Multi-Fidelity Aerodynamic Optimization Using Treed Meta-Models

Andrea Nelson,* Juan J. Alonso,†
and Thomas H. Pulliam‡

Stanford University, Stanford, CA 94305

The multi-fidelity Treed Meta-Model (TMM) framework developed and applied here creates a tree-based partitioning of an aerodynamic design space and employs independent Kriging surfaces in each partition to globally model computationally inexpensive low-fidelity analysis data. Using the low-fidelity meta-model to select points for evaluation with a high-fidelity analysis tool, a multi-fidelity global model is created from collocated high and low fidelity data points. A steady-state genetic algorithm is used to construct an optimal partitioning scheme based on the number of points in each partition and the combined predictive capabilities of the Kriging surfaces, as measured through cross-validation. The TMM framework mitigates the effects of the "curse of dimensionality" associated with surrogate modeling of large datasets, allows for parallelization of design space searches, and increases model flexibility by using a number of smaller Kriging surfaces. Uniform incremental sampling is incorporated to build the low-fidelity database progressively while the GA optimizes the partitioning scheme using available data. High-fidelity data points are chosen using a Least Angle Regression Scheme (LARS) to approximate the sensitivity of the meta-model to error in the low-fidelity points. The process is demonstrated from low-fidelity meta-model construction to multi-fidelity model optimization. The optimization includes direct searches on the design space using both the low and high-fidelity analysis tools, as well as searches of the low-fidelity and mulit-fidelity meta-models. The results will show not only the ability of the TMM framework to create a sufficiently accurate partitioned low-fidelity surface but also whether the selection of high-fidelity points will create a multi-fidelity model capable of locating a different optimal than the low-fidelity meta-model.

I. Introduction

In the area of aerodynamic optimization, the use of surrogate models or meta-models to reduce computational costs has quickly grown in popularity. Meta-models allow designers to conduct inexpensive searches of complex design spaces with the bulk of the computational expenses restricted to the initial database generation. As design problems grow in dimensionality however, the number of design evaluations needed to make useful meta-models creates a computational bottleneck in generating the model database. For most aerodynamic applications, designers are given a choice of analysis codes of varying computational expense to use in an optimization process. In direct search optimization, different analyses are used to reduce computation time, using low fidelity codes in the preliminary stages of optimization and higher fidelity codes in the final stages to verify optimal designs. The assumption that high-fidelity corrections are only needed near the low-fidelity extremum often leads a designer to a local high-fidelity extremum. However, this approach of using both high and low-fidelity data could be used to ease the database generation bottleneck of global meta-models. Creating a higher dimensional global meta-model to use in optimization requires using high-fidelity data in regions where the high and low-fidelity analyses differ, and low-fidelity data where they do not. This decision is not entirely straight forward and often requires user intuition or ad hoc methods.

*PhD student, Stanford University, anajn@stanford.edu
†Professor, Stanford University, jjalonso@stanford.edu
‡Associate Fellow AIAA, NASA Ames Research Center, tpulliam@mail.arc.nasa.gov

Addressing the primary computational bottleneck of database generation does not make one immune to the main drawback of meta-models, the so-called "curse of dimensionality," wherein the high dimensional design spaces require large populations of designs and cause the meta-model itself to become computationally intractable. Global meta-models can also fail when faced with design space discontinuities, a problem which becomes more frequent as aerodynamic design problems become more complex. Many methods attempt to address one or more of these problems, but few address all of them, and none have done so successfully.

In the past the meta-model curse of dimensionality has been handled successfully in the field of optimization, but usually in a way that leads the user away from an accurate global model. Trust-region methods reduce the size of the design space as an optimization algorithm progresses, creating a local meta-model that is more accurate.[1] These smaller models are less expensive in large design spaces, but the requirement of a coarse initial global model does not guarantee convergence to a global optimum. Other methods use global models that begin with a coarse data set and enrich the model by adding points along the search path of an optimization algorithm.[2] These methods also suffer from the inaccuracy of the initial coarse mesh, and the models can become ill-conditioned if search points are located too closely together.

The Treed Gaussian Process (TGP)[3] addresses a number of the problems common to meta-models. By partitioning the design space and fitting multiple Gaussian processes via Markov-chain Monte Carlo (MCMC) simulations, the TGP is adept at handling design space discontinuities. In addition it employs adaptive sampling which attempts to distribute resources more intelligently by balancing global exploration and local, feature-based exploration. The current implementation however, uses one large MCMC simulation to solve simultaneously for the models in all partitions, a method which becomes computationally expensive with large design populations.

The TGP approach presents a solution to many of the problems facing meta-model implementation. Partitioning presents an inherent ability to handling domain discontinuities as well as increased accuracy by allowing for different models in different areas of the domain. The curse of dimensionality, however, could be easilyt mitigated if each model were solved independently rather than as a whole system. When each model is solved independently, optimization on the partitioned domain becomes easily parallelizable, further reducing the computational cost of searching the design space. Adding the capability of multiple levels of solver fidelity would then create a framework that addresses all the major concerns facing efficient meta-model implementation.

The following is an implementation of the multi-fidelity Treed Meta-Model (TMM) framework, a modeling capability using multiple Kriging surfaces either for aerodynamic database creation or optimization. The TMM approach creates an optimal partitioning scheme of the design space, minimizing the global modeling error and computational cost of the separate Kriging surfaces. Though breaking up the data inherently leads to a loss of information in a global sense, in many cases partitioning the domain can create a more accurate model than a global Kriging with the ability to vary the local Kriging model variables applied in each area. By choosing the most appropriate local regression function and covariance function, the independent Kriging surfaces can model complex design spaces which often include regions of both well and ill-behaved data. Partitioning also allows handling of possible discontinuities in the data, and keeps the computational costs of a global model low by using smaller Kriging surfaces. During the optimization process within which the partitioning scheme is created, the TMM framework simultaneously builds the database of points on which the model is based. The parallel nature of the database construction and domain partitioning creates the ability to have an accurate low-fidelity global model by the time the database is sufficiently populated. In this manner, all available information is used to drive the evolution of the model while the usefulness of data points being generated is monitored. The process can then halt data point evaluations once augmentations to the database produce no significant improvement to the low-fidelity model. The multi-fidelity meta-model is created using a subset of the low-fidelity points and their corresponding high-fidelity responses. The high-fidelity subset is chosen using the Least Angle Regression Scheme (LARS)[4] to determine which low-fidelity design points cause the largest change in the low-fidelity global model if an error in the low-fidelity analysis is present. LARS obtains a lower order regression model of the meta-model sensitivity, which indicates a subset of the design points that are responsible for a majority of the response. Once the collocated multi-fidelity data have been evaluated with the high-fidelity analysis tool, the multi-fidelity global model is creating using the low-fidelity data as an input to the collocated high fidelity response. In this fashion, what was an N-dimensional design space in low-fidelity, using data locations as inputs and low-fidelity solutions as responses, turns into an N+1 dimensional space when the low-fidelity solutions are added as an input and the high-fidelity solution is the response. Though the multi-fidelity model is created using only the collocated

high and low-fidelity points, the entire low-fidelity dataset is used to evaluate the low-fidelity model which is needed as an input at any point the multi-fidelity response is desired.

The goal of this paper is to demonstrate the process from low-fidelity meta-model construction to multi-fidelity model optimization. The optimization example will include direct searches of a design space using the low and high-fidelity analysis tools TRAN2D and ARC2D, as well as searches using the meta-models constructed with the TMM framework. The comparison between the low-fidelity meta-model search and the direct search with TRAN2D will illustrate the ability of the low-fidelity meta-model to locate the TRAN2D optimal, an indication that the meta-model is a good approximation of the low-fidelity design space. The differences between the TRAN2D and ARC2D direct search results will demonstrate how comparable the high and low-fidelity optimal configurations are. The approximate high-fidelity optimal found by searching the multi-fidelity meta-model will then be compared to the optimal found by directly searching the space using ARC2D. The results will show not only the ability of the TMM framework to create a sufficiently accurate partitioned low-fidelity surface but also whether the selection of high-fidelity points will create a multi-fidelity model capable of locating a different optimal than the low-fidelity meta-model.

II. GA Partitioning Using Kriging Models

The design spaces of interest here are characterized as hypercubes of dimension N, where N is the number of design variables. The design variables function as inputs to the meta-model whose output is an approximation of an objective function. As we are studying aerodynamic databases, the objective function is related to the output of an aerodynamic analysis code. The partitioning of the hypercube domain is based on a binary tree structure that splits the design space perpendicular to the axes. Each partition is modeled with an independent Kriging surface based on the data within the partition boundaries. The predictive capability of each independent Kriging surface is determined via a validation of the surface, which is averaged over all the partitions to get a global figure of merit for each partitioning scheme. This figure of merit, along with a penalty function to restrict the number of points in each partition, is the objective function for a steady-state Genetic Algorithm,[5] which optimizes the partitioning tree structure to obtain an accurate, low-cost global model.

II.A. GA Design Variables

The design variables of the GA are encoded to completely describe any binary tree partitioning scheme. The number of design variables, or genes, is determined by the user based on the prescribed maximum depth of the tree structure. For each tree structure the genes that describe it are encoded as follows.

- Genes 1 through K determine upon which axis each split will occur, or whether no split will occur, with K equal to the total possible number of parent nodes based on the user-determined maximum tree depth.
- Genes $K+1$ through $2K+1$ determine where along the chosen axis each split will occur.

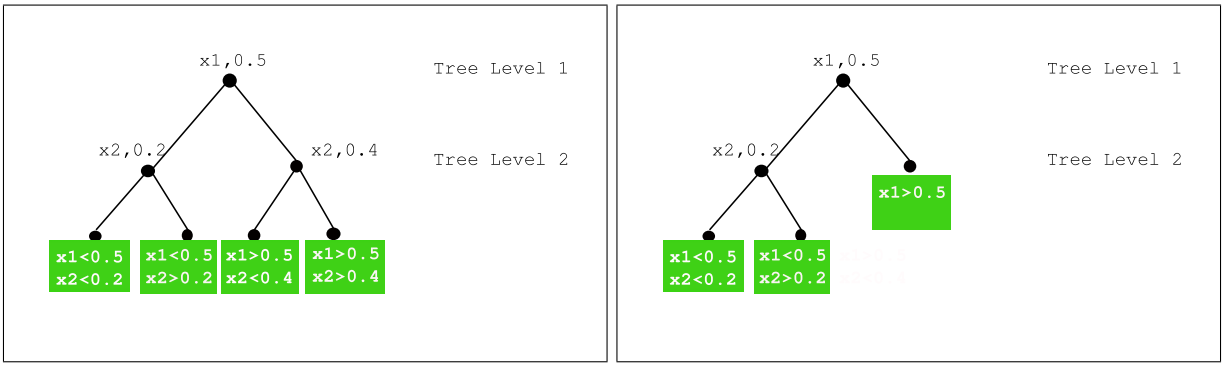
Figure 1(a) shows a tree of depth 2 with 4 possible partitions and 3 parent nodes.

The genes that determine along which axis to make each break are integers that vary from 0 to the number of axes in the design space, where a 0 value indicates no break at all. The genes that determine where along the chosen axis to make a break are integers that vary from 1 to the number of possible break locations. Due to the quick computational times of low-fidelity solvers, it is prudent to restrict the search space of the GA to aid in convergence to a good partitioning scheme. Limiting the break locations to lie on 10% intervals along an axis would give 9 possible break locations. The genes needed to create the tree in Fig. 1(a) would then be as follows:

$$[g1 = 1, g2 = 2, g3 = 2, g4 = 5, g5 = 2, g6 = 4] \tag{1}$$

The first gene indicates the first break will be along axis number 1, or the x1 axis. Accordingly, the second and third genes indicate the corresponding breaks will be along axis number 2, or the x2 axis. Genes 4 through 6 correspond to break locations at 50%, 20%, and 40%, respectively. The locations of the breaks are normalized, so with the fourth gene indicating a break location of 50%, the first break will be determined by

$$x_{break} = 0.5[x1_{max} - x1_{min}] + x1_{min} \tag{2}$$



(a) tree of depth 2 with 4 partitions

(b) tree of depth 2 with 3 partitions

Figure 1.

Subsequent breaks use the maximum and minimum values of the current partition in the above equation. For the tree in Fig. 1(a) the minimum and maximum values for both x1 and x2 are 0 and 1 respectively, so the actual breaks occur at x1=0.5, x2=0.2, and x2=0.4. If the design variables were modified to include a 0 for the third design variable instead of a 2, the tree in Fig. 1(b) would result. This tree shows that regardless of the values given for the break location, the axis along which to break the domain was given by the integer 0, which indicates no break is to occur.

II.B. Kriging Surface Construction

A Kriging surface, first developed in the geostatistics community,[6,7,8] models data based on the assumption that the responses of a set of data points are the outcome of a stochastic process, thus we can write the response vector Y as

$$Y = F\beta + e. \tag{3}$$

where F is a matrix of the underlying regression function on the specified design sites, and e is the white noise of a stochastic process. It is assumed that

$$E[e] = 0, \quad E[ee^T] = \sigma^2 R$$

Where $E[.]$ denotes the expected value, and R is the covariance matrix given by a correlation function between design sites. The Kriging model is created by solving for β and e with a generalized least squares fit given F and R.

The underlying regression function is chosen to be a polynomial of either zero, first or second order. The matrix F is then defined as follows[9]

Constant, $p = 1$:

$$f_1(x) = 1, \tag{4}$$

Linear, $p = n + 1$:

$$f_1(x) = 1, f_2(x) = x_1, ..., f_{n+1}(x) = x_n, \tag{5}$$

Quadratic, $p = \frac{1}{2}(n+1)(n+2)$:

$$\begin{array}{lll} f_1(x) = 1 & & \\ f_2(x) = x_1, & ..., & f_{n+1}(x) = x_n \\ f_{n+2}(x) = x_1^2, & ..., & f_{2n+1}(x) = x_1 x_n \\ f_{2n+2}(x) = x_2^2, & ..., & f_{3n}(x) = x_2 x_n \\ ... & ..., & f_p(x) = x_n^2. \end{array}$$

The correlation functions chosen for this work are in the form of products of 1-D correlations,[9] such that the matrix R is given as

$$R(\theta, d) = \prod_{j=1}^{n} R_j(\theta, d_j), \tag{6}$$

where θ is a range parameter and d is the distance between data points. The 1-D correlations used here are as follows:

Name	$R_j(\theta_j, d_j)$
Exponential	$exp(-\theta_j \mid d_j \mid)$
Gaussian	$exp(-\theta_j \mid d_j \mid^{\theta_{n+1}})$ $\quad 0 < \theta \leq 2$
Linear	$max(0, 1 - \theta_j \mid d_j \mid)$

The θ values in each dimension are chosen by the Kriging model algorithm, using an upper and lower bound and a pattern search optimization algorithm[10] that solves

$$min_\theta\{\psi(\theta) = \mid R \mid^{\frac{1}{m}} \sigma^2\}. \tag{7}$$

The choice of regression and covariance function is performed based on a reduction of $\mid R \mid^{\frac{1}{m}} \sigma^2$. Different regression functions and covariance functions are used in each partition of the design space, something that is not possible with traditional global Krigings.

The mean squared error of the Kriging predictor is given by the following,

$$s^2(x^*) = \sigma^2[1 - r'R^{-1}r + \frac{(1 - \mathbf{1'R^{-1}r})^\mathbf{2}}{\mathbf{1'R^{-1}1}}]. \tag{8}$$

where $\mathbf{1}$ is a vector of ones. The root mean squared error, or s is often referred to as the Kriging predicted standard error.

II.C. GA Objective Function

The objective function minimized by the GA is a combination of the Kriging predicted standard error and the 'standardized cross validated residual' (SCVR).[11] The SCVR is calculated by 1-off cross validation, meaning that data points in each partition are removed one at a time and a new data model is created with the remaining data. The new data model is used to predict the point that was removed from the dataset, and the comparison of the predicted and actual values is used to create a figure of merit for the model. With each prediction, the Kriging predictor returns as estimate of the standard error. The Kriging model relates to the standard error as a confidence interval, indicating there is a 99.7% certainty that the true value lies within a bound of +/-3 times the standard error. For each data point removed in a 1-off cross validation, the number of standard errors that the actual value is above or below the predicted value is the SCVR at that point. This figure of merit is calculated using the following equation:

$$SCVR_i = \frac{y(\mathbf{x}^{(i)}) - \hat{y}_{-i}(\mathbf{x}^{(i)})}{s_{-i}(\mathbf{x}^{(i)})} \tag{9}$$

In the above equation y is the actual value, $\hat{y}$ is the predicted value, and s is the predicted standard error. The -i subscript indicates a value resulting from a Kriging model that was not constructed using $y(\mathbf{x}^{(i)})$. According to Jones et.al.,[11] the absolute value of the SCVR should be below 3 in order for the model to be considered accurate. However, if the Kriging predicted standard error is high, we may have a large confidence interval and a bad model with an acceptable SCVR. In order to ensure the GA drives the partitioning to create a model with both good predictive properties and a tight confidence interval, the following equation is used as the objective function.

$$obj = \frac{1}{M}\sum_{k=1}^{M}\frac{1}{N_k}\sum_{i=1}^{N_k}(F(\mathbf{x}^{(i)}))$$

$$F(\mathbf{x}^{(i)}) = \begin{array}{l} s_{-i}(\mathbf{x}^{(i)}) + |SCVR_i| \\ s_{-i}(\mathbf{x}^{(i)}) \end{array} \quad \begin{cases} if & |SCVR_i| > 3 \\ if & |SCVR_i| \leq 3 \end{cases}$$

Where M is the number of partitions and N_k is the number of points in partition k.

With a traditional 1-off cross validation, every point in the dataset is removed one at a time and the figure of merit is averaged over the number of data points. However, this implies that for each partitioning scheme the Kriging surface must be generated N times, where N is the number of data points. This can be computationally expensive with large datasets, leading us to use a modified 1-off cross validation. The modified cross validation takes only a sample of the dataset, say 20 points in each partition, and uses these 20 points as the 1-off test points for cross validation. A drawback to using only a small sample of the data points for a 1-off cross validation is a less accurate representation of the true figure of merit. If the best member of the GA is reevaluated and a different set of data points are chosen to evaluate the Kriging figure of merit, the GA objective function may change which presents a hurdle for the convergence of the GA. In order to ensure convergence, the best GA population member is periodically reevaluated with a new sampling of data for the 1-off cross validation and the new objective function is averaged with the old objective function. In this manner, the best GA population member will asymptotically approach the value of a complete 1-off cross validated objective function as the GA evolves.

To keep the GA from choosing unfit partitioning schemes, a penalty function is applied to the number of points in each partition. If there are too few points, the Kriging surface will be ill-conditioned. If there are too many points the Kriging surface will be computationally expensive. The penalty function is a multiplicative penalty using a probability density function given by the following

$$P(M) = pdf(M, \mu, \sigma^2)/pdf(\mu, \mu, \sigma^2) \tag{10}$$

where M is the number of points in the current partition, μ is the desired number of points, which is 200 for the examples below, and σ^2 is the standard deviation, which is chosen to be 50 points. When the number of points in the parition is μ, $P = 1$, and $P > 1$ otherwise.

II.D. Partitioning Results

Major concerns related to partitioning the domain include lack of information in forming the local Kriging surfaces and discontinuities at the partition boundaries. To explore the modeling capabilities of the Treed Meta-Model, 2D and 4D databases were generated as example design spaces using the analysis codes TRAN2D and ARC2D described in section V, respectively. A fine-grid database was generated to represent the "true" surface, and a coarse-grid database was used to create the meta-model. The 2D coarse-grid database used to create the model was 500 points and while the 4D database was 2000 points. The 2D design space was modeled with both a TMM and a single Kriging, to illustrate the differences with the two approaches. The 4D database was only modeled with the TMM, as the coarse-grid database was beyond the capabilities of a single Kriging. The 2D meta-model surfaces are shown in Figs. 2 and 3. The "true" 2D surface, created with 1470 points is shown in Fig. 4. Both the treed Kriging and single Kriging surfaces model the true surface very closely, and there are no significant discontinuities at the partition boundaries of the TMM model. Slices of the 4D design space are shown in Figs. 5 thru 10. Again the treed Kriging surface slices compare well with the true surface, and do not show discontinuities at the partition boundaries. The mean squared errors of the 2D surfaces and 4D slices are given in Table 1. The mean squared errors of the 2D TMM and single Kriging surfaces are low and compare well with each other, further illustrating that both surfaces do a good job of modeling the true surface. The errors of the 4D slices are slightly higher than the 2D errors, but still indicate the 4D model compares well with the true 4D surface. In all of the plots showing the treed meta-models there are no significant discontinuities at the partition boundaries, most of the surface regions spanning the boundaries are quite smooth.

III. Database Creation

Creating the appropriate partitioning scheme is only half the battle of creating a good low-fidelity global model. If the database being used to create the model is too sparse or does not contain the interesting features of the design space, the resulting model will fail to resolve desirable global features. In order to concurrently build the database during the partitioning process it is necessary to incrementally add points in an intelligent way such that the generated surface converges toward the true surface.

Typical methods for incrementally harvesting data used to create surrogate models for optimization purposes include two types, those that employ available information about the surface, and those that do

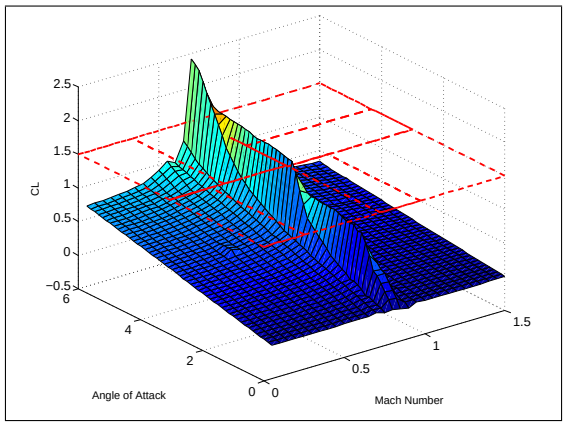


Figure 2. TMM using 500 Point Database

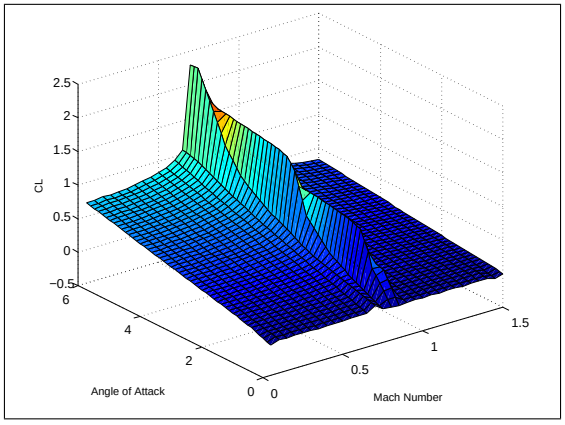


Figure 3. Single Kriging Model using 500 Point Database

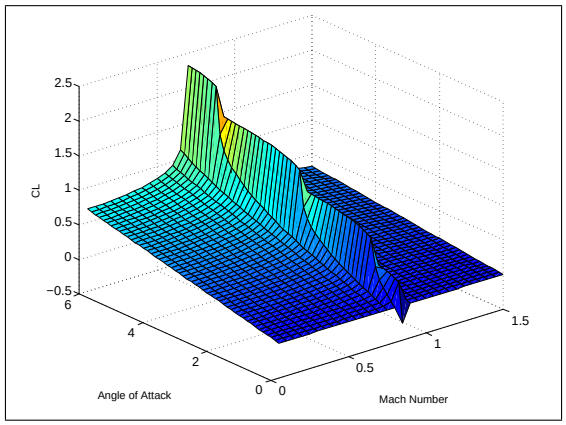


Figure 4. True 2D surface (obtained with 1470 Points)

Surface	MSE
2D Single Kriging	0.00072093
2D TMM	0.00083275
4D Slice #1, TMM	0.0016
4D Slice #2, TMM	0.00063803
4D Slice #3, TMM	0.0024

Table 1. MSE of 2D and 4D Slice surfaces

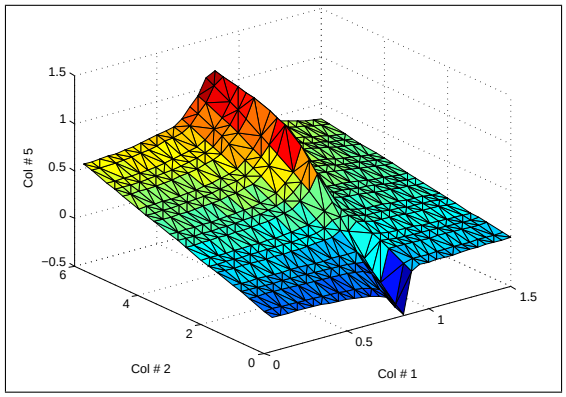


Figure 5. Slice #1 of 4D Surface (106,800 Points)

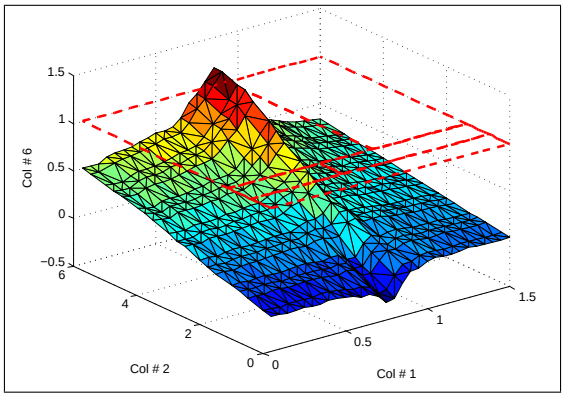


Figure 6. Slice #1 using TMM (2000 Points)

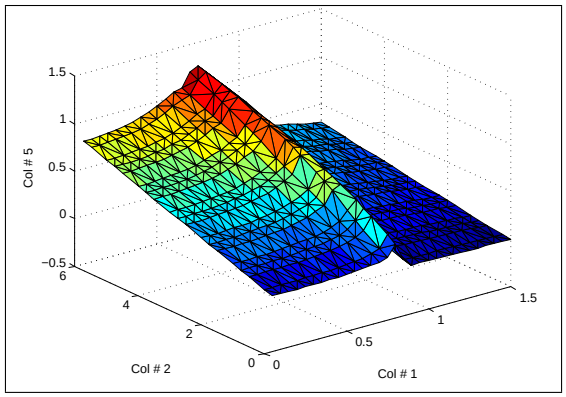


Figure 7. Slice #2 of 4D Surface (106,800 Points)

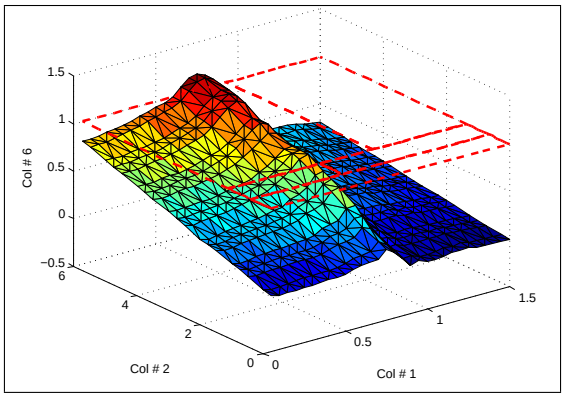


Figure 8. Slice #2 using TMM (2000 Points)

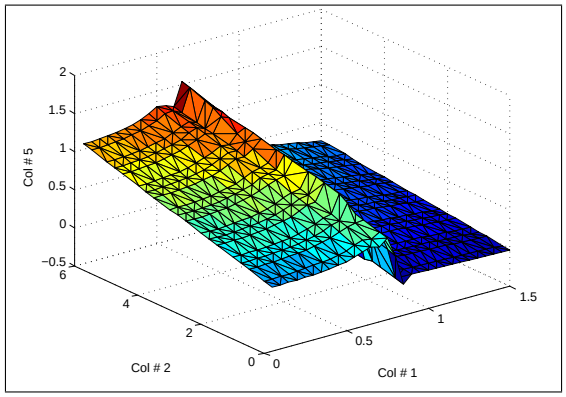


Figure 9. Slice #3 of 4D Surface (106,800 Points)

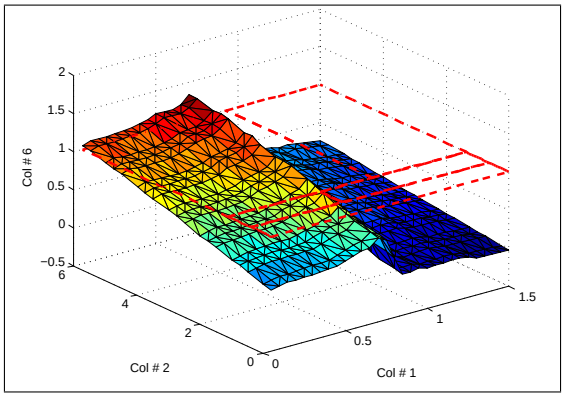


Figure 10. Slice #3 using TMM (2000 Points)

not. Methods of choosing data points that do not require information about the surface focus on sampling in a uniform manner (Latin hypercube, Monte-Carlo, etc), while those that do use available information tend to focus on locations of possible extrema (Expected Improvement Function[11]). In order to build an accurate database from the ground up, the primary stages of data harvesting should focus on uniform sampling. Once the addition of uniform data sets no longer improves the surface, the sampling method should switch to locating extrema, or what is considered "adaptive sampling".

III.A. Uniform Sampling

When dealing with large, complex design spaces, it is often unclear how many points may be necessary to resolve key features with a response surface. Incrementally adding data can create an opportunity to monitor resolution convergence, if the design sites are appropriately chosen. Ideally the addition of any points would improve the predictive capabilities of a response surface. New points that are chosen very near existing points, however, can produce little or no improvement. In some cases clustered points that are too close together can cause the Kriging matrix to become ill-conditioned. To get the maximum amount of information out of a minimum number of points with no a priori knowledge of the design space, requres a uniform sampling.

Several sampling methods are capable of producing relatively uniform samples, e.g. Latin Hypercube sampling, however not many methods allow incremental uniform sampling. Latin Hypercube sampling requires a priori knowledge of how many points are desired in order to divide the domain into the appropriate number of hypercubes. This method creates nearly uniform point distributions, but requires a completely new set of data if additional points are desired. In this fashion, true Latin Hypercube sampling does not allow for incremental point addition unless the maximum number of samples is known. Romero, et. al.[12,13] examine a number of uniform sampling techniques for progressively building response surfaces and measuring the convergence properties of those surfaces created with different sampling techniques. They note two distinct types of these sampling methods, those with fixed increment addition and those with

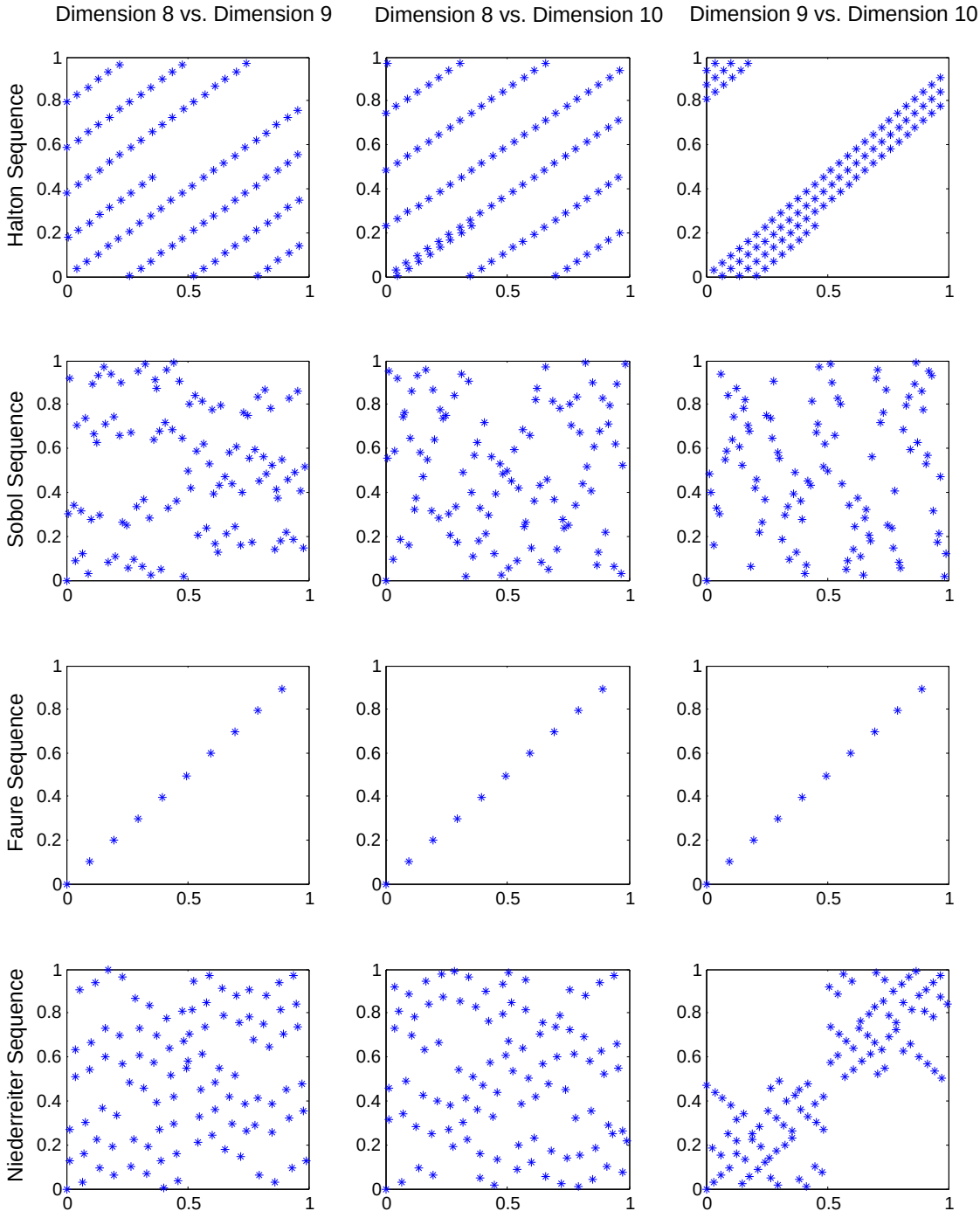


Figure 11. Digital Sequence Point Projections

flexible increment addition. Flexibility is desirable when we move to parallelization of the framework, as the number of available processors will not always match the number of points needed for each incremental sample. The ideal sampling method would create a uniform sample and maintain that uniformity regardless of the number of points added in each incremental update. Romero, et. al. describe the Halton sequence used in quasi-Monte Carlo sampling as having this property, but it can be shown[14] that when this sequence is extended to 10 dimensions, the uniformity in higher dimensions is only maintained if data increments are very large. Owen[14] describes other sampling methods used in quasi-Monte Carlo sampling called digital sequences that maintain more uniformity in higher dimensions (Sobol',[15] Faure,[16] and Niederreiter[17]).

Ideally we would like to choose a sampling scheme that demonstrates the uniformity of Latin Hypercube sampling, while allowing incremental sampling. With very large point samplings, all four sequences, Halton, Sobol, Faure, and Niederreiter, approach uniformity in both low and high dimensions, but this does not necessarily lend them to incremental sampling. Figure 11 shows the higher dimensional projections of the only the first 100 points of the Halton, Sobol, Faure and Niederreiter sequences on a unit hypercube of dimension 10. Visual inspection of these sequences shows that the Sobol sequence achieves the most uniform projection of the points in higher dimensions. This is an indication that the Sobol digital sequence will produce the most uniform samples on an incremental basis compared to the other sequences studied.

In order to compare the Sobol sequence with the known uniformity of a Latin Hypercube sampling, each method is used to generate a list of 500 points. This list is then incrementally sampled from and used to create a 4D global Kriging surface. The convergence of the Kriging surface toward the actual surface should depend largely on the uniformity of the sample data. Due to the stochastic nature of Latin Hypercube sampling, the Latin Hypercube list is generated 5 times using a different seed, and the results averaged to give a general figure of convergence. The convergence is predicted using the mean squared error of 500 randomly selected points as predicted by the Kriging surface. These points are held constant for both the Sobol sequence and each of the Latin Hypercube trials. Results of this convergence study are shown in Fig. 12. The comparison of the two methods show that the uniformity of the Sobol sequence and Latin Hypercube sampling are comparable in 4 dimensions.

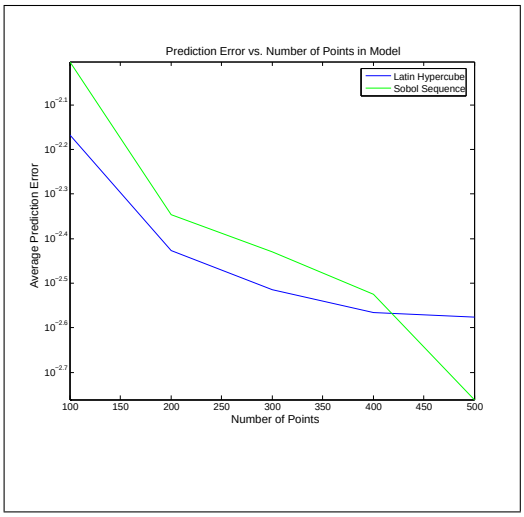


Figure 12. Prediction Error, Latin Hypercube Sampling and Sobol Sequence

IV. Parallel Implementation

The goal of any parallel implementation is proper load balancing such that all available processors are continuously busy. As described above, there are two main tasks in creating a treed data model: populating a database and partitioning the domain. Using a low-fidelity analysis tool for database creation, the data evaluations will be relatively quick, which leads to an equal subdivision of available processors for the two main tasks. Within each subdivision there is one processor that is designated as the "manager," responsible for data collection and convergence checks. The other processors are "workers," responsible for data analysis. Before the process begins the database is initialized with the first uniform sample.

Once the initial database has been evaluated, the GA is initialized and the GA worker processors begin evaluating population members. The processors creating the database are continuously evaluating uniformly sampled data points until the surface has converged and the GA processors have issued a stop command. The GA process and the addition of data points is completely independent. That is to say, the GA processor group does nothing when new points are added to the database by the data evaluation group, except to continue on with the GA. The current database resides in a file in the GA directory and is only called when a GA worker evaluates an objective function, which means at any point in the GA process, some population members could be out-of-date, having been evaluated with a previous dataset. In an effort the keep the GA converging to the optimal solution for the current database, the best population member is reevaluated each time new data are added to the dataset. If the best population member is no longer the best after this

reevaluation, the population is resorted and the new best member is also reevaluated with the new dataset. This updating procedure keeps the population members with the most influence on the evolution of the entire population are as current as possible.

V. Aerodynamic Analysis Codes

Using both high and low-fidelity analysis codes to form a multi-fidelity model eases the computational burden of populating a large database. The low-fidelity analysis code used here, TRAN2D, is based on a nonlinear full potential solver with similar capabilities to the late 1970's codes, TAIR[18] and FLO36.[19] The nonlinear full potential code is applied to two-dimensional (2D) airfoils on curvilinear meshes. Transonic full potential codes are very efficient, as compared to say Euler equation codes, but their solutions are limited in 2D to flows with pre-shock Mach numbers, $M_C \leq 1.3$. Above that critical limit Full Potential results produce inaccurate shock locations and therefore incorrect loads. The high-fidelity analysis code used here is ARC2D,[20] which solves the Euler or Navier-Stokes equations. The Euler analysis mode for airfoils in curvilinear coordinates is chosen here. Euler solutions do not suffer from the inaccuracies at high pre-shock Mach numbers and can produce very accurate results across the full subsonic to supersonic range. Details of these two methods and codes are available in numerous publications.[18,19,20] The homegrown transonic code, TRAN2D, was chosen because its numerical implementation is very close to that found in ARC2D. Therefore the two analysis codes will produce consistent results in the region of full potential applicability. Both analysis codes are applied to the same "C" mesh curvilinear grid, at the same flow conditions. The mode of operation and solution process are made to be consistent in terms of accuracy and solution convergence. Typically transonic full-potential codes are on the order of 10-20 times faster than an Euler code. Therefore, we characterized our low-fidelity analysis as being much faster than the high-fidelity analysis.

The airfoil shapes generated for analysis with TRAN2D and ARC2D are determined using 10 "PARSEC"[21] input parameters as shown in Fig. 13: leading edge radius, r_{le}, upper and lower crest location (chordwise locations, X_{up} and X_{lo}, and thickness values, Z_{up} and Z_{lo}), upper and lower crest curvature, Z_{XXlo} and Z_{XXup}, trailing edge coordinate, Z_{TE}, direction, α_{TE}, and wedge angle, β_{TE}.

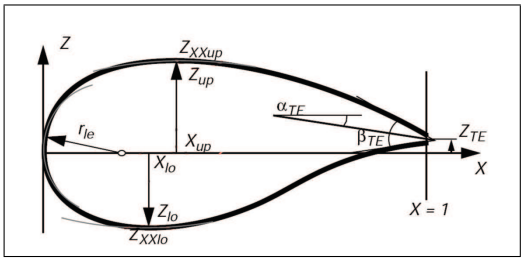


Figure 13. PARSEC airfoil geometry

V.A. Multi-Fidelity Meta-Model Examples

The examples included here demonstrate the convergence of the single-fidelity meta-models with the progressive uniform sampling, and the evaluation times required for obtaining a predictive mean squared error. In addition they extend the single-fidelity meta-models to multi-fidelity and analyze the predictive capability with the high-fidelity analysis tool.

As the uniform sampling progressively adds points to the low-fidelity database, the meta-model should converge to the true surface, reducing the predictive mean squared error (MSE). The predictive capability of each of the meta-models is measured by predicting the response on a coarse mesh and comparing the predicted values with the corresponding analysis tools to obtain a predictive MSE. In the first example, the domain is small enough to allow creation of a single Kriging surface with which the treed meta-model can be compared. The evaluation times of obtaining the predictive MSE are shown to analyze the cost benefit of partitioning the domain.

V.A.1. Two-Dimensional Meta-Model

This example investigates the 2D design space produced by analyzing a constant shape airfoil at varying M_∞ and α with data produced using both TRAN2D and ARC2D. The objective function is the lift coefficient of the airfoil, C_L.

The convergence of the 2D Treed meta-model is shown in Fig. 14 in terms of predictive MSE vs. number of points added to the database via progressive uniform sampling. The additional curves in Fig. 14 depict the convergence of a single Kriging surface and a treed meta-model with artificially created partitions that equally divide the design space. These curves are included to compare the convergence of no partitioning and an arbitrary partitioning scheme with the TMM partitioning created with the GA. Both partitioning

schemes converge faster than the single Kriging, indicating paritioning of the domain gives the meta-model more freedom, using different Kriging regression and correlation functions, to accurately model the surface. In addition, the partitions created with the GA show a faster convergence than the artificially created partitioning, illustrating importance of optimizing the partitioning scheme. The wallclock time, in seconds, required for one processor to construct the model can be seen in Fig. 15. Again the 2D TMM is compared to a single Kriging surface and the artificially partitioned surface. Both partitioned surfaces show comparable computation times, which are nearly an order of magnitude savings in cost over the single Kriging surface. The MSE plots can also be used to determine at which point the surfaces are converged in order to chose the database from which the high-fidelity points will be chosen. The 2D TMM surface converges with a uniform database of 600 points.

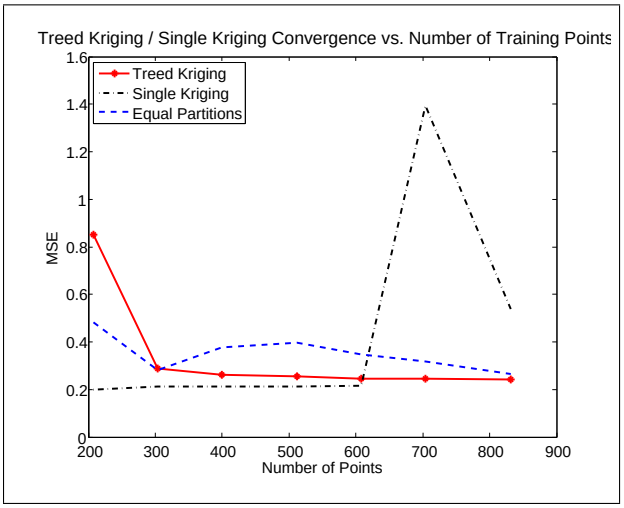


Figure 14. MSE vs. Number of Points

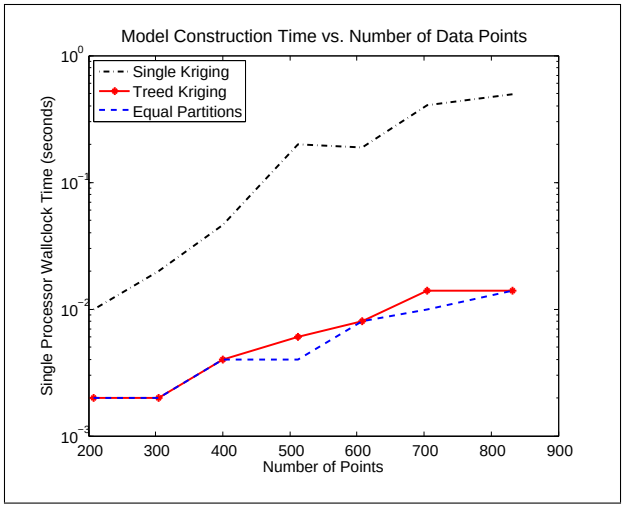


Figure 15. Computation Time for Constructing Meta-model

With the converged low-fidelity meta-model available, the Least Angle Regression Scheme was performed producing a 34-point high-fidelity subset. The low and high-fidelity 2D surfaces are shown in Figs. 16 and 17, while the multi-fidelity surface created with the 34 collocated high and low-fidelity points is shown in Fig. 18. The low and high-fidelity surfaces show significant differences at the peak near the transonic region around $M = 1$. Though the multi-fidelity surfaces is slightly noisy at the peak, it captures the magnitude of the high-fidelity surface and the approximate shape. The mean squared errors in Table 2 were calculated by using the low and multi-fidelity models to predict the high-fidelity surface at 500 test points. The accuracy of the multi-fidelity meta-model in predicting the high-fidelity response is reflected in the MSE, which is nearly an order of magnitude lower than the MSE of the low-fidelity meta-model in predicting the high-fidelity response.

V.A.2. Four-Dimensional Meta-Model

This example extends this 2D design space presented above to 4 dimensions by introducing Z_{up} and Z_{lo} as additional design variables and keeping the other PARSEC variables constant. For the 4D example design space, a single Kriging surface was not constructed due to the number of points needed in the database. The MSE and model construction time of the 4D Treed meta-model are shown in Figs. 19 and 20. The timing curves of the 2D TMM and single Kriging from the example above are included in Fig. 20 for comparison. The computation time of the 4D TMM continues along the same trend as the 2D TMM with significantly lower cost than the 2D single Kriging surface.

The 4D model appears to be converged at 1600 points but the MSE dips down again at 1800 points, therefore the 1800-point database will be used for the low-fidelity meta-model. Performing LARS on the converged low-fidelity database created a 201-point subset. Slices of the 4D multi-fidelity model created with 201 collocated points are shown in Figs. 23 and 24, along with the true slices shown in Figs. 21 and 22. The multi-fidelity slices shown in Figs. 23 and 24 capture the general shape of the true surface but do show some rounding off at the peaks and noise near the domain boundary. The mean squared errors in Table 2, calculated by using the low and multi-fidelity models to predict the high-fidelity surface at 500 test points,

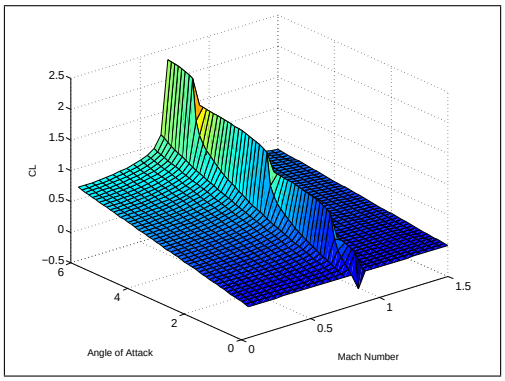


Figure 16. True Low Fidelity Surface (1470 Points)

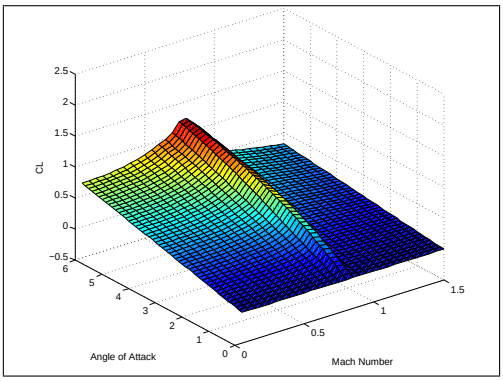


Figure 17. True High Fidelity Surface (1470 Points)

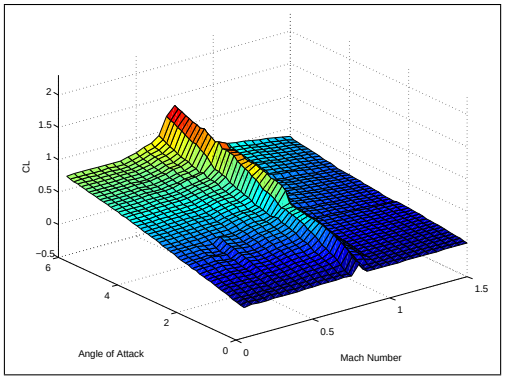


Figure 18. TMM Surface,34 High and Low-Fidelity points

TMM Surface	High-Fidelity MSE
2D Single-Fidelity, 608 Low-Fi points	0.0248
2D Multi-Fidelity, 34 Hi/Low-Fi points	0.0035
4D Single-Fidelity, 1800 Low-Fi points	0.0185
4D Multi-Fidelity, 201 Hi/Low-Fi points	0.0041

Table 2. MSE of 2D and 4D surfaces

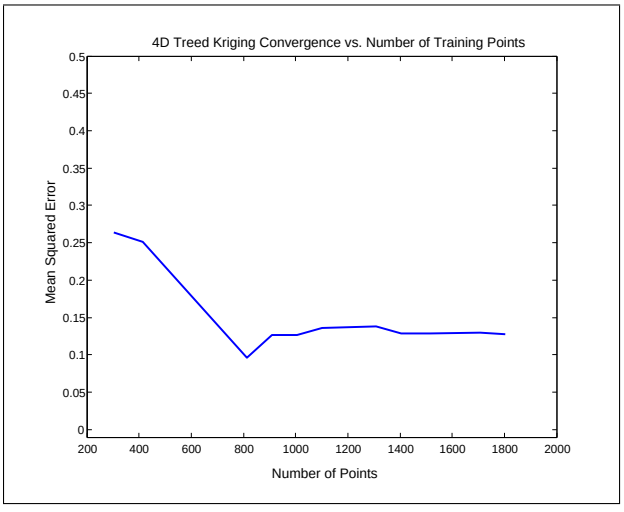


Figure 19. MSE vs. Number of Points

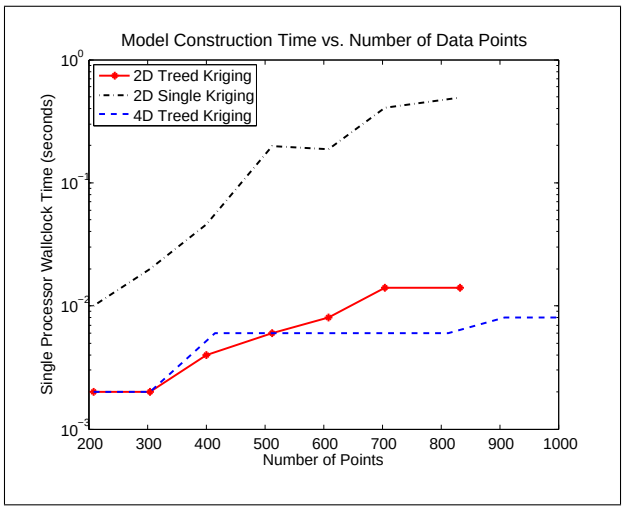


Figure 20. Evaluation Time for Constructing Meta-model

indicate that the multi-fidelity surface is still nearly five times more accurate than the low-fidelity model at predicting the high-fidelity response.

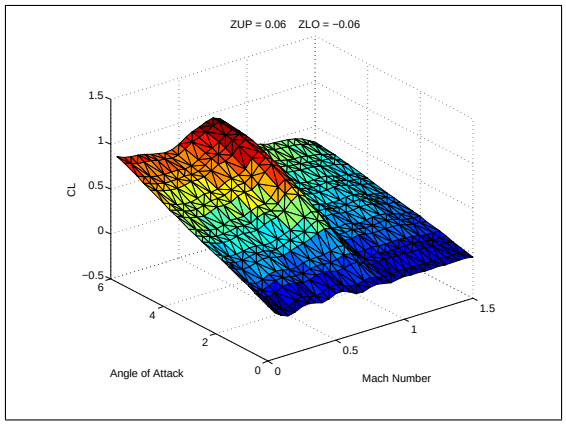


Figure 21. 4D TMM Slice #1 (201 collocated Points)

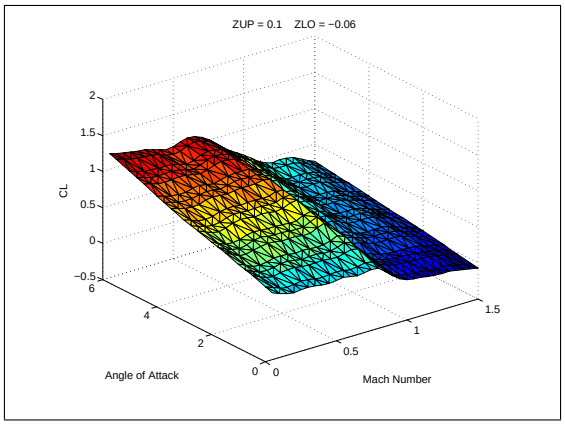


Figure 22. 4D TMM Slice #2 (201 collocated Points)

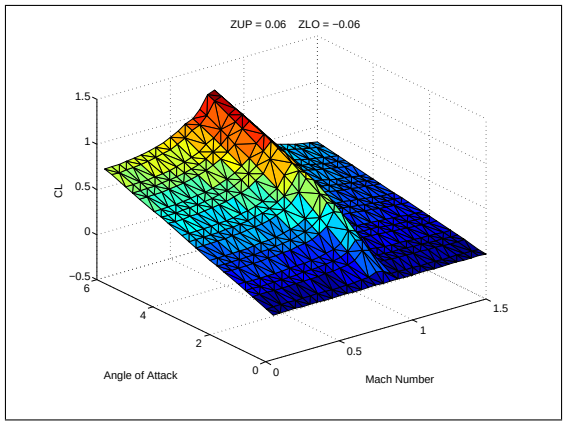


Figure 23. 4D True High Fidelity Slice #1

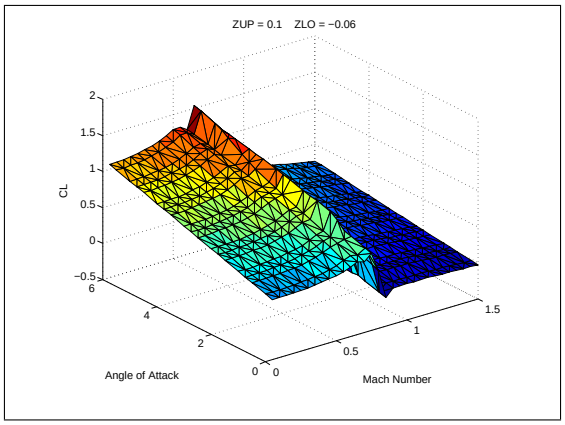


Figure 24. 4D True High Fidelity Slice #2

V.B. Meta-Model Optimization Example

This example keeps the flow conditions fixed at $M_\infty = 0.7$ and $\alpha = 2$. These conditions were chosen such that for the class of airfoils being investigated, there will be regions of the design space where TRAN2D and ARC2D are not consistent. The 4 design variables chosen to control the shape are X_{lo}, X_{up}, Z_{XXlo}, and Z_{XXup} . All other PARSEC variables are held constant.

The objective function for this optimization problem is based on both the lift and drag of the airfoil and a thickness constraint, defined by the following equation

$$f = (1 - \frac{C_L}{C_L^*})^2 + C_D + t_c \tag{11}$$

Where C_L^* is the target C_L, set at 0.55, and t_c is the thickness constraint. The thickness constraint is a soft constraint that imposes a penalty on the objective function if the average airfoil thickness is below some nominal value.

The goal here is to demonstrate the process from low-fidelity meta-model construction to multi-fidelity model optimization. Beginning with a direct search on the design space using a generational GA and the low-fidelity analysis tool TRAN2D, results are compared to an optimization performed on the low-fidelity meta-model combined with a steady-state GA. This comparison will illustrate the ability of the low-fidelity meta-model to locate the low-fidelity optimal, an indication that the low-fidelity model is a good approximation of the low-fidelity design space. The design space is then searched directly using a generational GA and the high-fidelity analysis tool ARC2D. This will demonstrate how comparable the high and low-fidelity optimal configurations are. The final step is to locate an approximate high-fidelity optimal

by searching the multi-fidelity meta-model with a steady-state GA and comparing it with the high-fidelity optimal found by directly searching the space using ARC2D. The results will show not only the ability of the TMM framework to create a sufficiently accurate partitioned low-fidelity surface but also whether the selection of high-fidelity points will create a multi-fidelity model capable of locating a different optimal than the low-fidelity meta-model.

V.B.1. Meta-Model Construction

The convergence of the partition optimizing steady-state GA, shown in Fig. 25, indicates the low-fidelity database is sufficiently populated once 1800 uniform points have been evaluated. The predictive capability of the low-fidelity model is then measured by comparing the model to 500 randomly selected test points evaluated with both the low and high-fidelity analysis tools. The calculated mean squared error of the low-fidelity model for predicting both the low and high-fidelity responses are shown in Table 3. The data indicates that the low-fidelity model is more accurate when predicting the high-fidelity response than the low-fidelity response. It is important to keep in mind that the low-fidelity analysis tool is not valid in all areas of the design space, leading to a noisy response that is more difficult to model. Adaptive sampling could be used to obtain a more accurate low-fidelity model, but that step is not currently built into the TMM framework and was not necessary for this optimization example.

The creation of a multi-fidelity model is aimed at predicting the high-fidelity response using collocated high and low-fidelity data points obtained with a knowledge of only the low-fidelity surface. If the points are chosen intelligently, the multi-fidelity model should show improvement on the predictive capabilities of the low-fidelity model. Using the 1800-point low-fidelity database, LARS produced a 143-point subset which was subsequently evaluated with the high-fidelity analysis tool. Using the location of the collocated points and the low-fidelity responses at each point as inputs, a 5-dimensional multi-fidelity model was created. The predictive capability of the multi-fidelity model was measured using the same 500 test points as was used with the low-fidelity model. The mean squared errors of the multi-fidelity model for predicting the low and high-fidelity responses are shown in Table 3. The data indicates that the multi-fidelity model is better at predicting both responses than the low-fidelity model, but is more accurate at predicting the high-fidelity response.

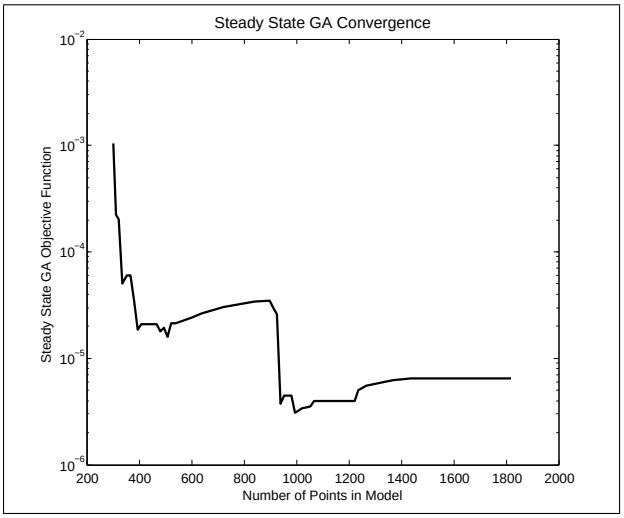


Figure 25. Convergence of Steady-State GA Objective Function

Model	Response	MSE
Low-Fidelity Meta-Model	Low-Fidelity Response	0.0596
Low-Fidelity Meta-Model	High-Fidelity Response	0.0297
Multi-Fidelity Meta-Model	Low-Fidelity Response	0.0226
Multi-Fidelity Meta-Model	High-Fidelity Response	0.0054

Table 3. Meta-Model Predictive MSE

V.B.2. Optimization Results

Following the construction of both a low-fidelity treed meta-model and a multi-fidelity meta-model, the optimzation begins by locating the low-fidelity optimal design by searching the design space directly using TRAN2D. The resulting design variables and Mach contours as analyzed by both TRAN2D and ARC2D, are shown in Table 4 and Figs. 26 and 27, respectively. The TRAN2D and ARC2D solutions for the low-fidelity

optimal are similar, which is expected given the pre-shock Mach number. Accordingly the objective function found with TRAN2D and ARC2D for this configuration are nearly equal.

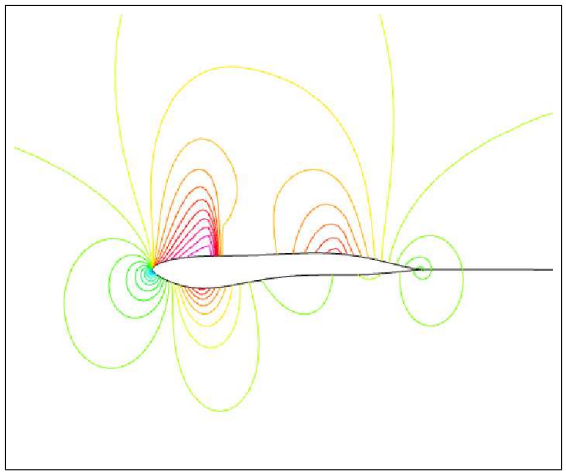


Figure 26. TRAN2D Direct Search Optimal Analyzed with TRAN2D

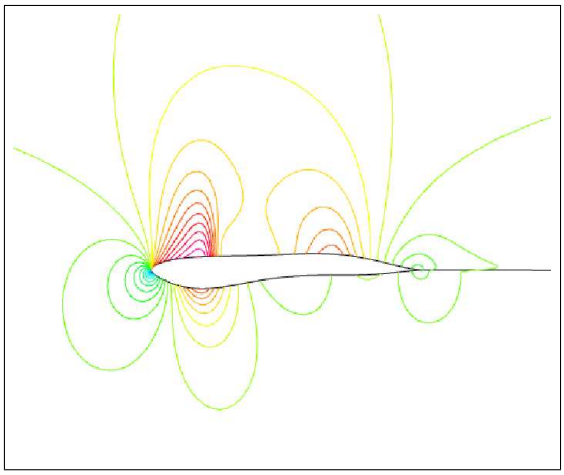


Figure 27. TRAN2D Direct Search Optimal Analyzed with ARC2D

X_{up}	Z_{XXup}	X_{lo}	Z_{XXlo}
0.606	-0.719	0.700	0.150

	Obj	M_C	C_L	C_D
TRAN2D	0.01071	1.29	0.56178	0.005247
ARC2D	0.01099	1.30	0.56044	0.004786

Table 4. TRAN2D Direct Search Optimization Results

The next step is to search the low-fidelity TMM to locate an approximation of the low-fidelity optimal and compare it with the true optimal found above. Mach contours as analyzed by both TRAN2D and ARC2D, are shown in Table 5 and Figs. 28 and 29, respectively. Although the MSE of the low-fidelity TMM was not as low as expected, the optimal configuration found using the meta-model is close to that found with TRAN2D. In addition the TRAN2D and ARC2D solutions are similar, indicating we are still in a region where the low and high-fidelity analysis tools agree. This is confirmed by noting the TRAN2D pre-shock Mach number is less than 1.3, leading to similar objective functions found with TRAN2D and ARC2D.

To determine whether the low-fidelity and high-fidelity optimal configurations are in the same location, a direct search of the design space using ARC2D is used to produce the optimal design variables in Table 6. The generational GA was run for 100 generations with 32 chromosomes before reaching convergence, requiring roughly 3000 ARC2D evaluations. The Mach contours obtained from the TRAN2D and ARC2D analyses of the resulting airfoil are shown in Figs. 30 and 31, respectively. The airfoil found with ARC2D is significantly different to the one found with the TRAN2D optimization above, which is explained by comparing the objective functions computed for this configuration using the two solvers. The objective function calculated with TRAN2D for this airfoil is significantly higher than the objective function calculated with ARC2D. This is a result of the pre-shock Mach number, which is outside the range of validity for TRAN2D. Accordingly the Mach contours shown from the analysis of the airfoil with TRAN2D and ARC2D show significant differences, as do the resulting values of C_L and C_D. Thus the high-fidelity optimal is in a different location than the low-fidelity optimal, and resides in a regions where TRAN2D is not applicable.

The final step is to search the multi-fidelity meta-model. The optimal configuration found with the multi-fidelity meta-model is shown in Table 7 and Figs. 32 and 33, in the same format as the other optimization results. Though the design variables of the optimal found with the multi-fidelity meta-model are not identical to the optimal found with ARC2D, visual inspection of the airfoil shows the two solutions are quite close. Additionally, the pre-shock Mach number is outside the range of validity of TRAN2D, leading to different Mach contours and objective functions from the TRAN2D and ARC2D analyses. Thus the multi-fidelity model has located an optimal near the high-fidelity optimal, in a region where the low-fidelity tool is unable to find the proper extrema.

In summary, the multi-fidelity meta-model was created with 1800 low-fidelity points and 143 high-fidelity

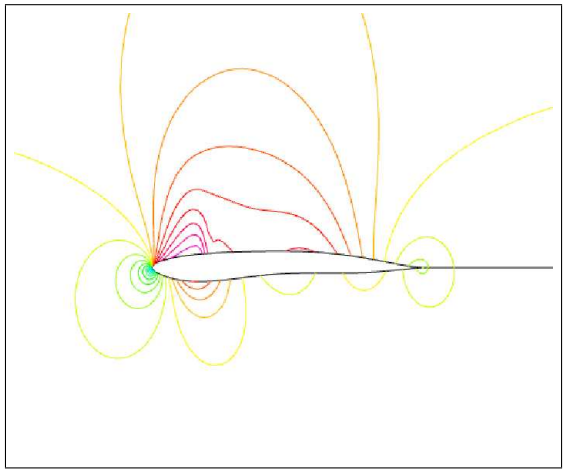


Figure 28. Low-Fidelity TMM Optimal Analyzed with TRAN2D

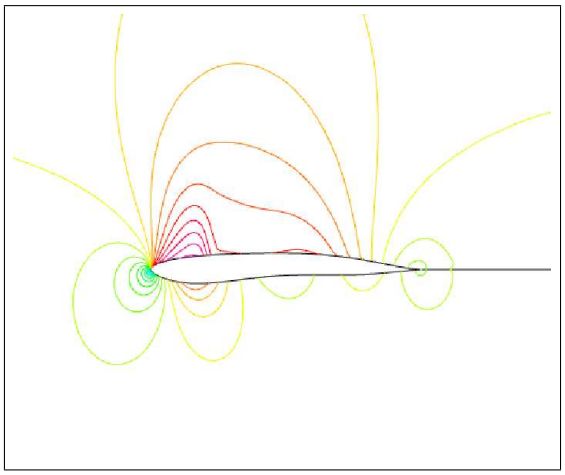


Figure 29. Low-Fidelity TMM Optimal Analyzed with ARC2D

X_{up}	Z_{XXup}	X_{lo}	Z_{XXlo}
0.468	-0.298	0.699	0.271

	Obj	M_C	C_L	C_D
TRAN2D	0.02779	1.14	0.53044	0.002004
ARC2D	0.02657	1.16	0.55365	0.000614

Table 5. Low-Fidelity TMM Optimization Results

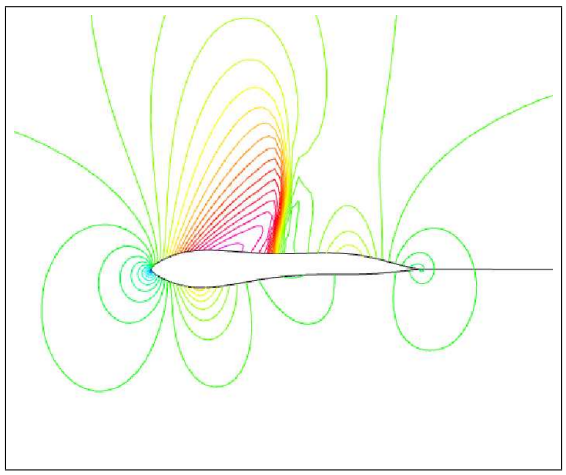


Figure 30. ARC2D Direct Search Optimal Analyzed with TRAN2D

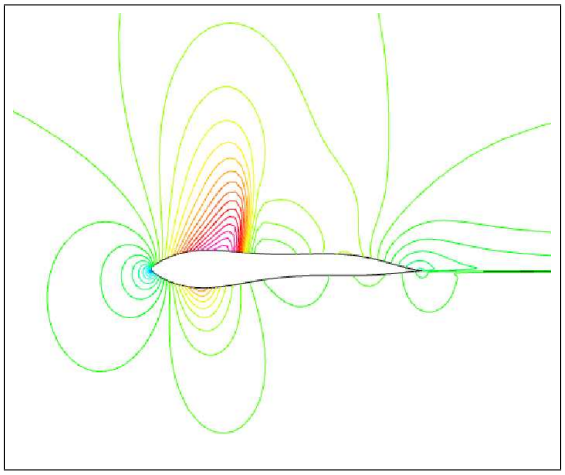


Figure 31. ARC2D Direct Search Optimal Analyzed with ARC2D

X_{up}	Z_{XXup}	X_{lo}	Z_{XXlo}
0.644	-0.750	0.700	0.159

	Obj	M_C	C_L	C_D
TRAN2D	0.14424	1.58	0.75438	0.04172
ARC2D	0.005834	1.54	0.547282	0.030711

Table 6. ARC2D Direct Search Optimization Results

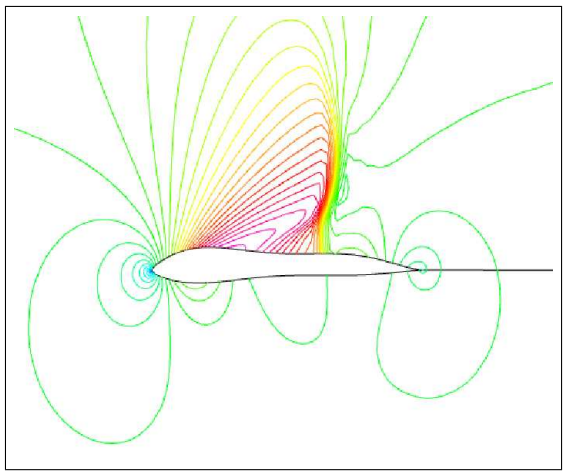


Figure 32. Multi-Fidelity TMM Optimal Analyzed with TRAN2D

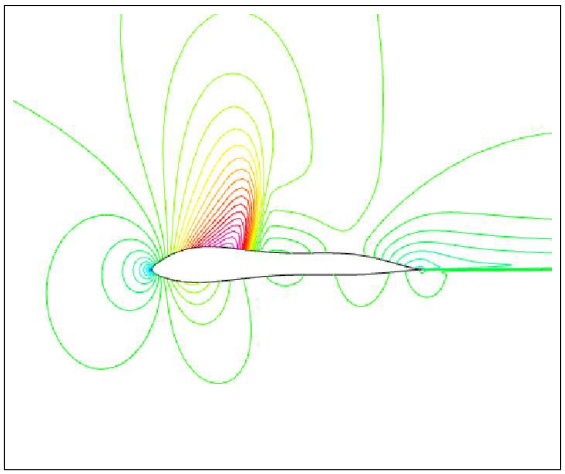


Figure 33. Multi-FIdelity TMM Optimal Analyzed with ARC2D

X_{up}	Z_{XXup}	X_{lo}	Z_{XXlo}
0.638	-0.526	0.697	0.284

	Obj	M_C	C_L	C_D
TRAN2D	0.83252	1.72	1.04905	0.068367
ARC2D	0.006898	1.64	0.553058	0.0481978

Table 7. Multi-Fidelity TMM Optimization Results

points. TRAN2D is roughly an order of magnitude faster to evaluate than ARC2D, thus the equivalent number of high-fidelity evaluations required for the multi-fidelity meta-model is on the order of 300. With the ARC2D direct search taking 3000 ARC2D evaluations, the multi-fidelity meta-model is 10 times less expensive than the direct search, and was able to locate a good approximation to the high-fidelity optimal where the low-fidelity data could not.

VI. Conclusions and Future Work

The multi-fidelity TMM framework makes it possible to extend global meta-modeling to higher dimensional design spaces and complex design problems. In this paper, a highly parallelizable domain partitioning algorithm and progressive sampling technique was presented that creates a global model with comparable predictive capabilities to a single Kriging surface and an order of magnitude speed-up in evaluation time. The implementation of LARS was shown to find a high-fidelity data subset creates a multi-fidelty global model that compares well with high-fidelity test data. In addition, the optimization of a 4D design space illustrated the capability of the multi-fidelity meta-model created with the TMM framework to locate an optimal near the global high-fidelity optimum that was unobtainable with the low-fidelity data alone.

Future work includes taking a more sophisticated approach to choosing high-fidelity points to add to the multi-fidelity surface in an attempt to improve the comparison of optimization solutions to the true optimal. This will include implementing adaptive sampling that will build on the LARS multi-fidelity database by choosing points that are likely to add new extrema, or information near existing extrema, to the meta-model. More difficult multi-fidelity problems will also be investigated, including the optimization of an airfoil using 10-15 design variables, and the optimization of a 3D wing using 5-10 design variables.

References

[1]Alexandrov, N., Dennis, Jr., J. E., Lewis, R. M., and Torczon, V., "A Trust Region Framework for Managing the Use of Approximation Models in Optimization," *Structural and Multidisciplinary Optimization*, Vol. 15, No. 1, 1998, pp. 16–23.

[2]Booker, A. J., Dennis, Jr., J. E., Frank, P. D., Serafini, D. B., Torczon, V., and Trosset, M. W., "A Rigorous Framework for Optimization of Expensive Functions by Surrogates," Technical Report CRPC–TR98739–S, Center for Research on Parallel Computation, Houston, TX, Feb. 1998.

[3]Gramacy, R. B. and Lee, H. K. H., "Adaptive design of supercomputer experiments," Tech. rep., Dept. of Applied Math & Statistics, University of California, Santa Cruz, 2006.

[4]Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R., "Least Angle Regression," *Annals of Statistics (with discussion)*, Vol. 32(2), 2004, pp. 407–499.

[5]Whitley, D. and Kauth, J., "GENITOR: A Different Genetic Algorithm," *Proceeding of the Rocky Mountain Converence on Artificial Intelligence*, Denver, CO, 1988.

[6]Matheron, G., "Principles of geostatistics," *Economic Geology*, Vol. 58, 1963, pp. 1246–1266.

[7]Cressie, N., "The Origins of Kriging," *Mathematical Geology*, Vol. 22, 1990, pp. 239–252.

[8]Cressie, N., *Statistics for Spatial Data*, John Wiley, New York, 1993.

[9]Lophaven, S. N., Nielsen, H. B., and Sondergaard, J., "Dace a Matlab Kriging Toolbox," Technical Report IMM–TR–2002–12, Informatics and Mathematical Modelling, DTU, 2002.

[10]Lophaven, S. N., Nielsen, H. B., and Sondergaard, J., "Aspects of the Matlab Toolbox Dace," Technical Report IMM–REP–2002–13, Informatics and Mathematical Modelling, DTU, 2002.

[11]Jones, D. R., Schonlau, M., and Welch, W. J., "Efficient Global Optimization of Expensive Black-Box Functions," *Journal of Global Optimization*, Vol. 13, 1998, pp. 455–492.

[12]Romero, V. J., Krishnamurthy, T., and Swiler, L. P., "Progressive Response Surfaces," Tech. rep., 9th ASCE Specialty Conference on Probabilistic Mechanics and Structural Reliability, Albuquerque, NM, July 2004.

[13]Romero, V. J., Slepoy, R., Swiler, L. P., Guinta, A. A., and Krishnamurthy, T., "Error Estimation Approaches for Progressive Response Surfaces," AIAA Paper 2005–1822, 46th Structures, Structural Dynamics, and Materials Conference, Austin, TX, April 2005.

[14]Owen, A., "Quasi-Monte Carlo Sampling," A tutorial for siggraph 2003, Stanford University Statistics Dept., 2003.

[15]Sobol', I. M., "The Distribution of Points in a Cube and the Accurate Evaluation of Integrals (in Russian)," *Zh. Vychisl. Mat. i Mat. Phys.*, Vol. 7, 1967, pp. 784–802.

[16]Faure, H., "Discrépance de suites associées à un système de numération (en dimension s)," *Acta Arithmetica*, Vol. 41, 1982, pp. 337–351.

[17]Niederreiter, H., "Point Sets and Sequences with Small Discrepancy," *Monatshefte fur mathematik*, Vol. 104, 1987, pp. 273–337.

[18]Holst, T., "Implicit Algorithm for the Conservative Transonic Full-Potential Equation Using An Arbitrary Mesh," *AIAA Journal*, Vol. 17, No. 10, 1979, pp. 1038–1045.

[19]A.Jameson, "Transonic Potential Flow Calculstion Using Conservative Form," *Second AIAA CFD Conference*, 1977, pp. 35–54.

[20]Pulliam, T. H., "Efficient Solution Methods for The Navier-Stokes Equations," *Lecture Notes for the von Karman Institue For Fluid Dynamics Lecture Series: Numerical Techniques for Viscous Flow Computation in Turbomachinery Bladings*, von Karman Institue, Rhode-St-Genese, Belgium, 1985.

[21]Sobieczky and Helmut, "Parametric Airfoils and Wings," *Notes on Numerical Fluid Mechanics*, Vol. 68, 1998, pp. 71–88.