Intelligent Multi-Resolution Modelling: Application to Synthetic Jet Actuation and Flow Control

Puneet Singla,a John L. Junkins,b Othon Rediniotisc
Texas A&M University, 3141-TAMU, College Station, TX 77843-3141
Kamesh Subbarao
University of Texas, Arlington, Box 19018, 211 Wooll Hall, Arlington, TX 76019-0018

A novel “directed graph” based algorithm is presented that facilitates intelligent learning and adaptation of the parameters appearing in a Radial Basis Function Network (RBFN) description of input output behavior of nonlinear dynamical systems. Several alternate formulations, that enforce minimal parameterization of the RBFN parameters are presented. An Extended Kalman Filter algorithm is incorporated to estimate the model parameters using multipe windows of the batch input-output data. The efficacy of the learning algorithms are evaluated on judiciously constructed test data before implementing them on real aerodynamic lift and pitching moment data obtained from experiments on a Synthetic Jet Actuation based Smart Wing.

Introduction

There is a significant thrust in the aerospace industry to develop advanced nano technologies that would enable one to develop adaptive, intelligent, shape controllable micro and macro structures, for advanced aircraft and space systems. These designs involve precise control of the shape of the structures with micro and nano level manipulations (actuation). The issue at hand is to derive comprehensive mathematical models that capture the input output behavior of these structures so that one can derive automatic control laws that can command desired shape and behavior changes in the structures. The development of the models using first principles, from classical mechanics fails at micro and nano scales and quantum mechanics applies only at scales less than pico-level. Thus, there is a lack of a unified modelling approach to derive macro models from those at the nano and micro scales. While the conventional modelling approaches evolve to handle these problems, one can pursue non parametric, multi-resolution, adaptive input/output modelling approaches to capture macro static and dynamic models directly from experiments. The purpose of this paper is to present an algorithm to learn a non-parametric mathematical model based upon radial basis functions that in essence aggregates information from large number of sensor measurements distributed over the structure. This aggregated information can be used to distribute actuation at specific points of the structure to achieve a desired shape. We will show the application of this algorithm to learn the mapping between the synthetic jet actuation parameters (frequency, direction, etc. for each actuator) and the resulting aerodynamic lift, drag, and moment.

Synthetic jet actuators (SJA) are devices used for active flow control that enable enhanced performance of conventional aerodynamic surfaces at high angles of attack and in many cases could lead to full replacement of hinged control surfaces thereby achieving perfect hingeless control. This active flow control is achieved by embedding sensors and actuators at nano and micro scales on an aerodynamic structure as shown in figure 1. The desired force and moment profile is achieved by impinging a jet of air using these actuators and thereby creating a desired pressure distribution over the structure. The distinguishing feature of synthetic jet actuation problem is that the relationship between input and output variables is not well known and is nonlinear in nature. Further, unsteady flow effects make it impossible to capture the physics fully from static experiments. The central difficulty in learning input/output mapping lies in choosing appropriate basis functions. While the brute force approach of using infinitely many basis functions is a theoretical possibility, it is intractable in a practical application because...
such an estimator will have far too many parameters to determine from limited number of observations. Alternatively, one can get around this problem by making use the prior knowledge of the problem's approximate physics and then supplementing it with an adaptation/learning algorithm that learns (compensates for) only the unknown modelling errors.

As one of the ways to facilitate such learning, the past two decades has seen the emergence and then significant advances in Artificial Neural Networks (ANNs) in areas of pattern classification, signal processing, dynamical system modelling and control. Neural networks have shown in some applications the ability to learn behavior where traditional modelling is difficult or impossible. However, the ANN approach is most definitely not a panacea! The traditional ANNs still have serious short-comings like

1. Abstraction: the estimated weights do not have physical significance.

2. Interpolation versus Extrapolation: How do we know when a given estimated model is sufficiently well-supported by the network having converged, and utilizing sufficiently dense and accurate measurements neighboring the desired evaluation point?

3. Issues Affecting Practical Convergence: A priori learning versus on-line adaptation? Actually, when the ANN architecture is fixed a priori, then the family of solvable problems is implicitly constrained that means the architecture of the network should be learned to ensure efficient and accurate modelling of the particular system behavior.

In this paper, we present an algorithm for learning an ideal two-layer neural network with radial basis functions as activation functions known as a Radial Basis Function Network (RBFN), to approximate the input-output response of synthetic jet actuators (SJA) based wing planform. The structure of the paper is as follows: a brief introduction to several existing learning algorithms will be provided followed by the details of the suggested learning algorithm. Finally, the performance of the learning algorithm will be demonstrated by different simulation and experimental results.

**Intelligent Radial Basis Function Networks**

In the past two decades, neural networks (NN) have emerged as a powerful tool in the areas of pattern classification, time series analysis, signal processing, dynamical system modelling and control. The emergence of NN can be attributed to the fact that they are able to learn behavior when traditional modelling is very difficult to generalize. While the successes have been many, there are also drawbacks of various fixed architecture implementations paving way for the necessity of improved networks that monitor the "health" of input-output models and learning algorithms. Typically, a neural network consists of many computational nodes called perceptrons arranged in layers. The number of hidden nodes (perceptrons) determine the degrees of freedom of the non-parametric model. A small number of hidden units may not be enough to capture the the complex input-output mapping and large number of hidden units may overfit the data and may not generalize behavior. Further, the optimal number of hidden units depends upon a lot of factors like - number of data points, signal to noise ratio, complexity of the learning algorithms etc. Beside this, it is also natural to ask "how many hidden layers are required to model the input-output mapping?" The answer to this question is provided by Kolmogorov's theorem.¹

Kolmogorov's Theorem. Let \( f(x) \) is a continuous function defined on a unit hypercube \( I^n (I = [0,1] \) and \( n \geq 2) \) then there exist simple functions \( \phi_j \) and \( \psi_{ij} \) such that \( f(x) \) can be represented in following form:

\[
    f(x) = \sum_{i=1}^{2n+1} \phi_j \left( \sum_{j=1}^{d} \psi_{ij}(x_j) \right)
\]

The relationship of Kolmogorov's theorem to practical neural networks is not straightforward as the functions \( \phi_j \) and \( \psi_{ij} \) can be very complex and not smooth as favored by NN. But Kolmogorov's theorem (later modified by other researchers²) can be used to prove that any continuous function from input to output can

¹Should not be confused with the literal meaning of the word "simple"...
be approximated by a two layer neural network. A more intuitive proof can be constructed by the fact that any continuous function can be approximated by an infinite sum of harmonic functions. Another analogy is the reference to bump or delta functions i.e. a large number of delta functions at different input locations can be put together to give the desired function. Such localized bumps might be implemented in a number of ways for instance by Radial basis functions.

Radial basis function based neural networks are two-layer neural networks with node characteristics described by radial basis functions. Originally, they were developed for performing exact multivariate interpolation and the technique consists of approximating a non-linear function as the weighted sum of a set of radial basis functions.

\[ f(x) = \sum_{i=1}^{h} w_i \phi_i(||x - \mu_i||) = w^T \Phi(||x - \mu||) \]  

where, \( x \in \mathbb{R}^n \) is an input vector, \( \Phi \) is a vector of \( h \) radial basis functions with \( \mu_i \) as the center of \( i^{th} \) radial basis function and \( w \) is a vector of linear weights or amplitudes. The two layers in RBFN perform two different tasks. The hidden layer with radial basis function performs a non-linear transformation of the input space into a high dimensional hidden space whereas the outer layer of weights performs the linear regression of this hidden space to achieve the desired value. The linear transformation followed by a non-linear one is summarized in Cover's theorem as follows,

**Cover's Theorem.** A complex pattern classification problem or input/output problem cast in a high-dimensional space is more likely to be approximately linearly separable than in a low-dimensional space.

According to Cover and Kolmogorov's theorems Multilayered Neural networks (MLNN) and RBFN can serve as “Universal Approximators”. While MLNN performs a global and distributed approximation at the expense of high parametric dimensionality, RBFN gives a global approximation but with locally dominant basis functions.

The main feature of radial basis functions (RBF) is that they are locally dominant and their response decreases (or increases) monotonically with distance from their center. This is ideally suited to modelling input-output behavior that shows strong local influence as with the synthetic jet actuation experiment.

Some examples of RBF are:

1. Thin-plate-spline function: \( \phi(r) = r^2 \log r \)
2. Multiquadric function: \( \phi(r) = (r^2 + \sigma^2)^{1/2} \)
3. Inverse Multiquadric function: \( \phi(r) = (r^2 + \sigma^2)^{-1/2} \)
4. Gaussian function: \( \phi(r) = \exp(-r^2/\sigma^2) \)

where \( r \) represents the distance between a center and the data points, usually taken to be Euclidean distance. \( \sigma \) is a real variable; for Gaussian functions it is a measure of the spread of the function. Among the above mentioned RBF, the Gaussian function is most widely used because its response can be confined to local dominance without altering the global approximation. Beside this, the shape of the Gaussian functions can also be adjusted appropriately by altering the parameters appearing in its description.

To summarize in general, the parameters needed to construct an RBFN can be enumerated as follows:

1. Number of RBF, \( h \)
2. The center of RBF, \( \mu_i \)
3. The spread of RBF (\( \sigma_i \) in case of Gaussian function)
4. The linear weights between hidden layer and the output layer, \( w_i \)

An adaptable, intelligent network would seek to update/learn some or all of the above mentioned parameters. Most importantly, the different learning parameters of RBFN live in the space of the inputs thereby enabling a physical interpretation of the parameters. This adaptation of the architecture of RBFN leads to a new class of approximators suitable for multi-resolution approximation applications. Conventionally (historically), the following form for the \( n \)-dimensional Gaussian functions is adopted,

\[ \phi(x, \mu_i, \sigma_i, q_i) = \exp\left(-\frac{1}{2}(x - \mu_i)^T P_i^{-1}(x - \mu_i)\right) \]

where, \( P_i^{-1} = \text{diag}(\sigma_{1i}^2 \cdots \sigma_{ni}^2) \). To learn the parameters mentioned above, different learning algorithms have been developed in the literature.

Poggio and Girosi introduced the traditional regularization technique to learn these parameters. Their RBFN has two layers with fixed number of hidden units (nodes) and center of hidden units chosen as a subset of the input samples. Algorithms such as forward selection can be used to find that subset. The linear weights connecting hidden layer to output layer can be found by Gradient Decent methods. The main disadvantage of this particular approach is the high computational cost involved. Besides this, a judicious choice of initial guess for weights is required as the algorithm can get stuck at local minima.

Moody and Darken introduced a low computation cost method which involves the concept of locally tuned neurons. Their algorithm takes the advantages of local methods conventionally used for density estimation, interpolation and approximation. Here too, the number of hidden units are chosen a priori. They used a standard \( k \)-means clustering algorithm to estimate the centers of the RBF and computed the
width values using various \textit{N nearest-neighbor} heuristics. While \textit{k-means} is suitable for pattern classification, it may not guarantee good results for function approximation because two samples close to each other in input space do not necessarily have similar outputs.\textsuperscript{5}

In 1991, Chen\textsuperscript{9} proposed an algorithm, known as \textit{Orthogonal Least Squares} (OLS) which makes use of Gram-Schmidt type orthogonal projection to select the best centers at a time. Starting from a large pool of candidate centers, OLS selects the predetermined number of centers that result in the largest reduction of error at output. Since it is not necessary that predetermined number of hidden units will always give us good approximation, Lee and Kil\textsuperscript{10} proposed a \textit{Hierarchically Self-Organizing} learning algorithm which is capable of automatically recruiting new hidden units whenever necessary.

Also in 1991, Platt\textsuperscript{11} proposed a sequential learning RBFN known as \textit{Resource Allocating Network} (RAN). The network proposed by Platt learns by allocating a new hidden unit or adjusting the parameters of existing hidden units for each input data. If the network performs well on a presented pattern, then the network parameters are updated using standard least mean squares gradient descent otherwise a new hidden unit is added to correct the response of the earlier network. A variation of this algorithm\textsuperscript{12} using extended Kalman filter\textsuperscript{13} for parameter adaptation is proposed by N. Sundararajan known as MRAN (Modified Resource Allocating Network). The advantages of RAN over any other learning algorithms can be summarized as follows.

- It is inherently sequential in nature and therefore can be used recursively in real-time to update the estimated model.
- The network architecture itself is adapted in contrast to adjusting weights in a fixed architecture network.

The adaptive architecture feature and the inherent recursive structure of the learning algorithm makes this approach ideal for multi-resolution modelling.\textsuperscript{10,12,14} While the methodology is very effective, it suffers from the drawback of potential explosion in the number of basis functions utilized to approximate the functional behavior. The reason for this stems from the fact that almost always, the basis functions are chosen to be circular. In some cases, the widths of the basis functions are chosen to be different. While this aids in improving the resolution, it does not significantly help in the reduction of the number of basis functions required. To overcome this problem, a pruning strategy is used posteriori\textsuperscript{4} but the convergence of network size is not guaranteed.

In the next section, we will propose an “Intelligent” scheme that sequentially learns the orientation of the data set with a fresh batch of data (can be in real time) and changes the orientation of the basis function, along with tuning of the centers and widths to enlarge the scope of a single basis function to approximate as much of the data possible. We see that this helps in reducing the overall number of basis functions and improving the function approximation accuracy. The orientation of the radial basis function can be modelled through a rotation parameter which for the two and three dimensional cases can be shown to the tangent of the half angle of the principal rotation vector.

Direction Dependent Approach

In this section, we present a novel learning algorithm for RBFN learning that is motivated through developments in rigid body rotational kinematics. The development is novel because of the application of the rotation ideas to the function approximation problem.

We try to move as well as rotate the Gaussian basis function to expand coverage, thereby reducing the total number of basis functions required for learning.

We propose adoption of the following \textit{n-dimensional} Gaussian function:

$$\Phi_i(x, \mu_i, \sigma_i, q_i) = \exp\left(-\frac{1}{2} (x - \mu_i)^T P_i^{-1} (x - \mu_i)\right)$$  \hspace{1cm} (4)

Where, \( P \) is \( n \times n \) fully populated symmetric positive definite matrix instead of diagonal one as in the case of conventional Gaussian function given by equation (3).

Now using spectral decomposition \( P^{-1} \) can be written as:

$$P_i^{-1} = C(q_i) S(\sigma_i) C^T(q_i)$$  \hspace{1cm} (5)

Where \( S \) is a diagonal matrix containing the eigenvalues, \( \sigma_i \) of covariance matrix \( P_i \), which dictates the spread of Gaussian function \( \Phi_i \). \( C(q_i) \) is a \( n \times n \) orthogonal rotation matrix. Though \( C(q_i) \) is a \( n \times n \) square matrix, we require only \( \frac{n(n-1)}{2} \) minimal parameters to describe it due to the orthogonality constraint. To enforce this constraint, we parameterize the matrix \( C(q_i) \) using the rotation parameter through the following result in matrix theory that is widely used in attitude kinematics namely, the Cayley Transform.\textsuperscript{15}

Cayley Transformation. \textit{If} \( C \in \mathbb{R}^{n \times n} \) \textit{is an orthogonal matrix and} \( Q \in \mathbb{R}^{n \times n} \) \textit{is an skew-symmetric matrix then the following transformations hold:}

\begin{enumerate}
  \item \textbf{Forward Transformations} \\
  \hspace{1cm} (a) \( C = (I - Q)(I + Q)^{-1} \) \\
  \hspace{1cm} (b) \( C = (I + Q)^{-1}(I - Q) \)
  \item \textbf{Inverse Transformations} \\
  \hspace{1cm} (a) \( Q = (I - C)(I + C)^{-1} \) \\
  \hspace{1cm} (b) \( Q = (I + C)^{-1}(I - C) \)
\end{enumerate}
As any arbitrary proper orthogonal matrix can be substituted into the above written transformations, Cayley Transformations can be used to parameterize the entire $O(n)$ group by skew symmetric matrices. The forward transformation is always well behaved however the inverse transformation encounter difficulty only near $180^\circ$ rotation. Thus as per the Cayley transformation, we can parameterize the orthogonal matrix $C(q_i)$ in equation (5) as:

$$C(q_i) = (I + Q)^{-1}(I - Q)$$

where, $q_i$ is a vector of distinct elements of skew symmetric matrix $Q$ i.e. $Q = -Q^T$. In addition to the parameters mentioned in last section, we now have to learn the additional parameters characterizing the orthogonal rotation matrix making total $\frac{(n+2)(n+1)}{2}$ parameters per Gaussian function for $n$ input single output system.

1. $n$ parameters for center of the Gaussian function i.e. $\mu$.
2. $n$ parameters for spread of Gaussian function i.e. $\sigma$.
3. $\frac{n(n-1)}{2}$ parameters for rotation of Gaussian function.
4. Weight $w_i$ corresponding to one output.

We shall develop learning algorithms for this extended parameter set. To our knowledge, this parameterization is unique and preliminary studies indicate a significant reduction in the number of basis functions required to accurately model functional behavior of the actual input output data.

The main feature of the proposed learning algorithms is the judicious choice for the location of the RBFs via a Directed Connectivity Graph approach which allows a priori adaptive sizing of the network for off-line learning and zeroth order network pruning. Beside this, Direction Dependent scaling and rotation of basis functions is provided for maximal trend sensing with minimal parameter representations and adaptation of the network parameters is done to account for on-line tuning.

Directed Connectivity Graph

The first step towards obtaining a zeroth order off-line model is the judicious choice of a set of basis functions and their locations, followed by proper initialization of the parameters. This exercise is the focus of this section.

To find the local extremum points of a given surface data, we divide the input space into several subspaces with the help of a priori prescribed hypercubes and find the relative maximum and minimum in each subspace. Now the set of extremum points of the surface data will be the subset of these relative maxima and minima for a particular size of the hypercube. Further, to choose centers out of these relative maxima and minima we construct directed graphs $\mathcal{M}$ and $\mathcal{N}$ of all the relative maxima sorted in descending order and all the relative minima sorted in ascending order respectively. We then choose the first points in $\mathcal{M}$ and $\mathcal{N}$ as candidates for Gaussian function centers with the function value as the corresponding starting weight of the Gaussian functions. The initial value of the covariance matrix $P$ is found by applying a local mask around the chosen center. Now using all the input data, we adapt the parameters of the chosen Gaussian functions sequentially using the extended Kalman filter\textsuperscript{13,16} and check the error residuals for estimation error. If the error residuals do not satisfy a predefined bound, we choose the next points in the directed graphs $\mathcal{M}$ and $\mathcal{N}$ as additional Gaussian RBF and repeat the whole process.

Notice, that the above choice of the Gaussian functions and the location of the centers is set around the fact that Gaussian functions are log-concave and this construction facilitates the evaluation of the extremal points.

**Definition 1.** A function $f : \mathcal{D} \subseteq \mathbb{R}^n \to \mathcal{R} \subseteq \mathbb{R}$ is concave if $-f$ is a convex function i.e. if $\mathcal{D}$ is a convex set and $\forall x, y \in \mathcal{D}$ and $\theta \in (0, 1)$, we have

$$-f(\theta x + (1 - \theta)y) \leq \theta(-f(x)) + (1 - \theta)(-f(y))$$

In other words $f(x)$ is a concave function if a line segment joining $(x, f(x))$ and $(y, f(y))$ lies below the graph of $f(x)$. Further if $f$ is a differentiable function then it can be shown that equation (7) is equivalent to following condition:

$$f(y) \leq f(x) + \frac{\partial f^T}{\partial x} (y - x)$$

The above-mentioned inequality leads to the following important property of a concave function.$^{17}$

**Lemma.** Let a function $f : \mathcal{D} \subseteq \mathbb{R}^n \to \mathcal{R} \subseteq \mathbb{R}$ is concave and differentiable then a point $x \in \mathcal{D}$ is a global maximum if $\frac{\partial f}{\partial x} |_x = 0$.

**Proof.** The proof can be found in Ref. 17

These results along with the following definition and properties of the log-concave function.$^{17}$ provides a theoretical basis for the specific choice of the RBF in this paper.

**Definition 2.** A function $f : \mathcal{D} \subseteq \mathbb{R}^n \to \mathcal{R} \subseteq \mathbb{R}$ is log-concave if $f(x) > 0$ for all $x \in \mathcal{D}$ and $\log f$ is concave.

Thus, to choose the location of the RBF centers, we make use of the fact that a Gaussian function is log-concave in nature and the response of the logarithm of
the Gaussian function is maximum at its center making the center of this function the extremum point i.e. \( \frac{d \log \Phi}{dx} |_{x=\mu} = 0 \) as per the above-mentioned lemma. Further, since log is a monotonically increasing function the center of the Gaussian function is also an extremum point of the Gaussian function. Therefore, all the extremum points of the given surface data should be the first choice for centers of Gaussian function with spread determined to first order by the covariance of the data confined in local mask around particular extremum point.

Though this whole process of finding the centers and evaluating the local covariance followed by the function evaluation with adaptation and learning seems computationally extensive it helps in reducing the total number of Gaussian functions and keeping the “curse of dimensionality” in check. Further, the rotation parameter of Gaussian function enables us to approximate the function with a greater accuracy. Since we use the Kalman filter to learn the parameters of the RBF network, the selection of centers can be made off-line with some experimental data and the same algorithm can be invoked online to adapt the parameters of off-line network. Any new Gaussian centers can be added to the existing network depending upon the statistics information of approximation errors. Additional localization and reduction in computational burden can be achieved by exploiting the local dominance near a given point, on only a small subset of RBFN parameters. Further information on the online version of the algorithm is presented in Ref.18.

### Extended Kalman Filter

Kalman filtering is a relatively recent (1960) development in the field of estimation.\(^{13,16}\) However, it has its roots as far back as in Gauss’s work in the 1800’s. The only difference between the Kalman filter and the sequential version of the Gaussian least squares is that the Kalman filter uses a dynamical model of plant to propagate the state estimates and corresponding error covariance matrix between two measurement sets. The implementation equations for the extended Kalman filter or “Kalman-Schmidt filter” are given in Table 1. The main advantage of the extended Kalman filter is that the nominal trajectory about which linearization takes place can be defined in real time. However, we have to pay the extra computational cost for linearization.

The sensitivity matrix \( H \) for the problem at hand is defined as:

\[
H = \frac{df(x, \mu, \sigma, q)}{d\Theta}
\]

where, \( f(x, \mu, \sigma, q) = \sum_{i=1}^{N} w_i \Phi_i(\mu_i, \sigma_i, q_i) \) and \( \Theta \) is a \( N \times \frac{(n+1)(n+2)}{2} \) vector given by:

\[
\Theta = \{ w_1 \quad \mu_1 \quad \sigma_1 \quad q_1 \quad \cdots \quad w_N \quad \mu_N \quad \sigma_N \quad q_N \} \tag{10}
\]

The partial derivatives required for the computation of the sensitivity matrix, \( H \) are given as follows:

\[
\frac{\partial f}{\partial u_k} = \phi_k
\]

\[
\frac{\partial f}{\partial \mu_k} = \left[ w_k \phi_k P_k^{-1} (x - \mu_k) \right]^T
\]

\[
\frac{\partial f}{\partial \sigma_{ki}} = w_k \phi_k \frac{y_i^2}{\sigma_k}, \quad y_i = C_k(x - \mu_k), \tag{13}
\]

\[
\frac{\partial f}{\partial q_{ki}} = -w_k \phi_k \left[(x - \mu_k)^T \frac{\partial C_k^T \Gamma_k C_k (x - \mu_k)}{\partial q_{ki}}\right] + (x - \mu_k)^T \Gamma_k \frac{\partial C_k}{\partial q_{ki}} (x - \mu_k)), \tag{14}
\]

Notice that, \( \frac{\partial C_k}{\partial q_{ki}} \) in equation (14) can be computed.
by substituting for $C$ from equation (6):
\[
\frac{\partial C_k}{\partial q_{k_i}} = \frac{\partial}{\partial q_{k_i}} (I - Q_k)^{-1} (I + Q_k) \\
+ (I - Q_k)^{-1} \frac{\partial}{\partial q_{k_i}} (I + Q_k) 
\]

Making use of the fact that $(I - Q)^{-1} (I - Q) = I$, we get
\[
\frac{\partial}{\partial q_{k_i}} (I - Q_k)^{-1} = (I - Q_k)^{-1} \frac{\partial Q_k}{\partial q_{k_i}} (I - Q_k)^{-1} 
\]

substitution of equation (16) in equation (15) gives:
\[
\frac{\partial C_k}{\partial q_{k_i}} = (I - Q_k)^{-1} \frac{\partial Q_k}{\partial q_{k_i}} (I - Q_k)^{-1} (I \\
+ Q_k) + (I - Q_k)^{-1} \frac{\partial Q_k}{\partial q_{k_i}} (I - Q_k)^{-1} (I \\
+ Q_k) + (I - Q_k)^{-1} \frac{\partial Q_k}{\partial q_{k_i}} (I - Q_k)^{-1} 
\]

Now equations (11)-(14) constitute the sensitivity matrix $H$ for the Extended Kalman Filter. We mention that although equation (5) provides a minimal parametrization of the covariance matrix $P$, it is highly nonlinear in nature and causes problems in the convergence of the Kalman filter in certain cases. To alleviate this potential difficulty and improve computational speed, we present an alternate representation of the covariance matrix $P$ known as Additive Decomposition.

Additive Decomposition of Covariance Matrix $P$

In this approach, we introduce the following parameterization of positive definite matrices:

Additive Decomposition. Let $P$ be a symmetric positive definite $n \times n$ matrix then $P^{-1}$ is also symmetric and positive definite and can be written as a sum of a diagonal matrix and a symmetric matrix:
\[
P_k^{-1} = \Gamma_k + \sum_{i=1}^{n} \sum_{j=1}^{n} e_i e_j^T q_{k_{ij}} 
\]

where $e_i$ is a $n \times 1$ vector with only $i^{th}$ element equal to one and rest of them zeros and $\Gamma_k$ is a diagonal matrix given by:
\[
\Gamma_k = \frac{1}{\sigma_k^2} I 
\]

subject to following constraints:
\[
q_{k_{ij}} = q_{k_{ji}} 
\]

$\sigma_k > 0$  

$q_{k_{ii}} > 0$  

$-1 \leq \frac{q_{k_{ij}}}{(\sigma_k + q_{k_{ij}})(\sigma_k + q_{k_{ji}})} \leq 1$  

It is worthwhile to mention that $q_{k_{ij}}$ signifies the stretching and rotation of the Gaussian function. If $q_{k_{ij}} = 0$ then we will obtain the circular Gaussian function.

Using the additive decomposition for the $P_i$ matrix in equation (4) the different partial derivatives required for synthesizing the sensitivity matrix $H$ can be computed by defining the following parameter vector $\Theta$
\[
\Theta = \{ w_1 \mu_1 \sigma_1 q_1 \cdots w_N \mu_N \sigma_N q_N \} 
\]

The different partial’s are then given as follows:
\[
\frac{\partial f}{\partial w_k} = \phi_k 
\]

\[
\frac{\partial f}{\partial \mu_k} = [w_k \phi_k P_k^{-1} (x - \mu_k)]^T 
\]

\[
\frac{\partial f}{\partial \sigma_k} = w_k \phi_k (x_i - \mu_k)^2, i = 1 \ldots n 
\]

\[
\frac{\partial f}{\partial q_{k_{ij}}} = -w_k \phi_k (x_i - \mu_k) (x_j - \mu_k), 
\]

$l = 1 \ldots n(n + 1)/2, i,j = 1 \ldots n$.  

Thus, equations (25)-(28) constitute the sensitivity matrix $H$. It is to be mentioned that even though the synthesis of the sensitivity matrix is greatly simplified, one needs to check the constraint satisfaction defined in equations (20)-(23) at every update. In case these constraints are violated, we invoke the parameter projection method to project the parameters back to the sets they belong to, thereby ensuring that the covariance matrix remains symmetric and positive definite at all times.

The various steps for the Directed Connectivity Graph Learning Algorithm are summarized as follows:

1. Find the interior extremum points of the given surface-data.
2. Divide the Input space into several subspaces with the help of equally spaced $1-D$ rays so that extremum points do not fall on the boundary of any subregion.
3. Find the relative maximum and minimum in each region.
4. Make a directed graph of all the maximum points sorted in descending order and call it M.
5. Make a directed graph of all the minimum points sorted in ascending order and call it N.
6. Choose first point in M and N as a candidate for Gaussian center and function values as the weight of those Gaussian functions.
7. For these points find the associated covariance matrix ($P$) with the help of local mask.
8. Initialize $q_{ij} = P_{ij}$ and $\sigma = 0$.  

7 OF 13  

AMERICAN INSTITUTE OF AERONAUTICS AND ASTRONAUTICS Paper 2004-0774
9. Learn $w, \mu, \sigma, g_{ij}$ using extended Kalman filter (Table 1) with the help of whole data.

10. Make sure that new estimated parameter vector satisfy the constraints given by equations (20-23)

11. Check the estimation error residuals. If they do not satisfy the required accuracy limit then choose second point in set $M$ and $N$ as Gaussian center and follow from step 7.

### Numerical Simulations and Results

This algorithm was tested on a variety of test functions and experimental data obtained by wind tunnel testing of synthetic jet actuation wing. In this section, we will present some results from the studies, importantly a test case for function Approximation and a dynamical System identification from wind tunnel testing of synthetic jet actuation wing.

#### Function Approximation

The test case for the function approximation is motivated by the following analytic surface function.\(^{19}\)

\[
 f(x_1, x_2) = \frac{1000}{(x_2 - x_1)^2 + (1 - x_1)^2 + 1} 
 + \frac{500}{(x_2 - 8)^2 + (5 - x_1)^2 + 1} 
 + \frac{500}{(x_2 - 8)^2 + (8 - x_1)^2 + 1} 
\]  

(29)

A random sampling of the interval \([0 – 10, 0 – 10]\) for $x_1$ and $x_2$ is used to obtain 60 samples of each. Figures 2(a) and 2(b) show the true surface and contour plots of the training set data points respectively. According to our experience this particular function has many important features such as sharp ridge line that is very difficult to learn with many existing function approximation algorithms with reasonable number of nodes. The failure of these many RBFN learning algorithms can be attributed to the inability of the circular Gaussian function to approximate a ridge kind of surface globally.

To approximate the function given in equation (29), we divide the whole input region into a total of 16 square regions (4 in each direction). Then according to the procedure listed in the previous section, we generate a directed connectivity graph of the local maxima and minima in each sub-region that finally add up to 32 radial basis functions to have approximation errors less than 5%. Figures 2(c) and 2(d) show the estimated surface and contour plots respectively. From these figures, it is clear that we are able to learn the analytical function given in equation (29) very well. Figures 2(e) and 2(f) show the error surface and error contour plots for the RBFN approximated function. From figure 2(e), it is clear that approximation errors are less than 5% whereas from figure 2(f) it is clear that even though we approximate the ramp surface very well most of the approximation errors are confined to this region alone.

#### SJ A Modelling

In this section, the RBFN modelling results for the synthetic jet actuator are presented. These results show the effectiveness of the directed connectivity graph learning algorithm in learning the input-output mapping for the synthetic jet actuation wing.

#### Experimental Set up

A Hingeless-Control-Dedicated experimental setup has been developed, as part of the initial effort, the heart of which is a stand-alone control unit, that controls all of the wing's and SJAs' parameters and variables. The setup is installed in the 3'x4' wind tunnel of the Texas A& M Aerospace Engineering Department (Figure 3). The test wing profile for the dynamic pitch test of the synthetic jet actuator is a NACA 0015 airfoil. This shape was chosen due to the ease with which the wing could be manufactured and the available interior space for accommodating the synthetic jet actuator (SJA).

Experimental evidence suggests that a SJA, mounted such that its jet exit tangentially to the surface, has minimal effect on the global wing aerodynamics at low to moderate angles of attack. The primary effect of the jet is at high angles of attack when separation is present over the upper wing surface. In this case, the increased mixing associated with the action of a synthetic jet, delays or suppresses flow separation. As such, the effect of the actuator is in the non-linear post stall domain. To learn this nonlinear nature of SJA experiments were conducted with the control-dedicated setup shown in figure 3. The wing angle of attack (AOA) is controlled by the following reference signal.

1. Oscillation type: sinusoidal Oscillation magnitude: 12.5°.

2. Oscillation offset (mean AOA): 12.5°

3. Oscillation frequency: from 0.2Hz to 2Hz.

In other words, the AOA of airfoil is forced to oscillate from 0° to 25° at a given frequency (see figure 4). The experimental data collected were the time histories of the pressure distribution on the wing surface (at 32 locations). The data was also integrated to generate the time histories of the lift coefficient and the pitching moment coefficient. Data was collected with the SJA on and with the SJA off (i.e. with and without active flow control). All the experimental data were taken for 5 sec at a 100 Hz sampling rate.

#### RBFN Modelling of Experimental Data

The experiments described above were performed at a freestream velocity of 25m/sec. From the surface
Fig. 2 Simulation Results For Analytical Function given by Equation 29
Fig. 3  Hingeless Control-Dedicated experimental setup for Synthetic Jet Actuation Wing

Fig. 4  Angle of Attack Variation.

a) Angle of Attack Variation without SJA.

b) Angle of Attack Variation with SJA actuation frequency of 60 Hz.
a) 0 Hz jet actuation frequency.

b) 60 Hz jet actuation frequency.

Fig. 5 Measured and Approximated Lift Coefficient.

Fig. 6 Approximation Error for Lift Coefficient.

Fig. 7 Measured and Approximated Pitching Moment Coefficient.
pressure measurements, the lift and pitching moment coefficients were calculated via integration. As the unknown SJA model is known to be dynamic in nature so SJA wing lift force and pitching moment coefficients are modelled by second order system i.e. they are assumed to be function of current and previous time states (angle of attack).

\[ C_L = C_L(a(t_k), a(t_{k-1})) \]  
\[ C_M = C_M(a(t_k), a(t_{k-1})) \]  

(30)  
(31)

The Directed Connectivity Graph Learning Algorithm was used to learn the unknown nonlinear behavior of SJA wing described by equations (30) and (31). The input space (angle of attack) is divided into \( 2 \times 2 \) grid giving us a freedom to choose maximum 8 radial basis functions. Figures 5(a) and 5(b) show the measured and RBFN approximated lift coefficient for zero and 60 Hz jet actuation frequency respectively with 8 radial basis functions. Figures 6(a) and 6(b) show the corresponding approximation error plots. From these figures, it is clear that we are able to learn the nonlinear relationship between lift coefficient and angle of attack with and without SJA on.

Similarly to model the pitching moment measurements, we again divide the input space into \( 2 \times 2 \) grid and finally pick up total of 8 basis functions to approximate the experimental data. Figures 7(a) and 7(b) show the measured and RBFN approximated pitching moment coefficient for zero and 60 Hz jet actuation frequency respectively. Figures 8(a) and 8(b) show the corresponding approximation error plots. From these figures, it is clear that we are able to learn the nonlinear relationship between moment coefficient and angle of attack (with and without SJA on) very well within experimental accuracy. From these plots, we can conclude that although the approximation error magnitude is slightly higher, the Directed Connectivity Graph Learning Algorithm has done a very good job in learning the overall behavior of the pitching moment and lift force.

**Concluding Remarks**

We remark that a reliable RBFN learning algorithm has been developed for continuous function approximation. The same approach can also be used for dynamic system modelling. Results presented in this paper serve to illustrate the usefulness of Directed Connectivity Graph Learning Algorithm in function approximation. The rotation of the Gaussian functions not only helps us in approximating the complex surfaces but also help in reducing the numbers of hidden units significantly. However substantial research is required to extend and optimize the methodology for multi-resolution approximation in high dimensional spaces. It is significant that the purpose of the approximation is to enable adaptive control. Therefore, we anticipate the rate of learning and dimensionality to be significant challenges that are yet to be resolved.

**Acknowledgement**

This work was sponsored by AFRL through the DOD SBIR Program (in collaboration with Aeroprobe Corp.). The authors would like to thank Drs. James Myatt and Chris Camphouse, the technical monitors of the SBIR project. Also, the authors wish to acknowledge the support of the Texas Institute for Intelligent Bio-Nano Materials and Structures for Aerospace Vehicles, funded by NASA Cooperative Agreement No. NCC – 1 – 02038.

**References**


16Singla, P., “A New Attitude Determination Approach using Split Field of View Star Camera,” Masters Thesis report, Aerospace Engineering, Texas A&M University, College Station, TX, USA.


