Assessment of Model Validation, Calibration, and Prediction Approaches in the Presence of Uncertainty

Nolan W. Whiting

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Aerospace Engineering

Christopher J. Roy, Chair
Heng Xiao
Leanna L. House

May 21, 2019

Blacksburg, Virginia

Keywords: Verification and Validation, Uncertainty Quantification, Prediction, Computational Fluid Dynamics,

Assessment of Model Validation, Calibration, and Prediction Approaches in the Presence of Uncertainty

Nolan W. Whiting

(Abstract)

Model validation is the process of determining the degree to which a model is an accurate representation of the *true value* in the real world. The results of a model validation study can be used to either quantify the model form uncertainty or to improve/calibrate the model. However, the model validation process can become complicated if there is uncertainty in the simulation and/or experimental outcomes. These uncertainties can be in the form of aleatory uncertainties due to randomness or epistemic uncertainties due to lack of knowledge. Four different approaches are used for addressing model validation and calibration: 1) the area validation metric (AVM), 2) a modified area validation metric (MAVM) with confidence intervals, 3) the standard validation uncertainty from ASME V&V 20, and 4) Bayesian updating of a model discrepancy term. Details are given for the application of the MAVM for accounting for small experimental sample sizes. To provide an unambiguous assessment of these different approaches, synthetic experimental values were generated from computational fluid dynamics simulations of a multi-element airfoil. A simplified model was then developed using thin airfoil theory. This simplified model was then assessed using the synthetic experimental data. The quantities examined include the two dimensional lift and moment coefficients for the airfoil with varying angles of attack and flap deflection angles. Each of these validation/calibration approaches will be assessed for their ability to tightly encapsulate the true value in nature at locations both where experimental results are provided and prediction locations where no experimental data are available. Generally it was seen that the MAVM performed the best in cases where there is a sparse amount of data and/or large extrapolations and Bayesian calibration outperformed the others where there is an extensive amount of experimental data that covers the application domain.

Assessment of Model Validation, Calibration, and Prediction Approaches in the Presence of Uncertainty

Nolan W. Whiting

(General Audience Abstract)

Uncertainties often exists when conducting physical experiments, and whether this uncertainty exists due to input uncertainty, uncertainty in the environmental conditions in which the experiment takes place, or numerical uncertainty in the model, it can be difficult to validate and compare the results of a model with those of an experiment. Model validation is the process of determining the degree to which a model is an accurate representation of the *true value* in the real world. The results of a model validation study can be used to either quantify the uncertainty that exists within the model or to improve/calibrate the model. However, the model validation process can become complicated if there is uncertainty in the simulation (model) and/or experimental outcomes. These uncertainties can be in the form of aleatory (uncertainties which a probability distribution can be applied for likelihood of drawing values) or epistemic uncertainties (no knowledge, inputs drawn within an interval).

Four different approaches are used for addressing model validation and calibration: 1) the area validation metric (AVM), 2) a modified area validation metric (MAVM) with confidence intervals, 3) the standard validation uncertainty from ASME V&V 20, and 4) Bayesian updating of a model discrepancy term. Details are given for the application of the MAVM for accounting for small experimental sample sizes. To provide an unambiguous assessment of these different approaches, synthetic experimental values were generated from computational fluid dynamics (CFD) simulations of a multi-element airfoil. A simplified model was then developed using thin airfoil theory. This simplified model was then assessed using the synthetic experimental data. The quantities examined include the two dimensional lift and moment coefficients for the airfoil with varying angles of attack and flap deflection angles. Each of these validation/calibration approaches will be assessed for their ability to tightly encapsulate the true value in nature at locations both where experimental results are provided and prediction locations where no experimental data are available. Also of interest was to assess how well each method could predict the uncertainties about the simulation outside of the region in which experimental observations were made, and model form uncertainties could be observed.

Acknowledgements

This work was supported by Intelligent Light (Dr. Earl Duque Project Manager) as part of a Phase II SBIR funded by the U.S. Department of Energy, Office of Science, Office of Advance Scientific Computing Research, under Award Number DE-SC0015162. This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

The authors would like to thank Cray Inc. for provided access to their corporate Cray XE40 computer, Geert Wenes of Cray Inc. for helping to acquire access and David Whitaker from Cray Inc. for assistance in porting of OVERFLOW2 to the XE40 and for streamlining the use of FieldView on their system. Special thanks to Dr. Heng Xiao and Dr. Jinlong Wu for providing their insight on Bayesian updating, and Prof. James Coder at the University of Tennessee in Knoxville for providing the setup of the OVERFLOW2 runs used for establishing the synthetic experimental data used in this study.

Table of Contents

Nomenclature

AVM	=	Area Validation Metric
CDF	=	Cumulative Distribution Function
c_l	=	2-D lift coefficient
$c_{m(c/4)}$	=	2-D moment coefficient about the quarter chord
D	=	mean experimental result
d	=	area validation metric
d^-	=	area validation metric for area smaller than simulation CDF (MAVM)
d^+	=	area validation metric for area larger than simulation CDF (MAVM)
dz/dx	=	derivative of the mean camber line with respect to x
E	=	model error
F(Y)	=	simulation CDF curve
f_*	=	single Gaussian Process posterior realization
GP	=	Gaussian Process
k	=	coverage factor
k(x,x')	=	covariance matrix between x and x'
l	=	length scale
MAVM	=	Modified Area Validation Metric
m(x)	=	mean function of the Gaussian Process prior
N	=	number of samples
n_{obs}	=	number of observations
$S_n(Y)$	=	experiment CDF curve
S	=	mean simulation result
s^2	=	variance in experimental data
SRQ	=	System Response Quantity
u_D	=	experimental data uncertainty
u_{input}	=	input uncertainty
u_{num}	=	numerical uncertainty
u_{val}	=	validation uncertainty
X	=	locations for which data are available in Bayesian updating
X_*	=	locations for which model discrepancy is predicted in Bayesian updating
Φ_1	=	conservativeness
Φ_2	=	tightness
$\Phi_{2,v}$	=	tightness for validation method
$\Phi_{2,c}$	=	tightness for calibration method
Φ	=	overall assessment
α	=	angle of attack, °
δ	=	flap deflection, °
δ_{model}	=	model discrepancy
σ_n^2	=	variance in observation model discrepancies

This thesis is an extension of an AIAA paper [24] authored by myself and Christopher Roy. Coauthors of the paper were Earl Duque and Seth Lawrence who provided the CFD results that were used to conduct this study.

1. Introduction

Mathematical models often fail to predict the true value in nature. Many models are useful as they can adequately predict real world physics, but nearly always have some uncertainty attached with their results. This uncertainty can come from many factors. It could be introduced from uncertainty in the measured inputs in the model or experiment. It could be a result of errors or uncertainties in the experimental data. There could be an error in the computer software that was created based off of the model. It can also be due the model's poor prediction capability relative to the actual value in a certain part of the operating space. Regardless of the reason, it is important to validate models in comparison with experimental or real world results. A validation experiment is an experiment conducted with the primary purpose of assessing the predictive capability of a model. Validation experiments differ from traditional experiments used for exploring a physical phenomenon or obtaining information about a system because the customer for the experiment is the model which is generally embodied within a simulation code. Six guidelines that define a good validation experiment are the presence of a joint computational-experimental effort, the measurement of all needed model input data, synergism between computation and experiments, independence/dependence between computation and experiment, hierarchy of experimental measurements, and estimation of experimental uncertainty [1].

One of the major sources of uncertainty in the final simulation system response quantities (SRQs) is due to the uncertainty in the model/experimental inputs. In these terms, there are typically two kind of uncertainties. The first being aleatory uncertainty in which some probability distribution is known about the input uncertainty (e.g. normal). The second is epistemic uncertainty which represents a lack of knowledge about the true value as it could be either characterized as an interval or a probability distribution.

1.1 Experimental Data Uncertainty

There are two types of measurement errors: random measurement errors and systematic (or bias) errors. The experimental uncertainty due to random error sources can be reduced by adding additional replicate measurements, with the uncertainty scaling as $1/\sqrt{N}$, where N is the number of experimental replicates. Bias errors, when estimated accurately, can be removed from the measurement via experimental calibration procedures. Unknown or estimated bias errors are usually converted to random errors via design of experiments [2] or other blocking techniques [3].

The uncertainty in an experimental measurement is the root sum square of the standard systematic uncertainty and the random uncertainty. See References [4-6] for details. Reported experimental uncertainty usually refers to a confidence level on the mean value. For example, a measurement reported with 10% uncertainty generally means that the true value (i.e., the actual mean value found in nature) is within the stated interval (reported value +/- 10%) with a confidence interval of 95%. However, in practice, systematic errors are often ignored and replicate measurements are seldom performed. In such cases, the stated uncertainty usually refers to the quoted uncertainty given by the measurement device manufacturer, a value that is almost always much smaller than the true experimental measurement uncertainty.

For MFU methods such as the Kennedy and O'Hagan model discrepancy and the ASME V&V 20 approaches, the true 95% confidence interval uncertainty is needed. However, for the Area Validation Metric approaches, a cumulative distribution function (CDF) of the experimental data is needed. In this latter case when the actual values of the experimental replicates are not given, the confidence intervals can be converted back to a CDF by assuming an underlying distribution for the population of measurements. A Gaussian (or normal) distribution is often assumed.

1.2 Automation of MFU Methods

High-level algorithms for implementing three of the MFU estimation methods discussed in Section 3 are given below. The MATLAB codes following these algorithms is given in the Appendix.

Algorithm 1: *Modified Area Validation Metric*

1. Compute: $p_F = \frac{1}{S}$ — ➢ Determine individual probability for simulation SRQ
2. Compute: $p_{Sn} = \frac{1}{N}$ — ➢ Determine individual probability for experimental samples
3. **for** j = 1 to N **do**
4. **while** i*p_F < j*p_{Sn} **do**
5. Compute: $\mathbf{d}(i) = (\mathbf{S}_n(\mathbf{Y}(j)) - \mathbf{F}(\mathbf{Y}(i)))*\mathbf{p}_F$ — ➢ Determine area between individual simulation SRQ and experiment CDF
6. Compute: i = i + 1 — ➢ Increase simulation SRQ index
7. **if** d(i) > 0 **do**
8. Compute: $\mathbf{d}^+ = \mathbf{d}^+ + \mathbf{d}(i)$ — ➢ Sum area greater than simulation CDF
9. **else do**
10. Compute: $\mathbf{d}^- = \mathbf{d}^- + \mathbf{d}(i)$ — ➢ Sum area less than simulation CDF
11. **end if**
12. **end while**
13. **end for**

Algorithm 2: *ASME's Standard Validation Uncertainty*

1. Compute: $\mathbf{E} = \text{mean}(\mathbf{S}_n(\mathbf{Y})) - \text{mean}(\mathbf{F}(\mathbf{Y}))$ — ➢ Determine error between simulation and experiment
2. Compute: $\mathbf{u}_{data} = 1.96\frac{std(S_n(Y))}{N-1}$ — ➢ Determine uncertainty in experimental data
3. Compute: $\mathbf{u}_{input} = \text{std}(\mathbf{F}(\mathbf{Y}))$ — ➢ Determine uncertainty in simulation due to nondeterministic inputs
4. Compute: $\mathbf{u}_{num} = \mathbf{u}_{ro} + \mathbf{u}_{iter} + \mathbf{u}_{DE}$ — ➢ Determine numerical uncertainty in simulation
5. Compute: $\mathbf{u}_{val} = \sqrt{\mathbf{u}_{data}^2 + \mathbf{u}_{input}^2 + \mathbf{u}_{num}^2}$ — ➢ Determine overall validation uncertainty

Algorithm 3: *Bayesian Updating (Kennedy and O'Hagan, 2001)*

1. **for** i = 1 to max(l)
2. $\mathbf{K}_o(\mathbf{X}, \mathbf{X}) = \boldsymbol{\sigma}_o^2 \exp(-\frac{|X - X^T|}{2l(i)^2})$ — ➢ Determine covariance matrix for observations
3. $\ln(\mathbf{p}(\mathbf{y}_o|\mathbf{x}_o,\boldsymbol{\sigma}_o,l)) = -\frac{1}{2}\boldsymbol{y}_o^T\boldsymbol{K}_o\boldsymbol{y}_o - \frac{1}{2}ln(\det(\mathbf{K}_o)) - \frac{n_{obs}}{2}\ln(2\pi)$ — ➢ Sample different *l* values for maximum likelihood
4. **end for**
5. $\mathbf{K}(X_*, X) = \boldsymbol{\sigma}_o^2 \exp(-\frac{|X - X_*|}{2l^2})$ — ➢ Determine covariance matrix for observations and predictions
6. $\mathbf{K}(X_*, X_*) = \boldsymbol{\sigma}_o^2 \exp\left(-\frac{|X_* - X_*^T|}{2l^2}\right)$ — ➢ Determine covariance matrix for predictions
7. $\overline{\boldsymbol{f}_*} = \boldsymbol{K}(X_*, X)[\mathbf{K}(X, X) + \boldsymbol{\sigma}_n^2 I]^{-1}\mathbf{y}$ — ➢ Determine mean posterior function
8. $\text{cov}(\mathbf{f}_*) = \mathbf{K}(X_*, X_*) - \mathbf{K}(X_*, X)[\mathbf{K}(X, X) + \boldsymbol{\sigma}_n^2 I]^{-1}\mathbf{K}(X, X_*)$ — ➢ Determine covariance of mean posterior function
9. Compute: $\mathbf{L} = \text{chol}(\text{cov}(\mathbf{f}_*))$ — ➢ Cholesky decompose posterior covariance matrix
10. **for** j = 1 to n **do**
11. Compute: $\mathbf{f}_*(j) = \mathbf{L}*\text{randn}(\text{length}(X^*),1)$ — ➢ Sample posterior realization
12. **end do**

1.3 Total Prediction Uncertainty

When input uncertainties are propagated through a model, the resulting uncertainty structure for System Response Quantities (SRQs) is a probability distribution (PDF or CDF) when only aleatory (i.e., random) uncertainties are present in the inputs, or when epistemic (i.e., lack of knowledge) uncertainties are characterized by probability distributions. For the case where epistemic uncertainties are characterized by intervals (a more conservative approach), the resulting structure for SRQs is a probability box, or p-box.

The two additional sources of total prediction uncertainty are numerical uncertainty [7] and MFU (the main topic of this paper). Both of these sources are generally characterized as intervals about the simulation result. Consider an example from Reference [8] where there is both aleatory and epistemic uncertainty in the model inputs. The aleatory uncertainty is characterized probabilistically and the epistemic uncertainty is characterized as an interval. These uncertainties are propagated through the model using segregated uncertainty propagation resulting in a p-box for the SRQ (in this case, thrust produced by a rocket nozzle) as indicated by the blue shaded region in Fig 1. This p-box represents the family of all possible CDFs that can exist within its bounds. The outer bounding shape of the p-box in Fig 1 is due to the probabilistically-characterized input uncertainty and the width of the p-box is due to the interval-characterized input uncertainty.

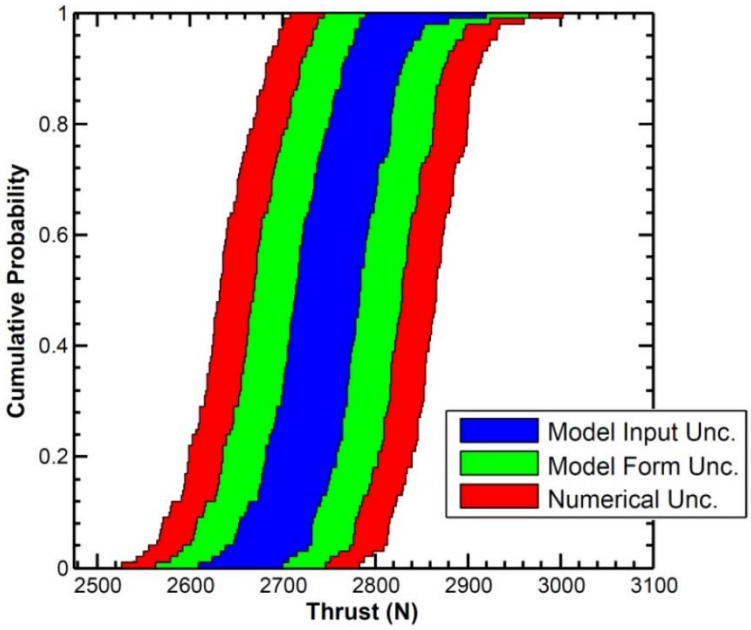


Fig 1. Example of total prediction uncertainty represented as an extended p-box (reproduced from Ref. [8]).

If the p-box of the SRQ resulting from propagating both the random and the interval-characterized model input uncertainties through the model is denoted by $F(x)$, then accounting for MFU (U_{MODEL}) and numerical uncertainty (U_{NUM}) would result in an extended p-box $F(x \pm U_{TOTAL})$ where $U_{TOTAL} = U_{MODEL} + U_{NUM}$. That is, the left side of the initial p-box resulting from the propagation of input uncertainties is displaced negatively by U_{TOTAL}, and the right side of the p-box is displaced positively by U_{TOTAL}, to obtain the extended p-box for the SRQ. In Fig 1, the contribution from MFU is shown in green and that from numerical uncertainty is shown in red.

The extended p-box can be interpreted as follows. Consider a requirement that the thrust produced by a rocket nozzle must be at least 2,600 N. After accounting for both MFU and numerical uncertainties, the probability that this requirement could be violated is in the interval range [0%, 22%]. That is, it could be as low as a 0% or as high as a 22% chance that the thrust will be below the required value. This interval range represents the lack of knowledge (i.e., ignorance) that the analyst has about the system, its inputs, and the modeling and simulation process. The prudent decision-maker would realize that the probability of violating the requirement could be as high as 22% and would look for ways the epistemic uncertainties could be reduced. It should be noted that if numerical uncertainty and MFU

were ignored, then the blue p-box of Fig 1 indicates that the thrust requirement will be met with nearly 100% probability.

1.4 Literature Review

For the implementation of ASME's Standard Validation uncertainty as referenced in Section III, Roache [10] provides a recommendation on how the true error and validation uncertainty should be applied for validation and calibration at locations where no data are available (predictions). It is proposed that for cases where E >> u_{val} that the simulation is calibrated by E, with an added uncertainty of U_{val} defined as

$$U_{val} = \sqrt{u_{val}^2 + u_{fit}^2}$$

where u_{val} is the validation uncertainty with an added 95% prediction interval and u_{fit} is the prediction interval on the interpolation of E between observation locations. For cases where E << u_{val}, it is recommended that E and u_{val} are used as a validation uncertainty about the original simulation result defined by

$$|\delta_{model}| \leq u_{val} + |E|$$

where δ_{model} is the discrepancy between the model and the true value in nature. For this case it is indicated that the true model discrepancy is most likely less than the sum of the total validation uncertainty and the absolute value of the true error. The reason this claim can be confidently made is because in situations where |E| << u_{val}, the true error is relatively small meaning that at prediction locations the model discrepancy will most likely be relatively small. Therefore the true value will most likely be easily encapsulated by the sum of the validation uncertainty and true error.

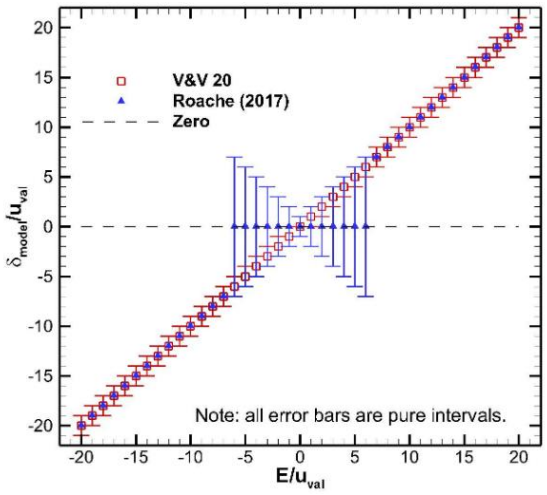
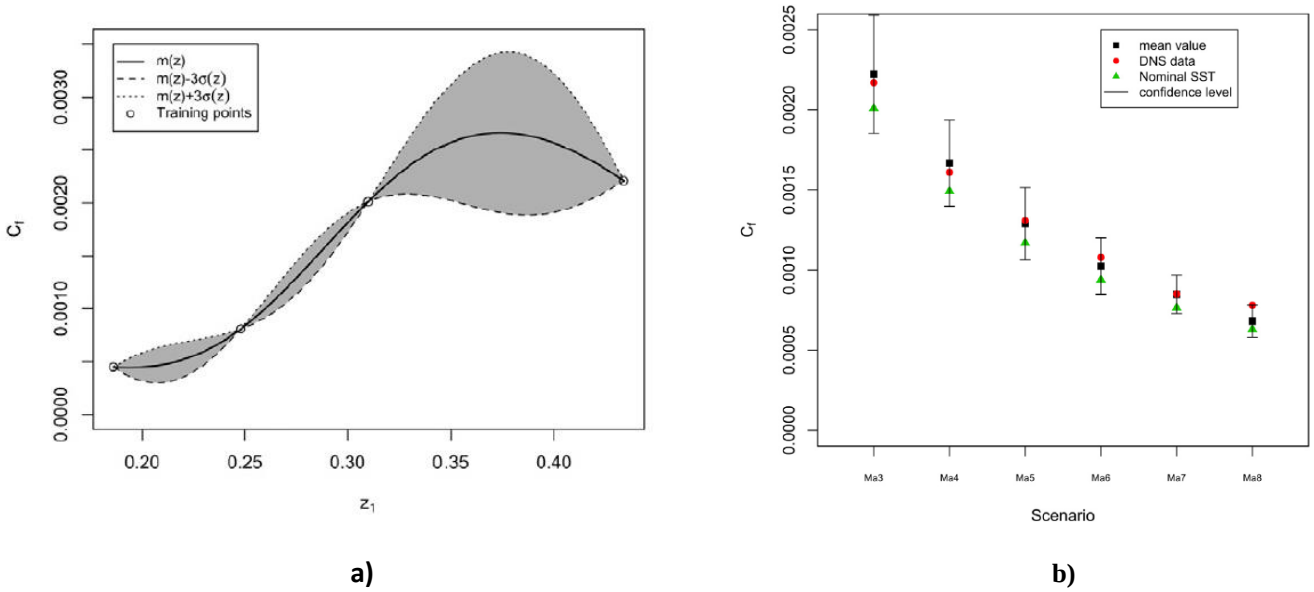


Fig 2 Demonstration of the application of V&V 20 as a validation or calibration method and its dependence on the ratio of difference in observed mean error and validation uncertainty.

There is plenty being done today with different validation and calibration approaches in determining how effective they are and comparing different methods with one another. Wu et. al [11] validates the component functions of model output between physical observation and computational model with the area metric. Results of several examples demonstrate the rationality of the area metric on validating computational results. Thacker et. al [12], using four different application problems (automotive crashworthiness, blast containment, spinal injury and space shuttle flow liner) provides an overview of some techniques of applying probabilistic methods to computationally expensive structural dynamics simulations. In Ling and Mahadevan [13], four types of validation methods are investigated, which are both classical and Bayesian hypothesis testing, a reliability-based method, and an area metric-based method. Two

types of validation experiments are considered for this assessment, fully characterized (all the model/experimental inputs are measured and reported as point values) and partially characterized (some of the model/experimental inputs are not measured or are reported as intervals). The area metric was shown to be sensitive to the direction of bias between model predictions and experimental data, as was the Bayesian interval hypothesis testing-based method. The Bayesian model validation result and reliability-based metric was found to be able to be directly incorporated in reliability analysis, thus explicitly accounting for model uncertainty.

The Bayesian model discrepancy method also mentioned in Section III has been used to compare turbulence models in computational fluid dynamics (CFD) with real world data. Bayesian updating was used for sensitivity analysis in determining the SST turbulence model parameters for a case of hypersonic flow over a flat plat for varying Mach numbers. Shown in Fig 2 is a) an updated model in predicting the skin friction coefficient across the flat plate, and b) different scenarios of SST turbulence model parameters in relation to the original skin friction coefficient predicted by the turbulence model, the real experimental value, and the mean value result from Bayesian updating and its associated uncertainty interval.



a) **b)**

Fig 3 a) An updated model in predicting the skin friction coefficient across the flat plate, and b) different scenarios of turbulence model parameters in relation to the original skin friction coefficient predicted by the turbulence model, the real experimental value, and the mean value result from Bayesian updating and its associated uncertainty interval (reproduced from [14]).

2. Model Validation and Calibration Methods

We will be assessing four different methods for model validation/calibration. Model validation is the process of determining the degree to which a model is an accurate representation of the *true value* in the real world, while calibration is the process of adjusting the computational model or its parameters to improve agreement with experimental data [1]. The two validation approaches that will be assessed are the area validation metric [9] and the modified area validation metric [15, 16], and the two validation/calibration approaches being examined are ASME's V&V 20 standard validation uncertainty [17] and a Bayesian model updating approach [18].

2.1 Area Validation Metric

The area validation metric [9] provides a method of estimating the model form uncertainty by placing uncertainty bounds about the simulation SRQs. After propagating the experimentally measured input uncertainty through the model, cumulative distribution functions (CDFs) are created for each SRQ. The magnitude of the model form uncertainty about these simulation results is estimated by determining the absolute value area between the experiment and simulation empirical CDFs, which will be referred to as $S_n(Y)$ and $F(Y)$ respectively. The model form uncertainty is therefore given as:

$$d = \int_{-\infty}^{\infty} |F(Y) - S_n(Y)| dY$$

where *d* is the area validation metric, and Y is the SRQ. Once *d* is determined, the interval in which the true value is estimated to lie is *[F(Y) – d, F(Y) + d]*. An example of the area validation metric applied for a case where only aleatory uncertainties are present in the model inputs is showed in Fig 4.

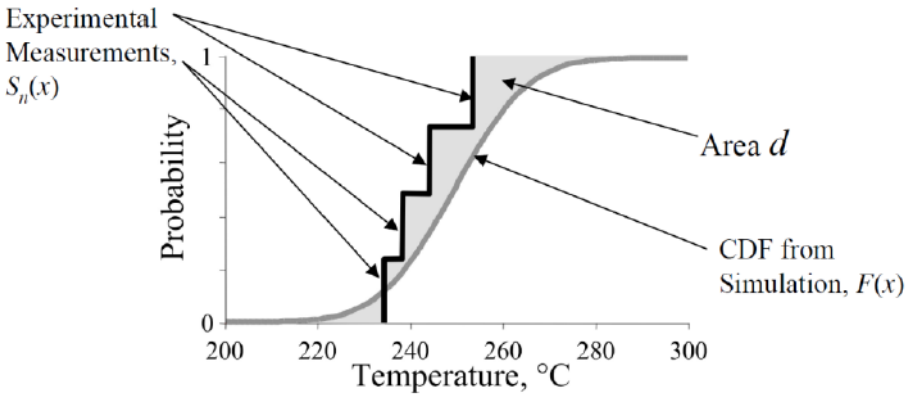


Fig 4. Area Validation Metric (reproduced from [9])

2.2 Modified Area Validation Metric

The area validation metric tends to produce an overly conservative result in relation to the true value due to the model uncertainty being applied symmetrically about the simulation, regardless of the relation between the experiment and simulation CDFs. In addition, the original AVM does not account for the additional uncertainty arising due to small experimental (or computational) sample sizes. In order to overcome these weaknesses, Voyles and Roy [15, 16] proposed the modified area validation metric (MAVM). Small sample sizes are accounted for by applying the 95% confidence interval for the mean about the entire experimental CDF. This approach is justified since, when the experimental and simulation CDFs do not overlap, the AVM simply defaults to the difference in the mean values of the distributions. Also note that small sample sizes in the simulation results could be accounted for similarly. Finally, the maximum area between the experimental confidence interval bounds and the simulation CDF are chosen because our goal is to estimate model form uncertainty. If instead we were seeking the "evidence for disagreement" between the simulation and experiment, the minimum area would be appropriate. The MAVM separately tracks two different areas between the CDFs. These two regions are the area where the right 95% confidence interval of the experimental CDF provides SRQs larger than the simulation CDF (d^+) and the area where the left 95% confidence interval of the experiment SRQs are smaller than the predicted SRQs from the model (d^-), as shown in Fig 5. The interval in which the metric estimates the true value most likely exists is $[F(Y) - d^-, F(Y) + d^+]$, where *F(Y)* is the simulation CDF.

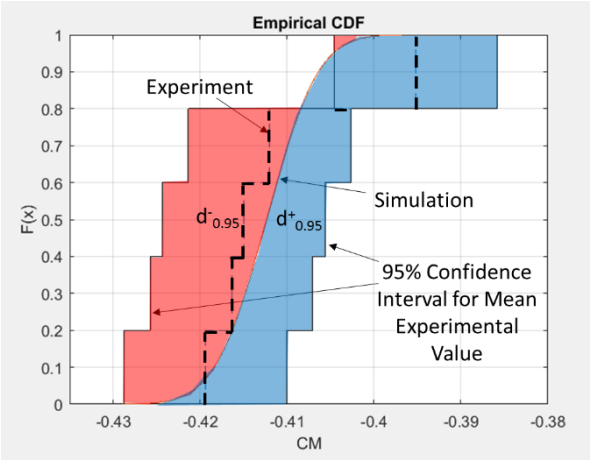


Fig 5. Modified Area Validation Metric with Confidence Intervals (reproduced from [15])

2.3 ASME V&V 20 Standard Validation Uncertainty

A validation standard was developed by ASME dealing with verification and validation of computational fluid dynamics and heat transfer [17], referred to herein as V&V 20. The implementation of V&V 20 involves the computation of two parameters. The first is the error E between the expected values of the simulation S and the experiment D (here taken to be the mean values).

$$E = S - D.$$

The second is the standard validation uncertainty, u_{val}, which is given by:

$$u_{val} = \sqrt{u_{num}^2 + u_{input}^2 + u_D^2}$$

where u_{num} is the numerical uncertainty in the model SRQs, u_{input} is the effect of propagating the input uncertainties through the simulation to the SRQs, and u_D is the uncertainty in the experimental data. When applying this metric to estimate where the true value exists, the true model error is given as:

$$\delta_{model} = [E - ku_{val}, E + ku_{val}]$$

where k is a coverage factor. Although not addressed directly in the standard, for cases where $|E| >> ku_{val}$, δ_{model} can just be taken to be E and the model results can possibly be updated as such. When $|E| \leq ku_{val}$, then the model cannot be corrected and the model uncertainty is expected to be less than or equal to ku_{val}. Since u_{val} represents a standard uncertainty (approximately 68% confidence for a normal distribution), a coverage factor must be included to achieve other confidence levels. For 95% confidence when the distribution is not known, the V&V 20 standard recommends coverage factors between 2 and 3. Here we will simply use a coverage factor of 2. When $E \leq ku_{val}$, the true values, with 95% confidence, are expected fall within the interval $S \pm ku_{val}$, where $k = 2$. The uncertainty in the experimental results is found by computing confidence intervals about the experimental mean using the Student's t-distribution:

$$u_D = 1.96\sqrt{\frac{s^2}{N-1}}$$

where s^2 is the variance of the experimental measurements and N is the number of samples. This results in the uncertainty estimate of the mean experimental value being quite small when many experiment replicates are available.

2.4 Bayesian Model Calibration

The final validation/calibration approach that will be assessed is provided by Kennedy and O'Hagan [18**]**. This method is a combined validation, calibration, and prediction approach that uses Gaussian Processes (GPs) to produce an updated model that is a better representation of the "true value" in nature. By obtaining experimental results at a range of input locations for the model, the observed discrepancy between the experiment and the simulation at these locations can be determined. These discrepancies can then be used to update a model discrepancy term to better represent the experimental data, i.e., Bayesian updating. Gaussian Processes are then used to connect the observations to one another over the input domain to create the updated model. The updated model function is defined as:

$$f(x) \sim GP(m(x), k(x, x'))$$

where *f(x)* is a posterior realization of the GP, *m(x)* is the mean posterior function selected to represent the model discrepancy (*m(x) = 0* for this case), and *k(x,x')* is the covariance matrix of the input quantities with one another. The way that the GP behaves between observation locations is dependent upon this covariance matrix which is given by

$$k(x, x') = \sigma_n^2 \exp\left(-\frac{|x - x'|}{2l^2}\right)$$

where σ_n^2 is the variance in the observation discrepancies and l is the correlation length scale, which denotes the typical distance over which the function values at different spatial locations become decorrelated. The length scale is determined here by using Maximum Likelihood Estimation [6] by selecting l such that the probability that the GP posterior correctly predicts the behavior of the true model is the highest. This is done by finding one such that the following function is at a maximum:

$$\ln(p(y_o|x_o, \sigma_n, l)) = -\frac{1}{2} y_o^T K_o y_o - \frac{1}{2}\ln(\det(K_o)) - \frac{n_{obs}}{2}\ln(2\pi)$$

where y_o are the observed model discrepancies, x_o are the locations of the input domain where observations were made, n_{obs} are the number of observations made, and K_o is the covariance matrix of the observation locations with itself ($K_o = k(x_o,x_o)$). The joint distribution of the observed target values and the function values at the test locations under the prior is then given as [19]:

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathcal{N}(0, \begin{bmatrix} K(X,X) + \sigma_n^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix})$$

where f_* is a single Gaussian Process realization, and it has a conditional (posterior) distribution of:

$$f_*|X, y, X_* \sim \mathcal{N}\left(\bar{f}_*, cov(f_*)\right), where$$
$$\bar{f}_* \triangleq E[f_*|X, y, X_o] = K(X_*,X)[K(X,X) + \sigma_n^2 I]^{-1} y,$$
$$cov(f_*) = K(X_*,X_*) - K(X_*,X)[K(X,X) + \sigma_n^2 I]^{-1} K(X,X_*)$$

where X are domain locations where experimental data is available and X_* are locations where model discrepancy is to be predicted. Fig 6 shows an example of posteriors created assuming a mean of zero and a large variance, and then the updated posterior after incorporating the observation data to reflect in the model discrepancy.

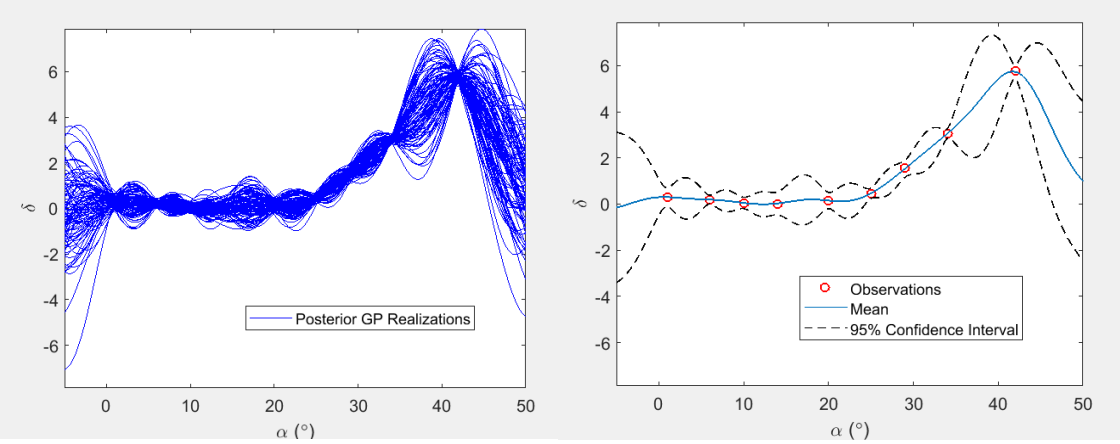


Fig 6. Display of updated posterior model using observational data

3. Method of Manufactured Universes and Validation Case

Since our goal is not to actually validate or calibrate a model, but instead to assess the usefulness of different model validation, calibration, and prediction frameworks, we will not use actual experimental data, which would leave ambiguity regarding the true value in nature. Instead we will use the Method of Manufactured Universes (MMU) developed by Stripling et al. [20] to unambiguously assess the different model validation, calibration, and prediction techniques. Similar to the method of manufactured solutions used for code verification, MMU involves the generation of a manufactured reality, or "universe," from which "experimental" observations can be made. A low-fidelity model is then developed with uncertain inputs and possibly including numerical solution errors. Since the true behavior of "reality" is known from the manufactured universe, the estimation of model form uncertainty and errors in the lower-fidelity model can be performed. It is suggested that the manufactured reality be based on a high-fidelity model so as to obtain similar qualitative behavior as that found in a real experiment. This approach can be used to compare different methods for estimating model form uncertainty and calibration in the presence of random experimental measurement error, experimental bias errors, modeling errors, and uncertainties in the model inputs. It can also be used to assess approaches for extrapolating model form uncertainty to conditions where no experimental data are available (i.e., model prediction locations).

Due to the cost of obtaining "real world" experimental data, a high fidelity CFD model was used as the "experiment". The experimental model was created using the OVERFLOW solver version 2.2n [22], featuring the 2-D flow around the MD 30P/30N multi-element airfoil [10]. The 2-D lift and moment coefficients were solved for seven different angles-of-attack, α = 0º, 5º, 10º, 15º, 20º, 25º, 30º, over five different flap deflection angles, δ = 0º, 10º, 20º, 30º, 40º. From these results, a manufactured universe was created using bi-harmonic curve fits to serve as the sampling space for obtaining "synthetic" experimental data. The low fidelity model used here is Thin Airfoil Theory [23]. The formulas given by Thin Airfoil Theory are:

$$c_l = \pi(2A_o + A_1)$$

$$c_{m(\frac{c}{4})} = \frac{\pi}{4}(A_2 - A_1)$$

$$\frac{x}{c} = \frac{1}{2}(1 + \cos\theta)$$

$$A_o = \alpha - \frac{\pi}{4}\int_0^{\pi} \frac{dz}{dx} d\theta$$

$$A_n = \frac{2}{\pi}\int_0^{\pi} \frac{dz}{dx}\cos(n\theta)\, d\theta \qquad \text{for n = 1, 2, 3…}$$

where c_l is the 2-D lift coefficient, $c_{m(c/4)}$ is the 2-D moment coefficient about the quarter chord, α the angle of attack,

and *dz/dx* is the derivative of the mean camber line with respect to *x*. Fig 8 show the lift and moment coefficient provided from the manufactured universe with the red x's representing the locations at which high-fidelity CFD results were provided. The lift and moment coefficients provided from Thin Airfoil Theory are shown in Fig 9. Note the linearity of the results for both coefficients. In this validation case, an uncertainty of 0.5° and 1.0° will be propagated through angle of attack and flap deflection, respectively. Each uncertain input will be treated as a normal distribution with a mean of the intended measurement and a standard deviation of its associated uncertainty.

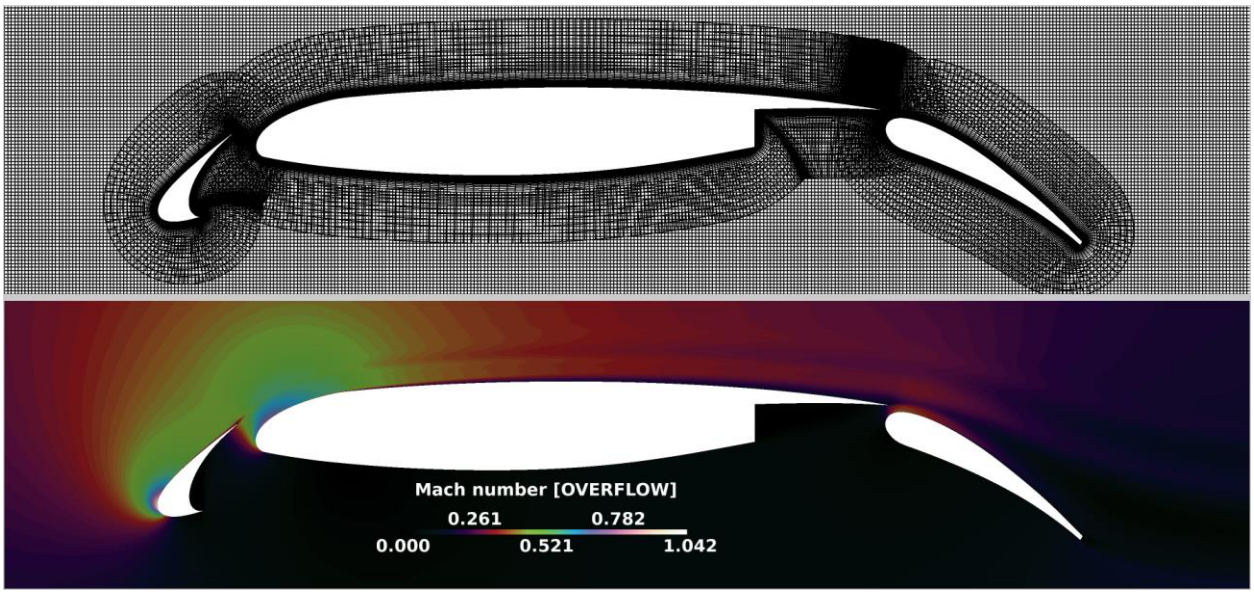
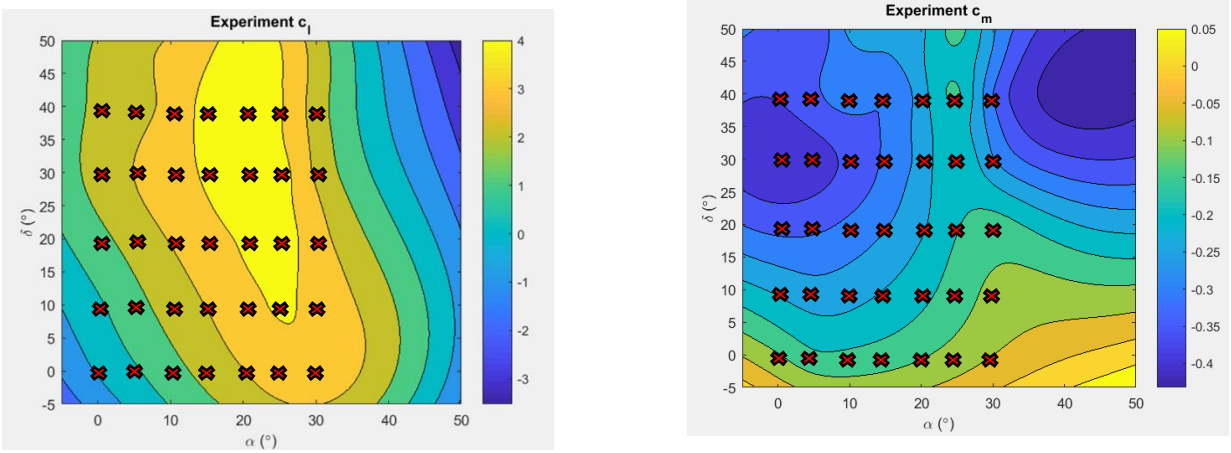
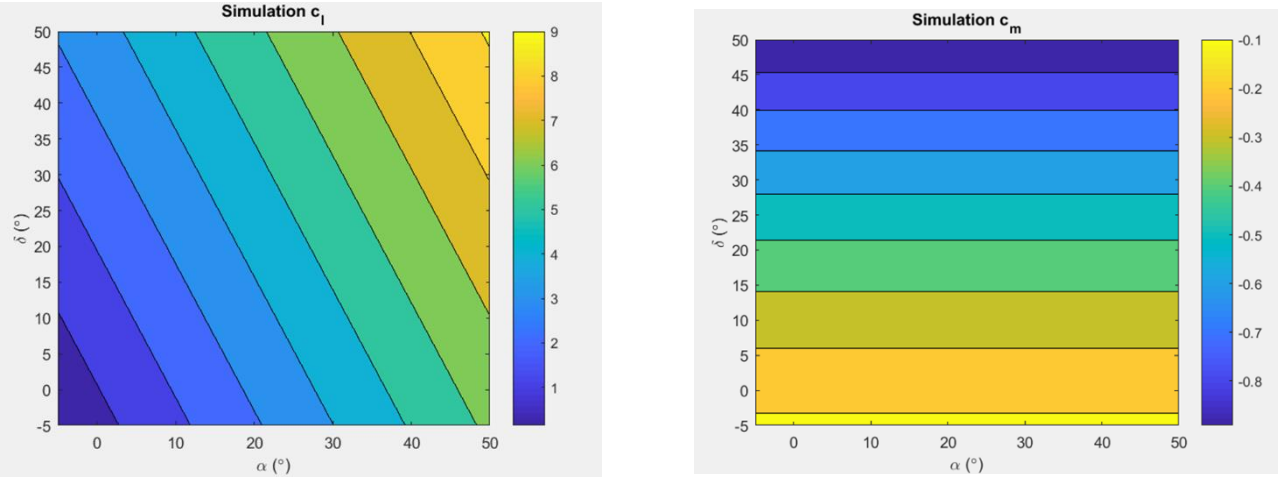


Fig 7. Overset grid system and typical flowfield result shown as a coordinate slice colored by local Mach number for the MD 30P/30N airfoil.



a) **b)**

Fig 8. a) c_l and b) $c_{mc/4}$ for Manufactured Universe (i.e. synthetic experimental data)



a) **b)**

Fig 9. a) c_l and b) $c_{mc/4}$ for Low Fidelity Model

4. Performance Metrics

In order to compare these validation, calibration, and prediction methods, it was observed how well they were able to capture the true value. To compare these four different validation and calibration methods, two metrics were used to assess them. The first is *conservativeness, φ_1*, which is the percentage of the time that a respective method encapsulates the true value in nature within its associated uncertainty bounds. In this case, the "true value" is taken as the mean of the experimental samples as the number of experimental samples approaches infinity (N = 10,000 in this case). The second factor taken into consideration is *tightness, φ_2*, which assesses how tightly the uncertainty interval about either the predicted or calibrated value bounds the true value. It should be noted that φ_2 is only calculated if a respective method is proven to be conservative in a particular instance, and it is measured as:

$$\varphi_{2,v} = \frac{|True\ Error|}{Uncertainty}$$

where $\varphi_{2,v}$ is the tightness assessment for validation methods and is simply the ratio of the true model error magnitude to the associated uncertainty about the mean simulation result. The reason this tightness calculation only applies to validation methods is due to the fact that it could penalize calibration methods that have an updated simulation mean close to the true value. Therefore, the following formula will be used to measure tightness for calibration methods:

$$\varphi_{2,c} = 1 - \frac{Uncertainty}{|True\ Error\ (orignial)|}$$

where $\varphi_{2,c}$ is more dependent on the measure of the ratio of the associated uncertainty with the calibration to the original true error magnitude that existed prior to calibration. For both validation and calibration cases, if a method fails to be conservative, then φ_2 will be taken to be zero for that case. These two assessment factors are then combined into an overall assessment by:

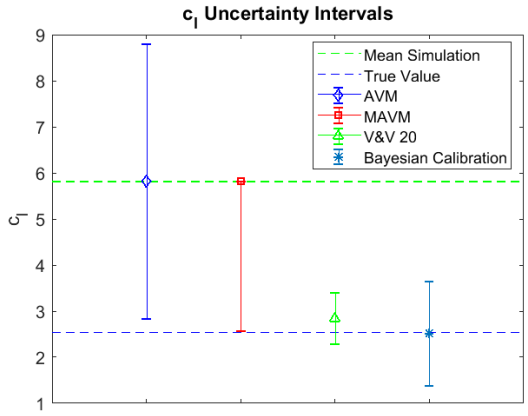
$$\varphi = \alpha_w\varphi_1 + (1 - \alpha_w)\varphi_2$$

where φ is the overall assessment and α_w is a weighting factor, typically set to be 0.5 (i.e. equal weighting) for low consequence applications such as preliminary design or 0.9 for higher consequence applications in order to place a greater significance on conservativeness.

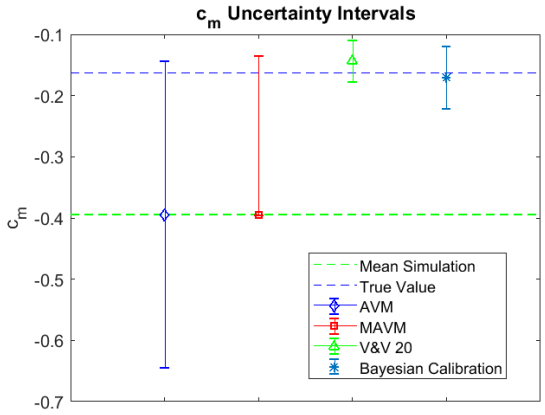
5. Results

5.1 Performance where Experimental Observations are Available

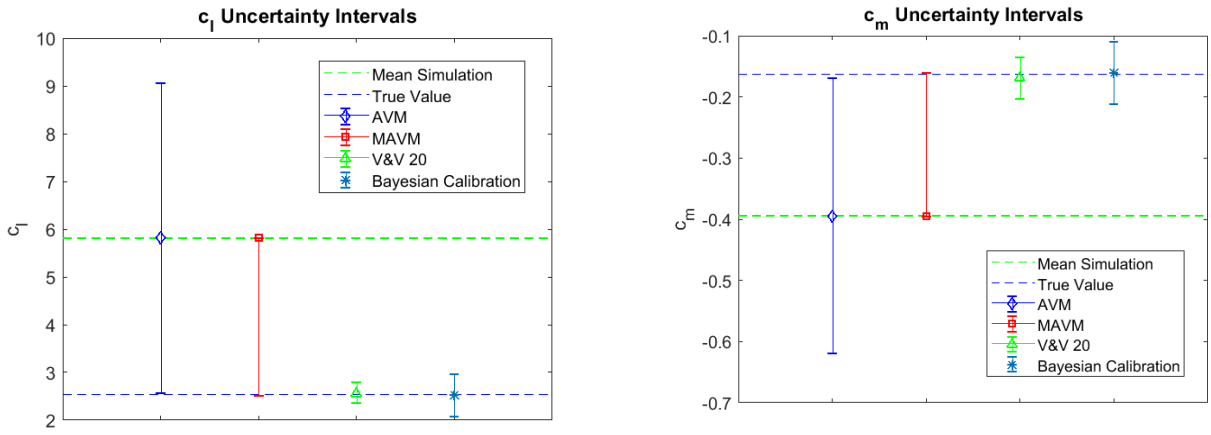
Each of the four methods previously discussed were first assessed at locations of the input domain where synthetic experimental results were provided (i.e., the red x's in Fig 8) and compared with results from the model. Uncertainty bounds about each of the results are shown in Fig 10 for both 2 and 16 experimental samples for the observation point of $\alpha = 34°$ and $\delta = 21°$. It is seen that as the number of experimental samples increases, the uncertainty bounds about the true value generally become smaller. In Fig 11, the conservativeness of each method aggregated over all of the observation points as a function of sample size is shown. As would be assumed, with the increase in experimental samples available the conservativeness of each method generally increased. However, this was not the case for area validation metric. This is because the area validation metric has no included confidence interval such as the modified area validation metric. Therefore, whether the area validation metric is conservative or not largely depends on the value of the mean experimental SRQs relative to the true value (i.e., it will be larger than the true value approximately 50% of the time). The same trend is also seen for the tightness measurement, shown in Fig 12, as the uncertainty bounds for each validation metric and calibration method becomes tighter around the true value, excluding the area validation metric. Fig 13 shows the overall assessment of the four methods for locations where experimental data are provided, with α_w taken to be 0.5. Based off of these results, it is shown that the modified area validation metric, V&V 20, and Bayesian calibration perform well even for low experimental sample sizes. The high overall assessment of the calibration methods is probably expected, however, due to the methods having experimental data to help obtain an idea of where the true value lies. The added confidence intervals about the experimental data when computing the MAVM showed an obvious improvement from the AVM in its tightness about the true value and its overall assessment.



a)

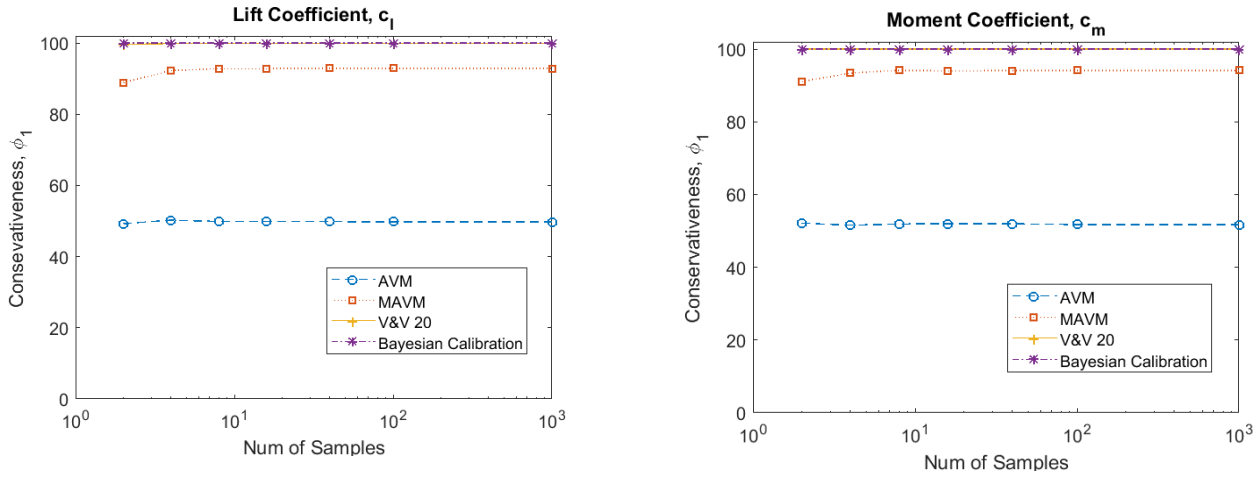


b)



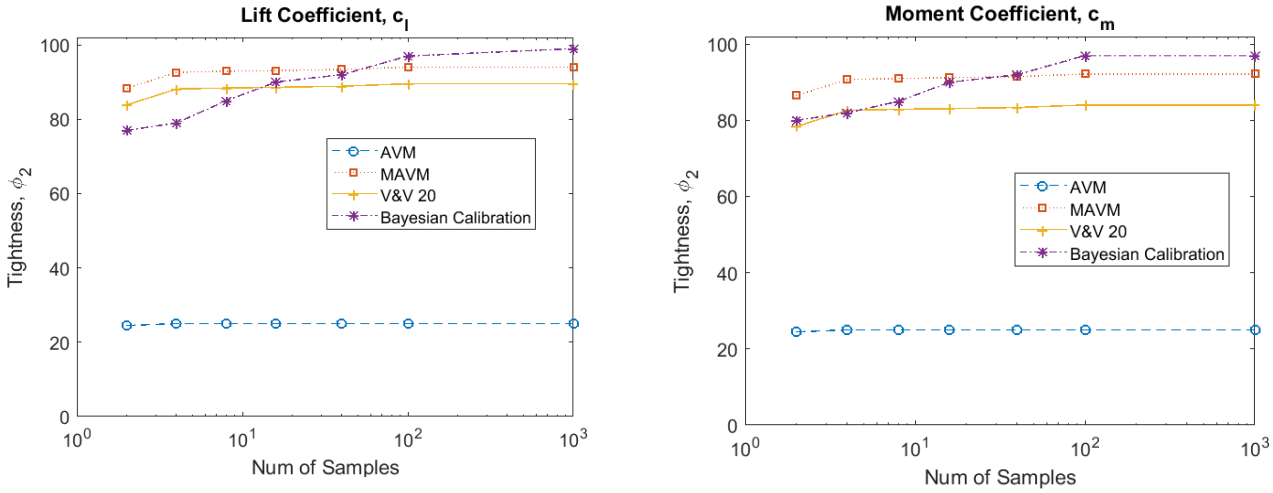
c) **d)**

Fig 10. a) Uncertainty Intervals for c_l (2 Experimental Samples), b) Uncertainty Intervals for $c_{mc/4}$ (2 Experimental Samples), c) Uncertainty Intervals for c_l (16 Experimental Samples), d) Uncertainty Intervals for $c_{mc/4}$ (16 Experimental Samples) at $\alpha = 34°$ and $\delta = 21°$



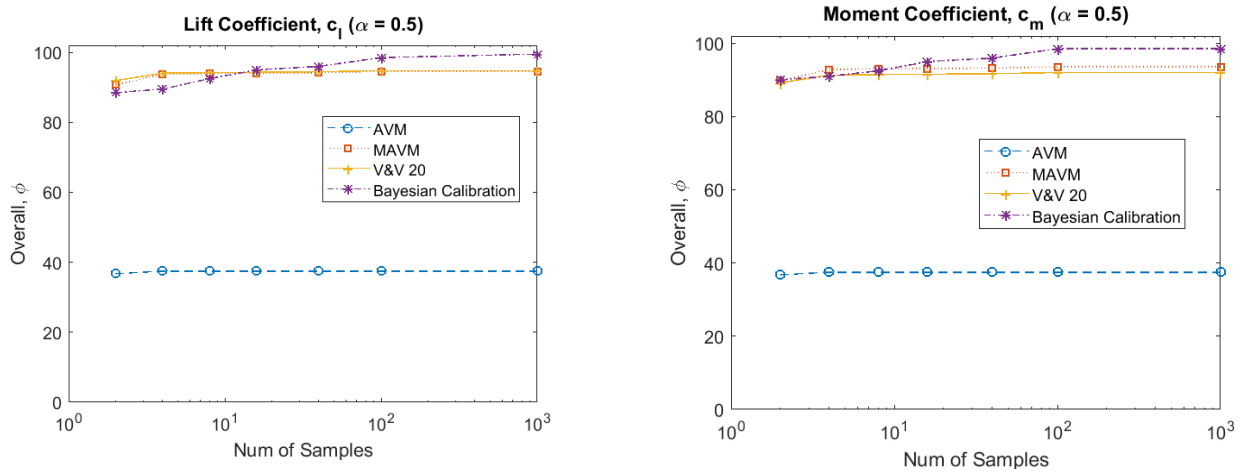
a) **b)**

Fig 11. Conservativeness as a function of sample size for each validation metric at locations were observation data are available for a) c_l and b) $c_{mc/4}$

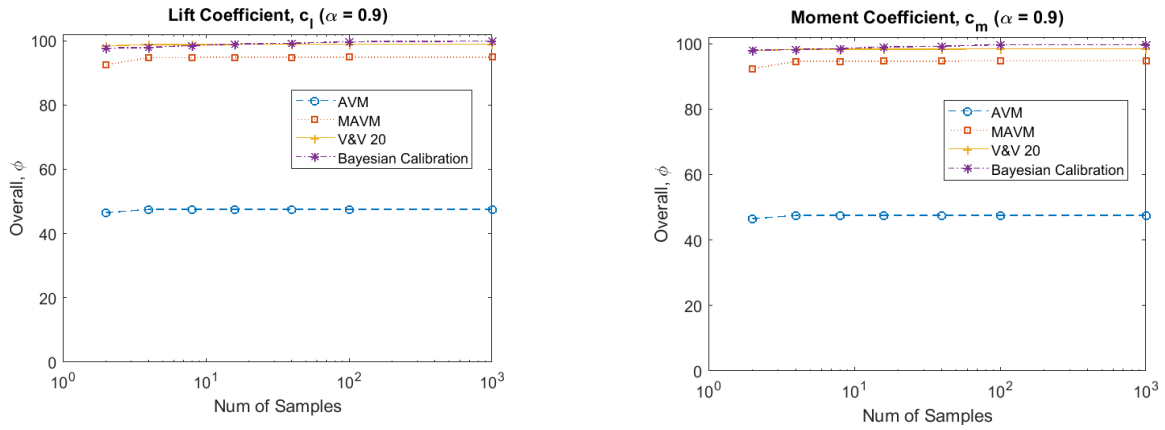


a) **b)**

Fig 12. Tightness measurements as a function of sample size for locations at which observation data are available for a) c_l and b) $c_{mc/4}$



a) **b)**



c) **d)**

Fig 13. Overall assessment as a function of sample size for locations at which observation data are available for a) c_l and b) $c_{mc/4}$ (assumes α_w = 0.5) and c) c_l and d) $c_{mc/4}$ (assumes α_w = 0.9)

5.2 Prediction for the Validation/Calibration Methods

The second part of comparing these methods was to observe how they performed at locations where no experimental results were provided, and the results had to be interpolated or extrapolated to the prediction locations. This was done by creating a best fit of the metric results at the locations where experimental SRQs were provided. Outlined in Roy and Oberkampf [1], 95% prediction intervals were then placed about the fit, and the upper limit of the prediction interval was taken as the metric result. Therefore, in this instance the upper level of the 95% prediction intervals were taken for d, d^+, d^-, E, and u_{val} then used as the uncertainty for their respective method. Note that for V&V 20, the model was calibrated by E and the associated uncertainty was the 95% predication interval for E plus the upper limit on the prediction interval for ku_{val} [10]. However, this did not have to be done for the Bayesian model updating as predictions are made directly from sampling the Gaussian Process model. A simple 1-D example of this interpolation is shown in Fig 14 for the extrapolation of the area validation metric using a quadratic regression fit. The uncertainties or model errors/discrepancies are extrapolated to twenty prediction locations located at angles of attack of 0°, 18°, 25°, 38°, and 45° and flap deflections of 0°, 8°, 17°, and 25°. For the case in which a sparse amount of experimental observations are available, a linear regression fit was used in both the dimension of angle of attack and flap deflection due to the limited number of available points. However, as the number of observations increased as for the moderate and plentiful cases, a higher order regression fit could be used and a quadratic fit was used for those cases.

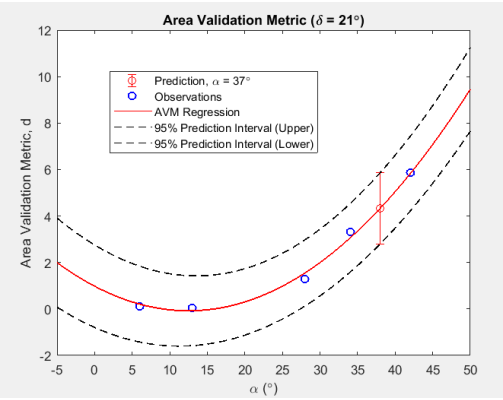


Fig 14. Prediction Regression for c_l Using Area Validation Metric

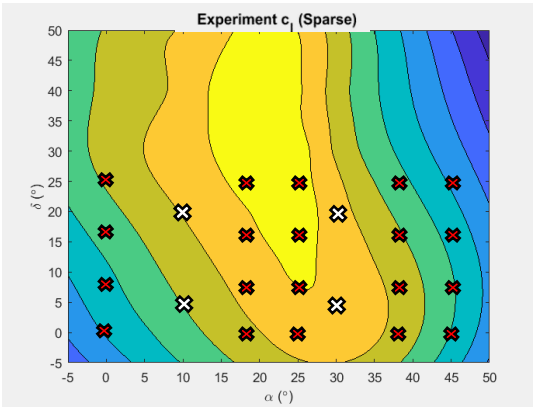
5.2.1 *Sparse Experimental Observations*

Four observations were assumed at angles of attack 10° and 30° and flap deflections of 5° and 20°. These locations are shown in Fig 15 with the observation locations being shown in white and prediction locations in red. The uncertainty intervals for each method are shown in Fig 16 for samples sizes of 2 and 16 at $\alpha = 18°$ and $\delta = 25°$. The locations where experimental data is available is defined as our validation domain, and the locations at which predictions are being made are the application domain. In this case the data is sparse and there is a significant amount of extrapolation that is required from the validation domain. It is seen how the modified area validation metric accounts for bias error by its one sided interval unlike the area validation metric's symmetric uncertainty interval. The uncertainty intervals on the modified area validation metric can also be seen closing in about the true value with the increase in sample size from 2 to 16 due to the reduced confidence intervals about the experimental mean. Additionally it is seen that the area validation metric and the modified area validation metric are overly conservative for lift coefficient compared to V&V 20 and Bayesian updating. This is because at this prediction location the true value and mean value from the simulation are relatively close to one another, and with the added prediction interval for those methods it makes them overly conservative at this location while having a tighter bound about the true value at prediction locations further away from the validation domain.

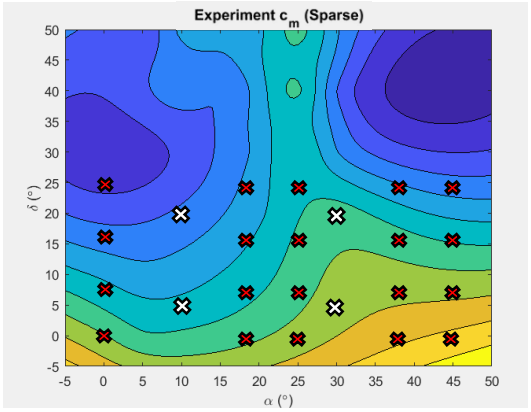
When examining the conservativeness of each method shown in Fig 17, it is seen that the area validation metric and the modified area validation metric are shown to be the most conservative of the four methods when a small amount of observations are available and there is significant extrapolation from the validation domain to the prediction domain. However, it is interesting to point out the different levels of conservativeness that are seen between lift coefficient and moment coefficient for the area validation metric and the modified area validation metric. This comparison shows the effectiveness of the extrapolation of these methods in being conservative is reliant upon where observations are made to collect data in their assessment. Because in this case only two observations are made in each respective dimension of the uncertain inputs, the information from these two methods can only be extrapolated via a linear regression fit. Due to the observations being made in locations where the experiment still behaves linearly for lift coefficient with respect to angle of attack and flap deflection, the linear fit closely encapsulates the true value for small angles of attack and flap deflection. However, when extrapolating out to regions of high angles of attack and flap deflections (where flow separation occurs and the lift coefficient becomes nonlinear), the methods tend to be not conservative. This is not the case for moment coefficient due to the experiment containing a lot of nonlinearity across the input domain, therefore the prediction intervals on the extrapolation regression fits are more easily to encapsulate the true value outside of the validation domain. It is also important to note the poor performance of Bayesian updating when a low amount of data are available. This is because Bayesian updating assumes a mean prior mean discrepancy of zero, therefore when making predictions for the mean discrepancy with few observations it is hard to acquire meaningful information outside of where those observations are made.

Examining the tightness in Fig 18, the modified area validation metric is shown to be the tightest in this case. Also it is shown that the area validation metric was about half as tight as the modified area validation metric as expected due to the modified area validation metric's separate tracking of d^+ and d^- areas to help account for bias error. The Bayesian updating method was seen to be the least conservative in part due to the fact that the method assumes a large uncertainty about the calibration outside of the domain where observations were made, therefore leading to a low tightness about the true value where conservative. However, it should be noted that all approaches have issues with tightness due to the large amount of extrapolation from the validation domain.

The overall combined assessment of the four methods shown in Fig 19. The modified area validation metric performed the best in the case where conservativeness and tightness were equally weighted ($\alpha_w = 0.5$). This is the case because of its ability to be nearly twice as tight as the area validation metric when conservative, and to still be as conservative as V&V 20 in the case of lift coefficient despite in its difficulty to be conservative in some locations outside of the validation domain. The same also holds true when increasing the weight on conservativeness to $\alpha_w = 0.9$ with the modified area validation metric still having the overall best performance.

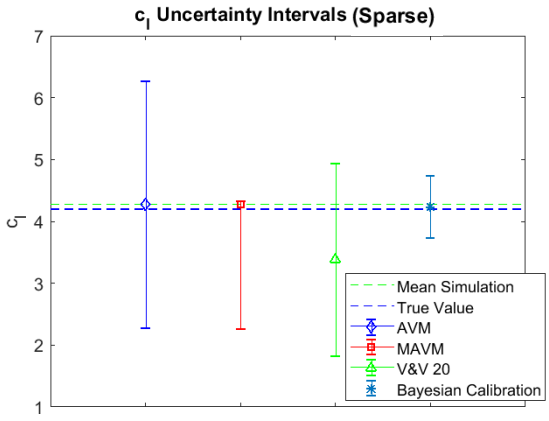


a)

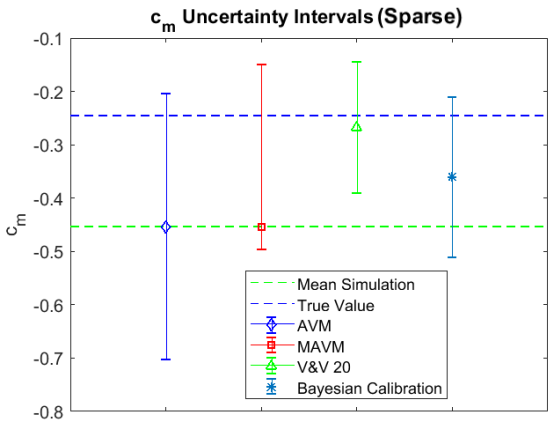


b)

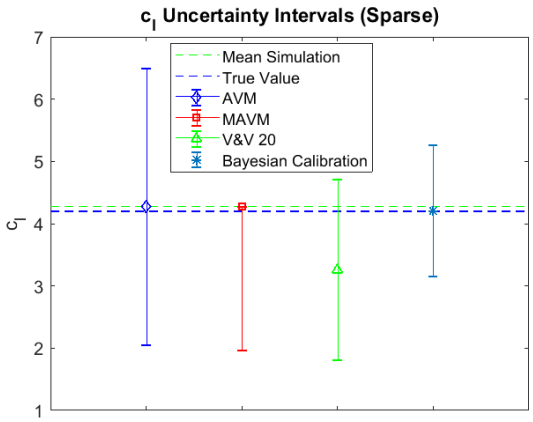
Fig 15. Prediction locations where validation/calibration methods are being assessed for a) c_l and b) $c_{mc/4}$. (White x's: Observations/Validation Domain, Red x's: Predictions/Application Domain)



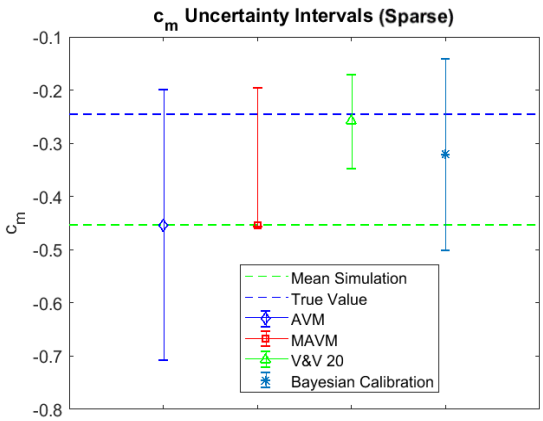
a)



b)

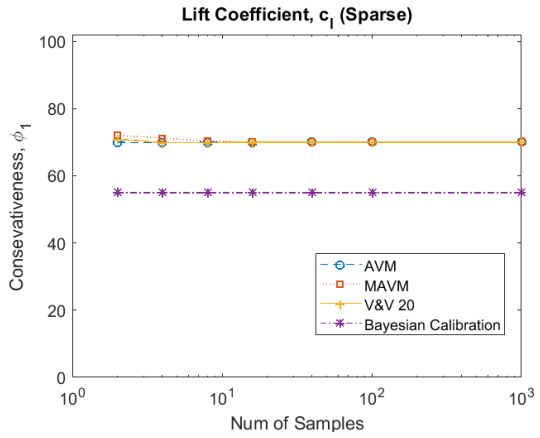


c)

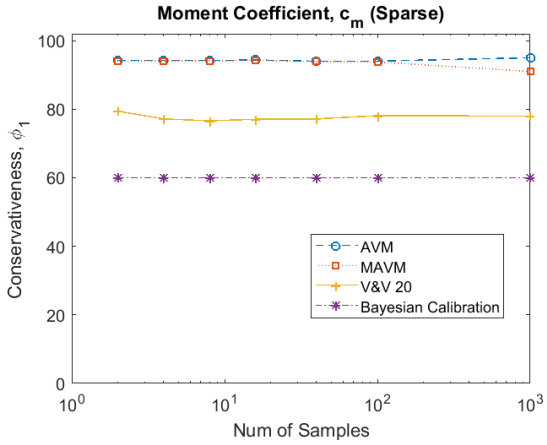


d)

Fig 16. Prediction Location Uncertainty Intervals a) Uncertainty Intervals for c_l (2 Experimental Samples), b) Uncertainty Intervals for $c_{mc/4}$ (2 Experimental Samples), c) Uncertainty Intervals for c_l (16 Experimental Samples), d) Uncertainty Intervals for $c_{mc/4}$ (16 Experimental Samples) at $\alpha = 18°$ and $\delta = 25°$.

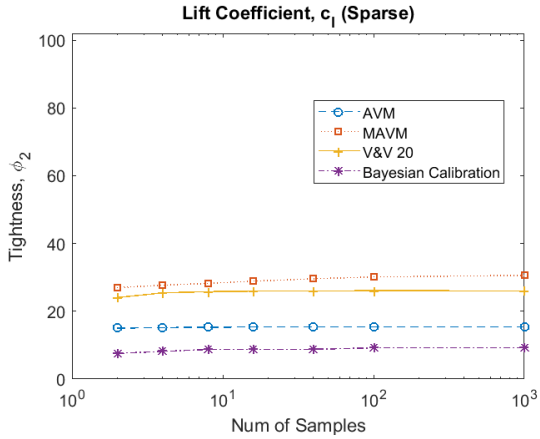


a)

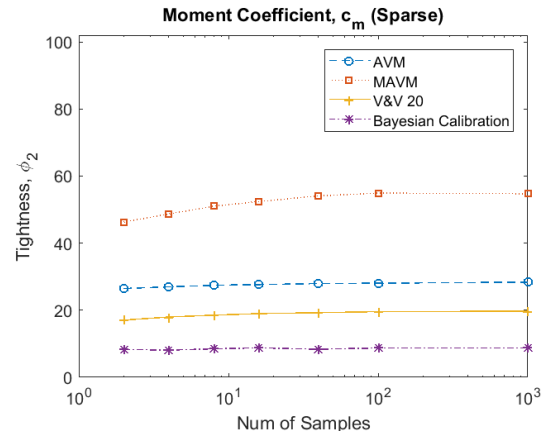


b)

Fig 17. Conservativeness at prediction locations for each method for a) c_l and b) $c_{mc/4}$

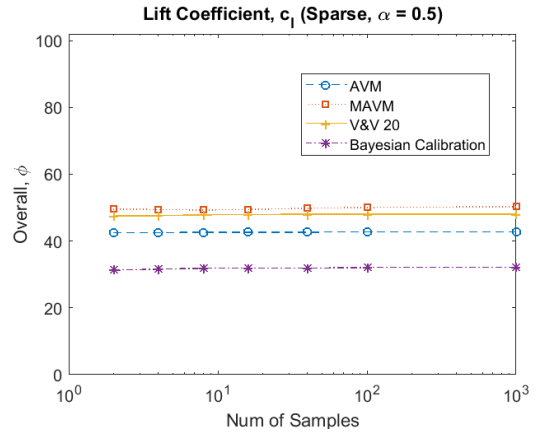


a)

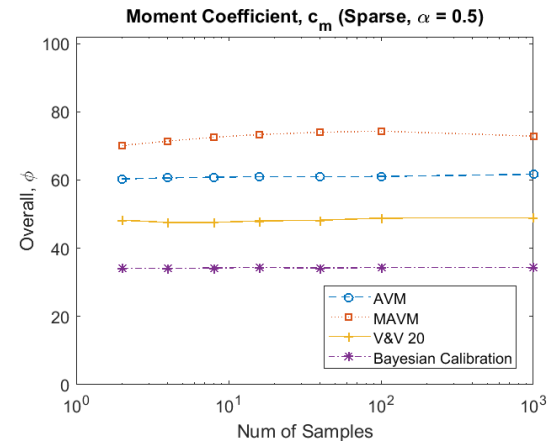


b)

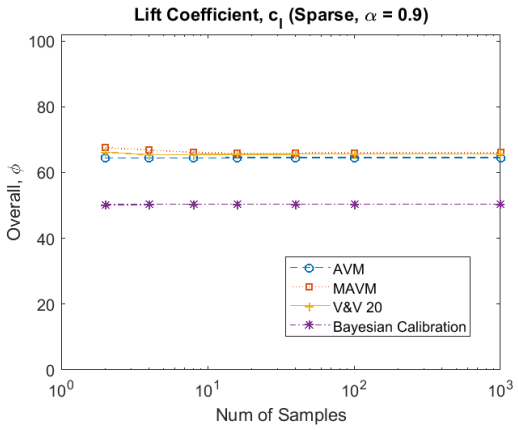
Fig 18. Tightness at prediction locations for each method for a) c_l and b) $c_{mc/4}$



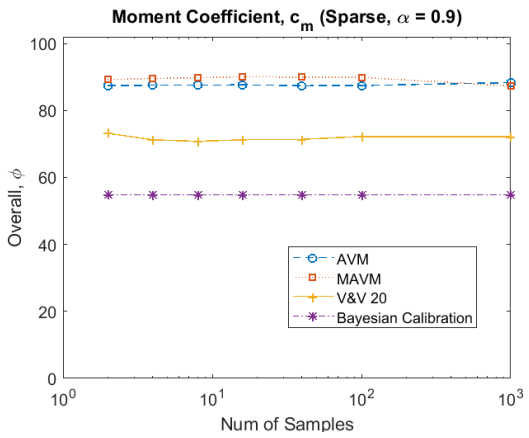
a)



b)



c)



d)

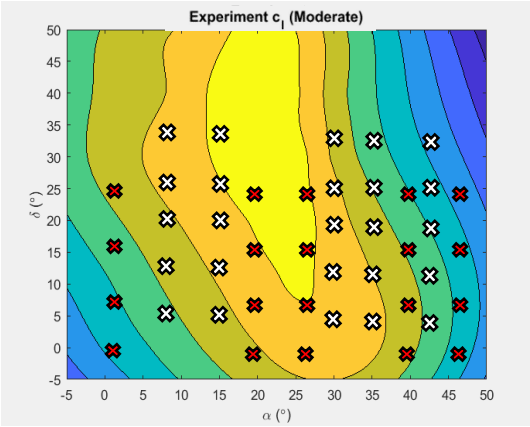
Fig 19. Overall assessment at predictions locations using α_w = 0.5 for a) c_l and b) $c_{mc/4}$ and α_w = 0.9 for c) c_l and d) $c_{mc/4}$

5.2.2 Moderate Experimental Observations

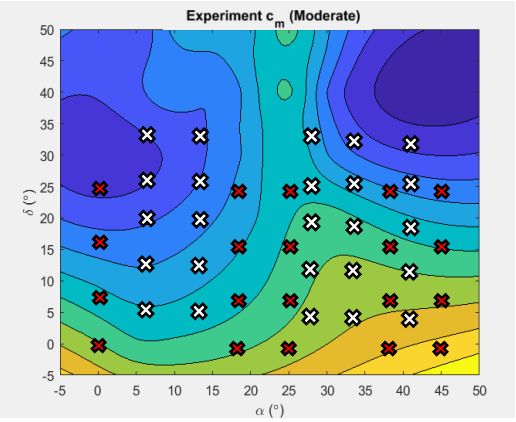
Twenty-five observation locations were used in this case, and they were located at angles of attack of 6°, 13°, 28°, 34°, and 42° and flap deflections of 5°, 13°, 21°, 27°, and 35° as shown in Fig 20. Most of the prediction locations involve interpolation, but a few at the higher flap deflection angles involve some mild extrapolation. Upon the interpolation/extrapolation of the four methods, it was seen as shown in Fig 21 that for AVM and MAVM their respective uncertainty intervals at prediction locations were larger than those at observation locations, making them slightly more conservative with their predictive capability. However for the calibration methods of V&V 20 and Bayesian model updating this was not the case. The conservativeness of these two methods when making predictions is largely dictated by the amount of observation locations available across the domain of the input space. Since these methods rely on observation data to create an updated model, the more experimental observations available would provide a better prediction about the true value. Fig 22 shows the conservativeness of each as function of sample size at the prediction locations. It is seen that Bayesian updating was usually conservative for all sample sizes at prediction locations, while the 95% prediction interval added to the AVM, MAVM, and V&V 20 interpolation also maintained a high conservativeness for all sample sizes.

When examining the tightness of these methods for prediction locations as displayed in Fig 23, it is shown that MAVM is the tightest. It is important to note the difference in tightness however for MAVM between lift coefficient and moment coefficient. This is because, as further investigation for the prediction intervals showed, some of the predictions have lift coefficient values similar to the experiment. Therefore the prediction interval greatly over predict the true value. However, the moment coefficient values from experiment contain more variability across the input domain, having the prediction locations closer the inner bound of the prediction intervals. The Bayesian updating method, which is a function of a prior chosen initial variance and correlation length scale, was found to be not very tight due to the large confidence intervals associated with the method at locations where no observational data is provided.

Assessing these methods using equal weighting of conservativeness and tightness (α_w = 0.5), MAVM is seen to perform slightly better. However, as α_w is increased placing a greater weight on conservativeness, Bayesian updating has the best overall performance, followed closely by the other three methods.

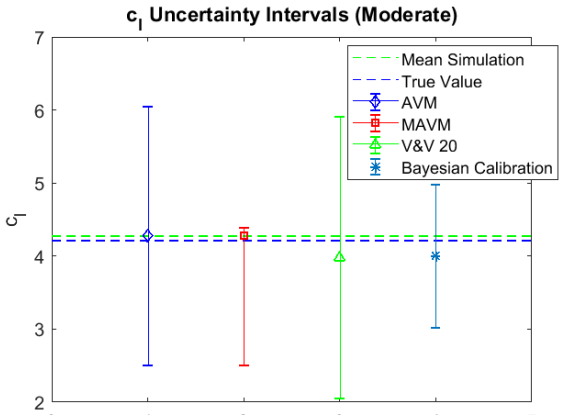


a)

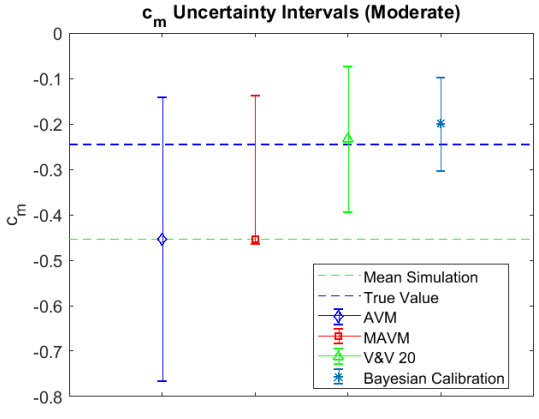


b)

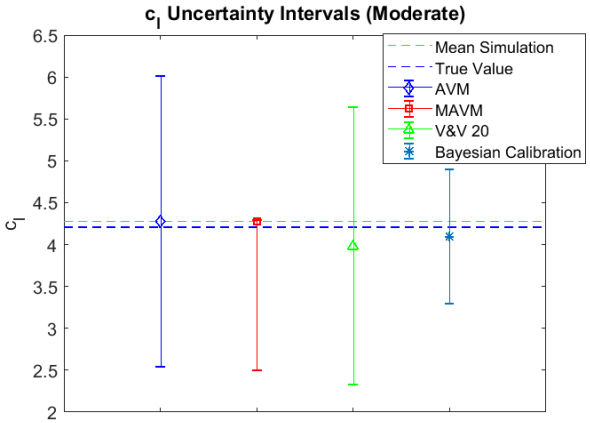
Fig 20. Prediction locations where validation/calibration methods are being assessed for a) c_l and b) $c_{mc/4}$. (White x's: Observations/Validation Domain, Red x's: Predictions/Application Domain)



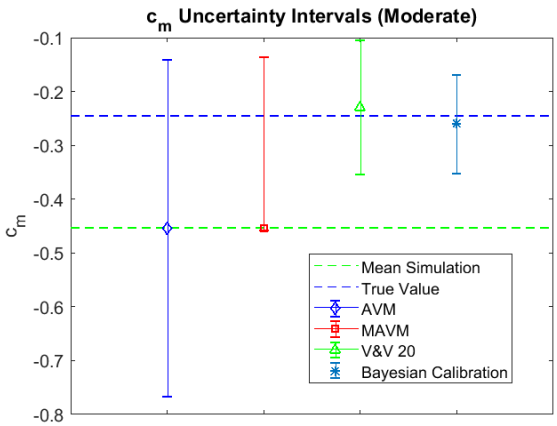
a)



b)

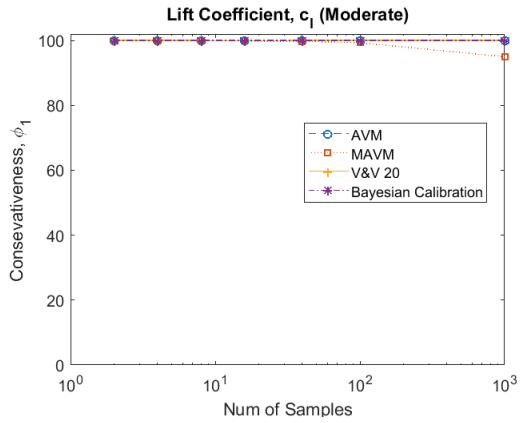


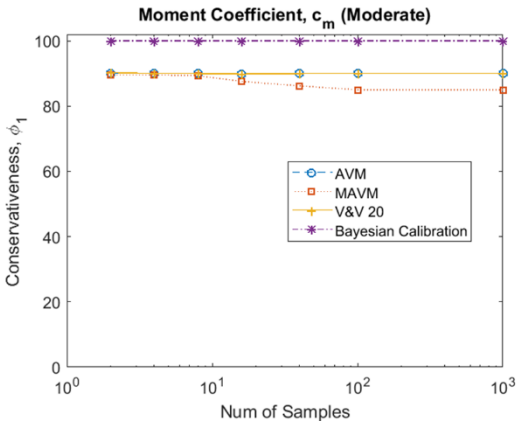
c)



d)

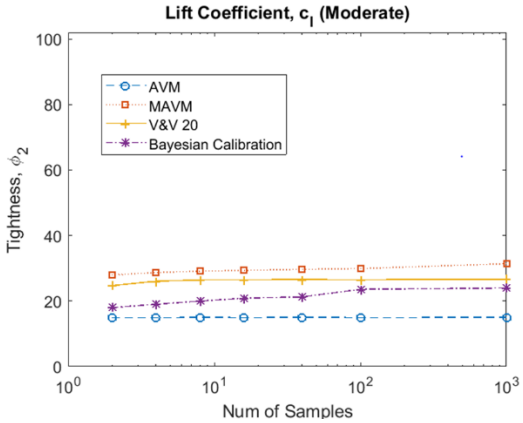
Fig 21. Prediction Location Uncertainty Intervals a) Uncertainty Intervals for c_l (2 Experimental Samples), b) Uncertainty Intervals for c_m (2 Experimental Samples), c) Uncertainty Intervals for c_l (16 Experimental Samples), d) Uncertainty Intervals for $c_{mc/4}$ (16 Experimental Samples) at $\alpha = 18°$ and $\delta = 25°$

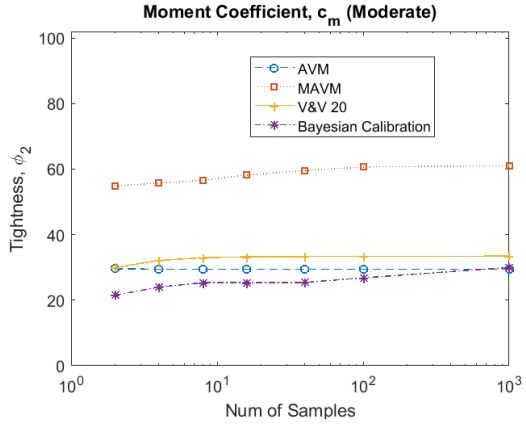




a) **b)**

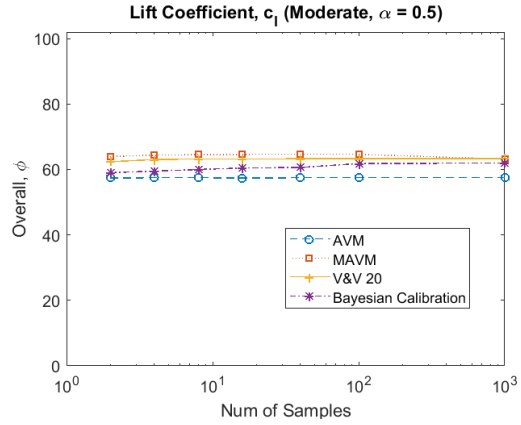
Fig 22. Conservativeness at prediction locations for each method for a) c_l and b) $c_{mc/4}$

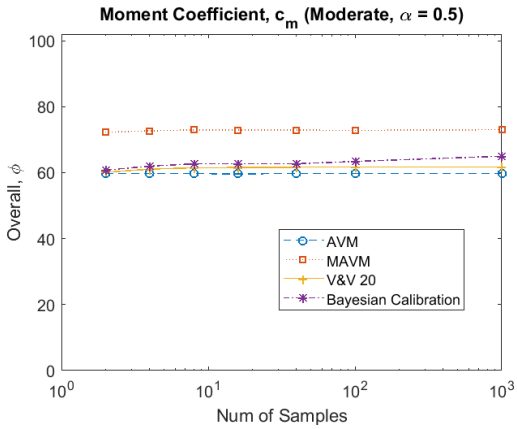




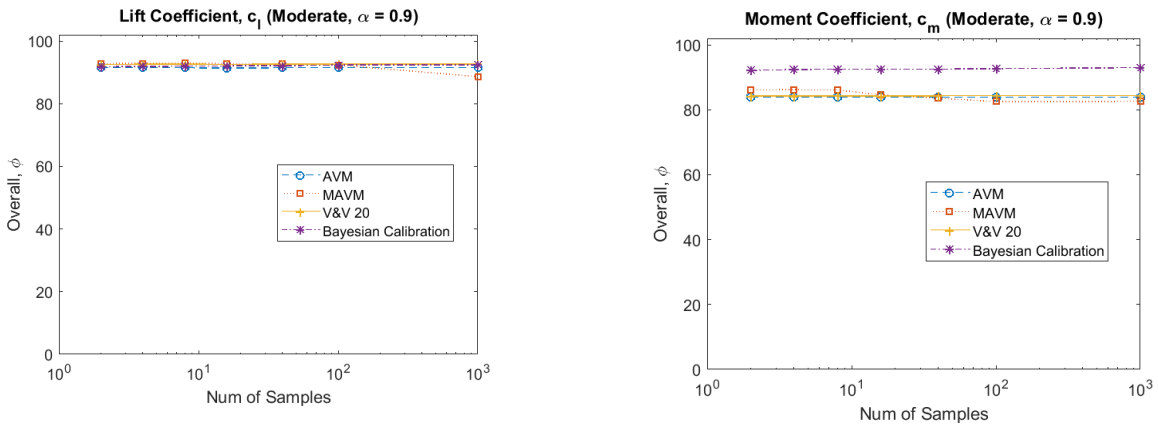
a) **b)**

Fig 23. Tightness at prediction locations for each method for a) c_l and b) $c_{mc/4}$





a) **b)**



c) **d)**

Fig 24. Overall assessment at predictions locations using α_w = 0.5 for a) c_l and b) $c_{mc/4}$ and α_w = 0.9 for c) c_l and d) $c_{mc/4}$

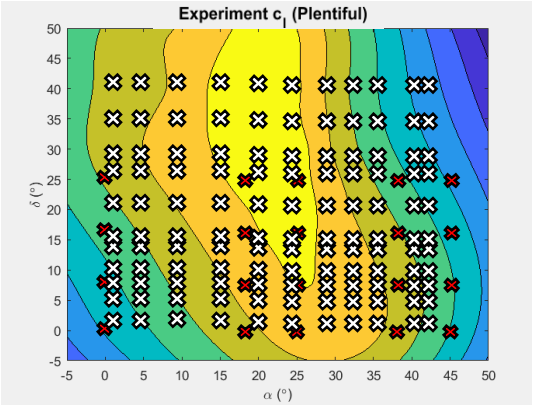
5.2.3 Plentiful Experimental Observations

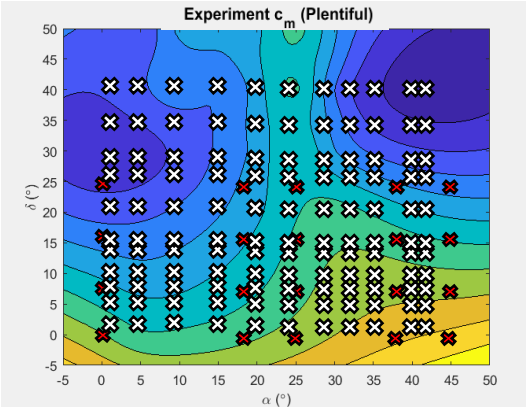
One hundred observations were made at angles of attack 1°,4°,9°,15°,19°,24°,28°,32°,35°,40°, and 41° and flap deflections of -1°, 2°, 5°, 7°, 10°, 13°, 15°, 21°, 26°, 29°, 35°, and 41°, and the uncertainties or model errors/discrepancies are extrapolated to the same twenty prediction locations used in the previous cases. These locations are shown in Fig 25 with the observation locations being shown in white and prediction locations in red. For this case, there are many experimental observation locations and the application domain is almost entirely within the validation domain (i.e. there is very little extrapolation). The uncertainty intervals for each method are shown in Fig 26 for samples sizes of 2 and 16. It is seen that even with the large amount of experimental data, the uncertainty intervals for each respective method are still relatively large due to the added prediction interval for the interpolation/extrapolation and the confidence interval for the Gaussian Process for Bayesian calibration.

In this case, each of the methods was found to be very conservative as shown in Fig 27. The Bayesian calibration method is conservative in nearly every prediction case due to the methods ability to actively update the model for a better approximation in relation to the experiment. The other three methods are also nearly conservative in every instance with the exception of one prediction location for the moment coefficient. This is due to the interpolations being non-conservative at one prediction location slightly outside the domain where observations were made. In measurement of tightness as shown in

Fig 28, Bayesian updating and the modified area validation metric were shown to be the tightest.

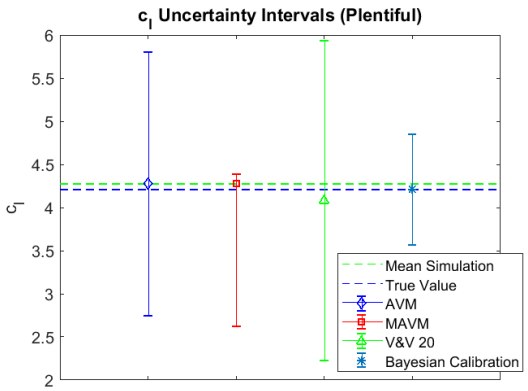
When looking at the overall assessment of the methods in Fig 29, the modified area validation metric and Bayesian calibration perform the best in preliminary design cases where conservativeness and tightness are equally weighted (α_w = 0.5). However, when the assessment weight factor in increased to α_w = 0.9, it is clear that Bayesian calibration, and to a lesser extent, the modified area validation metric are slightly better than the other two approaches in higher consequence scenarios where a higher conservativeness is preferred and plentiful data are available over the entire application domain.

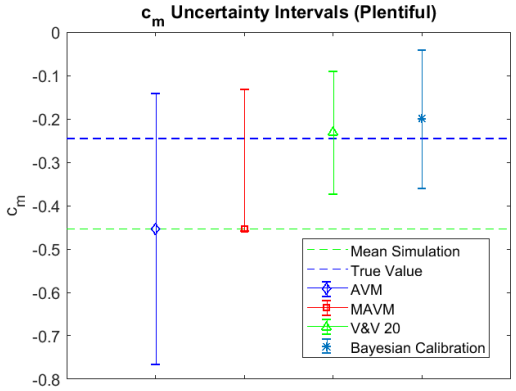




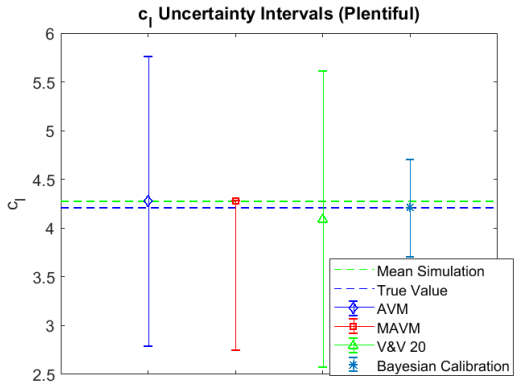
a) **b)**

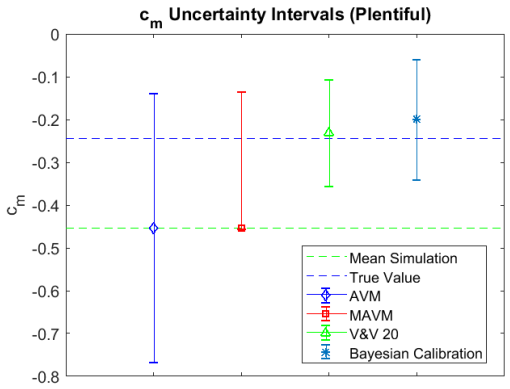
Fig 25. Prediction locations where validation/calibration methods are being assessed for a) c_l and b) $c_{mc/4}$. (White x's: Observations/Validation Domain, Red x's: Predictions/Application Domain)





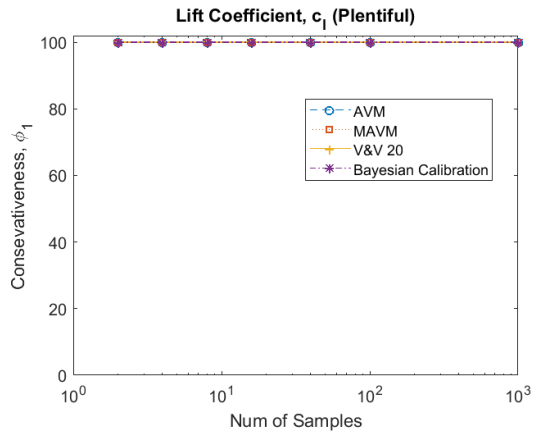
a) **b)**

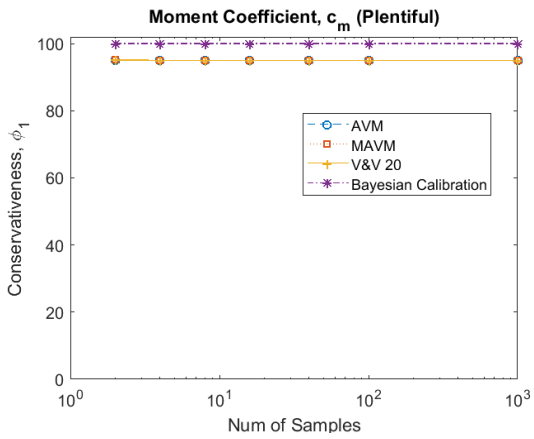




c) **d)**

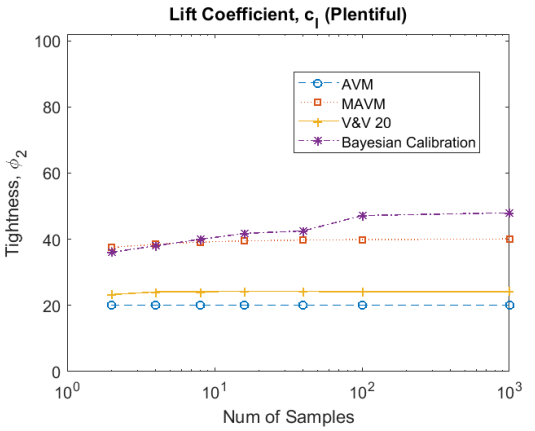
Fig 26. Prediction Location Uncertainty Intervals a) Uncertainty Intervals for c_l (2 Experimental Samples), b) Uncertainty Intervals for $c_{mc/4}$ (2 Experimental Samples), c) Uncertainty Intervals for c_l (16 Experimental Samples), d) Uncertainty Intervals for $c_{mc/4}$ (16 Experimental Samples) at $\alpha = 18°$ and $\delta = 25°$

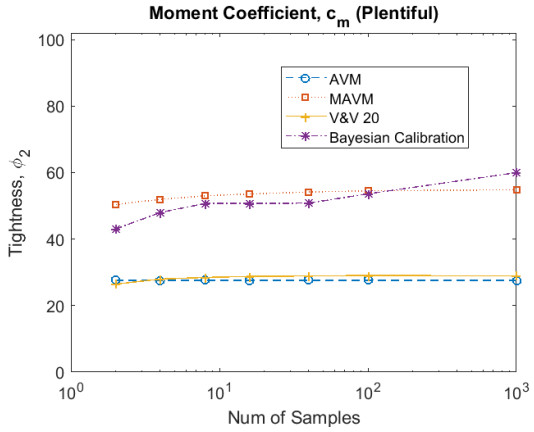




a) **b)**

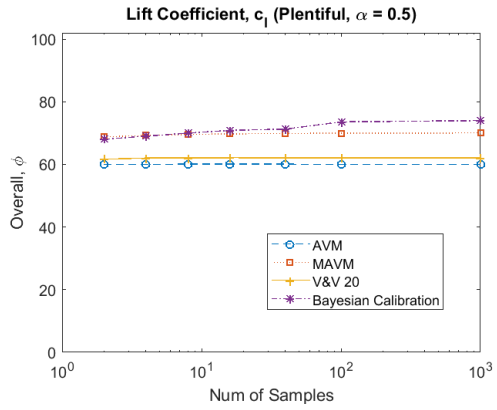
Fig 27. Conservativeness at prediction locations for each method for a) c_l and b) $c_{mc/4}$

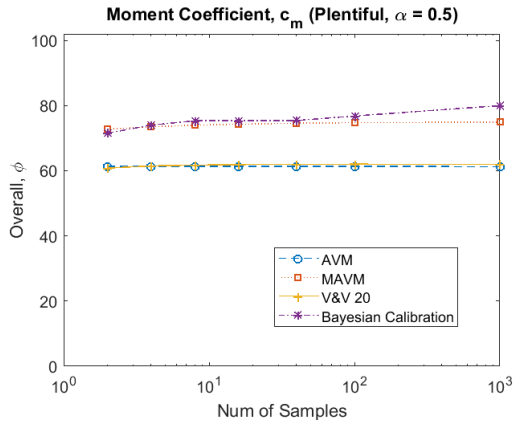




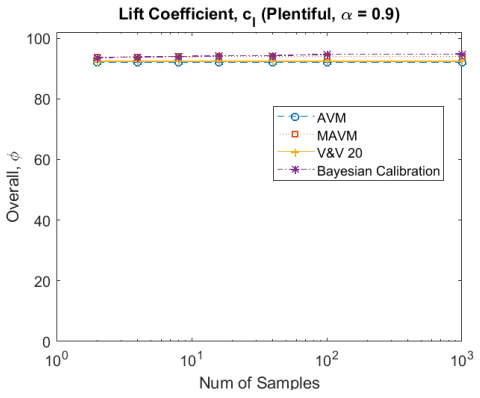
a) **b)**

Fig 28. Tightness at prediction locations for each method for a) c_l and b) $c_{mc/4}$

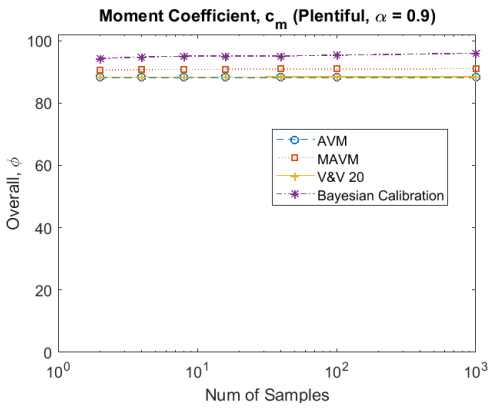




a) **b)**



c)



d)

Fig 29. Overall assessment at predictions locations using α_w = 0.5 for a) c_l and b) $c_{mc/4}$ and α_w = 0.9 for c) c_l and d) $c_{mc/4}$

6. Conclusions

Upon examining and comparing these four validation/calibration methods, it was observed that the area validation metric was shown to be the least conservative of the four methods for validation where data is available. The other three methods were generally conservative at locations with data. When examining their predictive capability, the modified area validation metric and Bayesian updating appeared to perform the best depending on the amount of observation data available when considering both conservativeness and tightness in the overall assessment. However, as more of a weight is placed on conservativeness, as it would be for high consequence applications, Bayesian updating performs better than the modified area validation metric for moderate amounts of data. For plentiful data, Bayesian calibration and the modified area validation metric slightly outperformed the other two methods. With more observation points, calibration is more attractive, but with limited data simply estimating the model form uncertainty (with no calibration) is recommended. These findings are summarized in Table 1. This table shows the recommended approach given the amount of experimental data or interpolation/extrapolation that is required from the validation domain to the application domain and also takes into account the level of risk one is willing to assume.

Table 1. Validation/Calibration recommendation as a factor of experimental data and decision risk.

	Decision Risk		
Amount of Experimental Data	Low (Preliminary Design)	Medium (Typical Analysis)	High (High Consequence)
Sparse/ Extensive Extrapolation	MFU only (MAVM)	MFU only (MAVM)	MFU only (MAVM)
Moderate/ Some Extrapolation	MFU only (MAVM)	MFU only (MAVM)	Calibration + MFU (K & O)
Plentiful/ Interpolation Only	Mainly Calibration (K & O)	Calibration + MFU (K & O)	Calibration + MFU (K & O)

MFU: Model Form Uncertainty

MAVM: Modified Area Validation Metric (Voyles and Roy, 2014)

V&V 20: ASME's Standard Validation Uncertainty (ASME, 2009)

K&O: Bayesian Calibration (Kennedy and O'Hagan, 2001)

References

1. Roy, C. J. and Oberkampf, W. L. (2011), "A Comprehensive Framework for Verification, Validation, and Uncertainty Quantification in Scientific Computing," *Computer Methods in Applied Mechanics and Engineering*, Vol. 200, pp. 2131–2144.

2. Montgomery, D. C., *Design and Analysis of Experiments*, 9th Ed., John Wiley & Sons, New Jersey, 2017.

3. Oberkampf, W. L. and Smith, B. L., "Assessment Criteria for Computational Fluid Dynamics Model Validation Experiments," *Journal of Verification, Validation, and Uncertainty Quantification*, Vol. 2, No. 3, 2017.

4. ASME PTC 19.1-2005, *Test Uncertainty*, 2005.

5. ISO Guide to the Expression of Uncertainty in Measurement, ISO, Geneva, Switzerland, 1995.

6. Coleman, H. W. and Steele, W. G., *Experimentation, Validation, and Uncertainty Analysis for Engineers*, 3rd Ed., Wiley and Sons, New York, 2009.

7. Oberkampf, W. L. and Roy, C. J., *Verification and Validation in Scientific Computing*,Cambridge University Press, New York, 2010.

8. Roy, C. J. and Balch, M. S. (2012), "A Holistic Approach to Uncertainty Quantification with Application to Supersonic Nozzle Thrust," *International Journal for Uncertainty Quantification*, Vol. 2, No. 4, pp. 363-381.

9. Ferson, S., Oberkampf, W. L., and Ginzburg, L. (2008), "Model Validation and Predictive Capability for the Thermal Challenge Problem," *Computer Methods in Applied Mechanics and Engineering*, Vol. 197, pp. 2408–2430.

10. Roache, P. J., "Interpretation of Validation Results Following ASME V&V20-2009," *Journal of Verification, Validation, and Uncertainty Quantification*, Vol. 2, 2017.

11. Wu, D., Lu, Z., Wang, Y., Cheng, L., Model validation and calibration based on component functions of model output, Reliability Engineering & System Safety, Volume 140, 2015, pp. 59-70, ISSN 0951-8320

12. Thacker, B. H., Riha, D. S., Nicolella, D. P., Hudak, S. J., Huyse, L. J., and Francis, L., "Uncertainty Quantification for Structural Dynamics and Model Validation Problems", Southwest Research Institute, San Antonio, TX

13. Ling, Y. & Mahadevan, S., (2012). Quantitative model validation techniques: New insights. Reliability Engineering & System Safety. 111. 10.1016/j.ress.2012.11.011.

14. Zhang, J. & Song, F., (2019). "An efficient approach for quantifying parameter uncertainty in the SST turbulence model." *Computers & Fluids.* 181. 10.1016/j.compfluid. 2019.01.017.

15. Voyles, I. T. and Roy, C. J., "Evaluation of Model Validation Techniques in the Presence of Uncertainty", AIAA Science and Technology Conference, Jan 2014, National Harbor, MD.

16. Voyles, I. T. and Roy, C. J., "Evaluation of Model Validation Techniques in the Presence of Aleatory and Epistemic Input Uncertainties", AIAA Science and Technology Conference, Jan 2015, Kissimmee, FL.

17. ASME V&V 20, *Standard for Verification and Validation in Computational Fluid Dynamics and Heat Transfer*, American Society of Mechanical Engineers, ASME Standard V&V 20-2009, New York, NY.

18. Kennedy, M. C. and O'Hagan, A., *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* Vol. 63, No. 3 (2001), pp. 425-464

19. Rasmussen, C. E. and Williams, C. K. I. (2006), *"Gaussian Processes for Machine Learning"*, the MIT Press, pp. 14-16

20. Stripling, H. F., Adams, M. L., McClarren, R. G., and Mallick, B. K., "The Method of Manufactured Universes for Validating Uncertainty Quantification Methods," *Reliability Engineering and System Safety*, Vol. 96, 2011, pp. 1242-1256.

21. Buning, P. G., Gomez, R. J., Scallion, W. I., "CFD Approaches for Simulation of Wing-Body Stage Separation", AIAA Paper 2004-4838, August 2004.

22. Morrison, J. H., "Numerical Study of Turbulence Model Predictions for the MD 30P/30N and NHLP-2D Three-Element Highlift Configurations.", NASA/CR-1998-208967, NAS 1.26:208967 (1998).

23. Anderson, J. D., *Fundamentals of Aerodynamics*, 5th Ed., McGraw-Hill, 2011.

24. Whiting, N. W., Roy, C. J., Duque, E., and Lawrence, S., "Assessment of Model Validation and Calibration Approaches in the Presence of Uncertainty," AIAA Paper 2019-1829, AIAA SciTech, January 2019.

Appendix: MATLAB Codes

1. Area Validation Metric

```
% Function created by Nolan Whiting on September 25, 2017 to calculate the
% area validation metric between two CDFs to estimate the model form
% uncertainty about a simulation.

% Inputs:   Exp: Vector of experimental samples
%           Sim: Vector of simulation samples
% Outputs:  diff: Area Validation Metric

function [diff] = AreaValidationMetric(Exp,Sim)

% Checks to see if largest experimental sample is smaller than smallest
% simulation sample or vice versa. If it is, then area validation metric is
% calculated to be the difference between the mean of the experimental
% valus and the mean of the simulation results.
if Exp(end) <= Sim(1) || Sim(end) <= Exp(1)
    diff = abs((mean(Exp) - mean(Sim)));

% If there is intersection between the simulation and experiment CDFs, then
% the area between each individual sample and the simulation CDF will be
% calucalated and summed to provided the area validation metric.
else
    sum_area = 0;
    prob_CFD = linspace(0,1,length(Exp)+1);
    prob_CFD = prob_CFD(2:end);
    prob_TAT = linspace(0,1,length(Sim)+1);
    prob_TAT = prob_TAT(2:end);
    count = 1;
    rem_area = 0;

    if length(Sim) < length(Exp)
        for i = 1:length(Sim)
            if rem_area ~= 0
                next_area = abs((Exp(count) - Sim(i))*(prob_CFD(count)-prob_TAT(i-1)));
                sum_area = sum_area + next_area;
                count = count + 1;
            end
            while prob_TAT(i) > prob_CFD(count)
                d = Exp(count) - Sim(i);
                sum_area = abs(d*prob_CFD(1)) + sum_area;
                count = count + 1;
            end
            rem_area = abs((Exp(count) - Sim(i))*(prob_TAT(i)-prob_CFD(count-1)));
            sum_area = sum_area + rem_area;
        end

    elseif length(Sim) == length(Exp)
        for i = 1:length(Sim)
            area = abs((Exp(i) - Sim(i))*prob_TAT(1));
            sum_area = area + sum_area;
        end

    else
        for i = 1:length(Exp)
            if rem_area ~= 0
                next_area = abs((Exp(i) - Sim(count))*(prob_TAT(count)-prob_CFD(i-1)));
                sum_area = sum_area + next_area;
                count = count + 1;
            end
```

```
            while prob_CFD(i) > prob_TAT(count)
                d = Exp(i) - Sim(count);
                sum_area = abs(d*prob_TAT(1)) + sum_area;
                count = count + 1;
            end
            rem_area = abs((Exp(i) - Sim(count))*(prob_CFD(i)-prob_TAT(count-1)));
            sum_area = sum_area + rem_area;
        end
    end
    diff = sum_area;
end
```

2. Modified Area Validation Metric

```
% Function created by Nolan Whiting on September 27, 2017 to calculate the
% d_plus and d_minus areas required to calculate the modified area
% validaiotn metric between two CDFs to estimate the model form
% uncertainty about a simulation.
% Note: This application of MAVM places a confidence interval about the
% experiment CDF in the calculation of model form uncertainty.
% Note: CDF_plots.m required for use of this function.

% Inputs:   Exp:        Vector of experimental samples
%           Sim:        Vector of simulation samples
%           S:          Number of samples
% Outputs:  diff_minus: Model form uncetainty smaller than simulation CDF
%           diff_plus:  Model form uncetainty larger than simulation CDF

function [diff_minus,diff_plus] = ModifiedAVM(Exp,Sim,S)

% Calcualte the confidence interval about the experimental samples, and
% shift the experiment CDF by that amount in each direction.
mn = mean(Exp);
std = sqrt(sum((Exp - mn).^2)/(S-1));
conf = 1.96*std/sqrt(S-1);
conf_inv = [mn - 1.96*std/sqrt(S-1),mn + 1.96*std/sqrt(S-1)];
Exp_left = Exp - 1.96*std/sqrt(S-1);
Exp_right = Exp + 1.96*std/sqrt(S-1);

[d_plus1,d_minus1] = CDF_plots(Exp_left,Sim);
[d_plus2,d_minus2] = CDF_plots(Exp_right,Sim);
if d_plus2 >= d_plus1
    d_max_plus = d_plus2;
else
    d_max_plus = d_plus1;
end
if abs(d_minus2) >= abs(d_minus1)
    d_max_minus = abs(d_minus2);
else
    d_max_minus = abs(d_minus1);
end

diff_minus = d_max_minus;
diff_plus = d_max_plus;
```

3. CDF_plots(used with Modified Area Validation Metric)

```
% Function created by Nolan Whiting on September 27, 2017 to calculate the
% d_plus and d_minus areas about one side of the experimental confidence
```

```
% interval CDF.
% Note: Used in association with ModifiedAVM.m

function [sum_area_plus,sum_area_minus] = CDF_plots(CFD_CL,TAT_CL)

% Inputs:    CFD_CL:            Vector of CDF of one side ofconfidence interval for
experimental samples
%           TAT_CL:         Vector of simulation samples
% Outputs:  sum_area_minus: Area smaller than simulation CDF
%           sum_area_plus:  Area larger than simulation CDF



% If there is intersection between the simulation and experiment CDFs, then
% the area between each individual sample and the simulation CDF will be
% calucalated and summed to provided the area validation metric.
sum_area_minus = 0;
sum_area_plus = 0;
prob_CFD = linspace(0,1,length(CFD_CL)+1);
prob_CFD = prob_CFD(2:end);
prob_TAT = linspace(0,1,length(TAT_CL)+1);
prob_TAT = prob_TAT(2:end);
count = 1;
rem_area = 0;

if length(TAT_CL) < length(CFD_CL)
    for i = 1:length(TAT_CL)
        if rem_area ~= 0
            next_area = (CFD_CL(count) - TAT_CL(i))*(prob_CFD(count)-prob_TAT(i-1));
            if next_area <= 0
                sum_area_minus = sum_area_minus + next_area;
            else
                sum_area_plus = sum_area_plus + next_area;
            end
            count = count + 1;
        end
        while prob_TAT(i) > prob_CFD(count)
            d = CFD_CL(count) - TAT_CL(i);
            if d <= 0
                sum_area_minus = d*prob_CFD(1) + sum_area_minus;
            else
                sum_area_plus = d*prob_CFD(1) + sum_area_plus;
            end
            count = count + 1;
        end
        rem_area = (CFD_CL(count) - TAT_CL(i))*(prob_TAT(i)-prob_CFD(count-1));
        if rem_area <= 0
                sum_area_minus = sum_area_minus + rem_area;
            else
                sum_area_plus = sum_area_plus + rem_area;
        end
    end

elseif length(TAT_CL) == length(CFD_CL)
    for i = 1:length(TAT_CL)
        area = (CFD_CL(i) - TAT_CL(i))*prob_TAT(1);
        if area <= 0
            sum_area_minus = sum_area_minus + area;
        else
            sum_area_plus = sum_area_plus + area;
        end
    end
```

```
else
    for i = 1:length(CFD_CL)
        if rem_area ~= 0
            next_area = (CFD_CL(i) - TAT_CL(count))*(prob_TAT(count)-prob_CFD(i-1));
            if next_area <= 0
                sum_area_minus = sum_area_minus + next_area;
            else
                sum_area_plus = sum_area_plus + next_area;
            end
            count = count + 1;
        end
        while prob_CFD(i) > prob_TAT(count)
            d = CFD_CL(i) - TAT_CL(count);
            if d <= 0
                sum_area_minus = d*prob_TAT(1) + sum_area_minus;
            else
                sum_area_plus = d*prob_TAT(1) + sum_area_plus;
            end
            count = count + 1;
        end
        rem_area = (CFD_CL(i) - TAT_CL(count))*(prob_CFD(i)-prob_TAT(count-1));
        if rem_area <= 0
            sum_area_minus = sum_area_minus + rem_area;
        else
            sum_area_plus = sum_area_plus + rem_area;
        end
    end

end
```

4. ASME's V&V 20

```
% Function created by Nolan Whiting on September 27, 2017 to calculate the
% E and u_val metrics needed for ASME's V&V 20

function [E,u_input,u_D,diff] = VV20(Exp,Sim,S)
% Inputs:   Exp:        Vector of experimental samples
%           Sim:        Vector of simulation samples
%           S:          Number of experimental samples
% Outputs:  E:          True model error
%           u_input:    Uncertainty due to uncertain inputs
%           u_D:        Uncertainty in the data
%           diff:       Total validation uncertainty multiplied by
%                       covervage factor (ku_val)
mn = mean(Sim);
u_input = sqrt(sum((Sim - mn).^2)/(length(Sim)-1));
std_E = sqrt(sum((Exp - mean(Exp)).^2)/(S-1));
u_num = 0;
u_D = 1.96*sqrt(std_E^2/(S-1));
u_val = sqrt(u_num^2 + u_input^2 + u_D^2);
E = -(mean(Exp) - mean(Sim));
diff = 2*u_val;
```

5. Overall Script for Making Predictions from Observations

```
% Script created by Nolan Whiting on February 26, 2018 to assess the
% predictive capability of Area Validation Metric, Modified Area Validation
% Metric, and V&V 20.

% Inputs:   T:           Vector of repeat tests
%           samples_vec: Vector of experiment samples
```

```
%           alpha_obs:   Angles of attack where observations are going to
%                        be made
%           delta_obs:   Flap deflections where observations are going to
%                        be made
%           alpha_pred:  Angles of attack where predictions are going to
%                        be made
%           delta_pred:  Flap deflections where predictions are going to
%                        be made


clear
clc
close all

T_vec = [5000,2500,1250,625,250,100,10];
samples_vec = [2,4,8,16,40,100,1000];

alpha_obs  = [10,30];%[1,4,15,19,24,28,32,35,40,41];%[6,13,28,34,42];%
delta_obs = [5,20];%[2,5,7,10,13,15,21,26,29,35,41];%[5,13,21,27,35];%
alpha_pred = [0,18,25,38,45];
delta_pred = [0,8,17,25];

assess = zeros(7,3,6);
[coeff_CFD,lift_CFD,moment_CFD] = Biharmonic_Fits();

for t = 1:length(T_vec)
    % Initialization for conservativeness/tightness assessment
    conserv_sum = zeros(1,6);
    tight_count = zeros(1,6);
    tight_sum = zeros(1,6);

    T = T_vec(t);
    samples = samples_vec(t);

    % Determining the true values
    S = 1000;
    for j = 1:length(delta_obs)
        for k = 1:length(alpha_obs)
            alpha = alpha_obs(k);
            delta = delta_obs(j);
            alpha_CFD = alpha + 0.5.*randn(1,S);
            delta_CFD = delta + randn(1,S);

            Coeff = coeff_CFD(alpha_CFD,delta_CFD);
            CL_CFD = Coeff(1:S);
            CM_CFD = Coeff(2*S+1:end);


            CL = sort(CL_CFD);
            CM = sort(CM_CFD);

            CL_mn_inf(j,k) = mean(CL);
            CM_mn_inf(j,k) = mean(CM);
        end
    end

    for j = 1:length(delta_pred)
        for k = 1:length(alpha_pred)
            alpha = alpha_pred(k);
            delta = delta_pred(j);
            alpha_CFD = alpha + 0.5.*randn(1,S);
            delta_CFD = delta + randn(1,S);
```

```
            Coeff = coeff_CFD(alpha_CFD,delta_CFD);
            CL_CFD = Coeff(1:S);
            CM_CFD = Coeff(2*S+1:end);

            CL = sort(CL_CFD);
            CM = sort(CM_CFD);

            CL_mn_inf_pred(j,k) = mean(CL);
            CM_mn_inf_pred(j,k) = mean(CM);
        end
    end

    for test = 1:T
    %     CL_mn = zeros(length(delta_obs),length(alpha_obs));
    %     CM_mn = zeros(length(delta_obs),length(alpha_obs));
        for j = 1:length(delta_obs)
            for k = 1:length(alpha_obs)

[CL_mn(j,k),CM_mn(j,k),diff_AVMCL(j,k),diff_AVMCM(j,k),diff_minus_CL(j,k),diff_plus_CL
(j,k),diff_minus_CM(j,k),diff_plus_CM(j,k),u_val_VVCL(j,k),u_val_VVCM(j,k),E_VVCL(j,k)
,E_VVCM(j,k)]                                                                    =
ValidationMetric_ALL(T,samples,alpha_obs(k),delta_obs(j),CL_mn_inf(j,k),CM_mn_inf(j,k)
,coeff_CFD);
            end
        end


        %AVM
        [CL_fit_AVM,pb_CL_AVM] = PredictionInterval_2D(alpha_obs,diff_AVMCL,delta_obs);
        [CM_fit_AVM,pb_CM_AVM] = PredictionInterval_2D(alpha_obs,diff_AVMCM,delta_obs);
        %MAVM
        [CL_fit_plus,pb_CL_plus]                                                 =
PredictionInterval_2D(alpha_obs,diff_plus_CL,delta_obs);
        [CL_fit_minus,pb_CL_minus]                                               =
PredictionInterval_2D(alpha_obs,diff_minus_CL,delta_obs);
        [CM_fit_plus,pb_plus_CM]                                                 =
PredictionInterval_2D(alpha_obs,diff_plus_CM,delta_obs);
        [CM_fit_minus,pb_minus_CM]                                               =
PredictionInterval_2D(alpha_obs,diff_minus_CM,delta_obs);

        %VV20
        [CL_fit_uval,pb_uvalCL]                                                  =
PredictionInterval_2D(alpha_obs,u_val_VVCL,delta_obs);
        [E_fit_VVCL,pb_VVCL] = PredictionInterval_2D(alpha_obs,E_VVCL,delta_obs);

        [CM_fit_uval,pb_uvalCM]                                                  =
PredictionInterval_2D(alpha_obs,u_val_VVCM,delta_obs);
        [E_fit_VVCM,pb_VVCM] = PredictionInterval_2D(alpha_obs,E_VVCM,delta_obs);

        %AVM
        pred_CLAVM = @(x,y) CL_fit_AVM(x,y) + pb_CL_AVM(x,y);
        pred_CMAVM = @(x,y) CM_fit_AVM(x,y) + pb_CM_AVM(x,y);
        %MAVM
        pred_plus_CL = @(x,y) CL_fit_plus(x,y) + pb_CL_plus(x,y);
        pred_minus_CL = @(x,y) CL_fit_minus(x,y) + pb_CL_minus(x,y);
        pred_plus_CM = @(x,y) CM_fit_plus(x,y) + pb_plus_CM(x,y);
        pred_minus_CM = @(x,y) CM_fit_minus(x,y) + pb_minus_CM(x,y);
        %VV20
        pred_VVCL = @(x,y) E_fit_VVCL(x,y) + CL_fit_uval(x,y) + pb_uvalCL(x,y);
        pred_VVCM = @(x,y) E_fit_VVCM(x,y) - CM_fit_uval(x,y) - pb_uvalCM(x,y);


        %% Conservativeness
```

```
        for i = 1:length(alpha_pred)
            for j = 1:length(delta_pred)
                [CL_a_pred,CM_4_pred] = ThinAirfoilTheory(alpha_pred(i),delta_pred(j));
                CL_mn_inf_p = CL_mn_inf_pred(j,i);
                CM_mn_inf_p = CM_mn_inf_pred(j,i);

                %AVM
                if CL_a_pred + pred_CLAVM(alpha_pred(i),delta_pred(j)) >= CL_mn_inf_p
&& CL_a_pred - pred_CLAVM(alpha_pred(i),delta_pred(j)) <= CL_mn_inf_p
                    conserv_CL_AVM_pred = 1;
                    tightness_AVMCL            =            abs(CL_a_pred            -
CL_mn_inf_p)/abs(2*pred_CLAVM(alpha_pred(i),delta_pred(j)));
                    tight_count(1) = tight_count(1) + 1;
                    tight_sum(1) = tight_sum(1) + tightness_AVMCL;
                else
                    conserv_CL_AVM_pred = 0;
                end

                if CM_4_pred + pred_CMAVM(alpha_pred(i),delta_pred(j)) >= CM_mn_inf_p
&& CM_4_pred - pred_CMAVM(alpha_pred(i),delta_pred(j)) <= CM_mn_inf_p
                    conserv_CM_AVM_pred = 1;
                    tightness_AVMCM            =            abs(CM_4_pred            -
CM_mn_inf_p)/abs(2*pred_CMAVM(alpha_pred(i),delta_pred(j)));
                    tight_count(2) = tight_count(2) + 1;
                    tight_sum(2) = tight_sum(2) + tightness_AVMCM;
                else
                    conserv_CM_AVM_pred = 0;
                end

                %MAVM
                if CL_a_pred + pred_plus_CL(alpha_pred(i),delta_pred(j)) >= CL_mn_inf_p
&& CL_a_pred - pred_minus_CL(alpha_pred(i),delta_pred(j)) <= CL_mn_inf_p
                    conserv_CL_MAVM_pred = 1;
                    tightness_MAVMCL           =            abs(CL_a_pred            -
CL_mn_inf_p)/(abs(pred_minus_CL(alpha_pred(i),delta_pred(j)))+abs(pred_plus_CL(alpha_p
red(i),delta_pred(j))));
                    tight_count(3) = tight_count(3) + 1;
                    tight_sum(3) = tight_sum(3) + tightness_MAVMCL;
                else
                    conserv_CL_MAVM_pred = 0;
                end

                if CM_4_pred + pred_plus_CM(alpha_pred(i),delta_pred(j)) >= CM_mn_inf_p
&& CM_4_pred - pred_minus_CM(alpha_pred(i),delta_pred(j)) <= CM_mn_inf_p
                    conserv_CM_MAVM_pred = 1;
                    tightness_MAVMCM           =            abs(CM_4_pred            -
CM_mn_inf_p)/(abs(pred_minus_CM(alpha_pred(i),delta_pred(j)))+abs(pred_plus_CM(alpha_p
red(i),delta_pred(j))));
                    tight_count(4) = tight_count(4) + 1;
                    tight_sum(4) = tight_sum(4) + tightness_MAVMCM;
                else
                    conserv_CM_MAVM_pred = 0;
                end
                %V&V 20 (Used as a Calibration)
                if       CL_a_pred    -    E_fit_VVCL(alpha_pred(i),delta_pred(j))    +
sqrt(pb_VVCL(alpha_pred(i),delta_pred(j))^2 + (CL_fit_uval(alpha_pred(i),delta_pred(j))
+   pb_uvalCL(alpha_pred(i),delta_pred(j)))^2)   >=   CL_mn_inf_p   &&   CL_a_pred   -
E_fit_VVCL(alpha_pred(i),delta_pred(j)) - sqrt(pb_VVCL(alpha_pred(i),delta_pred(j))^2 +
(CL_fit_uval(alpha_pred(i),delta_pred(j)) + pb_uvalCL(alpha_pred(i),delta_pred(j)))^2)
<= CL_mn_inf_p
                    conserv_CL_VV20_pred = 1;
```

```
                    tightness_VVCL                          =                          max(1-
abs(sqrt(pb_VVCL(alpha_pred(i),delta_pred(j))^2                                              +
(CL_fit_uval(alpha_pred(i),delta_pred(j))                                                    +
pb_uvalCL(alpha_pred(i),delta_pred(j)))^2)/CL_a_pred                                         -
E_fit_VVCL(alpha_pred(i),delta_pred(j))       -       CL_mn_inf_p),abs(CL_a_pred             -
E_fit_VVCL(alpha_pred(i),delta_pred(j))                                                      -
CL_mn_inf_p)/abs(sqrt(pb_VVCL(alpha_pred(i),delta_pred(j))^2                                 +
(CL_fit_uval(alpha_pred(i),delta_pred(j))                                                    +
pb_uvalCL(alpha_pred(i),delta_pred(j)))^2)));
                    tight_count(5) = tight_count(5) + 1;
                    tight_sum(5) = tight_sum(5) + tightness_VVCL;
                else
                    conserv_CL_VV20_pred = 0;
                    tightness_VVCL = 0;
                    tight_count(5) = tight_count(5) + 1;
                    tight_sum(5) = tight_sum(5) + tightness_VVCL;
                end

                if     CM_4_pred     -     E_fit_VVCM(alpha_pred(i),delta_pred(j))     +
sqrt(pb_VVCM(alpha_pred(i),delta_pred(j))^2 + (CM_fit_uval(alpha_pred(i),delta_pred(j))
+   pb_uvalCM(alpha_pred(i),delta_pred(j)))^2)   >=   CM_mn_inf_p   &&   CM_4_pred   -
E_fit_VVCM(alpha_pred(i),delta_pred(j)) - sqrt(pb_VVCM(alpha_pred(i),delta_pred(j))^2 +
(CM_fit_uval(alpha_pred(i),delta_pred(j)) + pb_uvalCM(alpha_pred(i),delta_pred(j)))^2)
<= CM_mn_inf_p
                    conserv_CM_VV20_pred = 1;
                    tightness_VVCM                          =                          max(1-
abs((sqrt(pb_VVCM(alpha_pred(i),delta_pred(j))^2                                             +
(CM_fit_uval(alpha_pred(i),delta_pred(j))                                                    +
pb_uvalCM(alpha_pred(i),delta_pred(j)))^2))/(CM_4_pred                                       -
E_fit_VVCM(alpha_pred(i),delta_pred(j))       -       CM_mn_inf_p)),abs(CM_4_pred            -
E_fit_VVCM(alpha_pred(i),delta_pred(j))                                                      -
CM_mn_inf_p)/sqrt(pb_VVCM(alpha_pred(i),delta_pred(j))^2                                     +
(CM_fit_uval(alpha_pred(i),delta_pred(j))                                                    +
pb_uvalCM(alpha_pred(i),delta_pred(j)))^2));%abs(CM_4_pred                                    -
CM_mn_inf_p)/abs(pred_VVCM(alpha_pred(i),delta_pred(j)));
                    tight_count(6) = tight_count(6) + 1;
                    tight_sum(6) = tight_sum(6) + tightness_VVCM;
                else
                    conserv_CM_VV20_pred = 0;
                    tightness_VVCM = 0;
                    tight_count(6) = tight_count(6) + 1;
                    tight_sum(6) = tight_sum(6) + tightness_VVCM;
                end


%                 %V&V 20 (Used as a Validation Standard, not calibration)
%                 if  CL_a_pred + pred_VVCL(alpha_pred(i),delta_pred(j)) >= CL_mn_inf_p
&& CL_a_pred -  pred_VVCL(alpha_pred(i),delta_pred(j)) <= CL_mn_inf_p
%                     conserv_CL_VV20_pred = 1;
%                                                 tightness_VVCL   =   abs(CL_a_pred   -
CL_mn_inf_p)/abs(pred_VVCL(alpha_pred(i),delta_pred(j)));
%                     tight_count(5) = tight_count(5) + 1;
%                     tight_sum(5) = tight_sum(5) + tightness_VVCL;
%                 else
%                     conserv_CL_VV20_pred = 0;
%                 end
%
%                 %if pred_VVCM(alpha_pred(i),delta_pred(j)) < 0
%                       if  CM_4_pred  +  abs(pred_VVCM(alpha_pred(i),delta_pred(j)))  >=
CM_mn_inf_p && CM_4_pred - abs(pred_VVCM(alpha_pred(i),delta_pred(j))) <= CM_mn_inf_p
%                     conserv_CM_VV20_pred = 1;
%                                                 tightness_VVCM   =   abs(CM_4_pred   -
CM_mn_inf_p)/abs(pred_VVCM(alpha_pred(i),delta_pred(j)));
```

```
%                     tight_count(6) = tight_count(6) + 1;
%                     tight_sum(6) = tight_sum(6) + tightness_VVCM;
%                 else
%                     conserv_CM_VV20_pred = 0;
%                 end

%                 else
%                           if CM_4_pred - pred_VVCM(alpha_pred(i),delta_pred(j)) <=
CM_mn_inf_p
%                         conserv_CM_VV20_pred = 1;
%                                                 tightness_VVCM = abs(CM_4_pred -
CM_mn_inf_p)/abs(pred_VVCM(alpha_pred(i),delta_pred(j)));
%                         tight_count(6) = tight_count(6) + 1;
%                         tight_sum(6) = tight_sum(6) + tightness_VVCM;
%                     else
%                         conserv_CM_VV20_pred = 0;
%                     end

                %end
                conserv_sum(1) = conserv_CL_AVM_pred + conserv_sum(1);
                conserv_sum(2) = conserv_CM_AVM_pred + conserv_sum(2);
                conserv_sum(3) = conserv_CL_MAVM_pred + conserv_sum(3);
                conserv_sum(4) = conserv_CM_MAVM_pred + conserv_sum(4);
                conserv_sum(5) = conserv_CL_VV20_pred + conserv_sum(5);
                conserv_sum(6) = conserv_CM_VV20_pred + conserv_sum(6);

            end
        end
        %Prevents division by zero
        for i = 1:length(tight_count)
            if tight_count(i) == 0
                tight_count(i) = 1;
            end
        end

    end


    for i = 1:length(conserv_sum)
        conserv_pred(i) = conserv_sum(i)/T/length(alpha_pred)/length(delta_pred);
        tight_pred(i) = tight_sum(i)/tight_count(i);
        assessment_pred(i) = 0.5*tight_pred(i) + 0.5*conserv_pred(i);
    end


    % Matrix of conservativeness, tightness, and overall assessment; indexing
    % (rows of sample sizes, columns of different methods, sheets of diffent variables
for conservativeness, tightness, and assessment)

    assess(t,:,:)                                                                    =
[conserv_pred(1),conserv_pred(3),conserv_pred(5);tight_pred(1),tight_pred(3),tight_pre
d(5);assessment_pred(1),assessment_pred(3),assessment_pred(5);conserv_pred(2),conserv_
pred(4),conserv_pred(6);tight_pred(2),tight_pred(4),tight_pred(6);assessment_pred(2),a
ssessment_pred(4),assessment_pred(6)]';
end
apd = linspace(-5,50,200);
dpd = linspace(-5,50,200);


% % Regression
% % AVM
% figure()
```

```
%
errorbar(alpha_pred(4),CL_fit_AVM(alpha_pred(4),delta_obs(2)),pb_CL_AVM(alpha_pred(4),
delta_obs(2)),'ro')
% hold on
%    h1    =    plot(alpha_obs,diff_AVMCL(2,:),'ob',apd,CL_fit_AVM(apd,delta_obs(2)),'-
r',apd,pred_CLAVM(apd,delta_obs(2)),'--k',apd,CL_fit_AVM(apd,delta_obs(2))-
pb_CL_AVM(apd,delta_obs(2)),'--k');
% legend('Prediction, \alpha = 37\circ','Observations','AVM Regression','95% Prediction
Interval (Upper)','95% Prediction Interval (Lower)')
% xlabel('\alpha (\circ)')
% ylabel('Area Validation Metric, d')
% title('Area Validation Metric (\delta = 21\circ)')
% set(h1,'LineWidth',1')
% xlim([apd(1),apd(end)])
%
% % AVM Plus
% figure()
%
errorbar(alpha_pred(4),CL_fit_plus(alpha_pred(4),delta_obs(2)),pb_CL_plus(alpha_pred(4
),delta_obs(3)),'ro')
% hold on
%   h1   =   plot(alpha_obs,diff_plus_CL(2,:),'ob',apd,CL_fit_plus(apd,delta_obs(2)),'-
r',apd,pred_plus_CL(apd,delta_obs(2)),'--k',apd,CL_fit_plus(apd,delta_obs(2))-
pb_CL_plus(apd,delta_obs(2)),'--k');
% legend('Prediction, \alpha = 37\circ','Observations','MAVM Regression','95% Prediction
Interval (Upper)','95% Prediction Interval (Lower)')
% xlabel('\alpha (\circ)')
% ylabel('Modified Area Validation Metric, d+')
% title('Modified Area Validation Metric (\delta = 21\circ)')
% set(h1,'LineWidth',1')
% xlim([apd(1),apd(end)])
%
% % AVM Minus
% figure()
%
errorbar(alpha_pred(4),CL_fit_minus(alpha_pred(4),delta_obs(3)),pb_CL_minus(alpha_pred
(4),delta_obs(3)),'ro')
% hold on
%  h1  =  plot(alpha_obs,diff_minus_CL(3,:),'ob',apd,CL_fit_minus(apd,delta_obs(3)),'-
r',apd,pred_minus_CL(apd,delta_obs(3)),'--k',apd,CL_fit_minus(apd,delta_obs(3))-
pb_CL_minus(apd,delta_obs(3)),'--k');
% legend('Prediction, \alpha = 37\circ','Observations','MAVM Regression','95% Prediction
Interval (Upper)','95% Prediction Interval (Lower)')
% xlabel('\alpha (\circ)')
% ylabel('Modified Area Validation Metric, d-')
% title('Modified Area Validation Metric (\delta = 21\circ)')
% set(h1,'LineWidth',1')
% xlim([apd(1),apd(end)])
%
% % VV 20, E
% figure()
%                                          h1                                          =
plot(alpha_pred(4),E_fit_VVCL(alpha_pred(4),delta_obs(3)),'ro',alpha_obs,E_VVCL(3,:),'
ob',apd,E_fit_VVCL(apd,delta_obs(3)),'-r',apd,E_fit_VVCL(apd,delta_obs(3))           +
pb_VVCL(apd,delta_obs(3)),'k--',apd,E_fit_VVCL(apd,delta_obs(3))                     -
pb_VVCL(apd,delta_obs(3)),'k--')
% legend('Prediction, \alpha = 37\circ','Observations','E Regression','95% Prediction
Interval')
% xlabel('\alpha (\circ)')
% ylabel('True Error, E')
% title('V&V 20, E (\delta = 21\circ)')
% set(h1,'LineWidth',1')
% xlim([apd(1),apd(end)])
```

```
%
% % VV20, uval
% figure()
%
errorbar(alpha_pred(4),CL_fit_uval(alpha_pred(4),delta_obs(3)),pb_uvalCL(alpha_pred(4)
,delta_obs(3)),'ro')
% hold on
%    h1    =    plot(alpha_obs,u_val_VVCL(3,:),'ob',apd,CL_fit_uval(apd,delta_obs(3)),'-
r',apd,CL_fit_uval(apd,delta_obs(3))+pb_uvalCL(apd,delta_obs(3)),'--
k',apd,CL_fit_uval(apd,delta_obs(3))-pb_uvalCL(apd,delta_obs(3)),'--k');
% legend('Prediction, \alpha = 37\circ','Observations','uval Regression','95% Prediction
Interval (Upper)','95% Prediction Interval (Lower)')
% xlabel('\alpha (\circ)')
% ylabel('uval')
% title('V&V 20, uval(\delta = 21\circ)')
% set(h1,'LineWidth',1')
% xlim([apd(1),apd(end)])


%% AVM
figure()
errorbar(alpha_pred,ThinAirfoilTheory(alpha_pred,delta_pred(1)),pred_CLAVM(alpha_pred,
delta_pred(1)),pred_CLAVM(alpha_pred,delta_pred(1)),'ro')
hold on
h1                  =                  plot(apd,ThinAirfoilTheory(apd,delta_pred(1)),'-
b',apd,lift_CFD(apd,delta_pred(1)),'-r',apd,pred_CLAVM(apd,delta_pred(1))            +
ThinAirfoilTheory(apd,delta_pred(1)),'--k',apd,ThinAirfoilTheory(apd,delta_pred(1))-
pred_CLAVM(apd,delta_pred(1)),'--k');
legend('Predictions','Simulation','Experiment','95% Prediction Interval (Upper)','95%
Prediction Interval (Lower)')
xlabel('\alpha (\circ)')
ylabel('c_l')
title('Area Validation Metric (\delta = 30\circ)')
set(h1,'LineWidth',1')
xlim([apd(1),apd(end)])


figure()
errorbar(alpha_pred,ThinAirfoilTheory(alpha_pred,delta_pred(3)),pred_CLAVM(alpha_pred,
delta_pred(3)),pred_CLAVM(alpha_pred,delta_pred(3)),'ro')
hold on
h1                  =                  plot(apd,ThinAirfoilTheory(apd,delta_pred(3)),'-
b',apd,lift_CFD(apd,delta_pred(3)),'-r',apd,pred_CLAVM(apd,delta_pred(3))            +
ThinAirfoilTheory(apd,delta_pred(3)),'--k',apd,ThinAirfoilTheory(apd,delta_pred(3))-
pred_CLAVM(apd,delta_pred(3)),'--k');
legend('Predictions','Simulation','Experiment','95% Prediction Interval (Upper)','95%
Prediction Interval (Lower)')
xlabel('\alpha (\circ)')
ylabel('c_l')
title('Area Validation Metric (\delta = 45\circ)')
set(h1,'LineWidth',1')
xlim([apd(1),apd(end)])


in = 1;
for i = 1:length(apd)
    [figCL_TAT(i),figCM_TAT(i)] = ThinAirfoilTheory(apd(i),delta_pred(in));
end
for i = 1:length(alpha_pred)
    [figCL_TATpred(i),figCM_TATpred(i)]                                                =
ThinAirfoilTheory(alpha_pred(i),delta_pred(in));
end
figure()
```

```
errorbar(alpha_pred,figCM_TATpred,pred_CMAVM(alpha_pred,delta_pred(in)),'ro')
hold on
h1           =           plot(apd,figCM_TAT,'-b',apd,moment_CFD(apd,delta_pred(in)),'-
r',apd,pred_CMAVM(apd,delta_pred(in))         +         figCM_TAT,'--k',apd,figCM_TAT-
pred_CMAVM(apd,delta_pred(in)),'--k');
legend('Predictions','Simulation','Experiment','95% Prediction Interval (Upper)','95%
Prediction Interval (Lower)')
xlabel('\alpha (\circ)')
ylabel('c_m')
title('Area Validation Metric (\delta = 0\circ)')
set(h1,'LineWidth',1')
xlim([apd(1),apd(end)])
%% MAVM
figure()
errorbar(alpha_pred,ThinAirfoilTheory(alpha_pred,delta_pred(1)),pred_minus_CL(alpha_pr
ed,delta_pred(1)),pred_plus_CL(alpha_pred,delta_pred(1)),'ro')
hold on
h1                  =                  plot(apd,ThinAirfoilTheory(apd,delta_pred(1)),'-
b',apd,lift_CFD(apd,delta_pred(1)),'-r',apd,pred_plus_CL(apd,delta_pred(1))           +
ThinAirfoilTheory(apd,delta_pred(1)),'--k',apd,ThinAirfoilTheory(apd,delta_pred(1))-
pred_minus_CL(apd,delta_pred(1)),'--k');
legend('Predictions','Simulation','Experiment','95% Prediction Interval (Upper)','95%
Prediction Interval (Lower)')
xlabel('\alpha (\circ)')
ylabel('c_l')
title('Modified Area Validation Metric (\delta = 30\circ)')
set(h1,'LineWidth',1')
xlim([apd(1),apd(end)])

in = 4;
figure()
errorbar(alpha_pred,ThinAirfoilTheory(alpha_pred,delta_pred(in)),pred_minus_CL(alpha_p
red,delta_pred(in)),pred_plus_CL(alpha_pred,delta_pred(in)),'ro')
hold on
h1                  =                  plot(apd,ThinAirfoilTheory(apd,delta_pred(in)),'-
b',apd,lift_CFD(apd,delta_pred(in)),'-r',apd,pred_plus_CL(apd,delta_pred(in))         +
ThinAirfoilTheory(apd,delta_pred(in)),'--k',apd,ThinAirfoilTheory(apd,delta_pred(in))-
pred_minus_CL(apd,delta_pred(in)),'--k');
legend('Predictions','Simulation','Experiment','95% Prediction Interval (Upper)','95%
Prediction Interval (Lower)')
xlabel('\alpha (\circ)')
ylabel('c_l')
title('Modified Area Validation Metric (\delta = 25\circ)')
set(h1,'LineWidth',1')
xlim([apd(1),apd(end)])

for i = 1:length(apd)
    [figCL_TAT(i),figCM_TAT(i)] = ThinAirfoilTheory(apd(i),delta_pred(in));
end
for i = 1:length(alpha_pred)
    [figCL_TATpred(i),figCM_TATpred(i)]                                          =
ThinAirfoilTheory(alpha_pred(i),delta_pred(in));
end
figure()
errorbar(alpha_pred,figCM_TATpred,pred_minus_CM(alpha_pred,delta_pred(in)),pred_plus_C
M(alpha_pred,delta_pred(in)),'ro')
hold on
h1           =           plot(apd,figCM_TAT,'-b',apd,moment_CFD(apd,delta_pred(in)),'-
r',apd,pred_plus_CM(apd,delta_pred(in))         +         figCM_TAT,'--k',apd,figCM_TAT-
pred_minus_CM(apd,delta_pred(in)),'--k');
legend('Predictions','Simulation','Experiment','95% Prediction Interval (Upper)','95%
Prediction Interval (Lower)')
xlabel('\alpha (\circ)')
```

```
ylabel('c_m')
title('Modified Area Validation Metric (\delta = 25\circ)')
set(h1,'LineWidth',1')
xlim([apd(1),apd(end)])


% in = 3;
% figure()
%
errorbar(alpha_pred,ThinAirfoilTheory(alpha_pred,delta_pred(in)),pred_minus_CL(alpha_p
red,delta_pred(in)),pred_plus_CL(alpha_pred,delta_pred(in)),'ro')
% hold on
%           h1           =           plot(apd,ThinAirfoilTheory(apd,delta_pred(in)),'-
b',apd,lift_CFD(apd,delta_pred(in)),'-r',apd,pred_plus_CL(apd,delta_pred(in))         +
ThinAirfoilTheory(apd,delta_pred(in)),'--k',apd,ThinAirfoilTheory(apd,delta_pred(in))-
pred_minus_CL(apd,delta_pred(in)),'--k');
% legend('Predictions','Simulation','Experiment','95% Prediction Interval (Upper)','95%
Prediction Interval (Lower)')
% xlabel('\alpha (\circ)')
% ylabel('c_l')
% title('Modified Area Validation Metric (\delta = 25\circ)')
% set(h1,'LineWidth',1')
% xlim([apd(1),apd(end)])
%
% in = 4;
% figure()
%
errorbar(alpha_pred,ThinAirfoilTheory(alpha_pred,delta_pred(in)),pred_minus_CL(alpha_p
red,delta_pred(in)),pred_plus_CL(alpha_pred,delta_pred(in)),'ro')
% hold on
%           h1           =           plot(apd,ThinAirfoilTheory(apd,delta_pred(in)),'-
b',apd,lift_CFD(apd,delta_pred(in)),'-r',apd,pred_plus_CL(apd,delta_pred(in))         +
ThinAirfoilTheory(apd,delta_pred(in)),'--k',apd,ThinAirfoilTheory(apd,delta_pred(in))-
pred_minus_CL(apd,delta_pred(in)),'--k');
% legend('Predictions','Simulation','Experiment','95% Prediction Interval (Upper)','95%
Prediction Interval (Lower)')
% xlabel('\alpha (\circ)')
% ylabel('c_l')
% title('Modified Area Validation Metric (\delta = 37\circ)')
% set(h1,'LineWidth',1')
% xlim([apd(1),apd(end)])

%% VV 20
figure()
errorbar(alpha_pred,ThinAirfoilTheory(alpha_pred,delta_pred(1)),pred_VVCL(alpha_pred,d
elta_pred(1)),zeros(1,length(alpha_pred)),'ro')
hold on
h1                  =                  plot(apd,ThinAirfoilTheory(apd,delta_pred(1)),'-
b',apd,lift_CFD(apd,delta_pred(1)),'-r',apd,ThinAirfoilTheory(apd,delta_pred(1))-
pred_VVCL(apd,delta_pred(1)),'--k');
legend('Predictions','Simulation','Experiment','Sim + (E + k*uval)')
xlabel('\alpha (\circ)')
ylabel('c_l')
title('V&V 20 (\delta = 30\circ)')
set(h1,'LineWidth',1')
xlim([apd(1),apd(end)])

in = 1;
for i = 1:length(apd)
    [figCL_TAT(i),figCM_TAT(i)] = ThinAirfoilTheory(apd(i),delta_pred(in));
end
for i = 1:length(alpha_pred)
```

```
    [figCL_TATpred(i),figCM_TATpred(i)]                                                    =
ThinAirfoilTheory(alpha_pred(i),delta_pred(in));
end
figure()
errorbar(alpha_pred,figCM_TATpred,pred_CMAVM(alpha_pred,delta_pred(in)),'ro')
hold on
h1           =           plot(apd,figCM_TAT,'-b',apd,moment_CFD(apd,delta_pred(in)),'-
r',apd,pred_CMAVM(apd,delta_pred(in))          +          figCM_TAT,'--k',apd,figCM_TAT-
pred_CMAVM(apd,delta_pred(in)),'--k');
legend('Predictions','Simulation','Experiment','95% Prediction Interval (Upper)','95%
Prediction Interval (Lower)')
xlabel('\alpha (\circ)')
ylabel('c_m')
title('Area Validation Metric (\delta = 0\circ)')
set(h1,'LineWidth',1')
xlim([apd(1),apd(end)])


% figure()
%                       errorbar(alpha_pred,ThinAirfoilTheory(alpha_pred,delta_pred(4))-
E_fit_VVCL(alpha_pred,delta_pred(4)),       pb_uvalCL(alpha_pred,delta_pred(4))       +
CL_fit_uval(alpha_pred,delta_pred(4)),'ro') %pb_VVCL(alpha_pred,delta_pred(4))+
% hold on
%           h1           =           plot(apd,ThinAirfoilTheory(apd,delta_pred(4)),'-
b',apd,lift_CFD(apd,delta_pred(4)),'-r');
% legend('Predictions','Simulation','Experiment')
% xlabel('\alpha (\circ)')
% ylabel('c_l')
% title('V&V 20 (\delta = 25\circ)')
% set(h1,'LineWidth',1')
% set(gca,'FontSize',12')
% xlim([apd(1),apd(end)])

figure()
errorbar(alpha_pred,ThinAirfoilTheory(alpha_pred,delta_pred(1))-
E_fit_VVCL(alpha_pred,delta_pred(1)),       pb_uvalCL(alpha_pred,delta_pred(1))       +
CL_fit_uval(alpha_pred,delta_pred(1)),'ro') %pb_VVCL(alpha_pred,delta_pred(1))+
hold on
h1                  =                  plot(apd,ThinAirfoilTheory(apd,delta_pred(1)),'-
b',apd,lift_CFD(apd,delta_pred(1)),'-r');
legend('Predictions','Simulation','Experiment')
xlabel('\alpha (\circ)')
ylabel('c_l')
title('V&V 20 (\delta = 30\circ)')
set(h1,'LineWidth',1')
set(gca,'FontSize',12')
xlim([apd(1),apd(end)])

for i = 1:length(dpd)
    d_CL(i) = lift_CFD(alpha_pred(5),dpd(i));
end
figure()
errorbar(delta_pred,ThinAirfoilTheory(alpha_pred(5),delta_pred),pred_minus_CL(alpha_pr
ed(5),delta_pred),pred_minus_CL(alpha_pred(5),delta_pred),'ro')
hold on
h1 = plot(dpd,ThinAirfoilTheory(alpha_pred(5),dpd),'-b',dpd,d_CL,'-r');
legend('Predictions','Simulation','Experiment','95% Predction Interval')
xlabel('\delta (\circ)')
ylabel('c_l')
title('MAVM (\alpha = 45\circ)')
set(h1,'LineWidth',1')
xlim([apd(1),apd(end)])
```

```
for i = 1:length(dpd)
    d_CL(i) = lift_CFD(alpha_pred(2),dpd(i));
end
figure()
errorbar(delta_pred,ThinAirfoilTheory(alpha_pred(2),delta_pred)-
E_fit_VVCL(alpha_pred(2),delta_pred),pb_VVCL(alpha_pred(2),delta_pred)+
pb_uvalCL(alpha_pred(2),delta_pred) + CL_fit_uval(alpha_pred(2),delta_pred),'ro')
hold on
h1 = plot(dpd,ThinAirfoilTheory(alpha_pred(2),dpd),'-b',dpd,d_CL,'-r');
legend('Predictions','Simulation','Experiment','Sim + (E + k*uval)')
xlabel('\delta (\circ)')
ylabel('c_l')
title('V&V 20 (\alpha = 18\circ)')
set(h1,'LineWidth',1')
xlim([apd(1),apd(end)])


for i = 1:length(dpd)
    d_CL(i) = lift_CFD(alpha_pred(5),dpd(i));
end
figure()
errorbar(delta_pred,ThinAirfoilTheory(alpha_pred(5),delta_pred)-
E_fit_VVCL(alpha_pred(5),delta_pred),pb_VVCL(alpha_pred(5),delta_pred)+
pb_uvalCL(alpha_pred(5),delta_pred) + CL_fit_uval(alpha_pred(5),delta_pred),'ro')
hold on
h1 = plot(dpd,ThinAirfoilTheory(alpha_pred(5),dpd),'-b',dpd,d_CL,'-r');
legend('Predictions','Simulation','Experiment')
xlabel('\delta (\circ)')
ylabel('c_l')
title('V&V 20 (\alpha = 45\circ)')
set(h1,'LineWidth',1')
xlim([apd(1),apd(end)])

in = 1;
for i = 1:length(apd)
    [figCL_TAT(i),figCM_TAT(i)] = ThinAirfoilTheory(apd(i),delta_pred(in));
end
for i = 1:length(alpha_pred)
    [figCL_TATpred(i),figCM_TATpred(i)]                                    =
ThinAirfoilTheory(alpha_pred(i),delta_pred(in));
end
figure()
errorbar(alpha_pred,figCM_TATpred-
E_fit_VVCM(alpha_pred,delta_pred(in)),pb_VVCM(alpha_pred,delta_pred(in))+
pb_uvalCM(alpha_pred,delta_pred(in)) + CM_fit_uval(alpha_pred,delta_pred(in)),'ro')
hold on
h1 = plot(apd,figCM_TAT,'-b',apd,moment_CFD(apd,delta_pred(in)),'-r');
legend('Predictions','Simulation','Experiment')
xlabel('\alpha (\circ)')
ylabel('c_m')
title('V&V 20 (\delta = 0\circ)')
set(h1,'LineWidth',1')
xlim([apd(1),apd(end)])
```

6. All Validation Methods at Observation Locations

```
function
[CL_mn,CM_mn,diff_AVMCL,diff_AVMCM,diff_minus_CL,diff_plus_CL,diff_minus_CM,diff_plus_
CM,diff_VVCL,diff_VVCM,E_VVCL,E_VVCM]                                      =
ValidationMetric_ALL(T,samples,alpha,delta,CL_mn_inf,CM_mn_inf,coeff_CFD)
```

```
% Function created by Nolan Whiting on December 13, 2017 to calculate the
% area validation metric between two CDFs to estimate the model form
% uncertainty about a simulation.

% Inputs:   T:              Number of repeat tests being conducted
%           samples:        Vector of simulation samples
%           alpha:          angle of attack where observation is made
%           delta:          flap deflection where observation is made
%           CL_mn_inf:      mean lift coefficient as exp. samples go to inf.
%           CM_mn_inf:      mean moment coefficent as exp. samples go to inf.
%           coeff_CFD:      Biharmonic fits for lift and moment coefficent
% Outputs:  CL_mn:          mean experiment lift coefficent
%           CM_mn:          mean experiment moment coefficient
%           diff_AVMCL      observed area validation metric results (CL)
%           diff_AVMCM      observed area validation metric results (CM)
%           diff_minus_CL   observed d- area (CL)
%           diff_plus_CL    observed d+ area(CL)
%           diff_minus_CM   observed d- area (CM)
%           diff_plus_CM    observed d+ area (CM)
%           diff_VVCL       observed validation uncertainty (CL)
%           diff_VVCM       observed validation uncertainty (CM)
%           E_VVCL          observed true error (CL)
%           E_VVCM          observed true error (CM)
%



alpha_TAT = lhsnorm(alpha,0.5,1000);
delta_TAT = lhsnorm(delta,1,1000);
for i = 1:length(alpha_TAT)
    [CL_a(i),CM_4(i)] = ThinAirfoilTheory(alpha_TAT(i),delta_TAT(i));
end

CL_a = sort(CL_a);
CM_4 = sort(CM_4);



S = samples;

%for i = 1:T
    alpha_CFD = alpha + 0.5.*randn(1,S);
    delta_CFD = delta + randn(1,S);

    Coeff = coeff_CFD(alpha_CFD,delta_CFD);
    CL_CFD = Coeff(1:S);
    CM_CFD = Coeff(2*S+1:end);
    CL = sort(CL_CFD);
    CM = sort(CM_CFD);

    CL = CL + sort(0.025*randn(1,S)).*CL;
    CM = CM + sort(0.025*randn(1,S)).*CM;

    CL = sort(CL_CFD);
    CM = sort(CM_CFD);

    CL_mn = mean(CL);
    CM_mn = mean(CM);

    % Area Validation Metric
    diff_AVMCL = AreaValidationMetric(CL,CL_a);
    if mean(CL_a) - diff_AVMCL <= CL_mn_inf && mean(CL_a) + diff_AVMCL >= CL_mn_inf
```

```
        conservCL_AVM(i) = 1;
    else
        conservCL_AVM(i) = 0;
    end


    diff_AVMCM = AreaValidationMetric(CM,CM_4);
    if mean(CM_4) - diff_AVMCM <= CM_mn_inf && mean(CM_4) + diff_AVMCM >= CM_mn_inf
        conservCM_AVM(i) = 1;
    else
        conservCM_AVM(i) = 0;
    end

    % Modified Area Validation Metric (CI)
    [diff_minus_CL,diff_plus_CL] = ModifiedAVM(CL,CL_a,S);
    if mean(CL_a)-diff_minus_CL <= CL_mn_inf && mean(CL_a) + diff_plus_CL >= CL_mn_inf
        conservCL_MAVM(i) = 1;
    else
        conservCL_MAVM(i) = 0;
    end


    [diff_minus_CM,diff_plus_CM] = ModifiedAVM(CM,CM_4,S);
    if  mean(CM_4) - diff_minus_CM <= CM_mn_inf && mean(CM_4) + diff_plus_CM >= CM_mn_inf
        conservCM_MAVM(i) = 1;
    else
        conservCM_MAVM(i) = 0;
    end


    % V&V 20
    [E_VVCL,u_input_VVCL,u_D_VVCL,diff_VVCL] = VV20(CL,CL_a,S);
    if CL_mn - diff_VVCL <= CL_mn_inf && CL_mn + diff_VVCL >= CL_mn_inf
        conservCL_VV20(i) = 1;
    else
        conservCL_VV20(i) = 0;
    end

    [E_VVCM,u_input_VVCM,u_D_VVCM,diff_VVCM] = VV20(CM,CM_4,S);
    if CM_mn - diff_VVCM <= CM_mn_inf && CM_mn + diff_VVCM >= CM_mn_inf
        conservCM_VV20(i) = 1;
    else
        conservCM_VV20(i) = 0;
    end
```

7. Prediction Fits

```
% Function created by Nolan Whiting on February 2, 2018 to fit the metrics
% from the observation locations, and use those fits to make predications.

function [fit,pb] = PredictionInterval_2D(alpha_fit,d_fit,delta_fit)

% Inputs:   alpha_fit:  Observation angles of attack
%           d_fit:      Matrix of observed metric that is being fit
%           delta_fit:  Observation flap deflections

% Outputs:  fit: Model form uncetainty smaller than simulation CDF
%           pb:  Model form uncetainty larger than simulation CDF

% Reorganizes observation metrics and observation conditions into ordered
% pairs for fitting.
count = 1;
```

```
for i = 1:length(delta_fit)
    for j = 1:length(alpha_fit)
       obs(count,:) = [alpha_fit(j),delta_fit(i)];
       count = count + 1;
    end
end
diff_fit = ones(length(alpha_fit)*length(delta_fit),1);
for i = 1:length(delta_fit)
    diff_fit((i-1)*length(alpha_fit)+1:i*length(alpha_fit)) = d_fit(i,:);
end


% If only a limited number of observations are provided, a linear surface
% fit will be used. However, if more than 6 observations are made, a
% quadratic fit will be used.
if length(diff_fit) < 6
    f = fit(obs,diff_fit,'poly11');
    a10 = f.p10;
    a01 = f.p01;
    a00 = f.p00;
    CL_fit1 = @(x,y) a10.*x + a01.*y + a00;
    CL_fit = CL_fit1;

else
    f = fit(obs,diff_fit,'poly22');
    a20 = f.p20;
    a02 = f.p02;
    a11 = f.p11;
    a10 = f.p10;
    a01 = f.p01;
    a00 = f.p00;
    CL_fit1 = @(x,y) a20.*x.^2 + a02.*y.^2 + a11.*x.*y + a10.*x + a01.*y + a00;
    fit = CL_fit1;
end

% Places 95% prediction interval about the regression fit.
t = 1.96;
N = length(alpha_fit);
conf_inf_l   =   @(x,y)fit(x,y)   -   t*std(std(d_fit)).*sqrt(1   +   1./N   +   N.*(x   -
mean(alpha_fit)).^2./(N.*sum(alpha_fit.^2) - (sum(alpha_fit)).^2));
conf_inf_u   =   @(x,y)fit(x,y)   +   t*std(std(d_fit)).*sqrt(1   +   1./N   +   N.*(x   -
mean(alpha_fit)).^2./(N.*sum(alpha_fit.^2) - (sum(alpha_fit)).^2));
alpha_pred = linspace(6,45,100);
pb = @(x,y) conf_inf_u(x,y) - fit(x,y);
end
```

8. Bayesian Updating

```
% Function created by Nolan Whiting on July 19, 2018 to assess the
% predictive capability of Bayesian updating

% Inputs:   N:           Number of simulation samples
%           S:           Number of experimental samples
%           n:           Number of GP realizations
%           samples_vec: Vector of experiment samples
%           alpha_obs:   Angles of attack where observations are going to
%                        be made
%           delta_obs:   Flap deflections where observations are going to
%                        be made
%           alpha_pred:  Angles of attack where predictions are going to
%                        be made
```

```
%           delta_pred:  Flap deflections where predictions are going to
%                        be made
clc
close all

global var mn Ko L CI samples var_PD

[coeff_CFD,lift_CFD,momt_CFD] = Biharmonic_Fits();


N = 1000; % Samples of TAT/LLT
S = 100; % CFD samples
n = 3000; % number of GP realizations

alpha_obs = [6,13,28,34,42];%[1,4,15,19,24,28,32,35,40,41];%[10,30];%
delta_obs = [5,13,21,27,35];%[2,5,7,10,13,15,21,26,29,35,41]; %[5,20];%

alpha_pred = [0,18,25,38,45];%[-2,4,6,13,17,28,34,37,40,42] + 1;
delta_pred = [0,8,17,25];%[0,5,7,10,15,21,25,27,35,42,45] - 1;

for kk = 1:length(delta_obs)
    for ii = 1:length(alpha_obs)
        alpha = alpha_obs(ii);
        delta = delta_obs(kk);
        alpha_TAT = lhsnorm(alpha,0.5,N);
        delta_TAT = lhsnorm(delta,1,N);
        for i = 1:N
            [CL_a(i),CM_4(i)] = ThinAirfoilTheory(alpha_TAT(i),delta_TAT(i));
        end


        for i = 1:T
            alpha_CFD = alpha + 0.5.*randn(1,S);
            delta_CFD = delta + randn(1,S);


            CL = lift_CFD(alpha_CFD,delta_CFD);
            CM = momt_CFD(alpha_CFD,delta_CFD);

            CL = sort(CL);
            CM = sort(CM);

            CL = CL + sort(0.025*randn(1,S)).*CL;
            CM = CM + sort(0.025*randn(1,S)).*CL;

            CL_mn = mean(CL);
            CM_mn = mean(CM);
            sig_CL(kk,ii) = std((mean(CL_a) - CL));
            sig_CM(kk,ii) = std((mean(CM_4) - CM));
            conf_inf_CL(kk,ii) = 1.96*sig_CL(kk,ii)/sqrt(S);
            conf_inf_CM(kk,ii) = 1.96*sig_CM(kk,ii)/sqrt(S);
            diff_CL(kk,ii) = -(CL_mn - mean(CL_a));
            diff_CM(kk,ii) = -(CM_mn - mean(CM_4));
        end
    end
end


xp = -5:1:50;
yp = -4:1:50;

xobs = meshgrid(alpha_obs,delta_obs);
yobs = meshgrid(delta_obs,alpha_obs);
```

```
nx = length(alpha_obs)*length(delta_obs);
xobs = reshape(xobs,[1,nx]);
%yobs = reshape(yobs,[1,nx]);
for i = 1:length(alpha_obs)
    yobs((i-1)*length(delta_obs)+1:i*length(delta_obs))                                    =
delta_obs;%reshape(yobs,[1,nx]
end
[X,XPrime] = meshgrid(xobs,xobs);
[Y,YPrime] = meshgrid(yobs,yobs);

count = 1;
for i = 1:length(delta_obs)
    for j = 1:length(alpha_obs)
       obs(count,:) = [alpha_obs(j),delta_obs(i)];
       count = count + 1;
    end
end

yo = reshape(diff_CL,[length(xobs),1]);
sig = sig_CL;
sig = reshape(sig,[length(xobs),1]);
[model_disc,CI_mat]                                                                    =
GP_pt2D(alpha_obs,delta_obs,xobs,yobs,X,XPrime,Y,YPrime,yo,xp,yp,sig,n,1,conf_inf_CL);
%[mean_CL,conf_u_CL,conf_l_CL] =


for i = 1:length(yp)
    for j = 1:length(xp)
        CL_true(i,j) = lift_CFD(xp(j),yp(i));
        CL_model(i,j) = ThinAirfoilTheory(xp(j),yp(i));
        CL_calib(i,j) = ThinAirfoilTheory(xp(j),yp(i)) - model_disc(i,j);
        CL_confUp(i,j) = CL_calib(i,j) + CI_mat(i,j);
        CL_confLow(i,j) = CL_calib(i,j) - CI_mat(i,j);
    end
end

count_x = 1;
for i = 1:length(alpha_pred)
    while alpha_pred(i) >= xp(count_x)
        count_x = count_x + 1;
    end
    flag_x = count_x;
    count_y = 1;
    for j = 1:length(delta_pred)
        while delta_pred(j) >= yp(count_y)
            count_y = count_y + 1;
        end
        flag_y = count_y;
        if      CL_confUp(flag_x,flag_y)      >=      CL_true(flag_x,flag_y)      &&
CL_confLow(flag_x,flag_y) <= CL_true(flag_x,flag_y)
            conserv(i,j) = 1;
            tightness(i,j)              =              max(abs((CL_true(flag_x,flag_y)-
CL_calib(flag_x,flag_y))/(CL_confUp(flag_x,flag_y)-CL_calib(flag_x,flag_y))),1       -
abs((CL_confUp(flag_x,flag_y)-CL_calib(flag_x,flag_y))/(CL_model(flag_x,flag_y)-
CL_true(flag_x,flag_y))));
        else
            conserv(i,j) = 0;
            tightness(i,j) = 0;
        end
    end
end
conservCL_GP = mean(mean(conserv));
tightCL_GP = mean(mean(tightness));
```

```
yo = reshape(diff_CM,[length(xobs),1]);
sig = sig_CM;
sig = reshape(sig,[length(xobs),1]);
[model_disc,CI_mat]                                                                    =
GP_pt2D(alpha_obs,delta_obs,xobs,yobs,X,XPrime,Y,YPrime,yo,xp,yp,sig,n,2,conf_inf_CM);
%[mean_CM,conf_u_CM,conf_l_CM] =


for i = 1:length(yp)
    for j = 1:length(xp)
        CM_true(i,j) = momt_CFD(xp(j),yp(i));
        [CL_model(i,j),CM_model(i,j)] = ThinAirfoilTheory(xp(j),yp(i));
        CM_calib(i,j) = CM_model(i,j) - model_disc(i,j);
        CM_confUp(i,j) = CM_calib(i,j) + CI_mat(i,j);
        CM_confLow(i,j) = CM_calib(i,j) - CI_mat(i,j);
    end
end



count_x = 1;
for i = 1:length(alpha_pred)
    while alpha_pred(i) >= xp(count_x)
        count_x = count_x + 1;
    end
    flag_x = count_x;
    count_y = 1;
    for j = 1:length(delta_pred)
        while delta_pred(j) >= yp(count_y)
            count_y = count_y + 1;
        end
        flag_y = count_y;
        if       CM_confUp(flag_x,flag_y)       >=       CM_true(flag_x,flag_y)       &&
CM_confLow(flag_x,flag_y) <= CM_true(flag_x,flag_y)
            conserv(i,j) = 1;
            tightness(i,j)              =              max(abs((CM_true(flag_x,flag_y)-
CM_calib(flag_x,flag_y))/(CM_confUp(flag_x,flag_y)-CM_calib(flag_x,flag_y))),1        -
abs((CM_confUp(flag_x,flag_y)-CM_calib(flag_x,flag_y))/(CM_model(flag_x,flag_y)-
CM_true(flag_x,flag_y))));
        else
            conserv(i,j) = 0;
            tightness(i,j) = 0;
        end
    end
end
conservCM_GP = mean(mean(conserv));
tightCM_GP = mean(mean(tightness));
```

9. Covariance Matrices for Bayesian Updating

```
function                              [model_disc,CI_mat]                              =
GP_pt2D(alpha_obs,delta_obs,xobs,yobs,X,XPrime,Y,YPrime,yo,xp,yp,sig,num,variable)%[me
an_gp,conf_u,conf_l] =
% Function created by Nolan Whiting on September 20, 2018 to assess the
% predictive capability of Bayesian updating with two uncertain inputs.
% This function creates the covariance matrices.

% Inputs:   alpha_obs:   Angles of attack where observations are going to
%                        be made
%           delta_obs:   Flap deflections where observations are going to
```

```
%                       be made
%           xobs:       Mesh grid of observation angles of attack
%           yobs:       Mesh grid of observation flap deflections
%           X:          Mesh grid of xobs with yobs
%           Xprime:     Tranpose of X
%           Y:          Mesh grid of yobs with xobs
%           Yprime:     Transpose of Y
%           yo:         Observation model discrepancies
%           xp:         Span of angles of attack over which is updated
%           yp:         Span of flap deflections over which is updated
%           num:        Number of GP realizations to be made
%           sig:        Uncertainty in the observation discrepancies
%           variable:   Variable being updated (1 for cl, 2 for cm)


xp_orig = xp;
yp_orig = yp;
global var mn Ko L CI samples var_PD

xp_grid = meshgrid(xp,yp);
yp_grid = meshgrid(yp,xp);
nx = length(xp)*length(yp);
xp = reshape(xp_grid,[1,nx]);
%yp = reshape(yp_grid,[1,nx]);
for i = 1:length(xp_orig)
    yp((i-1)*length(yp_orig)+1:i*length(yp_orig)) = yp_orig;%reshape(yobs,[1,nx]
end
[Xp,XpPrime] = meshgrid(xp,xp);
[Yp,YpPrime] = meshgrid(yp,yp);


[lx]                                                                                =
l_parameter(yo(1:length(delta_obs):end),alpha_obs,variable,sig(1:length(delta_obs):end
))
[ly] = l_parameter(yo(1:length(delta_obs)),delta_obs,variable,sig(1:length(delta_obs)))


 if variable == 1
     lx = 1;
     ly = 50;
     sig2 = 0.4;
 elseif variable == 2
    lx = 1.5;
    ly = 45;
    sig2 = 0.1;
 end
%sig2 = std(yo);



for j = 1:length(xobs)
    for i = 1:length(xp)
        K(j,i)    =    sig2.^2.*exp(-abs((xobs(j)-xp(i)).^2)/2/lx^2    +    -abs(yobs(j)-
yp(i)).^2/2/ly^2);
    end
end

Kp = sig2.^2.*exp(-abs(Xp - XpPrime)./2./lx.^2 + -abs(Yp - YpPrime)./2./ly.^2);

Ko = sig2.^2.*exp(-abs(X - XPrime)./2./lx.^2 + -abs(Y - YPrime)./2./ly.^2);

mn = K'*inv(Ko + (sig.^2).*eye(length(xobs)))*yo;
```

```
var = Kp - K'*inv(Ko  + (sig.^2).*eye(length(yo)))*K;

[V,D] = eig(var);
d = diag(D);
for i = 1:length(d)
    if d(i) < 1E-3
        d(i) = 1E-3;
    end
end
D_c = diag(d);
var_PD = V*D_c*V';
var_PD = real(var_PD);
L = chol(var_PD);
% L = chol(var + 1.*eye(length(mn)));

samples = zeros(num,nx);

figure()
for i = 1:num
    samples(i,:) = mn + (L'*randn(length(mn),1)); % + yo
    plot(linspace(1,nx,nx),samples(i,:))
    hold on
end
ylabel('\delta')
hold off

for i = 1:length(mn)
    mean_samp(i) = mean(samples(:,i));
end

CI = zeros(length(mn),1);
for i = 1:length(mn)
    CI(i) = 1.96*std(samples(:,i));
end
model_disc = reshape(mean_samp,[length(yp_orig),length(xp_orig)]);
CI_mat = reshape(CI,[length(yp_orig),length(xp_orig)]);
```

10. Length Scale for Bayesian Updating

```
function [l] = l_parameter(yo,x,variable,sig)
% Function created by Nolan Whiting on March 25, 2018 to determine the
% length scale for Bayesian updating
% Inputs:   yo:         observation discrepancies
%           x:          locations where observations were made
%           variable:   1 for cl, 2 for moment coefficent
%           sig:        uncertainty in the observations
% Outputs:  l:      length scale


l = 0:0.01:150;
p = zeros(length(l));
for kk = 1:length(l)
    for j = 1:length(x)
        for i = 1:length(x)
            Ko(i,j) = exp(-(abs(x(j)-x(i)).^2)/2/l(kk)^2);
        end
    end
    %p(kk) = (-0.5*yo'*(Ko)*yo - 0.5*log(det(Ko)) - length(yo)/2*log(2*pi));
    p(kk)   =   ((-0.5*yo'*(Ko   +   (sig.^2).*eye(length(yo)))*yo   -   0.5*log(det(Ko   +
(sig.^2).*eye(length(yo)))) - length(yo)/2*log(2*pi)));
end

[m,index] = max(p);
l = l(index);
```

11. Thin Airfoil Theory

```
% Function created by Nolan Whiting on November 14, 2017 to calculate the
% lift coeffcient and moment coefficient for a multielement airfoil.

% Inputs:   alpha: angle of attack
%           delta: flap deflection
% Outputs:  CL_a:  2-D lift coefficient
%           CM_4:  moment coefficient about quarter chord

function [CL_a,CM_4] = ThinAirfoilTheory(alpha,delta)

% Results of function below fitted exactly to reduce computation time when
% sampling many different conditions.
CL_a = @(x,y) 0.1097.*x + 0.05345.*y + 0.964;
CM_4 = @(x,y) -9.337e-05.*y.^2 + -0.01048.*y + -0.1332;
CL_a = CL_a(alpha,delta);
CM_4 = CM_4(alpha,delta);

% Comment out below to reduce computation time
xs = @(x) 1.0773.*x + 0.0477;
xm = @(x) 0.1415.*x.^2 - 0.0051.*x + 0.0559;
xf = @(x) -2.3524.*x.^2 + 2.0981.*x - 0.3881;

x1 = linspace(-0.045776,0.0104092,1000);
x2 = linspace(0.0104092,0.4876,1000);
x3 = linspace(0.4876,0.6342,1000);

xs = @(x) 1.0773.*x + 0.0477;
xm = @(x) 0.1415.*x.^2 - 0.0051.*x + 0.0559;
xf = @(x) -2.3524.*x.^2 + 2.0981.*x - 0.3881;
xf = @(x) xf(x) + (xm(x2(end)) - xf(x3(1)));

% figure()
% plot(x1,xs(x1),'r--',x2,xm(x2),'r--',x3,xf(x3),'r--')
% axis equal

% Rotates airfoil geometry in accordance with flap deflection
Rflap = [cosd(delta-30),sind(delta-30);-sind(delta-30),cosd(delta-30)];
x3rot = Rflap*([x3 - x2(end);xf(x3)- xf(x2(end))]);

X_total = zeros(1,length(x1)+length(x2)+length(x3));
Z_total = zeros(1,length(x1)+length(x2)+length(x3));
for i = 1:length(x1)
    X_total(i) = x1(i);
    Z_total(i) = xs(x1(i));
end

for i = length(x1)+1:length(x2)+length(x1)
    X_total(i) = x2(i-length(x1));
    Z_total(i) = xm(x2(i - length(x1)));
end

for i = length(x2)+length(x1)+1:length(x2)+length(x1)+length(x3)
    X_total(i) = x3rot(1,i - length(x1) - length(x2))+ x2(end);
    Z_total(i) = x3rot(2,i - length(x1) - length(x2)) + xf(x2(end));
end
Xm = [X_total(1), X_total(end)];
Zm = [Z_total(1),Z_total(end)];
pchord = polyfit(Xm,Zm,1);
chord = @(x) pchord(1).*x + pchord(2);
```

```
% figure()
%  plot(x1,xs(x1),'r--',x2,xm(x2),'r--',x3rot(1,:)+x3(1),x3rot(2,:)  +  xf(x3(1)),'r--
',X_total,chord(X_total))
% axis equal

theta = atan((Zm(2) - Zm(1))/(Xm(2) - Xm(1)));

Rotate = [cos(theta),sin(theta);-sin(theta),cos(theta)];

New = Rotate*[X_total;Z_total];

% figure()
% plot(New(1,:),New(2,:))
% axis equal

% Fits the mean camber lines to the geometries of the slat, main body, and
% the rotated flap section.
p1 = polyfit(New(1,1:1000),New(2,1:1000),1);
xc1 = @(x) 0;
for i = 1:length(p1)
    xc1 =  @(x) p1(i).*x.^(length(p1) - i) + xc1(x);
end

dxc1 = xc1(1);

syms x
p2 = polyfit(New(1,1001:2000),New(2,1001:2000),2);

xc2 = 0;
for i = 1:length(p2)
    xc2 = p2(i).*x.^(length(p2) - i) + xc2;
end


p3 = polyfit(New(1,2001:3000),New(2,2001:3000),2);

xc3 = @(x) 0;
for i = 1:length(p3)
    xc3 = p3(i).*x.^(length(p3) - i) + xc3;
end

dxc2 =  diff(xc2);
dxc3 =  diff(xc3);


dxc2 = matlabFunction(dxc2);
dxc3 = matlabFunction(dxc3);



% figure()
%             h1              =              plot(New(1,1:1000),xc1(New(1,1:1000)),'k-
',New(1,1000:2000),xc2(New(1,1000:2000)),'k-
',New(1,2000:3000),xc3(New(1,2000:3000)),'k-');
% axis equal
% grid on
% set(h1,'LineWidth',1')
% xlabel('X')
% ylabel('Y')
% legend('Slat','Main','Flap')

c = New(1,end);
theta1 = real(acos(1-New(1,1)/c*2));
```

```
theta2 = acos(1-New(1,1000)/c*2);
theta3 = acos(1-New(1,2000)/c*2);
theta4 = acos(1-New(1,2001)/c*2);

xct1p = @(theta) dxc1;
xct2p = @(theta) dxc2(c/2*(1-cos(theta)));
xct3p = @(theta) dxc3(c/2*(1-cos(theta)));

thetas1 = linspace(theta1,theta2,1000);
thetas2 = linspace(theta2,theta3,1000);
thetas3 = linspace(theta3,pi,1000);

for i = 1:1000
    xct1(i) = xct1p(thetas1(i));
    xct2(i) = xct2p(thetas2(i));
    xct3(i) = xct3p(thetas3(i));
end

p1 = polyfit(thetas1,xct1,1);
p2 = polyfit(thetas2,xct2,2);
p3 = polyfit(thetas3,xct3,2);

xct1 = @(x) 0;
for i = 1:length(p1)
    xct1 =  @(x) p1(i).*x.^(length(p1) - i) + xct1(x);
end

xct2 = @(x) 0;
for i = 1:length(p2)
    xct2 =  @(x) p2(i).*x.^(length(p2) - i) + xct2(x);
end

xct3 = @(x) 0;
for i = 1:length(p3)
    xct3 =  @(x) p3(i).*x.^(length(p3) - i) + xct3(x);
end

A0 = alpha*pi/180 - 1/pi*(integral(xct1,theta1,theta2)+integral(xct2,theta2,theta3) +
integral(xct3,theta3,pi));
for n = 1:4
   nA = @(theta) cos(n.*theta);
   An1 = @(theta) nA(theta).*xct1(theta);
   An2 = @(theta) nA(theta).*xct2(theta);
   An3 = @(theta) nA(theta).*xct3(theta);
   A(n)      =      2/pi*(integral(An1,theta1,theta2)+integral(An2,theta2,theta3)      +
integral(An3,theta3,pi));
end

cl = 2*pi*(A0 + A(1)/2);
cm_4 = pi/4*(A(2)-A(1));
```

12. Biharmonic Fits

```
function [coeff_CFD,drag_CFD,momt_CFD] = Biharmonic_Fits()
% Modified by Nolan Whiting on Novemeber 29, 2017
% Used to create manufactured universe for lift and moment coefficient from
% values supplied from OVERFLOW2
% Note: Used with createFits.m
%
```

```
% Variable list:
%
%    num :: the number of terms in the curve fit
%
%    A :: the complete list of angles of attack (normalized against 16
%         degrees); the angles of attack used in FLUENT runs will be
%         repeated six times
%
%    M :: the complete list of flap deflections; entries are repeated multiple
%         times to correspond to the angles of attack and the relevant
%         force/moment coefficient
%
%    Cd :: the vector of drag coefficients
%    Cl :: the vector of lift coefficients
%    Cm :: the vector of moment coefficients, about the quarter-chord
%
%    Variables followed by "1" (e.g., "A1", "M1", "Cd1", etc.) provide
%    vectors for data mirrored about alpha = 0

%clear


% Angles of attack
A = [0
    5
    10
    15
    20
    25
    30
    0
    5
    10
    15
    20
    25
    30
    0
    5
    10
    15
    20
    25
    30
    0
    5
    10
    15
    20
    25
    30
    0
    5
    10
    15
    20
    25
    30
    0
    5
    10
    15
    20
```

```
    25
    30];

% Flap deflection
M = [0
    0
    0
    0
    0
    0
    10
    10
    10
    10
    10
    10
    10
    20
    20
    20
    20
    20
    20
    20
    30
    30
    30
    30
    30
    30
    30
    40
    40
    40
    40
    40
    40
    40
    50
    50
    50
    50
    50
    50
    50];

%CFD CL results
a6CM = [9.67E-02
0.861209929
1.625529528
2.330048323
2.970992088
3.545929909
3.754305601
0.849718213
1.672124267
2.397075415
3.041237354
3.618926048
4.090542316
3.730473995
1.696801901
```

```
2.414883137
3.068836689
3.647276402
4.122953415
4.402859211
3.446215153
2.357216835
3.017752409
3.608569622
4.110393524
4.454927921
4.382623672
2.822540283
2.195100784
2.688181639
3.283240557
4.310677052
4.506865025
4.122920036
2.518360615
2.101033211
2.770459175
3.06085372
3.697404385
4.455102921
3.987341166
2.519794464];


%CFD CD results
% a6CD = [0.003418
% 0.196245
% 0.318953
% 0.499222
% 0.669664
% 0.751143
% 0.779657
% 0.839623
% -0.15104
% 0.162557
% 0.275614
% 0.447387
% 0.606866
% 0.691184
% 0.72476
% 0.803934];

a6CD = [4.42E-02
3.58E-02
2.37E-02
2.37E-02
3.14E-02
4.33E-02
9.52E-02
3.93E-02
2.75E-02
2.10E-02
2.50E-02
3.77E-02
5.30E-02
0.190754756
3.12E-02
2.50E-02
```

```
2.23E-02
2.93E-02
4.58E-02
8.48E-02
0.274429947
2.80E-02
2.66E-02
2.55E-02
3.55E-02
5.45E-02
0.121813491
0.561693728
5.05E-02
5.35E-02
2.86E-02
4.14E-02
6.50E-02
0.136368901
0.832785308
0.104590036
0.200958923
8.70E-02
0.151530892
8.15E-02
0.174034193
0.704929411];


%CFD CM results
VarName6 = [-9.64E-02
-0.141375244
-0.139986277
-0.127025887
-0.106460832
-0.079083323
-0.050520688
-0.20972687
-0.233196065
-0.222127214
-0.198481426
-0.167124778
-0.126346767
-0.089858033
-0.318495214
-0.313386202
-0.289355189
-0.25444904
-0.209192559
-0.154766142
-0.129443154
-0.388552517
-0.3718265
-0.337901294
-0.291833729
-0.231361523
-0.157671034
-0.246891394
-0.344657093
-0.310642302
-0.273331165
-0.297701716
-0.224392593
-0.143508881
```

```
-0.299283534
-0.315284818
-0.339788765
-0.239035964
-0.236173034
-0.210181415
-0.136632383
-0.241458729];

[drag_CFD,lift_CFD,momt_CFD,coeff_CFD] = createFits(A, M, a6CM, a6CD, double(VarName6));

alfa = linspace(-5,50,100);
delta = linspace(-10,50,75);

for j = 1:length(delta)
    for i = 1:length(alfa)
        Coeff = coeff_CFD(alfa(i),delta(j));
        CL_fit(i,j) = Coeff(1);
        CM_fit(i,j) = Coeff(3);
    end
end
```

13. Create Fits (used with Biharmonic Fits)

```
function [drag,lift,momt,coeff] = createFits(A, M, Cd, Cl, Cm)
%CREATEFITS(A,M,CD,CL,CM)
%  Create fits.
%
%  Data for 'drag1' fit:
%      X Input : A
%      Y Input : M
%      Z Output: Cd
%  Data for 'lift1' fit:
%      X Input : A
%      Y Input : M
%      Z Output: Cl
%  Data for 'momt1' fit:
%      X Input : A
%      Y Input : M
%      Z Output: Cm
%  Output:
%      drag1: Curve fit for drag coefficient data
%      lift1: Curve fit for lift coefficient data
%      momt1: Curve fit for quarter-chord moment coefficient data

%% Fit: 'drag1'.
[xData, yData, zData] = prepareSurfaceData( A, M, Cd );

% Set up fittype and options.
ft = 'biharmonicinterp';
opts = fitoptions( ft );

% Fit model to data.
[drag1, ~] = fit( [xData, yData], zData, ft, opts );

%% Fit: 'lift1'.
[xData, yData, zData] = prepareSurfaceData( A, M, Cl );

% Set up fittype and options.
ft = 'biharmonicinterp';
opts = fitoptions( ft );
```

```
% Fit model to data.
[lift1, ~] = fit( [xData, yData], zData, ft, opts );

%% Fit: 'momt1'.
[xData, yData, zData] = prepareSurfaceData( A, M, Cm );

% Set up fittype and options.
ft = 'biharmonicinterp';
opts = fitoptions( ft );

% Fit model to data.
[momt1, ~] = fit( [xData, yData], zData, ft, opts );


%% Outputs: Returns functions to be called with un-normalized inputs.
 drag=@(x,y)(drag1(x,y));
 lift=@(x,y)(lift1(x,y));
 momt=@(x,y)(momt1(x,y));
 coeff=@(x,y)([drag1(x,y) lift1(x,y) momt1(x,y)]);

%     function [lift] = lift2(x,y);
%
%         if (x == 0),
%
%             lift = 0.0;
%         else
%
%             lift = lift1(x/16,y);
%
%         end
%
%     end
%
%     function [momt] = momt2(x,y);
%
%         if (x == 0),
%
%             momt = 0.0;
%
%         else
%
%             momt = momt1(x/16,y);
%
%         end
%
%     end
%
% lift=@(x,y)(lift2(x,y));
% momt=@(x,y)(momt2(x,y));
% coeff=@(x,y)([drag1(x/16,y) lift(x,y) momt(x,y)]);
end
```

14. Bayesian Updating (1-D)

```
% Nolan Whiting
% Gaussian Process
% March 13, 2018
```

```
clear
clc
close all

[coeff_CFD,lift_CFD,momt_CFD] = Biharmonic_Fits();
global var L

T = 1;
N = 1000; % Samples of TAT/LLT
S = 4; % CFD samples
alfa = [1,6,10,14,20,25,29,34,42];
delta = 20;

for ii = 1:length(alfa)
    alpha = alfa(ii);

    alpha_TAT = lhsnorm(alpha,0.5,N);
    delta_TAT = lhsnorm(delta,1,N); % No uncertainty in flap deflection.
    for i = 1:length(alpha_TAT)
        [CL_a(i),CM_4(i)] = ThinAirfoilTheory(alpha_TAT(i),delta_TAT(i));
    end

    for i = 1:T
        alpha_CFD = alpha + 0.5.*randn(1,S);
        delta_CFD = delta + randn(1,S);

        CL_CFD = lift_CFD(alpha_CFD,delta_CFD);
        CM_CFD = momt_CFD(alpha_CFD,delta_CFD);
        CL = sort(CL_CFD);
        CM = sort(CM_CFD);

        CL = CL + sort(0.025*randn(1,S)).*CL;
        CM = CM + sort(0.025*randn(1,S)).*CL;

        CL_mn = mean(CL);
        CM_mn = mean(CM);

        sig_CL(ii) = std((mean(CL_a) - CL));
        sig_CM(ii) = std((mean(CM_4) - CM));
        conf_inf_CL(ii) = 4.303*sig_CL(ii)/sqrt(S);
        conf_inf_CM(ii) = 4.303*sig_CM(ii)/sqrt(S);
        diff_CL(ii) = -(CL_mn - mean(CL_a));
        diff_CM(ii) = -(CM_mn - mean(CM_4));
    end
end

n = 100;
xp = -5:0.25:50;
x = alfa;
yo = diff_CL';
sig = sig_CL;
[mean_CL,conf_u_CL,conf_l_CL] = GP_pt2(x,yo,xp,sig,n,1,conf_inf_CL);


yo = diff_CM';
sig = sig_CM;
[mean_CM,conf_u_CM,conf_l_CM] = GP_pt2(x,yo,xp,sig,n,2,conf_inf_CM);




CL_new = CL_a - mean_CL;
```

```
CM_new = CM_4 - mean_CM;
conf_u_new = CL_a - conf_u_CL;
conf_l_new = CL_a - conf_l_CL;


figure()
errorbar(x,CL_a(flag)-diff_CL,conf_inf_CL,conf_inf_CL,'or');
hold on
h1       =       plot(xp,CL_original,'-b',xp,CL_a,'-g',xp,CL_new,'-r',xp,conf_u_new,'--
k',xp,conf_l_new,'--k');
set(h1,'LineWidth',1')
set(gca,'FontSize',12)
hold on
legend('Observations','Experiment','Simulation','Updated Model','95% CI')
xlabel('\alpha (\circ)')
ylabel('CL')
xlim([xp(1),xp(end)])

%Conservativeness
%xpred = [6,13,21,28,32,34,37,42];
xpred = alpha_pred;
count = 1;
for i = 1:length(xpred)
    while xp(count) < xpred(i)
        count = count + 1;
    end
    flag2(i) = count - 1;
end

k = flag2;
conserv_sum = 0;
for i = 1:length(k)
    if CL_original(k(i)) >= conf_u_new(k(i)) && CL_original(k(i)) <= conf_l_new(k(i))
        conserv = 1;
    else
        conserv = 0;
    end
    conserv_sum = conserv_sum + conserv;
end

conserv_CL = conserv_sum/length(k);

CL = CL_new(flag2);
conf_CL = conf_u_new(flag2) - CL;

conf_u_new = CM_4 - conf_u_CM;
conf_l_new = CM_4 - conf_l_CM;

figure()
errorbar(x,CM_4(flag)-yo',conf_inf_CM,conf_inf_CM,'or')
hold on
h2       =       plot(xp,CM_original,'-b',xp,CM_4,'-g',xp,CM_new,'-r',xp,conf_u_new,'--
k',xp,conf_l_new,'--k');
set(h2,'LineWidth',1')
set(gca,'FontSize',12)
legend('Observations','Experiment','Simulation','Updated Model','95% CI')
ylabel('CM')
xlabel('\alpha (\circ)')
xlim([xp(1),xp(end)])
%
% %Conservativeness

conserv_sum = 0;
```

```
for i = 1:length(k)
    if CM_original(k(i)) >= conf_u_new(k(i)) && CM_original(k(i)) <= conf_l_new(k(i))
        conserv = 1;
    else
        conserv = 0;
    end
    conserv_sum = conserv_sum + conserv;
end

conserv_CM = conserv_sum/length(k);


CM = CM_new(flag2);
conf_CM = conf_u_new(flag2) - CM;
```

15. Covariance Matrices for Bayesian Updating (1-D)

```
function [mean_gp,conf_u,conf_l] = GP_pt2(x,yo,xp,sig,num,variable,conf_inf)

global var L
sig2 = std(yo);
% [l] = l_parameter(yo,x,variable,sig)
if variable == 1
    l = 4;
elseif variable == 2
    l = 0.75;
    sig2 = 1;
end

for j = 1:length(x)
    for i = 1:length(xp)
        K(j,i) = sig2^2.*exp(-abs((x(j)-xp(i)).^2)/2/l^2);
    end
end

for j = 1:length(x)
    for i = 1:length(x)
        Ko(i,j) = sig2^2*exp(-(abs(x(j)-x(i)).^2)/2/l^2);
    end
end

for j = 1:length(xp)
    for i = 1:length(xp)
        Kp(i,j) = sig2^2*exp(-(abs(xp(j)-xp(i)).^2)/2/l^2);
    end
end

mn = K'*inv(Ko + (sig.^2).*eye(length(yo)))*yo;
var = Kp - K'*inv(Ko + (sig.^2).*eye(length(yo)))*K;

L = chol(var + 0.000001*eye(length(xp)));


n = length(xp);
xvec = zeros(num,n);
max_abs = 0;
figure()
for ii = 1:num
    xvec(ii,:) = mn + L'*randn(length(xp),1);
    maxy = max(abs(xvec(ii,:)));
    if maxy > max_abs
        max_abs = maxy;
    end
    plot(xp,xvec(ii,:),'-b')
```

```
    hold on
end
xlabel('\alpha (\circ)')
ylabel('\delta')
legend('Posterior GP Realizations')
set(gca,'FontSize',12)
xlim([xp(1),xp(end)]);
ylim([-max_abs,max_abs])
hold off
for ii = 1:length(xp)
    mean_gp(ii) = mean(xvec(:,ii));
    conf_u(ii) = mean_gp(ii) + 1.96*std(xvec(:,ii));
    conf_l(ii) = mean_gp(ii) - 1.96*std(xvec(:,ii));
end


figure()
h2 = plot(x,yo,'or',xp,mean_gp,xp,conf_u,'--k',xp,conf_l,'--k')
hold on
%errorbar(x,yo,conf_inf,conf_inf,'or')
xlabel('\alpha (\circ)')
ylabel('\delta')
xlim([xp(1),xp(end)])
ylim([-max_abs,max_abs])
set(h2,'LineWidth',1')
set(gca,'FontSize',12)
legend('Observations','Mean','95% Confidence Interval')
```