Research Article

A Comprehensive Probabilistic Framework to Learn Air Data from Surface Pressure Measurements

Ankur Srivastava¹ and Andrew J. Meade²

¹Massachusetts Institute of Technology, Cambridge, MA 02139, USA

²Rice University, Houston, TX 77005, USA

Correspondence should be addressed to Ankur Srivastava; ankur.rice@gmail.com

Received 11 May 2015; Revised 13 August 2015; Accepted 23 August 2015

Academic Editor: Christopher J. Damaren

Copyright © 2015 A. Srivastava and A. J. Meade. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Use of probabilistic techniques has been demonstrated to learn air data parameters from surface pressure measurements. Integration of numerical models with wind tunnel data and sequential experiment design of wind tunnel runs has been demonstrated in the calibration of a flush air data sensing anemometer system. Development and implementation of a metamodeling method, Sequential Function Approximation (SFA), are presented which lies at the core of the discussed probabilistic framework. SFA is presented as a tool capable of nonlinear statistical inference, uncertainty reduction by fusion of data with physical models of variable fidelity, and sequential experiment design. This work presents the development and application of these tools in the calibration of FADS for a Runway Assisted Landing Site (RALS) control tower. However, the multidisciplinary nature of this work is general in nature and is potentially applicable to a variety of mechanical and aerospace engineering problems.

1. Introduction

INTRUSIVE aircraft anemometer booms have long been successfully used to measure air data parameters for subsonic and supersonic flight. Rapid development in aerospace and naval technology has continuously demanded smaller, less intrusive, and more accurate air data systems. Accuracy of probe based air data systems suffers at high angles of attack or when dynamic maneuvering is required. These same conditions can lead to degraded flying handling qualities [1]. Errors due to vibration, poor alignment, and physical damage during operation or maintenance can also pose significant limitations on the use of protruding booms. Probe based air data systems cannot be used with stealthy aircrafts because they increase the total radar cross-section of the body which can potentially jeopardize the mission and risk human lives. Hypersonic flight regimes are another area where probe based air data systems cannot be used because the vehicle nose reaches extremely high temperatures that might melt the protruding boom. Also, such flow protruding booms cannot be used with research aircrafts without disturbing the airflow or the boundary layer. Finally, air data booms are too heavy and costly to use with unmanned micro air

vehicles (MAVs) [2]. Reference [2] mentions that an 80% decrease in instrumentation weight and a 97% decrease in instrumentation cost can be gained by eliminating probe based air data systems from MAVs.

In order to overcome the above mentioned drawbacks, flush air data sensing (FADS) systems were developed. A popular flush air data system is a series of pressure taps mounted on a vehicle periphery that can be used to derive airflow parameters. FADS systems were first developed during the X-15 program. A hemispherical nose with pressure sensors was installed on this hypersonic aircraft to measure stagnation pressure and wind direction during reentry [3]. Further research on FADS for hypersonic vehicles was conducted during the Space Shuttle program [4]. After enjoying success on hypersonic flights the compatibility of FADS was tested on supersonic and subsonic flight regimes. The authors of [5] developed and flight-tested a flush air data sensing system for the F-18 Systems Research Aircraft (Figures 1 and 2). The authors concluded that the developed nonintrusive technique was clearly superior to the probe based air data systems. They showed that FADS were unaffected by dynamic flight maneuvers and they performed well in high angle of attack flights. FADS systems are also unaffected by vibration and

FIGURE 1: Flush air data system installation at the nose of the F-18 Systems Research Aircraft [1].

FIGURE 2: FADS Ports on Space Shuttle Columbia and F-18 [21].

icing effects and are less prone to damage during operation and maintenance. They are an ideal choice for stealth vessels because they are self-sustained and give no additional radar cross-sectional area to the vessel signature.

Other nonintrusive air data systems include flush mounted optical air data systems and flush mounted heat films. The optical air data systems transmit lasers or acoustic signals generated by electromechanical transducers or loud speakers through a fluid medium to one or multiple receivers and measure the travel time to infer air data information. Even though these systems are nonintrusive, they are not self-sustained and are expensive and heavy to use [6].

1.1. Challenges in Estimating Air Data of a Flush Air Data Sensing System. As previously mentioned, FADS systems infer the air data parameters from pressure measurements taken with an array of ports that are flush to the surface of the aircraft. However, because the locations of the pressure measurements are on the outer surface of the aircraft on geometrically simple components (e.g., hemispherical or conical nose), properties of the local flow fields like compressibility and flow separation can drastically affect these devices. Unlike aircraft, the FADS for bluff bodies like buildings must be placed

FIGURE 3: Two-dimensional flow about a right circular cylinder.

completely around the structure and can routinely experience separated and unsteady cross-flow. Finally, to compete with conventional air data systems, the FADS must be able to estimate wind speeds (V_∞) over a range of 40 to 120 fps, ± 3.4 fps (i.e., ± 2 knots), and wind directions (β) ranging from 0 to 360 degrees, ± 2 degrees. For more details on assessment of conventional air data systems please refer to [7].

Using the incompressible flow about a right circular cylinder as an example of a bluff body (Figure 3), the problem of estimating relative wind speed and direction from the surface pressure at position θ can be framed as either a forward or inverse mapping problem. That is,

the forward problem

$$P = F(\rho_\infty, V_\infty, \theta, \beta); \quad (1)$$

the inverse problem

$$\begin{aligned} V_\infty &= G(P, \rho_\infty, \theta, \beta), \\ \beta &= G(P, \rho_\infty, V_\infty, \theta). \end{aligned} \quad (2)$$

For simple geometries, closed-form semiempirical equations can be developed and used to solve the forward problem of estimating the pressure coefficient, C_p , given the freestream wind speed and direction [8]. Panel methods [9] and other popular computational fluid dynamic (CFD) based approaches solve the forward problem on more complex geometries given the freestream wind speed and direction. However, this approach requires guessing the freestream velocity and solving for the pressure distribution until it matches the measured values. To solve the inverse problem, look-up tables from experimental fluid dynamics (EFD) can be constructed for the mapping function G . Unfortunately, look-up tables suffer from a number of drawbacks when their real-time use is considered including the inability to handle high-dimensional inputs, noisy data, and nonlinear interpolation. It is proposed that machine learning methods, specifically Artificial Neural Networks (ANNs), be explored to solve the inverse problem. In addition, this paper will demonstrate how a particular form of ANN can also be used to determine the best locations for sensor placement,

to maximize available FADS data by seamlessly combining CFD and EFD outputs, and to help guide physical experiments thereby minimizing the time needed in testing facilities.

1.2. Intersection of Statistical Learning and Fluid Dynamics.

In general, mechanical and aerospace engineering systems present a number of challenging problems especially in the field of fluid dynamics. Understanding pressure and velocity distributions of the fluid flow is essential to the design and control of complex multicomponent interacting systems. Experimental and computational methods in fluid dynamics have long been successfully used to model such engineering systems. However, intensive research in both fields spanning over several decades has highlighted several shortcomings [10]. Over the last few decades, as complexity of engineering systems increased, it became a necessity rather than academic curiosity to adopt interdisciplinary approaches to study a system. Intelligent data understanding tools like Artificial Neural Networks (ANNs) [11] and kernel methods from machine learning and statistics have enjoyed increased popularity in mechanical and aerospace engineering problems. However, the following issues remain.

1.2.1. Formalization of the Final Network. A typical 3-layer ANN involves several parameters to be defined by the user including (a) number of neurons in the hidden layer, (b) initial weight matrix, (c) parameters of the basis function, and (d) values of the learning rates. Ad hoc procedures exist for choosing each of these parameters. However, lack of a principled approach to network construction generally leads to an overparameterized network. To optimize the structure of the network exhaustive grid search or cross-validation approaches are often needed. To attack this problem we focus on greedy scattered data approximation approaches [12]. In these approaches the user needs to load only the training and test sets and the algorithm solves for the required parameters in a self-adaptive greedy fashion. In other words, greedy scattered data approximation approaches provide a formal way to construct networks in an optimized manner.

1.2.2. Choice of Training Data. It is often unclear how many training points would be sufficient for an acceptable generalization error and how these training data should be chosen. Traditionally, network training and testing is done only after all the data is collected. This is called passive learning in the machine learning literature, where the learning algorithm does not take part in the data collection procedure. Collecting data without taking into account the input-output functional relationship like the traditional Latin Hypercube Design [13] methods can result in choice of data that add little to no information to training of the network and can even hurt the generalization ability of the training method. This is especially a problem when data collection is costly, like flight tests or when the input domain is excessively large to sample. Active learning [14] methods are popular in the machine learning literature where the learning algorithm takes part in the data collection procedure and new input configurations

FIGURE 4: (a) Wind tunnel model of the RALS tower [7] and (b) actual image of the RALS tower.

are sampled such that they maximize some information gain criterion in a principled manner. These sampling procedures are essentially sequential in nature and not only help the engineers to accelerate through the test matrix, but also allow the learning algorithm to have better generalization capability.

1.2.3. Data-Model Fusion: Fusing Sparse High Fidelity Wind Tunnel Data with Low Fidelity CFD Trends. ANNs provide a powerful tool to interpolate and extrapolate experimental data. However, it is beneficial to have a framework that can integrate experimental data with models of variable fidelity. Regularization approaches provide us with such a framework where it is possible to include physical knowledge of the system in the form of numerical simulations to be included as a priori information in the training of the network [15]. Such a data assimilation procedure can be seen as approximating noisy and scattered experimental data with physics based smoothness. It can also look as improving low fidelity numerical simulations with more accurate experimental data. This tool can result in acceptable flow field solutions by fusing fewer experimental data with possibly lower fidelity CFD data thereby saving precious resources.

For the purpose of solving the inverse problem of FADS calibration discussed in Section 1 and similar engineering problems, a multidimensional learning tool should address the hyperparameter selection problem previously mentioned, operate as easily on high-dimensional data as it does on low-dimensional ones, and provide the user with an assurance that the tool will give its best performance on an unseen test set. The tool should also require a minimum of storage and require as little user interaction as possible. To this end, we use Sequential Function Approximation (SFA), an adaptive tool, developed earlier for solving regression and classification problems [16] with little user interaction to determine hyperparameters.

The first objective in this work is to develop a computational surrogate that can predict wind speed and yaw angle from surface pressure measurements. The surrogate should not have wind speed prediction errors greater than ± 3.4 fps and yaw angle prediction errors greater than ± 2 degrees. The second objective is to investigate how data-model fusion can help to better represent the coefficient of pressure distribution across the bluff body surface. The third objective is to develop a general intelligent sampling strategy with active learning and SFA on regression problems.

Section 2 discusses estimation of air data parameters from surface pressure measurements in more detail. Section 2.2 presents freestream wind speed and direction estimation techniques and the results on the current problem. Section 3 briefly discusses the role of regularization techniques in fusing wind tunnel data with numerical models of variable fidelity. Section 4 begins by introducing the fields of active learning and optimum experiment design. Sections 4.1 and 4.2 discuss a G-optimal design procedure with SFA and present its implementation issues application on a simulated regression problem. Finally, Section 4.3 presents the application of this technique on the FADS problem.

2. Freestream Wind Speed and Direction Estimation

The Runway Arrested Landing Site (RALS) control tower is located at the Naval Air Warfare Center Lakehurst, New Jersey. Wind tunnel tests were conducted on a 1/72 scale model of the RALS control tower [7]. One pressure sensor was mounted on each face of the control tower as shown in Figure 4. A four-hole Cobra probe was used to measure the freestream wind speed and direction. The FADS system was tested on a variety of wind speeds ranging from 40 fps to 120 fps approaching the model from all 360 degrees of yaw. Variation of the wind incidence in three dimensions will be a

FIGURE 5: Surface pressure (psi) variation versus the yaw angle for different wind speeds.

part of the future work. In addition, the scope of this work is to develop a computational framework to predict wind speed and direction from pressure measurements and not to conduct the wind tunnel experiments. The experimental setup is mentioned here briefly for completeness.

Borrowing from flight mechanics literature and restricting our problem to two dimensions, β is defined as the yaw angle and is measured in the anticlockwise sense with respect to the centerline of the model. So winds approaching the south port have $\beta = 0$ degrees, for the east port $\beta = 90$ degrees, for north port $\beta = 180$ degrees, and for the west port $\beta = 270$ degrees. In Figure 5 shown surface pressure is plotted for each pressure port. The pressure port on the south face bears positive pressure values when wind is incident head-on at the pressure port and bears negative values for winds hitting the north face. Similarly the pressure port on the west face bears positive pressure values when the yaw angle is positive and bears negative values for negative yaw values. Also evident from Figure 5 is that the pressure ports do not show much variation when they are on the leeward side.

2.1. Forward and Inverse Problems. Computational algorithms like neural networks that learn from data can be very effective in solving inverse problems and bear potential advantages in the construction of the FADS mapping function G (Section 1.1) compared to look-up tables. Examples of neural networks advantages include high-dimensional mapping, smoother nonlinear control, intelligent empirical learning, and fewer memory requirements [17–20]. For speed of evaluation, reduced complexity, and the potential for graceful performance degradation with the failure of pressure sensors, the inverse problem is solved using only two sets of neural networks with pressures read from all surface sensors,

$$\begin{aligned} q &= G1(P_1, \dots, P_d), \\ \beta &= G2(P_1, \dots, P_d, V_\infty). \end{aligned} \quad (3)$$

2.1.1. Dynamic Pressure. Even though our problem involves turbulence and cross-flow about nontrivial geometries, the existence of a functional relationship between P and V_∞ and

FIGURE 6: Dynamic pressure hypercone represented as a function of (a) two pressure sensors and (b) one pressure sensor.

between C_p and β can be safely assumed. The wind speed was predicted using only the surface pressure data while the wind direction was predicted using the pressure coefficient data derived from measured pressure and predicted wind speed values. It is noted that this coupling of networks put the burden of high accuracy on the wind speed predictor. Freestream air density was kept constant in the current problem which makes prediction of freestream wind speed equivalent to dynamic pressure. Sequential Function Approximation was used to construct one RBF network to model G1 and the corresponding wind speed surrogate is represented by

$$\hat{q}_{pre} = \sum_{i=1}^n c_i \exp(\ln[\lambda_i] * (\vec{p}_{test} - \vec{p}_i^*) \cdot (\vec{p}_{test} - \vec{p}_i^*)) + b_i. \quad (4)$$

The hypersurface for the current dynamic pressure prediction problem can be imagined as a hypercone with surface pressure as input dimensions and dynamic pressure as output. Dynamic pressure increases linearly with surface pressure given a yaw angle β for all pressure sensor positions, θ . So each wind tunnel sweep at a constant speed would yield data points lying at the contours of the hypercone. Also, the slope of the hypercone corresponds to the coefficient of pressure at a given value of the yaw angle β .

Figures 6(a) and 6(b) show the aforementioned hypercone in two and one dimension, respectively. Looking at Figure 6(b), it is evident that the span of the cone includes all pressure measurements of the sensor at $\theta = 0$ degrees at all wind speeds and yaw angles. The right edge of the cone corresponds to the case when the sensor faces directly the wind or $\beta = \theta$. As one moves from the right to the left edge of the cone the separation at the pressure sensor increases. A RBF network constructed by SFA can suitably approximate this hypercone; however it is possible that a better approach might exist for this approximation. This will constitute a part of the future work.

FIGURE 7: Nonuniqueness of the yaw angle prediction problem at 120 fps with one pressure sensor.

2.1.2. Wind Direction. Once the wind speed predictor was available, the test surface pressure values were divided by the corresponding predicted dynamic pressure to obtain the test coefficient of pressure values. Several ways exist to predict the yaw angle β . One straightforward way is to construct one RBF network for G2 given by (5), where

$$\hat{\beta}_{pre} = \sum_{i=1}^n c_i \exp(\ln[\lambda_i] * (\vec{c}_{p,test} - \vec{c}_{p,i}^*) \cdot (\vec{c}_{p,test} - \vec{c}_{p,i}^*)) + b_i. \quad (5)$$

However, unlike dynamic pressure, yaw angle prediction is a strongly nonunique problem.

Figure 7 shows the variation of the yaw angle with bow side coefficient of pressure at 120 fps. It is evident from Figure 7 that at least two solutions for β exist for most values

of C_p , and except when the pressure sensor faces separated flow. Here it is important to elaborate on the meaning of nonuniqueness when learning yaw angle as a function of pressure measurements. If we try to fit a metamodel for yaw angle as a function of coefficient of pressure, then from Figure 7 we realize that, for any given coefficient of pressure, say zero, there are at least two possible yaw angles that can yield coefficient of pressure of value zero on the wall. Similarly for a coefficient of pressure value of -0.5 , there are more than 2 values of yaw angles that can result in coefficient of pressure of value -0.5 . Since building metamodels for yaw angle as a function of coefficient of pressure is nonunique we try to build metamodels for the forward problem as described below. It is important to mention here that as we add more sensors the problem becomes less nonunique and we have more information to predict the yaw angle. In this problem four sensors are not enough to build accurate metamodels as shown in (5), motivating development of method shown below. It will be a very interesting problem to determine the minimum number of sensors and their location to maximize prediction accuracy and will constitute part of future work. A simple forward problem approach was developed to estimate the yaw angle which is discussed next.

In this approach, the forward problem of approximating C_p as a function of the yaw angle β is addressed. It is well known from fluid mechanics that in external flow C_p is a function of both β and the surface pressure sensor position, θ . Constructing a surrogate model of C_p as a function of both β and θ is equivalent to approximating C_p for each pressure sensor as a function of just the yaw angle β . Even though this increases the memory requirements, it yields faster and more accurate models for C_p . Once a network has been constructed to represent each sensor, the objective function in (6) can be minimized to predict the yaw angle for a vector of test pressure coefficients:

$$\hat{\beta}_{\text{pre}} = \min_{\beta} \sum_{i=1}^{NS} \left(\frac{p_{\text{test}}(i)}{\hat{q}_{\text{pre}}} - \hat{c}_{p,i}(\beta) \right)^2. \quad (6)$$

Here NS is the number of pressure sensors, $\hat{c}_{p,i}$ represents the approximated C_p for the i th pressure sensor, p_{test} is the test vector of pressure values, and $\hat{q}_{\text{pre}}$ is the predicted dynamic pressure for p_{test} . It should be noted here that to predict wind direction, the predicted dynamic pressure is used and not the preknown dynamic pressure. A simple line search for β would suffice for the above minimization problem. However, searching for β over the entire range of $[-180, 180]$ degrees would be wasteful, so β could be searched over $[\delta - \varepsilon, \delta + \varepsilon]$, where δ is the pressure sensor position that bears the maximum surface pressure value and ε is a value of the yaw angle chosen by the user so that range for the line search is sufficiently large. Because L_2 norm is the most resilient to random noise it was chosen to express the differences between the test C_p and predicted C_p . Also, (6) allows the user to put weights on the contribution of each pressure sensor. Different weights could be helpful in situations where there is a combination of high and low fidelity sensors, or in the case of sensor damage or local flow field disturbance.

Besides ease of learning and better redundancy to noise this wind direction estimation approach uses all the available sensors for prediction instead of just those in a quadrant. The above mentioned algorithms have also been tested for a surface vessel and experimentally verified at the Fluid Mechanics Lab at the NASA Ames Research Center [16, 21]. Besides learning the air data parameters, this work extends to sequential experiment design and data-model fusion for the current problem.

2.2. Results. In this section wind speed and direction estimation accuracies are presented. Results for the RALS tower problem are presented using the estimation approaches discussed above. The SFA algorithm was implemented using the MATLAB programming environment on a Windows-configured PC with a Pentium 4 2.66 GHz processor and 1.0 GB of RAM.

Wind tunnel data for the RALS tower was available at speeds ranging from 40 fps to 120 fps as wind direction was varied from -180 to $+180$ degrees at increments of 2 degrees. All available data points were used in the test set, while a training set was created by taking data present at increments of 4 degrees at each wind speed. It should be noted here that, by this choice of training data, it consists only about half of the number of points in the test set. Once the training and test sets were constructed one network was constructed to predict dynamic pressure according to (4). The tolerance parameter was kept close to zero at a value of $1e-6$ so as to minimize any errors in the approximation of C_p . Figure 8(a) shows the prediction errors if SFA was used to construct one RBF network for wind speed prediction as a function of surface pressure. However, there are several errors larger than the tolerance of 3.4 fps at 120 fps and on a closer look one realizes that most of the errors occur at yaw angles in the vicinity of 0, 90, 180, and -90 degrees. One possible explanation for the wind speed errors displaying a bias at the highest wind speed is shown in Figure 8(b). In this problem, a hypercone is approximated with radial basis functions (RBFs). RBFs model the smooth slopes of the hypercone; however they do not approximate the flat top of the hypercone that exists due to the wind speeds of the highest magnitude.

Next we present a heuristic which could be used to reduce the errors shown in Figure 9(a). As mentioned before most of the errors occur in the vicinity of the 0, 90, 180, and -90 degrees. For the following ranges of the yaw angle $|\beta| \leq 25^\circ$, $|\beta - 90^\circ| \leq 25^\circ$, $|\beta| \geq 155^\circ$, and $|\beta + 90^\circ| \leq 25^\circ$, the corresponding pressure sensors face a head-on freestream wind and bear a positive pressure value and the remaining sensors bear a negative pressure value. This a priori information was used in conjunction with the dynamic pressure model to improve prediction in the following manner. Training points lying in the above mentioned range were set aside before starting to construct (4). During testing, if a test point is such that only one sensor bears positive pressure value, the dynamic pressure of that test point was predicted using a simple nearest neighbor search from the training points that were set aside. Prediction of dynamic pressure is coupled with coefficient of pressure or the yaw angle. If β of a test point is known a priori, linear dependence of dynamic pressure with surface pressure

FIGURE 8: (a) Wind speed prediction on RALS tower data using just one network for wind speed given by (4). (b) Example surrogate of the wind speed hypercone in 2 dimensions.

FIGURE 9: (a) Logarithm of discrete inner product norm of residual error. (b) Wind speed prediction errors on the RALS tower data when training was conducted on data at every 4-degree increments of yaw angle.

is easily used to predict the dynamic pressure of a test point. With the help of this heuristic one can approximately deduce β and use linear dependence of dynamic and static pressure to predict dynamic pressure and thereby wind speed of a test point.

Figures 9(a) and 9(b) show convergence of residual error and wind speed prediction errors on the test set, respectively. Blue dots show errors below the acceptable error tolerance of

3.4 fps. The maximum error of 25 fps shown in Figure 8(a) was reduced to less than 2 fps with the help of the nearest neighbor heuristic.

Once the wind speed predictor was in place, the predicted dynamic pressure values were used to obtain test coefficient of pressure values. The resulting test C_p values were then input to the forward approach discussed in the previous section. To implement the forward problem approach first four models

of pressure coefficient were constructed as a function of the yaw angle. Figures 10(a) and 10(b) show the residual error convergence and approximation of C_p for each pressure sensor, respectively. Since sufficient noise-free training data was available, tolerance was kept low and so the C_p models interpolate the wind tunnel data accurately.

Ability of SFA to handle noise and sparsity in training data will be covered in future work. Once the C_p models have been constructed the objective function shown in (6) was minimized by a line search to predict the yaw angle for each test point. Figure 11(a) shows an example of the logarithm of the objective function for a test point with a yaw angle of zero degrees.

Figure 11(b) shows yaw angle prediction errors on the RALS tower data using the forward problem approach.

3. Data-Model Fusion: Global Scaling via Surrogate Modeling

Estimating freestream wind speed and direction from surface pressure measurements is an ill-posed problem. The ill-posedness is induced due to the nonuniqueness of the problem which is worsened in the presence of noise in the pressure data. The inverse problem addressed in this paper is to reconstruct a complete pressure signal given sparse, noisy, and incomplete experimental data. This work develops an alternate method of fusing experimental and computational data to approximate pressure signals to estimate freestream wind speed and direction. The approach uses SFA as the inverse modeling tool and applies a simplified Tikhonov-related regularization scheme to correct for the original data error. The purpose of this section is to introduce a fusing approach using SFA that maximizes the use of experimental data with the help of CFD data in approximating a smooth, continuous, and accurate pressure distribution. The fusing approach first involves calculating the error function of the CFD and experimental data defined by the following equation:

$$e(\xi_i) = u^{\text{CFD}}(\xi_i) - u^{\text{EXP}}(\xi_i), \quad (7)$$

for $i = 1, \dots, s$, where s is the number of training data sets. The error vector, e , is then used to train the SFA network to a predetermined tolerance, τ . The resulting error surface, $e(\xi)$, will naturally involve some scatter directly related to the experimental data noise. Training the network to the given tolerance allows the SFA to regulate the noisy experimental data with a priori CFD information. Assuming the u^{CFD} surface is known, then the error surface approximation, e^{SFA} , can be subtracted from the $u^{\text{CFD}}(\xi)$ data to give the approximation in

$$u^{\text{SFA}} = u^{\text{CFD}} - e^{\text{SFA}}. \quad (8)$$

The τ value can be regarded as the regularization parameter and controls how well the approximations fit the experimental or CFD data. A very high tolerance value allows the training process to end prematurely with very few network units. As a result, the network “underlearns” the training data

and the majority of the approximations reach a value of zero. For data points with an error value of zero, (8) shows that the approximation value will reproduce the CFD data. On the other hand, a very small tolerance value will force the network to use too many network units to reach the smallest possible tolerance. In this case, the network “overlearns” the training data and will fit even the experimental noise in the error surface. As a result, the approximations will reproduce the experimental data. The user must carefully choose the tolerance value to best fit the experimental data using the CFD information based on the noise in the data. Similar approaches have been developed in [22–25].

In external flow past bluff bodies, coefficient of pressure is a function of the yaw angle β and local pressure sensor position θ . In the RALS tower problem wind tunnel data is available at only four values of $\theta = [-90, 0, 90, 180]$ degrees shown in Figure 12. A complete representation of C_p as a function of β and θ is necessary because it can possibly inform the experimentalist what locations should be chosen for pressure sensor installation to improve wind direction prediction accuracy.

Numerical solutions, however, provide full coefficient of pressure distribution as a function of yaw angle and pressure sensor position. The data-model fusion technique discussed here could similarly be applied to correct low fidelity C_p solutions even at values of θ where no wind tunnel data is available. This could be done simply by calculating the differences between the wind tunnel and CFD solutions at $\theta = [-90, 0, 90, 180]$ degrees. Once the differences have been calculated, it can be treated as a function approximation problem by SFA with β and θ as inputs and C_p as output. Once a surrogate to the hypersurface of differences has been created, it could be added to the CFD C_p surface to obtain a C_p distribution shown in Figure 13. The numerical solutions were extracted at 56 values of θ for each yaw angle.

4. Sequential Experiment Design

In the previous sections, passive learning techniques were discussed for the FADS problem, where a learning algorithm, like SFA, acts only after the training data set has been collected. The learning algorithm does not take part in the data collection procedure. Traditionally, popular space filling experiment design strategies like Latin hypercube sampling (LHS) are used to plan the data collection procedure. Use of LHS is popular with Monte Carlo techniques to estimate uncertainty in computer models. It has been proven that fewer samples are required with LHS compared to random sampling when estimating the output with a given variance [13]. Latin hypercube sampling is a simple and effective space filling sampling technique which does not need or assume the input-output functional relationship. Even though LHS is better than random sampling, a significant improvement is possible if the functional form of the input-output relationship is assumed and data are sampled from those regions where the output variance of the assumed input-output function is maximum. It is thought that advances in experiment design strategies will not only benefit the training of FADS but the fields of EFD and CFD in general. Suboptimal designs

FIGURE 10: (a) Residual error convergence and (b) C_p prediction by a model of each pressure sensor. Blue dots represent wind tunnel data and the red line represents approximation by SFA as a function of the yaw angle. Clockwise from top plots shows C_p model of pressure sensor at 0, 90, -90, and 180 degrees.

FIGURE 11: (a) Logarithm of objective function of (6) obtained after a line search conducted for a test point with a yaw angle of 0 degrees. (b) Yaw angle prediction errors on the RALS tower data when training was conducted on data at every 4-degree increments of yaw angle using the forward approach.

FIGURE 12: Wind tunnel coefficient of pressure as a function of yaw angle and pressure sensor position.

FIGURE 13: Fused coefficient of pressure surface as a function of yaw angle and pressure sensor position.

could be a serious problem if the input domain is excessively large or output determination is expensive. Careful selection of training points is also important to construct surrogates with low generalization error. Input sampling strategies where the learning algorithm actively takes part in the data collection procedure are studied in the field of active learning or query learning as suggested by its name. In statistics they are studied in the field of Optimal Experiment Design (OED) [26].

4.1. Active Learning for Regression Problems. For continuous valued problems, significant research has been done in the field of statistics under the name of Optimal Experiment Design [26]. An acceptable active learning method for regression problems should be able to minimize mean squared error

more than the passive learning methods using fewer training points. Consider a general regression problem in

$$y(x) = f(x, \beta) + \varepsilon \quad (9)$$

with d dimensional inputs $x \in \mathcal{R}^d$, output $y \in \mathcal{R}$, and random errors ε which we assume to be independent and uncorrelated. Given a small, finite number of training samples s , the parameters of this regression problem can be estimated that minimize the mean squared error given by

$$E_{\text{MSE}} = \int \|f(x, \hat{\beta}) - f(x)\|^2 dQ(x), \quad (10)$$

where Q is the environmental probability. However, since the environment is generally realized by the user only via the

s number of training points, the following estimated mean squared error in (11) is minimized to estimate the parameters β :

$$S(\bar{\beta}) = \min_{\bar{\beta}} \frac{1}{s} \sum_{i=1}^s (f(x_i, \bar{\beta}) - y(x_i))^2. \quad (11)$$

Given a finite number of training samples and a learning algorithm to estimate the parameters $\bar{\beta}$, an optimum active learning scheme would compute a sample point where the expected mean squared error over the whole input domain is maximum. Estimation of the future generalization error could be a very computationally demanding task for both pool based and population based active learning methods where information about the conditional probability distribution is either known or assumed. Kindermann et al. [27] use a Bayesian theoretic framework to develop an active learning method where they use Markov Chain Monte Carlo methods to approximate the expected loss. Roy and McCallum [28] estimate the expected error by adding all possible labels for each unlabeled sample to the training set. The proposed brute force approach is infeasible for regression problems and classification problems with many classes. The authors propose several ways to make this procedure efficient, for example, use of Naïve Bayes and SVMs where addition of a new training point does not need retraining, or estimate EMSE only on the neighborhood of the candidate points which compromises the generality of the approach. It is important to note here that methods such as Kriging/Gaussian Processes have the Expected Improvement criteria which can be used for adaptive sampling and estimating uncertainty in results. However, the authors wanted to show that a similar framework can be developed with other state-of-the-art metamodels. In the current FADS application, the number of training data points is high which would take longer for the Gaussian process or Kriging models to train. The SFA method trains quickly while preserving accuracy of the solution. And we show that with this new method we can also construct a framework for adaptive sampling and fusing high and low fidelity data.

Since estimation of expected future loss is infeasible, one can rely on the bias-variance decomposition of EMSE to estimate a sample where the expected error would be maximum. For a given sample x , let $y = f(x, \beta)$, $\bar{y} = f(x, \bar{\beta})$, and $\bar{y} = E[\bar{y}]$ be the true output, estimated output, and the expected value of the estimated output, respectively. The expected value can be computed by averaging over different permutations of randomly chosen training sets of size s . The expected mean squared error can be decomposed into bias and variance as shown in

$$\begin{aligned} E_{\text{MSE}} &= E[(\bar{y} - y)^2] = E[\bar{y}^2] + y^2 - 2y\bar{y} \\ &= E[\bar{y}^2] + y^2 - 2y\bar{y} + 2\bar{y}^2 - 2\bar{y}^2 \\ &= (y - \bar{y})^2 + E[\bar{y}^2] + \bar{y}^2 - 2\bar{y}E[\bar{y}] \\ &= \underbrace{(y - \bar{y})^2}_{\text{Bias}} + \underbrace{E[(\bar{y} - \bar{y})^2]}_{\text{Variance}}, \end{aligned} \quad (12)$$

where the first term represents bias and the second term represents variance of the model. To elaborate, bias of a model can be understood as, for example, a quadratic function is being estimated by a linear model. The errors of the model are independent of the accuracy of the estimated parameters. The variance on the other hand is the error due to the errors between the estimated and the true parameters. For a simple regression problem shown below in

$$y(x, \beta) = \beta_0 + \beta_1 x + \beta_2 x^2 + \dots + \beta_{p-1} x^{p-1},$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_s \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{p-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{p-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_s & x_s^2 & \dots & x_s^{p-1} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{p-1} \end{bmatrix}, \quad (13)$$

or $y = F\beta$.

Simple linear least squares regression dictates the estimated solution to be $\bar{\beta} = (F^T F)^{-1} F^T y$. The covariance matrix of the least squares estimator is given by $\text{var}(\bar{\beta}) = \sigma^2 (F^T F)^{-1}$. The $p \times p$ matrix $F^T F$ is called the Fisher Information Matrix (FIM) of β , the determinant of which is inversely proportional to the volume of the confidence ellipsoid of the parameters. The set of training points that maximize the determinant of FIM are called D-optimum. Given the variance of the parameter estimates, the standard variance of the predicted value of response at a given point x can also be calculated by

$$d(x) = \frac{\text{var}(\bar{y}(x))}{\sigma^2} = f^T(x) (F^T F)^{-1} f(x), \quad (14)$$

where $f^T(x)$ is a row of F evaluated at x . The standardized variance given in the above equation gives a way to calculate the estimated variance of a linear model whose parameters are computed given a finite training set. This means that for a model with insignificant bias, an optimum sample can be computed that maximizes (14). At this sample the expected mean squared error will be maximum making it an optimum choice for a training point. The set of training points that minimize the maximum standardized variance are called G-optimum. The previous analysis was derived for linear regression problems; an extension to nonlinear problems is possible using Taylor's expansion. Consider a nonlinear model with p parameters $y(x) = f(x, \bar{\beta})$, where $\bar{\beta} = [\beta_1, \beta_2, \dots, \beta_p]$. Taylor expansion of the model about $\bar{\beta}^0 = [\beta_1^0, \beta_2^0, \dots, \beta_p^0]$ ignoring higher-order derivatives would yield

$$y(x) = f(x, \bar{\beta}^0) + \sum_{i=1}^p (\beta_i - \beta_i^0) \left. \frac{\partial f}{\partial \beta_i} \right|_{\beta_i = \beta_i^0}. \quad (15)$$

Let $z(x) = y(x) - f(x, \vec{\beta}^0)$, $\gamma_i = \beta_i - \beta_i^0$, and $g_i = (\partial f / \partial \beta_i)|_{\beta_i = \beta_i^0}$, which gives us the following linearized form of

$$z(x) = \sum_{i=1}^p \gamma_i g_i. \quad (16)$$

The variance of this new linearized form can be computed in a similar manner as described above for a linear regression problem with p parameters. The standard variance of $z(x)$ would be the same as $y(x)$ and can be computed using (17) where

$$f^T(x) = \left[\frac{\partial f}{\partial \beta_1}, \frac{\partial f}{\partial \beta_2}, \dots, \frac{\partial f}{\partial \beta_p} \right] \Big|_{\beta = \beta^0}. \quad (17)$$

And computing the derivatives for SFA,

$$y(x) = \sum_{i=1}^n c_i \phi(x, \alpha_i, x_i^*) + b$$

$$\text{where } \alpha_i = \log(\lambda_i), \quad 0 < \lambda_i < 1, \quad n = 1,$$

$$\phi(x, \alpha_i, x_i^*) = \exp(\alpha_i (x - x_i^*)^2) = \phi_i$$

$$\frac{\partial y}{\partial c_i} = \phi_i$$

$$\frac{\partial y}{\partial \alpha_i} = c_i \phi_i (x - x_i^*)^2 \quad (18)$$

$$\frac{\partial y}{\partial x_i^*} = 2c_i \phi_i \alpha_i |x - x_i^*|$$

$$\frac{\partial y}{\partial b} = 1$$

$$f^T(x) = [\phi_1, c_1 \phi_1 (x - x_1^*)^2, 2c_1 \phi_1 \alpha_1 |x - x_1^*|, \phi_2, c_2 \phi_2 (x - x_2^*)^2, 2c_2 \phi_2 \alpha_2 |x - x_2^*|, \dots, 1] \in \mathbb{R}^{3n+1}.$$

4.2. Implementation Issues and Application. There are several implementation issues that need to be carefully considered when developing a G-optimal design procedure using SFA with RBFs.

4.2.1. Singularity of Fisher Information Matrix. The singularity of Fisher Information Matrix (FIM) poses a significant problem in the implementation of a G-optimal design procedure especially with RBFs. The FIM can become singular due to several reasons. If $f^T(x)$ represents a row vector of the derivatives of the surrogate model with respect to its parameters evaluated at point x , as shown in (19), FIM is constructed by computing $f(x_i) * f^T(x_i)$ which is a $p \times p$ matrix and summing it over the available training points. It is important to realize that $f(x_i) * f^T(x_i)$ is a matrix of rank

1 and that FIM gains rank as it is summed over increasing number of training points.

$$\text{FIM} = \sum_{i=1}^s f(x_i) * f^T(x_i). \quad (19)$$

This could serve as a guideline to select the minimum number of initial training points necessary to start the incremental G-optimal design procedure. The initial number of training points (s) should be greater than or equal to the number of parameters (p). For greedy algorithms like SFA, this could also decide the stopping criterion to add basis functions.

4.2.2. Overparameterization. Fukumizu [29] has shown that FIM can also become singular if any redundant basis functions are added to the approximation. FIM will become singular in the presence of singular basis functions even if it is constructed by summing over a large number of training points. This is also true for SFA in case a redundant basis function is added to the approximation when only noise is left in the residual error signal. A redundant RBF in a surrogate model constructed by SFA can easily be identified because its coefficient value will be very close to zero and the width of the RBF will also be very low, making the RBF appear as a spike. If this basis function is included in the surrogate model, the parameter derivatives and the FIM will bear this singularity resulting in spiking of the standardized variance values at the redundant basis function center. In order to avoid this problem, if the coefficient or the width of a RBF is unusually low then it should be eliminated from the surrogate model and the addition of basis functions should be stopped in order to preserve the smoothness of the surrogate model and the standard variance.

4.2.3. Initial Number of Training Points and Stopping Criteria. The initial number of training points can be chosen at random as long as it is large enough to avoid construction of highly inaccurate surrogate models. As mentioned before, the number of training points (s) should be larger than the number of parameters. A simple heuristic such as $s = 1.5p$ can be used to set the stopping criterion given the number of training points. However, this stopping criterion will continue adding an increasing number of basis functions as more training points are added, which will lead to overparameterization. This calls for an additional check that will stop the addition of basis functions if the coefficient, or the width, of the latest RBF falls below a user specified tolerance.

4.2.4. The Bias Problem. The strategy of maximizing the standardized variance will result in an optimal design sample only if the model bias is insignificant. Previous work done by Kanamori and Sugiyama [30–32] has focused on using weighted maximum log-likelihood approaches to result in robust active learning schemes in case that the interpolation model has been misspecified. Sugiyama [31] and Hering and Šimandl [33] have also addressed the issue of using the conditional expectation of the generalization error instead of minimizing the expected loss in an asymptotic sense.

However, these efforts are only possible for population based active learning methods where information about the probability distribution of the parameters is either available a priori or is assumed. In this paper, attention is focused only on pool based active learning methods where no a priori information about the parameter probability distributions is available and sufficient initial data is not present to make good estimations about global parameter probability distribution functions. In this work, the strong approximation capabilities of universal approximators like radial basis functions are relied on to attenuate the bias problem. A possible strategy is to start the active learning procedure with an overparameterized model and remove any redundant basis functions before constructing the FIM. As the number of training points increases the number of redundant basis functions should diminish. Another constraint on the number of basis functions is that the resulting number of parameters ($2n + 1$) should be less than the number of training points to prevent the FIM from being singular.

The final steps to implement the G-optimal design procedure with SFA are as follows:

- (1) Use LHS method to pick a small number of input points and evaluate the output on them to construct the initial training set.
- (2) Construct a surrogate model using SFA. The number of basis functions can be determined by using the heuristic $s \approx 1.5p = 1.5 * (2n + 1)$, where n is the number of basis functions.
- (3) Evaluate derivatives with respect to width and coefficient parameters of each basis function and the bias parameter and construct the FIM.
- (4) Do a line search to determine a sample input location that maximizes the standardized variance of the surrogate model output.
- (5) Add the chosen points to the training set and repeat the steps.

4.3. Sequential Experiment Design for FADS. In this section, the G-optimal design procedure developed with SFA is tested on the FADS problem for the RALS tower. Wind tunnel tests for the FADS problem measure surface pressure on the surface at fixed sensor locations (θ). Test runs consist of sweeps where the Mach number or the freestream wind speed is fixed and the bluff body model is rotated resulting in pressure measurements at different incident yaw angles (β). The time and effort of wind tunnel testing include changing pressure sensor locations (θ) and freestream velocity (V_∞). However, CFD simulations give pressure distribution for all θ for a given (V_∞) and yaw angle β . Here the time consuming part is repeating the simulations for different β . The time and effort required would increase drastically if 3D flow is considered and surface pressure is measured as a function of V_∞ , yaw angle β , and pitch γ . For 2D flow, if β is measured at increments of 2 degrees, pressure measurements have to be made 180 times at each V_∞ . For 3D flow, if β and γ are measured at increments of 2 degrees, 180×180 pressure measurements have to be made at each V_∞ . An experiment

design strategy for the hypersurface $C_p = f(\theta, \beta)$ could be helpful in reducing the time and cost of FADS. Optimal training data would also be able to give faster surrogates for C_p and would accelerate the forward problem approach discussed in Section 2. For demonstration purposes, it is assumed that one can freely change θ and β to arbitrary continuous values within the interval $[-180, 180]$. However, in reality wind tunnel tests can only freely change β while CFD runs can only change θ freely. Including V_∞ in the experiment design framework is left future work.

Figure 13 showed the hypersurface of C_p as a function θ and β which will be used to create the test data set for the design procedure. The G-optimal design procedure will be compared against the traditional LHS design procedure. An initial sample of 100 training points was chosen by LHS to start the design procedure. Surrogates were constructed by SFA using increasing number of LHS training points with increment of 5 points till a training set size of 500 was reached. For each training set size the training-testing procedure was repeated 10 times to eliminate any bias due to selection of training points and the mean and standard error of prediction accuracies were computed. The test set was constructed by the data-model fusion procedure. CFD solutions were constructed with a β increment of 1 degree and a θ increment of approximately 6 degrees. The wind tunnel experiment used β increments of 4 degrees and four pressure sensors. The resulting test set had 20577 ($= 361 \times 57$) points which captured the C_p variation in detail.

To implement the G-optimal design procedure, the stopping criteria discussed in Section 4 were used. This ensured that the FIM would not be singular. Also as the number of training points is increased the number of basis functions will also increase and address the bias problem. The lower bound on the RBF width parameter was constrained to $1e - 64$ and the center selection heuristic corresponding to the maximum absolute value of the residual was chosen. The basis centers were not optimized to save computational expense. One point was added to the training set at each iteration till 500 training points were collected. Again, to put the performance of the active learning procedure in perspective, its performance was also compared to the scenario when it is assumed that the true model is known and one point is chosen at each iteration. This corresponds to the maximum squared error of the current surrogate model.

It was also realized that additional information could also be incorporated in the training set for this problem. C_p^2 bears peaks that correspond to those combinations of θ and β that result in maximum flow separation. Due to the geometry of the RALS tower bluff body, sensors on each wall measure the lowest pressures when the freestream wind is directed against the adjacent wall.

This physical phenomenon was used to enhance the information contained in the training set by adding points where $\beta = [-180, -90, 0, 90, 180]$ degrees. Result for this case, shown in Figure 14, displays that active learning improves upon the LHS data collection method. Figure 15 shows the C_p^2 training points chosen by the G-optimal design procedure. The regions of the input domain corresponding to significant

FIGURE 14: Reduction of mean squared error with increasing number of training points. Bars show standard error.

FIGURE 15: Training points chosen by the G-optimal design procedure.

functional variation are efficiently covered by the design algorithm corroborating the claims of Optimal Experiment Design.

5. Conclusions

This work was devoted to studying the compatibility of flush air data systems with bluff bodies with arbitrary geometries. Nontrivial geometries and requirements for efficient real time usage resulted in a challenging inverse problem of determining air data parameters from surface pressure measurements that could be best solved by intelligent data driven algorithms. A data driven algorithm was developed based on the principles of scattered data approximation. It was proposed

that mathematical models can be utilized as a surrogate that solves the forward problem of generating pressure values from known wind speed and direction configuration. This surrogate can be used in conjunction with an array of flush mounted pressure taps to act as an air data measurement system. The proposed scattered data approximation algorithm, called Sequential Function Approximation (SFA), is a greedy self-adaptive function approximation tool that results in reliable and robust estimates without any user dependent control or kernel hyperparameter selection. In particular, a nonparametric and adaptive scattered data approximation tool accurately and efficiently mapped wind speed and yaw angle to pressure measurements taken from all four sides of the body. SFA was also used to reduce uncertainty in the wind tunnel tests by fusing data with low fidelity and smooth CFD models. Finally, to accelerate through the test matrices an intelligent sampling strategy was developed using principles of sequential experiment design.

Appendix

Sequential Function Approximation

Approximation of the unknown target function y is constructed by noting that a continuous d -dimensional function can be arbitrarily well approximated by a linear combination of radial basis functions ϕ . Sequential Function Approximation (SFA) was developed from mesh-free finite element research but shares similarities with the Boosting [34] and Matching Pursuit [35] algorithms. SFA was developed earlier for regression and classification problems [16] and it is presented here for completeness. Approximation of the target vector y is constructed by utilizing the Gaussian radial basis function ϕ :

$$y_n^a = \sum_{i=1}^n c_i (\phi(x, \beta_i) + b_i) \quad (\text{A.1})$$

$$\text{with } \phi(x, \beta_i) = \exp[\log[\lambda_i](x - x_i^*) \cdot (x - x_i^*)]$$

$$\text{for } 0 < \lambda_i < 1.$$

Traditionally, Gaussian radial basis functions are written as

$$\phi(x, \beta_i) = \exp\left[-\frac{(x - x_i^*) \cdot (x - x_i^*)}{\sigma_i^2}\right]. \quad (\text{A.2})$$

Radial basis function is written as (A.1) in order to set up the optimization problem for λ_i as a bounded nonlinear line search instead of an unconstrained minimization problem for σ_i . The basic principles of our greedy algorithm are motivated by the similarities between the iterative optimization procedures of Jones [36, 37] and Barron [38] and the Method of Weighted Residuals (MWR), specifically the Galerkin

method [39]. The function residual vector $\vec{r}_n$ at the n th stage of approximation can be written as in

$$\begin{aligned}\vec{r}_n &= y(x) - y_n^a(x) \\ &= y(x) - y_{n-1}^a(x) - c_n(\phi(x, \beta_n) + b_n) \\ &= \vec{r}_{n-1} - c_n(\phi(x, \beta_n) + b_n) = \vec{r}_{n-1} - c_n(\vec{\phi}_n + b_n).\end{aligned}\quad (\text{A.3})$$

Using the Petrov-Galerkin approach, a coefficient c_n is selected that will force the function residual to be orthogonal to the basis function and b_n using the discrete inner product $\langle, \rangle_D$ given by

$$\begin{aligned}\vec{r}_n \cdot (\vec{\phi}_n + b_n) &= \sum_{j=1}^s r_n(x_j) (\phi_n(x_j, \beta_n) + b_n) \\ &= \langle \vec{r}_n, (\vec{\phi}_n + b_n) \rangle_D = - \left\langle \vec{r}_n, \frac{\partial \vec{r}_n}{\partial c_n} \right\rangle_D \\ &= - \frac{1}{2} \frac{\partial \langle \vec{r}_n, \vec{r}_n \rangle_D}{\partial c_n} = 0\end{aligned}\quad (\text{A.4})$$

which is equivalent to selecting a value of c_n that will minimize $\langle \vec{r}_n, \vec{r}_n \rangle_D$. Writing (A.3) as (A.5) where $g_n = c_n \cdot b_n$:

$$\vec{r}_n = \vec{r}_{n-1} - c_n \vec{\phi}_n - g_n. \quad (\text{A.5})$$

Expanding $\langle \vec{r}_n, \vec{r}_n \rangle_D$ and taking the derivative with c_n , we get

$$\begin{aligned}\langle \vec{r}_n, \vec{r}_n \rangle_D &= \langle \vec{r}_{n-1}, \vec{r}_{n-1} \rangle_D + c_n^2 \langle \vec{\phi}_n, \vec{\phi}_n \rangle_D + s g_n^2 \\ &\quad - 2c_n \langle \vec{r}_{n-1}, \vec{\phi}_n \rangle_D - 2g_n \langle \vec{r}_{n-1} \rangle_D\end{aligned}$$

$$\sum_{j=1}^s r_n(x_j) r_n(x_j) = \|\vec{r}_n\|_{2,D}^2 = \langle \vec{r}_n, \vec{r}_n \rangle_D,$$

$$\begin{aligned}\langle \vec{r}_n, \vec{r}_n \rangle_D &= \left\langle \vec{r}_{n-1} - \frac{\langle \vec{r}_{n-1} \rangle_D}{s}, \vec{r}_{n-1} - \frac{\langle \vec{r}_{n-1} \rangle_D}{s} \right\rangle_D \\ &\quad \cdot \left(1 - \frac{\langle \vec{r}_{n-1} - \langle \vec{r}_{n-1} \rangle_D / s, \vec{\phi}_n - \langle \vec{\phi}_n \rangle_D / s \rangle_D^2}{\langle \vec{\phi}_n - \langle \vec{\phi}_n \rangle_D / s, \vec{\phi}_n - \langle \vec{\phi}_n \rangle_D / s \rangle_D \langle \vec{r}_{n-1} - \langle \vec{r}_{n-1} \rangle_D / s, \vec{r}_{n-1} - \langle \vec{r}_{n-1} \rangle_D / s \rangle_D} \right).\end{aligned}\quad (\text{A.10})$$

Recalling the definition of the cosine given by (A.11), using arbitrary functions f and v and the discrete inner product,

$$\cos(\theta) = \frac{\langle f, v \rangle_D}{\langle f, f \rangle_D^{1/2} \langle v, v \rangle_D^{1/2}}. \quad (\text{A.11})$$

$$+ 2c_n g_n \langle \vec{\phi}_n \rangle_D,$$

$$\frac{\partial \langle \vec{r}_n, \vec{r}_n \rangle_D}{\partial c_n} = 2c_n \langle \vec{\phi}_n, \vec{\phi}_n \rangle_D - 2 \langle \vec{r}_{n-1}, \vec{\phi}_n \rangle_D$$

$$+ 2g_n \langle \vec{\phi}_n \rangle_D = 0,$$

$$c_n = \frac{\langle \vec{r}_{n-1}, \vec{\phi}_n \rangle_D - g_n \langle \vec{\phi}_n \rangle_D}{\langle \vec{\phi}_n, \vec{\phi}_n \rangle_D}. \quad (\text{A.6})$$

Taking the derivative of $\langle \vec{r}_n, \vec{r}_n \rangle_D$ with g_n relates c_n and g_n in

$$\begin{aligned}\frac{\partial \langle \vec{r}_n, \vec{r}_n \rangle_D}{\partial g_n} &= 2s g_n - 2 \langle \vec{r}_{n-1} \rangle_D + 2c_n \langle \vec{\phi}_n \rangle_D = 0, \\ g_n &= \frac{\langle \vec{r}_{n-1} \rangle_D - c_n \langle \vec{\phi}_n \rangle_D}{s}.\end{aligned}\quad (\text{A.7})$$

Substitute g_n in (A.6) and solve for c_n in

$$c_n = \frac{(s \langle \vec{\phi}_n, \vec{r}_{n-1} \rangle_D - \langle \vec{r}_{n-1} \rangle_D \langle \vec{\phi}_n \rangle_D)}{(s \langle \vec{\phi}_n, \vec{\phi}_n \rangle_D - \langle \vec{\phi}_n \rangle_D \langle \vec{\phi}_n \rangle_D)}. \quad (\text{A.8})$$

Since $b_n = g_n/c_n$, (A.9) gives b_n :

$$b_n = \frac{(\langle \vec{r}_{n-1} \rangle_D \langle \vec{\phi}_n, \vec{\phi}_n \rangle_D - \langle \vec{\phi}_n \rangle_D \langle \vec{\phi}_n, \vec{r}_{n-1} \rangle_D)}{(s \langle \vec{\phi}_n, \vec{r}_{n-1} \rangle_D - \langle \vec{r}_{n-1} \rangle_D \langle \vec{\phi}_n \rangle_D)}. \quad (\text{A.9})$$

The discrete inner product $\langle \vec{r}_n, \vec{r}_n \rangle_D$, which is equivalent to the square of the discrete L_2 norm, can be rewritten, with the substitution of (A.8) and (A.9), as

Equation (A.10) can be written as

$$\begin{aligned}\|\vec{r}_n\|_{2,D}^D &= \left\langle \vec{r}_{n-1} - \frac{\langle \vec{r}_{n-1} \rangle_D}{s}, \vec{r}_{n-1} - \frac{\langle \vec{r}_{n-1} \rangle_D}{s} \right\rangle_D \sin^2(\theta_n) \\ &= \left(\|\vec{r}_{n-1}\|_{2,D}^D - \frac{\langle \vec{r}_{n-1} \rangle_D^2}{s} \left[2 - \frac{1}{s} \right] \right) \sin^2(\theta_n),\end{aligned}\quad (\text{A.12})$$

where θ_n is the angle between $\vec{\phi}_n$ and $\vec{r}_{n-1}$ since $\langle \vec{r}_{n-1} \rangle_D / s$ and $\langle \vec{\phi}_n \rangle_D / s$ are scalars. With (A.12) one can see that $\|\vec{r}_n\|_{2,D}^2 < \|\vec{r}_{n-1}\|_{2,D}^2$ as long as $\theta_n \neq \pi/2$, which is a very robust condition for convergence. By inspection, the minimum of (A.12) is $\theta_n = 0$, implying

$$c_n(\phi(x_i, \beta_n) + b_n) = r_{n-1}(x_i) \quad \text{for } i = 1, \dots, s. \quad (\text{A.13})$$

Therefore, to force $\|\vec{r}_n\|_{2,D}^2 \rightarrow 0$ with as few stages n as possible, a low dimensional function approximation problem must be solved at each stage. This involves a bounded nonlinear minimization of (A.10) to determine the two variables λ_i ($0 < \lambda_i < 1$) and index j^* ($x_n^* = x_{j^*} \in \mathcal{R}^d$) for the basis function center taken from the training set. The dimensionality of the nonlinear optimization problem is kept low since only one basis function needs to be solved at a time. To implement the SFA algorithm the user takes the following steps:

- (1) Initiate the algorithm with the labels of the training data $\vec{r}_0 = \{y(x_1), \dots, y(x_s)\}$.
- (2) *Basis center selection*: Search the components of $\vec{r}_{n-1}$ for the maximum magnitude. Record the component index j^* .
- (3) $x_n^* = x_{j^*}$.
- (4) *Basis width optimization*: With $\vec{\phi}_n$ centered at x_n^* , minimize (A.10) using optimization methods for bounded line search problems.
- (5) *Coefficient calculation*: Calculate the coefficient c_n and b_n from (A.8) and (A.9), respectively.
- (6) *Residual update*: Update the residual vector $\vec{r}_n = \vec{r}_{n-1} - c_n(\vec{\phi}_n + b_n)$. Repeat steps (3) to (6) until the termination criterion $\max|\vec{r}_n| \leq \tau$ has been met.

Our SFA method is linear in storage with respect to s since it needs to store only $s + sd$ vectors to compute the residuals: one vector of length $s(r_n)$ and s vectors of length d ($\xi_1, \dots, \xi_s$) where s is the number of samples and d is the number of dimensions. To complete the SFA surrogate requires two vectors of length n ($\{c_1, \dots, c_n\}$ and $\{b_1, \dots, b_n\}$) and n vectors of length $n + 1$ ($\eta_1, \dots, \eta_n$). The dimensionality of the nonlinear optimization problem is kept low since we are solving only one basis at a time. Also, Meade and Zeldin [40] showed that, for bell-shaped basis functions, the minimum convergence rate is

$$\|r_n\|_2^2 \cong \langle r_n, r_n \rangle_D = \exp[\kappa_1 - \kappa_2 n] \quad \text{where } n \geq 2, \quad (\text{A.14})$$

for positive constants κ_1 and κ_2 . Equation (A.14) shows that, for minimum convergence, the logarithm of the inner product of the residual is a linear function of the number of bases (n), which establishes an exponential convergence rate that is independent from the number of input dimensions (d). For a detailed discussion and comparison of SFA with other state-of-the-art learning tools for regression and classification problems, please refer [41, 42].

Nomenclature

$b_n, b1_n, b2_n$:	n th bias in the approximation
$c_n, c1_n, c2_n$:	n th linear coefficient in the approximation
C_p :	Coefficient of pressure = P/q
d :	Input dimension
f :	Arbitrary function
fps:	Feet per second
F :	Forward mapping function
G :	Inverse mapping function
n :	Number of bases
P, q :	Static gauge pressure and dynamic pressure
r_n :	n th stage of the function residual
$\mathbb{R}$:	Real coordinate space
$\mathbb{R}^d$:	d -dimensional vector space
s :	Number of training samples
$u(\xi)$:	Target function
$u_n^a(\xi)$:	n th stage of the target function approximation
V_∞ :	Freestream wind speed
$\langle f, v \rangle_D$:	$\sum_i^s (f_i v_i) =$ discrete inner product of f and v
$ f $:	Absolute value of f
$\ f\ _2$:	$(\int_\Omega f^2 d\xi)^{1/2} = L_2$ norm of f
$\ f\ _{2,D}$:	$(\sum_i (f_i)^2)^{1/2} =$ discrete L_2 norm of f .

Greek Symbols

β :	Yaw angle
η :	Set of nonlinear optimization parameters
ε :	Objective function
θ :	Polar coordinate
λ :	Width parameter of Gaussian basis function
ξ :	d -dimensional input of the target function
ξ_i :	i th sample input of the target function
ξ_n^* :	Sample input with component index j^* at the n th stage
τ :	User specified tolerance
ϕ :	Basis function
v :	Arbitrary function
δ :	Input sensitivity
ρ :	Density.

Subscripts

i :	Dummy index
j^* :	Component index of $r_{(n-1)}$ with the maximum magnitude
n :	Associated with the number of bases
s :	Associated with the number of samples
∞ :	Associated with freestream conditions.

Superscripts

a :	Associated with the approximation
d :	Associated with the input dimension.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

- [1] T. J. Rohloff, S. A. Whitmore, and I. Catton, "Fault-tolerant neural network algorithm for flush air data sensing," *Journal of Aircraft*, vol. 36, no. 3, pp. 541–549, 1999.
- [2] I. Samy, I. Postlethwaite, and D. Gu, "Subsonic tests of a flush air data sensing system applied to a fixed-wing micro air vehicle," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 54, no. 1–3, pp. 275–295, 2009.
- [3] T. J. Rohloff, *Development and evaluation of neural network flush air data sensing systems [Ph.D. thesis]*, Department of Mechanical Engineering, University of California, Los Angeles, Los Angeles, Calif, USA, 1998.
- [4] T. J. Larson, T. R. Moes, and P. M. Siemers III, *Wind-Tunnel Investigation of a Flush Airdata System at Mach Numbers From 0.7 to 1.4*, NASA Dryden Research Center, Palmdale, Calif, USA, 1990.
- [5] S. A. Whitmore, R. J. Davis, and J. M. Fife, "In-flight demonstration of a real-time flush Airdata sensing (RT-FADS) system," NASA Technical Memorandum 104314, Dryden Flight Research Center, 1995.
- [6] G. Johnson, *Wind Energy Systems*, Technology & Engineering, Prentice Hall, 1985.
- [7] K. Long, *Wind Tunnel Surface Pressure Measurements on a 1/72 Scale Model of the Runway Arrested Landing Site (RALS) Control Tower*, Naval Air Warfare Center Aircraft Division, Patuxent River, Md, USA, 2008.
- [8] L. Milne-Thompson, *Theoretical Hydrodynamics*, The Macmillan Company, London, UK, 1955.
- [9] J. Katz and A. Plotkin, *Low-Speed Aerodynamics*, Cambridge University Press, Cambridge, UK, 2nd edition, 2001.
- [10] S. S. Desai, "Relative roles of computational fluid dynamics and wind tunnel testing in the development of aircraft," *Current Science*, vol. 84, no. 1, pp. 49–64, 2003.
- [11] W. E. Faller and S. J. Schreck, "Neural networks: applications and opportunities in aeronautics," *Progress in Aerospace Sciences*, vol. 32, no. 5, pp. 433–456, 1996.
- [12] P. B. Nair, A. Choudhury, and A. J. Keane, "Some greedy learning algorithms for sparse regression and classification with Mercer kernels," *Journal of Machine Learning Research*, vol. 3, pp. 781–801, 2003.
- [13] M. D. McKay, "Latin hypercube sampling as a tool in uncertainty analysis of computer models," in *Proceedings of the 24th conference on Winter simulation*, pp. 557–564, Arlington, Va, USA, December 1992.
- [14] M. Hasenjäger, *Active data selection in supervised and unsupervised learning [Ph.D. thesis]*, Bielefeld University, Bielefeld, Germany, 2000.
- [15] A. J. Meade Jr. and B. A. Zeldin, "Integrating experimental data and mathematical models in simulation of physical systems," *AIAA Journal*, vol. 35, no. 11, pp. 1787–1790, 1997.
- [16] A. Srivastava, A. J. Meade, and K. R. Long, "Learning air-data parameters for flush air data sensing systems," *Journal of Aerospace Computing, Information and Communication*, vol. 9, no. 3, pp. 110–124, 2012.
- [17] D. Lowe and C. Zapart, "Point-wise confidence interval estimation by neural networks: a comparative study based on automotive engine calibration," *Neural Computing & Applications*, vol. 8, no. 1, pp. 77–85, 1999.
- [18] J. Hull, D. Ward, and R. R. Zakrzewski, "Verification and validation of neural networks for safety-critical applications," in *Proceedings of the American Control Conference*, vol. 6, pp. 4789–4794, May 2002.
- [19] P. J. Roebber, M. R. Butt, S. J. Reinke, and T. J. Grafenauer, "Real-time forecasting of snowfall using a neural network," *Weather and Forecasting*, vol. 22, no. 3, pp. 676–684, 2007.
- [20] P. J. Shayler, M. Goodman, and T. Ma, "Exploitation of neural networks in automotive engine management systems," *Engineering Applications of Artificial Intelligence*, vol. 13, no. 2, pp. 147–157, 2000.
- [21] K. R. Long, B. J. Porter, T. J. Garbeff, S. D. Schery, and L. E. Chen, "Development, testing, and validation of an anemometer employing a flush air data system (FADS)," in *Proceedings of the 50th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, AIAA 2012-0696, Nashville, Tenn, USA, January 2012.
- [22] R. Unger, M. Hutchinson, M. Rais-Rohani, R. Haftka, and B. Grossman, "Variable-complexity design of a transport wing," *International Journal of Systems Automation: Research and Applications*, vol. 2, pp. 87–113, 1992.
- [23] A. J. Keane and P. B. Nair, *Computational Approaches for Aerospace Design: The Pursuit of Excellence*, Wiley, 2005.
- [24] R. T. Haftka, "Combining global and local approximations," *AIAA Journal*, vol. 29, no. 9, pp. 1523–1525, 1991.
- [25] A. Srivastava, A. J. Meade, and A. A. Mokhtarzadeh, "A hybrid data-model fusion approach to calibrate a flush air data sensing system," in *Proceedings of the AIAA Infotech at Aerospace*, Atlanta, Ga, USA, April 2010.
- [26] A. C. Atkinson, A. N. Donev, and R. D. Tobias, *Optimum Experimental Designs, with SAS*, Oxford University Press, 2007.
- [27] J. Kindermann, G. Paass, and F. Weber, "Query construction for neural networks using the bootstrap," in *Proceedings of the International Conference on Artificial Neural Networks (ICANN '95)*, pp. 135–140, October 1995.
- [28] N. Roy and A. McCallum, "Toward optimal active learning through sampling estimation of error reduction," in *Proceedings of the 18th International Conference on Machine Learning (ICML '01)*, pp. 441–448, Williamstown, Mass, USA, June–July 2001.
- [29] K. Fukumizu, "A regularity condition of the information matrix of a multilayer perceptron network," *Neural Networks*, vol. 9, no. 5, pp. 871–879, 1996.
- [30] T. Kanamori and H. Shimodaira, "Active learning algorithm using the maximum weighted log-likelihood estimator," *Journal of Statistical Planning and Inference*, vol. 116, no. 1, pp. 149–162, 2003.
- [31] M. Sugiyama, "Active learning in approximately linear regression based on conditional expectation of generalization error," *Journal of Machine Learning Research*, vol. 7, pp. 141–166, 2006.
- [32] M. Sugiyama and S. Nakajima, "Pool-based active learning in approximate linear regression," *Machine Learning*, vol. 75, no. 3, pp. 249–274, 2009.
- [33] P. Hering and M. Šimandl, "Sequential optimal experiment design for neural networks using multiple linearization," *Neurocomputing*, vol. 73, no. 16–18, pp. 3284–3290, 2010.
- [34] Y. Freund, "Boosting a weak learning algorithm by majority," *Information and Computation*, vol. 121, no. 2, pp. 256–285, 1995.
- [35] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.
- [36] L. K. Jones, "Constructive approximations for neural networks by sigmoidal functions," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1586–1589, 1990.

- [37] L. K. Jones, "A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training," *The Annals of Statistics*, vol. 20, no. 1, pp. 608–613, 1992.
- [38] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 930–945, 1993.
- [39] C. A. Fletcher, *Computational Galerkin Methods*, Springer-Verlag, New York, NY, USA, 1984.
- [40] A. J. Meade and B. A. Zeldin, "Approximation properties of local bases assembled from neural network transfer functions," *Mathematical and Computer Modelling*, vol. 28, no. 9, pp. 43–62, 1998.
- [41] A. Srivastava and A. J. Meade, "A sparse greedy self-adaptive algorithm for classification of data," *Advances in Adaptive Data Analysis*, vol. 2, no. 1, pp. 97–114, 2010.
- [42] A. Srivastava, A. J. Meade, and J. Needham, "Adaptive algorithm for classifying launch and recovery of the HH-60H seahawk," *Journal of Aerospace Computing, Information and Communication*, vol. 6, no. 2, pp. 109–120, 2009.