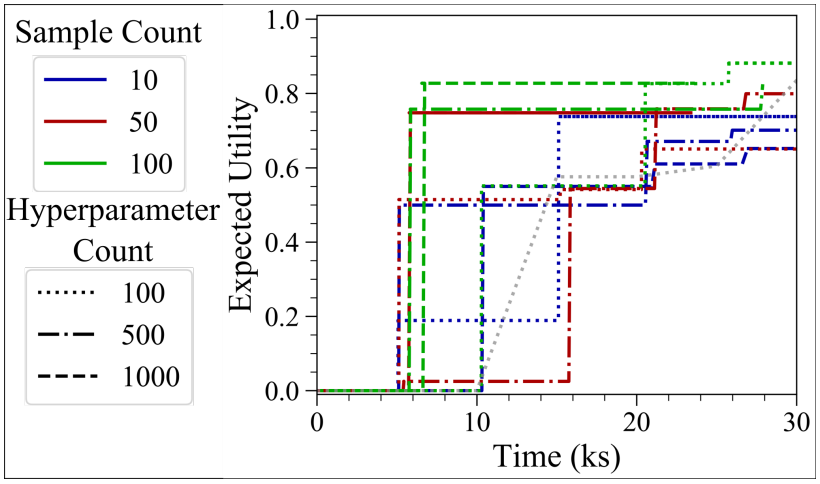


(a)

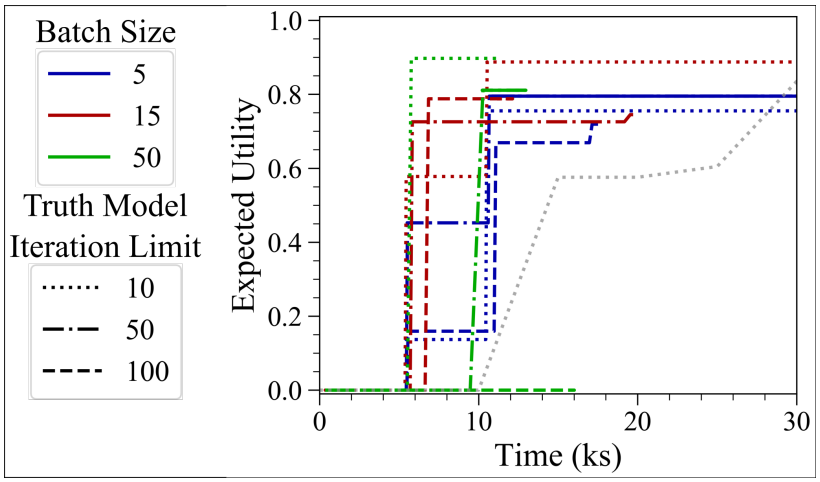


(b)

Figure 4.8: Comparison of model performance when optimizing the Three Hump Camel Function for single parameter tests (a) Truth Function Iteration Limit (b) No. of Fused Points (c) Reduced-order Model Cost Factor.

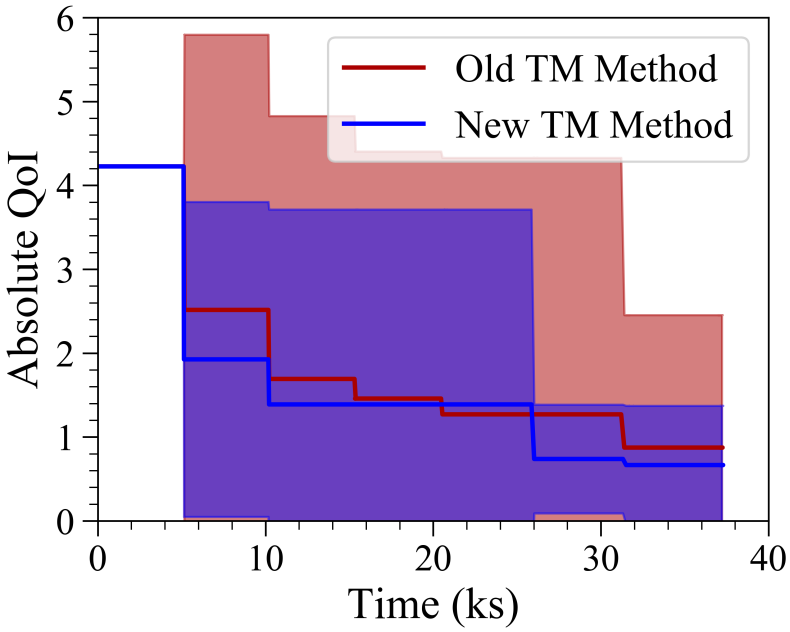


(a)

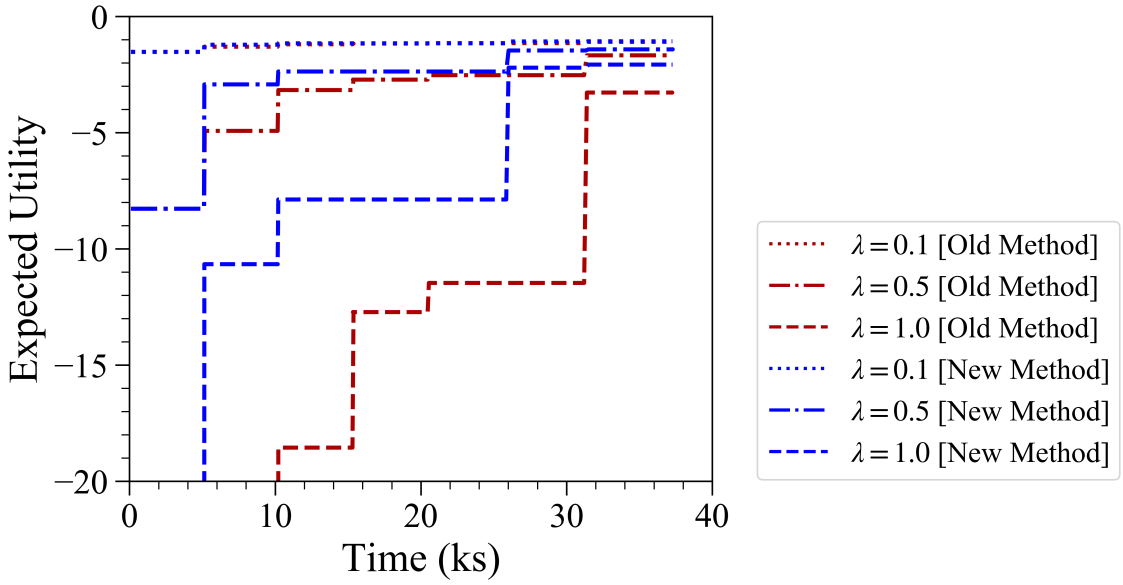


(b)

Figure 4.9: Comparison of framework performance when optimizing the Three Hump Camel Function and changing two different parameter values (a) Covariance Function and No. of Fused Points (b) Test Sample Count and Hyperparameter Count (c) Hyperparameter count and Batch Size.



(a)



(b)

Figure 4.10: Result comparison for the framework uncertainty associated with the old and new approaches to defining the next-best-points to query from the Truth Function. (a) shows the comparison of the mean and uncertainty from 5 calculations with the same initial start conditions and parameters. (b) shows the Expected Utility associated with these results when using different values for λ.

5. APPLICATION OF THE FRAMEWORK TO DFT MODELS AND INCORPORATION OF MACHINE LEARNING MODELS

5.1 Application of BAREFOOT to RHEA design space and DFT Modeling

High Entropy Alloys (HEAs) are an exciting branch of alloys and define the latest research area in alloy design to a large degree. Conventional alloys were defined by having only one or two principal alloying elements. This class of materials is probably most easily demonstrated by steels and Fe-based alloys. Fe is the dominant element with more than 50 at.% of the alloy consisting of the element in the vast majority of these materials. The other elements in the alloy are added in dilute amounts to improve the properties of the Fe. These alloys became the standard since alloys with high amounts of alloying elements tend to form intermetallic phases during operation. These intermetallic phases can be detrimental to the mechanical properties of the material.

The new paradigm, exemplified in HEAs, is to have alloys with equiatomic, or near equiatomic, concentrations of all elements. As such, there is no dominant element in the alloy. Miracle and Senkov draw a distinction between Multi-principal Element Alloys (MPEAs) and High Entropy Alloys (HEAs). HEAs aim to have a single-phase solid solution of all the alloying elements, while MPEAs can consist of multiple phases.

A significant amount of research is being conducted on these alloys since they show good mechanical properties and have many potential applications in the aerospace industry.

One of the more recent drives in HEA research is the idea of producing refractory HEAs. These alloys are defined as high entropy alloys with very high melting point. These alloys typically rely on alloying with W to produce materials with such high melting point. However, one challenge for high-W alloys is the mechanical properties of any alloy produced. One of the properties of these alloys that causes significant problems is the low temperature ductility of the alloys. Alloys with significant amounts of tungsten are typically very brittle at room temperature. This makes shaping or forming operations virtually impossible at low temperatures and still significantly challenging

at high temperatures. As a result, we would like to be able to explore the refractory HEA design space to find alloys that have sufficient ductility for processing at room temperature.

However, this optimization presents several problems. Firstly, the design space is enormous. The alloy system of interest is an eight-element system (Al, Cr, Fe, Mo, Nb, Ta, V, W). With steps of only 1 at.% in the alloy composition space, there are on the order of 10 trillion potential compositions. With a design space this large, it will never be possible to evaluate all combinations. So a targeted design is likely to be the only to discover possible alloys. We applied the BAREFOOT Framework to this problem using two empirical approximations of the Bulk Modulus and a Density Functional Theory (DFT) based calculation as the ground truth. The DFT-based calculation developed by Johnson et al. is a self-consistent DFT model that uses the Korringa-Kohn-Rostoker (KKR) method coupled with the coherent potential approximation (CPA) to calculate the electronic structure of random alloys. The DFT calculations give the gradient of the pressure and volume at the ground state. The Munarghan equation of state is then applied to calculate the Bulk Modulus. We refer to this DFT model the KKR Model.

The two empirical models are a linear model and a CALPHAD-like model. The Linear Model considers the atomic fraction of each element and the elemental Bulk Modulus as the only inputs and is shown in Equation 5.1. Where B is the Bulk Modulus, N is the atomic fraction and λ is a fitting parameter.

$$B_{alloy} = \sum_{i=1}^{8} \lambda B_i N_i \tag{5.1}$$

While the CALPHAD-like Model takes the form of a CALPHAD-like equation with higher-order terms and a logarithm, Equation 5.2. Included in this empirical model is a set of binary interactions between the elements.

$$B_{alloy} = \sum_{i=1}^{8} \left[\left(\lambda_i^a N_i + \lambda_i^b N_i log(N_i) + \lambda_i^c N_i^2 + \lambda_i^d N_i^3 \right) B_i \right] + \sum_{i=1}^{7} \sum_{j=i+1} 8\lambda_{i,j} \left(N_i B_i \right) \left(N_j B_j \right) \tag{5.2}$$

We use these two equations as the reduced-order models in the BAREFOOT Framework. In this application of the Framework, we provide no prior training of these empirical models. The weight parameters (λ) were trained as Truth Model evaluations were obtained. This was a good test of the BAREFOOT Framework's ability to train models during an optimization.

5.2 Exploration of the Refractory High Entropy Alloy Space using BAREFOOT

5.2.1 Method

Using the model setup above, we aimed to test how the BAREFOOT Framework approach could find areas in the design space with a maximum Bulk Modulus. We ran the calculations with two different configurations. In the first, we did not consider any prior calculations from the KKR Model and instead calculated initial data. In contrast, we considered about 230 prior calculations from the KKR Model and 50 calculations from each reduced-order model as initial data in the second approach. As such, we will refer to these two calculations as Calc-Init (calculated initial data) and Imp-Init (imported initial data).

The real strength of this approach is that all the models were incorporated and run automatically from the Framework. The Framework was capable of handling KKR Model calculations that failed to complete and ignore those results. This provides validation for the BAREFOOT Framework as an optimization tool that can be incorporated with complex computational models and indicates that it will be possible to integrate this approach directly with automated experimental testing in the future.

In this test, we also trained the reduced-order model parameter on the fly. To do this, we note that we can rewrite both reduced-order models using the form,

$$\bar{\lambda}\bar{X} = \bar{Y}, \tag{5.3}$$

using this knowledge, we can solve for the lambda parameters by finding the inverse of the X matrix and solving for $\bar{\lambda}$. Since we do not believe that the X matrix will be well-conditioned enough to find the true inverse, we calculate the Moore-Penrose inverse of the matrix and solve for the λ parameters.

The most important parameters used for these two calculations are shown in Table 5.1. The Framework was allowed to run for 500 iterations for the Calc-Init case and 100 iterations for the Imp-Init case. As can be seen, the only parameter that changes between the two approaches is the Truth Model iteration limit. This parameter determines the number of iterations of reduced-order model evaluations that need to be completed before calling the Truth Model. The motivation for the change was that the Imp-Init calculation already contains information about the models. So there is less need to query the reduced-order models as extensively.

Table 5.1: Summary of Important BAREFOOT Framework Parameters used when conducting the initial calculations with the KKR Model Ground Truth.

Parameter	Calc-Init	Imp-Init
Truth Model Iteration Limit	30	10
Sample Count	100	
Hyperparameter Set Count	500	
Batch Size	10	
Covariance Functions	Knowledge Gradient	
Acquisition Function	Matérn ($\nu = 3/2$)	

5.2.2 Results

Before discussing the results from the calculations, we note that one objective of the calculations was to train the reduced-order models while the calculations progressed. However, during the calculations, we discovered that the code was not working as intended. We chose to let the

calculations continue despite this. As a result, we conducted a more extensive test of the training procedure. The results from this test are shown later to demonstrate that the model training approach can be successful.

The Calc-Init approach resulted in 134 evaluations of the KKR-Model, and the Imp-Init approach resulted in 79 evaluations. To demonstrate how these results are distributed in the design space, we use the t-distributed Stochastic Neighbor Embedding (t-SNE) dimensionality reduction technique. To ensure that we fully represent the design space, we first take a uniform grid sampling over the design space with 10 points per dimension (10^8 samples) and add this data to the results from the BAREFOOT Framework. These addition points force the t-SNE algorithm to consider the entire design space, and we can also define the regions rich in specific elements. To do this, we color the markers in the scatter plot that have more than 60 at.% of any single element with different colors to represent each element.

To better understand how the optimization progresses, we identify the results from the Framework by changing them to an "x" marker. We then color the markers on a color scale related to the magnitude of the Bulk Modulus at that evaluation point. Finally, the marker size is changed to reflect the Iteration number of the queried point. Large markers mean that the point was evaluated at a later iteration. These results are shown in Figures 5.1 and 5.2

From the results in Figure 5.1, we can observe that the BAREFOOT Framework achieves two goals. Firstly, the evaluated points are spread quite significantly through the design space. Since the size of the markers indicates the iteration at which the point was evaluated, we can see that the Framework does not query high Bulk Modulus regions exclusively in later iterations. There are large markers in the Al-rich region, indicating that the Framework still evaluates low Bulk Modulus regions at later iterations. This exploration of the design space is a direct result of the clustering approach used in the batch step. This balance of exploration introduced through the clustering step is a strength of this Framework since it allows for more exploration of the design space than would typically be obtained, reducing the risk that the optimization gets trapped in a local minimum.

Secondly, we can see that the BAREFOOT Framework has been able to identify high Bulk

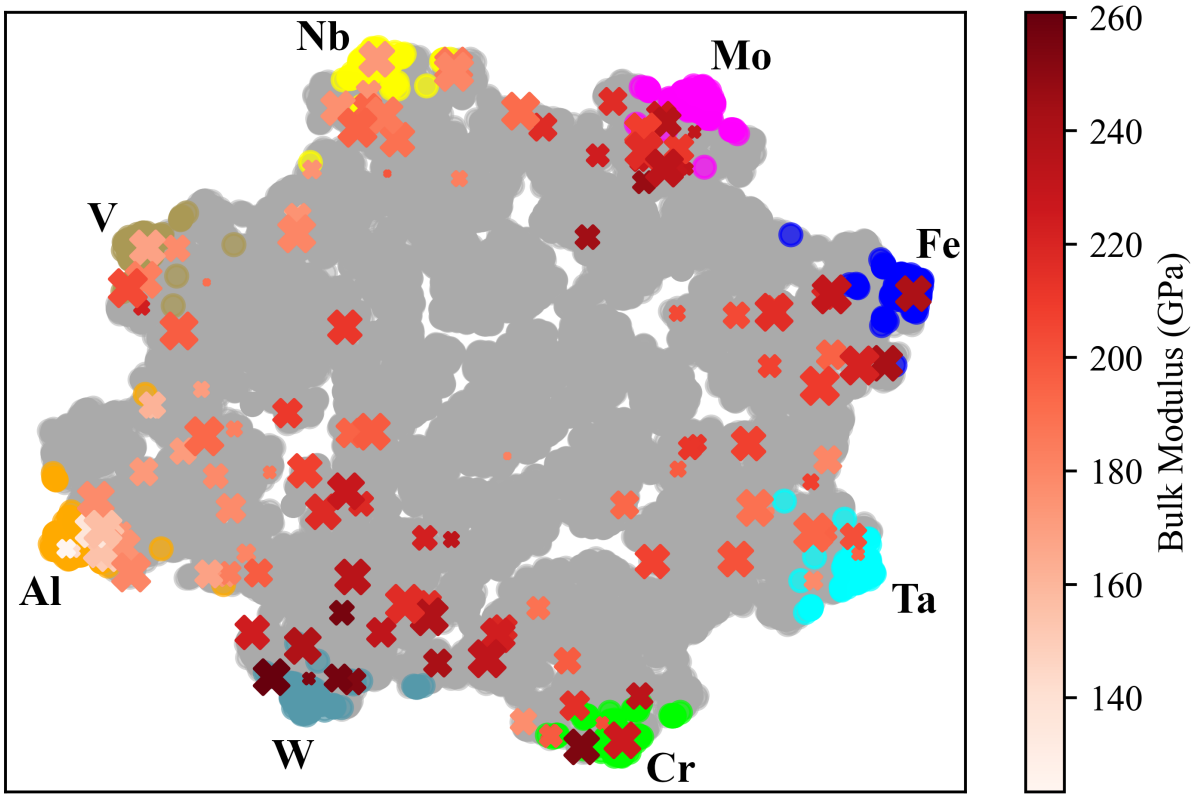


Figure 5.1: Results from the calculations using the KKR Model Ground Truth, and calculating the initial data. The plot shows the distribution of Truth Model evaluations in the design space with highlighted areas showing areas in the design space with 60 at.% or more of the labeled element. The cross markers show the Truth Model evaluations with color associated with the Bulk Modulus of that composition and the size indicating the iteration number for the evaluation (large marker shows larger iteration number).

Modulus alloys in a high dimensional space. Moreover, this has been achieved with a relatively small number of KKR Model evaluations (only 134 evaluations). While the KKR Model usually takes only a few hours to complete, this result is promising for coupling the Framework with more time-intensive models or experiments that can take several days to complete. From a practical standpoint, we can see that the Framework and the KKR Model results are reasonable and match with expected trends in the alloy space. Namely, the materials with the highest Bulk Modulus contain a significant amount of W, Mo, or Cr. These elements have the largest Bulk Moduli of the elements used in the study, and it would make sense that alloys with large amounts of these elements would also have high Bulk Modulus.

The results shown in Figure 5.2 show a similar trend to that seen in Figure 5.1. We can see that the algorithm explores the design space quite extensively. However, in this case, we see a large amount of initial data that concentrates results in certain regions. As a result, we do not see as much exploration of the element-rich regions. Furthermore, there appears to be a focus on the Fe-rich region of the design space. From these results, it appears that the Framework performs slightly better when it is given very little initial data.

5.3 Training models during BAREFOOT operation

5.3.1 Method

We used the refractory alloy design problem defined above to test the approach of training models during the BAREFOOT optimization. However, we implemented a small change to enable quick testing of the approach. Through the work exploring the design space and incorporating some prior calculations, we accumulated 500 evaluations of the KKR Model. Using all these evaluations, we trained the parameters of the CALPHAD type equation using the pseudo-inverse approach. In this case, we used the implementation of the Moore-Penrose pseudo-inverse in the `scipy.linalg` module. This trained model was then considered the Ground Truth for the optimization. To estimate the possible maximum of this function, we used a Genetic Algorithm optimization approach to optimize the function. For reference in later results, this approach used

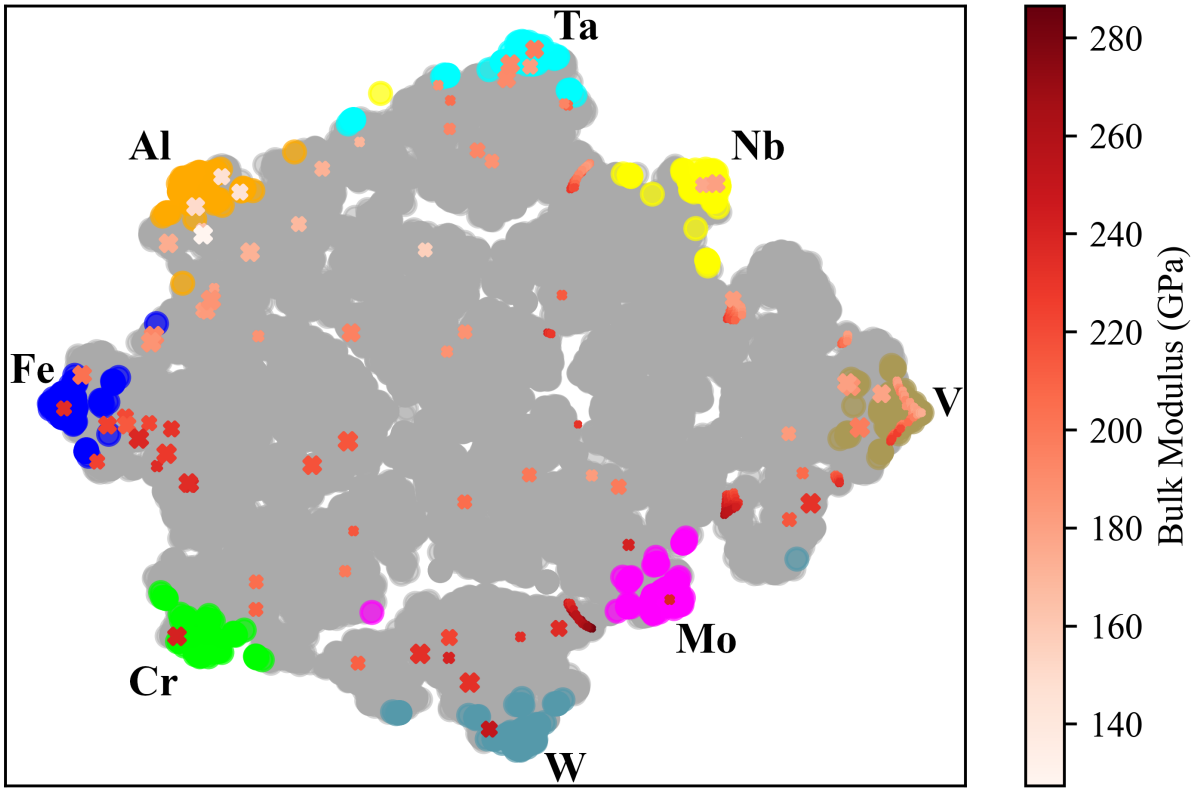


Figure 5.2: Results from the calculations using the KKR Model Ground Truth, and importing the existing KKR Model evaluations as initial data. The plot shows the distribution of Truth Model evaluations in the design space with highlighted areas showing areas in the design space with 60 at.% or more of the labeled element. The cross markers show the Truth Model evaluations with color associated with the Bulk Modulus of that composition and the size indicating the iteration number for the evaluation (large marker shows larger iteration number).

3000 iterations and about 1 million function evaluations to find the maximum value of 320 GPa. In all our analysis, this will be referred to as the "Known Maximum".

Having defined a new, quick-to-evaluate Truth Model, the reduced-order models were set to the models in Equations 5.1 and 5.2. We initially set all the weights in the models equal to one. We then proceeded with the optimization under four different conditions. Firstly, we considered the Barefoot approach that used both the Batch and Reification approaches. We did two calculations with the Barefoot approach, one with the reduced-order models trained through the optimization and one without training. The second approach used the Batch approach for the optimization. The final approach was a sequential Bayesian Optimization.

The parameters for these calculations were as shown in Table 5.2. It must be noted that all relevant parameters are listed, but not all are used in each calculation. Additionally, the iteration limit was set with the Reification approach in mind, so the iteration limit was divided by 5 (the number of ROM evaluation iterations) for the Batch approach. These calculations were replicated 30 times to obtain the statistical variance of the result. We calculated 30 sets of initial conditions to achieve this replication, and these initial conditions were used for all approaches.

Table 5.2: Summary of important parameters used for the calculations testing the training of the reduced order models during the optimization.

Parameter	Value
Hyperparameter sets	500
Batch Size	10
Sample Count	20
Iteration Limit	100
TM Iteration Limit	5
Covariance Function	
HP Upper Bound	1
HP Lower Bound	0.0001

These four different approaches are labeled as BAREFOOT with the training of the reduced-order models (BAREFOOT With Training), BAREFOOT calculations without training (BARE-

FOOT No Training), the plain Batch Bayesian Optimization (Batch BO), and a conventional, sequential, Bayesian Optimization (Sequential BO) approach. However, because it becomes difficult to compare the results when too many plots with uncertainty bands are shown on a single figure, we consider the Expected Utility to obtain single line plots for each result. The Expected Utility is calculated as,

$$EU(x) = -exp\left(-\lambda\left(\mu(x) - \frac{\lambda\sigma^2}{2}\right)\right) \tag{5.4}$$

where λ is the level of risk aversion. In the current work, we found that $\lambda = 0.01 - 0.5$ only changed the relative scale of the Expected Utility, but did not change the order of the plots, so the value of $\lambda = 0.01$ was chosen since it produced the most readable plots.

One of the significant challenges in this example was how to sample the design space. This is because the constraint in a composition space becomes very restrictive at high dimensions. In this regard, we utilize a sampling approach proposed by Woronow [71]. This approach scales a random sampling of the entire design space to transfer the sampling to the constrained design space (or simplex). For an n component system, the composition constraint implies that,

$$\sum_i = 1^n c_i = 1. \tag{5.5}$$

However, while this constraint allows us to identify one element as a balance element and work in an n-1 dimension design space, the sampling needs to be done in the full design space to work correctly. This is because the conversion assumes that all rows in the matrix must sum to 1. So, for this work, we take an LHS sampling of the design space, then use the following relationship to convert the values to the constrained design space,

$$x_{i,constr} = \frac{-ln(x_i)}{\sum -ln(x_i)}. \tag{5.6}$$

After querying the whole design space, we then use the first n-1 dimensions of the design space as the design variables.

5.3.2 Results

5.3.2.1 Optimization

The optimization results shown in Figure 5.3 show that there is little difference in the approaches except for the sequential Bayesian Optimization approach. Unfortunately, there are too many plots in Figure 5.3 to observe the differences between each of the Batch and Reification approaches clearly. So using expected utility 5.4 we can compare the results more quickly. The one result that is clear in both figures is that the sequential Bayesian Optimization does not perform as well as either the Batch or Reification approaches. Additionally, in Figure 5.3, we plot the results against the number of function evaluations, which means that the sequential Bayesian Optimization will take about ten times longer than any of the batch processes for the same number of function evaluations. As noted before, this is one of the significant strengths of the Batch-based approaches.

Observing the results in Figure 5.4 we can note that the approaches with and without training perform similarly. This might be that the models used in this test were a little too simple. However, the result possibly implies a change in previously observed results. In Chapter 2, it was observed that the quality of the reduced-order models affects the optimization. This result was supported by a simple test presented in Appendix 2. Therefore, there is a chance that the reason for the better performance is that while the untrained models are not perfect for optimization, they are still good models. This theory would be supported by the fact that one model is the same model as the Truth Model, just without training. Overall, the Batch Bayesian Optimization approach, which uses only the Ground Truth model, performs fractionally better than the BAREFOOT approach.

5.3.2.2 Parameter fit

Despite the optimization results showing that the training of the models during the optimization does not necessarily provide any advantage over using generic models, the results from the training of the parameters were ecouraging. In all of the calculations, the optimization was able to reproduce the parameters of the Calphad model in approximately six sets of Truth Model evaluations

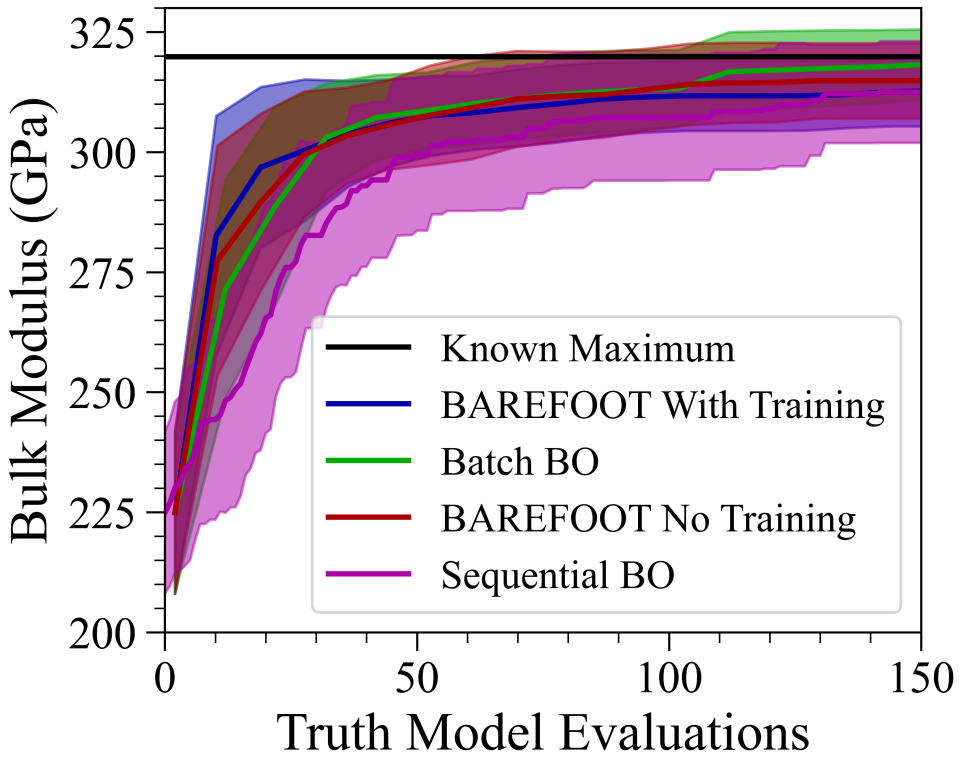


Figure 5.3: Results showing the mean and approximate 95% confidence interval for the optimization of the test functions used when testing the model training approach.

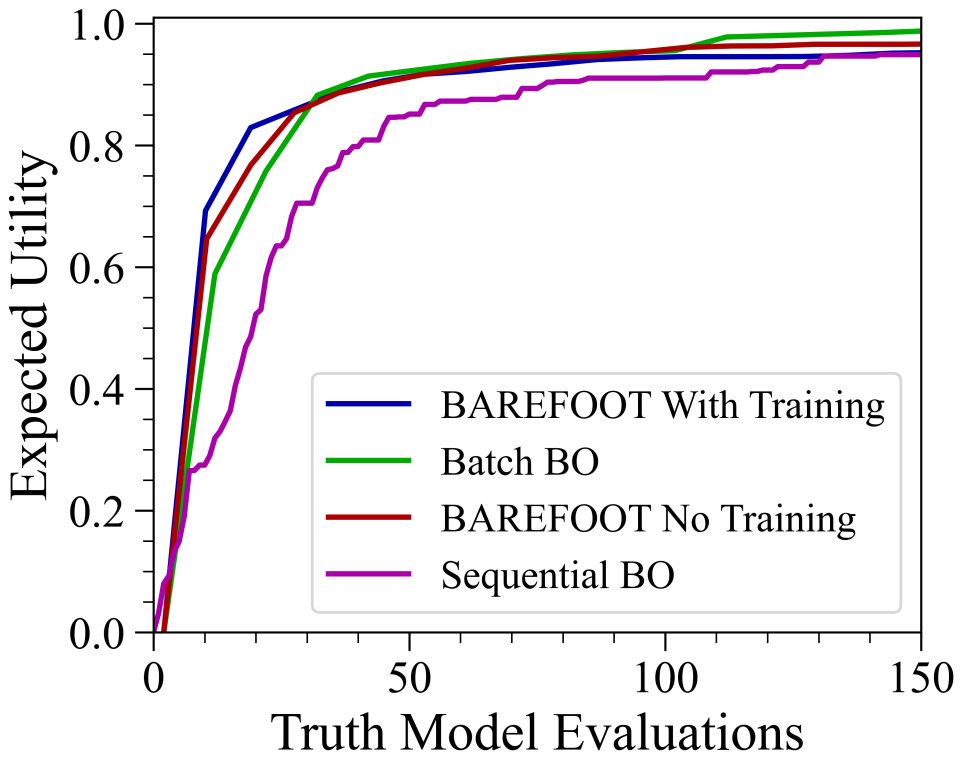


Figure 5.4: Results showing the Expected Utility for the optimization of the test functions used when testing the model training approach.

(approximately 60 Truth model evaluations). Furthermore, once the model parameters matched the true parameter values, they did not change significantly for the remainder of the optimization. An example of the results are shown in Figures 5.5 to 5.9.

Figures 5.5 to 5.9 show the fit of the CALPHAD type model for a random set of points comparing Truth Model (the model trained on 500 KKR Model evaluations) and the CALPHAD Model (trained only on output from the truth model. As we can observe, the models start with a large amount of error in the predictions, but within a couple of training iterations, the fit is perfect. To complement this, we can observe that the model's parameters slowly converge to near-identical values over the same number of training iterations.

This result indicates that while there might be no benefit in the optimization performance, there is a significant benefit in training models while the optimization is ongoing. This result is promising for the potential training of more complex machine learning models such as Neural Networks, Support Vector Regression models, or Random Forest models. However, this kind of approach still needs to be investigated thoroughly. The ultimate aim of any of these approaches is to train models during the optimization approach so that an optimized design and a trained reduced-order model can be obtained.

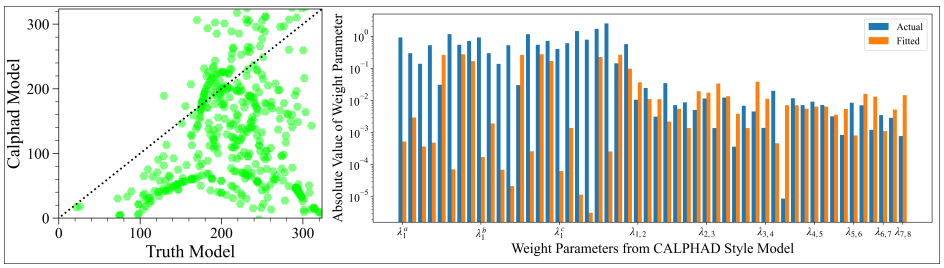


Figure 5.5: Model fit and parameter comparison for training step 1

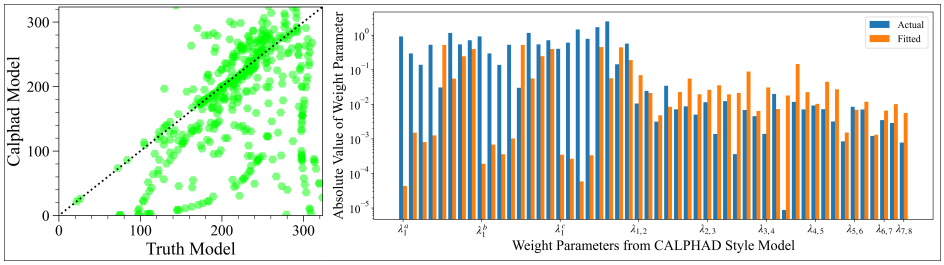


Figure 5.6: Model fit and parameter comparison for training step 2

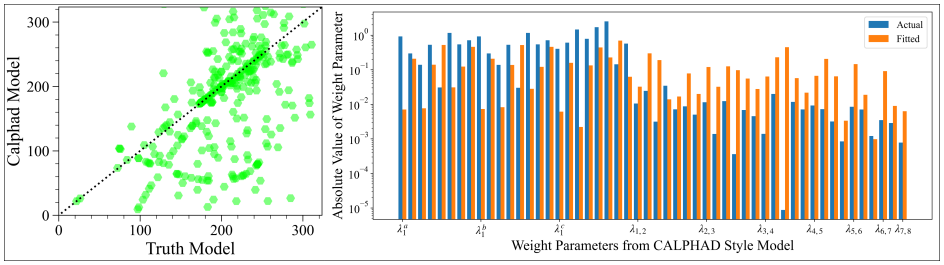


Figure 5.7: Model fit and parameter comparison for training step 3

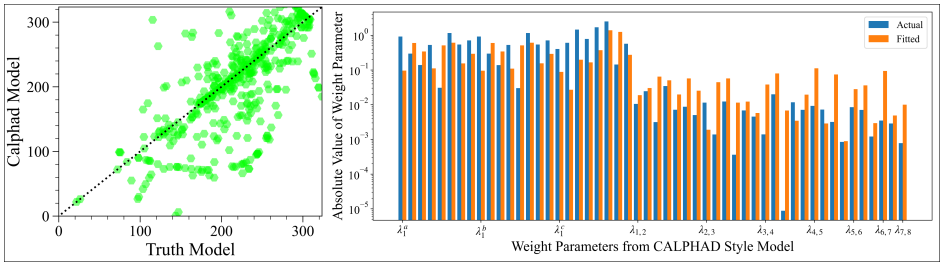


Figure 5.8: Model fit and parameter comparison for training step 4

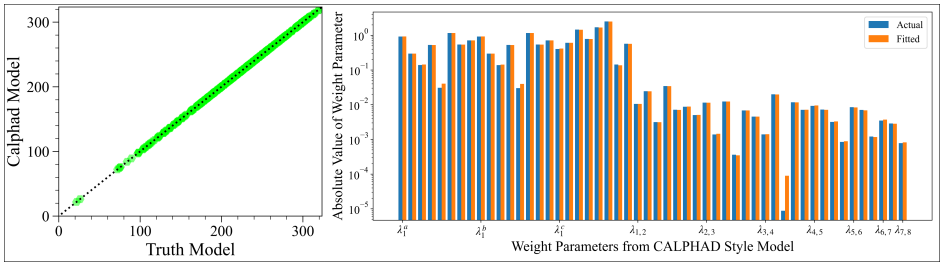


Figure 5.9: Model fit and parameter comparison for training step 5

6. THERMODYNAMIC MODELING WITH UNCERTAINTY USING THERMO-CALC AND GAUSSIAN PROCESS REGRESSORS*

6.1 Introduction

There are many ways to approach materials design within the Integrated Computational Materials Engineering (ICME) framework. In these ICME frameworks, one of the approaches is to formulate process-structure-property (PSP) relationships that can then be inverted to discover regions in the alloy-processing space with optimal performance. One of the key components of these PSP relationships is the process-to-structure relationships. Formulation of the process-structure relations in any alloy system is usually achieved in one of two ways.

Experimental methods can determine the phase fraction and composition of an alloy. Various metallography techniques can determine the phase fractions [72]–[75]. And by using spectrometry methods it is also possible to measure phase compositions. This approach provides the most accurate measure of a material's microstructure. But, it is costly, in both material and time costs. Thermodynamic models can predict the equilibrium state of the material [76]. However, thermodynamic models are significantly less accurate unless they have been properly assessed and validated against experiments. The advantage of using thermodynamic models is that they come with a significant cost benefit compared to experimental methods.

Due to the inherent heterogeneity of materials, it is helpful to be able to account for uncertainty in the thermodynamic models. There is a fairly large body of work in the literature that considers parametric uncertainty in Thermodynamic modeling with work by Olbricht [77], Otis [78], Honarmandi [79] and Duong [80] being good examples of a fully Bayesian approach to quantifying parametric uncertainty in CALPHAD models.

*Reprinted with permission from "Utilizing Gaussian processes to fit high dimension thermodynamic data that includes estimated variability," R. Couperthwaite, D. Allaire, and R. Arróyave, Computational Materials Science, p. 110133, Nov. 2020, doi: 10.1016/j.commatsci.2020.110133. Copyright 2020 Elsevier B.V.

All these approaches utilize Markov-chain Monte-Carlo (MCMC) sampling of the model parameters of the CALPHAD models used in thermodynamic simulations [77]–[82]. However, these methods rely on access to the thermodynamic models and good quality thermodynamic databases. In contrast, it is far more common to have access to commercially available thermodynamic software such as Thermo-Calc™ [83]. Utilizing such software tools comes with the drawback that access to the models and parameters is restricted, and so it is not possible to use the MCMC methods mentioned above to calculate the uncertainty in the model outputs. Therefore the current work proposes a more conventional surrogate modeling approach that is still capable of accounting for some sources of uncertainty.

The current work aims to develop a framework to transform a thermodynamic-based simulation framework connecting chemistry and phase constitution into a surrogate modeling scheme that can predict the volume fraction and phase compositions of a two-phase material. The inputs to this model are the composition of the material as well as the temperature of a single-stage intercritical annealing treatment.

A key aspect of the proposed work is the development of a process-structure model capable of offline evaluation (i.e. without the need to explicitly call a thermodynamic engine) that predicts the mean response accurately and also provides a measure of quantified uncertainty. This kind of model and the approach used in the current work will be generally applicable to any ICME approach requiring thermodynamic models. A further aspect of the current work is to insure that the models are computationally cheap to evaluate. The motivation for this is that design optimization frameworks typically require many function evaluations, and a computationally cheap model will add less overhead to the optimization framework.

Steel alloys are the focus of the current work since these materials are still of significant interest in many industries [74], [84]. Steel production entails significant variability, weighing differences between batches, spillage, evaporation, and inaccurate temperature measurements are a selection of the many process parameters that can introduce variability. In most experimental or production processes it is very difficult if not impossible to account for these parameters fully and so we would

classify this uncertainty as residual variability. While the authors acknowledge that this variability does involve some uncertainty that could be reduced by better processing methods, we assumed that significant process optimization has already been accomplished and any further reduction in uncertainty would be minimal.

To reiterate, the current work aims to achieve two goals. The first goal is to generate a surrogate for the thermodynamic model that ensures the models are cheap to query and accurately reproduce the thermodynamic model. In this way, the surrogate model becomes an off-line model that allows the thermodynamic response to be used in an ICME approach without explicit calls to the thermodynamic engine. The second goal is to propose a method of propagating uncertainty through the surrogate using parametric variability of the input parameters. The distributions from which to sample the input parameters are defined by the controllable composition of elements in production-grade steel.

6.2 Methods

In the current work, we utilize the Thermo-Calc™ [83] model as a simulator model of a real heat-treatment process. This simulator model provides information on the volume fraction and phase compositions of steel materials using a CALPHAD based approach. The current work aims to build a statistically based emulator, or surrogate, model using Gaussian Processes (GPs) that can accurately replicate the Thermo-Calc™ and be used to probe the parametric variability of the model.

As a result, the current work is divided into two stages. The first stage is the generation of a surrogate model based on data obtained from Thermo-Calc™ [83]. As already indicated, the current work uses GP models for the surrogate models. The second stage is the propagation of parametric variability through the surrogate model. In this second part, the composition of production-grade steel alloys informs the distribution shape for the parametric variability.

6.2.1 Thermodynamic Assessment with Thermo-Calc™

Thermo-Calc™ [83] utilizes the CALPHAD method to determine equilibrium phase fractions and compositions in multi-component systems. The current work uses the TCFE7 iron database for the thermodynamic data, and the Matlab interface for Thermo-Calc™ was used to complete the computations.

The focus of the current work is dual-phase (martensite-ferrite) steel alloys containing C, Si, and Mn (Fe in balance). We assumene that these alloys to have been subjected to a single-stage intercritical annealing heat treatment followed by quenching. As such, the input space for the models in the current work is the composition (wt%) of the C, Si, and Mn and the temperature for the intercritical annealing heat treatment (T_{IA}).

For the composition, we selected two common dual-phase steels to guide the limits of the region of interest. These alloys are DP-980 and DF-140T and Table 6.1 shows the composition of both alloys. It is necessary to define upper and lower bounds to the composition of the elements in the alloy to avoid the computational space from becoming too large. Therefore, we chose bounds that encompassed both alloys and ensured that the results would also apply to a larger range of alloys to allow for possible comparison with results in the literature. For the intercritical annealing temperature, we chose a range such that it was possible to produce material with 100% ferrite and 100% austenite within the input space. Table 6.1 shows the chosen bounds for the composition and intercritical annealing temperature.

Table 6.1: DP980 and DF140T Nominal Composition and the Composition of the design space in the current work

	C (wt.%)	Mn (wt.%)	Si (wt.%)	Fe (wt.%)	Temperature (°C)
DP980	0.09	2.15	0.60	bal.	-
DF140T	0.15	1.45	0.30	bal.	-
Model Input Bounds	0.0-1.0	0.0-3.0	0.0-2.0	bal.	650-850

Thermo-Calc™ calculates equilibrium phases, and so it is not possible to obtain the martensite fraction from Thermo-Calc™ [1]. Therefore, we used the Koistinen-Marburger relation shown in Equation 6.1 [85] to determine the fraction of austenite converted to martensite under different quenching temperature conditions. The Koistinen-Marburger relation requires the martensite start temperature (T_{ms}) and the quench temperature (T_Q). The martensite start temperature was calculated using the formula presented by Andrews [86] shown in Equation 6.2. The quench temperature was assumed to be 25°C.

$$V_f^{mart} = 1 - e^{(-0.011(T_{ms} - T_Q))}. \tag{6.1}$$

$$\begin{aligned} T_{ms}(K) = 812 - 423X_C - 30.4X_{Mn} \\ - 0.075X_{Si}. \end{aligned} \tag{6.2}$$

One of the key assumptions at this point is that due to the fast cooling during quenching there is insufficient time for diffusion to occur and so the composition of the martensite phase is the same as the high-temperature austenite phase.

To get the data for the construction of the emulator model, several different samplings of the design space were used. Firstly, uniform sampling with either 6, 7, 8, or 9 samples per dimension was used. This produced four data-sets of 1296, 2401, 4096, and 6561 samples. Secondly, Latin hypercube sampling of the space was used to generate the sampling points. Latin hypercube sampling is typically used when the sampling region is very large or has many dimensions. The approach subdivides each input dimension into the number of points required and then randomly combines these input values ensuring that each value on each dimension is used exactly once [87]. In this case, the number of samples obtained matched the data-set sizes of the uniform sampling.

The motivation for using both approaches was to compare which approach is capable of produc-

[1] Thermo-Calc™ versions from 2019a do include the possibility for calculating the martensite volume fractions, however, this method can only be used with the TCFE9, or later, database and also does not provide the phase composition and so was not utilized in the current work

ing the most accurate surrogate model with the smallest number of training points. Since reducing the number of training points will greatly reduce the amount of time that it takes to invert the GP kernel matrix a smaller training sample size will speed up queries to the surrogate model.

There are 7 outputs of the Thermo-Calc™ model. The first is the volume fraction of the martensite phase (calculated from the Koistinen-Marburger relation). The next six outputs are the elemental weight fractions of Si, Mn, and C in both the martensite and ferrite phase. Since there is an assumption of no losses during heat treatment, the elemental composition of the two phases must obey a mass balance. As such, it won't be necessary to model the full composition of both phases. However, the composition of both phases was considered to determine which would create a better surrogate model.

6.2.2 Source of Uncertainty

Uncertainty in both modeling and experimental work has multiple sources. For the current work, the sources of uncertainty in computer models will be discussed in detail. Some of the sources of uncertainty in experimental work have already been mentioned. The following is a summary of the sources of computational uncertainty identified in the work by Kennedy and O'Hagan [21], the interested reader is referred to their work for a more complete discussion.

- *Parameter Uncertainty*: This is the uncertainty associated with not knowing the true value of the parameters of the model. The assumption behind this uncertainty is that there is a single true value for the parameter.

- *Parametric Variability*: Parametric variability, in contrast to parametric uncertainty, is when the parameter in question does not have a unique value, but rather has a distribution of possible values.

- *Model Inadequacy*: Model inadequacy is the discrepancy between the output of the model and typically the mean of a real-world result. This assumes that the model is being utilized with the correct values for all parameters.

- *Residual Variability*: Residual variability encompasses two sources of uncertainty that are difficult if not possible to differentiate between. The first is that there could be missing inputs that if fully specified would reduce this variability, and the second is that the real world process itself might be stochastic.

- *Observation Error*: Observation error applies to a real-world process where there is often an error associated with the measurement of the output.

- *Code Uncertainty*: Despite it being theoretically possible to predict the outcome of a mathematical model, the complexity of many models and the requirements of running the model for hours or even days means that it is not possible to know the exact outcome from the model. This is classified as code or interpolation uncertainty.

The approach described by Kennedy and O'Hagan [21] is the basis for the approach used in the current work, however, for the current work the model is not calibrated against experimental results. The approach can be expanded to include experimental results for future work. Since experimental results are not considered in the current work, the sources of uncertainty that will be considered are observation error and code uncertainty. The uncertainty in the output of the surrogate model will be introduced by considering parametric variability in the inputs to the surrogate model.

The authors acknowledge that residual variability in the production of steels is very distinct from the parametric uncertainty or variability in the emulator, or surrogate, model. However, the current work proposes that the residual variability can provide information to inform the parametric variability of the surrogate model. How this is to be achieved is to use the compositional variation of production-grade steels to define the distributions for the input parameters to the surrogate model.

6.2.3 Gaussian Process Fitting of Thermodynamic Results

The current work aims to define a surrogate model or emulator of the Thermo-Calc™ model. This can be defined as the determination of a function f such that $f : \chi \longrightarrow \mathcal{Y}$. In this case,

$Y(\boldsymbol{x}) \in \mathcal{Y} \subseteq \mathbb{R}$ is the univariate output of the Thermo-Calc™ model at a given input $\boldsymbol{x} \in \chi \subseteq \mathbb{R}^q$, where χ is the q-dimensional domain of interest or design space. The measurement, or observation, error ($\epsilon_{obs}(\boldsymbol{x})$) is defined as the uncertainty in the measurement of $Y(\boldsymbol{x})$, however, since the Thermo-Calc™ model is a deterministic model there is no error associated with the result. As a result, this term will be replaced by a uniform noise variance in the current work to ensure computational stability. This is discussed in further detail later.

$$Y(\boldsymbol{x}) = f(\boldsymbol{x}) + \epsilon_{obs}(\boldsymbol{x}). \tag{6.3}$$

The current work uses a GP for the emulator model. GPs have become one of the most widely used statistical models [88] since they provide the ability to analyze and quantify uncertainty in functions, provide excellent flexibility through the different covariance functions that can be employed as well as having attractive statistical properties.

A GP is a non-parametric statistical model that defines a stochastic process $f(\boldsymbol{x})$ such that all the finite distributions of the model are assumed to be multi-variate normal. As a result of this, the joint probability distribution of the outputs from the stochastic process for any finite set of inputs $\mathbb{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$ may be modeled as an n-dimensional multivariate normal distribution:

$$p(f(\boldsymbol{x}_1), ..., f(\boldsymbol{x}_n)) \sim \mathcal{N}_n(\boldsymbol{\mu}, \boldsymbol{C}). \tag{6.4}$$

where $\boldsymbol{\mu}$ is the mean vector and $\boldsymbol{C}$ is the covariance function. These are defined by a mean function $\mu(\cdot)$ and a covariance function $C(\cdot, \cdot)$ with the following properties:

$$\mu(\boldsymbol{x}_i) = \boldsymbol{\mu}_i = \mathbb{E}\left[f(\boldsymbol{x}_i)\right]. \tag{6.5}$$

$$C(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{C}_{i,j} = cov\left[f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)\right]. \tag{6.6}$$

Considering this context, we will define a GP as $f(\cdot) \sim \mathcal{GP}(\mu, C)$. A more detailed explanation

of this kind of stochastic process is provided in the work by Rasmussen and Williams [58].

The covariance function captures the spatial dependence between two different input locations, and along with the mean function plays a role in the final probability distribution of the outputs of the stochastic process. The probability distribution of the surrogate model outputs defines the interpolation uncertainty, or to use the terminology defined by Kennedy and O'Hagan, code uncertainty [21].

This approach for defining the GP model allows for the development of the model definition in Equation 6.3 to include two sources of uncertainty. This approach defines $f(\boldsymbol{x})$ as the mean response of the GP, $\epsilon_{obs}(\boldsymbol{x})$ as the observational error, and $\epsilon_{code}(\boldsymbol{x})$ as the interpolation error from the GP. Where the distribution of the code error is defined by $\epsilon_{code}(\boldsymbol{x}) \sim \mathcal{N}(0, C)$.

$$Y(\boldsymbol{x}) = f(\boldsymbol{x}) + \epsilon_{obs}(\boldsymbol{x}) + \epsilon_{code}(\boldsymbol{x}). \tag{6.7}$$

The observational error ($\epsilon_{obs}(\boldsymbol{x})$) can be handled in two ways. The first is to measure the error of each observation explicitly, while the second is to use the observational error as an additional parameter in the building of the GP. This second approach is referred to as using a nugget [89].

There are many different possible covariance functions available for use with GPs. Rasmussen and Williams [58] provide definitions of many of the more commonly used covariance functions. Two of these covariance functions are utilized in the current work. The squared exponential and Matérn($v = 5/2$) covariance functions.

The squared exponential function calculates the covariance of the input space as a weighted euclidean distance between the input variables and can be parameterized according to Equation 6.8, where n is the number of dimensions, σ_f^2 is the signal variance, and l is the characteristic length scale or smoothness parameter.

$$C(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \, exp\left(-\frac{1}{2}\sum_{h=1}^{n}\left[\frac{(x_{i,h} - x_{j,h})}{l_h}\right]^2\right). \tag{6.8}$$

The squared exponential function was implemented in MATLAB using code based on the ap-

proach developed by Ghoreishi *et al*. [13].

The Matérn class of covariance functions is defined in a *single dimension* by Equation 6.9, where K_v is a modified Bessel function[58]. However, it is more common to define the function by specifying a specific value for v. One of the more commonly used values is $v = 5/2$ [58]. This choice reduces the covariance function to that shown in Equation 6.10. This equation shows a generic multi-dimension representation of the covariance function with $v = 5/2$, where n is the number of dimensions, and l_h is the characteristic length scale of dimension h.

$$C(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \left[\frac{2^{1-v}}{\Gamma(v)} \left(\frac{\sqrt{2v}(x_i - x_j)}{l} \right)^v K_v \left(\frac{\sqrt{2v}(x_i - x_j)}{l} \right) \right]. \tag{6.9}$$

$$C(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \sum_{h=1}^{n} \left(1 + \frac{\sqrt{5}(x_i - x_j)}{l_h} + \frac{5(x_i - x_j)^2}{3l_h^2} \right) exp \left(-\frac{\sqrt{5}(x_i - x_j)}{l_h} \right). \tag{6.10}$$

The implementation of the Matérn covariance function was done within Python using the "George.py" module [90].

Since the input parameters have different units, all the inputs were scaled to the interval $[0, 1]$. For mathematical convenience, the outputs from Thermo-Calc™ were standardized to have a mean of zero and a variance of 1. This approach allows us to specify the mean function as $\mu(\boldsymbol{x}) = \mathbf{0}$. The observation error term defined in Equation 6.3 is added to the covariance function when calculating the output of the GP model. However, if the results do not contain a measurement error $\epsilon_{obs}(\cdot)$, as in the case of the output from a deterministic model, a small value, often referred to as a nugget [89], can be added in place of observation error to provide numerical stability in the calculation of the matrices and their inverses. Doing this assumes that the errors are all identically and independently distributed with a normal distribution of zero mean and σ_n^2 variance, $\epsilon(\boldsymbol{x}) \sim \mathcal{N}(0, \sigma_n^2)$. The variance of the errors (σ_n^2) is also referred to as the noise variance [58].

There are two standard approaches to optimizing the values of the hyper-parameter for GPs. The first is to use gradient-based approaches [58]. The second is to use Bayesian approaches such

as Markov-chain Monte-Carlo methods [21]. It was chosen to use the gradient-based approach in the current work since the gradient-based approach is typically less computationally expensive.

While the gradient-based approach is usually less computationally costly than the Bayesian approach it has been noted that there is a possibility for there to be multiple local optima in the parameter space. Rasmussen and Williams [58] discuss this briefly and indicate that this is more likely to occur when there is less data available since there will be more combinations of the parameters that can provide a sufficient fit to the data.

Due to the chance of local optima in the hyper-parameter space, the optimization used a multi-start approach in an attempt to avoid having the optimization process get stuck in local optima. Since all the input values were scaled to the interval $[0, 1]$ the initial guesses for the length scale hyperparameters were selected from that interval. As discussed, to aid the inversion of the matrices a nugget term with $\sigma_n^2 = 0.05$ was included.

For the current work, it is important that the surrogate model is an accurate representation of the Thermo-Calc™ data, therefore, the results from the GP were validated against 10,000 data points calculated by uniform sampling with 10 samples for each dimension. The coefficient of determination (Equation 6.12) was used as the measure of fit.

In addition to measuring the coefficient of determination, the results were plotted against the Thermo-Calc™ results for the two alloys of interest in the current work. For this comparison, the composition of the alloys was fixed and the temperature varied over the design range. Since there is no training data that directly corresponds to the Thermo-Calc™ data for these two alloys, this method helps provide visual confirmation of how well the model is predicting general values of the design space.

6.2.4 Uncertainty Propagation

As discussed earlier, the compositional variation of production-grade steels informs the parametric variability. Since the published steel grades show the possible variation in the elemental composition it is possible to define a maximum and minimum value for any given input.

Two approaches were used in the current work, however, it is noted that these were chosen for

convenience rather than being definitive methods for approaching this kind of problem. The first method was to assume that the input distribution was a uniform distribution between the maximum and minimum values for the input. This is considered a non-informative approach. The second method was to assume that the input was normally distributed and that the maximum and minimum values define a distance of two standard deviations away from the mean. This approach does make a strong assumption about the input distribution being normal.

Using these two distributions the parametric variability of the model is assessed by sampling from the input distributions and then calculating the mean and variance of the mean output from the GP. This is one of the simpler methods for obtaining the parametric variability since it doesn't take into account the code uncertainty of the GP. This also simplifies the definition of the parametric variability to be a multivariate normal distribution with mean 0 and variance σ ($\epsilon_{par} \sim \mathcal{N}(0, \sigma)$) and the model definition given in Equation 6.3 and developed in Equation 6.7 can be further developed to include the uncertainty due to parametric variability.

$$Y(\boldsymbol{x}) = f(\boldsymbol{x}) + \epsilon_{obs}(\boldsymbol{x}) + \epsilon_{code}(\boldsymbol{x}) + \epsilon_{par}(\boldsymbol{x}). \tag{6.11}$$

6.3 Results

6.3.1 Building of the surrogate model

The first tests involved building the GP model with the squared exponential kernel. The optimization of the GP hyperparameters was conducted with training points from both the uniform and Latin hypercube sampling. For each sampling method, the best performing hyperparameters were recorded for the seven output values from Thermo-Calc. The hyperparameters of interest in the current work are the characteristic length scale (l) and the signal variance (σ_f), since the noise variance, or nugget (σ_n), was kept constant.

To measure the accuracy of the GP model, the coefficient of determination was used. The Coefficient of determination uses the ratio between the residual sum of squares (Equation 6.13) and the total sum of squares (Equation 6.14), where y_i is a Thermo-Calc™ output, $\bar{y}$ is the mean

of all Thermo-Calc™ outputs and f_i is the surrogate model value corresponding to the inputs of Thermo-Calc™ output y_i. The residual sum of squares is the square of the distance between the predicted values and the true values, while the total sum of squares is the distance between the true values and the mean of the true values. The coefficient of determination is defined for the interval $[0, 1]$ with a value of 1 indicating a perfect fit, and a value of 0 indicating that the prediction is no better than the mean. Any values outside of this range indicate that the predicted values from the model are further from the true values than the mean.

$$CoD = 1 - \frac{SS_{res}}{SS_{tot}}. \tag{6.12}$$

$$SS_{res} = \sum_i (y_i - f_i)^2. \tag{6.13}$$

$$SS_{tot} = \sum_i (y_i - \bar{y})^2. \tag{6.14}$$

The accuracy of each of the GPs using the optimum length scale and noise variance hyper-parameter results are shown in Table 6.2. As can be seen, the accuracy of the GP increases with increasing sample size. This effect is more noticeable with the Latin hypercube training data than the uniform training data. What is interesting to note is that the uniform training data has a much higher accuracy against the test data than the same number of training points generated from a Latin hypercube sampling. As already noted, Latin hypercube sampling is typically used when only a very sparse sampling of the input space is possible. Therefore, for a 4-dimensional problem such as is considered in the current work, the sample size of 1000 to 7000 is sufficiently large for the uniform sampling to perform better. Choosing smaller sample sizes or increasing the dimensionality of the problem will almost certainly result in the Latin hypercube sampling performing better than the uniform sampling.

The scatter plots in Figure 6.1 show how as the sample size increases the fit improves. However, this representation also provides a further observation. The fit deviates most significantly for higher

Table 6.2: Coefficient of determination for the 10,000 test data points for each of the sample sets used in both uniform and LHS sampling

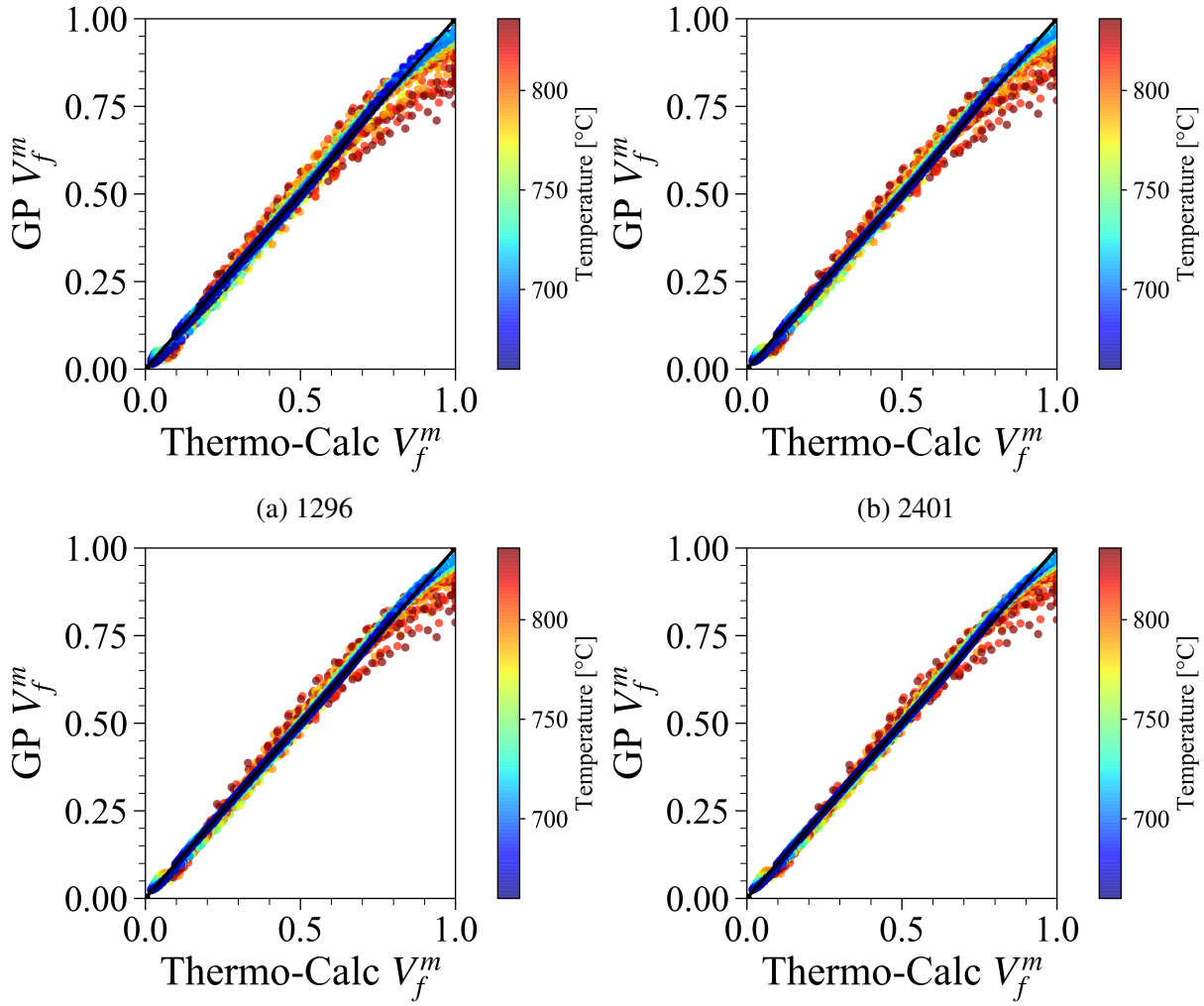
	Uniform Data				LHS Data			
	1296	2401	4096	6561	1296	2401	4096	6561
V_f^{mart}	0.99	0.99	1.00	1.00	0.48	0.48	0.95	1.00
X_C^{mart}	0.98	0.99	0.99	0.99	0.57	0.42	1.00	0.97
X_{Si}^{mart}	0.99	1.00	1.00	1.00	0.57	0.53	0.41	1.00
X_{Mn}^{mart}	0.99	1.00	1.00	1.00	0.72	0.54	1.00	0.99
X_C^{ferr}	0.97	0.98	0.99	0.99	0.66	0.54	0.27	1.00
X_{Si}^{ferr}	0.99	1.00	1.00	1.00	0.46	0.31	1.00	1.00
X_{Mn}^{ferr}	0.98	0.99	0.99	1.00	0.70	0.62	1.00	1.00

temperature samples. As such, if the input space was reduced to lower temperatures, it might be possible to achieve the same accuracy with fewer training samples.

Using the data for the two alloys DP980 and DF140T, the trained GPs were used to predict the outputs from the Thermo-Calc model to assist in providing a visual representation of how well the GPs are performing. When comparing the GP output with that of Thermo-Calc for the two alloys specifically, the performance is not as good as the performance on the test points, particularly when it concerns the composition of the phases in the alloy. Table 6.3 shows the results for the DP980 alloy. The DF140T alloy results showed a similar trend.

Table 6.3: Coefficient of determination for the DP980 data set when using the samples as specified with the Squared Exponential covariance function

	Uniform Data				LHS Data			
	1296	2401	4096	6561	1296	2401	4096	6561
V_f^{mart}	0.99	0.99	0.98	0.98	0.26	-0.22	0.92	0.97
X_C^{mart}	0.76	0.70	0.66	0.64	-1.25	-5.17	0.68	0.66
X_{Si}^{mart}	-159.84	-124.60	-100.29	-92.65	0.15	0.02	-2.21	-97.55
X_{Mn}^{mart}	-0.20	-0.47	-0.57	-0.64	-3.37	-7.40	-0.67	-0.47
X_C^{ferr}	0.07	-0.61	-1.30	-1.74	-11.61	-10.95	-4.18	-1.21
X_{Si}^{ferr}	-284.15	-195.05	-146.80	-130.84	0.00	-1.19	-112.38	-116.29
X_{Mn}^{ferr}	-51.08	-60.73	-61.58	-63.01	-31.91	-93.48	-53.73	-64.72



(a) 1296

(b) 2401

(c) 4096

(d) 6561

Figure 6.1: Scatter plot of the 10,000 volume fraction results from Thermo-Calc™ with the results from the GP with a squared exponential covariance function.

The results of the fit to the composition of the phases in the two test alloys is slightly concerning and also surprising. The fit to the test points for the composition results is almost perfect, while the fit to the composition values of the test alloys is very bad. As a test, the number of training points was increased to 10,000 and the results still showed the same problem.

The next step was to implement the GP model with a Matérn (5/2) covariance function. This approach resulted in a significantly better fit for the composition of the phases, Table 6.4. This indicates that the Matérn (5/2) is a better covariance function to use in the current work. Despite the

better fit, Figure 6.2 shows that the interpolation error of the composition values is still significant. The error has been truncated at zero since a negative composition has no physical meaning.

The results in Table 6.4 also show that the predictions for the martensite phase composition are better than for the ferrite phase. As such, it would be best to use the GPs to predict the martensite composition and then calculate the ferrite composition using the mass balance of the elements.

Table 6.4: Results from the Matérn Kernel fit using only the sample set with 6561 samples and uniform sampling

	DF140T	DP980
V_f^{mart}	0.991	0.985
X_C^{mart}	0.999	0.998
X_{Si}^{mart}	0.952	0.899
X_{Mn}^{mart}	0.997	0.995
X_C^{ferr}	0.981	0.987
X_{Si}^{ferr}	0.79	0.719
X_{Mn}^{ferr}	0.91	0.869

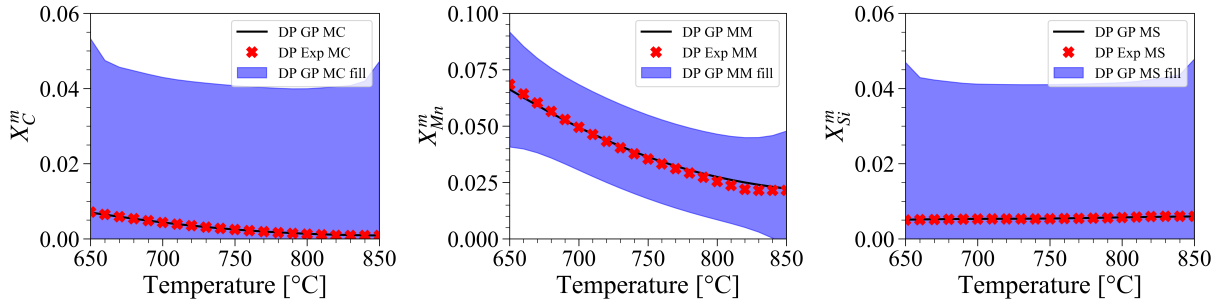


Figure 6.2: Comparison of the GP outputs and the Thermo-Calc™ results for the elemental compositions of the martensite phase for the DF140T alloy. These show the very large confidence interval around the GP prediction for the phase composition.

Since one of the aims of the current work is to have a fast surrogate model, the time taken to calculate the 10,000 samples was measured for each of the GPs constructed. This measurement was

done by repeating the calculation of the test set 20 times and averaging the results. These results are shown in Table 6.5. These show that even for the largest training sample size the time taken to calculate 10,000 data points is reasonably small. Considering this, and the increased accuracy that using the larger training set provides, it was decided that the largest training set would be used in the subsequent analyses.

The timing of the code was done by building and training the GP model and then querying 10,000 data points from the model. This measurement was repeated 20 times and the results were averaged. These results are shown in Table 6.5. These show that even for the largest training sample size the time taken to calculate 10,000 data points is reasonably small (17s for the MATLAB implementation and 8s for the Python Implementation. This shows that for this particular application, it is not necessary to sacrifice the accuracy of the largest dataset for a faster GP model since the times taken by the model built from the largest dataset are small enough to not significantly impact an optimization approach.

Table 6.5: Time taken to build the GP and query 10,000 data points simultaneously

Training	Time to Test 10,000 points (s)	
Sample Size	MATLAB	Python
1296	5.23 ± 0.05	0.70 ± 0.02
2401	6.93 ± 0.11	1.53 ± 0.05
4096	10.70 ± 0.09	3.63 ± 0.05
6561	16.84 ± 0.24	7.83 ± 0.08

6.3.2 Parameter Variability

Using the two methods described the parametric variability was added to the results from the surrogate model. As discussed in the methods, the approach in the current work was to predict the mean response at each sampling from the composition and temperature input space and then find the average and variance of this mean output. This is used to define the normally distributed parametric error. Following Equation 6.11 this parametric error was added to the code or interpo-

lation error and the 95% confidence interval was calculated. These results for the two sampling procedures are shown in Figure 6.3.

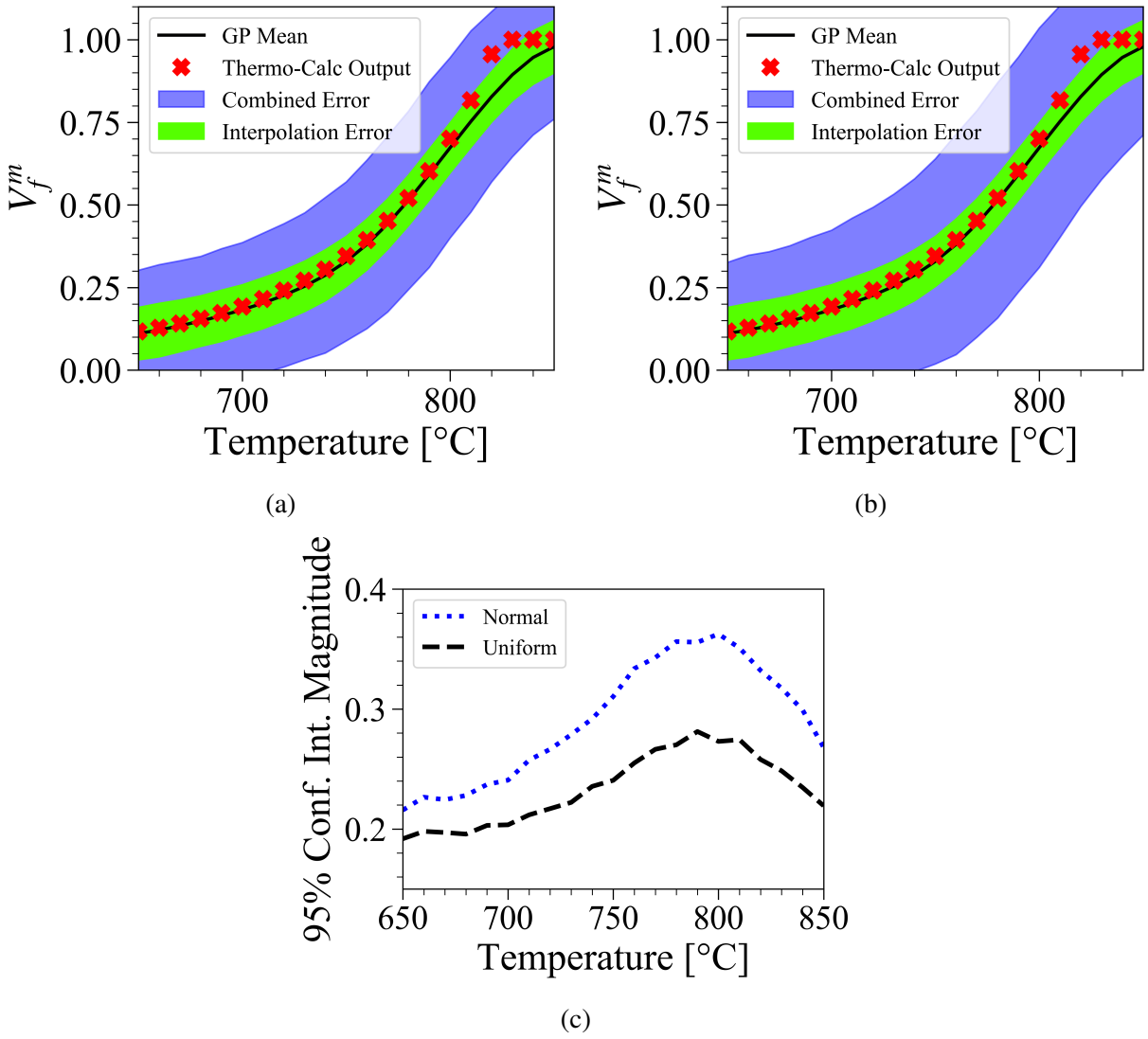


Figure 6.3: Surrogate model outputs for the DP980 alloy showing 95% confidence intervals defined by interpolation error only (green), and the combination of interpolation error and parametric variability (blue) for samplings of the parametric error from (a) Uniform and (b) Normal Distributions with (c) showing a comparison of the magnitude of one side of the 95% confidence interval for the two sampling approaches.

As can be seen the parametric error approximately doubles the 95% confidence interval of the

data. However, the uniform sampling results in a smaller confidence interval. The most likely reason for this is that using the interval between the bounds results in a smaller sampling region around the mean. The normal distribution allows values outside of the region in the uniform sampling approach.

6.4 Discussion

During the current work, namely fitting a GP surrogate model to Thermo-Calc™ results several noteworthy results were obtained. Firstly, it was found that using a squared exponential function for the GP provided a good fit to the volume fraction output of the code, but failed when fitting the composition of the phases. The exact reason for this is not known, however, this result does show that testing of multiple covariance functions is necessary to ensure the best fit for the results.

The second noteworthy finding was that Latin hypercube sampling produces GPs with a worse fit to the data when compared to the same number of samples from a uniform sampling procedure. This would possibly be a result of the Thermo-Calc™ output being relatively smooth over most of the domain, however, more testing would be required to confirm this. The second reason for the poor performance of the Latin hypercube approach is that the design space is still small enough to be effectively sampled by uniform sampling. Therefore, when expanding this work to larger design spaces with more dimensions, the Latin hypercube sampling will become a better approach since it won't be computationally possible to consider uniform sampling.

The procedure followed in the current work developed a set of GP surrogate models that were able to separately account for three sources of uncertainty in the modeling process. These sources were observation error, code uncertainty, and parametric variability. While in the current work, the observation error is neglected since the Thermo-Calc™ result is a deterministic result, this approach would be able to account for this error. This would be done by including the observation error as the noise variance (σ_n) for each of the observations.

The code uncertainty in the models of the current work is reasonable for the prediction of the volume fraction, however, it was observed that the code uncertainty for the elemental composition of the phases was significantly larger. This could potentially be decreased by using a larger sample,

however, the time cost of the larger training set would need to be tested to determine the optimal size that can lower the interpolation uncertainty while not increasing the computational time to unreasonable levels.

Parametric variability was added to the surrogate model by using two distributions, uniform and normal, defined by the residual variability of production-grade steel. This approach was found to produce reasonable uncertainty to the results. Both distributions of the input parameters approximately doubled the size of the 95% confidence interval. However, the uniform distribution had a smaller 95% confidence interval. As a result, using the normal distribution will result in a more conservative estimate of the error.

6.5 Conclusions

The current method developed a set of GP surrogate models that can be easily integrated with ICME materials design approaches. These models provide basic information on the microstructure of a material following a simple heat-treatment process. Further, these models can be evaluated quickly, which means that they will not increase the computational time of an optimization approach significantly.

Using the residual uncertainty in the composition of production-grade steel materials, the distributions for the calculation of parametric uncertainty were defined. This provides the opportunity for propagating this uncertainty through structure-to-property models in the PSP chains used in ICME approaches.

While the surrogate model developed in this work can be easily integrated into any existing ICME approach, the intention is to integrate this model into the multi-information source fusion method presented by Ghoreishi *et al* [13]. This is to expand the work on the multi-information source fusion approach to include materials composition and processing parameters since this was identified as an area for development in this approach.

As a final note, while these results are a useful addition to the modeling of material microstructures, the volume fraction of phases and phase composition are only some of the variables that are needed to fully determine the mechanical properties of a material. One of the other significant

parameters that are necessary is the grain size of the phases. Therefore, while the current work has proven that it is possible to obtain good results from a GP fit to Thermo-Calc™ data, in addition to the expansion of the thermodynamic modeling explained earlier, it is necessary to expand the current work to include calculations of the grain size of the material.

7. SUMMARY AND FUTURE WORK

In summary, this work presents a Bayesian Optimization Framework for materials design. This Framework enables the easy implementation of Bayesian optimization as a tool for material design. The Framework currently offers three different optimization approaches a Reification approach, a Batch Bayesian Optimization approach, and the Barefoot approach, combining the Reification and Batch approaches. All the testing conducted on the Framework has shown that all three of these approaches are beneficial for material design.

Of particular importance to note is the distinction between what reduced order models are available. The results in the present work indicate that when good quality reduced order models are available then the Barefoot approach is likely to perform better than the Batch only approach. If these reduced order models are not available, or the reduced order models do not provide sufficient information about the ground truth, then the Batch only approach is the better alternative to use.

The Framework currently provides access to all the most common acquisition functions used in Bayesian Optimization (except entropy search) and a somewhat limited selection of surrogate models. The Framework is capable of constrained optimization, with all the standard constraints able to be applied. And finally, it is possible to increase the speed of the calculations by extending the Framework to work on multiple nodes in a high-performance computing cluster.

To further bolster the Framework's usefulness, it has been built specifically to handle the types of situations that arise when doing material design. The Framework is coded in Python, which enables interfacing with almost any computational model available. We can also interface the Framework with experimental materials analysis in two ways, namely by direct interface with so-called self-driving or autonomous labs and through an output file with test parameters if the experiment must be conducted offline.

The Framework has been tested on two materials design problems and some standard optimization test functions. These have all been done for single-objective optimization calculations. In these calculations, it has been demonstrated that the Framework can operate in input spaces with

up to eight dimensions. While initial testing of the Framework for multi-objective optimization has been conducted, further testing is still required. The future work emanating from this project can be classified into two broad categories. Firstly, there is further development to the BAREFOOT Framework that will both increase functionality and performance. The second category of future work is the application of the Framework in material research problems.

For the further development of the BAREFOOT Framework, there are still many aspects that we can improve. These are divided into two groups, Framework optimization and functionality improvement. For Framework optimization, there is still room for improvement to increase the speed at which the Framework conducts operations. When the calculations are in low dimension design spaces, the cost of running the Framework is minimal. However, in larger dimension spaces, the computational cost starts to become quite significant. One of the areas that needs specific focus is calculating the fused model in the Reification-based approaches. The possible improvement, in this case, is to implement the Reification object using multi-objective GPs. The current method requires a separate GP model for each reduced-order model and the discrepancy model associated with it. The construction of all these GP models can take a significant portion of time. A multi-objective GP model might solve the problem since the covariance matrix will only need to be calculated once. The other Framework optimization improvements required are to ensure that the code is structured so that we can add additional approaches to the Framework with ease. This leads to the possible functionality improvements that we should implement. The first functionality improvement is implementing n-objective optimization since the Framework can only do two objective optimizations. Other optimization approaches have shown promise, with a good example being the active-subspace optimization approach. Implementation of this optimization approach in the Framework could be highly beneficial. Finally, it would be advantageous to implement a more comprehensive selection of covariance functions since the Framework's covariance functions are currently very limited. In addition to new covariance functions, new research into the use of Bayesian Additive Regression Trees (BART) has shown significant improvement over the use of Gaussian Process Models. As such, it would be worthwhile to investigate the implementation of

such models in the Framework as an alternative to the GP models. Finally, after the results showing that the training of the reduced-order models could be beneficial, it would be worthwhile to implement several machine learning models to use as reduced-order models. This will enable the Framework to be used simultaneously as an optimization approach and a model training approach.

When considering potential applications of the BAREFOOT Framework, several possibilities already exist. We will continue with the RHEA problem that has already been started. The immediate aim will be to expand the problem to a two-objective optimization that considers both the bulk modulus and the shear modulus. The shear modulus can be approximated quite easily using the Bulk Modulus and the average Poisson's Ratio of the alloy in question. Considering these two properties would allow us to have a measure of the ductility of the RHEA. At the same time, it will be necessary to include a constraint for single-phase Body Centred Cubic (BCC) regions in the design space. This would require the use of thermodynamic calculations and the integration of Thermo-Calc™with the Framework. Currently, this is a possibility since Thermo-Calc™is available on the same High-Performance Computing Cluster as the BAREFOOT Framework is running on. However, the nature of the calculations might cause the Thermo-Calc™evaluations to be too time-consuming. In which case, a method similar to that followed in Chapter 6 of this work will be necessary to speed up the calculations. A final application that is being investigated is for the framework to be used in research on Zn Flow Batteries. In this particular problem the aim is to optimize the solution used in the Zn Flow Batteries to maximize the coulombic efficiency of the battery. This application would require some testing since the problem is inherently categorical in nature. To date the Barefoot framework has not been tested on categorical problems, however, it should be possible to handle these problems with some modifications to either the GP models that are used, or the sampling approaches that are implemented in the Framework.

REFERENCES

[1] J. Allison, D. Backman, and L. Christodoulou, "Integrated computational materials engineering: A new paradigm for the global materials profession," *Jom*, vol. 58, no. 11, pp. 25–27, 2006.

[2] J. J. de Pablo, B. Jones, C. L. Kovacs, V. Ozolins, and A. P. Ramirez, "The materials genome initiative, the interplay of experiment, theory and computation," *Current Opinion in Solid State and Materials Science*, vol. 18, no. 2, pp. 99–117, 2014.

[3] G. B. Olson, "Designing a new material world," *Science*, vol. 288, no. 5468, p. 993, May 2000. DOI: 10.1126/science.288.5468.993.

[4] A. E. Tallman, L. P. Swiler, Y. Wang, and D. L. McDowell, "Uncertainty propagation in reduced order models based on crystal plasticity," *Computer Methods in Applied Mechanics and Engineering*, vol. 365, p. 113 009, Jun. 2020. DOI: 10.1016/j.cma.2020.113009.

[5] T. A. Parthasarathy and R. John, "A microstructure-sensitive location-specific design tool for predicting the yield and creep behavior of LSHR ni-base superalloy," *Materials Science and Engineering: A*, vol. 712, pp. 502–512, Jan. 2018. DOI: 10.1016/j.msea.2017.11.097.

[6] D. L. McDowell, "Microstructure-sensitive computational structure-property relations in materials design," in *Computational Materials System Design*, D. Shin and J. Saal, Eds., Cham: Springer International Publishing, 2018, pp. 1–25, ISBN: 978-3-319-68280-8. DOI: 10.1007/978-3-319-68280-8_1.

[7] J. H. Panchal, S. R. Kalidindi, and D. L. McDowell, "Key computational modeling issues in Integrated Computational Materials Engineering," *Computer-aided multi-scale materials and product design*, vol. 45, no. 1, pp. 4–25, Jan. 2013, ISSN: 0010-4485. DOI: 10.1016/

j.cad.2012.06.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010448512001352.

[8] S. P. Han, "A globally convergent method for nonlinear programming," *Journal of Optimization Theory and Applications*, vol. 22, no. 3, pp. 297–309, Jul. 1977. DOI: 10.1007/BF00932858.

[9] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st. USA: Addison-Wesley Longman Publishing Co., Inc., 1989, ISBN: 0-201-15767-5.

[10] M. Pincus, "Letter to the editor—a monte carlo method for the approximate solution of certain types of constrained optimization problems," *Operations Research*, vol. 18, no. 6, pp. 1225–1228, 1970, _eprint: https://doi.org/10.1287/opre.18.6.1225. DOI: 10.1287/opre.18.6.1225.

[11] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1995, 1942–1948 vol.4.

[12] J. Močkus, "On bayesian methods for seeking the extremum," in *Optimization Techniques IFIP Technical Conference Novosibirsk, July 1–7, 1974*, G. I. Marchuk, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 1975, pp. 400–404, ISBN: 978-3-540-37497-8.

[13] S. F. Ghoreishi, A. Molkeri, A. Srivastava, R. Arroyave, and D. Allaire, "Multi-information source fusion and optimization to realize ICME: Application to dual-phase materials," *Journal of Mechanical Design*, vol. 140, no. 11, 2018.

[14] S. F. Ghoreishi, A. Molkeri, R. Arróyave, D. Allaire, and A. Srivastava, "Efficient use of multiple information sources in material design," *Acta Materialia*, vol. 180, pp. 260–271, 2019.

[15] W. D. Thomison and D. L. Allaire, "A model reification approach to fusing information from multifidelity information sources," in *19th AIAA Non-Deterministic Approaches Conference*, ser. AIAA SciTech Forum, American Institute of Aeronautics and Astronautics, Jan. 2017. DOI: 10.2514/6.2017-1949.

[16] D. W. Shahan and C. C. Seepersad, "Bayesian network classifiers for set-based collaborative design," *Journal of Mechanical Design*, vol. 134, no. 71001, Jun. 2012. DOI: 10.1115/1.4006323.

[17] C. Seepersad, "Challenges and opportunities in design for additive manufacturing," *3D Print. Addit. Manuf.*, vol. 1, pp. 10–13, 2014. DOI: 10.1089/3dp.2013.0006.

[18] J. Mullins and S. Mahadevan, "Bayesian uncertainty integration for model calibration, validation, and prediction," *Journal of Verification, Validation and Uncertainty Quantification*, vol. 1, no. 11006, Feb. 2016. DOI: 10.1115/1.4032371.

[19] J. Matthews, T. Klatt, C. Morris, C. C. Seepersad, M. Haberman, and D. Shahan, "Hierarchical design of negative stiffness metamaterials using a bayesian network classifier1," *Journal of Mechanical Design*, vol. 138, no. 4, 2016. DOI: 10.1115/1.4032774.

[20] C. Li and S. Mahadevan, "Role of calibration, validation, and relevance in multi-level uncertainty integration," *Reliability Engineering & System Safety*, vol. 148, pp. 32–43, 2016. DOI: https://doi.org/10.1016/j.ress.2015.11.013.

[21] M. C. Kennedy and A. O'Hagan, "Bayesian calibration of computer models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 3, pp. 425–464, Jan. 2001. DOI: 10.1111/1467-9868.00294.

[22] N. M. Al Hasan, H. Hou, S. Sarkar, S. Thienhaus, A. Mehta, A. Ludwig, and I. Takeuchi, "Combinatorial synthesis and high-throughput characterization of microstructure and phase transformation in NiTiCuV quaternary thin-film library," *Engineering*, May 2020. DOI: 10.1016/j.eng.2020.05.003.

[23] M. A. Melia, S. R. Whetten, R. Puckett, M. Jones, M. J. Heiden, N. Argibay, and A. B. Kustas, "High-throughput additive manufacturing and characterization of refractory high entropy alloys," *Applied Materials Today*, vol. 19, p. 100 560, Jun. 2020. DOI: 10.1016/j.apmt.2020.100560.

[24] Y. Lyu, Y. Liu, T. Cheng, and B. Guo, "High-throughput characterization methods for lithium batteries," *High-throughput Experimental and Modeling Research toward Advanced Batteries*, vol. 3, no. 3, pp. 221–229, Sep. 2017. DOI: 10.1016/j.jmat.2017.08.001.

[25] P. Liu, B. Guo, T. An, H. Fang, G. Zhu, C. Jiang, and X. Jiang, "High throughput materials research and development for lithium ion batteries," *High-throughput Experimental and Modeling Research toward Advanced Batteries*, vol. 3, no. 3, pp. 202–208, Sep. 2017. DOI: 10.1016/j.jmat.2017.07.004.

[26] T. Wang, Y. Xiong, Y. Wang, P. Qiu, Q. Song, K. Zhao, J. Yang, J. Xiao, X. Shi, and L. Chen, "Cu3erte3: A new promising thermoelectric material predicated by high-throughput screening," *Materials Today Physics*, vol. 12, p. 100 180, Mar. 2020. DOI: 10.1016/j.mtphys.2020.100180.

[27] X. Zhang and Y. Xiang, "Combinatorial approaches for high-throughput characterization of mechanical properties," *High-throughput Experimental and Modeling Research toward Advanced Batteries*, vol. 3, no. 3, pp. 209–220, Sep. 2017. DOI: 10.1016/j.jmat.2017.07.002.

[28] D. Ginsbourger, R. Le Riche, and L. Carraro, "Kriging is well-suited to parallelize optimization," in *Computational Intelligence in Expensive Optimization Problems*, Y. Tenne and C.-K. Goh, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 131–162, ISBN: 978-3-642-10701-6. DOI: 10.1007/978-3-642-10701-6_6.

[29] J. Azimi, A. Jalali, and X. Fern, *Hybrid Batch Bayesian Optimization*. 2012.

[30] E. Contal, D. Buffoni, A. Robicquet, and N. Vayatis, "Parallel gaussian process optimization with upper confidence bound and pure exploration," in *Machine Learning and Knowledge Discovery in Databases*, H. Blockeel, K. Kersting, S. Nijssen, and F. Železný, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 225–240, ISBN: 978-3-642-40988-2.

[31] T. Desautels, A. Krause, and J. W. Burdick, "Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization," *Journal of Machine Learning Research*, vol. 15, no. 119, pp. 4053–4103, 2014.

[32] J. Gonzalez, Z. Dai, P. Hennig, and N. Lawrence, "Batch bayesian optimization via local penalization," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, A. Gretton and C. C. Robert, Eds., ser. Proceedings of Machine Learning Research, vol. 51, Cadiz, Spain: PMLR, May 9, 2016, pp. 648–657.

[33] T. T. Joy, S. Rana, S. Gupta, and S. Venkatesh, "Batch bayesian optimization using multi-scale search," *Knowledge-Based Systems*, vol. 187, p. 104 818, Jan. 2020. DOI: 10.1016/j.knosys.2019.06.026.

[34] D. L. McDowell and S. R. Kalidindi, "The materials innovation ecosystem: A key enabler for the materials genome initiative," *MRS Bulletin*, vol. 41, no. 4, pp. 326–337, 2016. DOI: 10.1557/mrs.2016.61.

[35] A. G. Kusne, H. Yu, C. Wu, H. Zhang, J. Hattrick-Simpers, B. DeCost, S. Sarker, C. Oses, C. Toher, S. Curtarolo, A. V. Davydov, R. Agarwal, L. A. Bendersky, M. Li, A. Mehta, and I. Takeuchi, "On-the-fly closed-loop materials discovery via Bayesian active learning," *Nature Communications*, vol. 11, no. 1, p. 5966, Nov. 2020. DOI: 10.1038/s41467-020-19597-w.

[36] M. Aldeghi, F. Häse, R. J. Hickman, I. Tamblyn, and A. Aspuru-Guzik, *Golem: An algorithm for robust experiment and process optimization*, 2021. arXiv: 2103.03716 [math.OC].

[37] F. Häse, M. Aldeghi, R. J. Hickman, L. M. Roch, M. Christensen, E. Liles, J. E. Hein, and A. Aspuru-Guzik, *Olympus: A benchmarking framework for noisy optimization and experiment planning*, 2021. arXiv: 2010.04153 [stat.ML].

[38] A. E. Gongora, K. L. Snapp, E. Whiting, P. Riley, K. G. Reyes, E. F. Morgan, and K. A. Brown, "Using simulation to accelerate autonomous experimentation: A case study using

mechanics," *iScience*, vol. 24, no. 4, p. 102 262, 2021. DOI: https://doi.org/10.1016/j.isci.2021.102262.

[39] M. M. Noack, K. G. Yager, M. Fukuto, G. S. Doerk, R. Li, and J. A. Sethian, "A Kriging-Based Approach to Autonomous Experimentation with Applications to X-Ray Scattering," *Scientific Reports*, vol. 9, no. 1, p. 11 809, Aug. 2019. DOI: 10.1038/s41598-019-48114-3.

[40] K. I. Schwendner, R. Banerjee, P. C. Collins, C. A. Brice, and H. L. Fraser, "Direct laser deposition of alloys from elemental powder blends," *Scripta Materialia*, vol. 45, no. 10, pp. 1123–1129, 2001. DOI: https://doi.org/10.1016/S1359-6462(01)01107-1.

[41] F. Häse, L. M. Roch, and A. Aspuru-Guzik, "Chimera: Enabling hierarchy based multi-objective optimization for self-driving laboratories," *Chem. Sci.*, vol. 9, no. 39, pp. 7642–7655, 2018, Publisher: The Royal Society of Chemistry. DOI: 10.1039/C8SC02239A.

[42] R. Couperthwaite, R. Arroyave, A. Molkeri, D. Khatamsaz, A. Srivastava, and D. Allaire, *Barefoot framework*, https://github.com/RichardCouperthwaite/BAREFOOT-Framework, 2020.

[43] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, Dec. 1998. DOI: 10.1023/A:1008306431147.

[44] H. J. Kushner, "A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise," *Journal of Basic Engineering*, vol. 86, no. 1, pp. 97–106, Mar. 1964. DOI: 10.1115/1.3653121.

[45] D. D. Cox and S. John, "A statistical method for global optimization," *[Proceedings] 1992 IEEE International Conference on Systems, Man, and Cybernetics*, 1241–1246 vol.2, 1992.

[46] D. D. Cox and S. John, "Sdo: A statistical method for global optimization," in *in Multidisciplinary Design Optimization: State-of-the-Art*, 1997, pp. 315–329.

[47] P. I. Frazier, W. B. Powell, and S. Dayanik, "A knowledge-gradient policy for sequential information collection," *SIAM Journal on Control and Optimization*, vol. 47, no. 5, pp. 2410–2439, Jan. 2008. DOI: 10.1137/070693424.

[48] W. R. THOMPSON, "ON THE LIKELIHOOD THAT ONE UNKNOWN PROBABILITY EXCEEDS ANOTHER IN VIEW OF THE EVIDENCE OF TWO SAMPLES," *Biometrika*, vol. 25, no. 3-4, pp. 285–294, 1933, _eprint: https://academic.oup.com/biomet/article-pdf/25/3-4/285/513725/25-3-4-285.pdf, ISSN: 0006-3444. DOI: 10.1093/biomet/25.3-4.285. [Online]. Available: https://doi.org/10.1093/biomet/25.3-4.285.

[49] M. Hoffman, E. Brochu, and N. de Freitas, "Portfolio Allocation for Bayesian Optimization," in *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, ser. UAI'11, event-place: Barcelona, Spain, Arlington, Virginia, USA: AUAI Press, 2011, pp. 327–336, ISBN: 978-0-9749039-7-2.

[50] M. Emmerich, K. Giannakoglou, and B. Naujoks, "Single- and multiobjective evolutionary optimization assisted by gaussian random field metamodels," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 421–439, 2006. DOI: 10.1109/TEVC.2005.859463.

[51] G. Zhao, R. Arroyave, and X. Qian, *Fast exact computation of expected hypervolume improvement*, 2019. arXiv: 1812.07692 [stat.ML].

[52] R. L. Winkler, "Combining probability distributions from dependent information sources," *Management Science*, vol. 27, no. 4, pp. 479–488, Apr. 1981. DOI: 10.1287/mnsc.27.4.479.

[53] C. Poloni, A. Giurgevich, L. Onesti, and V. Pediroda, "Hybridization of a multi-objective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynamics," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2, pp. 403–420, 2000, ISSN: 0045-7825. DOI: https://doi.org/10.1016/S0045-

7825(99)00394-1. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045782599003941.

[54] A. Talapatra, S. Boluki, T. Duong, X. Qian, E. Dougherty, and R. Arróyave, "Autonomous efficient experiment design for materials discovery with bayesian model averaging," *Physical Review Materials*, vol. 2, no. 11, p. 113 803, 2018, Publisher: APS.

[55] R. Arróyave and D. L. McDowell, "Systems approaches to materials design: Past, present, and future," *Annual Review of Materials Research*, vol. 49, pp. 103–126, 2019, Publisher: Annual Reviews.

[56] A. Aspuru-Guzik and K. Persson, "Materials acceleration platform: Accelerating advanced energy materials discovery by integrating high-throughput methods and artificial intelligence.," *Mission Innovation*, 2018, Publisher: Canadian Institute for Advanced Research.

[57] P. I. Frazier, "A tutorial on bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.

[58] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006, ISBN: 0-262-18253-X.

[59] M. Stein, *Interpolation of Spatial Data*. Springer-Verlag, New York, 1999.

[60] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10, Madison, WI, USA: Omnipress, 2010, pp. 1015–1022, ISBN: 978-1-60558-907-7.

[61] W. B. Powell and I. O. Ryzhov, *Optimal learning*. John Wiley & Sons, 2012, vol. 841.

[62] W. Voigt, "On the relation between the elasticity constants of isotropic bodies," *Ann. Phys. Chem.*, vol. 274, pp. 573–587, 1889.

[63] A. Reuss, "Berechnung der fließgrenze von mischkristallen auf grund der plastizitätsbedingung für einkristalle .," *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift*

für Angewandte Mathematik und Mechanik, vol. 9, no. 1, pp. 49–58, Jan. 1929. DOI: 10.1002/zamm.19290090104.

[64] O. Bouaziz and P. Buessler, "Mechanical behaviour of multiphase materials : An intermediate mixture law without fitting parameter," *Revue de Métallurgie*, vol. 99, no. 1, pp. 71–77, 2002. DOI: 10.1051/metal:2002182.

[65] P. Ludwik, *Elemente der technologischen Mechanik.* 57 p. 20 illus., III fold. diagr. Berlin: J. Springer, 1909.

[66] R. G. Sargent, "Verification and validation of simulation models," *Journal of Simulation*, vol. 7, no. 1, pp. 12–24, 2013, Publisher: Taylor & Francis _eprint: https://doi.org/10.1057/jos.2012.20. DOI: 10.1057/jos.2012.20.

[67] R. Couperthwaite, A. Molkeri, D. Khatamsaz, A. Srivastava, D. Allaire, and R. Arroyave, "Materials design through batch bayesian optimization with multisource information fusion," *JOM*, Oct. 2020. DOI: 10.1007/s11837-020-04396-x.

[68] L. Kaufman and P. Rousseeuw, "Clustering by means of medoids," in *Statistical Data Analysis Based on the L1–Norm and Related Methods*, Elsevier/North Holland, 1987, pp. 405–416.

[69] J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.

[70] T. J. Sargent, *Macroeconomic Theory.* 2nd ed., ser. Economic theory, econometrics, and mathematical economics. Academic Press, 1987, ISBN: 0-12-619751-2.

[71] A. Woronow, "Generating random numbers on a simplex," *Computers & Geosciences*, vol. 19, no. 1, pp. 81–88, Jan. 1993, ISSN: 0098-3004. DOI: 10.1016/0098-3004(93)90045-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0098300493900457.

[72] S. Gündüz, "Effect of chemical composition, martensite volume fraction and tempering on tensile behaviour of dual phase steels," *Materials Letters*, vol. 63, no. 27, pp. 2381–2383, Nov. 2009. DOI: 10.1016/j.matlet.2009.08.015.

[73] V. L. de la Concepción, H. N. Lorusso, and H. G. Svoboda, "Effect of carbon content on microstructure and mechanical properties of dual phase steels," *Procedia Materials Science*, vol. 8, pp. 1047–1056, Jan. 2015. DOI: https://doi.org/10.1016/j.mspro.2015.04.167.

[74] H. Ashrafi, M. Shamanian, R. Emadi, and N. Saeidi, "A novel and simple technique for development of dual phase steels with excellent ductility," *Materials Science and Engineering: A*, vol. 680, pp. 197–202, Jan. 2017. DOI: 10.1016/j.msea.2016.10.098.

[75] J. Sun, T. Jiang, Y. Sun, Y. Wang, and Y. Liu, "A lamellar structured ultrafine grain ferrite-martensite dual-phase steel and its resistance to hydrogen embrittlement," *Journal of Alloys and Compounds*, vol. 698, pp. 390–399, Mar. 2017. DOI: 10.1016/j.jallcom.2016.12.224.

[76] H. Bhadeshia, "Computational design of advanced steels," *Scripta Materialia*, vol. 70, pp. 12–17, Jan. 2014. DOI: 10.1016/j.scriptamat.2013.06.005.

[77] W. Olbricht, N. D. Chatterjee, and K. Miller, "Bayes estimation: A novel approach to derivation of internally consistent thermodynamic data for minerals, their uncertainties, and correlations. part i: Theory," *Physics and Chemistry of Minerals*, vol. 21, no. 1, pp. 36–49, May 1994. DOI: 10.1007/BF00205214.

[78] R. A. Otis and Z.-K. Liu, "High-throughput thermodynamic modeling and uncertainty quantification for ICME," *JOM*, vol. 69, no. 5, pp. 886–892, May 2017. DOI: 10.1007/s11837-017-2318-6.

[79] P. Honarmandi and R. Arroyave, "Using bayesian framework to calibrate a physically based model describing strain-stress behavior of TRIP steels," *Computational Materials Science*,

vol. 129, pp. 66–81, Supplement C Mar. 2017. DOI: 10.1016/j.commatsci.2016.12.015.

[80] T. C. Duong, R. E. Hackenberg, A. Landa, P. Honarmandi, A. Talapatra, H. M. Volz, A. Llobet, A. I. Smith, G. King, S. Bajaj, A. Ruban, L. Vitos, P. E. Turchi, and R. Arróyave, "Revisiting thermodynamics and kinetic diffusivities of uranium–niobium with bayesian uncertainty analysis," *Calphad*, vol. 55, pp. 219–230, Dec. 2016. DOI: 10.1016/j.calphad.2016.09.006.

[81] N. D. Chatterjee, R. Krüger, G. Haller, and W. Olbricht, "The bayesian approach to an internally consistent thermodynamic database: Theory, database, and generation of phase diagrams," *Contributions to Mineralogy and Petrology*, vol. 133, no. 1, pp. 149–168, Oct. 1998. DOI: 10.1007/s004100050444.

[82] M. Stan and B. Reardon, "A bayesian approach to evaluating the uncertainty of thermodynamic data and phase diagrams," *Calphad*, vol. 27, no. 3, pp. 319–323, Sep. 2003. DOI: 10.1016/j.calphad.2003.11.002.

[83] J.-O. Andersson, T. Helander, L. Höglund, P. Shi, and B. Sundman, "Thermo-calc & DICTRA, computational tools for materials science," *Calphad*, vol. 26, no. 2, pp. 273–312, Jun. 2002. DOI: 10.1016/S0364-5916(02)00037-8.

[84] H. Bhadeshia and S. R. Honeycombe, *Steels (Third Edition)*. Oxford: Butterworth-Heinemann, Jan. 2006, ISBN: 978-0-7506-8084-4. DOI: 10.1016/B978-075068084-4/50003-0.

[85] D. Koistinen and R. Marburger, "A general equation prescribing the extent of the austenite-martensite transformation in pure iron-carbon alloys and plain carbon steels," *Acta Metallurgica*, vol. 7, no. 1, pp. 59–60, Jan. 1959. DOI: 10.1016/0001-6160(59)90170-1.

[86] K. Andrews, "Empirical formulae for the calculation of some transformation temperatures," *Journal of the Iron and Steel Institute*, vol. 203, pp. 721–727, 1965.

[87] M. D. McKay, R. J. Beckman, and W. J. Conover, "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979, Publisher: [Taylor & Francis, Ltd., American Statistical Association, American Society for Quality]. DOI: 10.2307/1268522.

[88] A. O'Hagan, "Polynomial chaos : A tutorial and critique from a statistician ' s perspective," 2013.

[89] I. Andrianakis and P. G. Challenor, "The effect of the nugget on gaussian process emulators of computer models," *Computational Statistics & Data Analysis*, vol. 56, no. 12, pp. 4215–4228, Dec. 2012. DOI: 10.1016/j.csda.2012.04.020.

[90] S. Ambikasaran, D. Foreman-Mackey, L. Greengard, D. Hogg, and M. O'Neil, "Fast direct methods for gaussian processes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, pp. 1–1, 2015. DOI: 10.1109/TPAMI.2015.2448083.

[91] E. Galindo-Nava and P. Rivera-Díaz-del-Castillo, "A model for the microstructure behaviour and strength evolution in lath martensite," *Acta Materialia*, vol. 98, pp. 81–93, 2015.

[92] P. Chen, H. Ghassemi-Armaki, S. Kumar, A. Bower, S. Bhat, and S. Sadagopan, "Microscale-calibrated modeling of the deformation response of dual-phase steels," *Acta Materialia*, vol. 65, pp. 133–149, 2014.

[93] A. Srivastava, A. Bower, L. Hector Jr, J. Carsley, L. Zhang, and F. Abu-Farha, "A multiscale approach to modeling formability of dual-phase steels," *Modelling and Simulation in Materials Science and Engineering*, vol. 24, no. 2, p. 025 011, 2016.

[94] A. Srivastava, H. Ghassemi-Armaki, H. Sung, P. Chen, S. Kumar, and A. F. Bower, "Micromechanics of plastic deformation and phase transformation in a three-phase TRIP-assisted advanced high strength steel: Experiments and modeling," *Journal of the Mechanics and Physics of Solids*, vol. 78, pp. 46–69, 2015.

[95] D. Gerbig, A. Srivastava, S. Osovski, L. G. Hector, and A. Bower, "Analysis and design of dual-phase steel microstructure for enhanced ductile fracture resistance," *International Journal of Fracture*, pp. 1–24, 2017.

APPENDIX A

SUPPLEMENTARY DATA FOR PAPER IN CHAPTER 3

A.1 Methods

A.1.1 K-Medoids

The K-Medoids approach [68] is an unsupervised clustering technique similar to the K-Means [69] approach. The main difference between the K-Medoids and K-Means approaches is that the K-Medoids approach selects the data point that is closest to the centroid of the cluster, rather than giving the location of the exact centroid as done in K-Means. This method is used as the clustering technique in the current work since it avoids needing to calculate a new sample set after the clustering step.

The K-Medoids method assumes that there is a set of objects that can be denoted as $X = \{x_1, x_2, \ldots, x_n\}$. Further, the distance between objects x_i and x_j is defined as $d(i, j)$. Each cluster will be defined by a single representative object (medoid). Therefore, a set of representative objects is defined by $Y = \{y_1, y_2, \ldots, y_n\}$ where y_i is a one-zero type object that takes on a value of 1 if the object is selected as a medoid, and zero if not selected.

The second set of one-zero type objects is defined by variables z_{ij} which indicates whether object x_j has been assigned to the cluster with medoid y_i. The k-medoids approach aims to partition the objects in X into clusters to solve the minimization problem;

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} d(i, j) z_{ij}, \tag{A.1}$$

subject to the following constraints,

$$\sum_{i=1}^{n} z_{ij} = 1 \qquad j = 1, 2, \ldots, n, \tag{A.2}$$

$$z_{ij} \leq y_i \qquad i, j = 1, 2, \ldots, n, \tag{A.3}$$

$$\sum_{i=1}^{n} y_i = k \qquad k = \text{no. of clusters}, \tag{A.4}$$

$$y_i, z_{ij} \in (0, 1) \qquad i, j = 1, 2, \ldots, n. \tag{A.5}$$

Where the constraints given in Equations A.2 and A.5 ensure that each object can only exist in a single cluster, while the constraint in Equation A.3 ensures that objects can only be assigned to a medoid if that medoid exists. Finally Equation A.4 ensures that there are the correct number of clusters.

A.1.2 Micromechanical models

The Reification-Fusion approach works by fusing low-order models to create an approximation to a truth model. In the current work, three low-order models were used. These low-order models were an isostrain model [62] where the strain is assumed to be homogeneous through the two phases in the material, an isostress model [63] where the stress was assumed to be constant in the two phases, and an isowork model [64] where the work is constant in both phases.

The models all included isotropic hardening that followed Ludwik's Power Law [65],

$$\tau^p = \tau_0^p + K^p (\epsilon_{pl}^p)^{n^p}, \tag{A.6}$$

where τ^p is the flow stress, τ_0^p is the yield stress, K^p is the strengthening coefficient, n^p is the strain hardening exponent and ϵ_{pl}^p is the plastic strain.

For the two phases, it was assumed that the martensite phase strength was only affected by the carbon content of the martensite, while the ferrite phase strength was only affected by the manganese and silicon content of the ferrite. This leads to the yield strength of the martensite

being defined by,

$$\tau_o = 400 + 1000\ (X_C^{mart})^{1/3}, \tag{A.7}$$

and the ferrite yield strength is defined by,

$$\tau_o = 200 + 213(X_{Mn}^{ferr})^{0.5} + 732(X_{Si}^{ferr})^{0.5} \tag{A.8}$$

where the solid solution coefficients were estimated using the method described by Galindo-Nava et al. [91]. The parameters of the constitutive equation were chosen to ensure that the material consisted of a soft phase (ferrite) and a harder (martensite) phase [92]–[95]. The parameters are shown in Table A.1.

Table A.1: Parameterization of the Ludwik power law for the constituent phases of the dual-phase microstructure.

Constituent Phase, p	K^p **[MPa]**	n^p
Soft (ferrite)	2200	0.5
Hard (martensite)	450	0.06

The truth model in the current work was a finite-element calculation based on a representative volume element (RVE) representation of the dual-phase microstructure. For the dual-phase microstructure calculations in the current work, the procedure for generating a 3D representative element model is explained in detail in [13], [14], [95].

As with the reduced-order models, the hardening of the phases in the finite-element calculations was modeled using Ludwik's Power Law A.6, and the phase properties were calculated using the strength models for martensite A.7 and ferrite A.8. The finite-element calculations were conducted in ABAQUS. We note that to speed up the computation and to carry out a statistically representative comparison of the different methods/scenarios used, the full RVE model was not used and instead

was replaced by a surrogate Gaussian Process model fit to 1,400 data points obtained from the RVE model. Despite using a surrogate model, the cost and time of the calculations of the truth model were estimated as that of the full RVE model.

A.1.3 Thermodynamic Calculations

The composition of the phases in the mechanical model was determined using Thermo-Calc™. To ensure the seamless operation of the framework on the super-computing resources used for the calculations, a surrogate model of the Thermo-Calc™ results was used instead of direct Thermo-Calc™ evaluations.

To build this surrogate model data was obtained by uniform sampling within the design space shown in Table A.2. The output of Thermo-Calc™ was limited to the austenite and ferrite phases for each temperature and composition combination. It was assumed that the material was quenched very quickly from the inter-critical annealing temperature to room temperature and so the composition of the martensite formed was assumed to be the same as the austenite phase calculated from Thermo-Calc™.

Table A.2: Composition and temperature range of the design space in the current work

C (wt.%)	**Mn** (wt.%)	**Si** (wt.%)	**Fe** (wt.%)	T_{IA} (°C)
0.0-1.0	0.0-3.0	0.0-2.0	bal.	650-850

A Gaussian Process surrogate model was built using the data from Thermo-Calc™. Since there are only two phases, the surrogate was built to only predict the composition and volume fraction of the martensite phase. The composition of the ferrite phase is calculated using a mass balance under the assumption that there are no material losses when heat treating the material.

A.1.4 Evaluation of Effect of Iteration Limit in Iteration Controlled Optimization

The iteration limit imposed in the iteration controlled optimization processes was chosen arbitrarily, however, we note that there is likely to be an optimum value for this framework parameter.

However, as with the Gaussian Process Hyper-parameters, it will likely be impossible to properly assess this value before the optimization is started. As a result, we propose that a reasonable approach to setting this kind of parameter would be to assess how many of the most expensive experiments can be conducted within the given project and then choose an iteration limit that ensures that number is not exceeded in the course of the optimization process.

We assume that the choice of the iteration limit will affect the results to some degree. Therefore, to assess the effect of the arbitrarily chosen limit of 25 iterations before calling the RVE model, the Cost Constrained - Iteration Controlled (CC-IC) approach was tested using iteration limits of 10 and 50. In all three sets of calculations (iteration limit of 10, 25, and 50) the same initial conditions (2 random points in the design space) were used and the maximum value of the RVE found was recorded. Calculations with 15 sets of initial conditions were conducted and the results were averaged.

A.1.5 Sequential Batch Optimization

To provide a better comparison between the effect of using batch optimization over sequential Bayesian optimization a standard Bayesian Optimization was conducted. In this approach, the only model used was the RVE surrogate model, and the acquisition function used to evaluate the next best point was Knowledge Gradient. As with the Batch Optimization approach, the RVE was initialized with 2 random points in the design space and 20 different initializations were used to find the average performance of the sequential optimization approach. The initial points used in the sequential approach were the same as those used in the batch approach.

As with the batch optimization approach, the initial sample size for evaluating the Knowledge Gradient was 500 samples, obtained by Latin hypercube sampling of the design space. On each iteration, this amount was incremented by 1 to produce a finer search of the design space over time. As with the batch approach, the cost of this approach was estimated using the computational clock time for the RVE calculation as well as the computational clock time required for calculating the Knowledge Gradient at each iteration. The sequential optimization approach was limited to 200 iterations to match the number of iterations used in the batch approach.

A.2 Results

A.2.1 Comparison of different iteration limits

Figure A.1 to A.3 show the results from testing the three different iteration limits for all batch sizes used in the current work. Several observations can be made. Firstly, Figure A.1 shows that setting the limit on the RVE calls lower (10 iterations) the framework optimizes the normalized strain hardening rate slightly faster than when the iteration limit is 25, and quite significantly faster than when the iteration limit is 50. This indicates that in this particular application, a shorter iteration limit is a better choice.

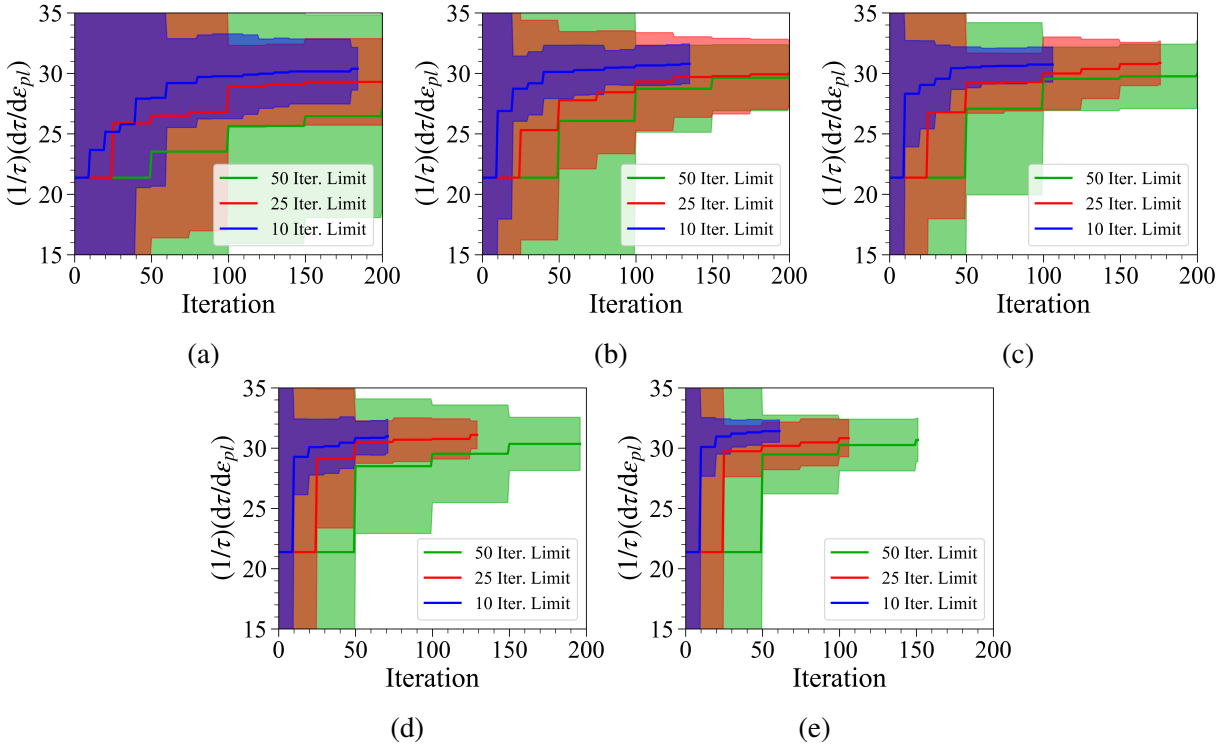


Figure A.1: Maximum RVE Result found as a function of iterations of the optimization process for different iteration limits before calling the RVE function. Plots show the results for Batch Sizes a) 1, b) 2, c) 3, d) 5, and e) 7. The shaded regions reflect the 95% confidence interval of the optimization process for calculations done with 15 different initial conditions.

However, when considering the cost of the process, Figure A.2, the start to depend quite signif-

icantly on the batch size. This can be seen by comparing the results of the Batch Size 1 calculations with the Batch Size 7 calculations. In the Batch Size 1 calculations, the shorter iteration limit results in a much more significant cost than the longer iteration limits, while with the results from Batch Size 7, this increased cost is not as evident. Since there is an overall cost constraint in the optimization approach used for these results, the Batch Size 7 results are possibly being affected by this and as a result, it is not possible to see the whole picture. And this is reflected in the results showing the time of the optimization process.

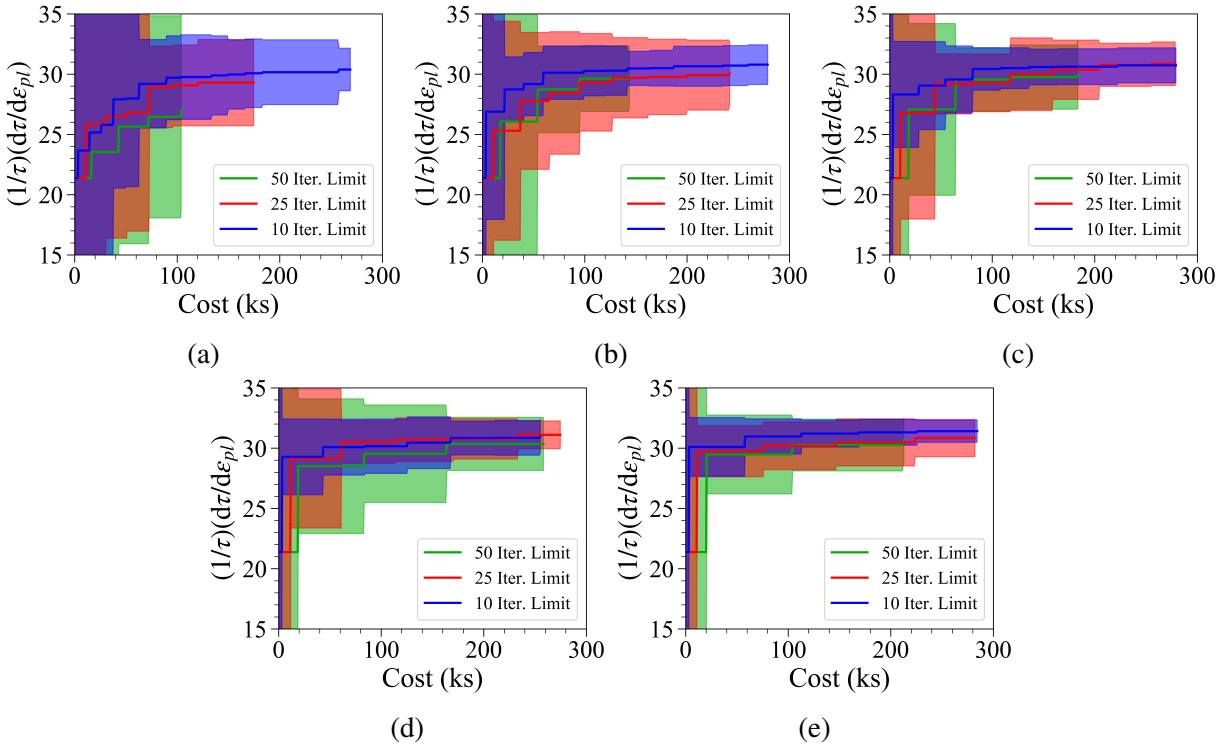


Figure A.2: Maximum RVE Result found as a function of the cost of the optimization process for different iteration limits before calling the RVE function. Plots show the results for Batch Sizes a) 1, b) 2, c) 3, d) 5, and e) 7. The shaded regions reflect the 95% confidence interval of the optimization process for calculations done with 15 different initial conditions.

Figure A.3 shows the maximum normalized strain hardening rate found against the time taken for the optimization. Again we can see that for Batch Size 1, the shorter iteration limit causes the optimization process to take a significantly longer amount of time, while the longer iteration

limit ends much faster. However, as the Batch Size is increased, the situation is reversed and the optimization with the shorter iteration limit reaches an optimum value much quicker than the optimization with a longer iteration limit.

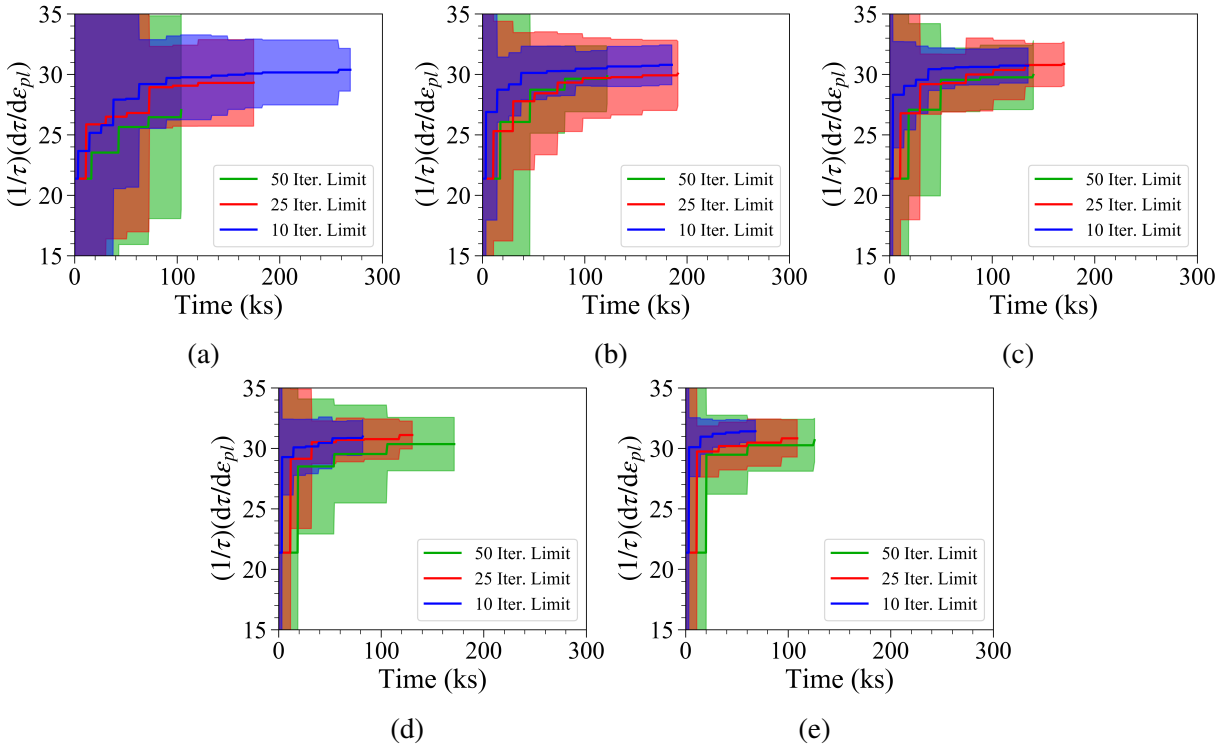


Figure A.3: Maximum RVE Result found as a function of the time of the optimization process for different iteration limits before calling the RVE function. Plots show the results for Batch Sizes a) 1, b) 2, c) 3, d) 5, and e) 7. The shaded regions reflect the 95% confidence interval of the optimization process for calculations done with 15 different initial conditions.

The plots from batch sizes 1, 3, 7 are compared on a grid in Figure A.4. This clearly shows how as the batch size is increasing down the rows, the number of iterations is decreasing and with it the total time. What can also be seen is that the cost of the process is increasing as the batch size increases. The decreases cost of the 50 Iteration Limit approach with batch size 1 is due to the hard limit imposed on the number of iterations for the process. In future work, this limit will be removed and only the final cost will limit the process.

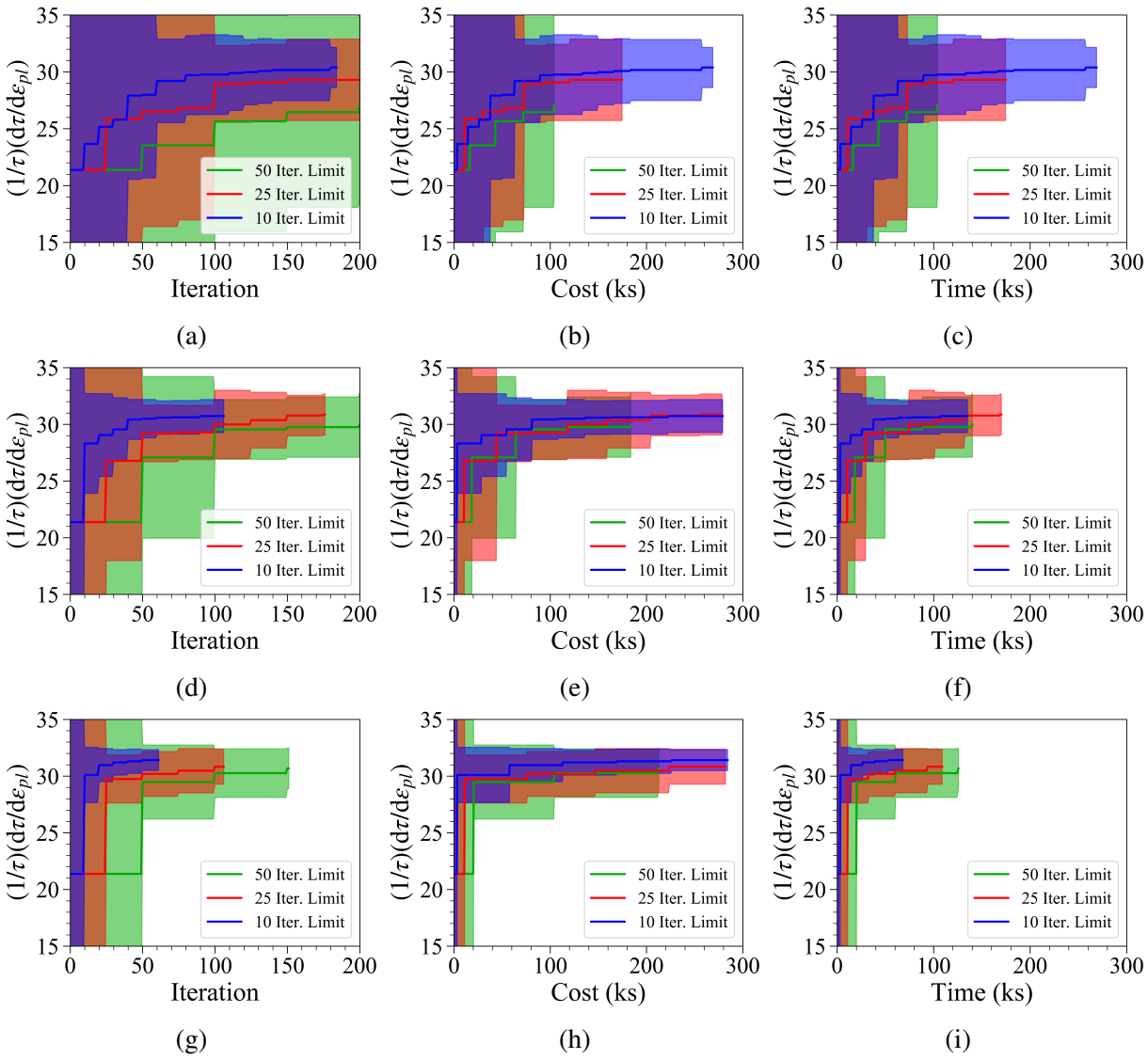


Figure A.4: Direct comparison of Batch Size 1, 3 and 7 showing the iterations in [(a),(d),(g)], cost in [(b),(e),(h)] and the time in [(c),(f),(i)].

All three of these results together show that there is an inter-dependence on the iteration limit and the batch size in the current framework. We postulate that this is directly related to the amount of information that is gained about the system at each iteration. Since the optimization with batch size 1 only adds 1 data point at each iteration, there is a cost and time benefit to waiting longer between calling the RVE model to ensure that the framework builds up sufficient information. However, when the batch size is 7, a significantly larger amount of information is added on each

iteration and so the framework builds sufficient data fast enough to benefit from a shorter iteration limit.

A.2.2 Comparison of Sequential and Batch Optimization Results

These results extend the results shown in the main text by showing the 95% confidence interval for the sequential Bayesian optimization results. As can be observed in Figures A.5 - A.6, the 95% confidence interval is not as smooth as that seen in the batch optimization approach. This would indicate that obtaining the additional information from the reduced-order models helps to maximize the normalized strain hardening rate regardless of the initial conditions provided.

The first results show the comparison between the results from the sequential BO optimization and batch sizes 1 and 7 with respect to the number of iterations. These results show that for the same number of iterations of the optimization approach, the sequential optimization performs better. However, this approach has 200 RVE calculations where the largest number of RVE calculations in the BBO approach is for the No Cost Constraint approach which has under 60 RVE calculations.

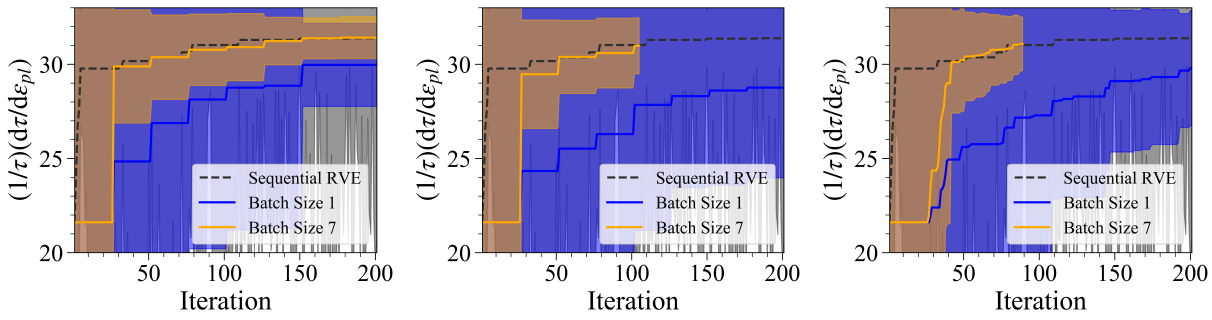


Figure A.5: Maximum normalized strain hardening rate achieved from RVE calculations as a function of the number of iterations of the optimization process for (a) No Cost Constraint, (b) Cost Constrained - Iteration Controlled and (c) Cost Constrained - Cost Controlled optimization cases. The shaded regions of the plots indicate the 95% confidence interval calculated from the results of 20 optimization calculations for each batch size

Having noted that the sequential optimization has a far greater number of RVE calculations, the results in Figure A.6 confirm that the number of RVE calculations has a significant effect on the cost. As can be seen, the sequential optimization, while achieving a higher value for the quantity of interest, costs significantly more than the BBO approach in the current work.

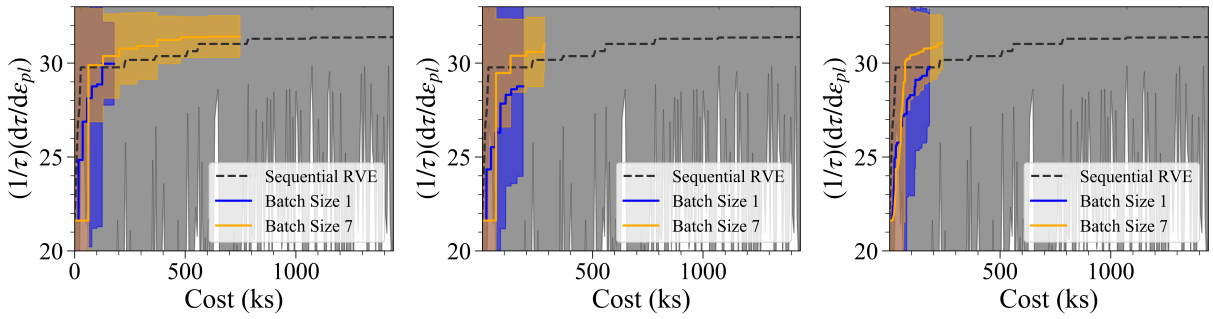


Figure A.6: Maximum normalized strain hardening rate achieved from RVE calculations as a function of the total cost of the optimization process for (a) No Cost Constraint, (b) Cost Constrained - Iteration Controlled and (c) Cost Constrained - Cost Controlled optimization cases. The shaded regions of the plots indicate the 95% confidence interval calculated from the results of 20 optimization calculations for each batch size

To compare the times taken for the optimizations, it was necessary to evaluate the plots with a log scale. This is the most significant result of the current work since it shows that the BBO approach used in the current work can decrease the optimization time by at least an order of magnitude. While it is noted that this is one particular application of this framework, these results hold significant promise for the accelerated design of materials.

As can be observed in Figures A.5 to A.7, another of the advantages of the current framework is the reduced sensitivity of the framework to the initial conditions. The sequential approach has much greater variance in the results than the current framework results. This is another important result of the current work. This shows that in addition to reducing the cost and time of the optimization, there is less uncertainty related to the predictions from the framework when compared to the predictions from a sequential Bayesian optimization.

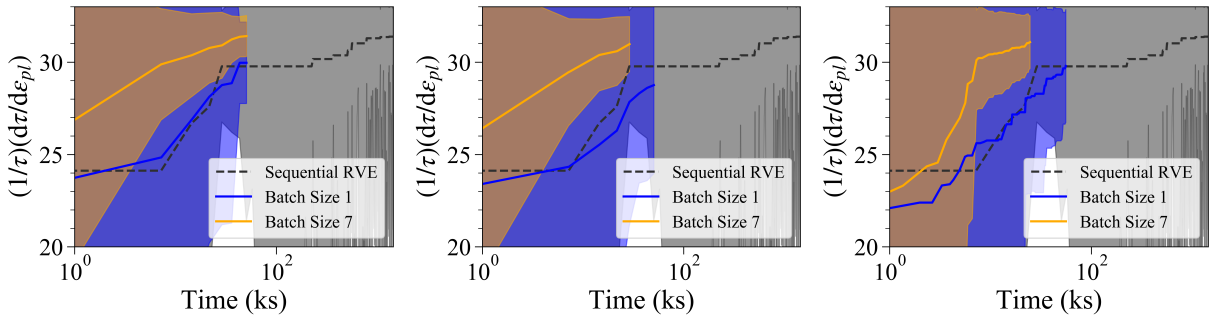


Figure A.7: Maximum normalized strain hardening rate achieved from RVE calculations as a function of the total time of the optimization process for (a) No Cost Constraint, (b) Cost Constrained - Iteration Controlled and (c) Cost Constrained - Cost Controlled optimization cases. The shaded regions of the plots indicate the 95% confidence interval calculated from the results of 20 optimization calculations for each batch size

APPENDIX B

SUPPLEMENTARY MATERIAL FOR PAPER IN CHAPTER 4

B.1 Reduced-Order Model Accuracy Test

In order to test the effect of the accuracy of the reduced-order models on the efficiency of the optimization we considered a test function based on the eggholder function,

$$f(x) = |x_1| sin\,(5x_1) + |x_2| sin\,(6x_2) \tag{B.1}$$

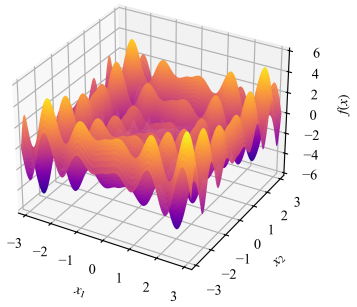


Figure B.1: Plot of the sample test function used in the accuracy test

which is defined for $x_1, x_2 \in [-\pi, \pi]$. The model is shown in Figure B.1. As can be observed, the function has many local optima, however, the global maximum is $f(x) = 5.719076$ at $x = [2.841, -2.889]$.

Using a Fourier series expansion, we defined three models with increasing accuracy using the 2, 4, and 6 terms of the Fourier expansion and modifying the factors slightly to produce different models. In this way, we created three sets of three models each that were progressively more accurate predictors of the Truth Model. All 9 of these models are shown in Figure B.2.

Using a fixed set of parameters for the BAREFOOT Framework, we ran each set of models with

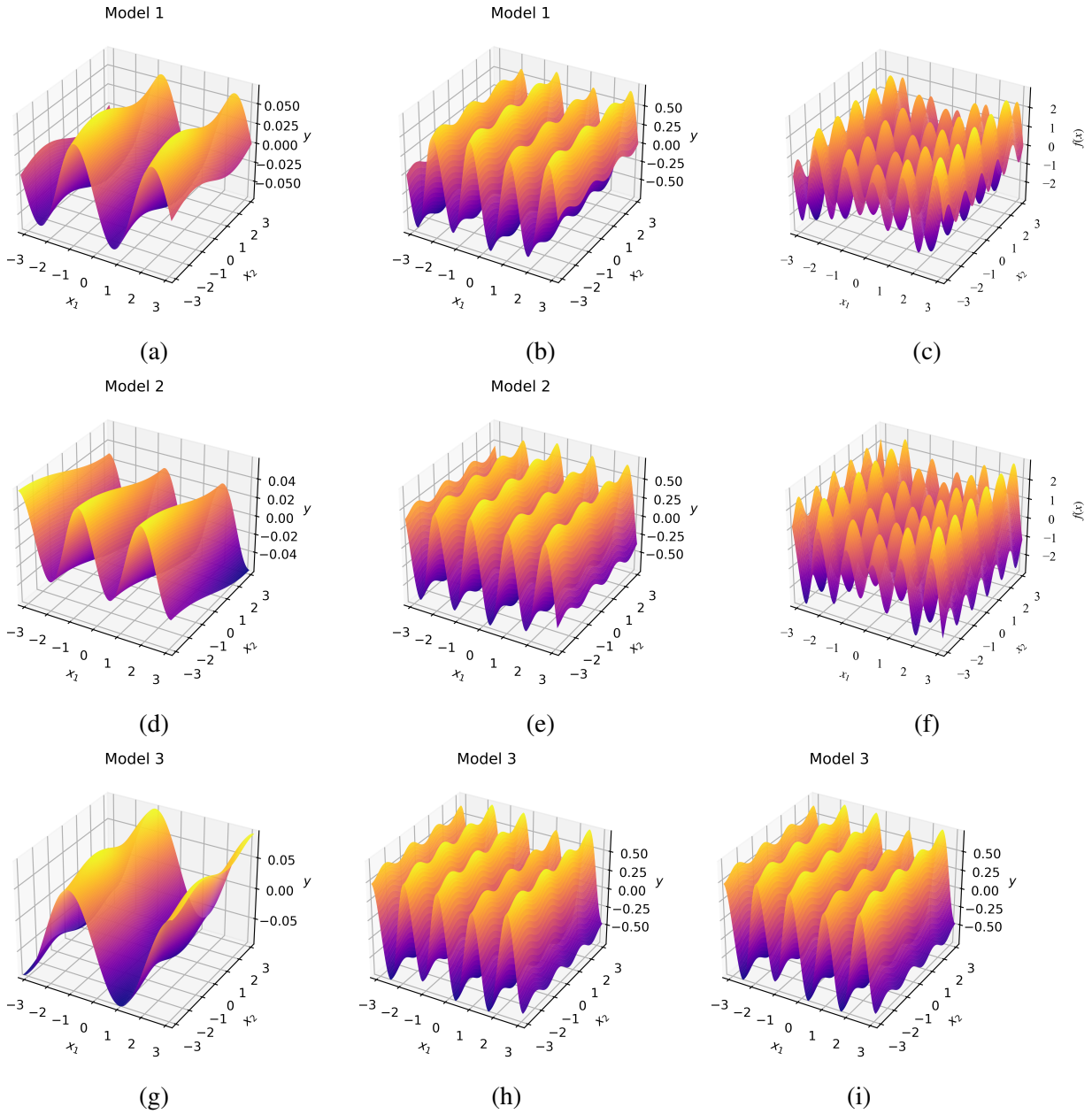


Figure B.2: Representation and comparison of the three reduced order model sets used in the test of the model accuracy on framework performance. The rows are for each of the models (a,b,c) Model 1, (d,e,f) Model 2, and (g,h,i) Model 3. The columns correspond with the degree of Fourier expansion, (a,d,g) first expansion or 2 terms, (b,e,h) second expansion or 4 terms, (c,f,i) third expansion or 6 terms.

10 sets of initial values. The initial data consisted of a single point evaluated in the design space.

These optimizations were run for 50 iterations, and the Ground Truth model was called every five

iterations. The results from this test are shown in Figure B.3. These indicate that the more accurate the reduced-order models are, the better the Framework works. However, these results are not very conclusive considering the confidence intervals for the results and should be tested further.

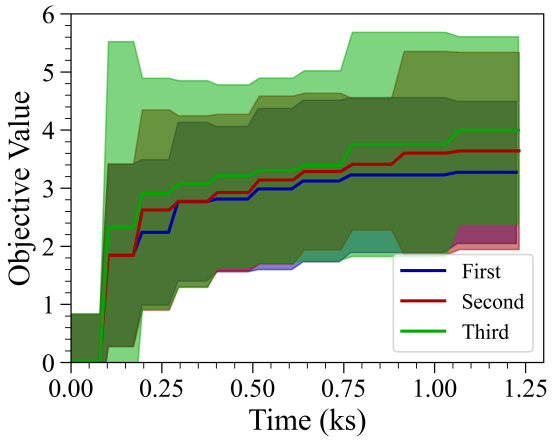
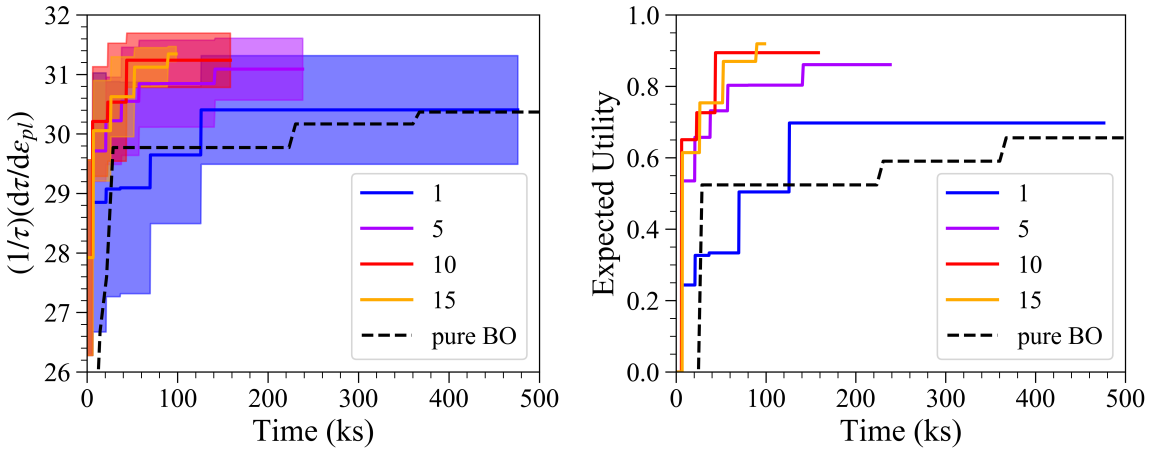


Figure B.3: Results from running the test function with reduced-order models of different levels of accuracy. The uncertainty bounds show two standard deviations of the optimum value found from 10 optimizations. The labels first, second, and third refer to the level of Fourier expansion terms used where first corresponds with the first two terms, Second the first four and Third the first 6.

B.2 Mechanical Model Test Case Results

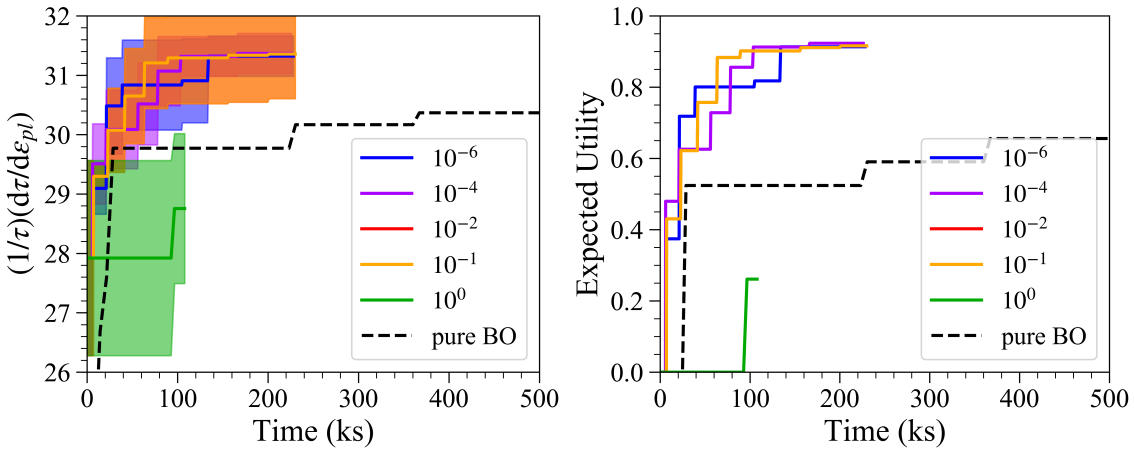
These results show the full results for every test done on the hyperparameters for the Mechanical Model Test Case. The plots show both the mean and uncertainty from 5 iterations and the Expected Utility values. In Figure B.4 the results show how increasing the batch size decrease the time for the optimization. However, we note that there does not appear to be much difference between the final results of the 10 and 15 Batch Size cases. This could indicate that there is an upper limit on the batch size for each optimization that corresponds with the most efficient optimization.

The results in Figure B.5 demonstrate that as long as the reduced-order models are cheaper than the Ground Truth model, the cost of the reduced-order models does not have a significant effect on the final results.



(a) (b)

Figure B.4: Plots showing the effect of batch size on the optimization of the normalized strain hardening rate in the mechanical model case study

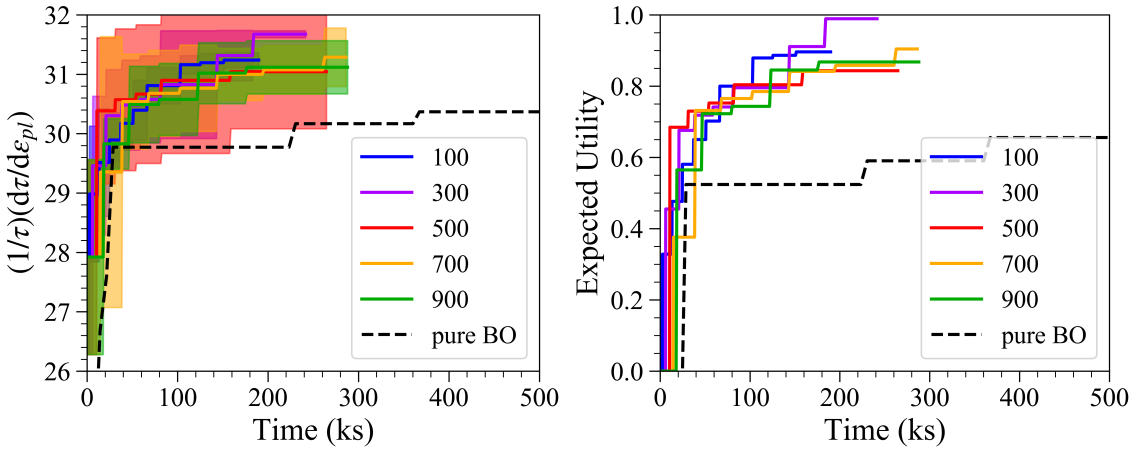


(a) (b)

Figure B.5: Plots showing the effect of reduced order model costs on the optimization of the normalized strain hardening rate in the mechanical model case study

In Figure B.6 we show the results from tests with varying hyperparameter counts, where the hyperparameter count is the number of hyperparameter sets used in the Batch Optimization approach. These results indicate that a moderate number of 100-300 hyperparameter sets is something of an optimal value. However, care must be taken with these results since they may apply to this specific system of equations, and there could also be other interactions with other framework parameters

that also have an effect.



(a) (b)

Figure B.6: Plots showing the effect of the number of hyperparameter sets on the optimization of the normalized strain hardening rate in the mechanical model case study

The results from varying the amount of initial data show an interesting effect (Figure B.7, which has been noted in other works. This result indicates that the number of initial data does not significantly affect the final result. And it even appears like having less initial data helps the optimization to work better. What can be observed as a significant effect is that the initial mean changes quite significantly as the number of initial data increases.

When observing the results from varying the lower bound of the hyperparameters for the optimization (Figure B.8), we note that there was no real effect in the optimizer's performance until we decreased the lower limit to 0.0001. This indicates that keeping the lower bound of the hyperparameters at around 0.0001 would be ideal.

The number of samples at which the acquisition functions are evaluated is another important parameter, and the results in Figure B.9 demonstrate that there appears to be an optimum in this parameter at around 30-50 samples. Much like the hyperparameter count results presented before this, it is suspected that this framework parameter might have significant interaction with other

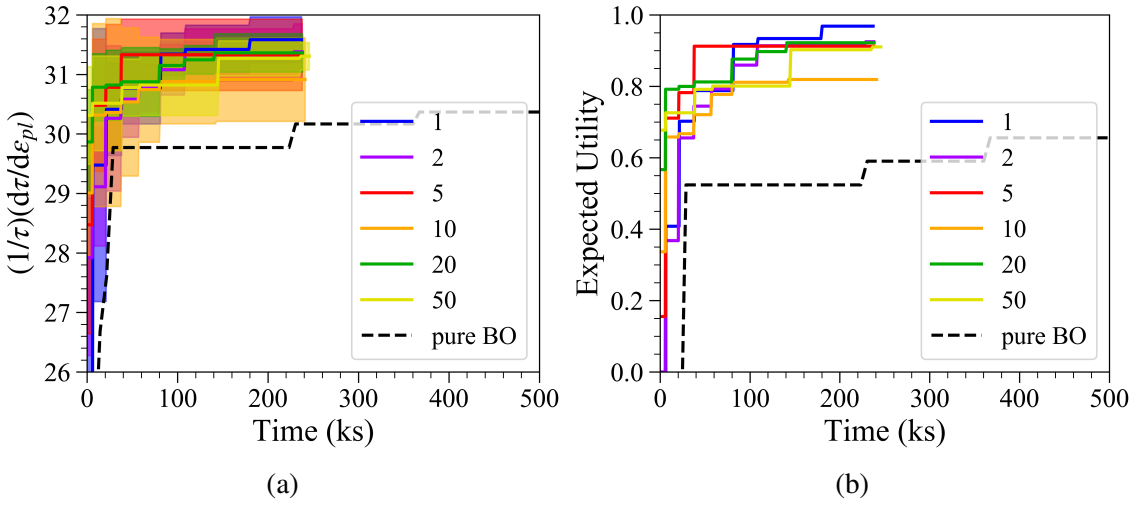


Figure B.7: Plots showing the effect of amount of initial data on the optimization of the normalized strain hardening rate in the mechanical model case study

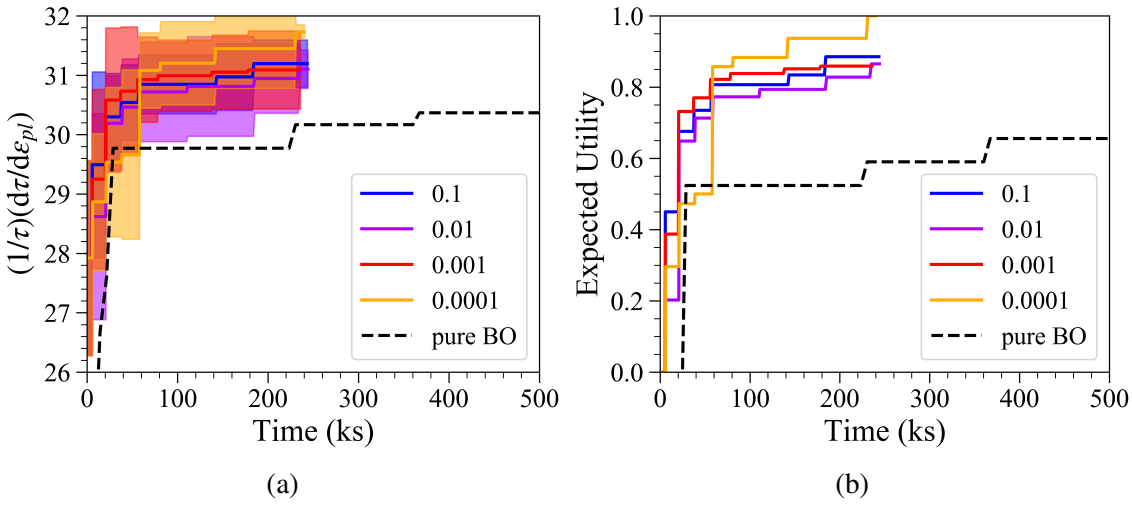
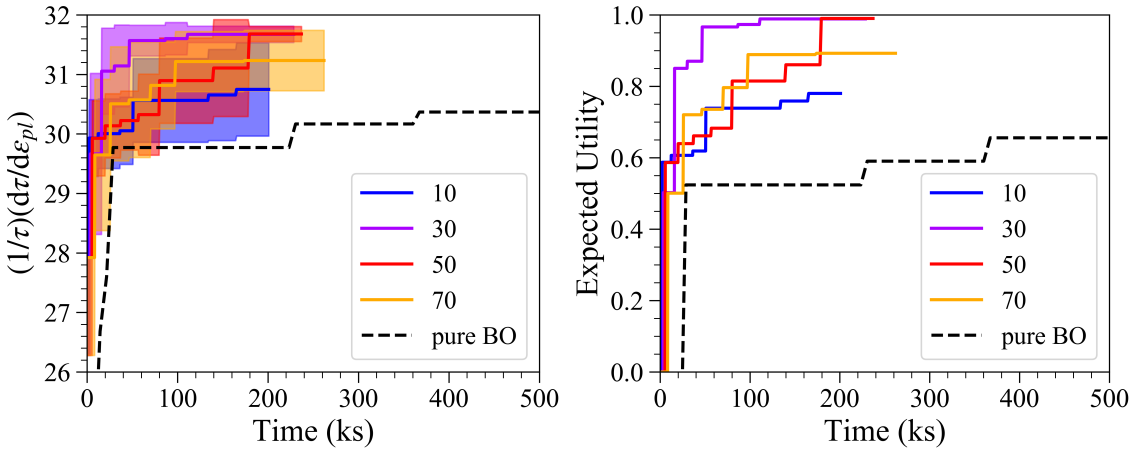


Figure B.8: Plots showing the effect of the hyperparameter lower bound on the optimization of the normalized strain hardening rate in the mechanical model case study

parameters. As such, while these results indicate an excellent initial value to use, it might be necessary to account for other parameters.

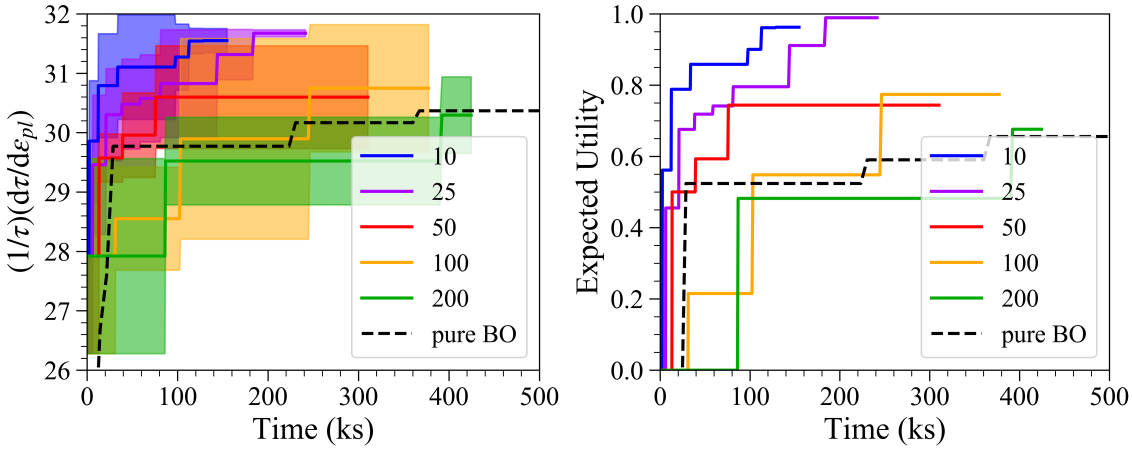
The results from tests varying the iteration limit for the Ground Truth queries (Figure B.10) indicate that having a lower iteration limit has a significant and positive effect on the Framework's performance. However, it also has a considerable impact on how quickly the budget for the opti-



(a) (b)

Figure B.9: Plots showing the effect of sample count on the optimization of the normalized strain hardening rate in the mechanical model case study

mization is consumed.

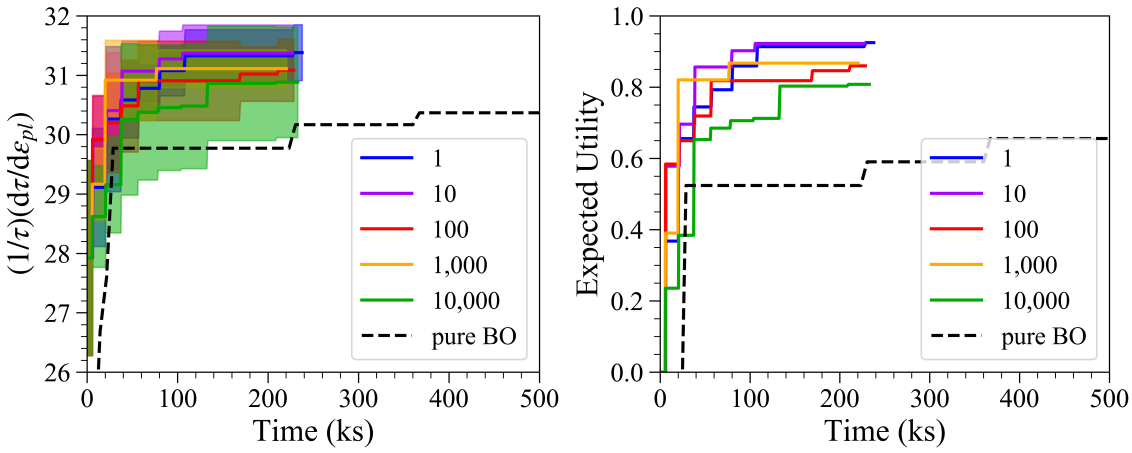


(a) (b)

Figure B.10: Plots showing the effect of the iteration limit for querying the Truth Model on the optimization of the normalized strain hardening rate in the mechanical model case study

The results for varying the upper bound of the hyperparameters B.11 show that there isn't any benefit in increasing the upper bound much beyond the length of the design space. Since the input

space is in a unit hypercube, it isn't necessary to increase the upper bound of the hyperparameters above 1.



(a) (b)

Figure B.11: Plots showing the effect of the upper bound of the hyperparameters on the optimization of the normalized strain hardening rate in the mechanical model case study

B.3 Three Hump Camel Test Case Results

For the Three Hump Camel Test Case, five reduced-order models were created. The equations that define these models are,

$$
\begin{aligned}
f_{R1}(x_1, x_2) &= -1.05(x_1 - 0.5)^4 - \frac{x_1^6}{6} - x_1 x_2 - x_2^2 \\
f_{R2}(x_1, x_2) &= -2(x_1 + 0.5)^2 - \frac{x_1^6}{6} - x_1 x_2 - x_2^2 \\
f_{R3}(x_1, x_2) &= -2\left(\frac{x_1}{2}\right)^2 + 1.05x_1^4 - x_1 x_2 - x_2^2 \\
f_{R4}(x_1, x_2) &= -2(2x_1)^2 + 1.05x_1^4 - \frac{x_1^6}{6} - x_2^2 \\
f_{R5}(x_1, x_2) &= -2x_2^2 + 1.05x_1^4 - \frac{x_1^6}{6} - x_1 x_2
\end{aligned}
\tag{B.2}
$$

The results in this section only show the Expected Utility. As demonstrated in the previous

section, using the Expected utility allows for easier visualization of the results. Starting with Figure B.12(a), we can observe that the effect of batch size in this test function is similar to the effect seen when considering the mechanical model test. However, what we also observe in this case is that the final value found is very similar for all batch sizes. The only significant change is in how quickly the Framework achieves that value. When considering the results in Figure B.12(b), we observed that the Squared Exponential function does not perform as well as the Matern covariance functions with the Mater($\nu = 3/4$) covariance function performing better. Despite this, when only varying the covariance function, none of the optimizations perform as well as the conventional Bayesian Optimization approach. This is at least consistent with all the tests in this Test Case, showing that either the reduced-order functions used significantly affect how well the Framework operates or that the Framework is not helpful in all applications.

Figure B.12(c) shows that a more coarse grid for calculating the fused mean and variance reduces the Framework's performance slightly. While the results in Figure B.12(d) demonstrate that a moderate number of hyperparameters allows the Framework to perform better.

Figure B.13(a) demonstrates the result observed earlier that certain qualities of the reduced-order models have a detrimental effect on the optimization. Although this is only one possible interpretation for why there is such a decrease in the performance when increasing the number of reduced-order models to 4 but a reversion to almost the same performance as before when increasing the number of models to 5, this result needs further investigation. From the results seen in the mechanical model test, it was not expected to see such a change in the performance of the Framework when changing the relative cost of the reduced-order models (Figure B.13(b)). These results could indicate that the stochastic nature of the Framework is having more of an effect than previously thought. And this result, in particular, triggered the modification of the approach by using the fused model itself when determining the points to evaluate from the Ground Truth.

The effect of the sample count is not as pronounced as it was in the Mechanical Model test case (Figure B.13(c), however, it is still observable that lower values for this parameter increase the performance of the optimization. While in Figure B.13(d), we can observe that the cost of

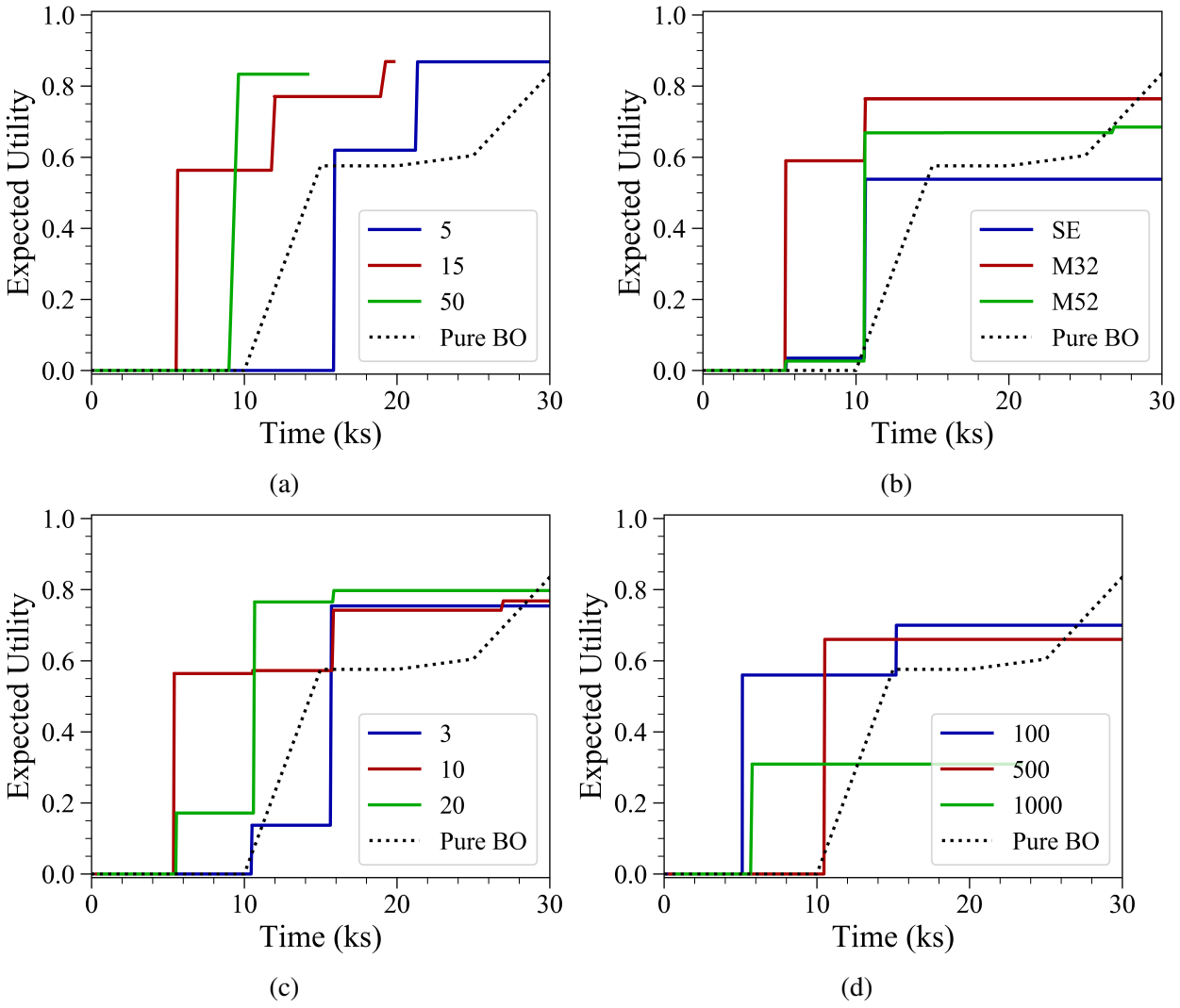


Figure B.12: Plots showing the effect of a) Batch Size, b) Covariance Function, c) Number of fused point, and d) the number of hyperparameter sets on the optimization performance in the Three- hump camel case study

the ground truth model has a significant effect on the performance of the Framework. This is an expected result and confirms that the Framework operates expectedly in this regard. As seen previously, the iteration limit for calling the Ground Truth has a significant effect on the performance of the optimization (Figure B.13(e)). However, in these results, while lower iteration limits attain higher values, the larger iteration limits perform at least as well for much of the time. With the net result that if the optimization is cut off at 20,000s, the largest iteration limit would have the best

performance by a large margin. It appears that it is not a simple matter of saying that the iteration limit should take a particular value. It might be related to the earlier research question of whether the nature of the reduced-order models affects the Framework's performance.

Following on from the discussion of Figure B.13, there is further evidence that the iteration limit for the Ground Truth queries is coupled to the models. As observed in Figure B.14 we can see that the lower iteration limits perform better as the batch size increases. This would indicate that the iteration limit for calling the Ground Truth is related to the amount of information that we can obtain from the reduced-order models before augmenting that information with new Ground Truth Data. Taking all the results in Figure B.14 shows a contrary result to that seen previously for this Three Hump Camel test case, namely that the lower iteration limit performs better, regardless of batch size. However, the strength of this effect increases with increasing batch size.

The results in Figure B.15 show a fascinating but mostly predictable result. When comparing the Squared Exponential and Mater($\nu = 3/2$) covariance functions, there is a clear benefit for using smaller numbers of fused points (a more granular sampling grid) when using the smoother Squared Exponential covariance function. This indicates that having a finer sampling grid benefits a less smooth covariance function. However, contrary to previous results, the Squared Exponential Function performs well with all tests in these results. At the same time, the Matern covariance functions struggle when the number of fused points is too low. As a result, it would be suggested to keep the number of fused points around 10 when using the BAREFOOT Framework. This could cause problems for high-dimensional problems, so this assertion will need to be re-evaluated when testing a large input space.

The results from testing the effect of hyperparameter count and batch size (Figure B.16) show a positive correlation between the batch size and the hyperparameter count. However, it does appear that the effect observed previously, namely that more moderate hyperparameter counts work better than large counts, still applies.

The results from the test looking at the sample count and the hyperparameter count there is

quite a lot of overlap in the results (Figure B.17), but there are some observations that we can make. One interesting observation is that the larger sample count performs uniformly better than the lower sample count. This is contrary to the observation made when considering the results from only varying the sample count. There is no noticeable trend for the hyperparameter count, with both large and small hyperparameter counts performing well. Except for when the sample count is at 10, the hyperparameter counts of 100 and 1000 perform better than when the count is 500.

When observing the results for comparing the Ground Truth Model cost and the reduced order model cost, we can see the expected trend of the Ground Truth Model cost very quickly (Figure B.18). However, there does not appear to be any apparent effect from the reduced-order model cost.

Again, testing the Ground Truth Model cost and the Truth Model iteration limit show an expected result (Figure B.19). However, the effect of the iteration limit is not quite as evident in this case. This, again could be an effect of other interactions within the Framework, or it could be related to the stochastic nature of the process.

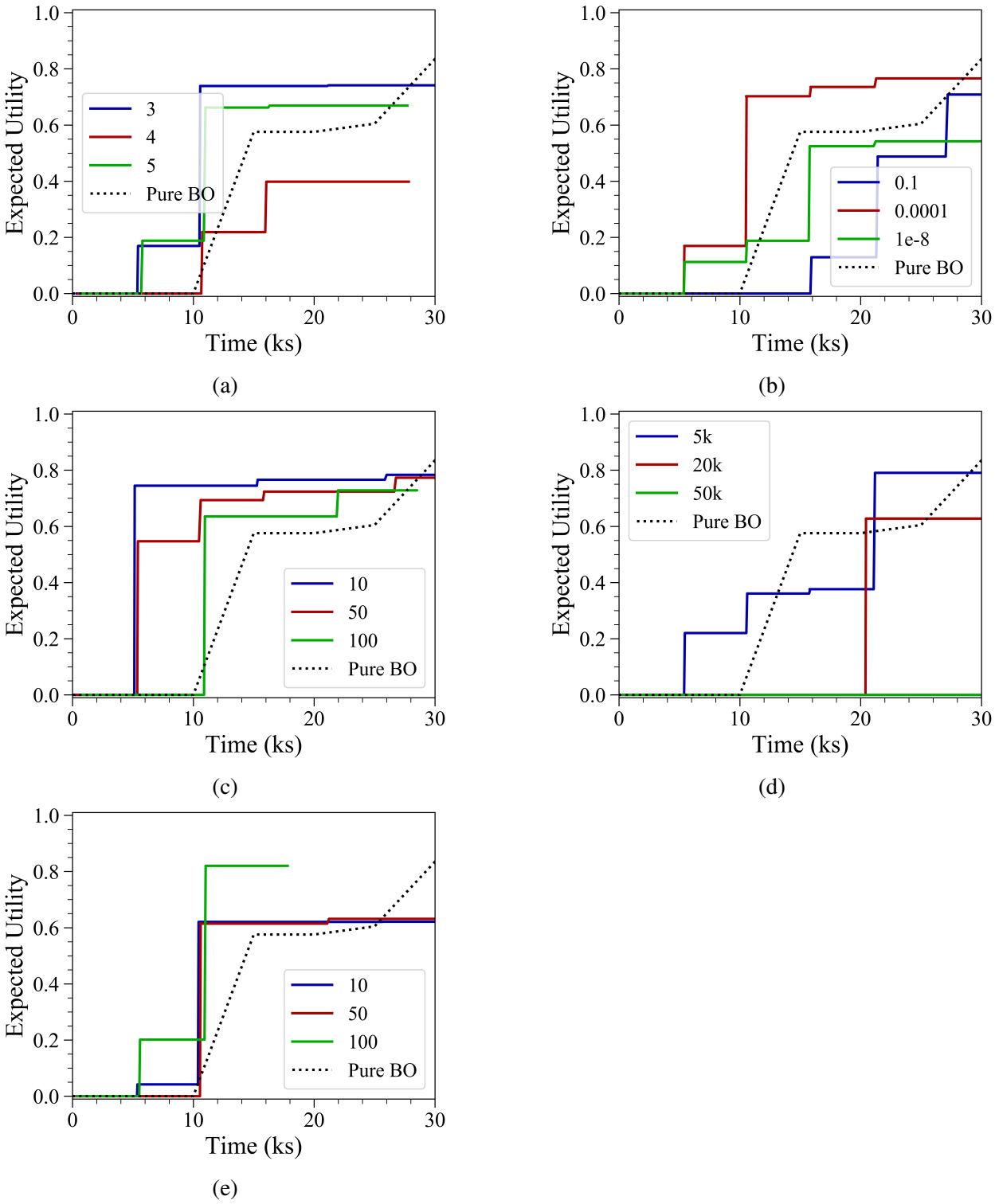


Figure B.13: Plots showing the effect of a) number of Reduced-Order Models, b) reduced-order model cost, c) sample count, d) the truth model cost and e) the truth model iteration limit on the optimization performance in the Three- hump camel case study

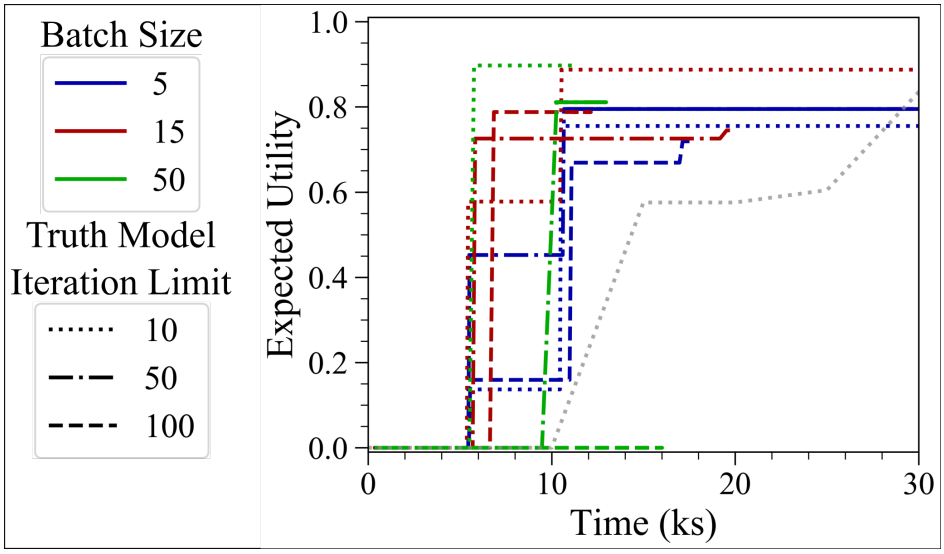


Figure B.14: Two-way parameter test for the Batch Size and Truth Model Iteration limit in the Three-Hump Camel Case Study

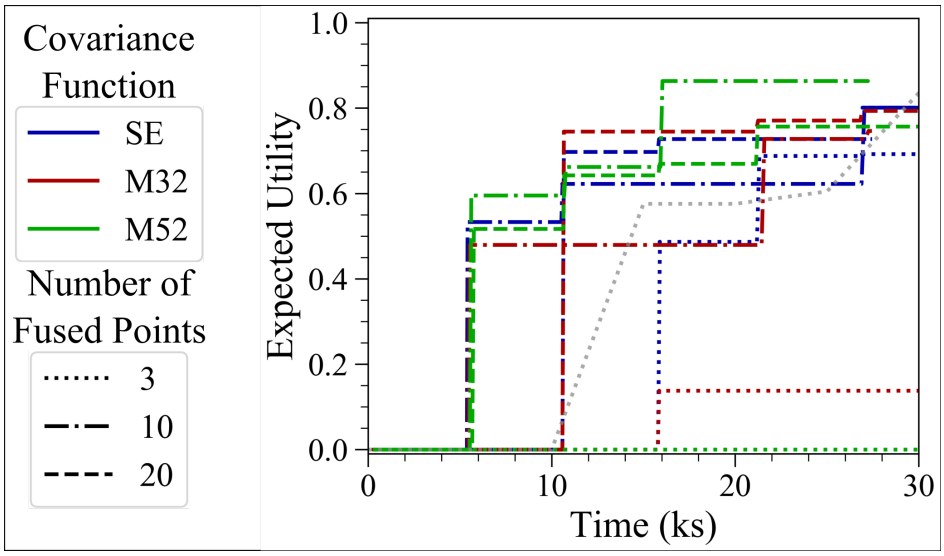


Figure B.15: Two-way parameter test for the Covariance Function and the number of fused points in the Three-Hump Camel Case Study

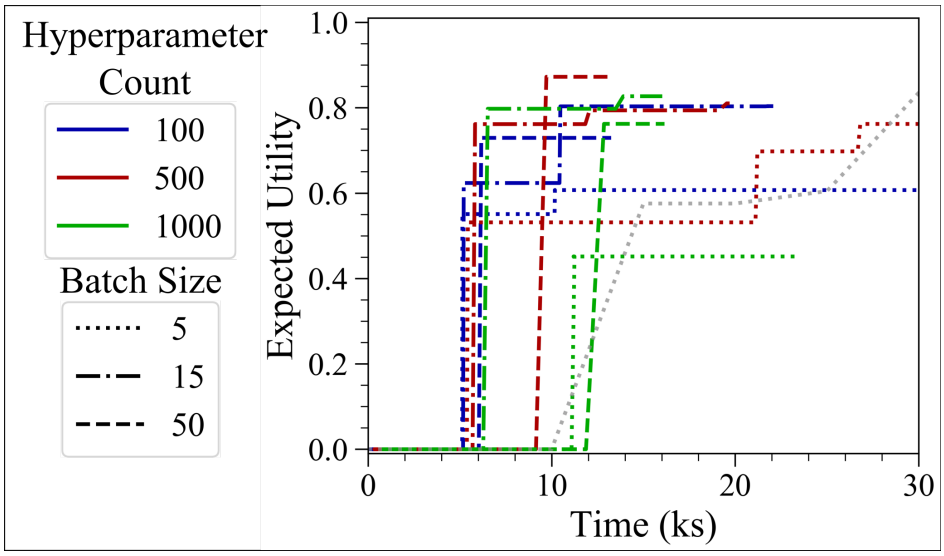


Figure B.16: Two-way parameter test for the Hyperparameter count and the Batch Size in the Three-Hump Camel Case Study

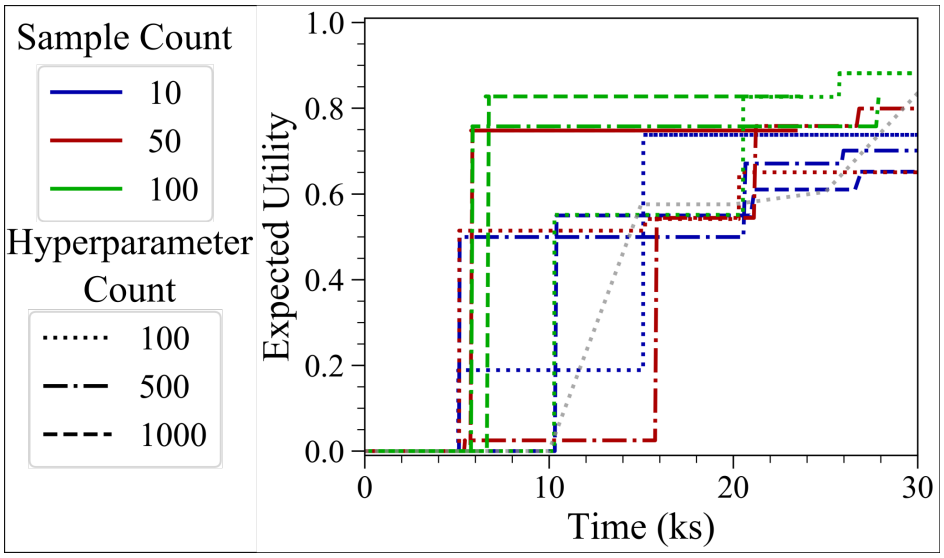


Figure B.17: Two-way parameter test for the Sample Count and the Hyperparameter Count in the Three-Hump Camel Case Study

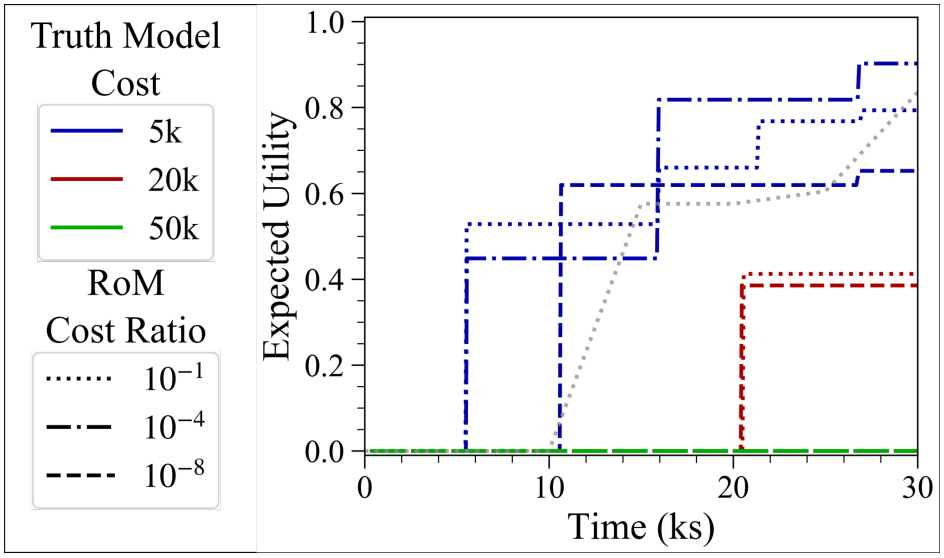


Figure B.18: Two-way parameter test for the Truth Model and Reduced Order Model costs in the Three-Hump Camel Case Study

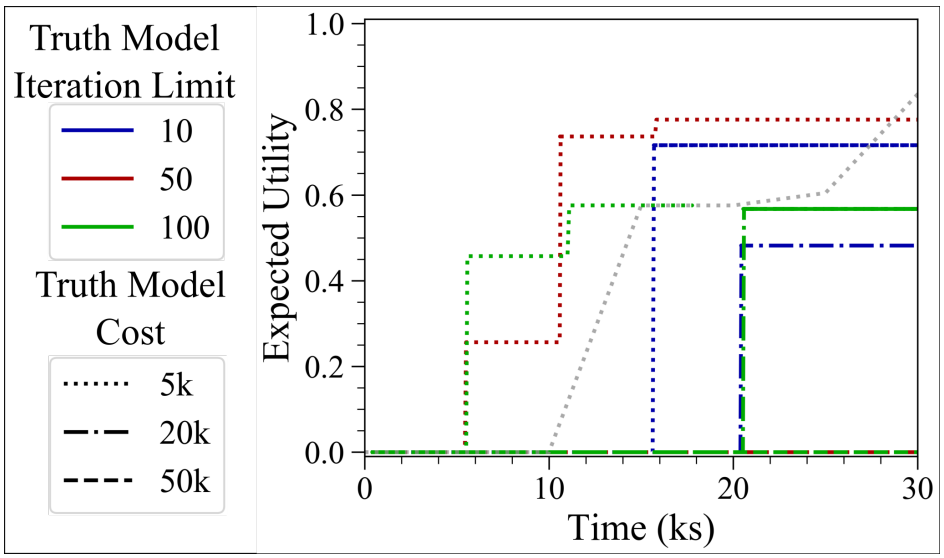


Figure B.19: Two-way parameter test for the Truth Model iteration Limit and the Truth Model Cost in the Three-Hump Camel Case Study