AIAA 2004-0774
Intelligent Multi-Resolution Modelling: Application to Synthetic Jet Actuation and Flow Control
Puneet Singla, Kamesh Subbarao, John L. Junkins, Othon Rediniotis
Texas A & M University, 3141-TAMU, College Station, TX 77843-3141

42nd AIAA Aerospace Sciences Meeting and Exhibit
January 5–8, 2004/Reno, NV

For permission to copy or republish, contact the American Institute of Aeronautics and Astronautics
1801 Alexander Bell Drive, Suite 500, Reston, VA 20191–4344
Intelligent Multi-Resolution Modelling:
Application to Synthetic Jet Actuation and Flow Control

Puneet Singla, Kamesh Subbarao, John L. Junkins, Othon Rediniotis
Texas A & M University, 3141-TAMU, College Station, TX 77843-3141

Introduction

There is a significant thrust in aerospace industry to develop advanced nano technologies that enable us to make adaptive, intelligent, shape controllable micro and macro structures, for both advanced aircraft and space systems. The design of such structures involves precise control of the shape at micro and nano level. The issue at hand is to derive comprehensive mathematical models that capture the input output behavior of these structures. The first principle approach 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 waiting for the evolution of physics-based models, we can pursue 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 one of the popular devices used for Active Flow Control which is quite useful in designing intelligent control structure integrating the functions of all the flight control actuators and also adapting efficiently over different flight regimes. Active flow control is achieved by embedding the sensors and actuators at nano and micro scales on the 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, un-steady 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. The brute force method will be to choose a complete (infinite) set of basis functions but such an estimator will have far too many parameters to determine from limited number of observations. Alternatively, we can use prior knowledge of the problem’s approximate physics to only learn only the unknown model errors.

In past two decades, Artificial Neural Networks (ANNs) have emerged in some areas of pattern classification, signal processing, dynamical modelling and
control. Neural networks have shown in some applications, 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 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.

In this paper, we will present an algorithm for the learning of an ideal two-layer neural network with radial basis functions as activation functions known as radial basis function network (RBFN), to approximate the input-output response of synthetic jet actuators (SJA). The structure of paper is as follows: first a brief introduction to different existing learning algorithm will be given 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 past two decades, neural networks have emerged as a powerful tool in the areas of pattern classification, time series analysis, signal processing, dynamical system modelling and control. While successes have been many, there are also drawbacks of various fixed architecture implementations. Improved networks are needed that monitor the “health” of input-output models and learning algorithms. Neural networks broadly have emerged because they are able to learn behavior when traditional modelling is very difficult to generalize. Typically, a neural network consists of many computational nodes called perceptrons arranged in layers. The number of hidden nodes (perceptrons) determine the degree of freedom of the non-parametric model. The 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. The optimal number of hidden units depend upon lot of factors like number of data points, noise to signal ratio, complexity of the learning algorithms. 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 \( I^n \) \( (I = [0,1] \text{ and } n \geq 2) \) then there exist simple functions \( \phi_j \text{ and } \psi_{ij} \) such that \( f(x) \) can be represented in following form:

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

The relationship of Kolmogorov’s theorem to practical neural network is not much clear as the functions \( \phi_j \) and \( \psi_{ij} \) can be very complex and not smooth as favored by Neural networks. But Kolmogorov’s theorem modified by others can be used to prove that any continuous function from input to output can be approximated by two layer neural network.\(^2\) A more instructive prove can be inspired by the fact that any continuous function can be approximated by infinite sum of harmonic functions. Other instructive analogy can be bump or delta functions i.e. a large number of delta functions at different input locations can be put together to give our desired function.\(^7\) Such localized bumps might be implemented in a number of way for instance by Radial basis functions.

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

\[
f(x) = \sum_{i=1}^{h} w_i \phi_i(\|x - \mu_i\|) = w^T \Phi(\|x - \mu\|) \tag{2}
\]

where \( \Phi \) 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 task. The hidden layer with radial basis function performs a non-linear transformation of input space into high dimensionality hidden space whereas outer layer of weights performs the linear regression of hidden space to achieve the desired value. The linear transformation followed by nonlinear one is based upon the famous Cover’s theorem.\(^2\)

**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 (MLP) and RBFN can serve as “Universal Approximators”. MLP performs
a global and distributed approximation but the bad news is that high parametric dimensionality of MLP is not exactly an advantage. On other side, RBFN gives a global approximation but with locally dominant basis functions.

The main feature of radial basis functions (RBF) is that their response decreases (or increases) monotonically with distance from their center. 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 it is taken to be Euclidean distance. \( \sigma \) is a real variable, for Gaussian functions it is a measure of the spread of the function. Among above mentioned RBF, the Gaussian function is most widely used because its response can be confined to local dominance without altering the global approximation. Besides this, the shape of Gaussian functions can also be adjusted appropriately. Generally, to construct a RBFN, we need to learn following four parameters:

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, \( w_i \)

Most importantly, the different learning parameters of RBFN lives in the space of inputs enabling the physical interpretation of the parameters. Adapting the architecture of RBFN leads to a new class of approximators suitable for multi-resolution approximation applications. Conventionally, the following form for the \( n \)-dimensional Gaussian functions is adopted,

\[
\Phi(x, \mu_i, \sigma_i, q_i) = \exp\left(-\frac{1}{2}(x-\mu_i)^T \text{diag}(\sigma_{1i}^2 \cdots \sigma_{ni}^2)(x-\mu_i)\right)
\]

To learn the parameters mentioned above, different learning algorithms have been developed in literature. Poggio and Girosi introduced the traditional regularization technique to learn these parameters. Their RBFN has two layers with the number of hidden units (nodes) fixed apriori 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 Descent methods. The main disadvantage of this particular approach is the high computational cost involved. Beside 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 apriori. They used a standard \( k \)-means clustering algorithm to estimate the centers of the RBF and computed the width values using various \( N \) nearest-neighbor heuristics. While \( k \)-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.

In 1991, Chen proposed a algorithm, known as 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 largest reduction of error at output.

It is not necessary that predetermined number of hidden units will always give us good approximation. Therefore, Lee and Kil proposed a Hierarchically Self-Organizing learning algorithm which is capable of automatically recruiting new hidden units whenever necessary.

In 1991, Platt proposed a sequential learning RBFN known as 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 using extended Kalman filter 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{9,11,13}

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 post\textsuperscript{3}ri but the convergence of network size is not guaranteed. In next section, we will propose an “Intelligent” scheme that sequentially learns the orientation of the data set in real time and changes the orientation of the basis function itself, 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 $n$-dimensional Gaussian function:

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

(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:

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

(5)

Where $S$ is a diagonal matrix containing the eigen values, $\sigma_i$s, of covariance matrix $P_i$ which dictate the spread of Gaussian function $\Phi_i$. $C(q_i)$ is $n \times n$ orthogonal rotation matrix. Although $C(q_i)$ is a $n \times n$ square matrix but due to orthogonality constraint we require only $\frac{n(n-1)}{2}$ minimal parameters to describe it. We will introduce one of the key result in attitude kinematics known as Cayley Transformation.\textsuperscript{14}

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

1. Forward Transformations

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

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

2. Inverse Transformations

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

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

As any arbitrary orthogonal matrix can be substituted into above written transformations so Cayley Transformations can be used to parameterize whole O($n$) group by skew symmetric matrices.

As per Cayley transformation, we can parameterize the orthogonal matrix $C(q_i)$ in equation (5) as:

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

(6)

where, $q_i$ is 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 parameters characterizing the orthogonal rotation matrix making total $\frac{n^2+2(n-1)}{2}$ parameters per Gaussian function for $n$ inputs 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 parametrization 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 features of the proposed learning algorithms is the judicious choice for the location of the RBFs via a Directed Connectivity Graph approach which allows a\textsuperscript{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

Before describing the procedure for the selection of RBF centers, we will give a brief introduction to concave functions and their properties.

**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:

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

The above written inequality is one of the most important property of a concave function and leads to one of the most famous result result about concave functions.\(^{15}\)

**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 iff \( \frac{\partial f}{\partial x} |_{x} = 0 \).

**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.

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

To find the extremum point of a given surface data, we divide the input space into several subspaces with the help of 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 maximum and minimum for a particular length of hypercube. Further, to choose centers out of these relative maximum and minimums we first make a directed graphs \( \mathcal{M} \) and \( \mathcal{N} \) of all the relative maximum points sorted in descending order and all the relative minimum points sorted in ascending order respectively and then choose first point in \( \mathcal{M} \) and \( \mathcal{N} \) as a candidate for Gaussian function center with function value as the corresponding weight of the Gaussian functions. The initial value of covariance matrix \( \mathbf{P} \) is found by taking a local mask around the chosen center. Now using all the input data, we adapt all the parameters of the chosen Gaussian function sequentially using the extended Kalman filter\(^{12,16}\) and check the error residuals for estimation error. If the error residuals do not satisfy the predefined bound then we choose the next point in directed graph \( \mathcal{M} \) and \( \mathcal{N} \) as another Gaussian function and repeat the whole process.

Although this whole process is computationally extensive but it helps in reducing the total number of Gaussian function and further rotation parameter of Gaussian function enable us to approximate the function with greater accuracy. As we use Kalman filter to learn the parameters of our RBF network therefore the selection of centers can be made off-line with some experimental data and 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. Further information on the online version of the algorithm is presented in.\(^{17}\)

**Extended Kalman Filter**

Kalman filtering is a relatively recent (1960) development in the field of estimation.\(^{12,16}\) However, it has its root as far as back Gauss (1795). The only difference between the Kalman filter and the sequential version of Gaussian least square is that the Kalman filter uses a dynamical model of plant to propagate the state estimates and corresponding error covariance matrix between two sets of measurements. In 1960, Stanley F. Schmidt proposed the application of the Kalman filter for systems involving non-linear dynamic and measurement models and it has been called the “Kalman-Schmidt filter”. The extended Kalman filter is based on the assumption that the estimated state is close to the true state and therefore the error dynamics can be represented as the first order Taylor series expansion of the non-linear error dynamics. However this assumption can be very fatal for the case of large initial errors or a highly non-linear system. The extended Kalman filter for non-linear systems uses the linearize state space model about the current estimates of the states to generate the current update at a measurement time, but propagates the estimates non-linearly 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.
We need to find the sensitivity matrix $\mathbf{H}$ defined as:

$$
\mathbf{H} = \frac{\partial f(x, \mu, \sigma, q)}{\partial \Theta}
$$

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

$$
\Theta = \left\{ w_1 \; \mu_1 \; \sigma_1 \; q_1 \; \cdots \; w_N \; \mu_N \; \sigma_N \; q_N \right\}
$$

Now the different partial’s for the computation of sensitivity matrix, $\mathbf{H}$ are given as follows:

$$
\frac{\partial f}{\partial w_k} = \phi_k \quad (11)
$$

$$
\frac{\partial f}{\partial \mu_k} = \left[ w_k \phi_k \mathbf{P}_k^{-1}(x - \mu_k) \right]^T \quad (12)
$$

$$
\frac{\partial f}{\partial \sigma_{ki}} = w_k \phi_k \frac{y_i^2}{\sigma_{ki}} \quad i = 1 \ldots n \quad (13)
$$

$$
\frac{\partial f}{\partial q_{ki}} = -\frac{w_k}{2} \phi_k \left[ (x - \mu_k)^T \frac{\partial \mathbf{C}_k}{\partial q_{ki}} \Gamma_k \mathbf{C}_k (x - \mu_k) \right] + \left( x - \mu_k \right)^T \mathbf{C}_k \Gamma_k \left( x - \mu_k \right),
$$

$$
l = 1 \ldots n(n-1)/2 \quad (14)
$$

Further, $\frac{\partial \mathbf{C}_k}{\partial q_{ki}}$ in equation (14) can be computed by substituting for $\mathbf{C}$ from equation (6):

$$
\frac{\partial \mathbf{C}_k}{\partial q_{ki}} = \frac{\partial}{\partial q_{ki}} (\mathbf{I} - \mathbf{Q}_k)^{-1} (\mathbf{I} + \mathbf{Q}_k) + (\mathbf{I} - \mathbf{Q}_k)^{-1} \frac{\partial}{\partial q_{ki}} (\mathbf{I} + \mathbf{Q}_k) \quad (15)
$$

Making use of the fact that $(\mathbf{I} - \mathbf{Q})^{-1} (\mathbf{I} - \mathbf{Q}) = \mathbf{I}$, we get

$$
\frac{\partial}{\partial q_{ki}} (\mathbf{I} - \mathbf{Q}_k)^{-1} = (\mathbf{I} - \mathbf{Q}_k)^{-1} \frac{\partial \mathbf{Q}_k}{\partial q_{ki}} (\mathbf{I} - \mathbf{Q}_k)^{-1}
$$

substitution of equation (16) in equation (15) gives:

$$
\frac{\partial \mathbf{C}_k}{\partial q_{ki}} = (\mathbf{I} - \mathbf{Q}_k)^{-1} \frac{\partial \mathbf{Q}_k}{\partial q_{ki}} (\mathbf{I} - \mathbf{Q}_k)^{-1} (\mathbf{I} + \mathbf{Q}_k) + (\mathbf{I} - \mathbf{Q}_k)^{-1} \frac{\partial \mathbf{Q}_k}{\partial q_{ki}} (\mathbf{I} - \mathbf{Q}_k)^{-1} \quad (17)
$$

Now equations (11)-(14) constitute the sensitivity matrix $\mathbf{H}$ for Kalman filter. We mention that although equation (5) provide all the minimal parametrization of covariance matrix $\mathbf{P}$ but it is highly nonlinear in nature and some time causes the problem in the convergence of the Kalman filter. Therefore, we come up with a novel way of representing the covariance matrix $\mathbf{P}$ known as additive approach.

Additive Decomposition of Covariance Matrix $\mathbf{P}$

In this approach, we introduce a new way to write a positive definite matrices:

Additive Decomposition. Let $\mathbf{P}$ is a positive symmetric $n \times n$ matrix then $\mathbf{P}^{-1}$ is also positive and symmetric matrix and can be written as a sum of a diagonal and symmetric matrix:

$$
\mathbf{P}_k^{-1} = \Gamma_k + \sum_{i=1}^{n} \sum_{j=1}^{n} e_i e_j^T q_{ki,j} \quad (18)
$$

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:

$$
\Gamma_k = \frac{1}{\sigma_k^2} \mathbf{I} \quad (19)
$$

subject to following constraints:

$$
q_{ki,j} = q_{kj,i} \quad (20)
$$

$$
\sigma_k > 0 \quad (21)
$$

$$
q_{ki,j} > 0 \quad (22)
$$

$$
-1 \leq \frac{q_{ki,j}}{(\sigma_k + q_{ki,j}^2)(\sigma_k + q_{kj,i}^2)} \leq 1 \quad (23)
$$

It is worthwhile to mention that $q_{ki,j}$ signifies the stretching and rotation of Gaussian function. If $q_{ki,j} = 0$ then we will get circular Gaussian function.

Now, using the additive decomposition for $\mathbf{P}_k$ matrix in equation (4) the different partial for sensitivity matrix $\mathbf{H}$ can be computed by defining the following parameter vector $\Theta$

$$
\Theta = \left\{ w_1 \; \mu_1 \; \sigma_1 \; q_1 \; \cdots \; w_N \; \mu_N \; \sigma_N \; q_N \right\}
$$

(24)
The different partial’s are given as follows:

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

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

\[
\frac{\partial f}{\partial \sigma_{k_i}} = w_k \phi_k \frac{(x_i - \mu_{k_j})^2}{\sigma_{k_i}^3}, i = 1 \ldots n
\]

\[
\frac{\partial f}{\partial q_{k_l}} = -w_k \phi_k (x_i - \mu_{k_j})^T (x_j - \mu_{k_j}), \\
l = 1 \ldots n(n+1)/2, i, j = 1 \ldots n.
\]

Therefore, equations (25)-(28) constitute the sensitivity matrix \( \mathbf{H} \). We also mention that whenever constraints defined in equations (20)-(23) are violated, we can invoke the parameter projection method to satisfy the constraints.

The various steps for the Directed Connectivity Graph Learning Algorithm are listed 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 \( 1 \rightarrow D \) 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 \( \mathbf{M} \).
5. Make a directed graph of all the minimum points sorted in ascending order and call it \( \mathbf{N} \).
6. Choose first point in \( \mathbf{M} \) and \( \mathbf{N} \) 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 \( \mathbf{P} \) with the help of local mask.
8. Initialize \( q_{ij} = P_{ij} \) and \( \sigma = 0 \).
9. Learn \( w, \mu, \sigma, q_{ij} \) 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 \( \mathbf{M} \) and \( \mathbf{N} \) 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 function approximation consist of approximating the following analytic surface function which is a slight modification of famous May-West function.\(^7\)

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

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 ridge which are very difficult to learn with many existing function approximation algorithm with reasonable number of nodes. The failure of many RBFN learning algorithms can be attributed to the inability of 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 total 16 square regions (4 in each direction). Then according to procedure listed in previous section, we make a directed connectivity graph of local maxima and minima’s in each sub-region and finally come up with total 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 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 have got the ramp surface very well but the approximation errors are more in that region only.

**SJA Modeling**

In this section, the RBFN modelling results for synthetic jet actuator are presented. These results show the effectiveness of the directed connectivity graph learning algorithm in learning the input-output mapping for synthetic jet actuation wing.
Fig. 2 Simulation Results For Analytical Function given by Equation 29

American Institute of Aeronautics and Astronautics Paper 2004-0774
Fig. 3  Hingeless Control-Dedicated experimental setup for 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 SJA’s 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 20Hz.

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.

The experiments described above were performed at a freestream velocity of 25m/sec. From the surface 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).

\[ C_L = C_L(\alpha(k), \alpha(k-1)) \]
\[ C_M = C_M(\alpha(k), \alpha(k-1)) \]

RBFN Modelling of Experimental Data

The Direct connectivity graph 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 × 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 × 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
Fig. 5 Measured and Approximated Lift Coefficient.

Fig. 6 Approximation Error for Lift Coefficient.

Fig. 7 Measured and Approximated Pitching Moment Coefficient.
experimental accuracy. From these plots, we can conclude that although approximation error magnitude is bit more but Directed connectivity graph algorithm has done a very good job in learning the overall behavior of 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 also. Results presented in this paper serve to illustrate the usefulness of Directed Connectivity Graph approach in function approximation. The rotation of Gaussian function 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 multiresolution 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 yet to be resolved.

Acknowledgement

This work has been partially supported by TiiMS project of URETI. We also acknowledge the grant provided by AFRL and Aeroprobe.

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.