ADAPTIVE COMPUTING FOR OPTIMIZING HIGH-FIDELITY SIMULATION RUNTIMES

BY

JANELLE DOMANTAY

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois Urbana-Champaign, 2024

Urbana, Illinois

Adviser:

Assistant Professor Katherine Driggs-Campbell

ABSTRACT

Physics-based and high-fidelity simulations are often leveraged to predict real-world trends and optimize resource consumption. However, these simulations are often computationally expensive and time consuming. Alternatively, small scale simulations can be performed at a fraction of the cost, but generate a host of scalability issues when translated to large-scale applications. To address model scalability issues, it is necessary to identify cost efficient methods for running physics-based models. Here we demonstrate how adaptive computing can be leveraged to create surrogate models that accurately approximate high-fidelity simulation behavior at a reduced runtime. We introduce a pipeline for training surrogate models that reaffirms the effectiveness of low-fidelity simulations. We elaborate on this pipeline for multi-scale simulations which demonstrate how adaptive computing can be used to stagger high-fidelity queries during surrogate training. These implementations demonstrate how adaptive computing can be used to manipulate model run-time and increase accuracy with reduced computational budgets. These implementations can be leveraged in any existing modeling pipeline regardless of discipline.

To my parents, closest friends, and of course: Meatball.

ACKNOWLEDGMENTS

I could not have completed this report without Dr. Juli Mueller and especially Dr. Kevin Griffin. Thank you for your endless kindness in mentoring me. I had the pleasure of working with Marc Henry de Frahan who familiarized me with the KMC codebases and helped troubleshoot my compilation issues.

I'd also like to express immense gratitude to Dr. Katie Driggs-Campbell and Dr. Lei Zhao for their support in my unconventional research interests. Thank you for your feedback and recommendations in the writing of this report.

Lastly, I must acknowledge my family for their unwavering love and encouragement.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 OVERVIEW

Simulations and black-box optimization have a vast number of applications in a variety of fields including energy, materials science, and computational engineering [1]. However, the costs associated with simulating and evaluating these functions can be excessive, and compound as these processes are scaled up or applied outside of laboratory settings [2]. In order to address the challenges and computational overhead associated with model scalability, we discuss several approximate modeling methods that can be used to perform efficient global optimization [3] [4]. Specifically, we present a framework which approximate modeling to improve model performance at a fraction of estimated costs.

The main approximate modeling method we consider is referred to as multi-fidelity modeling. Model fidelity refers to how faithfully a model recreates its ended real-world behavior, or more succinctly, its accuracy. High-fidelity models are often, but not necessarily, highly dimensional models with high accuracy. In the same vein, low-fidelity models are often simpler and produce less reliable results. Multi-fidelity modeling then strives to combine models of multiple fidelities, hence multi-fidelity, into a single model for improved performance [5].

In addition to multi-fidelity models, we also present adaptive computing, a term we will use to refer to an outer-loop framework developed by the National Renewable Energy Lab. The National Renewable Energy Lab defines adaptive computing as a method for addressing scalability challenges in modelling by "strategically deploy[ing] simulations and experiments to guide decision making for scale-up analysis" [6]. Specifically, we will consider a novel framework proposed by [6], which combines multi-fidelity models and computational resource management into a novel framework that can be used to address common roadblocks associated with computational bloat [6].

1.2 MOTIVATION

In 2014, over 50 percent of the world's population was recorded to reside in urban areas [7], with this number being expected to increase dramatically. According to the 2017 International Energy Outlook, cities account for 75 percent of primary energy usage and 80 percent of greenhouse gas emissions. Of that energy consumption, over 40 percent is projected to come from buildings, with approximately 90 percent of buildings wasting 30 percent of energy consumption due to energy inefficiency [8]. With growing concerns regarding carbon

emissions and climate change, Urban Building Energy Modeling has become an attractive method for simulating and optimizing building energy consumption. Urban Building Energy Models can be used to identify buildings that have high energy consumption as well as study energy conservation protocols in buildings [9]. However, as these simulations become more complex, computational expenses becomes an insurmountable roadblock. Through the use of Adaptive Computing and multi-fidelity modeling, it is possible to approximate the performance of these models without expending excessive computing hours.

Modeling and simulations are widely used in a variety of different fields including renewable energy, bio fuel production, and building energy management. For the scope of this project, we focus primarily on the motivations of modeling through the perspective of building energy management. While models provide valuable information about the natural world, extending these systems to the real world present a myriad of challenges.

Adaptive computing provides a modeling approach that addresses technical challenges associated with scaling up data-driven models and physics-based simulations [10]. For instance, data-driven models need to be extrapolated when scaled up, which is inherently uncertain, while high-fidelity models are often too expensive to use on a large scale [10]. By addressing these challenges, scientists, researchers, and policymakers can leverage the accuracy of high-fidelity simulations for large scale practical applications such as city-wide energy modeling and predicting global trends in climate.

1.2.1 Problem Statement

Common methods for addressing the computational bloat of high-fidelity models include hybrid or surrogate models. These models involve the fusion of physics-based and data-based models. Consequently, we aim to address the following: How can adaptive computing and surrogate modeling be used to optimize resource consumption of high-fidelity models, thereby addressing scalability challenges and making high performance modeling more accessible?

1.3 OUTLINE

We begin by surveying previous research in modeling including physics-based, learning-based, and hybrid-based models. We talk generally about the benefits and drawbacks of each and move into a discussion of how approximate and surrogate modeling can be used to address some of the challenges associated with modeling. We continue by broadly defining adaptive computing and multi-fidelity modeling. We will introduce a novel framework proposed by the National Renewable Energy Lab [6], and discuss the impacts of various

sampling strategies on model accuracy. Lastly, we investigate how adaptive computing can be applied to multi-scale single-fidelity simulations. Specifically, we examine how adaptive parameters can influence the rate at which our proposed model burns computational hours. We conclude this thesis by analyzing the results of our experiments and proposing future applications for adaptive computing.

1.3.1 Contributions

This thesis presents the following contributions:

1. We define, implement, and analyze the effectiveness of various sampling strategies for a multi-fidelity framework as described by [11].
2. We implement the outer-loop framework described by [10] and [6].
3. We analyze the effects of adaptive parameters on computational runtime and model efficiency in the aforementioned adaptive computing framework.
4. We propose applications for adaptive computing and multi-fidelity modeling for energy efficient simulations and building energy modeling.

CHAPTER 2: LITERATURE REVIEW

2.1 BACKGROUND

The computational costs associated with large scale optimization and model training has necessitated a demand for energy efficient solutions. While models serve a variety of purposes, we primarily investigate how these solutions can be leveraged to decrease the computational expenses associated with energy modeling and consequently, energy consumption.

2.1.1 Challenges of the Everyday Model

Models are designed to predict real-world behavior; however, models by design are merely approximations [2]. To approximate real-world behavior, models primarily fall into one of three categories, physics-based, data-driven, or hybrid [12].

Physics-based Models

Physics-based models, also known as white box models or forward models, approximate physical behavior by relying on mathematics and statistics. Their main strengths lie in being easily analyzed and extrapolated and having high reliability. These physics-based models rely on heat and mass balance equations using physical parameters about a given building. In this way, heat transfer can be reliably estimated. In the realm of building energy modeling, popular methods for physics based models include EnergyPlus, Transient System Simulation Tool (TRNSYS), DOE-2, and Matlab [13]. EnergyPlus is considered the most widely used of these methods [14]. Generally, these models rely on Computational Fluid Dynamics (CFD), which can be used to describe the transfer of heat within and between the indoor and outdoor environment [15]. For this reason, physics-based models are often regarded as the most accurate or highest-fidelity model although this is not always the case. What is often true, however, is that these models are often time-consuming, resource-intensive and require extensive expertise and labor to properly use [12].

A clear example of this trade-off can be seen when we consider the aforementioned building energy models. In addition to requiring some level of technical knowledge for model calibration, physics-based models such as EnergyPlus rely on extensive input parameters to describe the physical components of buildings for simulations.This can include room dimensions, building material, and HVAC systems as well as the locations of doors and windows

[16]. Since data to this extent is not easily accessible on a large scale, white-box models are impractical for large-scale simulations. For this reason, white-box models cannot work in real-time, and they are not practical for modeling large buildings or multiple buildings [17]. These challenges are not unique to building modelling, as the heavy reliance on input parameter specificity combined with long runtime is a pervasive obstacle for scaling up physics-based models.

Data-driven Models

Data-driven, also known as black-box or inverse models are aptly named due to their "lack of explainability" as opposed to the physics-based white box model. Rather than relying on physical laws or explainable algorithms, data-driven models rely on historical data to train deep learning models. Popular methods include Support Vector Machines (SVM), linear regression, and Artificial Neural Networks (ANN) [13, 17]. While white-box models rely on simulating physical laws, black-box models establish linear and nonlinear relevance between inputted historical data and building consumption. As a result, they are simpler models and do not require any physical knowledge of the system. Consequently, these models are generally beginner friendly and have the capacity to run in real-time, however, training these models can still be time-consuming [12].

Black box models are proficient in interpolation but their accuracy is largely dependent on the quality of the inputted data. In the scope of building energy modelling, large amounts of historical data such as socio-demographic, historical, or billing data are used to estimate energy use in buildings. However, such datasets may not necessarily be indicative of future behavior, and may omit crucial information that contributes to energy demand. For instance, changes in climate, population, and building density can all have drastic effects on energy demand that are not reflected by historical data samples [18, 19].

Additionally, due to a reliance on interpolation scaling up black-box models poses inherent risks due to the uncertainty of extrapolation [6]. Since it is difficult to analyze the workings of a black box model, the accuracy of extrapolated predictions cannot be validated.

The accuracy of both black box and white box models is largely reliant on the quality of inputted data and parameters; however, white box models are generally regarded as more reliable due to their explainability and reliance on physics-based laws. Due to a large dependence on data, major challenges for building energy modeling include data availability, accuracy, scalability, and generalizability.

Hybrid Models

To address some of the aforementioned challenges associated with white-box and black-box models, hybrid modeling methods have been proposed. Grey box models refer to a literal combination of white box and black box models. In the scope of building energy modeling, a popular grey box model is known as a Resistance Capacitance (RC) [20] or thermal-network model which can be used to estimate the thermal resistance (R) and energy storage capacity (C) of individual rooms in a building. These parameters can be estimated from either a physics or data-driven model and can be fed into an advanced controller such as a predictive model or reinforcement learning model [17]. As a result, grey-box models can perform in real-time while maintaining the reliability of a physics-based model. Unfortunately, the definitions and implementations of grey-box models are still vague and lack specific guidelines [12].

At this point, we have discussed three forms of "primary models," physics-based, data-driven, and hybrid models. While each model presents its own individual challenges, a common obstacle present in each method is the scale-up process. In order to address these scalability issues, we discuss the applications of the approximate model.

2.1.2 Approximate Modeling

The answer to capturing the behavior of physical phenomena lies in approximate modeling. While it can be argued that every existing model is an approximation of the real world in some capacity, we define approximate models as lower-dimensional simplifications of high-fidelity (see physics-based) models. That is, where we consider physics-based models as the closest model to ground truth, approximate models purposely omit or generalize behavior in the interest of simplifying design [21]. It should be noted, however, that approximate models, including surrogate models, can be used to approximate any model's behavior.

Applications for approximate modelling date back to the 1970s wherein optimization and statistic-based methods informed airplane design exploration [21, 22, 23].

Model approximation techniques ordinarily involve four tasks:

1. **Select a sampling method:** These sample points can either be derived from ground-truth observations, or from evaluating a high-fidelity model. Sample points may be selected using either Bayesian-based methods, space-filling methods or some combination of the two [4].

2. **Selecting a model to fit sample points:** Models may be parametric, non-parametric, or semi-parametric. Parametric models use a pre-determined functional form; as a re-

sult, Parametric models can be evaluated quickly, but have limited flexibility and rely on the pre-selected form closely resembling the underlying model [4]. In contrast, non-parametric models attempt to derive the function from scratch [4].

3. **Fitting the model using a selected method** This process is usually via a regression model such as least squares regression.

4. **Validating the model** Commonly used metrics for model validation include accuracy, interpretability, robustness, dimensionality, and efficiency.

[4].

By simplifying dimensionality of a model, the approximate model strives to maintain the behavior of the original "high-fidelity model" while affording a cheaper run-time. Various forms of approximate modeling can realize this goal through various methods. One such method involves combining models of fidelities not unlike the aforementioned hybrid model.

Multi-fidelity Modeling

Multi-fidelity Modeling is a form of approximate modeling that combines models of varying fidelities into a single model to optimize computational resources [24].

Model fidelity is defined as how "faithful" or accurate a model is to capturing a desired behavior. Generally, models which have more fidelity tend to be high-dimensional and consider multiple attributes. Consider how physics based models perform more reliably by integrating additional parameters. These high accuracy models are defined as high-fidelity models (HFM). In a similar fashion, models with lower fidelity, or low-fidelity models (LFM) are less faithful due to being more simplistic. Like any approximate model, LFM's are often derived from HFM's by reducing the number of dimensions, using simpler physics models, or employing coarser domains [24]. It should be noted that the terms HFM and LFM are relative to one another and this relationship is exploited in the creation of a multi-fidelity model (MFM) [24].

While there are several methods for creating a multi-fidelity model, one of the most popular methods is surrogate-based multi-fidelity modeling. As a form of approximate modeling, multi-fidelity modeling follows the aforementioned approximation tasks in order to create this surrogate model.

In the context of surrogate-based multi-fidelity modeling, the multi-fidelity model is derived by fitting augmenting the existing LFM based on HFM evaluations [24]. The resulting surrogate model is computed by using additive, multiplicative, comprehensive or space

mapping correction paradigms on the LFM to compute expected values of the HFM. Consequently, these models are able to leverage the accuracy of HFMs while maintaining the cheaper runtime of LFMs[24, 25].

The main distinction of multi-fidelity models when compared to generic approximate modeling methods is the distinction between combining data from LFM and HFM queries. The challenges associated with this sub task will be elaborated on in the subsequent chapter.

2.1.3 Adaptive Computing

While MFM and UBEM refer to specific models that can be used to represent physical phenomena, adaptive computing is an outer loop framework designed to allocate resources over successive batch iteration [6]. The adaptive nature of this loop lends itself well to large-scale optimization or model training. Adaptive computing is designed to address scale-up problems that require the collection of many data samples and are thus computationally exhausting to run [6]. The adaptive computing framework leverages surrogate modeling and adaptive sampling strategies to optimize data sample selection [6].

Surrogate modeling is a form of approximate modeling which acts as a cheaper stand-in for an otherwise costly high-fidelity model [26].However, the quality of this surrogate model is dependent the quality and efficiency of sampling, thus various sampling methods, such as adaptive sampling algorithms, can be employed to yield higher quality models [26]. In a similar fashion, adaptive computing facilitates the selection of high quality data samples for the purposes of yielding more effective results in a reduced time frame.

2.2 APPLICATIONS AND RELATED WORK

As previously stated, approximate modeling is a useful tool for engineering design due to its ability to predict physical phenomena at a reduced cost. In a similar fashion, model coupling via explicit grey-box models or approximate models can be leveraged to optimize accuracy and cost.

Urbanization in conjunction with decreased vegetation and global warming have caused Urban heating effects in highly industrialized areas [14, 18]. Consequently, building energy consumption is projected to decrease due to increased cooling demands. Building energy models can be used to estimate and optimize building energy consumption or test the effectiveness of retrofit policies [18, 27].

Zonal models, as defined by Megri and Haghighat [28], aim leverage high cost CFD models to embellish simplified nodal models. This allows the lower dimensional nodal models to

consider how additional environmental factors such as temperature, pollutants, and fauna affect airflow through buildings [28].

Surrogate models are often derived from computationally expensive approximate simulations to reduce runtime [27, 29]. These surrogate models have been used in lieu of high-fidelity simulations but can be in danger of producing imprecise or biased results. Thus, [29] proposes combining detailed simulations and approximate simulations to produce a more accurate surrogate model.

While this thesis highlights how these approximate modeling techniques relate to building energy models specifically, there is no shortage of applications to engineering design in aerospace [22, 23], industrialization [30], and natural resource management [31]. With model coupling, surrogate models, and approximate models being highly applicable to any form of intensive modeling, we consider the utility of adaptive computing.

CHAPTER 3: ADAPTIVE SAMPLING

3.1 OVERVIEW

As aforementioned, the high costs associated with high-fidelity models (HFM) can make them impractical. Therefore, employing surrogate models allow users to reduce costs for optimization and uncertainty quantification [5]. By leveraging these technologies, highly accurate models and simulations can become accessible to a wide variety of fields and applications.

Generally, as depicted in Figure 3.1, more accurate models tend to be more complex, and thus have high computational costs. Reducing dimensionality reduces some of these costs but generally results in less accurate models. These models are referred to as low-fidelity models (LFM). Multi-fidelity surrogate models offer a compromise between HFMs and LFMs, by extrapolating HFM behavior based on samples taken from both the HFM and LFM.

When designing multi-fidelity surrogate models, design considerations include (1) choosing an underlying surrogate, (2) determining how to relate the model fidelities to one another, and (3) communicating uncertainty [6]. In this chapter, we consider design consideration (2) in the form of adaptive sampling.

Multi-fidelity surrogate models are created by sampling both high and low-fidelity models

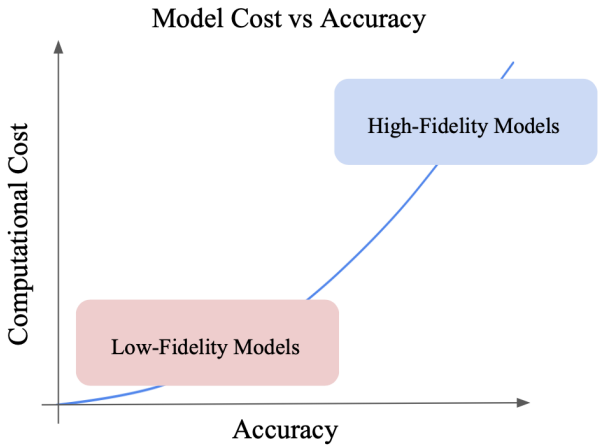


Figure 3.1: The trade-off between accuracy and computational cost. High-fidelity models are more complex and yield more accurate predictions but are computationally expensive. As we simplify these models, they become less expensive but less accurate. The goal of multi-fidelity models is to combine both high and low-fidelity models to produce a surrogate model that is both accurate and inexpensive.

to create a surrogate model [32]. However, due to the nature of low-fidelity models, it is not always evident how helpful the low-fidelity information is for predicting HFM behavior. This is particularly concerning for LFMs that have distinctly low correlation with their corresponding black-box HFMs. Consequently, we propose a number of sampling strategies that can be used to evaluate the reliability of the LFM for informing whether to sample the low-fidelity or high-fidelity model at a given point in order to facilitate locating an HFM global optimum. The contributions of this section include the following:

1. Analyzing the effects of low-fidelity model samples on finding the global minima of the high-fidelity model.

2. Comparing the effectiveness of four sampling strategies for accurately predicting and sampling the global minima.

3.2 IMPLEMENTATION

To construct a multi-fidelity surrogate model, we select a reference function that will define the behavior of our HFM. Ordinarily, an LFM is defined by reducing the dimensionality of the HFM, but for our purposes we create functions that resemble the HFM with slight deviations as shown in Section 3.3.2. Next, we leverage a space-filling sampling algorithm to seed our surrogate model. Specifically, we use Latin-Hypercube-Sampling [33] to evaluate the LFM and HFM and random points. Using this training input, we utilize Gaussian Process Regression [34, 35] to train our initial surrogate model. This surrogate model functions as the multi-fidelity model (MFM) which predicts the behavior of the HFM, and therefore, the global minimum.

After creating our surrogate model, we continuously train this model over a series of iterations by taking subsequent samples from either the high-fidelity (HF) sample or low-fidelity (LF) sample from a point selected by an acquisition function. Our acquisition function determines which sample point will maximize the expected improvement of the MFM. We refer to this acquisition function as the Expected Improvement function.

Once our acquisition function selects a sample point, we must decide between sampling the HFM and the LFM. In order to reduce computational costs we are interested in minimizing the number of HFM queries while maintaining accuracy. The MFM is updated with the new samples and the process is repeated over a series of user-defined iterations with the goal of returning a near-global minimum of the HFM. However, in order to validate our proposed global minima, it is necessary for one of our HFM samples to be the global minimum of the

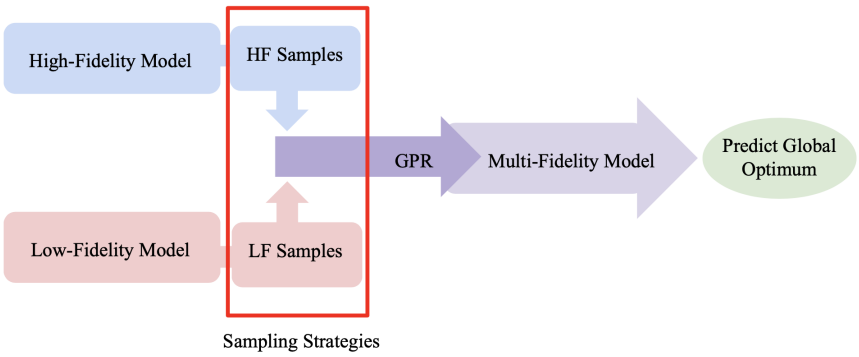


Figure 3.2: Flowchart illustrating pipeline of multi-fidelity model construction. We employ sampling strategies in the training portion outlined in red.

HFM. In order to determine whether to sample from the HFM or LFM, we employ several sampling strategies.

3.3 NUMERICAL EXPERIMENTS

The numerical experiments of this study were run using the Surrogate Modeling Toolbox [36], in conjunction with the NREL Adaptive Computing codebase[11].

3.3.1 Sampling Strategies

Defining our sampling strategies relies on several key factors, namely correlation and sample ratio. Specifically, we must consider how well correlated the LFM and HFM are, and how that will impact MFM accuracy. While it's commonly accepted that highly correlated LFM's increase surrogate model accuracy [32, 37], it is necessary for our strategies to perform well on low correlated models. This is to account for experiments where the correlation between the LFM and HFM is unknown.

To determine the ratio of HFM to LFM samples, we reference [37], which concludes that LFM samples should utilize between 10 percent and 80 percent of the simulation budget while still exceeding the number of HFM samples.

Thus, we define four separate sampling strategies for every sample point s:

1. **Correlation-based sampling** refers to sampling the HFM when the LFM has relatively low correlation to the HFM. We determine this by training two separate models via Gaussian Process Regression: one for the LFM and one for the HFM. For every s,

we query these two Gaussian Process Regression models around the neighborhood of s. We then compute the correlation coefficient r between the outputs of the low-fidelity Gaussian Process Regression and high-fidelity Gaussian Process Regression. If the correlation coefficient is high, we default to sampling the LFM, otherwise we sample the HFM. As evidenced by previous research [32, 37], the correlation coefficient should be relatively high with $r^2 > 0.9$ [37].

2. **Iteration-based sampling** refers to sampling some percentage of the established energy budget from the LFM, and the energy budget from the HFM. This comes from the assumption that after some point, LFM samples will not improve the global minimum, so any additional improvements may only be made by sampling the HFM. We limit the iteration bound i to be $0.1 \leq i \leq 0.8$, based on [37].

3. **Uncertainty-based sampling** refers to checking the uncertainty of s. If the point s is located in an area with high uncertainty, that is there are a low number of samples taken in the adjacent area, we sample the HFM, otherwise we sample the LFM.

4. **Improvement-based sampling** refers to checking the current predicted optimum based samples up to sample point s. If there is no change in the predicted optimum for the previous samples taken, we sample the HFM at the current predicted optimum. We then sample areas around the predicted optimum for the next n samples where n is a user-set parameter.

3.3.2 1-Dimensional Models

To determine the effectiveness of our sampling strategies on 1-dimensional models, we test our strategies on the Forrester model. The Forrester model is a well-known benchmark problem for testing multi-fidelity methods [38] and is defined as

$$f(x) = (6x - 2)^2 \sin(12x - 4). \tag{3.1}$$

We utilize three different variations of the Forrester model, $f_1(x)$, $f_2(x)$ and $f_3(x)$ where $f_1(x)$ is most correlated and $f_3(x)$ is least correlated [38], such that

$$f_1(x) = (5.5x - 2.5x)^2 \sin(12x - 4) \tag{3.2}$$

$$f_2(x) = 0.75f(x) + 5(x - 0.5) - 2 \tag{3.3}$$

$$f_3(x) = 0.5f(x) + 10(x - 0.5) - 5 \tag{3.4}$$

.

We noticed that the control strategy performed well in all test cases. However, the control was outperformed by the iteration-strategy when given a bound of 0.5. The iteration strategy also appeared to be less effective as correlation increased. In comparison to the other sampling strategies, the iteration sampling strategy made the most noticeable change.

3.3.3 2-Dimensional Models

In order to test the performance of our sampling strategies on a 2-dimensional model, we introduce the Branin model as a test function. We define the Branin model as

$$b(x_0, x_1) = (x_1 - \frac{5.1}{4\pi^2}x_0^2 + \frac{5}{\pi}x_0 - 6)^2 + 10(1 - \frac{1}{8\pi}\cos x_0) \tag{3.5}$$

where $x_0 \in$ [-5, 10] and $x_1 \in$ [0, 15] [39]. Similar to the Forrester function, we also employ a low-fidelity version of the Branin function as

$$b_l(x_0, x_1) = b(x_0, x_1) - (A + 0.5)(x_0 - \frac{5.1}{4\pi^2}x_0^2 + \frac{5}{\pi}x_0 - 6)^2, \tag{3.6}$$

where the correlation of the LFM is defined by $A \in 0, 0.25, 0.5$ such that A $= 0$ is most correlated and A $= 0.5$ is least correlated.

We determined the control strategy to be relatively effective in comparison to the individual sampling strategies. Of the individual strategies, the iteration-based sampling strategy

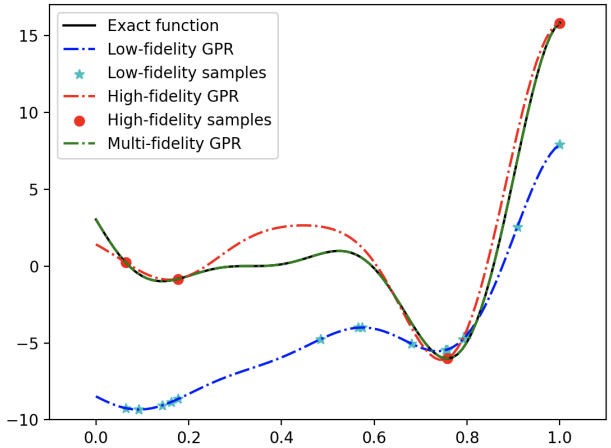


Figure 3.3: A multi-fidelity model trained on a low correlation Forrester model. Multiple low-fidelity model samples were taken throughout the model. These initial samples were used to prompt an accurate high-fidelity model sample near the global minimum.

was most effective for the well correlated LFM models, while the correlation based strategy performed best for uncorrelated LFM models. The best performing models consisted of a combination of iteration-based and improvement-based sampling strategies for A = 0 and A = 0.25 models. The least correlated model A = 0.5 was best predicted by a combination of correlation, iteration, and uncertainty based sampling strategies. The results of the aforementioned experiments are displayed in Table 3.1.

3.4 RESULTS

We observe that the LFM samples taken in this study had a marked impact on the quality of HFM samples and accuracy of the overall MFM model. The impact of the iteration sample strategy is well illustrated in Figure 3.4 and Table 3.2. As illustrated in Figure 3.4, by initially sampling the LFM exclusively, we were able to identify several local minima within the model without expending the energy required of sampling the HFM. Afterwards, the model is able to compare the true values of each local minima against each other by sampling the HFM at each point of interest.

While this holds well for our one-dimensional models, we find that the sampling strategies were less effective for 2-dimensional models. For the Branin model, we found the iteration sampling strategy to be most effective individually as compared to the other sampling strategies. However, neither of these strategies were able to independently outperform our current

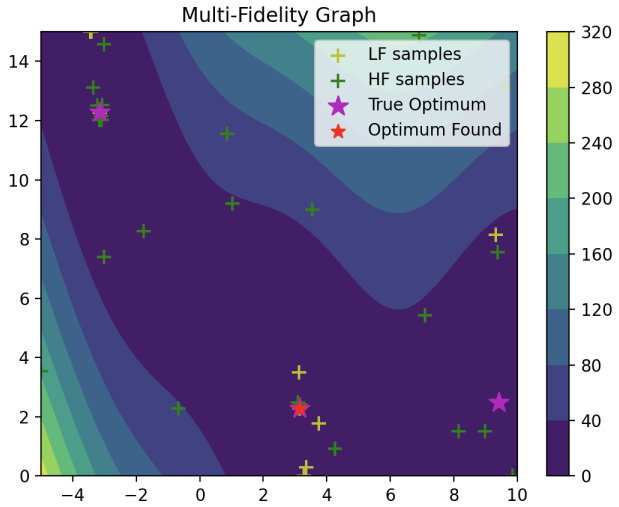


Figure 3.4: A contour graph of the Branin function using a combination of sampling strategies 1, 2, and 4. The HF samples tend to congregate around each of the three global optimums.

implementation. We found a combination of iteration and improvement based sampling strategies to be most effective for well-correlated LFMs (A = 0, 0.25), while a combination of correlation, iteration, and uncertainty tended to work best for poorly correlated LFMs (A = 0.5). These methods outperformed the control method in accuracy, but tended to consume a large number of HF samples. We identified methods that consumed less HF samples but had slightly higher error values. Generally, we were not able to establish a generalized strategy for all correlation values.

Further testing is needed to evaluate the effectiveness of these strategies for models with additional dimensions. Specifically, we are interested in studying how altering parameters associated with iteration, i, and correlation, r, may impact the number of LFM samples as well as the accuracy of the resulting surrogate model. We are also interested in exploring alternate optimization methods and evaluating how the results of this thesis may apply to models with dimensions greater than two.

3.5 SUMMARY

In this section, we analyzed the effects of different sampling strategies on global minima prediction for MFMs. We presented two main contributions. Firstly, we identified an effective sampling strategy to increase the accuracy of global minima predictions for 1-dimensional models. Secondly, we compared the accuracy and computational resource consumption of four different sampling strategy.

The simulation results reveal several key points. Namely, we identified the iteration-based sampling strategy with an iteration bound of 0.5 to have the lowest error when predicting global minima. However, our experiments yielded less conclusive results for our 2-dimensional model. Generally, we found that combinations of sampling strategies were most effective, but further study is needed to establish a reliable strategy for 2-dimensional and n-dimensional models.

In future work, we aim to conduct more rigorous numerical experiments for single dimensional and multi-dimensional models. Specifically, we aim to define statistically significant correlations between sampling strategy methods, and global minima prediction accuracy.

Correlation	Sampling Strategy	Samples at Convergence	Error
A = 0	Control	40 LF, 12 HF	3.37E-01
	1	95 LF, 1 HF	4.88E-01
	2	25 LF, 10 HF	8.57E-01
	3	90 LF, 2 HF	4.21E+00
	4	65 LF, 7 HF	3.60E+00
	1, 2	25 LF, 15 HF	1.35E-01
	2, 3	20 LF, 9 HF	3.46E-02
	2, 4	**15 LF, 6 HF**	**5.17E-05**
	1, 2, 4	15 LF, 14 HF	1.83E-03
	2, 3, 4	3 LF, 10 HF	5.80E+00
A = 0.25	Control	27 LF, 15 HF	3.15E-01
	Control	405 LF, 19 HF	1.30E-02
	1	90 LF, 2 HF	1.14E+00
	2	25 LF, 15 HF	2.49E-01
	3	95 LF, 1 HF	4.13E+00
	4	51 LF, 5 HF	5.80E+00
	1, 2	20 LF, 15 HF	8.18E-01
	2, 3	25 LF, 14 HF	8.78E-03
	2, 4	**15 LF, 14 HF**	**2.29E-04**
	1, 2, 4	10 LF, 5 HF	6.77E-03
	2, 3, 4	4 LF, 10 HF	5.80E+00
A = 0.5	Control	35 LF,13 HF	2.27E-02
	1	90 LF, 2 HF	6.57E-01
	2	25 LF, 10 HF	1.82E+00
	3	95 LF, 1 HF	4.69E+00
	4	11 LF, 1 HF	5.80E+00
	1, 2	20 LF, 13 HF	1.48E-01
	2, 3	25 LF, 12 HF	3.34E-01
	2, 4	15 LF, 14 HF	3.65E-01
	1, 2, 4	**14 LF, 13 HF**	**1.14E-02**
	2, 3, 4	4 LF, 8 HF	5.80E+00

Table 3.1: Results from testing sampling strategies on Branin test functions. Correlation values 0, 0.25, and 0.5 indicate how well correlated the low-fidelity model is to the high-fidelity model where 0 is most correlated and 0.5 is least correlated. The control strategy refers to a model run on the current implementation of the multi-fidelity model without any additional sampling strategies. All models are run with an energy budget of 100. The iteration sample strategy was run with a bound of 0.25. The samples at convergence column denote the quantity of samples taken by the model once it converges to a global minimum.

Correlation	**Iter. Bound**	**Samples at Convergence**	**Error**
Low (f_3)	Control	6 LF, 4 HF	1.48E-05
	0.1	3 LF, 4 HF	1.06E-03
	0.25	7 LF, 2 HF	4.50E-06
	0.5	**13 LF, 2 HF**	**5.00E-08**
	0.75	21 LF, 1 HF	1.74E-04
Medium (f_2)	Control	8 LF, 2 HF	3.66E-06
	0.1	2 LF, 3 HF	2.77E+00
	0.25	5 LF, 2 HF	2.87E-03
	0.5	**10 LF, 1 HF**	**9.20E-07**
	0.75	21 LF, 1 HF	1.74E-04
High (f_1)	Control	8 LF, 2 HF	3.49E-06
	0.1	2 LF, 3 HF	2.75E+00
	0.25	5 LF, 2 HF	4.12E-03
	0.5	**10 LF, 1 HF**	**9.40E-07**
	0.75	21 LF, 1 HF	1.74E-04

Table 3.2: Results of testing iteration sampling strategy on 1-dimensional Forrester function. Iter. Bound refers to the percentage of energy budget that select low-fidelity model for sampling. After training exceeds this percentage, the strategy will exclusively sample from the high-fidelity model. The energy budgets were determined by identifying the minimum budget needed for the control method to converge. A Control iteration bound refers to running the current implementation of the multi-fidelity model without any sampling strategies. F4 methods were run with an energy budget of 26, while F3 and F2 used an energy budget of 18. Samples at Convergence refers to the quantity and fidelity of samples taken at convergence of the high-fidelity model such that n LF, m HF means that n low-fidelity queries and m high-fidelity queries were made prior to convergence.

CHAPTER 4: ADAPTIVE BUDGET

4.1 OVERVIEW

In the previous chapter, we investigated how sampling strategies affect the performance of the multi-fidelity surrogate model. We will now investigate how altering the thresholds of the adaptive computing outer loop can be used to select effective simulations and improve runtime for single-fidelity and multi-fidelity simulations.

We have previously introduced adaptive computing as an outer computing loop for "strategically deploy[ing] simulations and experiments to guide decision making for scale-up analysis" [6]. Specifically, adaptive computing was designed with the intent of facilitating the transition of laboratory or research-based models into large-scale applications [10]. However, this process often faces challenges associated with model scalability. To combat these issues, the National Renewable Energy Lab introduces "strategic simulation," which succinctly highlights the two core goals of our proposed methods: (1) Optimize Candidate Simulations - determining the best simulations to run. (2) Optimize Simulation Run-Time - determining at what point to run a simulation and additionally whether to run a low-fidelity or high-fidelity simulation.

By introducing candidate simulations, we can employ a "vetting process" wherein each candidate simulation is evaluated for its utility prior to its runtime. This pre-runtime check allows the user to avoid spending computational hours on a simulation that provides irrelevant data or is otherwise redundant.

The goal of our pipeline is to determine which candidate simulations will provide the most valuable information for our surrogate model. Generally, the more high-fidelity samples used, the more accurate the surrogate model. In order to maximize the effectiveness of the surrogate model, we can prioritize simulation candidates with higher variance. through the use of a threshold parameter. We will later introduce how adapting this threshold parameter can be used to manipulate the computational burn rate of the adaptive computing model.

Consequently, this section contributes the following:

1. Implementing an adaptive computing framework for multi-fidelity simulations and multi-scale simulations.

2. Constructing a pipeline for adjusting computational burn rate to a pre-specified target function.

3. Analyzing impacts of methods for adjusting "adaptive threshold" parameter on computational burn rate.

4.2 IMPLEMENTATION

As previously stated, adaptive computing can be used for general use multi-fidelity simulations. However, this implementation was originally developed for the Adaptive Mesh and Algorithm Refinement (AMAR) multi-scale modeling framework. Rather than strictly utilizing high-fidelity and low-fidelity simulations as specified in Figure A.1, the Adaptive Mesh and Algorithm Refinement framework differs between calling a micro scale kinetic Monte-Carlo simulation, and estimating the micro scale simulation by referring to the surrogate model. In this case, the model functions more as a single-fidelity simulation. Consequently, the results of this experiment do not query a low-fidelity simulation and instead rely only on the single simulation. An accurate summary of this model is depicted in Figure 4.1.

To implement the following adaptive computing model, we created an interface that passes information between the Python-based multi-fidelity modeling toolbox [11], and a placeholder simulation model written in C++. Eventually, this placeholder model is intended to be replaced with a kinetic Monte-Carlo simulation as specified by [40].

4.2.1 Objectives

In addition to the previously explored multi-fidelity model, we address the challenges associated with scale-up using two additional components. The first is by introducing candidate simulations which can be iterated through via our adaptive computing loop. The second is an adaptive threshold, which serves as a mechanism for adapting the burn rate of our computational budget.

4.2.2 Candidate Simulations

Constrained by both time and resources, it is necessary to maximize the effectiveness of each simulation. In the previous chapter, we utilize online training, meaning that a simulation would be evaluated for either a high-fidelity or low-fidelity query. With the introduction of the adaptive computing loop, we can instead process simulations in batches. These batches of candidate simulations allow each simulation to have a comparison point. A single batch of candidate simulations will be constrained by a single threshold value. This threshold value can be adapted, as we will describe in subsequent sections, to favor running candidate simulations with the highest variance in the batch.

4.2.3 Budget Comparisons

Optimizing simulation run-time requires the introduction of two more key resources: computational budget and time budgets. Since high-fidelity simulations take a considerable amount of time, it is necessary to provide some limit to the number of high-fidelity simulations in the interest of running additional low-fidelity simulations. Similarly, one wants to allot enough time to the wall-clock timer to allow for a sufficient number of high-fidelity samples to be run. The specifics of this trade-off are further outlined in the previous chapter.

Computational and Real-time Budgets

When scheduling high-performance computing, modellers are often allotted a set number of server hours and a set number of run-time hours. This allows them to designate how long the simulation should run for, or a wall-clock time limit, as well as how many hours should be used specifically for running a high-fidelity simulation. These two time limits define the real-time budget and computational budget respectively.

In addition to managing the how many hours to give to each budget it is also necessary to consider how often to sample the high-fidelity model. Thus, we define a target-burn rate. Ideally, a simulation should consume all allotted computational budget hours by the time the designated wall-clock budget ends. This requires us to distinguish between two different burn rates.

Burn Rates

Our first burn-rate is constant, as this burn-rate is representative of the wall-clock timer. Since the wall-clock budget is consumed regardless of the amount of computational hours

consumed, the wall-clock budget serves as our ideal target burn rate. In contrast, the computational budget is only consumed every time a high-fidelity simulation is queries. Thus, the computational burn rate is determined by the rate at which the computational budget is consumed over with respect to the wall-clock timer's rate of consumption.

In the initial deployment of [11], the burn-rate is not explicitly specified. Instead, the frequency of high-fidelity simulation queries was determined by using an arbitrarily defined threshold. This threshold served as a comparison point for the predicted variance of a candidate simulation.

We have explored the effects of modifying this threshold in our previous experiments. Values used to define his threshold included a correlation coefficient, projected uncertainty, or predicted optimum samples. In the following numerical experiment we will instead introduce an adaptive threshold which is used to manipulate the burn-rate of the computational budget thus ensuring that high-fidelity evaluations are spread across the entirety of the wall clock simulation time.

Having defined the two additional components of our pipeline: candidate simulation batches and adaptive computation burn rates, we can establish our pipeline for training the adaptive computing model.

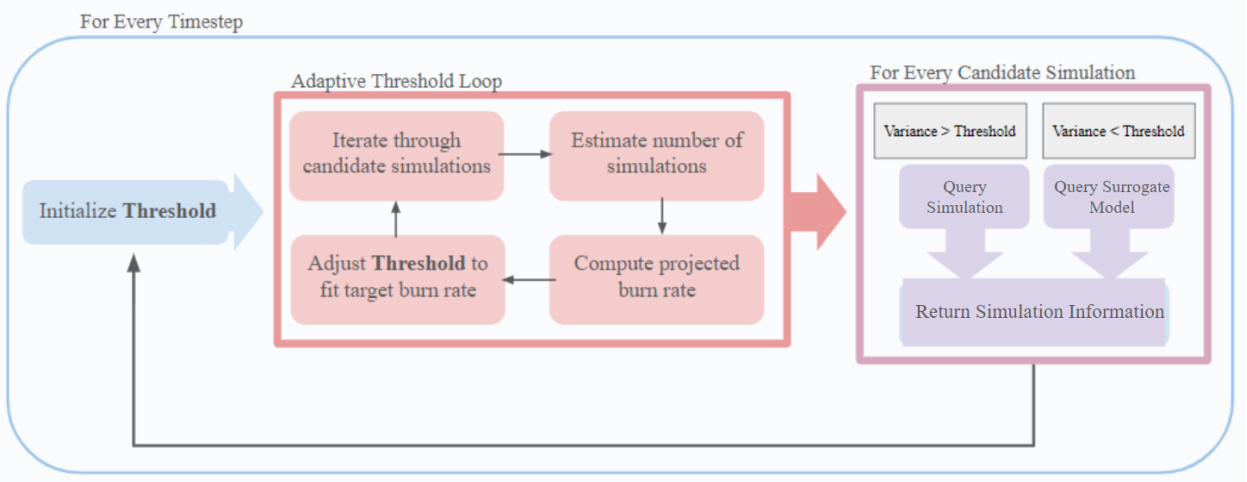


Figure 4.1: Outline of micro-scale simulations for Adaptive Computing results. The adaptive threshold parameter is initialized at the beginning of each timestep. The threshold is fine-tuned to fit a specified burn rate in the Adaptive Threshold Loop, wherein the burn rate is estimated for a list of candidate simulations. After fine-tuning, the threshold is used to designate which fidelity models to run for each candidate simulation. The resulting data is added to the surrogate model.

4.2.4 Pipeline

The process of training the model, or incorporating samples into our surrogate model can be broken down into two main processes. It should be noted that the implementation of this model was intended for single fidelity modelling. However, the concept of deciding between querying a surrogate model and querying a simulation can be applied to the multi-fidelity framework discussed in previous chapters. It should be noted, that in the following discussion, any reference to querying a surrogate model can be replaced with querying a low-fidelity model in order to apply this framework to multi-fidelity modeling.

Model training is performed for every time step $n \in [0, 1, ...N]$ We begin by estimating the burn-rate of our computational budget in order to adjust our adaptive parameter, the adaptive threshold. This preliminary loop is referred to as the "Adaptive Threshold Loop" in Figure A.1.

Budget Estimation

We begin with an initialized threshold value t_0. Let $C_n = [c_{n0}, ..., c_{np}]$ be a batch of candidate simulations in a set of batches $[C_0, .., C_N]$. For every iteration $m \in [0, 1, ..., M]$ we will perform the following. If $m = 0$, let $t_m = t_0$.

In order to estimate the computational burn-rate, we must iterate through each candidate simulation c of C_n and perform a check wherein we compare the estimated variance of c to our current threshold, t_m. This process is largely similar to the check performed we performed when training a multi-fidelity surrogate model in the previous chapter. The check is used to determine whether to query the surrogate model to estimate a return value, or to consume computational hours running a high-fidelity simulation. However, rather than immediately proceeding to query either model, we simply note down the number of computational hours that would be consumed had we run the high-fidelity simulation. These initial estimates are used to define a hypothetical computational burn-rate for our current threshold value.

To compute the number of computational hours consumed during our estimates, we multiply the runtime of our high-fidelity simulation by the number of estimated high-fidelity queries to our projected-burn rate. In other words, our projected-burn rate is computed at time step n by

$$cr_{nm} = \text{comp}_n + \text{num}_{HFm} * \text{time}_{HF} \tag{4.1}$$

where comp_n represents the current number of computational hours consumed, num_{HFnm} represents the number of high-fidelity simulations estimated from iteration m for candidate

simulations C_n, and time_{HF} represents the amount of time taken to run a single high-fidelity simulation. For the purposes of the following experiments, we set $\text{time}_{HF} = 1$, such that a single computational budget hour can be used to run a single high-fidelity simulation. While we use a dummy simulation for our purposes, in practice a single high-fidelity simulation can take anywhere from a few minutes to several hours depending on the complexity.

Now, with an estimated burn-rate for our computational budget, we compare our projected computational budget to our target burn rate. Our target burn rate should resemble the consumption rate of our real-time budget. Since the real-time budget is consumed at a constant rate, we can determine if the computational budget is being consumed too quickly. If the computational budget consumption rate exceeds that of our time budget consumption rate, we can increase our adaptive threshold to decrease the number of high-fidelity simulations, num_{HF} performed on our candidate simulations. Similarly, if the computational budget is burning too slowly, we can decrease the threshold to encourage additional high-fidelity simulations.

This process is repeated for M iterations in order to fine-tune the adaptive threshold prior to our final iteration.

Final Iteration

The final pass of our candidate simulation batch, labeled as "For Every Candidate Simulation" in Figure A.1, performs a single loop through our candidate simulation batch. With a finalized threshold adapted from the previous estimates, every candidate simulation is used to query either the surrogate model or high-fidelity simulation accordingly. Following the final iteration, the process is repeated with a new batch of candidate simulations, C_{n+1} and a re-initialized threshold t_0.

4.2.5 Definitions

For the following numerical experiments, we set a CPU budget of 100, $N = 65$ and $M = 10$. Consequently, a complete simulation run can have up to a maximum of 100 high-fidelity simulations. For every time step n, each batch of simulations, C_n, is composed of 64 candidate simulations such that $C_n = [c_{n0}, ..., c_{n64}]$. Each simulation, c_{ni} requires 4 input values for a query. The continuous input values $[x_{i0}, x_{i1}, x_{i2}, x_{i3}]$ are computed for each candidate simulation c_{ni} at a given time step n as follows:

$$x_{i0} = 20.0 + \frac{n + \frac{i}{64}}{2} \tag{4.2}$$

$$x_{i1} = \frac{0.5 + n + \frac{i}{64}}{10} \tag{4.3}$$

$$x_{i3} = \frac{n + \frac{i}{64}}{65} \tag{4.4}$$

$$x_{i4} = 1 - x_{3i} \tag{4.5}$$

where $x_{i0} \in [0, 300]$, $x_{i1} \in [0, 100]$, $x_{i2} \in [0, 1]$, and $x_{i3} \in [0, 1]$. At the beginning of each time step, we re-initialize our adaptive threshold to some value t_0. We can estimate the ratio of computational hours elapsed for a given threshold t_m at iteration n using cr_{nm}. For every iteration m, t_m is modified in accordance with one of the three following methods.

1. Percent - t_m at timestep n is increased or decreased by some percentage value ch_i or ch_d such that if $cr_{nm} < (n/N)$,

   $$t_m = t_{m-1} - ch_d t_{m-1} \tag{4.6}$$

   and if $cr_{nm} > (n/N)$,

   $$t_m = t_{m-1} + ch_i t_{m-1}. \tag{4.7}$$

2. Ratio - the adaptive threshold is increased and decreased by a percentage computed from the difference between elapsed time budget and the estimated elapsed computational budget such that

   $$t_m = t_{m-1} + (cr_{nm} - \frac{n}{N}). \tag{4.8}$$

3. Secant - the adaptive threshold, t_m is modified in accordance to the secant method at every adaptive threshold iteration m, such that

   $$t_m = t_{m-1} - \frac{r_{m-1}(t_{m-1} - t_{m-2})}{(r_{m-1} - r_{m-2})}, \tag{4.9}$$

   given that $r_{m-1} = cr_{nm} - \frac{n}{N}$, $t_1 = t_0 1.1$, and $r_0 = 0.5$.

4.2.6 Test Functions

We implemented a four-dimensional Branin [39] function to serve as a stand-in for a micro-scale simulation or single fidelity simulation. While we initially created four-dimensional implementations for the Rosenbrock [41] and Forrester [38] test cases, we did not notice any differences in error metrics between either of our test functions. For this reason, we have elected to solely present findings from the Branin model. In contrast to the Branin model implemented in the previous chapter, we only utilize a single fidelity model for our testing purposes as depicted in Figure 4.1.

Given $a = 1$, $b = \frac{5.1}{(4\pi)^2}$, $c = \frac{5}{\pi}$, $r = 6, s = 10$, and $t = \frac{1}{8\pi}$, we define the Branin model as:

$$b(x_{i0}, x_{i1}, x_{i2}, x_{i3}) = a(x_{i1} - bx_{i0})^2 + c(x_{i0} - r)^2 + s(1 - t)\cos(x_{i0}) + s + x_{i2} + x_{i3}, \quad (4.10)$$

4.3 NUMERICAL EXPERIMENTS

4.3.1 Control

We define two control methods to compare the effectiveness of adaptive parameters.

Baseline

We create a baseline to compare our method against by removing the "Adaptive Threshold Loop" as depicted in Figure A.1. Without any comparisons being formed between the budget consumption rates, this results in the same threshold being used for all simulations. As a result, the performance of the baseline model is wholly dependent on t_0, the threshold initialization. Consequently, manually selecting a threshold comes with several challenges. Since the threshold is constant, selecting a threshold that is too high will result in the threshold always being higher than the variance, and no high-fidelity simulations will run. In contrast, selecting a threshold that is too low will result in a simulation always being run, and the computational budget will be entirely consumed at the very beginning of the experiment without regard for the remaining time steps. Even an ideal scenario as shown in Figure 4.2, in which the threshold bisects the variance to some degree to have enough variation, is at risk of consuming the computational budget quickly. Thus, a constant threshold is insufficient for meeting our objectives.

In order to fine-tune the threshold, it is necessary to increase or decrease the threshold by some value in response to the average variance of the candidate simulation batch. During

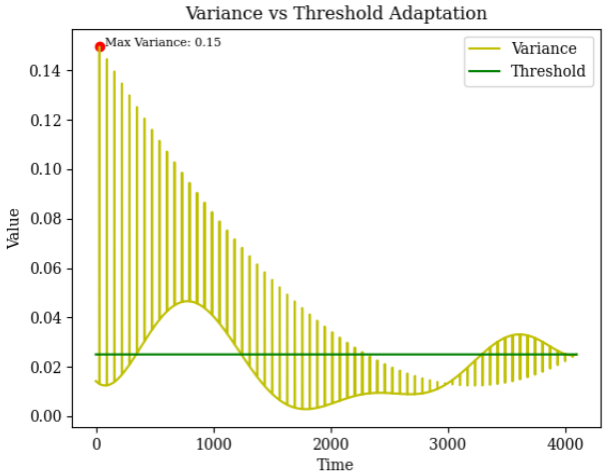


Figure 4.2: An ideal constant threshold for our baseline methods. Manually selecting the threshold boasts a variety of challenges. Ideally, the threshold should alternate between below and above the variance in order to allow the model to evaluate the high-fidelity model at different points of the simulation.

the adaptive computing loop, the threshold oscillates accordingly, as depicted in the right hand image of Figure 4.3. During the final iteration, the fine-tuned threshold is used to determine which high-fidelity queries to run. An example comparison between the threshold values in the adaptive computing loop and the final iteration loop is present in Figure 4.3. We explored several methods for fine-tuning the threshold during the training process.

Percent

Our initial method involved increasing and decreasing the threshold by an arbitrary percentage as defined by ch_i and ch_d. Similar to our baseline methods, the percent method's effectiveness relies heavily on the interaction between t_0, ch_i and ch_d. Generally, if the difference between t_0 and the model's variance exceeds the range that can be covered by ch_i and ch_d during the adaptive threshold iteration cycles, t_m will oscillate between approaching 0 and approaching inf. In other words, the model will adopt the strategy of either running every candidate simulation of a given time step n as a high-fidelity simulation, if t_m approaches 0, or as a low-fidelity simulation if t_m approaches ∞.

In contrast, when ch_i and ch_d are sufficiently small, the threshold value is able to adjust more gradually. resulting in an evenly spaced staircase pattern alongside the target function. However, if the increments are too small, there is a risk of the model not being able to converge to a threshold within the adaptive computing loop. This can lead to budgets not being completely consumed, or t_m stalling at a specific value, as shown in Figure 4.4. In this

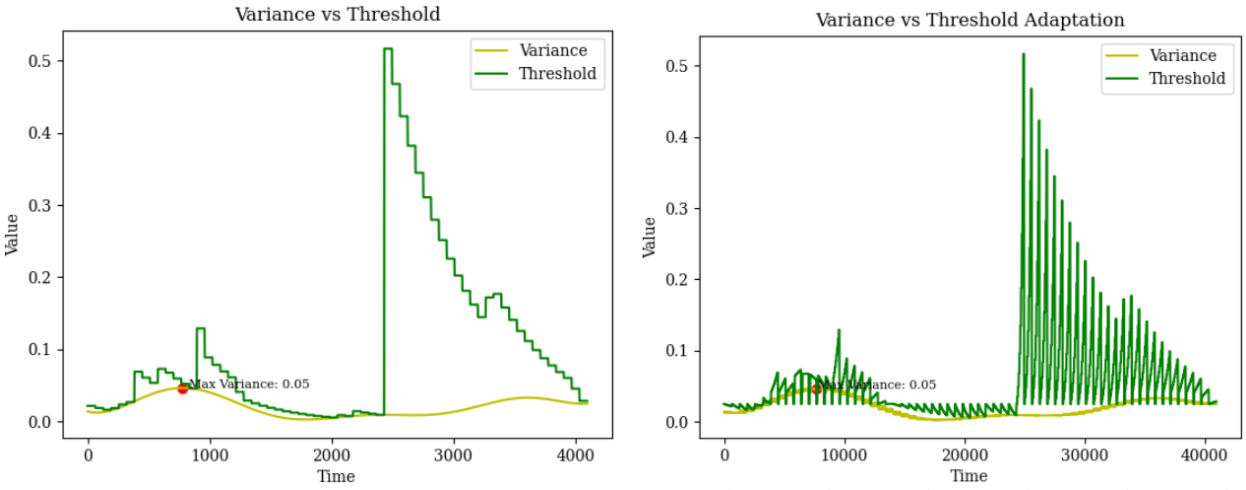


Figure 4.3: Finalized thresholds for each iteration (left) versus adaptive threshold trends for each candidate simulation (right). During the adaptive threshold loop, the threshold will oscillate by some assigned value as pictured in the right hand image. After the adaptive threshold loop is completed, a finalized threshold is used to determine which simulations should be run. The trend of these finalized thresholds is shown in the lefthand image. The above metrics were computed from the ratio method with an threshold initialization of 0.05.

case, it is necessary to either increase the number of iterations in M or increase ch_i and ch_d respectively.

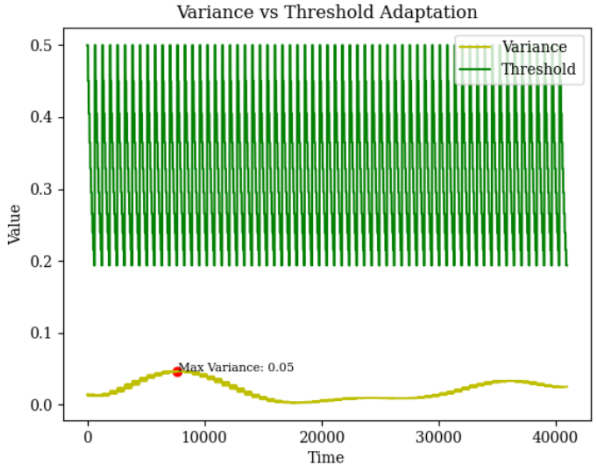
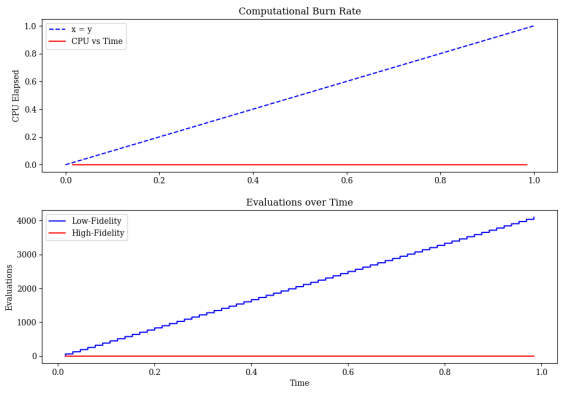
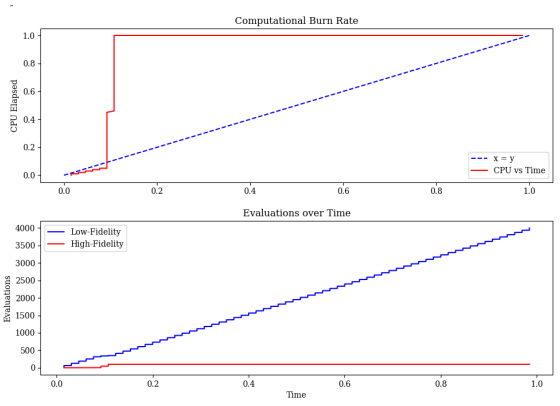


Figure 4.4: Adaptive threshold oscillation fails to fine-tune threshold. The above metrics are computed from a Percent model with an initial threshold of 0.5. The threshold is adapted over 10 adaptive threshold iterations, but is unable to decrease quickly enough in the allotted threshold iterations. As a result, every batch simulation is run with a threshold of 0.2, the lowest value reached during the threshold iteration.

With these challenges in mind we designed three general strategies for selecting t_0, ch_i



(a) Baseline Method ($t_0 = 0.5$)



(b) Baseline method ($t_0 = 0.025$).

Figure 4.5: Pitfalls of control methods. Abbreviations t_0 = Initial Threshold.

and ch_d. Our first strategy involves assigning $t_0 = 1$, $ch_i = 0.25$, and $ch_d = 0.5$. This allows the model to test threshold values inclusive of both the maximum and minimum threshold values. The threshold begins at the maximum value and decreases until converging near model variance. The threshold can be adjusted accordingly. While ch_i and ch_d can be chosen arbitrarily, ch_d must be sufficiently large so that $t_0 ch_d^M$ approaches 0. Ideally, $ch_d > ch_i$ as we expect maximum variance to be below t_0.

A secondary strategy involves assigning $t_0 = 0.01$ and assigning ch_d and ch_i such that $t_0 ch_d^M$ approaches 1. This strategy works best when the variance is expected to be close to 0, otherwise we are at risk of t_m never being greater than our expected variance.

While the findings of this method were specific to constant definitions for ch_i and ch_d, they helped inform additional methods that use adaptive versions of these parameters.

4.3.2 Adaptive Methods

Rather than relying on an explicitly defined value for ch_i and ch_d, the secant and ratio method compute ch_i and ch_d based on the behavior of the model. The biggest benefit of the following methods is the lack of additional parameters to fine-tune. These adaptive methods circumvent many of the pitfalls associated with the aforementioned control methods. Namely, both the Secant and Ratio method avoid becoming indefinitely trapped above or below the variance line. This circumvents the pitfalls of either immediately exhausting the computational budget, or never consuming the budget as depicted in Figure 4.5.

Method	t_0	**MAE**	**MSE**	**ISE**	**Max Var**	**Int Var**	**Int Thresh**	**HF**
Baseline	0.05	0.2987	0.1423	0.1358	0.1496	0.0212	-	25
Percent	0.5	0.1958	0.0547	0.0535	0.0466	0.0207	0.4737	59
Secant	0.1	0.1200	0.0206	0.0205	0.0466	0.0207	0.4996	100
Ratio	0.05	**0.1076**	**0.0181**	**0.0181**	0.0466	0.0207	0.1289	100

Table 4.1: Best performing runs for each adaptive threshold method. Abbreviations: t_0 = threshold initialization, MAE = Mean Absolute Error, MSE = Mean Squared Error, ISE = Integrated Squared Error, Max Var = Maximum Variance, Int Var = Integrated Variance, Int Thresh = Integrated Threshold, HF = Number of High-fidelity simulations conducted. Integrated variance is computed by integrating the variance values of the surrogate model for each time step. Similarly, Integrated threshold values consist of the integration of finalized thresholds at each timestep. Best performing refers to runs which achieved the lowest error scores. The ratio method consistently returned the lowest error scores for Mean Absolute Error (MAE), Mean Squared Error (MSE), and Integrated Squared Error (ISE). Interestingly, this method also had the lowest integrated threshold (Int. Thresh) value. The maximum variance (Max Var) and integrated variance (Int. Var) remained constant for each of our test methods.

Secant Method

The secant method relies on computing the rate of change between two data points, namely between cr_{nm} and $cr_{n(m-1)}$. The goal is to adjust t_m such that cr_{nm} approaches 0. If $cr_{nm} = 0$, we set $t_{m+1} = t_m$. Otherwise, if $cr_{nm} - cr_{n(m-1)} = 0$, we set $t_{m+1} = 0.01$. This means that the values of ch_i and ch_d are affected by both the current discrepancy between the projected rate and the target function as well as the previous discrepancy. This results in a method where the current rate is affected by the previous rate at each iteration.

Ratio Method

Rather than retaining some memory of previous thresholds, the ratio method adjusts the current threshold by the error margin between the target burn rate and the projected computational burn rate. Although simpler, the ratio method has the benefit of exhausting the entirety of the allocated computational budget regardless of threshold initializaiton. Moreover, as evidenced by the metrics reported in Table 4.1 and Table A.1, the ratio method consistently adheres the best to the target function of the four presented methods. Adherence to the target function was measured using several error metrics.

4.3.3 Evaluation Metrics

To evaluate the effectiveness of our methods we are interested in minimizing the maximum variance computed by our surrogate model. Variance is computed for every candidate simulation during each loop of the adaptive threshold loop, and again during the final model evaluation cycle.

We utilize Mean Average Error, Mean Squared Error, and Integrated Squared Error to determine how closely our budget consumption rate adheres to the target function. Our target function is determined by the rate at which our real-time budget, N, is consumed.

4.4 RESULTS

4.4.1 Insensitivities

Interestingly, while the maximum variance and integrated variance values differed between the baseline methods and the test methods defined in were identical for each of our adaptive threshold methods. Each adaptive threshold method returned a maximum variance of 0.0466 and an integrated variance of 0.0207 regardless of threshold initialization, threshold iteration, or input parameters. However, our test methods still returned lower values for variance than our baseline method which returned a maximum variance of 0.1496 and an integrated variance of 0.0212 in all runs.

Similarly, our error metrics were also insensitive to the model used to run our methods. For this purpose, we utilized the Branin model for our numerical experiments.

4.4.2 Error

Due to the insensitivity of our variance metrics, we rely on our previously defined error metrics to measure the success of each of our methods. Generally, the ratio method outperformed both the secant and percent methods, only being outperformed by the secant method when the threshold initialization was set to 0.75. The ratio method also consistently exhausted the entirety of the computational budget and would thus, be ideal for experiments that which to exhaust the budget. Secant performance was similar to the ratio method, but refrains from oversampling the model after threshold initialization is above 0.5. This method might be a good consideration for frugality. The percent method consistently had the largest error out of our test methods and requires additional runtime to fine tune the ch_i and ch_d parameters.

ch_i	ch_d	t_0	**MAE**	**MSE**	**ISE**	**Int Thresh**	**HF**
0.25	0.5	0.01	0.2629	0.0980	0.09719	0.0724	100
		0.25	0.2411	0.0903	0.8984	0.9690	100
		0.5	0.2206	0.0755	0.0752	0.9229	100
		0.75	0.2383	0.0893	0.0889	0.9690	100
		1	0.2629	0.0980	0.0972	0.9690	100
0.5	0.25	0.01	0.2408	0.0902	0.0899	0.4054	100
		0.25	0.2024	0.0684	0.0685	0.8007	100
		0.5	0.1958	0.0547	0.0535	0.4737	59
		0.75	0.3334	0.3206	0.0466	0.0563	0
		1	0.3334	0.3206	0.0466	0.7508	0

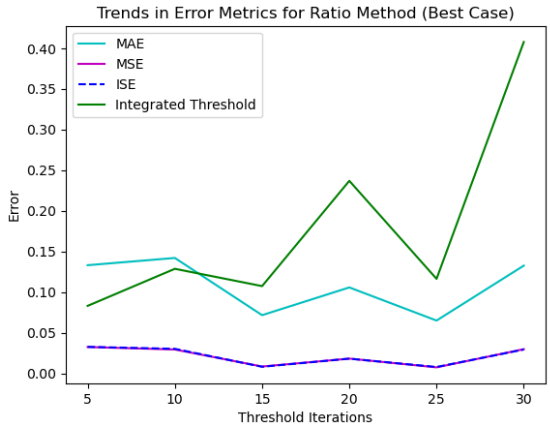
Table 4.2: Differences in Percent method performance based on magnitude of threshold adaptation. Abbreviations: ch_i = percentage value increase for adaptive threshold, ch_d = percentage value decrease for adaptive threshold step, t_0 = threshold initialization, MAE = Mean Absolute Error, MSE = Mean Squared Error, ISE = Integrated Squared Error, Max Var = Maximum Variance, Int Var = Integrated Variance, Int Thresh = Integrated Threshold, HF = Number of High-fidelity simulations conducted. Despite the first section of Percent method runs utilizing the entirety of the computational budget, swapping the ch_i and ch_d values in subsequent runs, the second row of the table, results in higher error values. Moreover, as t_0 increases, the model loses the ability to perform high-fidelity evaluations.

Each of our test methods out performed the baseline method, however the Ratio method performed the most consistently and adhered to the target function the best out of our test methods. Threshold initialization did not impact variance for any of our runs, but appears to impact computational burn rate.

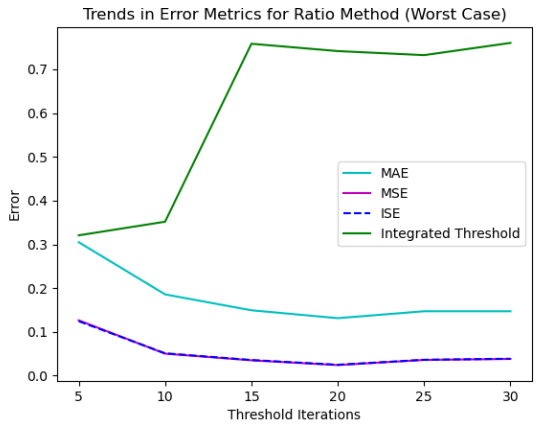
4.4.3 Threshold Initialization

While threshold initialization did not impact model variance, varying t_0 did have an impact on several error metrics. However, the more marked affect of altering t_0 was in how in affected the number of high-fidelity evaluations. Generally, the greater the distance between t_0 and the model's variance, the fewer high-fidelity evaluations would occur. This is due to the issue described in Section 4.3.1 where-in the threshold is not able to adjust quickly enough to account for the distance between t_0 and model variance.

This issue is most evident in Baseline and Percent methods as showcased in Figure 4.5, and Table 4.2. As a result, the number of high-fidelity evaluations executed during our Percent method runs varied greatly depending on the combination of t_0, ch_i and ch_d. In contrast, both the Secant and Ratio methods performed more consistently relative to the Percent method regardless of t_0. That being said, varying t_0 still created noticeable changes



(a) Ratio method ($t_0 = 0.05$)



(b) Ratio method ($t_0 = 1$)

Figure 4.6: Error trends for best and worst case ratio methods based on number of threshold iterations.

in our error metrics for both methods. Notably, higher t_0 values generally decrease the number of high-fidelity evaluations a model is able to run within our allotted budget, with the exception of the ratio method which consumed the entirety of its budget regardless of t_0.

4.4.4 Threshold Iteration

The second main facet that affected high-fidelity evaluations and error metrics was the number of threshold iterations indicated in the adaptive threshold loop. Generally, adding additional iterations to the adaptive threshold loop as illustrated in Figure A.1 is expected to decrease our computed error metrics, and decrease the integrated average of the threshold as the adaptive threshold has more time to converge. However, the relationship is not one to one, as a comparison of the error metrics for different threshold iteration ranges can be seen for the ratio method in Figure 4.6 having diminishing returns from additional threshold iterations.

Generally, increasing threshold iterations to 15 could slightly decrease error or increase the maximum number of high-fidelity evaluations for under-performing models, but threshold iterations were generally less influential than altering c_i, c_d, and t_0 values. Moreover, the benefits of increasing iteration time tended to become less obvious after threshold iterations were increased past 20.

4.5 SUMMARY

In this chapter, we presented an adaptive computing outer loop implementation for training surrogate models for single fidelity Kinetic Monte Carlo simulations. Our implementation consisted of three major contributions. First, we created an adaptive computing framework that can be used for training both multi-fidelity and single-fidelity surrogate models. Moreover, we introduced a method for adjusting the frequency of high-fidelity model evaluations by defining a target computational burn rate. Lastly, we analyzed the impacts of several methods for adjusting the aforementioned threshold parameter on computational burn rates and model variance for single fidelity surrogate models.

These simulation results revealed four key points. First, single fidelity surrogate models with adaptive thresholds consistently have lower model variance than those with constant thresholds. Secondly, altering threshold initialization did not impact variance for either of our models, but did impact computational burn rates and the number of high-fidelity evaluations made over the course of the simulated run. Thirdly, changing the target function used to simulate our high-fidelity model did not have an impact on variance or computational burn rate. Lastly, of our established methods, the ratio method adhered to our target computational burn rate the best throughout all experiments.

In future work, we aim to extend this implementation to multi-fidelity surrogate models, practical applications, and compare the effectiveness of our implementations to current implementations of multi-fidelity models.

CHAPTER 5: CONCLUSION

5.1 SUMMARY

In the previous chapters we explored the applications of adaptive computing for multi-fidelity and multi-scale modeling. Specifically, we aimed to address challenges associated with scale-up by optimizing computational budgets and model queries for both multi-fidelity and single-fidelity models.

5.1.1 Multi-fidelity Surrogate Models

Multi-fidelity surrogate models provide a means for training an optimization model based off of low-fidelity and high-fidelity models in the interest of reducing computational load. In Chapter 3 we explore how modifying sampling strategies for selecting low-fidelity model samples impact a multi-fidelity surrogate model's ability to predict the global minima of a high-fidelity model. We introduce four sampling strategies, and determine an iteration-based sampling method to be the most effective at predicting global minima for one-dimensional models. In short, our surrogate models were most accurate when exclusively trained on low-fidelity samples for the first half of the training period before sampling from the high-fidelity model.

We attempted to replicate these results on a 2-dimensional Branin model but found the experiments to be less conclusive. Generally, combinations of sampling strategies achieved higher accuracy than their single strategy counterparts.

5.1.2 Single-fidelity Surrogate Models

In the subsequent chapter, we analyze how adaptive computing can be used to manipulate the computational burn rate of single-fidelity multi-scale simulations. We present and implement a pipeline for training a surrogate model off a single-fidelity simulation. Further, we introduce an adaptive threshold variable that allows the model to adapt its computational burn rate based on the model's variance. We introduced four methods for manipulating the adaptive threshold parameter and compare the adaptive threshold to a constant threshold control. We found that, for a given target burn rate, methods which adapted the threshold parameters based on the disparity between projected and target computational burn rates had the greatest success at adhering to the target burn rate. However, model variance was generally insensitive to either of our proposed test methods.

5.2 FUTURE WORK AND DISCUSSION

The previous chapters explored potential solutions to scale-up challenges that leverage adaptive computing frameworks for multi-fidelity and single-fidelity simulations. We proposed implementations for strategic sampling and adaptive computational burn rates in the interest of optimizing works of the previous chapters rely on test functions and should be regarded as a proof of concept. Future directions for adaptive computing would involve leveraging the aforementioned techniques and pipelines for practical high-fidelity models and simulations. Moreover, many of the aspects explored in Chapter 4 can be extended to multi-fidelity simulations.

The contributions of the previous chapters highlight how adaptive computing pipelines can reduce the computational expenses associated with physics-based simulations. These computational savings can be extended to virtually any model to improve surrogate model accuracy and efficiency.

REFERENCES

[1] S. Alarie, C. Audet, A. E. Gheribi, M. Kokkolaras, and S. Le Digabel, "Two decades of blackbox optimization applications," *EURO Journal on Computational Optimization*, vol. 9, p. 100011, Jan 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2192440621001386

[2] G. E. P. Box, "Science and statistics," *Journal of the American Statistical Association*, vol. 71, no. 356, pp. 791–799, Dec. 1976.

[3] 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, p. 455–492, 1998. [Online]. Available: http://link.springer.com/10.1023/A:1008306431147

[4] S. Shan and G. G. Wang, "Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions," *Structural and Multidisciplinary Optimization*, vol. 41, no. 2, pp. 219–241, Mar. 2010. [Online]. Available: http://link.springer.com/10.1007/s00158-009-0420-2

[5] M. G. Fernández-Godino, "Review of multi-fidelity models," 2023.

[6] H. Egan, K. P. Griffin, M. T. H. de Frahan, J. Mueller, D. Vaidhynatha, D. Wald, R. Chintala, O. A. Doronina, R. King, J. Sanyal, and M. Day, "Adaptive computing for scale-up problems," 2024.

[7] "World's population increasingly urban with more than half living in urban areas — UN DESA — United Nations Department of Economic and Social Affairs — un.org," https://www.un.org/development/desa/en/news/population/world-urbanization-prospects.html#:~:text=The%20urban%20population%20of%20the,Caribbean%20with%2013%20per%20cent., [Accessed 06-06-2024].

[8] U. E. I. Administration, "International energy outlook 2017," https://www.eia.gov/outlooks/ieo/pdf/0484(2017).pdf, 2017, [Accessed 06-06-2024].

[9] E. Kamel, "A Systematic Literature Review of Physics-Based Urban Building Energy Modeling (UBEM) Tools, Data Sources, and Challenges for Energy Conservation," *Energies*, vol. 15, no. 22, p. 8649, Nov. 2022. [Online]. Available: https://www.mdpi.com/1996-1073/15/22/8649

[10] M. Day and D. Vaidhynathan, "Adaptive computing and multi-fidelity strategies for control, design and scale-up of renewable energy applications," 2023.

[11] K. Griffin, H. Egan, M. Day, J. Domantay, D. Abdulah, D. M. Febba, D. Wald, M. Henry de Frahan, and O. Doronina, "Adaptive computing," https://github.com/NREL/AdaptiveComputing, 2023.

[12] Y. Li, Z. O'Neill, L. Zhang, J. Chen, P. Im, and J. DeGraw, "Grey-box modeling and application for building energy simulations - a critical review," *Renewable and Sustainable Energy Reviews*, vol. 146, p. 111174, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032121004639

[13] Y. Chen, M. Guo, Z. Chen, Z. Chen, and Y. Ji, "Physical energy and data-driven models in building energy prediction: A review," *Energy Reports*, vol. 8, pp. 2656–2671, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352484722001615

[14] N. Sezer, H. Yoonus, D. Zhan, L. L. Wang, I. G. Hassan, and M. A. Rahman, "Urban microclimate and building energy models: A review of the latest progress in coupling strategies," *Renewable and Sustainable Energy Reviews*, vol. 184, p. 113577, Sep. 2023. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1364032123004343

[15] B. Blocken, "Computational fluid dynamics for urban physics: Importance, scales, possibilities, limitations and ten tips and tricks towards accurate and reliable simulations," *Building and Environment*, vol. 91, p. 219 – 245, 2015, cited by: 746; All Open Access, Green Open Access. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84930180900&doi=10.1016%2fj.buildenv.2015.02.015&partnerID=40&md5=c7888dadbdd3030f646daea8da80c456

[16] "Energyplus™, version 00," 9 2017. [Online]. Available: https://www.osti.gov//servlets/purl/1395882

[17] Y. Balali, A. Chong, A. Busch, and S. O'Keefe, "Energy modelling and control of building heating and cooling systems with data-driven and hybrid models—A review," *Renewable and Sustainable Energy Reviews*, vol. 183, p. 113496, Sep. 2023. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1364032123003532

[18] X. Li, Y. Zhou, S. Yu, G. Jia, H. Li, and W. Li, "Urban heat island impacts on building energy consumption: A review of approaches and findings," *Energy*, vol. 174, pp. 407–419, May 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0360544219303895

[19] S. G. Yalew, M. T. H. van Vliet, D. E. H. J. Gernaat, F. Ludwig, A. Miara, C. Park, E. Byers, E. D. Cian, F. Piontek, G. Iyer, J. Mouratiadou, J. Glynn, M. Hejazi, O. Dessens, P. Rochedo, R. Pietzcker, R. Schaeffer, S. Fujimori, S. Dasgupta, S. Mima, S. R. S. da Silva, V. Chaturvedi, R. Vautard, and D. P. van Vurren, "Impacts of climate change on energy systems in global and regional scenarios," *Nat Energy*, vol. 5, pp. 794–802, 2020.

[20] O. T. Ogunsola and L. Song, "Review and Evaluation of Using R-C Thermal Modeling of Cooling Load Prediction for HVAC System Control Purpose," in *Volume 7: Fluids and Heat Transfer, Parts A, B, C, and D.* Houston, Texas, USA: American Society of Mechanical Engineers, Nov. 2012. [Online]. Available: https://asmedigitalcollection.asme.org/IMECE/proceedings/IMECE2012/45233/735/364742 pp. 735–743.

[21] E. Cramer, "Using approximate models for engineering design," in *7th AIAA/USAF/-NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. St. Louis,MO,U.S.A.: American Institute of Aeronautics and Astronautics, Sep. 1998. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.1998-4716

[22] G. J. Eckard, I. H. Rettie, M. J. Healy, and J. S. Kowalik, *Preliminary airplane design aided by mathematical optimization*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1979, pp. 176–188. [Online]. Available: https://doi.org/10.1007/BFb0120863

[23] M. J. Healy, J. S. Kowalik, and J. W. Ramsay, "Airplane engine selection by optimization on surface fit approximations," *Journal of Aircraft*, vol. 12, no. 7, pp. 593–599, July 1975. [Online]. Available: https://arc.aiaa.org/doi/10.2514/3.44477

[24] M. G. Fernández-Godino, "Review of multi-fidelity models," 2016, publisher: arXiv Version Number: 4. [Online]. Available: https://arxiv.org/abs/1609.07196

[25] A. I. Forrester, A. Sóbester, and A. J. Keane, "Multi-fidelity optimization via surrogate modelling," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 463, no. 2088, pp. 3251–3269, Dec. 2007. [Online]. Available: https://royalsocietypublishing.org/doi/10.1098/rspa.2007.1900

[26] A. Bhosekar and M. Ierapetritou, "Advances in surrogate based modeling, feasibility analysis, and optimization: A review," *Computers Chemical Engineering*, vol. 108, pp. 250–267, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0098135417303228

[27] R. E. Edwards, J. New, L. E. Parker, B. Cui, and J. Dong, "Constructing large scale surrogate models from big data and artificial intelligence," *Applied Energy*, vol. 202, pp. 685–699, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261917307043

[28] A. C. Megri and F. Haghighat, "Zonal modeling for simulating indoor environment of buildings: Review, recent developments, and applications," *HVAC and R Research*, vol. 13, no. 6, p. 887 – 905, 2007, cited by: 129. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-36549038648&doi=10.1080%2f10789669.2007.10391461&partnerID=40&md5=bc375770121f5669bcbaca2599cc9577

[29] Z. Qian, C. C. Seepersad, V. R. Joseph, J. K. Allen, and C. F. Jeff Wu, "Building Surrogate Models Based on Detailed and Approximate Simulations," *Journal of Mechanical Design*, vol. 128, no. 4, pp. 668–677, 06 2005. [Online]. Available: https://doi.org/10.1115/1.2179459

[30] K.-D. Lee and K.-Y. Kim, "Shape optimization of a fan-shaped hole to enhance film-cooling effectiveness," *International Journal of Heat and Mass Transfer*, vol. 53, no. 15-16, p. 2996 – 3005, 2010, cited by: 179. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-77953286438&doi=10.1016%2fj.ijheatmasstransfer.2010.03.032&partnerID=40&md5=2ae2746f42d47bd626a2974ebf21cc28

[31] S. Razavi, B. A. Tolson, and D. H. Burn, "Review of surrogate modeling in water resources," *Water Resources Research*, vol. 48, no. 7, 2012, cited by: 674; All Open Access, Bronze Open Access. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84864535862&doi=10.1029%2f2011WR011527&partnerID=40&md5=2749b1cfb2ed96fc0d9bf23f44c717cd

[32] A. I. Forrester, A. Sóbester, and A. J. Keane, "Multi-fidelity optimization via surrogate modelling," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 463, no. 2088, p. 3251–3269, Dec 2007. [Online]. Available: https://royalsocietypublishing.org/doi/10.1098/rspa.2007.1900

[33] M. D. McKay, "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. 239–45, 1979.

[34] L. Brevault, M. Balesdent, and A. Hebbal, "Overview of gaussian process based multi-fidelity techniques with variable relationship between fidelities, application to aerospace systems," *Aerospace Science and Technology*, vol. 107, p. 106339, 2020.

[35] J. Wang, "An intuitive tutorial to gaussian processes regression," 2020. [Online]. Available: https://arxiv.org/abs/2009.10862

[36] M. A. Bouhlel, J. T. Hwang, N. Bartoli, R. Lafage, J. Morlier, and J. R. R. A. Martins, "A python surrogate modeling framework with derivatives," *Advances in Engineering Software*, p. 102662, 2019.

[37] D. J. J. Toal, "Some considerations regarding the use of multi-fidelity kriging in the construction of surrogate models," *Structural and Multidisciplinary Optimization*, vol. 51, no. 6, p. 1223–1245, Jun 2015. [Online]. Available: http://link.springer.com/10.1007/s00158-014-1209-5

[38] L. Mainini, A. Serani, M. P. Rumpfkeil, E. Minisci, D. Quagliarella, H. Pehlivan, S. Yildiz, S. Ficini, R. Pellegrini, F. Di Fiore, D. Bryson, M. Nikbay, M. Diez, and P. Beran, "Analytical benchmark problems for multifidelity optimization methods," Apr 2022. [Online]. Available: http://arxiv.org/abs/2204.07867

[39] J. Mueller, "Online supplement: An algorithmic framework for the optimization of computationally expensive bi-fidelity black-bos problems."

[40] M. H. de Frahan, E. Young, H. Sitaraman, and R. Larsen, "Amar: Adaptive mesh and algorithm refinement modeling — computational science — nrel," accessed on October 17, 2024. [Online]. Available: https://www.nrel.gov/computational-science/adaptive-mesh-algorithm-refinement-modeling.html

[41] H. H. Rosenbrock, "An Automatic Method for Finding the Greatest or Least Value of a Function," *The Computer Journal*, vol. 3, no. 3, pp. 175–184, Mar. 1960. [Online]. Available: https://academic.oup.com/comjnl/article-lookup/doi/10.1093/comjnl/3.3.175

APPENDIX A: ADDITIONAL FIGURES

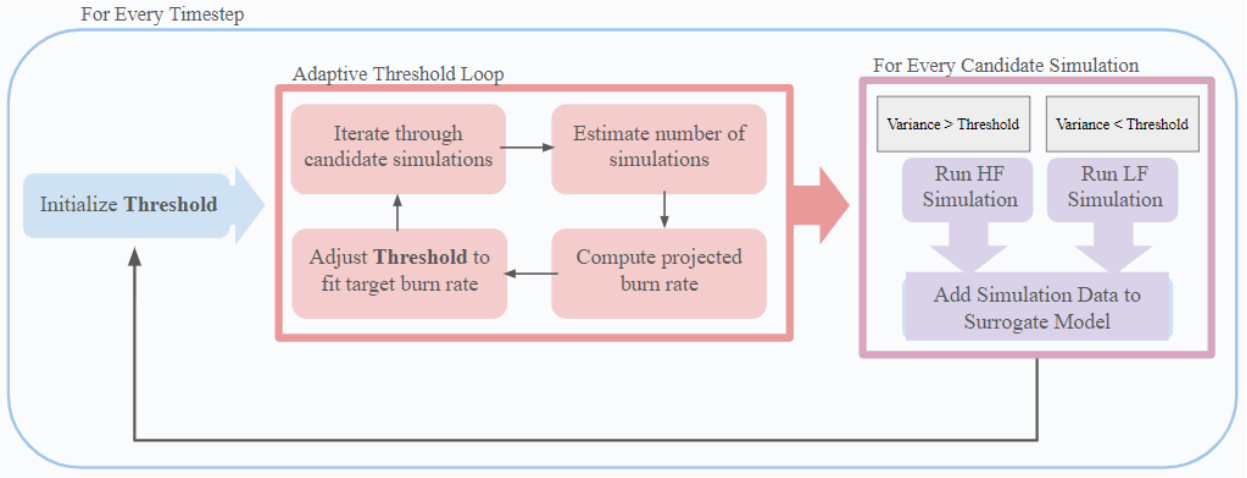


Figure A.1: Adaptive Parameters for Adaptive Computing Pipeline.

Method	t_0	MAE	MSE	ISE	HF
Baseline	0.025	0.4099	0.2433	0.2401	100
	0.05	**0.2987**	**0.1423**	**0.1358**	**25**
	0.1	0.3994	0.2352	0.2255	11
	0.25	0.5000	0.3334	0.3206	0
	0.5	0.5000	0.3334	0.3206	0
	0.75	0.5000	0.3334	0.3206	0
	1	0.5000	0.3334	0.3206	0
Secant	0.025	0.1927	0.0641	0.0638	100
	0.05	0.1994	0.0712	0.0710	100
	0.1	**0.1200**	**0.0206**	**0.0205**	**100**
	0.25	0.1729	0.0424	0.0433	100
	0.5	0.1670	0.0378	0.0372	64
	0.75	0.1670	0.0378	0.0372	64
	1	0.1670	0.0378	0.0372	64
Ratio	0.025	0.1213	0.0265	0.0261	100
	0.05	**0.1076**	**0.0181**	**0.0181**	**100**
	0.1	0.1476	0.0345	0.0352	100
	0.25	0.1527	0.0375	0.0381	100
	0.5	0.1422	0.0296	0.0304	100
	0.75	0.1712	0.0440	0.0448	100
	1	0.1857	0.0504	0.0512	100
Percent	0.025	0.2629	0.0980	0.0972	100
	0.05	0.2383	0.0893	0.0889	100
	0.1	0.2411	0.0903	0.0889	100
	0.25	0.2206	0.0903	0.0898	100
	0.5	**0.2206**	**0.0755**	**0.0752**	**100**
	0.75	0.2383	0.0893	0.0889	100
	1	0.2629	0.0980	0.0972	100

Table A.1: Error metrics for adaptive computing methods for 10 threshold iterations. Abbreviations: MAE = Mean Absolute Error, MSE = Mean Squared Error, ISE = Integrated Squared Error, HF = Number of high-fidelity simulations conducted. The percent method assigns $c_i = 0.25$ and $c_d = 0.5$.