38th Fluid Dynamics Conference and Exhibit<BR>
23 - 26 June 2008, Seattle, Washington

AIAA 2008-4162

Gaussian Process Metamodeling applied to a Circulation Control Wing

Scott Turner[1], Tyler Ball[1], and David D. Marshall[2]
California Polytechnic State University, San Luis Obispo, California, 93401, USA

Metamodeling fluid systems is an efficient way to do complex analysis on multivariate problems and can allow for time savings in an optimization setting. Gaussian process metamodels provide a flexibility in modeling that can be extended to both experimental and deterministic experiments. This paper specifically addresses such models applied to computational fluid dynamics analysis of a 3D circulation control wing. The framework for a generalized approach is first overviewed and then applied to the analysis of the aerodynamics. The Gaussian process regression was essential for both simplifying the calculations required for balanced field length computation and for analysis of complex multi-variable behavior.

Nomenclature

BFL = balanced field length
C_D = drag coefficient
C_L = lift coefficient
C_M = moment coefficient
C_μ = blowing coefficient
CCW = circulation control wing
CFD = computational fluid dynamics
DOE = design of experiments
E = expected value
f = Gaussian process function space
$\bar{f}$ = Gaussian process function space mean
GP = Gaussian Process
k, cov = covariance function
L = Cholesky factor
= length scale hyperparameter
M = Mach number
m = mean function
$\dot{m}$ = mass flow rate
NS = Navier-Stokes
P(A) = probability of event A
P(A|B) = probability of A given B
$P(A \cap B)$ = probability of the intersection of A and B
RBF = radial basis function
RS = response surface
Sref = wing reference area
STOL = short takeoff and landing
σ = variance
~ = distributed according to

I. Introduction

Historically there are a number of ways to approach the analysis of data for fluid dynamics problem, which are all dependent on the type of data that has been collected. Experimental data has error that is intrinsically different than the error for a CFD calculation, which is itself different than the inaccuracies of theoretical models. Many approaches exist to model these different types of analyses, including response surfaces, splines, and Gaussian process regression models to name a few. Response surface methods particularly favor experimental data as they are built around a statistical framework that assumes each point has a built in error noise and also that the error noise is normally distributed[4]. Experimental data typically comes with both of these conditions as data collection is never exact and each replicate of an experiment will yield different responses for the same set of input values. However, computer experiments, which are increasingly important in fluid analysis, have only systematic error with no noise[16]. This systematic error results from either an inappropriate model or numerical errors. This lack of noise means that a modeled curve of computer data should fall through each gathered data point[22]. Thus models that fit a curve through each point, like a spline fit, would be more appropriate; this approach is a distinctly different model than a response surface. This disconnect between experimental and computational data analysis can be bridged by the use of Gaussian

---

[1] Graduate Student, Aerospace Engineering, student member AIAA
[2] Assistant Professor, Aerospace Engineering, senior member AIAA

process analysis. Gaussian process models have the ability to act as both splines or response surfaces based on the hyper-parameters and covariance functions specified when defining the process. This property makes them flexible and a good fit for fluid analysis where there are highly non-linear flow regimes and many types of data to be analyzed.

This paper seeks to extol the benefits of Gaussian process models in representing different fluid systems, with the ability to capture physical flow phenomena with versatility not available to other model types. First a comparison of different models types will be conducted, followed by an implementation of the Gaussian process architecture. This Gaussian process is then compared in different forms and finally applied to the complex 3D flow of a circulation control wing.

II. Metamodeling Basics

At the most basic level, metamodeling is the synthesis of models that approximate the output of other models. A polynomial fit to lift data from a panel method, a linear fit through different cost model data, a spline fit through FEA data points, are all metamodels. Each of these examples uses some sort of mathematical curve fit to predict results of some other model for a range where that model has not been tested. By its very nature, metamodeling is a multidisciplinary field that is based on statistics, linear algebra, and the specific field being modeled. The goal of the metamodel is to approximate a model the same way a model approximates reality. At first glance this appears an unnecessary complication because it adds in more opportunity for error and an additional process on top of the modeling process. However, the reality of most high fidelity models is that they do not provide continuous or quick solutions to physical problems, which is not acceptable in most engineering analysis applications.

The language of metamodeling should be reviewed before a discussion of the metamodeling process can be undertaken. Metamodels are built off of points taken from an underlying model. These points are called training points as they are what the behavior of the metamodel is based on. Training points are the ones obtained through expensive testing techniques and the number should be minimized. The goal of a metamodel is to get functional values between these training points. The points that have not been gathered are termed test points. They are the points where a test input is given and some output is taken from the metamodel. Training and test points are broken into two components each, inputs (or factors) and outputs (targets or responses). So for each training point there is an input set that corresponds to an output gathered from the base model. The test point input is the point in the input space, or hyperspace, that defines the point where a regression value is desired. The output for the test point is the value in question or the answer. Gaussian process regression metamodels, which will be shortened to just Gaussian processes (GP), use training data in two ways. The first way is that they use the training points directly for regression since they are local metamodels. Like an interpolation, they use training points near the desired test point to determine the test output at that location. The second way is that GPs use training points to determine the relationship between test points and training points.

As has been noted, metamodels come in a variety of different flavors. There are linear metamodels with coefficients determined through regression schemes. There are spline metamodels that are defined in terms of their neighbor points. Any interpolation scheme is a type of metamodel. Neural nets and genetic algorithms can also be metamodels. However, all metamodels fall into two generalized categories, local or global.

Global metamodels are based on all of the training data in a data set. A linear regression for instance is based on linear algebra that utilizes every point in the matrix inversion to calculate the coefficients. Because they are based on every data point in the set they are sensitive to outliers. An outlier in a global metamodel will affect the predictions across the entire space, which could be undesirable. Another feature of global metamodels is that they typically can not adjust well to sharp gradients in data. They will either tend to smooth out these abrupt changes over the surrounding area or they will be highly oscillatory in smooth regions away from the sharp gradient. This is obviously not acceptable for a good model. In general global metamodels do well at modeling parameters that are relatively smooth. Their chief advantage is that they provide an explicit functional definition once the coefficients are defined. Explicit functions allow for quick executions and implementation in code form. For this paper response surfaces (RS) were used as the explicit function of choice when appropriate. Response surface methods have a collection of statistical methods associated with them that allow for prediction errors. For the statistical analysis typically the data has to be orthogonal and have normal residuals[12].

Local metamodels are generally functions of the immediate points in their neighborhood. A linear interpolation is a classic example of a local metamodel. The function value between two points is only dependent on the value of those two points and not any of the data in the set. Typically this is overly simplistic and will lead to large errors if there are only a few input points in a large data space. Certain local metamodels, such as Gaussian processes (GPs) and radial basis functions (RBFs), have a term that is a length scale[26]. This term determines how close training points must be to affect a test point. Short length scales mean that only the closest points will affect the behavior of the regression so it will behave more like an interpolation. Long length scales include a large number of the training data points and behave more like a global metamodel. A final way to look at this concept is that the length scale is about how long it takes for a regression to make a significant change in output value. Linear polynomial regression can actually be shown to be subset of a GP[21].

The metamodels investigated in this paper were the RS and GP metamodel. Response surfaces are often used for statistical analysis of non-deterministic experiments, but as was mentioned the statistical benefits can not be realized with deterministic experiments[28]. The goal of using the RS/GP combination was to see how existing tool sets that many are familiar with compares to a more applicable technology. GP metamodels have the ability to have varying length scales and thus should always get estimates that are at least as good as their RS counterparts. A certain degree of familiarity is assumed with a RS type metamodel. Many have seen this type of metamodel in the form of simple least squares regressions. RSs are based around least squares regressions and can be done in n dimensions.

III. Gaussian Process Overview

Gaussian process regression is founded on the idea of Bayesian inference and probability theory. A brief overview covers some of the important background ideas and references good sources for getting much more detailed information.

1. Bayesian Inference

Typical probability analysis looks at the likelihood of something happening, but does not address the past behavior to determine an updated probability. In other words a classic probability gives the likelihood of an event independent of any knowledge about past behavior. Bayesian inference is the other side of the coin. It takes into account the past behavior to make a judgment on the probability given that information. Gaussian process metamodels are built around the idea of using training data to make estimates of the probability that a function is close to the data points.

The definition of the term P(A|B) is given as the probability of both events A and B divided by the probability of B. Thus Bayes theorem can be derived from the definition of P(A|B) and P(B|A) as shown below in equation 1.

$$\begin{aligned} P(A|B) &= \frac{P(A \cap B)}{P(B)} \\ P(B|A) &= \frac{P(A \cap B)}{P(A)} \\ P(A \cap B) &= P(A|B)P(B) = P(B|A)P(A) \\ P(A|B) &= \frac{P(B|A)P(A)}{P(B)} \end{aligned} \tag{1}$$

Equation 1 is Bayes theorem which relates the probability of A, B and B given A to find the probability of A given B. P(A) is known as the prior because it is the prior probability of A without use of any information from B. P(B|A)/P(B) is known as the normalized likelihood function since the probability of B given A gets divided by P(B) normalizing it on that probability. This essentially results in a percentage change in P(A) based on the normalized likelihood function. The components of the normalized likelihood function are the likelihood and the marginal likelihood (or evidence).

An example from DeLoach[5] helped to clarify this idea of training data being used to adjust a probability. For example, if there is a coin which may be fair, a 1:1 weighting, or may be unfair, a 2:1 weighting of heads to tails, then we should be able to perform a number of tests and make a determination of fairness based on that data. From equation 1, the prior statement A is that the coin is fair and the prior P(A) is 0.5 since it is equally likely that the coin is fair as it is unfair. If 100 flips are then performed and the results recorded, say 43 heads and 57 tails, the likelihood and marginal likelihood can then be calculated based off binomial probability formula, equation 2. This yields P(B|A)=0.0301

$$\begin{aligned} P(B|A) &= \frac{N!}{x!(N-x)!} p^{x} (1-p)^{N-x} \\ &= P(43\ Heads\ \ fair) \end{aligned} \tag{2}$$

The marginal likelihood P(B) has to be modified to take into account the probability of the coin being either fair or unfair. B is the condition of flipping 43 heads, and since that value is not possible to calculate individually, it has to be broken into separate equations. This is based on the identity that $P(B) = P(A \cap B) + P(\bar{A} \cap B)$, where $\bar{A}$ is the case where the coin is unfair. Using the conditional probability definition from equation 2 transforms P(B) to $P(B)=P(B|A)P(A)+P(B|\bar{A})P(\bar{A})$, which can then be solved since $P(B|A)=0.0301$, $P(A)=0.5$, $P(B|\bar{A})=0.0107$, and $P(\bar{A})=0.5$. This solves the modified Bayes equation shown in equation 3.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{P(B|A)P(A)}{P(B|A)P(A)+P(B|\bar{A})P(\bar{A})} \tag{3}$$

The result is that P(A|B) = 0.738 or there is a 73.8 % chance that the coin is fair based on the observation of the 43 heads and knowing the probabilities of both the fair and unfair coins. This is significant because a probability was taken that reflected minimum knowledge about a situation but given data an updated estimation could be made.

This is the idea behind the Gaussian process regression. A Gaussian process, in the mathematical sense, is a set of random parameters that define functions. Thus a Gaussian process defines a set of random functions. A random function is a function whose defining parameters are determined through some sort of random distribution. So for the case of the Gaussian process the random function parameters are distributed according to a Gaussian distribution. Since the random functions are defined according to a statistical distribution Bayesian inference can be used to train the distribution of functions on training data much the way the probability of the coin was trained on the training data from flipping coins.

2. Bayesian Inference in Gaussian Process Regression

The ability to adapt the probability based on past information is the key component of a Gaussian Process regression. The GP regression uses this principle twice to make a prediction about the behavior of the posterior. First it uses Bayesian inference to find the parameters used to define the random function and then it uses Bayesian inference again to choose the random function that best fits the data. The first step requires that the marginal likelihood be calculated and the most likely parameter values found, these parameters are called hyperparameters. Often this involves an optimization step. The marginal likelihood is simply the integral of the likelihood and the prior and will be discussed in detail later. The second step calculates the mean and variance for a set of test points.

A way to arrive at the equation for the GP regression is to start from the function space perspective and start with some definitions. The following derivation was fully developed in Rasmussen.

By strict definition, a GP is a collection of random variables, any finite number of which have a joint Gaussian distribution. A GP can be defined entirely in terms of its mean and covariance function.

$$\begin{aligned} m(\boldsymbol{x}) &= E[f(\boldsymbol{x})] \\ k(\boldsymbol{x},\boldsymbol{x'}) &= E[(f(\boldsymbol{x})-m(\boldsymbol{x}))(f(\boldsymbol{x'})-m(\boldsymbol{x'}))] \end{aligned} \tag{4}$$

After which these two equations can be rolled into one general GP equation

$$f(\boldsymbol{x}) \sim GP(m(\boldsymbol{x}),k(\boldsymbol{x},\boldsymbol{x'})) \tag{5}$$

The m(x) term is the mean function for the GP. This is typically set to 0, though non-zero mean functions are also possible with added complexity[26]. The k(x,x') term is the covariance function term, which itself is a function of both test points, x, and training points, x'. f(x) is the real process, which is our desired metamodel. This equation shows mathematically the definition earlier, there is a mean function m(x) and a covariance function k(x,x') that completely define the distribution f(x).

Since zero mean GPs are the norm, the covariance function is what affects the behavior of the metamodel. A covariance function specifies the covariance between pairs of random variables[26]. Since a GP is built up off a collection of random variables the covariance specifies how the GP changes across the range of a design space. A common covariance function and the one used for this research is the squared exponential (SE) covariance function.

$$cov(f(\boldsymbol{x}_p),f(\boldsymbol{x}_q)) = k(\boldsymbol{x}_p,\boldsymbol{x}_q) = \exp\left(-\frac{1}{2}(\boldsymbol{x}_p-\boldsymbol{x}_q)^2\right) \tag{6}$$

It can be shown[26], that the SE covariance function is a Bayesian linear regression model with an infinite number of basis functions. From a slightly different angle, the SE covariance function can be defined from an infinite number of Gaussian shaped basis functions.

Some of the benefits of the SE covariance function are that it is infinitely differentiable, infinitely mean-square differentiable, and that it has a length scale that can be controlled by a parameter value with an appropriate modification to equation 4 shown below as equation 5.

$$k_{SE}(\boldsymbol{x}_p, \boldsymbol{x}_q) = \exp\left(-\frac{|\boldsymbol{x}_p - \boldsymbol{x}_q|^2}{2 l^2}\right) \tag{7}$$

It is assumed that of all the functions defined by the GP there is a function that not only fits the best but is simple. This function is found through using Bayesian inference to provide information about the set of random functions at certain points. Given that the random functions go through or near these points a new distribution can be determined. The mean of this new function distribution is defined as the best function, f_*, and is the regression value for the test points. This most likely function is the one that should be used to model the data based on likelihood information from the training data points. This means you want to find some function, f_*, by taking the Gaussian process and conditioning it based on the training points, y.

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}_* \end{bmatrix} \sim N\left(\boldsymbol{0}, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right) \tag{8}$$

In different terms, the distribution on the right in equation 6 that defines the underlying function distribution is conditioned based on the training outputs y. The case shown has the covariance function broken into sub-matrices based on X_* and X. These define the input points for the test points and training points respectively. A training point is a data point that was gathered from an external source and used to train the GP. A test point is a point where a mean value and covariance will be calculated, mean f_* and cov(f_*). There is a covariance relationship between all combinations of these points. K(X,X) is the covariance of training points, K(X_*,X_*) the covariance of test points, and K(X_*,X) the cross term covariance.

For actual data there is often noise associated with the measurements, this means that there should be noise associated with the training points. Assuming that the noise is Gaussian with a zero mean and a covariance σ_n, the noise term can be added to the diagonal of the training covariance matrix yielding equation 7.

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}_* \end{bmatrix} \sim N\left(\boldsymbol{0}, \begin{bmatrix} K(X,X)+\sigma_n^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right) \tag{9}$$

Finally, conditioning the distribution based on the training output values yields the most likely GP. This GP, like all GPs, is defined by its mean function and covariance function, which in this case are the mean and covariance for the test points on that most likely function. In other words like the coin example the probability has been adjusted based on the data and the function which is now defined by f_* is defined in terms of a mean and covariance. Equation 8 shows how to calculate the mean of f_* and cov(f_*).

$$\begin{aligned} \boldsymbol{f}_*|X,\boldsymbol{y},X_* &\sim N\left(\bar{\boldsymbol{f}}_*, cov(\boldsymbol{f}_*)\right), \text{ where} \\ \bar{\boldsymbol{f}}_* &= E\left[\boldsymbol{f}_*|X,\boldsymbol{y},X_*\right] = K(X_*,X)\left[K(X,X)+\sigma_n^2 I\right]^{-1}\boldsymbol{y} \\ cov(\boldsymbol{f}_*) &= K(X_*,X_*) - K(X_*,X)\left[K(X,X)+\sigma_n^2 I\right]^{-1} K(X,X_*) \end{aligned} \tag{10}$$

To reiterate the last point, by definition of conditioning the function the new distribution f_* is the most likely function and therefore it is defined by its mean and covariance function. This is the mean function for all of the test points and the variance at each of those points. However, we only know the mean and covariance, we do not ever actually know the underlying function explicitly. This is not a problem because a series of test points can be calculated and the mean plotted. This plot will show the behavior even if there is no explicit function.

This form can be simplified by assuming that there is only one test point and by defining some of the covariance sub-matrices differently. k_* is the test and training point covariance, $k(x_*,x_*)$ is the test point covariance, and K is the training point covariance. The results are shown in equation 11.

$$\begin{aligned} \bar{f}_* &= \boldsymbol{k}_*^T\left[K+\sigma_n^2 I\right]^{-1}\boldsymbol{y} \\ V(f_*) &= k(\boldsymbol{x}_*,\boldsymbol{x}_*)-\boldsymbol{k}_*^T\left[K+\sigma_n^2 I\right]^{-1}\boldsymbol{k}_* \end{aligned} \qquad (11)$$

To review, a prior guess was made as to what type of function was being modeled. That guess was the prior GP, which was zero mean and based on a covariance function. The covariance function determined what functions could be defined by the Gaussian Process. Basically the covariance function states which functions can be randomly determined from that particular GP. With the prior defined, like the coin example, the training data was used to condition the prior. This meant that the probability of the different functions were taken into account given the observed data. This resulted in a new probability distribution, a new GP. This final GP was the posterior and was defined by a mean of f_* and predictive variance $V(f_*)$. This mean and the variance define the regression function at the test inputs even though f_* is not explicitly known. All of the probabilities from the prior and the training data are used to make a prediction about the responses of the test points, which is the desired values for regression.

GP Regression Algorithm

The algorithm outlined in Rasmussen and Williams[26] goes over the steps just outlined with some additional computational components. The inputs to the algorithm are a set of training inputs X, a set of training outputs or targets, y, a covariance function, k, and a test input, x_*. The algorithm then uses equation 12 to calculate the mean, covariance, and log likelihood. The algorithm is as follows.

$$\begin{array}{lcclr} 1: & L & = & cholesky(K+\sigma_n^2 I) & \\ 2: & \boldsymbol{\alpha} & = & L^T \setminus (L \setminus \boldsymbol{y}) & \\ 3: & \bar{f}_* & = & \boldsymbol{k}_*^T\boldsymbol{\alpha} & Predictive\,mean \\ 4: & \boldsymbol{v} & = & L \setminus \boldsymbol{k}_* & \\ 5: & \boldsymbol{V} & = & k(\boldsymbol{x}_*,\boldsymbol{x}_*)-\boldsymbol{v}^T\boldsymbol{v} & Predictive\,variance \\ 6: & \log p(\boldsymbol{y}|X) & = & -\frac{1}{2}\boldsymbol{y}^T\boldsymbol{\alpha}-\sum_i \log L_{ii}-\frac{n}{2}\log 2\pi & Marginal\log likelihood \end{array} \qquad (12)$$

The Cholesky function in the algorithm is a shorthand for a Cholesky factorization, which can be used to solve for matrix inverse problems. It is computationally more stable and efficient than other methods for matrix inversion problems, but the matrix must be positive definite. The rest of the components of the algorithm are just a result of the Cholesky factorization. Another benefit of the Cholesky factorization is that it can be used in the log likelihood instead of the actual inverse, thus speeding up the process. The marginal log likelihood is useful for optimizing the parameters that define the covariance function and for additional discussion see Rasmussen and Williams.

3. Complex Function

A version of Rasmussen's code GP regression was used for a stress test. The test was a set of functions combined into one complex function space that would provide a significant challenge to a regressing algorithm. The goal of this function was to see if a GP could handle the complex behavior found in aerodynamic analysis. This could include steps from shear layers or shocks along with the oscillatory behavior found in unsteady flow. A 3D test of each of these functions simultaneously was a way to see if a GP regression could handle such varying functional requirements. Each quadrant of the input space was weighted so that one function would be dominant in that quadrant. Figure 1 shows the function used. Starting at the bottom left quadrant and proceeding clockwise, there was a constant region followed by a sinusoidal region. The top right was an exponential and the top left was the step region. A varying number of points were taken from this distribution and used as training points for the GP. The output of these points was then compared to the original function.

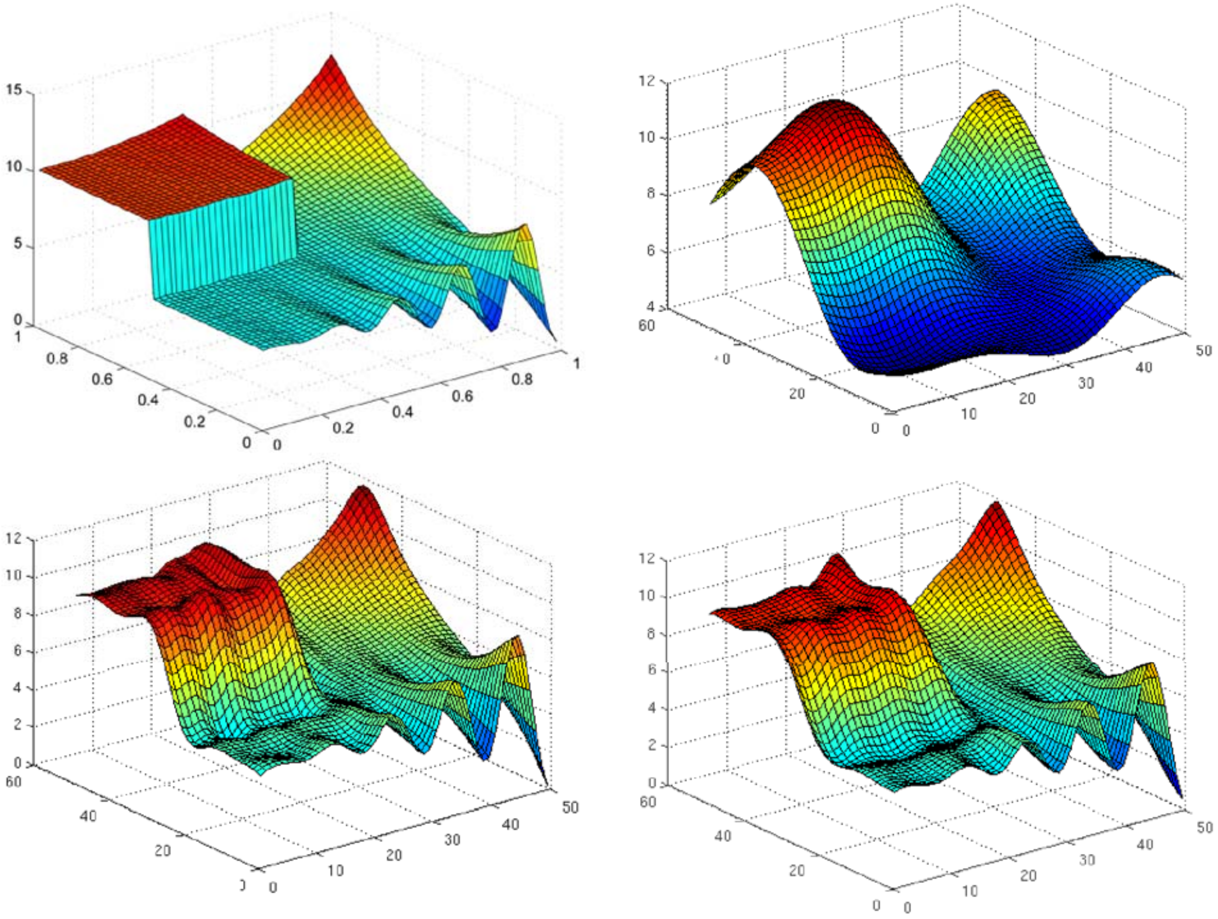


Figure 1: GP regression of complex composite function. Starting clockwise top left: actual function, 100 training point GP, 500 training point GP, 750 training point GP

In order to get a decent approximation of the underlying function a number of training points had to be used. The additional points essentially help to resolve the step portion of the function. While they helped with the overall accuracy of the model everywhere the step function portion always had the largest errors.

The most telling comparison of the performance of the GP is to that of a response surface model for the same number of data points. A response surface was built up for the same 750 point case that the GP metamodel was constructed for and the difference is quite telling about the ability to capture the locally important behavior, Figure 2.

As can be seen from the picture, the response surface based on a polynomial regression simply can not deal with the complex behavior of the function. While locally it may be able to deal quite well with any component of the complex function, when examined as a whole model, the RS fails to capture any of the significant behavior.

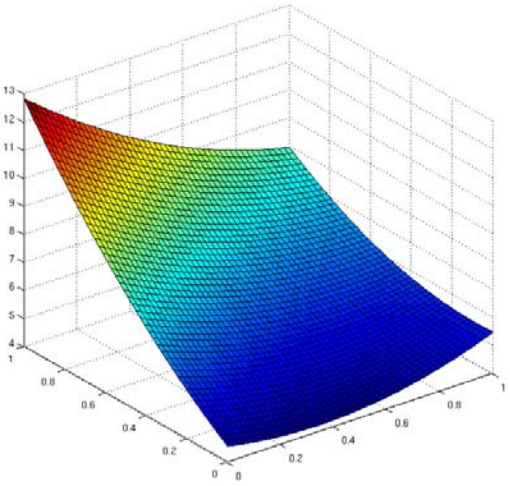


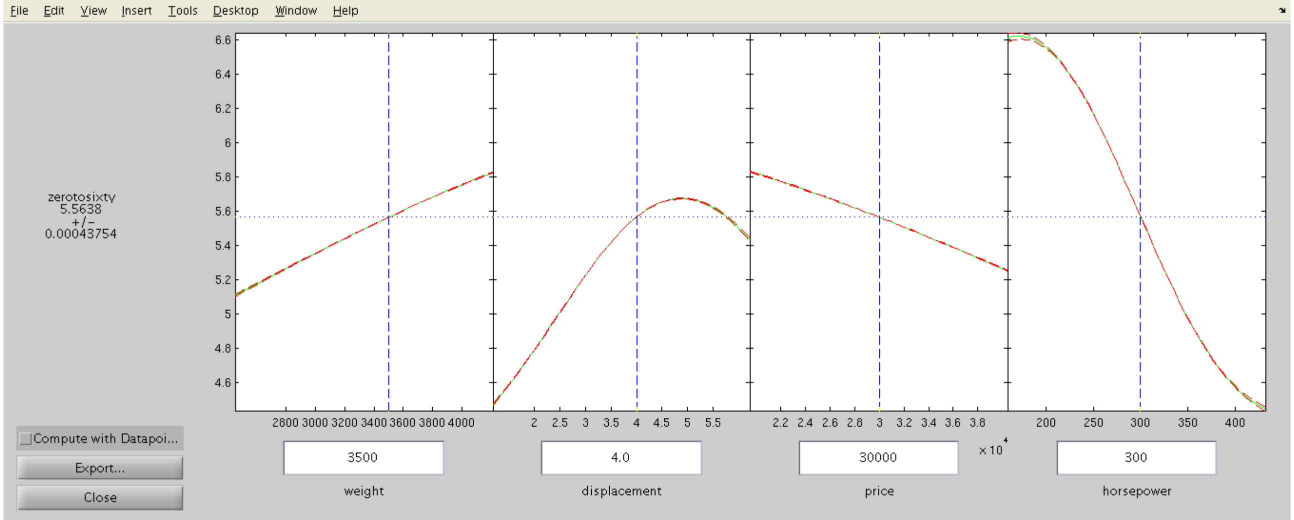
Figure 2: Response surface metamodel of complex function based on 750 points

IV. Tool Development

Three primary tools were developed in order to visualize the model and to quantify the fit of the model. The first tool developed was termed a multiplot tool because it allowed interactive visualization of n plots that represented the n inputs possible for the regression, Figure 3. Each input appears as a separate plot on the chart, but the plots are coupled together so dragging the line on one chart changes the position of the other plots on the chart. The tool is based around the Matlab response surface tool GUI design. The big advantage of the multiplot is that any point in the design space can be examined and trend behavior can be seen. This means that minimums and maximums can be picked out of an n-dimensional space with ease, while the sensitivities of variables to other variables can also be found.

The next tool developed was an extension of the multiplot idea that allowed for the visualization of the GP in three dimensions while holding the other dimensions constant. That GUI is referred to later in the paper, in Figure 13. Reproduction here of that plot was of limited utility since it was typically only useful for visualizing points of uncertainty and coupling between variables. Both which could be seen in the multiplot with the correct manipulations.

The final tool developed was an error estimation code that established the error of the GP in a number of different ways. Since the variance provided as an output of the GP is not really a data variance in terms of a classic ANOVA, a separate error analysis must be completed that looks at the model in a different light. The typical ways of doing error analysis on a GP are to add or remove a point and then analyze the models behavior with the point removed. Specifically the residuals are the metric typically compared between different models.



zerotosixty
5.5638
+/-
0.00043754

weight: 3500

displacement: 4.0

price: 30000

horsepower: 300

Figure 3: Multiplot interface

For this work, three new methods were employed to gauge the amount of error present in the GP and these were compared to two methods from the literature. They are compared on a metamodel of 2D CFD data for a car wing. The model was relatively cheap for a CFD model and provided similar types of flow phenomena as would be seen with CCW. The first error method, worst case error, used a gradient based optimizer to identify the worst error location, the top plot of Figure 4. The input parameters were used as the optimizer's inputs. This allowed for use of the fmincon function in Matlab. To insure that a global minimum was achieved, the optimization was started from 30 random locations across the input space and the minimum negative error of these was chosen so as to find the global maximum error.

The second method, sequential training error, was based around adding a point to the model and then repeating the analysis to get a slightly different model, Figure 4. The training point data was built up sequentially in the .gpm file so instead of fitting all of the data at once, points were taken and added to the model one at a time with the residuals calculated at the point of a residual from one model to the next. For example, if there were 20 points in the model, a test point was run at where the 21st point would be placed and compared to the actual value of the 21st point. This meant as the model got more and more complete adding more training data would do little to change the model and the residuals of these training points should begin stabilizing and dropping towards zero.

The third method, random test point error, was similar to the second method in that sequential errors were calculated at points, Figure 4. However, the difference was a set of random test points were picked before any models were fit. These points were then examined as a new training point was added to the model. For each successive point the residual was calculated based on the change in the model at those random points. A point of note is that for this method there was no mathematically rigorous way to prove this would be beneficial. Through experience of looking at GP regressions it was noticed that under certain situations with deterministic data the functional behavior would switch from oscillatory to smooth and the test point method was an attempt to capture those transitions. For most actual cases it ended up showing very little since the jump from oscillatory to smooth occurred smoothly over a number of models.

The fourth error method, Figure 4, was similar to the second two except that for each point addition it removed all of the training points one at a time and compared the function change at the removed data point value. The process was as follows: remove a point, note the change in function value for the new model, replace the point and remove a new point. This was repeated until residuals at all the points were taken and a RMSE calculated. This means that if there were 30 points in the set. One of the 30 was removed, a residual found, the point put back and the process repeated. This is a standard method for quantifying error in a metamodel with an expensive underlying model and is called the leave one out or LOO method[21]. The advantage is that no additional training points need to be run and in the right situation it can be a reasonable estimation of the actual RMSE for the dataset.

All of these methods are based on some combination of residuals from test point or training point information, which comes from multiple points for each case. To get a single number that could be quickly examined some form of averaging had to be employed. A test case was studied to see if the form of averaging would affect the overall error prediction. The worst case error plot of Figure 4 shows one such case for a 70 training point data set and the worst case error method.

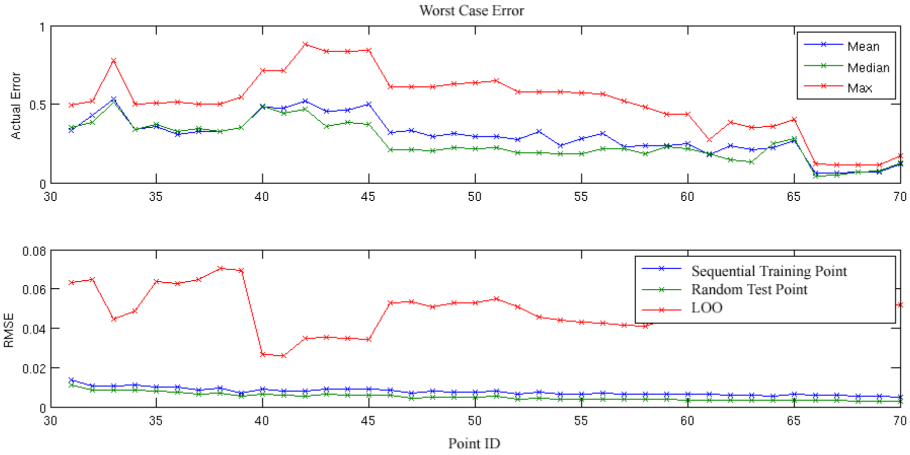


Figure 4: First four error methods for GP regression, car wing case

The plot shows the same general trends regardless of the averaging type between median, mean and max. Even looking at the magnitudes, the max error shows a relatively small increase over the median error for the worst case error method. The bottom plot of Figure 4 shows the RMSE for three of the error methods described above. The LOO method had the highest RMSE but also showed little change after about 55 points. Again these plots are the easy calculations that are cheap to obtain because they use the current set of training points. Plots that change little as points are added to the model indicate model convergence.

Figure 5 shows similar results that are converted to percent errors for the sequential training, random test point, and LOO methods. The first two are contained on the top plot while the LOO method was broken into a separate plot for comparison. The LOO method was found to be sensitive to which averaging method was used since values near zero

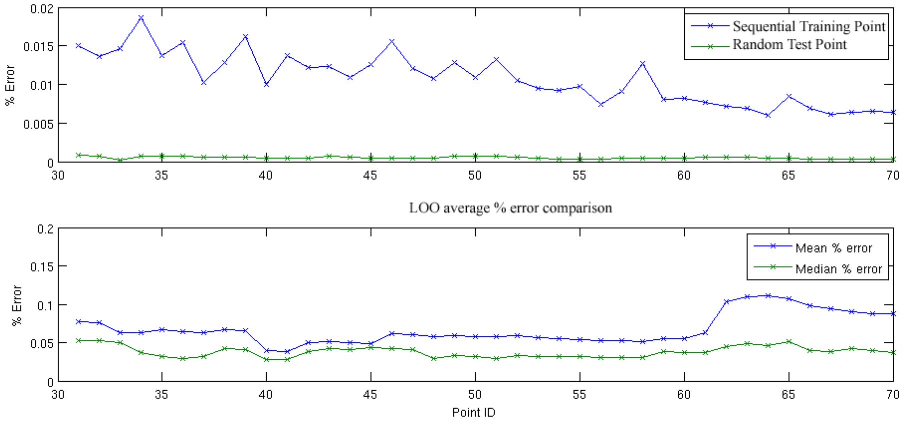


Figure 5: Percent errors for sequential and random point methods compared to LOO

resulted in large percent errors, therefore two averaging methods are plotted. Again, these plots show convergence around 65 points.

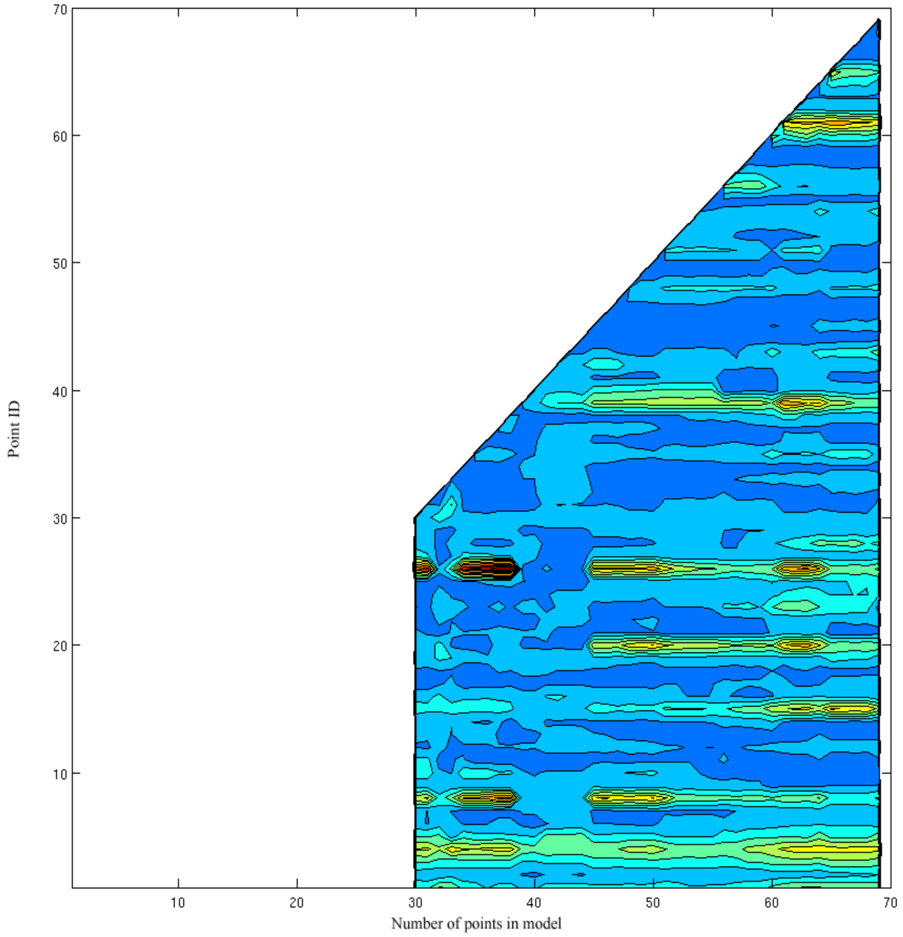


Figure 6: Absolute error for individual points

Figure 6 attempts to capture the idea that individual error points can have a large effect on the error. This figure shows the local error by each point. Training points are added to the model as one travels to the right on the plot. The model begins with 30 points and is complete with 70. The points with red contours show which points have a large percent error when removed from certain sized models. Interestingly, influential points seem to stay influential regardless of when they are added or removed from the model, such as the case with point 62 in the above model. This means that particular point has a very large effect on the size of the error at that location when removed from the model. With enough points all of the contours should be eliminated because no one point has an effect on the overall model.

Originally, this plot was computed in only terms of percent errors. It showed considerably more banding and washed out the details with large percent errors. The cause was certain points close to zero were inflating the percent error and limiting the usefulness of the plot. However, a plot of the absolute errors, Figure 6, had no such limitations at points near zero.

The same type of distinct banding shows up indicating the effect of particular points on the overall error of the model. This banding seems to be caused by outliers that have few nearby points. Though overall the trend is for the error to become of a smaller magnitude as the number of points were increased it was difficult to insure that every point had an equal influence on the metamodel. The only way to insure this would be to use a uniform distribution which would take require a large number of points.

The final error analysis method examined was the true RMSE analysis where additional experimental runs are compared to the results of the metamodel without being used as training data. This is expensive and does not allow for

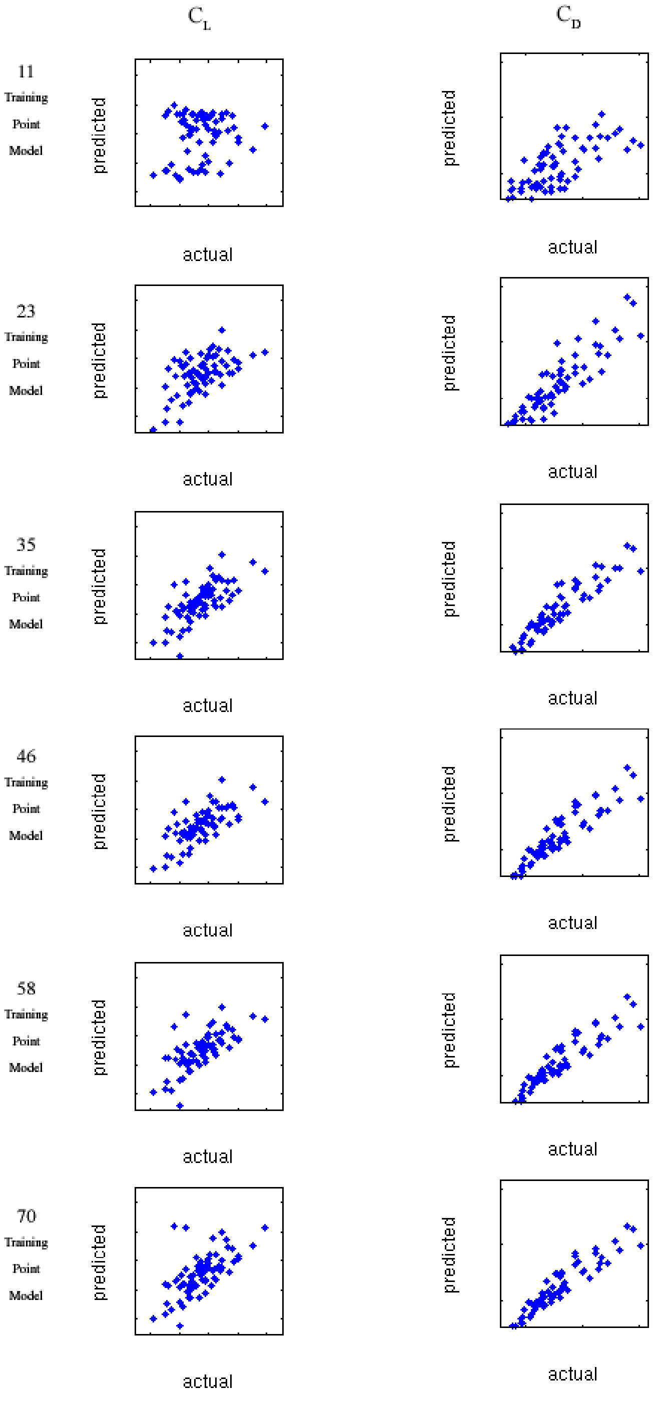


Figure 7: Actual RMSE for different size models based on additional data points

model improvement with the same quantification of error. Fortunately, with GP metamodeling the points can be added to the metamodel once they have been used for the true RMSE validation. For the race car wing example, 70 additional data points were taken from the CFD model and compared to the metamodel at those 70 input locations. Plots, Figure 14, of actual versus predicted values were created so that the behavior could be compared in both C_L and C_D.

These plots show that the model reaches a fairly constant error distribution somewhere between a model built off of 46 and 58 training points. In general, the less costly error methods indicate a convergence somewhere around 60 points. While more cases would need to be run to really validate these trends, the first assessment shows that the error methods created by cheap analyses do not show convergence until slightly later than actual. This helps prove the case that the cheap methods provide a conservative estimate on when the metamodel is converged enough to be useful. This particular data set contains a relatively large amount of variance even for the case with the most data points, this is from the unsteady nature of the CFD solution on which this solution is based.

V. Circulation Control Wing Background

With the algorithm constructed, a visualization suite assembled and an error method tested, the primary mission could be undertaken. The goal implementation of this work was for use in a balanced field length simulation code to incorporate CFD data into the simulation process to increase the fidelity of results. Circulation control wings use the Coanda effect to deflect high speed flow and create an effective increase in airfoil camber thus increasing lift. This effect creates complex nonlinear flow phenomena that can not be captured without use of a viscous computational solver, which is inevitably time consuming. Furthermore, the time necessary becomes unrealistic when the goal is to model an entire takeoff regime. Thus there needs to be some way to approximate the CFD model's behavior with only a few number of data points, a natural fit for a metamodel. This metamodel could then be integrated into the code and integrated over the entire takeoff regime. The final result was a takeoff model that was based on a GP metamodel of CFD data. The takeoff model was within 6 and 10% of FLOPS (a NASA takeoff code) and test data respectively.

Circulation control wings are based around changing the airflow around the wing to provide an artificial camber to the airfoil. A high speed jet flow is ejected from just upstream of the trailing edge, which then remains attached to the rest of the trailing geometry which then turns the flow. The trailing geometry can be a circular trailing edge, a dual radius flap, or some other type of geometry that turns the flow. The objective of the trailing edge component is to deflect the flow as much as possible while keeping the flow on the surface of the airfoil attached as long as possible. The trailing edge selection for this project was a dual radius flap. The aircraft to be modeled for takeoff was a short takeoff and landing (STOL) transport, which typically have fairly high cruise speeds. The advantage of a dual radius flap is that it has better performance at high speeds when the CCW component is not activated. This comes from the more classic, sharp trailing edge compared to the circular trailing edge.

The flow follows the surface of the trailing geometry and ends up with a velocity component that is essentially straight down. CCW is only employed in the low subsonic regime to provide the high lift required for short takeoffs and landings. To accomplish this, the jet flow at the back of the airfoil induces the stagnation point on the leading edge to move underneath the airfoil. This yields the desired camber effect as seen in Figure 8[6].

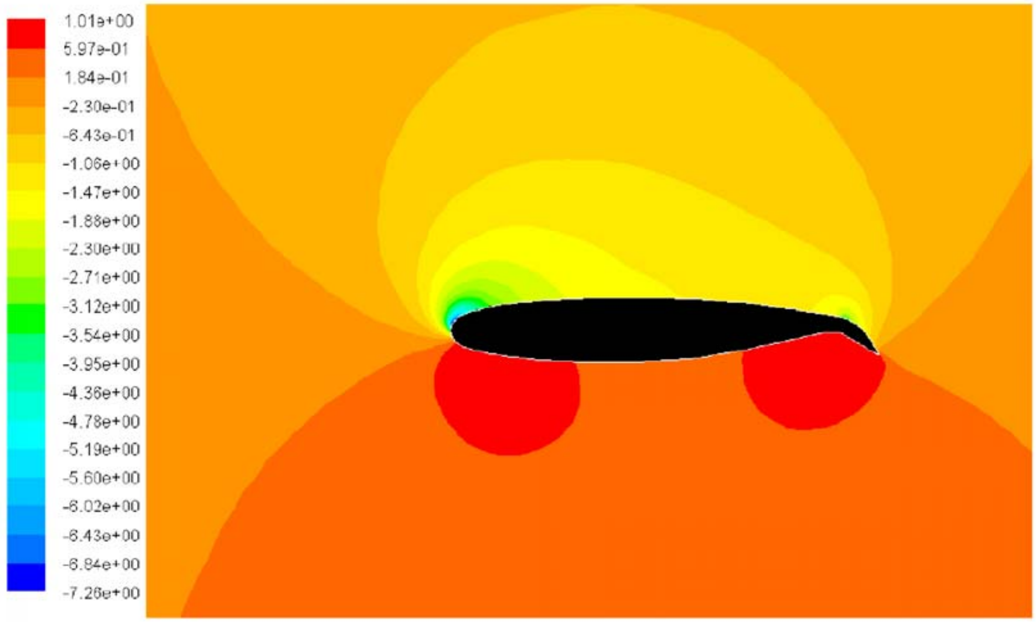


Figure 8: Pressure contours of an example CCW geometry[6]

The aircraft geometry used for this project was based around a Cal Poly / NASA Ames conceptual transport. That transport was designed in the summer of 2004 for the NASA Ames ESTOL sector as an ESTOL study reference vehicle and the planform modified by de la Montanya[6]. The wing planform had a reference area of 72 square meters, 9.7 degrees of leading edge sweep, a wingspan of 26.2 meters, and a mean aerodynamic chord length of 3.1 meters. The airfoil section was a modified SC(2)-0414. The modification was made such that the dual radius flap and the slot could be added to the wing with as little change as possible to the supercritical shape. The slot extended over 80% of the wing in order to maximize the blown area. All issues with controllability and design feasibility were assumed to be addressed by the original designing team.

For the CFD modeling, only the wing, slot and flap were modeled. The body of the aircraft was excluded to cut down on computation time. A full aircraft model would have been the way to get the best results but the idea was to accurately model the CCW and adding in additional geometry lowered the ability to accurately refine the CCW regions. Due to computer constraints the computation limits were being reached with just the wing alone. Additional geometry would have required lowering grid refinement, which would have reduced the likelihood of proving grid independence.

VI. CCW, Takeoff Simulation and Metamodeling

With an experimental design selected, a set of data points was generated and the CFD analysis performed to get lift, drag, and moment coefficient data for each input set. The C_L, C_M, and C_D became the training outputs for the metamodel while the randomly generated points became the training inputs. These points were gathered using a combination of Gambit, for griding, and Fluent, for solving. The turbulence model used was Spallart-Allmares, which was the recommended model for use in CCW applications[3]. This was based on comparison with data from Jones and Joslin[15] and Liu et al.[18] Grid size studies were performed along with verification of input variable effects on the model. Data from the CFD runs was input into a database architecture where it was processed via text based scripts and stored. The stored data could then be pulled out of the database and automatically formatted for use in the Matlab based visualization tools and takeoff simulation.

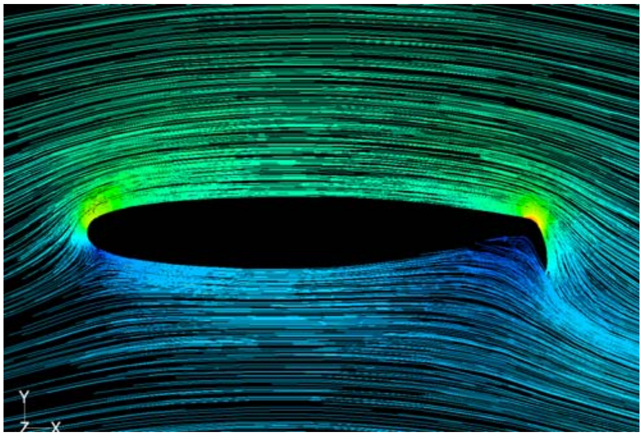


Figure 9: Typical CCW result for takeoff run

Specifically, 40 separate data points were used to define the 4D design space. They were randomly distributed between minimum and maximum values. The results were divided into two categories, typical and atypical. A typical case occurred at a Mach number of 0.15, an angle of attack of 4˚, a mass flow rate of 9.4 kg/s, and a dual radius flap deflection of 66˚. Figure 9 shows streamlines colored by Mach number at a slice mid span. This figure shows the characteristic flow path of a typical circulation control wing. The stagnation point is rotated and the high velocity flow from the slot is ejected out the back and follows the flap down affecting the free stream flow. Although the flap is only at 66˚, the actual flow coming off the trailing edge of the flap continues on that angle essentially extending the flap and increasing the camber. The curvature of the pathlines in the flowfield also indicates this significant increase in camber and therefore a reasonably high lift coefficient.

Although Figure 9 illustrates ideal circulation control characteristics, not all cases in the design space did. These atypical results lead to some interesting flow phenomenon. They began to occur at high angles of attack and high mass flow rates and an example of this can be seen in Figure 5 with the case defined as follows: Mach number of 0.15, angle of attack of 7.9, mass flow rate of 0.8 kg/s (equivalent to about 20 kg/s for the 3-D wing), and a flap deflection of 88˚.

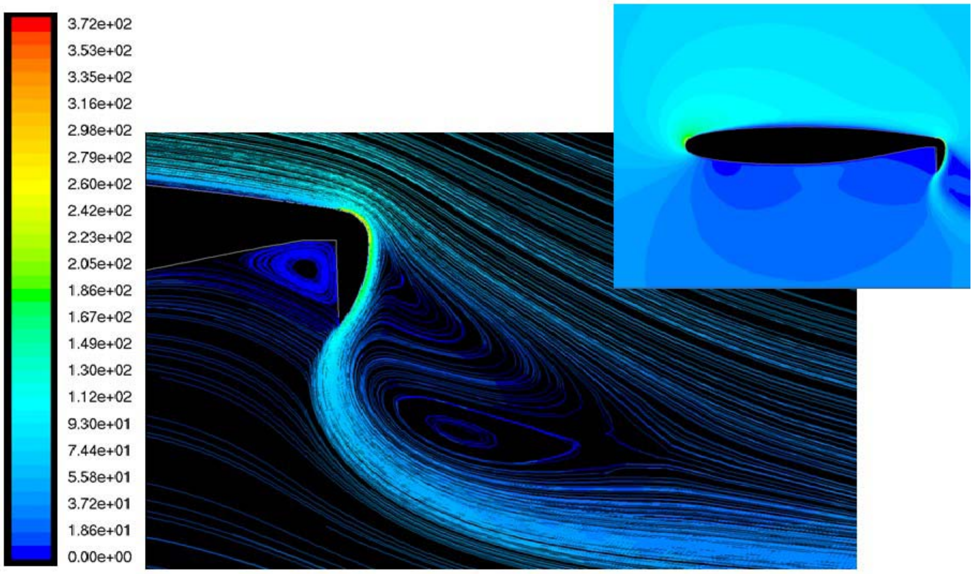


Figure 10: Region of separated flow behind CCW flap at high flap deflections

Figure 10 is a 2-D case, which more clearly shows the behavior of the 3D results for the same conditions. The 3D equivalent plot has intersecting lines because of the projected nature of converting the 3D information into a 2D plot. This case illustrates the near maximum of three of the input parameters: angle of attack, mass flow rate, and flap deflection. These extreme parameters lead to an extreme position of the forward stagnation point and a number of recirculation regions at the trailing edge of the wing. The freestream flow was being forced to turn all the way around the leading edge of the wing and then attempt to turn over 90˚ at the flap. With the exception of the extremely high speed flow coming out of the slot, almost none of the flow was entrained completely around the flap. As a result, the peculiar recirculation region appeared directly aft of the flap. This loss of entrainment ended up significantly reducing the lift generated by the circulation control. However, the net drag ended up decreasing at the high flap deflections. This was the result of the decreased induced drag which decreased more than the added parasite drag from the recirculation region. These results led to the idea that there was an ideal flap deflection in order to produce the maximum amount of lift and that it was less than the maximum deflection. More discussion about ideal flap deflections and mass flow rates will be given in the section regarding the performance of STOL aircraft.

The metamodel was designed to be implemented in the takeoff simulator using the GP algorithm outlined above. Since visualization and error analysis were not needed once the model was determined to be adequate, an optimized GP layout was used for the aerodynamics model. This allowed the simulation to run as fast as possible. Since

BFL is an iterative process the reduction in the run time of each takeoff calculation was important. Shown below in Figure 11 is the block diagram for the takeoff calculation.

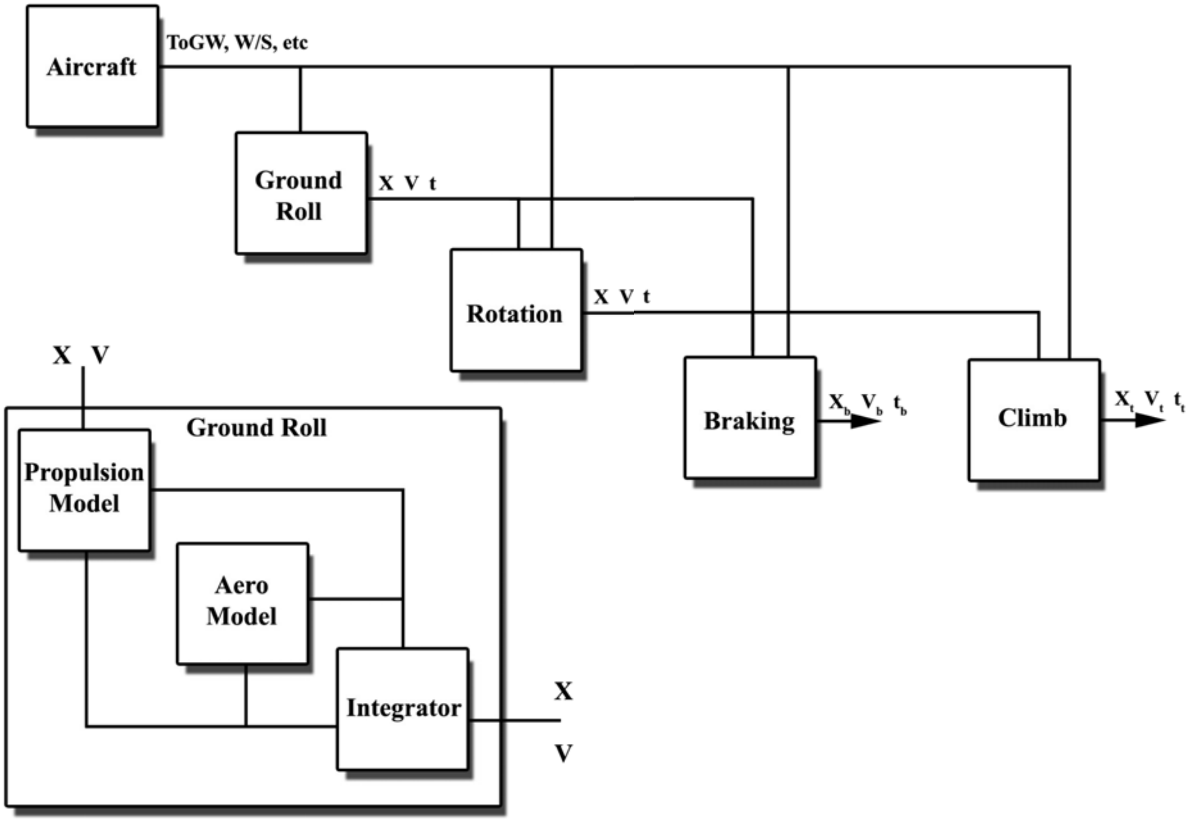


Figure 11: Take off simulation flow chart with inset for the internal flow of the ground roll block. The other blocks all have similar internal layout with their own appropriate physics.

As can be seen from the ground roll block, each section of the takeoff has an aerodynamics GP model included. The GP models were given inputs from the takeoff model that were of the same form as the data used for training. So the takeoff code gives the GP aerodynamics model a Mach number, angle of attack, flap deflection and mass flow. The first two of these are determined by the position on the runway and the equations of motion. The second two are dependent on a user defined function or can be constants. A variety of settings were chosen for the mass flow rate and the flap deflection that attempted to model realistic scenarios, easy to pilot scenarios, and minimized takeoff distance scenarios.

Like any model there were a set of assumptions inherent with the implementation of the takeoff model. The foremost assumption was that rotation occurred instantaneously. This simplified the situation considerably, because the moments don't have to be balanced. This is especially relevant in a CCW setting where there are very strong restoring moments from the large amount of lift generated on the latter portion of the wing. Also, along the same vein there were no control effects taken into account. So deflections required to trim out engine failure conditions and any other moments were not included. The windmilling drag of the failed engine was also not included. However, the modular nature of the code would allow for the implementation of any of these components. The basic integration scheme could also be converted to deal with the additional degree of freedom of rotation.

VII. Metamodeling Results

Before the metamodel could be put into the actual takeoff code, verifications had to be performed that insured the model was a legitimate estimation of the physics represented in the CFD and ideally real life. The first component of this was a qualitative analysis performed during the data collection process. As CFD data points were finished they were sequentially added to the model and the functional behavior plotted using the multiplot tool previously discussed. The lift versus angle of attack curve was a good way to monitor the behavior of the function because it should qualitatively be linear in the low angle of attack regions with a stall region at higher angles of attack. Some early models, especially when not normalized, did not show this trend and thus it was clear additional data was required.

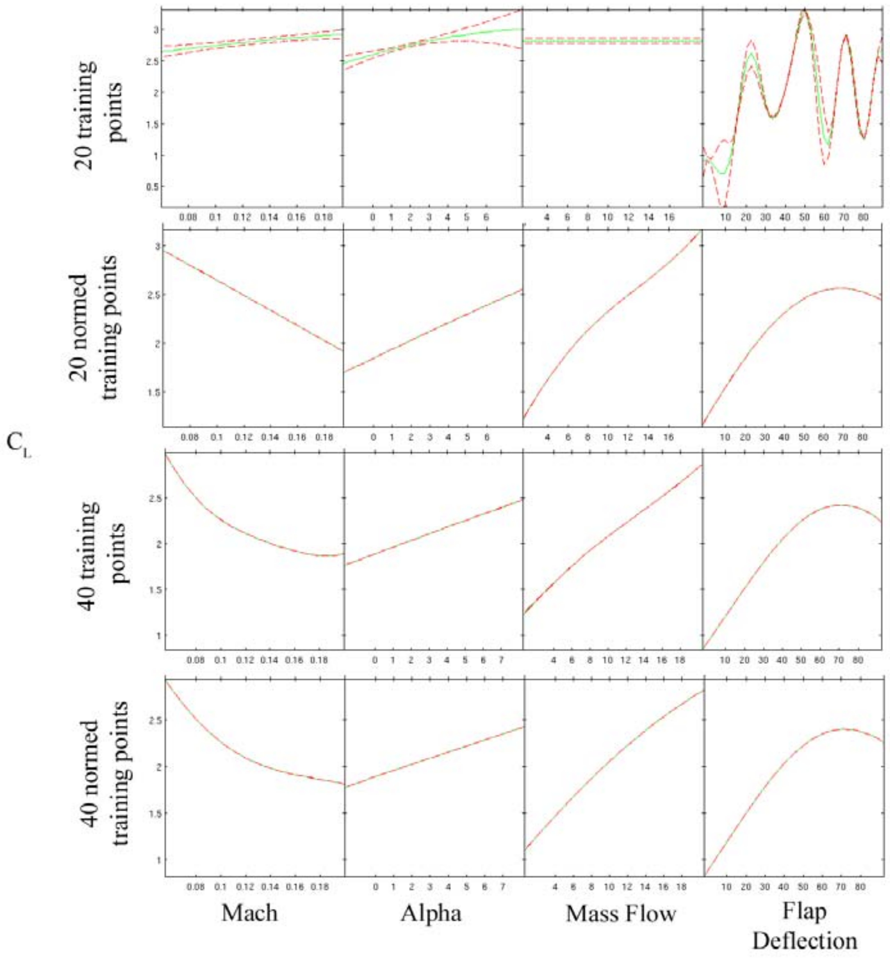
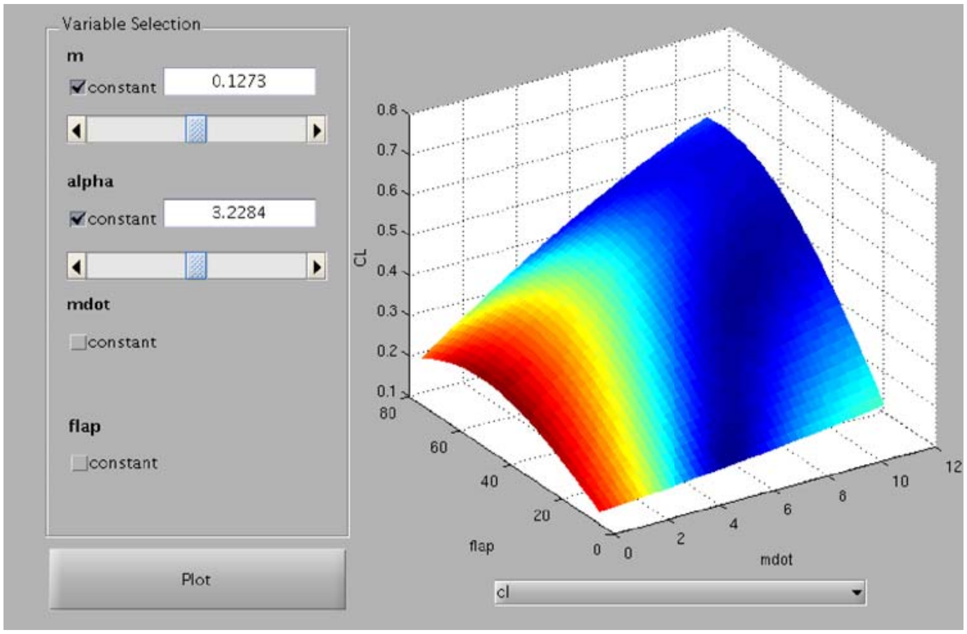


Figure 12: Comparison of different numbers of training points and normalization affect on prediction of CL

The second visualization tool, previously mentioned, was the 3D visualization tool. This tool allowed for visualization of the coupling between variables. This is shown in Figure 13. Slider bars provide a way to change the input of the constant variables so that all ranges of the design space can be explored. While this essentially shows the same information as the multiplot tool it allowed for better visualization of the coupling between different points. This was used to help a user quickly judge the behavior of a model, identify the important areas, and then go to the multiplot tool to get specific numbers. The surface was colored on the log of predictive variance to allow for visualization of the worst regions of a surface. The log was used because it increased the range of the predictive variances and made for a more instructive color distribution on the surface. Poor predictive variances are an indication of very few points in that region.

As more and more points were added to the model the error estimation tool had to be used to verify the change in the model. This provided a quantitative basis for establishing the models validity. In practice, these steps were followed with a couple practical differences. Two batches of data were collected each of 20 data points. Since development of the GP codes was undertaken at the same time as the first CFD runs were being completed, some of the diagnostic tools were not available for the first batch. The multiplot tool was the chief tool for analysis of the first 20 points since it provided a clear view as to the functional behavior across the input space. It was determined from the plots, especially when not normalized, that the behavior seemed to be erratic and that more data points were needed. Another batch of 20 points was added to the first dataset. Again, this could not have been done with any type of factorial design since factorial designs prescribe where you put all of the points and are not valid (as a full factorial) until the whole set of data is completed. The addition of the final 20 points fixed many model issues and was verified with the, then recently finished, error analysis tool. The error analysis for the full data set showed that LOO method and the worst case error method showed a convergence around 30-35 points respectively. With a finalized GP model set, the regression could be used to model the aerodynamics for the takeoff simulation.


Variable Selection

m

☑ constant 0.1273

alpha

☑ constant 3.2284

mdot

☐ constant

flap

☐ constant

Figure 13: 3D visualization tool for circulation control wing example

VIII. Takeoff Results

The next step was to determine how close the results of the model, without integration of the metamodel, were to actual vehicle results. This was required before the metamodel based version could be analyzed. Two validation cases were taken from an undergraduate study of preliminary takeoff analysis programs and the data used for validation of this takeoff code[19]. The goal of the report was to compare the known values for aircraft to the results from numeric solutions to the physical equations of motion. The known results which were used to validate the current model included data for the McDonnell Douglas DC-9 from NASA Langley's Flight Optimization System (FLOPS) and also flight test data for a Boeing 747-400[14]. Those results were compared to two existing methods, the simplified method proposed by Powers and the modified version of the method proposed by Krenkel and Salzman[17].

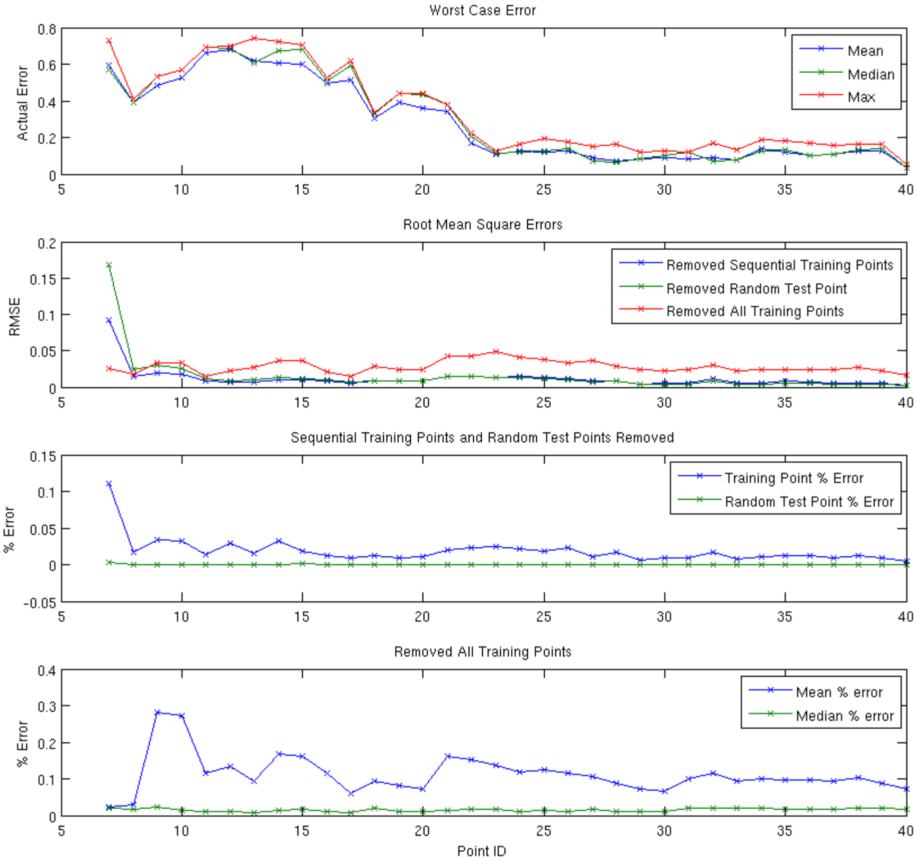


Figure 14: Error metrics for CCW metamodel

The DC-9 FLOPS results included BFL results so it was used as the first stress test for the takeoff model. All of the aircraft parameters that were input into the FLOPS model were also included with Lynn and were input into the BFL model. These parameters included the quadratic thrust model, thrust, weight, wing area, and the assumed C_L and C_D for the different phases of takeoff. The detailed circulation control aerodynamic block was taken out and replaced with a model that attempted to match the physics outlined by the take off report. The major difference between the two codes was a rotation assumption that was implemented to insure rotation of powered lift aircraft. For more details on the derivation and necessity of this method see Ball[3]. FLOPS was assumed to use the traditional method outlined in sources such as Torenbeek[29]. With all of the data entered into the current model, the BFL was calculated, Figure 15. Although the different rotation velocities tended to vary up to 12% from FLOPS, the overall distances were much closer with takeoff and BFL being about 4 and 7% different respectively.

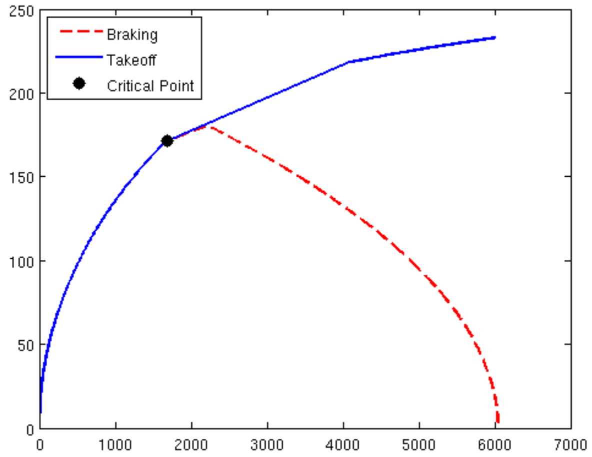


Figure 15: DC-9 Validation BFL

The other validation case was for the takeoff run only and was compared to flight test results of a 747-400. Since the known results were from a flight test, it was difficult to reconstruct an actual airplane to fit into a preliminary takeoff model. Fortunately some of the references used also had the same flight data available for the 747-400. The same assumptions about thrust, lift, and drag which were used from Lynn[19], in the Powers and modified Krenkel and Salzman codes. The results matched test results within about 5% for the takeoff distance. However, there did exist a discrepancy for their lift off distance with the Powers method producing a distance over 20% shorter than the given 7,500 ft. This was taken as an acceptable difference. Initially, there was a significant (about 18%) difference between the takeoff distances. It was thought that it was possibly the modified rotation calculation causing some of the error so the case was run again with the traditional method for calculating rotation velocity based on the given C_{Lmax}. Those results can be seen in Figure 16. With the traditional rotation velocity calculation, the results matched very well with the flight test with error of the overall takeoff distance decreasing from 18 % to 6% and the other velocities all matching within 3% or better. Although the traditional rotation calculations agreed better with the test data in this case, more research will have to be put forth to improve the idea that the rotation speed can be a function of lift over weight rather than solely C_{Lmax}. Englar[7,8,9] has also addressed this problem and future results may encompass some of his work to better improve the rotation model. It is also worth noting that some of the error could have originated from the input data for the 747 since the from Powers also differed greatly from the test data. More test data or FLOPS results are needed to fully validate the model and the modified rotation velocity calculations. However, the model itself has shown that it works well at predicting takeoff performance for some traditional aircraft.

Results: Performance of STOL Aircraft

With the BFL code verified, a flap deflection and mass flow rate study was completed to determine optimal settings for BFL reduction. The result was that balanced field lengths as low as 2,400 feet could be achieved but those were obtained with diminishing returns since

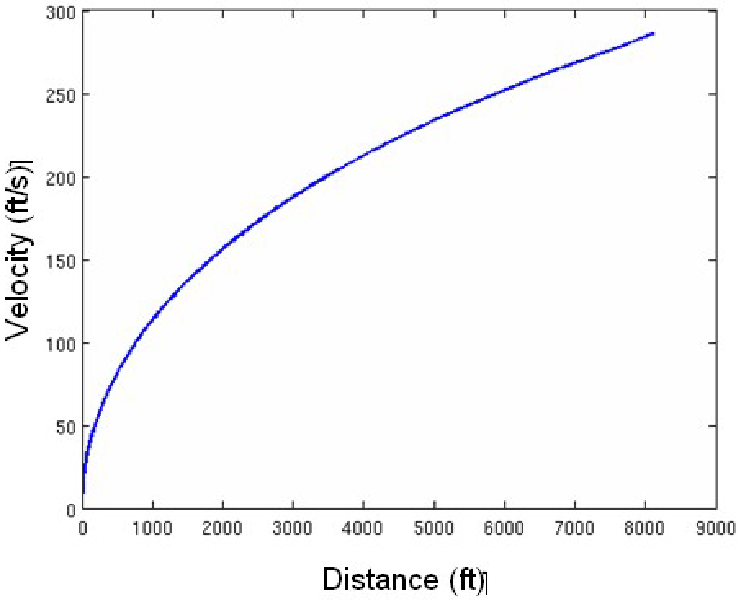


Figure 16: 747 Takeoff validation

mass flows of 20% less could obtain a BFL of 2,500 feet. This analysis was based on a STOL transport designed for the 2007 AIAA undergraduate design competition. This gave a design for which numbers for drag, engine sizing, and airfoil performance were well known. Perhaps most importantly, it gave a thrust, weight and wing area that were designed for a 2,500 foot BFL, which helped with debugging and is also the range of current industry STOL transport work. The thrust was increased above the original design up to 130,000 lbf, to provide a wider range of feasible takeoff conditions. The results of the mass flow and flap deflection study are shown in Figure 17.

As the flap deflection was increased there became a point where a minimum BFL was reached for any particular mass flow rate. This was due to the loss of super circulation caused by separation, not over the entire wing, but just aft of the flap. One might expect that there would be a sharp change at this point due to increased drag caused by the recirculation, but after looking at the metamodel and data points, it was noticed that the total drag actually goes down in the regions that experience separated flow off of the flap. The reason the total drag decreases is the loss of lift and reduction of induced drag. This decrease in drag helps BFL while the loss in lift hurts. The net overall effect is that the BFL ends up changing smoothly from one radius of curvature to a different radius once flap separation occurs. Basically the BFL distance changes at a slower rate with large flap deflections when compared to the small flap deflections.

The mass flow rate term, unlike the flap, always provides an additional benefit to BFL, but the amount by which it improves is clearly diminishing. As mentioned above, for a 22% decrease in mass flow, you lose only 4% of your balanced field length (2,400 feet to 2,500 feet). A possible explanation of this is that with additional flow the shear layer has a higher gradient across it and little additional energy is transferred into the adjoining flow. And finally, it is interesting to note that for high mass flow rates, the effect of flap deflection decreases.

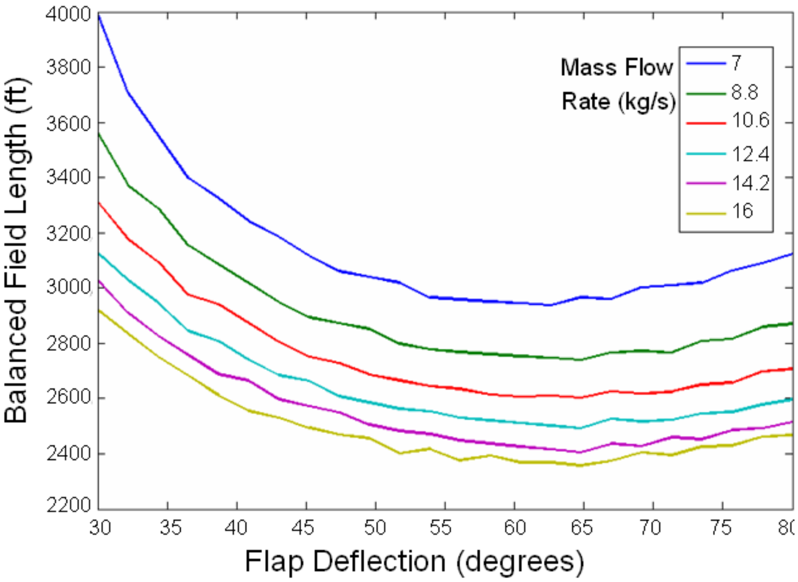


Figure 17: Variation in BFL for STOL concept with varying mass flow and flap deflection

A number of additional problems could be addressed now that there was an aerodynamics model that represented CCW geometry. Particularly, the effect of engine coupling could be examined since many CCW configurations require engine bleed to feed air to the high speed air slot. While he control effects were still not accounted for, just taking into account the coupling between engine and mass flow changed the BFL. Coupling in the engine hurt the BFL when compared to a variety of constant mass flow situations, Figure 18.

The takeoff simulation was useful because it captured the flow physics of a CCW aircraft and allowed for important performance predictions to be made. However, a very challenging issue with CCW aircraft is the source of the mass flow for the CC system. The mass flow can come from one of two sources, either dedicated gas generation or bleed air from the engines. Both present significant design issues. An embedded gas generator system has the addition of complex engine components, which add in additional challenges in structural layout, fuel placement, and catastrophic failure recovery, with the benefit being a decoupling of thrust and lift. The main engine bleed scenario results in a complex problem of ducting, thrust loss during acceleration, over sized main engines, and some very serious control issues due to the coupling of lift and thrust. The loss of a main engine results in substantial pitch, roll, and yaw moment on the aircraft. The drastic loss of lift on one wing would induce an obvious roll moment, the change in pitching moment coefficient would cause the pitch moment, and the engine out scenario would cause the yaw moment, though the adverse yaw induced from the roll would help fight the engine out yaw moment. Nonetheless, designers often elect to duct bleed air from the main engines in order to fight cost and keep the design as simple as possible. This requires that the takeoff analysis have coupling between the engines and the aerodynamic system.

Cal Poly, San Luis Obispo recently received a NASA grant for design of a n+2 generation STOL transport testbed that would test different advanced STOL concepts. It was logical that the previous work on STOL transport takeoff analysis and metamodeling would be applied to this new project wherever possible. The first place recognized was in the propulsion modeling and integration with the takeoff model.

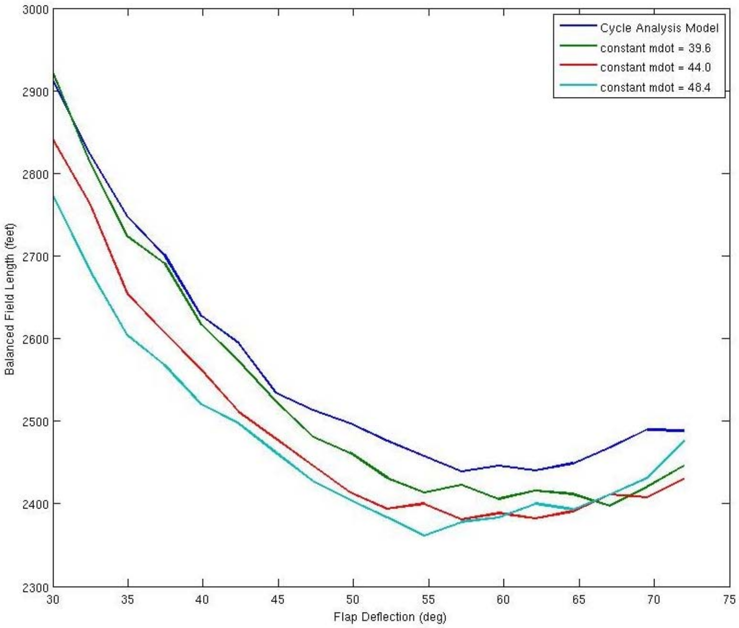


Figure 18: Adjust BFL calculations for coupled aero model

This analysis would ideally help size engines and feed into the design process early on, since all that is needed is the analytic based propulsion model and the CFD based aerodynamics model. As mentioned earlier, the takeoff code is modular enough to allow for substitution of different code components if a better model becomes available.

IX.Acknowledgments

This work was sponsored in part by the Department of the Navy, Office of Naval Research, under Award #N00014-06-1-111.

X. Conclusion

There is potential for a variety of metamodeling investigations that build on the work presented in this paper. There is much work to be done regardless of whether the focus is on the statistical methods, further tool development, or additional case studies. Of particular interest are investigating different ways to further model the balanced field length scenario to understand the benefits and costs of a CCW design. The results of this paper show that advanced engineering concepts can be incorporated into the design process earlier with this type of analysis practice. The small number of points required to get reasonable metamodels was the key component that made the Gaussian process regression approach an attractive and practical method. A response surface would have washed out the fine details that make the CFD solutions important and a spline model, would have taken many additional points to resolve the behavior.

References

[1]Abramson, J., and E.O. Rogers. *High-Speed Characteristics of Circulation Control Airfoils*. AIAA-83-0265.

[2]Anderson, John D. *Aircraft Performance and Design*. McGraw-Hill Science Engineering, New York. Ch 6. 1998.

[3]Ball, Tyler, Turner, Scott, and Marshall, David, D., “Short Takeoff Performance using Circulation Control,” *Proceedings of the 46th AIAA Aerospace Sciences Meeting and Exhibit,* American Institute of Aeronautics and Astronautics, Reno, NV 2008.

[4]Box, George E.P., Draper, Norman R. *Reponse Surfaces, Mixtures, and Ridge Analysis.* John Wiley & Sons, Inc., Hoboken New Jersey, 2007.

[5]DeLoach, Richard, “Bayesian Inference in the Modern Design of Experiments,” *Proceedings of the 46th AIAA Aerospace Sciences Meeting and Exhibit,* American Institute of Aeronautics and Astronautics, Reno, NV 2008 pp. 2-19.

[6]de la Montanya, Julianna Belle, “Circulation Control and its Applications to Extreme Short Take-Off and Landing Vehicles,” Master Thesis, Department of Aerospace Engineering, California Polytechnic State University, San Luis Obispo, California, 2006.

[7]Englar, Robert J. “Overview of Circulation Control Pneumatic Aerodynamics: Blown Force and Moment Augmentation and Modification as Applied Primarily to Fixed-Wing Aircraft,” Progress in Astronautics and Aeronautics, vol.214, American Institute of Aeronautics and Astronautics, pp. 23-64, 2006.

[8] Englar, Robert J., and Gregory G. Huson. “Development of Advanced Circulation Control Wing High-Lift Airfoils.” Journal of Aircraft, Vol. 21, No. 7, (1984): 476-483.

[9]Englar, Roert J., Smith, Maryillyn J., Kelley, Sean M., Rover, Richard C, III. *Development of Circulation Control Technology for Application to Advanced Subsonic Transport Aircraft*. Aerospace Laboratory, Georgia Tech Research Institute, Atlanta, GA. AIAA 1993-0644.

[10]Federal Aviation Administration. Regulatory and Guidance Library.. Washington D.C. 2007. Part 25: 105, 107, 109, 111, 113, 121. http://rgl.faa.gov/Regulatory_and_Guidance_Library.

[11]Forta, Ben. *MySQL Crash course*, Sams Publishing, USA, 2006

[12]Grafen, Alan, and Hails, Rosie, *Modern Statistics for the Life Sciences*, Oxford University Press, New York, 2002.

[13]Hall, Dave et. al. Summer 2004 CalPoly/NASA ESTOL Work Presented to NASA/Ames ESTOL Vehicle Systems Sector. PowerPoint Presentation. August 27, 2004.

[14]Hank, C.R., et al, "The Simulation of a Jumbo Jet Transport Aircraft. Volume 2: Modeling Data," NASA CR 114494, also N73-10027 Sept. 1970.

[15]Jones, Gregory S., and Ronald D. Joslin. *Introduction: 2004 NASA/ONR Circulation Control Workshop*. 2004 NASA/ONR Circulation Control Workshop, March 16-17, 2004, NASA/CP-2005-213509.

[16]Koehler, J.R., Owen, A.B. **"**Computer Experiments," *Handbook of Statistics,* Elsevier Science, vol.13, pp.261-308, 1996.

[17]Krenkel, A.R., Salzman, A., "Takeoff Performance of Jet-Propelled Conventional and Vectored-Thrust STOL Aircraft," *Journal of Aircraft*, Vol. 5, No. 5, 1968.

[18]Liu, Yi, Sankar, Lakshmi N., Englar, Robert J., Ahuja, Krishan K. *Numerical Solutions of the Steady and Unsteady Aerodynamic Characterishtics of a Circulation Control Wing.* GTRI Report A5928/2003-1 App. B. Georgia Tech Research Institute, GA 2003.

[19]Lynn, Sean. *Summary Report for an Undergraduate Research Project to Develop Programs for Aircraft Takeoff Analysis in the Preliminary Design Phase*, Undergraduate Research Project, Virginia Polytechnic Institute and State University. Blacksburg, VA. May 11, 1994.

[20]Mayfield, Jerry. "Circulation Control Demonstrates Greater Lift." Aviation Week and Space Technology, March 19, 1979.

[21]Mechesheimer, Martin, Booker, Andrew J., Barton, Russell R., and Simpson, Timothy W., "Computationally Inexpensive Metamodel Assessment Strategies," *AIAA Journal*, Vol. 40, No. 10, 2002.

[22]McDonald, Robert A. *Error Propagation and Metamodeling for a Fidelity Tradeoff Capability in Complex System Design*. Ph.D Thesis, Georgia Institute of Technology, GA, 2006.

[23]McLister, Bob, Boswell, Matthew. "0-60 Times and 0-60 Comparisons," Performance Car News, [http://www.performancecarnews.com/ Accessed 4/22/2008]

[24]Percey, Bobbitt J., Eagle Aeronautics, Margason, Richard. *Analysis of the Take-off and Landing of Powered Lift Aircraft*. American Institute of Aeronautics and Astronautics, Inc., 2007.

[25]Powers, S. A., "Critical Field Length Calculations for Preliminary Design," *Journal of Aircraft*, Vol. 18, No. 2, 1981.

[26]Rasmussen, Carl Edward, Williams, Christopher K.I. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge Massachusetts, 2006.

[27]Raymer, Daniel P. *Aircraft Design: A Conceptual Approach Fourth Edition*. American Institute of Aeronautics and Astronautics, Inc., Virginia, 2006

[28]Simpson, T.W., Peplinski, J.D., Koch, P.N., Allen, J.K., "Metamodels for Computer-based Engineering Deisgn: Survey and recommendations," *Engineering with Computers*, Vol. 17, 2001, pp. 129-150.

[29]Teorey, Toby, Lightstone, Sam, Nadeau, Tom. *Database Modeling and Design: Logical Design*. Elsevier Inc, San Francisco, 2006.

[30]Torenbeek, Egbert. Synthesis of Subsonic Airplane Design. Delft University Press, Delft, The Netherlands. Ch5 and App. K 1982.