An Automatic, Multi-Fidelity Framework for Optimizing the Performance of Super-Cavitating Hydrofoils Using Gaussian Process Regression and Bayesian Optimization

by

Benjamin W. Parker

B.A. Mathematics, Northwest University (2007)
M.S. Applied Mathematics, University of Washington (2010)
M.S. Systems Engineering, Colorado State University (2014)

Submitted to the Department of Mechanical Engineering in partial fulfillment of the requirements for the degrees of Naval Engineer and Master of Science in Mechanical Engineering at the MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018

© Massachusetts Institute of Technology 2018. All rights reserved.

DISTRIBUTION A. Approved for public release: distribution unlimited.

Signature redacted

Acceptance signature

Rohan Abeyaratne
Chairman, Department Committee on Graduate Theses
An Automatic, Multi-Fidelity Framework for Optimizing the Performance of Super-Cavitating Hydrofoils Using Gaussian Process Regression and Bayesian Optimization

by

Benjamin W. Parker

Submitted to the Department of Mechanical Engineering on May 11, 2018, in partial fulfillment of the requirements for the degrees of Naval Engineer and Master of Science in Mechanical Engineering

Abstract

Computer automated design of complex physical systems is often limited by the computational resources required for the high precision solvers. Determining an optimum design necessitates high accuracy simulations due to the multi-dimensional design space and the interconnectedness of the constraint and objective quantities. This paper will present an automated framework for iterating through a design loop that includes both physics-based computer simulations and surrogate model training using machine learning techniques. To alleviate the computation burden and efficiently explore the design space, a surrogate model for each quantity of interest that cannot be found deterministically will be utilized. Further reduction of the computational cost is accomplished by utilizing both low- and high-fidelity data to build the response surfaces. These response surface models will be trained using multi-fidelity Gaussian process regression. The models will be iteratively improved using Bayesian optimization and additional high-fidelity simulations that are automatically initiated within the design loop. In addition, Bayesian optimization will be used to automatically determine the optimum kernel for the Gaussian regression model. The feasibility of this framework is demonstrated by designing a 2D super-cavitating hydrofoil and comparing the optimum shape found with a known benchmark design. This automated multi-fidelity Bayesian optimization framework can aid in taking the human out of the design loop, thus freeing manpower resources and removing potential human bias.

Thesis Supervisor: Michael Triantafyllou
Title: Professor
Acknowledgments

I am very grateful to Luca Bonfiglio and Paris Perdikaris from MIT’s Sea Grant for being encouraging, understanding, and approachable. This thesis would not have been possible without their teaching and guidance. I am also thankful for Professor George Em Karniadakis from Brown University for his big-picture ideas and direction, and for my advisor Prof. Michael Triantafyllo for his flexibility and patience. While working on this thesis, I concurrently worked on a yearlong, ship design project with two other U.S. Navy officers, Adam Campbell and Jordan Fouquette. I am appreciative for their help and expertise with the project, and for them allowing me to incorporate machine learning techniques into our project. I am thankful for the other officers in the MIT 2N program for their support and comradery through the last three years. Finally, I want to thank my family for their encouragement, support, and prayers: my wife Megan, my four boys Jonathan, Thaddeus, Zachariah, and Isaiah, my parents Lonnie and Joy Parker, my mother- and father-in-law Alvin and Penny Timm, and my church family at Genesis Community Church in Woburn, MA.
Contents

1 Introduction 15

2 Modeling and Optimization Methods 19
  2.1 Modeling: Gaussian Process Regression .......................... 19
    2.1.1 Gaussian Processes ......................................... 20
    2.1.2 Prediction .................................................. 21
    2.1.3 Covariance Functions ...................................... 26
    2.1.4 Optimizing Hyperparameters ................................. 38
    2.1.5 Multi-Fidelity Regression ................................. 40
    2.1.6 Kernel Selection ........................................... 43
  2.2 Optimizing: Bayesian Optimization ............................... 45
    2.2.1 Expected Improvement Acquisition Function ................ 46
    2.2.2 Constrained Expected Improvement Acquisition Function ... 49

3 Design Examples 51
  3.1 Single-Fidelity and Manual Design Example: Large Surface Combatant Project ........................................ 51
    3.1.1 Project Background ........................................ 51
    3.1.2 Design Analysis ........................................... 54
  3.2 Multi-Fidelity and Automatic Design Example: 2D Super-Cavitating Hydrofoil Project .................................... 70
    3.2.1 Project Background ........................................ 71
    3.2.2 Design Analysis ........................................... 74
3.3 Multi-Fidelity and Manual Design Example: Super-Cavitating Hydrofoil Optimization with Composite Kernel

4 Conclusion

A Figures
# List of Figures

1-1 1-D Example of Surrogate Modeling. In this example of curve fitting using least squares polynomial fit there are three different orders of the polynomial: linear, quadratic, and cubic. ........................................... 16

1-2 2-D Example of Surrogate Modeling .................................................. 17

2-1 1-D Example of Prior and Posterior Functions. In the upper panel there are 20 random functions from a GP prior. In the lower panel there are 20 random functions from the posterior and four noiseless observation points in blue. .................................................. 23

2-2 1-D Example of Prior mean and variance. This figure shows the zero mean prior and two standard deviations above and below the mean. ........................................... 24

2-3 1-D Example of Posterior mean and variance. This figure shows the predicted mean, two standard deviations above and below the mean, and four noiseless observation points in blue. ........................................... 24

2-4 Example of three RBF covariance functions. ................................. 28

2-5 Sample of Priors each made with an RBF kernel, but with different hyperparameters. ........................................... 29

2-6 Example of four RQ covariance functions. ................................. 30

2-7 Sample of Priors each made with an RQ covariance function, but with different hyperparameters. ........................................... 31

2-8 Example of two LIN covariance functions. ................................. 32

2-9 Sample of Priors each made with a LIN covariance function, but with different hyperparameters. ........................................... 33
2-10 Example of three PER covariance functions. .......................... 34
2-11 Sample of Priors each made with a PER covariance function but with
different hyperparameters. .................................................. 35
2-12 Example of three Constant covariance functions. .................... 36
2-13 Sample of Priors each made with a Constant covariance function but
with different hyperparameters. ............................................ 37
2-14 Composite kernel search prediction performance.[5] ................. 45
2-15 1-D Example of Bayesian Optimization ................................. 48

3-1 Analytic Hierarchy Process Overview ................................. 55
3-2 Baseline Ship Variants ...................................................... 59
3-3 Overview of RSDE [?] ....................................................... 60
3-4 1st Iteration: OMOE vs Cost ............................................. 61
3-5 1st Iteration: OMOE vs Cost with Quadrant Descriptions ........... 62
3-6 1st Iteration: OMOE vs Cost containing only cost acceptable quadrants 63
3-7 2nd Iteration: OMOE vs Cost ............................................. 64
3-8 3rd Iteration: OMOE vs Cost ............................................. 65
3-9 LSC Bayesian Optimization Design Loop .............................. 66
3-10 Initial situation prior to beginning LSC Bayesian Optimization Design
Loop .................................................................................. 67
3-11 LSC BO Iteration 1 results. Training dataset contains 300 observation.
   With 100,000 samples, the top 50 BO results had an average EI of 4.6.
   With 200 RSDE simulations, there were 10 designs that converged. .. 68
3-12 LSC BO Iteration 2 results. Training dataset contains 310 observation.
   With 100,000 samples, the top 50 BO results had an average EI of 2.7.
   With 200 RSDE simulations, there were 19 designs that converged. .. 68
3-13 LSC BO Iteration 3 results. Training dataset contains 329 observation.
   With 100,000 samples, the top 50 BO results had an average EI of 0.7.
   With 200 RSDE simulations, there were 30 designs that converged. .. 69
3-14 Final Design of Large Surface Combatant ............................ 70
3-15 Hydrofoil shapes designed for different cavitation environments. Red curves show the cavity contours and the blue lines show the flow streamlines.

3-16 Hydrofoil Design Variable, $x$. Fixed points are indicated by green boxes. Arrows indicate control point movement is allowed. Note that $x_{15}$ controls the vertical movement of three control points.[12]

3-17 Benchmark Hydrofoil Shape

3-18 Benchmark hydrofoil pressure and vapor distributions

3-19 Automatic Design Loop for 2D Super-Cavitating Hydrofoil

3-20 Phase 1 Results

3-21 Phase 1 Results: Hydrofoil Improved Shapes by Iteration

3-22 Hydrofoil pressure and vapor distributions: Iterations 4-5

3-23 Hydrofoil pressure and vapor distributions: Iteration 6

3-24 Hydrofoil pressure and vapor distributions: Iterations 7-47

3-25 Hydrofoil pressure and vapor distributions: Iterations 48-100

3-26 Phase 2 Results

3-27 Phase 2 Results: Hydrofoil Improved Shapes by Iteration

3-28 Final Results

3-29 Hydrofoil pressure and vapor distributions: Iteration 101

3-30 Hydrofoil pressure and vapor distributions: Iterations 102-105

3-31 Hydrofoil shape, pressure and vapor distributions of Final Design

A-1 1-D Example of Prior and Posterior Functions. In the upper panel are five random functions from a GP prior. In the lower panel are five random functions from the posterior and four noiseless observation points in blue.

A-2 1-D Example of Prior and Posterior Functions. In the upper panel are 20 random functions from a GP prior. In the lower panel are 20 random functions from the posterior and four noiseless observation points in blue.(Figure 2-1)
A-3  Multi-fidelity covariance matrix with three levels of fidelity: low-fidelity (level 1), medium-fidelity (level 2), and high-fidelity (level 3) 97

A-4  Multi-fidelity covariance matrix with two levels of fidelity: (1) high-fidelity, h, and low-fidelity, l. 97
List of Tables

3.1 Sponsor Requirements .................................................. 53
3.2 Design Space Variable Inputs to RSDE .............................. 57
3.3 LSC Zonal Breakdown .................................................... 58
3.4 Upper and lower bounds of design variables ..................... 76
3.5 Results from Phase 1 ..................................................... 81
3.6 Results from Phase 2 ..................................................... 85
3.7 Results using optimized composite kernel ......................... 92
Chapter 1

Introduction

This paper is intended to serve as a primer for a set of machine learning techniques applied to engineering design problems. The machine learning techniques discussed in this paper are Gaussian process (GP) regression, multi-fidelity regression, Bayesian optimization (BO), and composite kernel selection. After explaining these concepts, application examples will be presented. The climactic example will demonstrate the feasibility of using an automated engineering design loop which incorporates both physics-based solvers and machine learning techniques. As an introduction to the topics in this primer, three motivations are presented.

Motivation 1: Model-Based Optimization Engineering design problems generally include multiple design inputs and difficult to evaluate quantities of interest. The quantities of interest, which are the design objective and constraints, are determined by computer simulation or experimentation. Since both evaluation methods are often expensive, in terms of time, they often inadequately explore the design space. Thorough design space exploration often requires orders of magnitude more simulations than budgeted. An example of this budget inadequacy will be highlighted in Chapter 3. Therefore, model-based optimization is used to "fill in the gaps" in the design space between simulation results.

Model-based optimization, also referred to as surrogate modeling and response sur-
face modeling, is the method of creating models from training data that can recreate simulation results and make predictions at untested locations. The input training data is purposely chosen to span the design space and the black-box model will attempt to mimic the simulations outputs. If the design input is one dimensional, then the surrogate modeling is the process of fitting a curve to the data as shown in Figure 1-1. If the design input is two dimensional, then the surrogate modeling is the process of fitting a surface to the data as shown in Figure 1-2. Higher order design inputs are more difficult to visualize, but the modeling concept is the same. Modeling allows the designers to optimize their design by more thoroughly exploring the design space. These simple examples don’t show the model uncertainty, but this important factor will be addressed in Section 2.1.2.

Figure 1-1: 1-D Example of Surrogate Modeling. In this example of curve fitting using least squares polynomial fit there are three different orders of the polynomial: linear, quadratic, and cubic.
Motivation 2: Gaussian Process Regression  The theory behind of Gaussian process regression, also referred to as kriging, will be explained in Section 2.1, but this section will focus on the advantages of this response surface modeling technique. First, GP regression is flexible. GP modeling is capable of handling a large variety of engineering problems, including both regression and classification problems, and producing effective predictive models. Second, GP is user-friendly. Since GP is commonly used, there are many GP regression tools, training resources, and support venues. Because of the many resources available and the efficacy of GP for many engineering problems, I call GP user-friendly. Third, GP is a non-linear and non-parametric modeling paradigm. This enables more complex and better modeling solutions. Fourth, GP regression is computationally tractable. There are other regression techniques that are computationally cheaper, but GP gives advanced modeling solutions with an understood and stable computational cost.

Motivation 3: Automatic Design Loop  One of the main goals of this paper is to demonstrate the feasibility of an automatic design loop, integrating physics-based solvers and machine learning principle. There are many advantages to having an automated design loop. These advantages include (1) freeing man-power resources to
work on problems for which computers are ill-equipped, (2) decisions can be made quicker as humans and computers work simultaneously, (3) more thorough exploration of the design space is possible using surrogate models and digital twins, (4) removal of potential human bias which will increase the probability of discovering fresh design solutions and reduce the probability of inadvertently making design decisions earlier than necessary, and (5) easily allows for design bounds and constraints to be adjusted as decision makers learn in greater detail the scope of the engineering problem. There are many ways to attempt automation for an engineering design problems, but according to Shahriari et al. Bayesian optimization "promises greater automation so as to increase both product quality and human productivity" [13].
Chapter 2

Modeling and Optimization Methods

2.1 Modeling: Gaussian Process Regression

Gaussian process regression is an inductive, non-parametric method of modeling unknown functions from a finite set of training points for the purpose of making predictions. The authoritative reference for GP is *Gaussian Processes for Machine Learning* by C. E. Rasmussen and C. K. I. Williams [11]. In addition to this reference, much of Chapter 2 was written based on concepts learned through numerous discussions with Paris Perdikaris, Assistant Professor of Mechanical Engineering and Applied Mechanics at University of Pennsylvania and previously a post-doctoral researcher at MIT, and lectures available online including University of British Columbia course Computer Science 540 - Machine Learning taught by Nando de Freitas in 2013 and lectures given by Philipp Hennig given at the Machine Learning Summer School 2013, held at the Max Planck Institute for Intelligent Systems in TÃ­jibingen, Germany.

GP regression assigns an *a priori*, also referred to as prior, probability to every function within the design space. After training data is considered, the probability of each function conditioned on the training data can be determined. This is the posterior distribution of functions. However, the set of possible functions within the design space is infinite and each needs to have its probability quantified. To solve this hurdle, the function distribution is assumed to be a specific stochastic process
called a Gaussian process. A GP is a generalization of the joint Gaussian probability distribution. Stochastic processes describe random functions while probability distributions describe random variables. Therefore, a GP is a gathering of random variables which will fall under a multivariate Gaussian distribution when a finite number of the random variables are selected.

2.1.1 Gaussian Processes

Before describing regression modeling in greater detail it is important to understand the training dataset, \( D \), and the stochastic process called Gaussian process. \( D \) is the set of \( n \) observations of inputs and corresponding outputs, as shown in Equation 2.1.

\[
D = \{(x_i, y_i)| i = 1, ..., n\}
\] (2.1)

The independent input \( x \) is a vector of length \( D \), which is often called the number of design features. For an engineering design problem, \( x \) represents a design variant within the specified design space. The input design vectors are collected to form a \( D \times n \) design matrix, \( X \). The dependent output \( y \) is a scalar value and represents the engineering quantity of interest for each design variant. The output scalars are collected to form a output vector, \( y \), of length \( n \). Therefore, the training dataset is also expressed as \( D = (X, y) \).

The underlying assumption when building a surrogate model is that there exists an unknown function that relates the input vector and the output value, and this function can be learned with quantified uncertainty even in the presence of noisy data. This relationship is shown in Equation 2.2.

\[
y_i = f(x_i) + \epsilon \quad \text{for} \quad i = 1, 2, ..., n
\] (2.2)

\( f(x) \) is a GP and is specified by a mean function, \( m(x) \), and a covariance function, or kernel, \( k(x, x') \). The implication of this is that at any location \( x_i \), \( f(x_i) \) is a random
variable with a normal distribution, and for any finite set of locations $x_1, x_2, ..., x_m$, the random variables $f(x_1), f(x_2), f(x_m)$ will have a joint Gaussian distribution. The mean and covariance functions are shown in Equations 2.3 and 2.4.

$$m = \mathbb{E}[f(x)] \quad (2.3)$$

$$k(x, x') = \text{cov}(f(x), f(x')) = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))] \quad (2.4)$$

The GP $f$ is defined as shown in Equation 2.5 and shorthand notation for $f$ is shown in Equation 2.6.

$$f_i = f(x_i) \sim \mathcal{N}
\begin{pmatrix}
  m(x_{i1}) \\
  m(x_{i2}) \\
  \vdots \\
  m(x_{in})
\end{pmatrix},
\begin{pmatrix}
  k(x_{i1}, x_{i1}) & k(x_{i1}, x_{i2}) & \cdots & k(x_{i1}, x_{in}) \\
  k(x_{i2}, x_{i1}) & k(x_{i2}, x_{i2}) & \cdots & k(x_{i2}, x_{in}) \\
  \vdots & \vdots & \ddots & \vdots \\
  k(x_{in}, x_{i1}) & k(x_{in}, x_{i2}) & \cdots & k(x_{in}, x_{in})
\end{pmatrix}
\quad (2.5)$$

$$f(x) \sim \mathcal{N}(m(x), k(x, x')) \quad (2.6)$$

In Equation 2.2, $\epsilon$ is the noise in the observation data and is assumed to be an independent, identically distributed (i.i.d) Gaussian distribution with zero mean and variance $\sigma^2$.

$$\epsilon \sim \mathcal{N}(0, \sigma^2) \quad (2.7)$$

### 2.1.2 Prediction

GP regression is initialized by selecting appropriate mean and covariance functions for the prior function space. For convenience, the mean function will be set equal to zero. For the covariance function, any kernel that produces a positive semidefinite covariance matrix is valid. Covariance functions will be explained in greater detail in Sections 2.1.3 and 2.1.4, but this section will introduce their structure and application.
The covariance matrix is size $n \times n$ and is defined in Equation 2.8.

$$K = K(X, X) = \begin{bmatrix}
    k(x_1, x_1) & k(x_1, x_2) & \ldots & k(x_1, x_n) \\
    k(x_2, x_1) & k(x_2, x_2) & \ldots & k(x_2, x_n) \\
    \vdots & \vdots & \ddots & \vdots \\
    k(x_n, x_1) & k(x_n, x_2) & \ldots & k(x_n, x_n)
\end{bmatrix} \quad (2.8)$$

The mean and covariance function, and the assumed error distribution, are combined to form the GP prior for the target values, as shown in Equation 2.9.

$$y \sim \mathcal{N}(0, K + \sigma^2 I) \quad (2.9)$$

The GP prior over all possible functions is then conditioned on the training dataset to produce the posterior. An example of this idea is demonstrated in Figures 2-1, 2-2, and 2-3. In this 1-D example, a sample of the prior distribution of all possible functions within the design is shown in the top panel, and the posterior distribution, which has rejected all the functions that aren’t in agreement with the four noise-free observation points, is in the bottom panel.
Figure 2-1: 1-D Example of Prior and Posterior Functions. In the upper panel there are 20 random functions from a GP prior. In the lower panel there are 20 random functions from the posterior and four noiseless observation points in blue.
1.0
0.5-
Prior

Figure 2-2: 1-D Example of Prior mean and variance. This figure shows the zero mean prior and two standard deviations above and below the mean.

Figure 2-3: 1-D Example of Posterior mean and variance. This figure shows the predicted mean, two standard deviations above and below the mean, and four noiseless observation points in blue.

While visually removing functions that aren't in accordance with the data is a useful tool to understand the methodology, in practice probabilistic analysis is actually used.
Since a GP prior is combined with Gaussian noise, the posterior is also a GP and can be analytically determined and prediction at the test point \( x_* \) is possible. The joint distribution of the training data and the test data is shown in Equation 2.10.

\[
\begin{bmatrix}
  y \\
  f(x_*)
\end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix}
  K + \sigma^2 I & K(X, x_*) \\
  K(x_*, X) & K(x_*, x_*)
\end{bmatrix}\right)
\] (2.10)

The expanded covariance matrix is a composed of blocks: (1) the original covariance matrix of the training data, \( K \), which is size \( n \times n \), (2) two covariance vectors between the training data inputs and the testing data inputs, \( K(X, x_*) \) and \( K(x_*, X) \), both of which are length \( n \), and (3) the kernel function evaluated at the test point, \( k(x_*, x_*) \), which is a scalar. To simplify the notation, the vector \( K(X, x_*) \) will be written as \( k_* \), the vector \( K(x_*, X) \) as \( k_*^T \), the scalar \( K(x_*, x_*) \) as \( k_* \), and the unknown random function \( f(x_*) \) as \( f_* \). Therefore, Equation 2.10 can be rewritten as shown in Equation 2.11.

\[
\begin{bmatrix}
  y \\
  f_*
\end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix}
  K + \sigma^2 I & k_* \\
  k_*^T & k_*
\end{bmatrix}\right)
\] (2.11)

The predictive distribution is conditioned on \( x_* \) as well as \( X \) and \( y \), which is a straightforward operation for Gaussian processes. Therefore, the GP fundamental predictive equation is shown in Equation 2.12 and is defined by a mean and variance function which are shown in Equations 2.13, and 2.14.

\[
f_*|\mathcal{D}, x_* \sim \mathcal{N}(\mathbb{E}[f_*], \mathbb{V}(f_*))
\] (2.12)

\[
\mathbb{E}[f_*] = k_*^T(K + \sigma^2 I)^{-1}y
\] (2.13)

\[
\mathbb{V}(f_*) = k_* - k_*^T(K + \sigma^2 I)^{-1}k_*
\] (2.14)

The predicted quantity of interest value with input \( x_* \) is found by evaluating Equation 2.13 and the variance at the test point is found by evaluating Equation 2.14.
2.1.3 Covariance Functions

At the end Section 2.1.2 a predictive GP regression model was presented. The two critical components of the model are the covariance function and the hyperparameters of the covariance function. This section will discuss covariance function properties, show examples of common kernels and associated GP priors, and exhibit how varying hyperparameters affect the shape of the covariance functions and GP priors. Section 2.1.4 will discuss the process of hyperparameter optimization.

The covariance function is the function used to create the covariance matrix. The output value of a covariance function represents the "similarity" between the two inputs. When choosing a covariance function, you are immediately embedding assumptions into the model. Ideally, the kernel selection is based on known behaviors within the design problem. However, kernels are often chosen based on convention, tribal knowledge, or convenience. Diagnostic methods exist to search for better kernels based on the design problem and data set. One of these methods is presented in Section 2.1.6 and demonstrated in Section 3.3.

Not any arbitrary function will produce an appropriate covariance matrix. The covariance function must be produce a covariance matrix that is positive semidefinite and symmetric. Therefore, instead of trying to construct a function from scratch, there are a number frequently used valid kernels. The common kernels that are introduced in this paper are the Radial Basis Function (RBD), the Rational Quadratic (RQ) kernel, the Periodic (PER) kernel, the Linear (LIN) kernel, and the Constant kernel. Since there is such flexibility with kernels, they are defined in subtly different ways throughout the literature on the subject. The kernels described in this primer are based on the kernel descriptions in Structure Discovery in Nonparametric Regression through Compositional Kernel Search by Duvenaud et al. [5], Gaussian Processes for Machine Learning by C. E. Rasmussen and C. K. I. Williams [11], and the documentation for the machine learning in Python module scikit-learn. While
highlighting these kernels, hyperparameter effects and kernel properties will also be explained.

**Radial Basis Function Kernel**

The Radial Basis Function, also known as the Square Exponential or Gaussian kernel, is the most ubiquitous kernel. This is because RBF kernels have a number of convenient properties. First, RBF kernels only have two hyperparameters, the signal variance, $\sigma^2$, and the lengthscale, $l$. The signal variance, also called the output variance, is a hyperparameter common to all kernels and magnifies (if greater than one) or reduces (if less than one) the correlation between inputs. The lengthscale is a correlation window size, where a large lengthscale indicates strong correlation even when inputs are far apart and a small lengthscale indicates weak correlation when inputs are far apart. Therefore, the magnitude of the lengthscale will determine the waviness of the prior. Second, RBF kernels are stationary. A stationary kernel means that the function value is only dependent on the distance between the two inputs, regardless of where those two inputs are located within the design space. Third, and probably most influential, RBF kernels produce priors which are infinitely differentiable and thus very smooth. Finally, since RBF kernels are popular, there is a significant amount of resources and training tools on the topic, thus reinforcing their popularity.
\[ k_{RBF}(x, x') = \sigma^2 \exp \left( -\frac{(x - x')^2}{2l^2} \right) \]  

(2.15)

Figure 2-4: Example of three RBF covariance functions.

28
Rational Quadratic Kernel

The Rational Quadratic covariance function is an infinite sum of RBFs with varying lengthscales. The hyperparameter, $\alpha$, is a scale mixture parameter which affects the shape of the function as shown in Figure 2-6. As $\alpha \to \infty$, the RQ kernel becomes the RBF kernel. Since the RQ kernel is also stationary and yields smooth, differentiable priors, it is also very popular.

$$k_{RQ}(x, x') = \sigma^2 \left( 1 + \frac{(x - x')^2}{2\alpha l^2} \right)^{-\alpha}$$  \hspace{1cm} (2.16)
Figure 2-6: Example of four RQ covariance functions.
Figure 2-7: Sample of Priors each made with an RQ covariance function, but with different hyperparameters.

**Linear Kernel**

By selecting the Linear kernel the designer is actually implementing traditional Bayesian regression. Therefore, this kernel can easily be used to train a benchmark model to compare against other models, using Negative Log Marginal Likelihood as the comparative metric (this metric is discussed in greater detail in Section 2.1.4). Also, by multiplying learner kernels together higher order polynomial kernels can be created.
This will be explained more in 2.1.3 and 2.1.6. The Linear kernel is a non-stationary kernel. If a kernel is non-stationary kernels this means that changing the location within the design space can yield different covariances even if the difference between the two inputs is the same. The non-stationary property is displayed below in Figure 2-8.

$$k_{LIN}(x, x') = \sigma_b^2 + \sigma_v^2 (x - l)(x' - l)$$

(2.17)

Figure 2-8: Example of two LIN covariance functions.
Figure 2-9: Sample of Priors each made with a LIN covariance function, but with different hyperparameters.

**Periodic Kernel**

2.18 The periodic kernel, also called the Exponential-Sine-Squared kernel, is used to model functions which repeat themselves. In addition to a signal variance and length-scale hyperparameters, there is another hyperparameter \( p \) which is the periodicity. The periodicity value determines the length between intervals and this is illustrated in Figure 2-11. The periodic kernel is also non-stationary.

\[
k_{PER}(x, x') = \sigma^2 \exp\left(-\frac{2}{l^2} \sin^2\left(\frac{\pi}{p} |x - x'|\right)\right)
\]  \hspace{1cm} (2.18)
Figure 2-10: Example of three PER covariance functions.
Figure 2-11: Sample of Priors each made with a PER covariance function but with different hyperparameters.

**Constant Kernel**

The constant kernel is a stationary kernel that gives a constant value as long as both inputs are within the design space. The usefulness of the stationary constant kernel is when it is combined with other kernels and will be more clearly explained in the next Section and in Section 2.1.6.

\[ k_{Constant}(x, x') = \sigma^2 \quad \forall x, x' \]  

(2.19)
Figure 2-12: Example of three Constant covariance functions.
Figure 2-13: Sample of Priors each made with a Constant covariance function but with different hyperparameters.

**Kernel Combinations**

One final kernel property that will be used extensively in Section 2.1.6 is the fact that the sum or product of two valid kernels is also a valid kernel. To prove this, let $f_1(x)$ and $f_2(x)$ be two independent GPs. The sum of these GPs is another GP $f_{sum}(x) = f_1(x) + f_2(x)$ and this GP has the kernel $k_{sum}(x, x') = k_1(x, x') + k_2(x, x')$. Therefore, the GP $f_{sum}(x)$ has a valid kernel which is composed by adding two other kernels. Similarly, the product of two GPs is another GP $f_{product}(x) = f_1(x) \times f_2(x)$ and this GP has the kernel $k_{product}(x, x') = k_1(x, x') \times k_2(x, x')$. Therefore, the GP
$f_{\text{product}}(x)$ has a valid kernel which is composed by multiplying two other kernels.

Therefore, there is no end to the number of or depth of the kernels available for modeling. Section 2.1.6 will highlight one approach to find kernels custom-made for each modeling problem. One extremely common example of composing a kernel from the product of other kernels is automatic relevance determination (ARD). This extremely common training method for multi-dimensional problems uses an RBF kernel for each design input and multiplies all these together to compose a single kernel and subsequent covariance matrix. The ARD kernel is shown in Equation 2.20.

$$
k_{\text{ARD}}(x, x') = \prod_{i=1}^{D} k_{\text{RBF}}(x_i, x'_i) = \left( \sigma^2 \exp \left( -\frac{(x_1 - x'_1)^2}{2l_1^2} \right) \right) \times \cdots \times \left( \sigma^2 \exp \left( -\frac{(x_D - x'_D)^2}{2l_D^2} \right) \right)
$$

$$
= \sigma^2_{\text{ARD}} \exp \left[ -\frac{1}{2} \sum_{i=1}^{D} \left( \frac{x_i - x'_i}{l_i} \right)^2 \right]
$$

ARD gets its name from the lengthscale hyperparameters, $l_i$ which automatically determine how important each feature is with regards to affecting change on the quantity of interest. When the lengthscales are relatively large, then the exponent approaches zero and that feature become irrelevant compared with the other design features. Often, the learned hyperparameters are rewritten as $\lambda_i = \frac{1}{l_i}$ so that irrelevant features will have hyperparameters near zero.

### 2.1.4 Optimizing Hyperparameters

Once a predictive model is known, it needs to be compared against other possible models. This is because kernel functions contain hyperparameters, $\theta$, which are free to change values and still form valid covariance matrices but give different predictive models. In fact, the reason that covariance function parameters are called "hyperparameters" is to highlight their value freedom and the non-parametric advantage of GP regression. Therefore, by varying hyperparameters the model can im-
proved. The metric used to compare predictive models is the Negative Log Marginal Likelihood (NLML), also called the evidence. Integrating the product of the likelihood and the prior over all prior functions, \( f \), yields the Marginal Likelihood:

\[
\text{Marginal Likelihood} = \int \text{Likelihood} \times \text{Prior}.
\]

The equation for Marginal Likelihood is shown in Equation 2.21. Here, a new notation is introduced to show which variables are dependent on the hyperparameters.

\[
p(y|X, \theta) = \int p(y|f, X, \theta)p(f|X, \theta)df
\]  

(2.21)

The prior is a GP, \( f|X, \theta \sim \mathcal{N}(0, K_\theta) \). The likelihood is also a GP, \( y|f, X, \theta \sim \mathcal{N}(f, \sigma^2 I) \). Two Gaussians multiplied together yields a new Gaussian and the Marginal Likelihood can be determined. Finally, by taking the logarithmic of both sides of the Marginal Likelihood equation and multiplying through by negative one Equation 2.22 is produced, which is used to calculate the \( \text{NLML} = -\log(p(y|X, \theta)) \).

\[
\text{NLML} = \frac{1}{2} y^T (K_\theta + \sigma^2 I)^{-1} y + \frac{1}{2} \log|K_\theta + \sigma^2 I| + \frac{n}{2} \log(2\pi)
\]  

(2.22)

The hyperparameter space is searched for hyperparameters that minimize the NLML, or, conversely, to maximize the Marginal Likelihood.

\[
\theta^* = \arg\min_\theta \text{NLML}(\theta)
\]  

(2.23)

For this paper, the search for hyperparameters was conducted using a quasi-Newton, gradient descent optimization algorithm. This method essentially goes through the following optimization loop: (1) with the current hyperparameters (or the initial hyperparameters, if the hyperparameters learning is just beginning) determine NLML using Equation 2.22, (2) calculate the direction of steepest descent of the NLML using Equation 2.24, (3) take step in this direction in the hyperparameter space, and (4) check for convergence. This method depends on the step-size, the initial value of the hyperparameters, and the convergence criteria. While this optimization technique doesn’t ensure that the global optimum will be found, the method is well understood,
efficient, and can be compared against results found with different initial hyper-
parameters and random restarts. In Equation 2.24, the matrix derivative properties
shown in Equations 2.25 and 2.26 were used and the following notation was applied,
\( K_n(\theta) = K_\theta + \sigma^2 I \).

\[
\frac{\partial}{\partial \theta_i} NLML(\theta) = -\frac{1}{2} y^T K_n^{-1} \frac{\partial K_n(\theta)}{\partial \theta_i} K_n^{-1} y + \frac{1}{2} tr(K_n^{-1} \frac{\partial K_n(\theta)}{\partial \theta_i}) \tag{2.24}
\]

\[
\frac{\partial}{\partial \theta_i} K_n^{-1}(\theta) = -K_n^{-1} \frac{\partial K_n(\theta)}{\partial \theta_i} K_n^{-1} \tag{2.25}
\]

\[
\frac{\partial}{\partial \theta_i} |K_n(\theta)| = tr(K_n^{-1} \frac{\partial K_n(\theta)}{\partial \theta_i}) \tag{2.26}
\]

### 2.1.5 Multi-Fidelity Regression

The flexibility of Gaussian process regression allows for different fidelity levels for
the training data. A multi-fidelity dataset may be the result of different types of
physics-based computer simulations, varied mesh granularities for the simulations, or
combining simulation data with physical experiment results. In general, it is rightly
assumed that fidelity level is inversely proportional to "cost." Cost can be measured
in computational, temporal, or monetary resources. Overall, the goal is to make
high accuracy predictions while training a model with multiple fidelity levels of data.
Cheaper low-fidelity data is used to make a better model than that produced solely
with expensive high fidelity data.

There isn’t a limit on the number of fidelity levels, but this paper will illustrate
the concept using only three levels of fidelity, low-fidelity (level 1), medium-fidelity
(level 2), and high-fidelity (level 3). Additionally, the results with two levels of fidelity
are also presented at the end of this section. The multi-fidelity training dataset is
shown in Equation 2.27 with \( D_1, D_2, \) and \( D_3 \) also defined below. The number of
low-fidelity data points, \( n_1 \), is generally greater than the number of medium-fidelity

\[
2.1.5 \quad \text{Multi-Fidelity Regression}
\]

The flexibility of Gaussian process regression allows for different fidelity levels for
the training data. A multi-fidelity dataset may be the result of different types of
physics-based computer simulations, varied mesh granularities for the simulations, or
combining simulation data with physical experiment results. In general, it is rightly
assumed that fidelity level is inversely proportional to "cost." Cost can be measured
in computational, temporal, or monetary resources. Overall, the goal is to make
high accuracy predictions while training a model with multiple fidelity levels of data.
Cheaper low-fidelity data is used to make a better model than that produced solely
with expensive high fidelity data.

There isn’t a limit on the number of fidelity levels, but this paper will illustrate
the concept using only three levels of fidelity, low-fidelity (level 1), medium-fidelity
(level 2), and high-fidelity (level 3). Additionally, the results with two levels of fidelity
are also presented at the end of this section. The multi-fidelity training dataset is
shown in Equation 2.27 with \( D_1, D_2, \) and \( D_3 \) also defined below. The number of
low-fidelity data points, \( n_1 \), is generally greater than the number of medium-fidelity

\[
2.1.5 \quad \text{Multi-Fidelity Regression}
\]

The flexibility of Gaussian process regression allows for different fidelity levels for
the training data. A multi-fidelity dataset may be the result of different types of
physics-based computer simulations, varied mesh granularities for the simulations, or
combining simulation data with physical experiment results. In general, it is rightly
assumed that fidelity level is inversely proportional to "cost." Cost can be measured
in computational, temporal, or monetary resources. Overall, the goal is to make
high accuracy predictions while training a model with multiple fidelity levels of data.
Cheaper low-fidelity data is used to make a better model than that produced solely
with expensive high fidelity data.

There isn’t a limit on the number of fidelity levels, but this paper will illustrate
the concept using only three levels of fidelity, low-fidelity (level 1), medium-fidelity
(level 2), and high-fidelity (level 3). Additionally, the results with two levels of fidelity
are also presented at the end of this section. The multi-fidelity training dataset is
shown in Equation 2.27 with \( D_1, D_2, \) and \( D_3 \) also defined below. The number of
low-fidelity data points, \( n_1 \), is generally greater than the number of medium-fidelity

\[
2.1.5 \quad \text{Multi-Fidelity Regression}
\]

The flexibility of Gaussian process regression allows for different fidelity levels for
the training data. A multi-fidelity dataset may be the result of different types of
physics-based computer simulations, varied mesh granularities for the simulations, or
combining simulation data with physical experiment results. In general, it is rightly
assumed that fidelity level is inversely proportional to "cost." Cost can be measured
in computational, temporal, or monetary resources. Overall, the goal is to make
high accuracy predictions while training a model with multiple fidelity levels of data.
Cheaper low-fidelity data is used to make a better model than that produced solely
with expensive high fidelity data.

There isn’t a limit on the number of fidelity levels, but this paper will illustrate
the concept using only three levels of fidelity, low-fidelity (level 1), medium-fidelity
(level 2), and high-fidelity (level 3). Additionally, the results with two levels of fidelity
are also presented at the end of this section. The multi-fidelity training dataset is
shown in Equation 2.27 with \( D_1, D_2, \) and \( D_3 \) also defined below. The number of
low-fidelity data points, \( n_1 \), is generally greater than the number of medium-fidelity

\[
2.1.5 \quad \text{Multi-Fidelity Regression}
\]

The flexibility of Gaussian process regression allows for different fidelity levels for
the training data. A multi-fidelity dataset may be the result of different types of
physics-based computer simulations, varied mesh granularities for the simulations, or
combining simulation data with physical experiment results. In general, it is rightly
assumed that fidelity level is inversely proportional to "cost." Cost can be measured
in computational, temporal, or monetary resources. Overall, the goal is to make
high accuracy predictions while training a model with multiple fidelity levels of data.
Cheaper low-fidelity data is used to make a better model than that produced solely
with expensive high fidelity data.

There isn’t a limit on the number of fidelity levels, but this paper will illustrate
the concept using only three levels of fidelity, low-fidelity (level 1), medium-fidelity
(level 2), and high-fidelity (level 3). Additionally, the results with two levels of fidelity
are also presented at the end of this section. The multi-fidelity training dataset is
shown in Equation 2.27 with \( D_1, D_2, \) and \( D_3 \) also defined below. The number of
low-fidelity data points, \( n_1 \), is generally greater than the number of medium-fidelity
data points, $n_2$, and much greater than the number of high-fidelity data points, $n_3$.

$$D = [D_1, D_2, D_3]$$  \hspace{1cm} (2.27)

$$D_1 = \{(x_i, y_i) | i = 1, ..., n_1\}$$  \hspace{1cm} (2.28)

$$D_2 = \{(x_j, y_j) | j = 1, ..., n_2\}$$  \hspace{1cm} (2.29)

$$D_3 = \{(x_l, y_l) | l = 1, ..., n_3\}$$  \hspace{1cm} (2.30)

Like the single-fidelity GP regression, there is an assumption of the existence of an unknown function for each fidelity level that maps inputs to an output.

$$y_1 = f_1(x_1) + \epsilon_1$$  \hspace{1cm} (2.31)

$$y_2 = f_2(x_2) + \epsilon_2$$  \hspace{1cm} (2.32)

$$y_3 = f_3(x_3) + \epsilon_3$$  \hspace{1cm} (2.33)

The noise for all observations, regardless of fidelity level, is assumed to be normally distributed.

$$\epsilon_1 \sim \mathcal{N}(0, \sigma_1^2)$$  \hspace{1cm} (2.34)

$$\epsilon_2 \sim \mathcal{N}(0, \sigma_2^2)$$  \hspace{1cm} (2.35)

$$\epsilon_3 \sim \mathcal{N}(0, \sigma_3^2)$$  \hspace{1cm} (2.36)

Equations 2.37, 2.38, and 2.39 show the three zero-mean Gaussian processes that are used. These Gaussian processes are independent of each other, $f_1 \perp \delta_2(x) \perp \delta_3(x)$. The flexibility of covariance functions allows for each GP to have different types of kernels, $k_1$, $k_2$, and $k_3$. However, when a prediction is made using 2.47, a single kernel function will be used.

$$f_1(x) \sim \mathcal{GP}(0, k_1(x, x'; \theta_1))$$  \hspace{1cm} (2.37)

$$\delta_2(x) \sim \mathcal{GP}(0, k_2(x, x'; \theta_2))$$  \hspace{1cm} (2.38)

$$\delta_3(x) \sim \mathcal{GP}(0, k_3(x, x'; \theta_3))$$  \hspace{1cm} (2.39)
Next, Equations 2.37, 2.40 and 2.41 show the assumed function distribution relationships in terms of the independent, zero-mean GPs defined above. These model relationships were used by Kennedy and O’Hagan [8], Bonfiglio et al. [3], [4], and Perdikaris and Karniadakis [10]. For these relationships, two regression scaling parameters, $\rho_1$ and $\rho_2$, are also introduced.

\[
f_2(x) = \rho_1 f_1(x) + \delta_2(x) \tag{2.40}
\]

\[
f_3(x) = \rho_2 f_2(x) + \delta_3(x) = \rho_1 \rho_2 f_1(x) + \rho_2 \delta_2(x) + \delta_3(x) \tag{2.41}
\]

The input/output equations, Equations 2.31 - 2.33, are combined with Equations 2.40 and 2.41 to produce Equations 2.42 - 2.44.

\[
y_1 = f_1(x_1) + \epsilon_1 \tag{2.42}
\]

\[
y_2 = \rho_1 f_1(x) + \delta_2(x) + \epsilon_2 \tag{2.43}
\]

\[
y_3 = \rho_1 \rho_2 f_1(x) + \rho_2 \delta_2(x) + \delta_3(x) + \epsilon_3 \tag{2.44}
\]

Therefore, the joint distribution with all three levels of data is shown in Equation 2.45, where $y^T = [y_1^T y_2^T y_3^T]$. This equations is reached as a result of the assumed independence of the three GPs defined above and the model relationships defined in Equations 2.40 and 2.41. For readability, Equation 2.45 is restated in larger font in Appendix A.

\[
y \sim N \left(0, \begin{bmatrix}
\rho_1 k_1(x_1, x'_1; \theta_1) + \sigma^2_T & \rho_1 k_1(x_1, x'_1; \theta_1) & \rho_1 k_1(x_1, x'_1; \theta_1) \\
\rho_1 k_1(x_1, x'_1; \theta_1) & \rho_1 k_1(x_2, x'_2; \theta_2) + \sigma^2_T & \rho_1 k_1(x_2, x'_2; \theta_2) + \sigma^2_T \\
\rho_1 k_1(x_1, x'_1; \theta_1) & \rho_1 k_1(x_2, x'_2; \theta_2) + \sigma^2_T & \rho_1 k_1(x_3, x'_3; \theta_3) + \sigma^2_T \\
\rho_1 k_1(x_1, x'_1; \theta_1) & \rho_1 k_1(x_2, x'_2; \theta_2) + \sigma^2_T & \rho_1 k_1(x_3, x'_3; \theta_3) + \sigma^2_T \\
\rho_1 k_1(x_1, x'_1; \theta_1) & \rho_1 k_1(x_2, x'_2; \theta_2) + \sigma^2_T & \rho_1 k_1(x_3, x'_3; \theta_3) + \sigma^2_T \\
\rho_1 k_1(x_1, x'_1; \theta_1) & \rho_1 k_1(x_2, x'_2; \theta_2) + \sigma^2_T & \rho_1 k_1(x_3, x'_3; \theta_3) + \sigma^2_T \\
\rho_1 k_1(x_1, x'_1; \theta_1) & \rho_1 k_1(x_2, x'_2; \theta_2) + \sigma^2_T & \rho_1 k_1(x_3, x'_3; \theta_3) + \sigma^2_T \\
\rho_1 k_1(x_1, x'_1; \theta_1) & \rho_1 k_1(x_2, x'_2; \theta_2) + \sigma^2_T & \rho_1 k_1(x_3, x'_3; \theta_3) + \sigma^2_T \\
\end{bmatrix}\right) \tag{2.45}
\]

Let $K_{mf}$ be the size $(n_1 + n_2 + n_3) \times (n_1 + n_2 + n_3)$ multi-fidelity block covariance matrix of $y$. Each diagonal block of the covariance matrix is used to model data from a single fidelity data class. The non-diagonal blocks are used for modeling correlations between classes of data.

With the three data sets grouped together, the block covariance matrix, and a predictive kernel function, the model can be trained. This means the optimum set of
hyperparameters and model parameters, $\Theta$, are learned. The model parameters, $\rho_1, \rho_2, \sigma_1^2, \sigma_2^2,$ and $\sigma_3^2$, and free parameters, $\theta_1, \theta_2,$ and $\theta_3$, are shown below in Equation 2.46.

$$\Theta = [\theta_1, \theta_2, \theta_3, \rho_1, \rho_2, \sigma_1^2, \sigma_2^2, \sigma_3^2]$$

Equation 2.12 from the single-fidelity regression description is still valid. The NLML, predicted function value, and predictive variance equations now use the block covariance matrix, $K_{mf}$, and are shown below. Again, NLML is minimized to find the optimum set of hyperparameter.

$$E[f_*] = k_*^T K_{mf}^{-1} y$$

$$V(f_*) = k_* - k_*^T K_{mf}^{-1} k_*$$

$$\Theta^* = \arg \min_{\Theta} NLML(\Theta) = \frac{1}{2} y^T K_{mf}^{-1}(\Theta) y + \frac{1}{2} \log |K_{mf}(\Theta)| + \frac{n}{2} \log(2\pi)$$

Without going through the same level of detail, the relationships and equations for a dual fidelity level GP regression are presented below. With only two fidelity classes the notation is changed levels 1, 2, and 3 to "high" and "low" fidelity levels. These results are used with a case example in Chapter 3.

$$\mathcal{D} = [\mathcal{D}_L, \mathcal{D}_H]$$

$$f_H(x) = \rho f_L(x) + \delta(x)$$

$$\Theta = [\theta_L, \theta_H, \rho, \sigma_L^2, \sigma_H^2]$$

$$y \sim \mathcal{N}(0, \begin{bmatrix} k_L(x_L, x'_L; \theta_L) + \sigma_L^2 I & \rho k_L(x_L, x'_H; \theta_L) \\ \rho k_L(x_H, x'_L; \theta_L) & \rho^2 k_L(x_H, x'_H; \theta_L) + k_H(x_H, x'_H; \theta_H) + \sigma_H^2 I \end{bmatrix})$$

### 2.1.6 Kernel Selection

Section 2.1.3 discussed common kernels and Section 2.1.4 explained how the hyperparameters can be tuned to improve the NLML. This section will briefly describe a method for searching for a composite kernels as presented by Duvenaud et al. [5].
For this kernel search methodology, the designer has three main decisions to make: (1) what base kernels to use in the dictionary of kernels, (2) what search depth to use, and (3) the number of iterations per level. First, the kernel search methodology begins by setting the dictionary of base kernels to be used. The base kernels can be any of the covariance functions described in Section 2.1.3, as well as any other valid kernel. However, the decision maker can save computational resources by intentionally choosing the kernels to put the dictionary. If there are kernels that don't make sense for the particular design problem, then by excluding them from the dictionary computational resources can be redirected. Second, the search depth indicates the number of kernels to use to form the composite kernel. To give an example of search depth, if there are 20 design variable and the search depth is four, then composite kernel will be made from base kernels applied to four of the 20 design variables. Third, the number of iterations per level is the number of random restarts at each level with different random hyperparameters and the NLML is stored. After using this GP kernel search methodology, the optimum kernel returned will be the composite kernel with the best NLML. An example of this methodology is presented in Section 3.3[5]. Figure shows the performance with an example problem. This Figure 2-14 shows for different kernels the mean squared error (MSE) as a function of dataset size and illustrates how the structure search performance well as the dataset becomes more populated.
2.2 Optimizing: Bayesian Optimization

After constructing a surrogate model using GP regression, the next step is to efficiently search the model for an optimum design. Global optimization is often computationally expensive requiring many function evaluations. The goal is to use the predictive model instead of physics-based solver to explore the design space. Simple methods to explore the tradespace include random sampling, grid sampling, and cluster sampling around observed data. In these methods, design points are selected based on the optimization paradigm, predictive values found, and the design with the minimum objective value is the new candidate designs which is then simulated. A gradient-based solver can also be applied with the global search, but the model may be non-convex and this will significantly add to the computational cost.

A potent tool for exploring the design space and searching for the global minimum while observing a computational budget is Bayesian optimization [9]. This block-box optimizer uses the predictive response surface, which provides both the predicted
value and the uncertainty, and an acquisition function to find a minimum within the design space. Instead of minimizing the quantity of interest directly, the acquisition function, also referred to as the utility function, is maximized (or minimized, based on utility function). There are a number of available acquisition functions, but the acquisition function that this primer will focus on is the Expected Improvement (EI) acquisition function. The EI acquisition function balances exploring and exploiting the response surface [7]. It explores the model by giving credit to areas within the design space that have high uncertainty. Simultaneously, it exploits the model by giving credit to areas within the design space that have desirable quantity of interest values. Section 2.2.1 gives the theory for the general EI utility without constraints and Section 2.2.2 explains how the EI utility function can be used when inequality constraints are present [6].

2.2.1 Expected Improvement Acquisition Function

As stated above the goal of Bayesian optimization with an EI acquisition function is to search the predictive model for the global minimum and for areas within the model with high uncertainty. The engineering design problem is to minimized the objective value, as shown in Equation 2.54, but BO will instead search for a new design that maximizes the utility function, as shown in Equation 2.55

\[
\begin{align*}
    x^* &= \arg \min_x f_*(x) \quad (2.54) \\
    x_{n+1} &= \arg \max_x \alpha_{EI}(x) \quad (2.55)
\end{align*}
\]

To develop the EI utility function, first the Improvement function is defined in Equation 2.56. Since the prediction from the surrogate model is a random variable, the Improvement function is also a random variable. Therefore, taking the expected value of the Improvement function yields the Expected Improvement acquisition function. Let \(x^\Delta\) be the design from the current dataset with the best observed quantity of in-
terest, referred to as the "best" design, and \( x^+ \) be the candidate design being tested.

\[
I(x^+) = \max(0, f(x^\Delta) - f_*(x^+)) \tag{2.56}
\]

\[
\alpha_{EI}(x^+) = E[I(x^+)] \tag{2.57}
\]

A closed form of Equation 2.57 was developed by Jones et al. [7] and is shown in Equation 2.58. In Equation 2.58, \( \phi \) is the standard Gaussian probability density function and \( \Phi \) is the standard Gaussian cumulative distribution function. Also, let \( Z_f \) be the quotient of the difference between the mean predicted value at \( x^+ \) and the observed "best" value and the variance at the predicted value. \( Z_f \) is defined in Equation 2.59.

\[
\alpha_{EI}(x^+) = \Phi(Z_f) \left( E[f_*(x^+)] - f(x^\Delta) \right) + \phi(Z_f) V[f_*(x^+)] \tag{2.58}
\]

\[
Z_f = Z_f(x^+) = \frac{E[f_*(x^+)] - f(x^\Delta)}{V[f_*(x^+)]} \tag{2.59}
\]

In Equation 2.58 note the two components that are combined to form the EI utility function. The left term gives credit for improving upon the best design. Therefore, the left term is the exploitation term. The right term gives credit where there is high uncertainty. At first this may seem odd, but this is the exploration term. The exploration component gives credit to regions which need more data to improve the model.

To explore the tradespace using BO, the design space is sampled, the EI utility function and gradient are calculated, and the design follows the path of greatest ascent until a maximum is reached. In each iteration, the maximum EI reached for each restart is stored, the maximums are compared, and the design with the greatest EI is the next design simulated with the physics-based solver. This new input/output pair is added to the training dataset and the model is re-trained. BO can be considered a "greedy" optimization methodology, since with each iteration either the "best" design or the model is improved. Figure 2-15 illustrates the BO iterative process in 1-D.
Figure 2-15: 1-D Example of Bayesian Optimization
2.2.2 Constrained Expected Improvement Acquisition Function

When constraints are imposed upon the design selection, the original EI acquisition function will most likely nominate infeasible designs. Therefore the probability of satisfying all the constraints must be factored into the BO EI methodology. Let \( c_k \) be constraint \( k \) and assume all constraints are of the form \( c_k \geq 0 \). Constraints not originally expressed in this manner can be mathematically rewritten into this form, as shown below.

For constraints \( l_k \leq 0 \), let \( c_k = -l_k \)

For constraints \( l_k \geq T \), let \( c_k = l_k - T \)

For constraints \( l_k \leq T \), let \( c_k = -l_k + T \)

The EI acquisition function value found using the predictive model for the quantity of interest is then adjusted based on the constraint surrogate model values. However, the constraint models provides mean and variance function values, just as the object regression model. Therefore, there will never be certainty whether a design is feasible or infeasible using the predictive models, but rather a probability of satisfy all the constraints. These probabilities are multiplied with the EI acquisition function defined in 2.58 to produce the Constrained Expected Improvement (CEI) acquisition function. The probabilities are able to be simply multiplied together because the models, and therefore the probabilities, are assumed to be independent.

\[
\alpha_{CEI}(x^+) = \alpha_{EI}(x^+) \prod_{k=1}^{K} Pr\left(c_k(x^+) \geq 0\right) \tag{2.60}
\]

Conveniently, the other surrogate models were also built using GP regression and each constraint probability is found using a Gaussian cumulative distribution function \( \Phi \). This concept was already observed in Section 2.2.1 where the exploitation term in the EI acquisition function was adjusted based on the confidence of improving using a
Gaussian cumulative distribution function. There is now a $Z_k$ value for each constraint $k$, as shown in Equation 2.62, in addition to the $Z_f$ defined in 2.59. Finally, while the unconstrained EI function defined the "best" design as the design with the minimum engineering objective value, $x^\Delta$, with the inclusion of constraints in design problem, the "best" design is redefined as the observed design with the minimum engineering objective value that satisfies all constraints. While they are defined differently, the same notation will be used for the "best" observed design. The Constrained Expected Improvement acquisition function is defined in Equation 2.61.

$$\alpha_{CEI}(x^+) = \Phi(Z_f) \left( \mathbb{E}[f_*(x^+) - f(x^\Delta)] + \phi(Z_f)\mathbb{V}[f_*(x^+)] \right) \prod_{k=1}^{K} \Phi(Z_k)$$

(2.61)

$$Z_k = Z_k(x^+) = \frac{\mathbb{E}[c_k(x^+)] - 0}{\mathbb{V}[c_k(x^+)]} = \frac{\mathbb{E}[c_k(x^+)]}{\mathbb{V}[c_k(x^+)]}$$

(2.62)
Chapter 3

Design Examples

3.1 Single-Fidelity and Manual Design Example: Large Surface Combatant Project

The machine learning principles introduced in Chapter 2 are first manually applied to a single-fidelity design example. This example was part of a year-long team design project for MIT course 2.705. In addition to me, the other team members were LT Jordan Fouquette and LT Adam Campbell.

3.1.1 Project Background

Due to the evolving capabilities of foreign navies and the aging inventories of United States Navy (USN) Cruisers and Destroyers, the time has come for the USN to determine effective and viable replacements for the future fleet. This led to the creation of the Future Surface Combatant Force comprised of large, small, and unmanned surface combatant ships. Specifically, the Large Surface Combatant will serve as the replacement for Cruisers and Destroyers by encompassing the multi-mission capabilities of each platform, respectively.

The goal for this project was to conduct an in-depth evaluation of a concept design for the Large Surface Combatant. The LSC is a multi-mission capable platform
with the capability and flexibility to take advantage of both current and future technologies. This study included and in-depth analyses of requirements for adaptable design, the integration of advanced weapon systems including the electromagnetic railgun and high-energy laser, and incorporation of various UxV platforms.

The LSC was designed with a flexible architectural arrangement for inclusion of both existing and future technologies. This was accomplished by taking a zonal approach for ship systems and equipment configurations. Each system is comprised of zones based on their specific functionality. These zones are predetermined spaces with maximum allowances for space (area and volume), weight, power, and cooling (SWAPC). The zones are categorized into four overall system groups:

1. Weapon Systems
2. Sensors
3. Aviation
4. Waterborne Vehicles

Project Requirements

The LSC project requirements are expressed in terms of the key performance parameters (KPPs). The KPPs for this project were developed by LSC design sponsors. These sponsors work in the following USN workcenters: (1) Supervisory Program Manager office for the Zumwalt class Destroyer (PMS 500), (2) the Director of Surface Warfare’s office (OPNAV N96), and from the Naval Surface Warfare Center (NSWC) Carderock Division. The LSC project KPPs are listed in Table 3.1.
Table 3.1: Sponsor Requirements

<table>
<thead>
<tr>
<th>Design Tools</th>
</tr>
</thead>
</table>
| The design team utilized multiple programs and tools to complete the various analyses associated with this project. These tools are summarized below:

- **Advanced Ship and Submarine Evaluation Tool (ASSET)** - Ship design and evaluation tool utilized for ship convergence and synthesizing.

- **Rhino** - 3D CAD Program utilized for analyzing shipboard arrangements.

- **Maestro** - Design, analysis, and evaluation tool utilized for structural and see-keeping analyses.
- Rapid Ship Design Environment (RSDE) - Ship evaluation tool that rapidly generates a complete design space through the design and analysis of thousands of ship variants.

- MIT 2N Cost Model - Academic weight-based cost model utilized for providing lead and follow-on ships cost estimates.

The main tool that used in collaboration with the machine learning tools was RSDE. RSDE is a parametric, physics-based solver. RSDE was used to physically simulate ship design feasibility and performance when given independent variable with bounds and an ASSET converged baseline design. Outputs from RSDE were imported into the team's Analytic Hierarchy Process framework to calculate the objective and constraint quantities. Therefore, GP regression was used to model the RSDE simulations.

### 3.1.2 Design Analysis

#### Decision Framework

A major aspect of the decision framework utilized for this project was the incorporation of an Analytic Hierarchy Process (AHP) into the decision-making process. The design team utilized an AHP framework containing four levels. Level 1, the upper-most tier, is the OMOE which is the sum of all the normalized components found in level 2. Level 2 is comprised of the overall ship systems that were used for the creation of the zonal architecture zones. Level 3 is comprised of the actual physical zones. Finally, level 4 is a subdivision of "Armament" in level 3 in order to provide more discrete sizes and increased clarity with regard to functionality. Figure 3-1 provides an illustrative breakdown of this AHP framework.

This AHP framework was used to form a survey allowing the project sponsors the ability to provide their inputs and desires regarding the importance of every ship system/mission capability versus one another. This was accomplished as the survey was comprised of pairwise-comparisons between each component of levels 2, 3, and 4 amongst all other components within their respective levels. This process allowed the sponsors to unbiasedly prioritize certain overall ship systems, missions capabilities,
Figure 3-1: Analytic Hierarchy Process Overview
and specific functional zones over one another, greatly aiding in the overall design of the ship structure and hullform.

Based on survey results from our sponsors, the KPPs were weighted and functions to score each KPP were developed. Each design variant was assigned an Overall Measure of Effectiveness (OMOE) score based on the KPPs gained from the sponsor survey. Measures of Performance (MOP) functions were also determined from the sponsor survey, and were used to map ship characteristics (i.e. speed, endurance, etc) into KPP satisfaction. AHP weights were then used to determine the OMOE's for each specific variant.

In addition to determining OMOE's for each variant, the design team calculated the Overall Measure of Cost (OMOC), or also refereed to as cost, for each variant. The cost of each variant will include projected non-recurring engineering expenses, production cost, and operations and support values.

Finally, RSDE outputs were directly imported into the OMOE and OMOC functions for each variant. The design objective is to maximizes OMOE while satisfying the OMOC constraint, as shown in Equation 3.1.

\[
\text{maximize } \quad OMOE(x) \\
\text{subject to } \quad OMOC(x) \leq 3.1B
\]

Design Space

Each design is characterized by the vector of its independent design variables, \(x\), and the baseline ship.

**Independent Design Variables**  OMOE and OMOC are functions of the ship design inputs. Traditionally, ships are designed for specific capabilities or systems. The modern design paradigm for LSC is a zonal approach, which seeks to maximize
SWAPC in the zones listed previously. An overview of the zones is shown in Table 3.3 and the actual design variables are shown in 3.2. Each design variant is stored as vector, \( \mathbf{x} \) which represents is composed of values for each of the 59 design variables in Table 3.2.

Table 3.2: Design Space Variable Inputs to RSDE
Each of these design variables were given an established range between a minimum and maximum value associated with each key performance parameter. The majority of these variations occurred through the system zones’ weight and power allocations within their respective Payload and Adjustment entries.

**Baseline Ships**  In order to use RSDE an ASSET-converged baseline ship is needed as an initial starting point. This was accomplished by taking into account three primary baseline ships using the following hullforms:

1. Ticonderoga-Class Cruiser (CG) Hull
2. Arleigh Burke-Class Destroyer (DDG) Hull
3. San Antonio-Class Landing Platform Dock (LPD) Hull
While the dimensions and systems of these ships were slightly modified from the existing design, these three ships were chosen based on their proven performance characteristics and comparable sizes. They provided a meaningful range of hull characteristics, such as volume, slenderness/fullness, and curvature, to create the numerous variants to be analyzed within the design space. These ASSET models of these
baseline ships were then used within RSDE to create new variants.

Establish Benchmark Design and Create Training Dataset

With all three ASSET baseline models complete, the next step was to utilize RSDE to generate the design space comprised of numerous variants of the CG, DDG, and LPD hullforms created and modified in ASSET. Figure 3-3 provides a general overview of how RSDE is used to accomplish this task. In addition to the baseline models, RSDE also has design variable inputs. These design variables are also the inputs for machine learning that will take place in the next design phase. The design variables are listed in Table 3.2.

Figure 3-3: Overview of RSDE [?]

60
Each baseline, along with all of their design variables, were provided as inputs into RSDE. All variable bounds were adjusted through many iterations conducted by RSDE in order to create a large design space of variants with each containing unique characteristics. This process generated approximately 2000 variants for each baseline. Each of these approximately 6000 unique variants were assigned an OMOE and OMOC score. Next, all design variants were plotted with OMOE versus OMOC. This plot was used to assess dominance amongst all variants with the goal to have a high OMOE and low cost. The plot of first iteration is seen in Figure 3-4.

![OMOE vs Cost](image)

**Figure 3-4: 1st Iteration: OMOE vs Cost**

Included within Figure 3-4 is a Pareto front which is used to highlight the more ideal candidate concepts. To aid in the selection process, Figure 3-4 was subdivided into four quadrants as seen in Figure 3-5.
Quadrant 1 is described as having a desirable OMOE, but being cost prohibitive. Quadrant 2 variants possess desirable OMOEs and are cost acceptable. Quadrant 3 contains undesirable OMOEs, but the variants are cost acceptable. Finally, Quadrant 4 variants have undesirable OMOEs and are cost prohibitive. With cost serving as a key performance parameter, Quadrants 1 and 4 were removed from further consideration due to their variants’ costs exceeding the cost threshold of $3.1 billion. Figure 3-6 is a plot of OMOE versus cost for only Quadrants 2 and 3, which were utilized for subsequent analyses due to them meeting the cost threshold.
With this plot of OMOE versus cost created for cost acceptable variants, additional iterations were conducted by the design team on select variants in order to refine a set of dominant solutions. The goal for these additional iterations were to create a higher granularity set amongst the dominant solutions. The second iteration was completed on one CG, three DDGs, and one LPD variant. At least one variant created from each of the three baseline models were included within this iteration. They were selected based on their superior OMOE and cost relationships compared to all other variants. These refined iterations were conducted by inputting these selected variants back into RSDE, along with all of their associated design variables as listed in Table 3.2, but with a variable range of 5% above and below their respective values. This provided a more detailed set of solutions centered around these more optimal variants as seen in Figure 3-7.
It can be seen from Figure 3-7 that the CG variants, created from C429, possess the most desirable OMOEs while a majority of these variants are cost acceptable below the $3.1 billion threshold. This refined iteration was validated by creating an efficiency term which was simply a ratio of OMOE to cost.

A third iteration was conducted on the same CG variant, C429, with the number of ASSET required iterations for convergence reduced from 12 to 10. This was completed to ensure that the variants could be converged and synthesized properly within ASSET. This iteration resulted in the creation of three new variants, C429”, as seen in Figure 3-8.
Finally, after the third iteration a benchmark design was established. The benchmark design is labeled *Design C429_rev2_18* and has an OMOE of 0.3172 and satisfied the cost constraint.

**Methodology and Results**

The previous section described the iterative process that led to a benchmark design variant. This section will describe the machine learning techniques used to find a new design that improves upon the benchmark design. The machine learning techniques applied are single-fidelity Gaussian process regression and Bayesian optimization.

RSDE is a complex physics-based design tool with a high computational cost. The computational resources spent to determine the 6000 design variants described pre-
viously was greater than 1000 CPU-hours. This is because RSDE simulates physical phenomena with a multi-dimensional design space and interconnected, multi-dimensional outputs. Since RSDE simulations are time intensive, it is difficult to use this tool to thoroughly explore the design space.

The initial training dataset was composed of 300 pairs of design inputs and quantity of interest output. The only baseline ship considered was the CG class, since this was the same baseline of the benchmark design. These were used to train initial surrogate models for OMOE and OMOC using Gaussian process regression. With these models, Bayesian Optimization was performed with a Constrained Expected Improvement acquisition function. For each BO step in the process, 100,000 samples were taken and the top 50 designs in terms of CEI were used to set up the next RSDE simulation. For each design input, the RSDE bounds we set using the minimum and maximum from top 50 designs. Next, a RSDE simulation was initiated with 200 runs, using a Latin Hypercube sampling method within the new bounds. Then, the results from the RSDE simulations that converged were then added to the dataset. This Bayesian Optimization design loop is displayed in Figure 3-9.
The LSC BO design loop was terminated when a design was found that significantly improved upon the benchmark design and showed that CEI was converging towards zero. Even with thorough sampling within the tradespace (100,000 restarts), each iteration through the design loop took less than five CPU-hours.

Figures 3-10 through 3-13 show the results from each iteration of the design loop. The caption in each Figure further describes each iteration.
Figure 3-11: LSC BO Iteration 1 results. Training dataset contains 300 observation. With 100,000 samples, the top 50 BO results had an average EI of 4.6. With 200 RSDE simulations, there were 10 designs that converged.

Figure 3-12: LSC BO Iteration 2 results. Training dataset contains 310 observation. With 100,000 samples, the top 50 BO results had an average EI of 2.7. With 200 RSDE simulations, there were 19 designs that converged.
Figure 3-13: LSC BO Iteration 3 results. Training dataset contains 329 observation. With 100,000 samples, the top 50 BO results had an average EI of 0.7. With 200 RSDE simulations, there were 30 designs that converged.

**Example Conclusion** In conclusion, a new design was discovered that improved OMOE by more than 17% and slightly reduced Cost. The preferred concept design is labeled $BO_{-} CG_{-} 35$ and has an OMOE of 0.3848. This analysis was accomplished in three iterations with a computational cost less than 15 CPU-hours. The computational resources spent during machine learning analyses was 98% less than the computational resources spent to generate the initial benchmark design using RSDE alone. Using this design variant, detailed engineering analysis was performed. The final outfitted design of the Large Surface Combatant is shown in Figure 3-14.
3.2 Multi-Fidelity and Automatic Design Example: 2D Super-Cavitating Hydrofoil Project

The machine learning principles introduced in Chapter 2 are now applied to a multi-fidelity design example in an automatic design loop. This example is a variation of the work done by J. Royset, L. Bonfiglio, G. Vernengo, and S. Brizzolara [12]. In this paper, the authors designed a hydrofoil using Risk-Adaptive Set-Based design. Instead of focusing on risk, this thesis focuses on implementing an automatic design framework for shape optimization. The same computational fluid dynamic (CFD) solvers are used and similar quantities of interest (QoIs) explored to optimize the shape of a 2D super-cavitating hydrofoil. The physics-based solvers implemented were provided by Luca Bonfiglio, Post-doctoral researcher in Mechanical Engineering department at Massachusetts Institute of Technology and are detailed in a number
of articles [2][14]. Surrogate modeling is accomplished using GP regression with an ARD kernel.

### 3.2.1 Project Background

The goal of this project is to optimize the design of a hydrofoil to be used at ultra-high speeds. The surface-piercing hydrofoil is designed in two dimensions in an assumed stable super-cavitating environment. Figure 3-15 presents hydrofoil shapes designed for different regimes and was produced by Auslaender, J.[1][14].

![Figure 3-15: Hydrofoil shapes designed for different cavitation environments. Red curves show the cavity contours and the blue lines show the flow streamlines.](image-url)

<table>
<thead>
<tr>
<th>Environment</th>
<th>Hydrofoil Shapes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully Wetted</td>
<td>Subcavitating</td>
</tr>
<tr>
<td>Incipient Cavitation</td>
<td>Supercavitating (Vapor or Air Cavity)</td>
</tr>
<tr>
<td>Cavitation Erosion</td>
<td>Supercavitating (Fully Ventilated Cavity)</td>
</tr>
<tr>
<td>Buffering</td>
<td>Base Vented</td>
</tr>
</tbody>
</table>
Designing in stable super-cavitating regime means a number of parameters and operating conditions are set constant while the shape of the hydrofoil is optimized. The constant operating conditions and hydrofoil parameters are the following: (1) operating speed \( U = 61.667m/s = 120\text{knots} \), (2) angle of attack \( \alpha = 6^\circ \), (3) water density \( \rho = 997.2kg/m^3 \), (4) cavitation index \( \sigma = 0.05 \), and (5) hydrodynamic chord length \( c_H = 0.66m \).

**Project Requirements**

The objective quantity is to maximize the Lift-to-Drag ratio, \( L/D \), where the lift force is found using \( L = C_L\rho cU^2 \) and the drag force is found using \( D = C_D\rho cU^2 \). Therefore, the object QoI is re-written as \( L/D = \frac{C_L\rho cU^2}{C_D\rho cU^2} = \frac{C_L}{C_D} \).

In addition to the objective QoI, there are three constraint QoIs. All four QoIs are shown in Equation 3.7. The first constraint QoI that is modeled is the lift coefficient, \( C_L \). The lift coefficient is defined as \( C_L = \frac{L}{\rho cU^2} \). The hydrofoil must produce an appropriate lift force to support the displacement of the ship. Therefore, to ensure a minimum lift is generated to prevent sinkage, the following constraint is imposed: \( C_L \geq 0.3 \). Also, to prevent the ship from experiencing over-lift, a maximum lift constraint is imposed: \( C_L \leq 0.4 \).

The second constraint QoI that is modeled is the thickness of the cavitation bubble at 10\% of the chord length, \( t_c^{10\%} \). To ensure a stable super-cavitating hydrofoil, a minimum cavitation thickness is imposed: \( t_c^{10\%} \geq 0.004m \).

The final constraint QoI is the profile inertia modulus, \( w \). However, this constraint doesn't require a surrogate model, since it can be analytically calculated using the shape of the hydrofoil. To ensure the profile has adequate strength to undergo the high pressure loads necessary for developing high values of lift, the final QoI constraint
is the following: \( w \geq 7.5^{-6} m^3 \).

**Design Tools**

Two physics-based design tools with different levels of fidelity were utilized to simulate the QoIs in described in Section 3.2.1. The low-fidelity Computational Fluid Dynamics (CFD) simulator is a boundary element method (BEM) potential flow solver. The high-fidelity CFD simulator is an unsteady Reynolds-averaged Navier-Stokes (URANS) solver. Using the same machine, the CPU time for the low-fidelity solver is about 5 seconds and the time for the high-fidelity solver is about 50 minutes.

**High-fidelity CFD** The high-fidelity solver for the multi-phase flow around the hydrofoil is a viscous solution of the URANS equations. The Navier-Stokes equations with incompressible fluid are shown in Equation 3.1 and 3.2 and are written in the Cartesian notation.

\[
\begin{align*}
\frac{\partial (\rho u_i)}{\partial x_i} &= 0 \\
\frac{\partial u_i}{\partial t} + \frac{\partial (u_i u_j)}{\partial x_j} &= -\frac{1}{\rho} \frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j} \nu \frac{\partial u_i}{\partial x_j} + f_i
\end{align*}
\]  

These governing equations were solved using a Reynolds averaging methodology that involved the \( k - \epsilon \) model and volume of fluid (VOF) equations as described by Bonfiglio et al. [4] [2].

**Low-fidelity CFD** The low-fidelity CFD solver is a BEM potential flow simulator implemented by Vernengo et al. [14]. Potential flow implies irrotational flow and incompressible fluid, and the assumption of inviscid flow was also applied. With these assumptions, the Laplace equation for the scalar velocity potential, \( \varphi \), is shown in Equation 3.3. For pressure, \( p \), the Bernoulli equation for unsteady flow is used and is shown in Equation 3.4. The Laplace and Bernoulli equations replace the continuity

\[
\Box
\]
and momentum equations as the governing equations for fluid motion.

\[ \nabla^2 \varphi(x) = 0 \quad (3.3) \]

\[ p = -\rho \left[ \frac{\partial \varphi}{\partial t} + 0.5|\nabla \varphi|^2 + g z \right] + f \quad (3.4) \]

To solve the system of equations during cavitation, two boundary conditions are applied to the problem. The first is to describe the cavity surface as a flow streamline defined by the bubble thickness, \( t_c \). This boundary condition is expressed in Equation 3.5 where \( n \) is the local boundary normal. The second boundary condition is shown in Equation 3.6 and applies the saturation pressure to the cavity boundary cavitating. These governing equations and boundary conditions were used in an iterative process to converge on the hydrodynamic forces. Numerical integration and extrapolation techniques were applied to calculate the forces on all faces of the hydrofoil. Finally, the viscous affects were also applied using flat plate approximations for local frictional coefficients [14].

\[ \frac{D}{Dt}(n - t_c(x)) = 0 \quad (3.5) \]

\[ p = p_{\text{vapor}} \quad (3.6) \]

### 3.2.2 Design Analysis

#### Decision Framework

The ideal design is the hydrofoil that maximizes the object QoI and satisfies all the constraint QoIs. The decision framework is expressed mathematically in Equation 3.7. The most unique aspect of this decision framework is that decisions will be automatically within the design loop without human intervention. The program is written to automatically train the surrogate models, search the models for new designs, balance improving the design and improving the surrogate models, and initiate
high-fidelity simulations for candidate designs.

\[
\begin{aligned}
\text{maximize} & \quad \frac{C_L}{C_D} \\
\text{subject to} & \quad C_L \geq 0.3 \\
& \quad C_L \leq 0.4 \\
& \quad t_{10\%} \geq 0.004m \\
& \quad w \geq 7.5^{-6}m^3
\end{aligned}
\]

(3.7)

**Design Space**

Each design is characterized by the vector of 15 independent design variables, \( x = (x_1, \ldots, x_{15}) \). Each of the 15 design variables correspond to a component of a Cartesian coordinate, which together form a control polygon that describes the unique shape of a hydrofoil. Each of the design variables has a minimum and maximum value that describe the breadth of the design space. Figure 3-16 shows how the 15 design variable in \( x \) form a hydrofoil.[12] Table 3.4 describes the lower and upper bounds for each of the 15 design variables.

![Figure 3-16: Hydrofoil Design Variable, \( x \). Fixed points are indicated by green boxes. Arrows indicate control point movement is allowed. Note that \( x_{15} \) controls the vertical movement of three control points.[12]](image)
The shape of the hydrofoil is controlled by three types of control points: (1) five fixed points which are at the leading edge, the trailing edge, and one on the tail pressure side, (2) five points on the pressure side, described by $x_1 - x_9$, and (3) five points on the suction side, described by $x_{10} - x_{15}$. The five control points on the pressure side govern the Lift-Drag ratio and coefficient of Lift QoIs. The five control points on the suction side govern the inertia modulus and cavity bubble thickness QoIs.

### Establish Benchmark Design and Create Training Dataset

An initial multi-fidelity sampling plan was generated using both low- and high-fidelity simulators. A Latin hypercube sampling methodology was used to get a near-random sample of design parameter values between the upper and lower bounds for each parameter. In the initial dataset there were 442 low-fidelity simulations and 111

---

<table>
<thead>
<tr>
<th>Design Variable, $x_i$</th>
<th>Lower Bound, $x_{i}^{\text{min}}$</th>
<th>Upper Bound, $x_{i}^{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$4.50000000E - 02$</td>
<td>$1.50000000E - 01$</td>
</tr>
<tr>
<td>$x_2$</td>
<td>$2.00000000E - 01$</td>
<td>$3.00000000E - 01$</td>
</tr>
<tr>
<td>$x_3$</td>
<td>$3.15000000E - 01$</td>
<td>$4.49862678E - 01$</td>
</tr>
<tr>
<td>$x_4$</td>
<td>$5.00000000E - 01$</td>
<td>$5.92719056E - 01$</td>
</tr>
<tr>
<td>$x_5$</td>
<td>$0.00000000E + 00$</td>
<td>$5.00000000E - 03$</td>
</tr>
<tr>
<td>$x_6$</td>
<td>$2.00000000E - 03$</td>
<td>$1.10000000E - 02$</td>
</tr>
<tr>
<td>$x_7$</td>
<td>$1.38637470E - 02$</td>
<td>$3.38000000E - 02$</td>
</tr>
<tr>
<td>$x_8$</td>
<td>$5.00000000E - 03$</td>
<td>$2.50000000E - 02$</td>
</tr>
<tr>
<td>$x_9$</td>
<td>$5.00000000E - 04$</td>
<td>$5.00000000E - 03$</td>
</tr>
<tr>
<td>$x_{10}$</td>
<td>$1.77388706E - 01$</td>
<td>$2.50100000E - 01$</td>
</tr>
<tr>
<td>$x_{11}$</td>
<td>$3.69330422E - 01$</td>
<td>$4.50000000E - 01$</td>
</tr>
<tr>
<td>$x_{12}$</td>
<td>$4.95000000E - 01$</td>
<td>$6.05000000E - 01$</td>
</tr>
<tr>
<td>$x_{13}$</td>
<td>$2.81721660E - 02$</td>
<td>$3.52000000E - 02$</td>
</tr>
<tr>
<td>$x_{14}$</td>
<td>$5.04000000E - 02$</td>
<td>$6.35876580E - 02$</td>
</tr>
<tr>
<td>$x_{15}$</td>
<td>$6.90000000E - 02$</td>
<td>$7.83869890E - 02$</td>
</tr>
</tbody>
</table>
From the initial dataset, a benchmark design was established. The benchmark design was the design with the greatest Lift-Drag ratio and satisfied all the constraints. The benchmark design has a Lift-Drag ratio of 10.76 and is shown in Figure 3-17.

Figure 3-17: Benchmark Hydrofoil Shape

(a) Vapor Content of benchmark hydrofoil
(b) Pressure Distribution of benchmark hydrofoil

Figure 3-18: Benchmark hydrofoil pressure and vapor distributions
Methodology and Results

The 2D super-cavitation hydrofoil automatic design loop is shown in Figure 3-19. The Figure gives an overview of the automatic design process, including the input and output files for each step and a short description. A more detailed description of each step of design loop is presented below the Figure. To initiate the design loop, the initial multi-fidelity dataset was used to train initial surrogate models.

**Surrogate Environment**  
Training the surrogate models requires four inputs: 1) a low-fidelity design matrix, $X_L$, 2) a high-fidelity design matrix, $X_H$, 3) a low-fidelity simulation results matrix, $y_L$, and 4) a high-fidelity simulation results matrix, $y_H$. Regardless of the dataset revision (the initial or an appended using simulation results) multi-fidelity Gaussian regression is used to train surrogate models for each QoI. The kernel used in each iteration is the product of RBF kernels for each design variable, which is called automatic relevance determination and defined in Section 2.1.3. Previous regression models are not used to train future models, only the data is used to train...
the models. As described in Section 2.1, training a model means identifying the set of hyperparameters that minimizes the negative log-marginal likelihood of the model. The hyperparameters of the model are the low- and high-fidelity lengthscales for the kernel, $\theta_L$ and $\theta_H$, the scaling parameter that relates the low- and high-fidelity Gaussian processes, $\rho$, and the low- and high-fidelity noise variances, $\sigma^2_L$ and $\sigma^2_H$. Once the optimum hyperparameters are determined, the surrogate model is found for that QoI. This is done for each QoI and the models are saved as DL_surrogate_rev_k.pckl, CL_surrogate_rev_k.pckl, and tk_surrogate_rev_k.pckl, where k is the design loop iteration number. The tool that calculates each model is named multifidelity_GP.py.

Using the most recently trained surrogate models, the design space is explored for new candidate designs. There are two options within the design loop for finding new designs. One method uses Bayesian optimization to maximize the Constrain Expected Improvement acquisition function. This method is computationally cheaper than the second method and incorporates the probability of satisfying all the constraints. Also, this method automatically balances exploiting the design space (identifying designs where model thinks an optimum exists) and exploring the surrogate model (identifying design regions where the model has relatively high uncertainty). The second method directly searches for designs that maximizes the optimizing QoI starting from random initial design values. This method is computationally more expensive, but will search for the global optimum without being held back by constraint QoIs which might be incorrectly preventing the first method from exploring a rich section of the design space. Figure 3-19 shows the design loop with the Bayesian optimization search method, since this was the method used for this project. The Bayesian optimization tool is named BO.py and the direct prediction tool is named Direct_Predict.py. When initializing the automatic design loop, the human designer must choose which search paradigm to utilize. Regardless of the search method, the top candidate is saved as CandidateDesigns_Iter_k.dat and is the input into the Simulation Environment.
**Simulation Environment**  The simulation environment within the 2D hydrofoil design loop is the high-fidelity CFD URANS solver. The low-fidelity solver was only used to generate part of the initial training set. The simulation environment has two inputs: 1) the candidate design vector, x, and 2) an operating conditions vector, v. The ultimate output for the simulation environment are the time-averaged QoIs over steady-state time intervals.

In the first module, Geometry Setup, the inputs x and v are used to generate the input files for meshing engine (GMSH), which are a 2D geometry data file (geometry.dat) and the mesh instructions. Next, the mesh instructions, 2D geometry, and operating condition vector are input into the Mesh Generation module. This module will produce a mesh file (geometry.msh) compatible with the CFD solver. Next, the CFD Interface module will use the mesh file and the operating condition vector to automatically set up the initial and boundary conditions, the fluid characteristics, the flow model, and the numerical solvers and schemes necessary for the flow prediction. These output will be compiled into a single data file (inputCFD.dat). The inputCFD.dat file is the input into the CFD Solution module which will calculate the CFD solution, storing each QoI data at each time step, by utilizing OpenFOAM libraries. The outputs from this module are time-step folders containing field variables and integral quantities, such as forces (force.dat and forceCoeffs.dat). Finally, the QoI Prediction module is a post-processing tool that takes the CFD Solution outputs and produces the QoIs for the design vector, x, under the operating conditions, v.

After each iteration, the "best" design is compared with the simulation environment output. If the newest design that was simulated has in improved Lift-to-Drag ratio and satisfies all the constraint, it replaces the "best" design and is used in future iterations.

**Results**  The automatic design loop was implemented in two phases: Phase 1 is a coarse design search and Phase 2 is a fine design search.
Phase 1  Phase 1 started with the initial dataset and trained surrogate models, and iterated through the automatic design loop 100 times. In each iteration, the number of Bayesian optimization restarts was 1000. The number of BO restarts is the number of random design points that are used to explore and search the surrogate models. A higher number of BO restarts indicates a more refined search of the model. In Phase 1, the number of BO restarts is moderate, since the goal of Phase 1 is to improve the models and make obvious improvements upon the benchmark design. The results from Phase 1 are shown in Table 3.5 and Figures 3-20 through 3-25.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>D/L</th>
<th>L/D</th>
<th>Percent Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.092902789</td>
<td>10.76</td>
<td>0.00%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.091376341</td>
<td>10.94</td>
<td>1.67%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.090972396</td>
<td>10.99</td>
<td>2.12%</td>
</tr>
<tr>
<td>7</td>
<td>0.088483169</td>
<td>11.30</td>
<td>4.99%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>0.088079071</td>
<td>11.35</td>
<td>5.48%</td>
</tr>
</tbody>
</table>
(a) Phase 1: Lift-Drag Ratio

(b) Phase 1: Percent Improvement

Figure 3-20: Phase 1 Results
Figure 3-21: Phase 1 Results: Hydrofoil Improved Shapes by Iteration

Figure 3-22: Hydrofoil pressure and vapor distributions: Iterations 4-5
Figure 3-23: Hydrofoil pressure and vapor distributions: Iteration 6

(a) Vapor Content of best hydrofoil: Iteration 6

(b) Pressure Distribution of best hydrofoil: Iteration 6

Figure 3-24: Hydrofoil pressure and vapor distributions: Iterations 7-47

(a) Vapor Content of best hydrofoil: Iterations 7-47

(b) Pressure Distribution of best hydrofoil: Iterations 7-47

Figure 3-25: Hydrofoil pressure and vapor distributions: Iterations 48-100

(a) Vapor Content of best hydrofoil: Iterations 48-100

(b) Pressure Distribution of best hydrofoil: Iterations 48-100
Phase 2  Phase 2 started with the final dataset and trained surrogate models from Phase 1, and iterated through the design loop five times. In each iteration, the number of Bayesian optimization restarts was 1,000,000. The number of BO restarts in Phase 2 is 99900% greater than the Phase 1 value. This is consistent with the goal of Phase 2, which is to use the improved models created in Phase 1 to finely search the models for global improvements of the hydrofoil design. The results from Phase 2 are shown in Table 3.6 and Figures 3-26 through 3-30.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>D/L</th>
<th>L/D</th>
<th>Percent Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.092902789</td>
<td>10.76</td>
<td>0.00%</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>100</td>
<td>0.088079071</td>
<td>11.35</td>
<td>5.48%</td>
</tr>
<tr>
<td>101</td>
<td>0.085346869</td>
<td>11.717</td>
<td>8.85%</td>
</tr>
<tr>
<td>102</td>
<td>0.085283969</td>
<td>11.726</td>
<td>8.93%</td>
</tr>
<tr>
<td>103</td>
<td>0.085283969</td>
<td>11.726</td>
<td>8.93%</td>
</tr>
<tr>
<td>104</td>
<td>0.085283969</td>
<td>11.726</td>
<td>8.93%</td>
</tr>
<tr>
<td>105</td>
<td>0.085283969</td>
<td>11.726</td>
<td>8.93%</td>
</tr>
</tbody>
</table>

Table 3.6: Results from Phase 2
Figure 3-26: Phase 2 Results
(a) Best Hydrofoil: Iteration 101

(b) Best Hydrofoil: Iterations 102-105

Figure 3-27: Phase 2 Results: Hydrofoil Improved Shapes by Iteration
Figure 3-28: Final Results
Example Conclusion In conclusion, a new design was discovered that improved the Lift-to-Drag ratio by more than 8.8%. This was done automatically, without spending human resources or introducing potential human bias to the final solution. This analysis was accomplished in 105 iterations, with more than 5.1 million designs being explored in the simulation environment. The final design results are shown in Figure 3-31.
Figure 3-31: Hydrofoil shape, pressure and vapor distributions of Final Design
3.3 Multi-Fidelity and Manual Design Example: Super-Cavitating Hydrofoil Optimization with Composite Kernel

To demonstrate the feasibility of using optimized custom kernels with multi-fidelity data, the Kernel Search methodology described in Section 2.1.6 was applied to the hydrofoil design problem described in Section 3.2.

For this analysis, the initial high-fidelity dataset was used to search for an optimum composite kernel for each of the QoI. The dictionary of kernels was composed of the following kernels: RBF, RQ, LIN, and PER. The maximum search depth was ten variables. The number of iterations per kernel level was 1000. This means for each level of the search, there were 1000 random iterations which varied based on combination of kernels types and hyperparameters. It is possible to improve the composite kernel by increasing the search depth or the number of iterations per level. With this search criteria, the optimum composite kernels for Drag-Lift ratio, Lift Coefficient, and cavitation bubble thickness were determined and are shown in Equations 3.8, 3.9, and 3.10, respectively. The structure search yields both the covariance function type and the full set of hyperparameters.

\[
k_{DL}(\mathbf{x}, \mathbf{x'}) = \left( LIN(x_5, x_5'; \theta_5) \times LIN(x_8, x_8'; \theta_8) + LIN(x_7, x_7'; \theta_7a) \right) \times \\
\left( LIN(x_0, x_0'; \theta_0) \times RBF(x_2, x_2'; \theta_2) \times PER(x_3, x_3'; \theta_3) + \\
\left( LIN(x_1, x_1'; \theta_1) \times LIN(x_4, x_4'; \theta_4) \right) \times \\
\left( LIN(x_6, x_6'; \theta_6) \times LIN(x_7, x_7'; \theta_7a) \right) \right)
\]

\[
k_{CL}(\mathbf{x}, \mathbf{x'}) = \left( LIN(x_6, x_6'; \theta_6) \times LIN(x_7, x_7'; \theta_7) + LIN(x_0, x_0'; \theta_0) \right) \times \\
\left( LIN(x_2, x_2'; \theta_2) + LIN(x_3, x_3'; \theta_3) \right) \times RBF(x_5, x_5'; \theta_5) + \\
\left( LIN(x_1, x_1'; \theta_1) \times LIN(x_4, x_4'; \theta_4) \right)
\]
\[ k_t(x, x') = \left( \left( \text{LIN}(x_1, x_1'; \theta_1) \times \text{LIN}(x_2, x_2'; \theta_2) \right) + \text{LIN}(x_9, x_9'; \theta_9) \right) \times \\ \text{RBF}(x_5, x_5'; \theta_5) + \text{LIN}(x_3, x_3'; \theta_3) \times \\ \text{LIN}(x_7, x_7'; \theta_7) \times \text{LIN}(x_{12}, x_{12'}; \theta_{12}) + \\ \left( \text{LIN}(x_8, x_8'; \theta_8) \times \text{LIN}(x_9, x_9'; \theta_{9b}) \right) \] (3.10)

These kernels were used to train surrogate models using the initial multi-fidelity dataset described in Section 3.2. The composite kernels were optimized using only the high-fidelity, but were applied to the entire multi-fidelity dataset. Finally, the surrogate models were explored using BO with the CEI utility function. This project used a moderate search granularity of 10,000 BO restarts. After ordering the results by CEI, five of the top thirteen candidate design were selected for high-fidelity URANS simulation. A variety of top candidates were selected to simulate both exploration and exploitation candidates. The results from these simulations are presented in Table 3.7.

<table>
<thead>
<tr>
<th>Candidate El Rank</th>
<th>Satisfy Constraints?</th>
<th>L/D</th>
<th>Percent Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>10.984</td>
<td>2.08%</td>
</tr>
<tr>
<td>2</td>
<td>Yes</td>
<td>10.943</td>
<td>1.70%</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>10.769</td>
<td>N/A</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>10.755</td>
<td>N/A</td>
</tr>
<tr>
<td>13</td>
<td>No</td>
<td>10.756</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Example Conclusion** This analysis proved the viability of using optimized composite kernels for multi-fidelity design optimization. In only one iteration the composite kernel produced a design that significantly improved upon the benchmark design.
Chapter 4

Conclusion

This thesis accomplished its two main goals: (1) create a primer on machine learning techniques applicable to engineering design problems including Gaussian process regression, multi-fidelity regression, Bayesian optimization, and composite kernel optimization and (2) demonstrate the effectiveness of an automated engineering design loop which includes machine learning techniques and physics-based solvers. My hope is that someone will read this thesis and, in a few hours, learn the topics it explains which took me months to grasp.

Since this field of research is promising and continuing to grow, there are plenty areas for future study. Specific to the topics discussed in this these, the following concepts are worthy of future study: (1) include an optimized kernel search within an automatic design loop, (2) incorporate multi-disciplinary optimization in an automatic design loop, and (3) compare the results of automatic design loops with different modeling and optimization methodologies.
Figure A-1: 1-D Example of Prior and Posterior Functions. In the upper panel are five random functions from a GP prior. In the lower panel are five random functions from the posterior and four noiseless observation points in blue.
Figure A-2: 1-D Example of Prior and Posterior Functions. In the upper panel are 20 random functions from a GP prior. In the lower panel are 20 random functions from the posterior and four noiseless observation points in blue. (Figure 2-1)
\[
y \sim \mathcal{N} \left( 0, \begin{bmatrix}
k_1(x_1, x'_1; \theta_1) + \sigma_1^2 I & \rho_1 k_1(x_1, x'_2; \theta_1) & \rho_1 \rho_2 k_1(x_1, x'_3; \theta_1) \\
\rho_1 k_1(x_2, x'_1; \theta_1) & k_2(x_2, x'_2; \theta_2) + \sigma_2^2 I & \rho_2 k_1(x_2, x'_3; \theta_2) \\
\rho_1 \rho_2 k_1(x_3, x'_1; \theta_1) & \rho_2^2 k_1(x_3, x'_2; \theta_1) + \rho_2 k_2(x_3, x'_3; \theta_2) & k_3(x_3, x'_3; \theta_3) + \sigma_3^2 I
\end{bmatrix} \right)
\]

Figure A-3: Multi-fidelity covariance matrix with three levels of fidelity: low-fidelity (level 1), medium-fidelity (level 2), and high-fidelity (level 3).

\[
y \sim \mathcal{N} \left( 0, \begin{bmatrix}
k_L(x_L, x'_L; \theta_L) + \sigma_L^2 I & \rho k_L(x_L, x'_H; \theta_L) \\
\rho k_L(x_H, x'_L; \theta_L) & \rho^2 k_L(x_H, x'_H; \theta_L) + k_H(x_H, x'_H; \theta_H) + \sigma_H^2 I
\end{bmatrix} \right)
\]

(A.1)

Figure A-4: Multi-fidelity covariance matrix with two levels of fidelity: (1) high-fidelity, h, and low-fidelity, l.
Bibliography


