

REAL-TIME LEARNING OF AIRCRAFT PARAMETERS USING RECURSIVE LEAST-SQUARES TO TRAIN RBF NETWORKS¹

J. NICHOLAS LANEMAN

Massachusetts Institute of Technology, Cambridge, Massachusetts, 02139

GERALD E. PETERSON

McDonnell Douglas Corporation, St. Louis, Missouri, 63166

ABSTRACT:

We describe a real-time algorithm for learning aircraft parameters to be used by an adaptive controller. Learning consists of training a collection of radial basis function neural networks to approximate the incoming data stream in the least-squares sense. Only the heights of the basis functions are trained; heuristics are used to find their centers and widths. Since the heights enter the equations linearly, we employ recursive least-squares to quickly obtain the new heights when we incorporate additional data points and basis functions. In order to keep the computations manageable, we break the data stream into segments, with each segment approximated by about 10 basis functions. We illustrate the algorithm on a set of F-15 flight data.

INTRODUCTION

Designing gains for aircraft flight controllers requires estimates of the stability and control derivatives, which are the primary components of the A and B matrices in a state-space model of the dynamics. Estimates of these derivatives are normally obtained from a database of aerodynamic information derived primarily from wind tunnel testing. Controller development proceeds with the gains being designed initially using the aerodynamic database and then refined using information obtained over many flight tests.

There is a desire in the aerospace community to provide a better response to changes in aerodynamic loading, control surface failures, or minor aircraft damage during flight by adapting the flight control laws in real time. However, this approach requires online estimates of the time-varying stability and control derivatives. To estimate these parameters, the aircraft dynamics must be excited in such a way that the parameters influence (are observable in) the sensor measurements. For example, a derivative with respect to α requires movement in the α dimension. During periods when accurate estimates are possible, they must be stored and made available for when accurate estimates are not possible. This storage function can be provided by a neural network.

¹This work was supported by the McDonnell Douglas Independent Research and Development Program.

A radial basis function (RBF) neural network is a good candidate for this online storage function because the learning can be localized to the area of current flight, and because these networks can be trained quickly. Localization along the trajectory is accomplished by placing the centers of the RBFs along the flight path in parameter space. The widths of the basis functions and their spacing are chosen in advance using heuristics, *e.g.*, requiring each point along the trajectory to be supported by at least four basis functions. With the widths and centers chosen, the heights of the RBFs can be quickly determined to satisfy the least sum-of-squared errors criterion by using the recursive least-squares methodology.

Recursive least-squares (RLS) is a method for maintaining a least-squares solution as new data points are added to the training cases or as new basis functions are added to the RBF network. RLS has been used extensively in adaptive filter theory (Haykin, 1991) and adaptive control (Åström and Wittenmark, 1989). If points or basis functions are added one at a time, no matrix inversions are required by this process; only matrix multiplications and additions are necessary. However, when a basis function is added, the size of the primary matrix grows by one row and one column. We take additional steps to limit the size of this matrix and ensure the possibility of a real-time implementation.

The remainder of this paper is organized as follows. In the next section we develop the RBF network model used in our approximations. We derive the RLS equations used to incorporate new basis functions and data points into the training model, and we discuss steps taken to reduce the amount of computation required to maintain the least-squares solution. Finally, we illustrate our algorithm by approximating aerodynamic derivatives from a segment of flight data.

TRAINING RBF NETWORKS USING RECURSIVE LEAST-SQUARES

Suppose data $\{z_i, y_i\}_{i=1}^n$ is given, where z_i is the input vector $[z_{i1}, \dots, z_{i\ell}]^T$ and the measurement y_i is a scalar. A *radial basis function* (RBF) neural network is a model of the form

$$y \approx b_1 f_1(z) + \dots + b_m f_m(z) \quad (1)$$

where the order of the model satisfies $m < n$ to achieve compression. Here $\{f_j(z), j = 1, \dots, m\}$ are the radial basis functions, which are derived from a radially symmetric function $f(z)$, called the *basic* RBF (Bishop, 1995). The vector of coefficients $b = [b_1 \dots b_m]$ must be determined so that the model fits the data in the least-squares sense.

Since the b_j occur linearly in (1), we may write the model as a set of linear equations. Specifically, let the measurement data vector be $y = [y_1 \dots y_n]^T$, and let $R = (r_{ij})$ be the $n \times m$ regression matrix defined by $r_{ij} = f_j(z_i)$. Then (1) becomes $y \approx Rb$, and we assume that the columns of R are linearly independent. To fit the data in the least-squares sense, that is, to find the heights such that

$$\hat{b} = \arg \min_b (Rb - y)^T (Rb - y)$$

requires

$$\hat{b} = [R^T R]^{-1} R^T y \quad (2)$$

which is the standard least-squares solution to an over-determined set of linear equations (Strang, 1980).

Adding a New Basis Function to the Model

Given a solution in the form of (2), suppose we wish to add an additional function $f_{m+1}(z)$ to the model and calculate the new solution with a minimum of additional computation. The regression matrix becomes $R' = [R H]$, where

$$H = [f_{m+1}(z_1) \quad \dots \quad f_{m+1}(z_n)]^T$$

Let

$$\hat{b}' = [\hat{b}'_1 \quad \dots \quad \hat{b}'_{m+1}]^T$$

be the new vector of coefficients to be determined. By (2),

$$\hat{b}' = \begin{bmatrix} R^T R & R^T H \\ H^T R & H^T H \end{bmatrix}^{-1} \begin{bmatrix} R^T \\ H^T \end{bmatrix} y \quad (3)$$

A calculation shows that

$$\begin{bmatrix} R^T R & R^T H \\ H^T R & H^T H \end{bmatrix}^{-1} = \frac{1}{\theta} \begin{bmatrix} \theta[R^T R]^{-1} - cc^T & c \\ c^T & -1 \end{bmatrix} \quad (4)$$

where $c = [R^T R]^{-1} R^T H$ and $\theta = (Rc - H)^T H$. Equation (4) is valid whenever $\theta \neq 0$. Substituting (4) into (3) gives

$$\hat{b}' = \begin{bmatrix} \hat{b} - \alpha c \\ \alpha \end{bmatrix} \quad (5)$$

in which

$$\alpha = \frac{(Rc - H)^T y}{\theta} \quad (6)$$

Thus (4) may be used to efficiently calculate the new inverse matrix from the old, and (5) may be used to obtain the new least-squares solution.

Adding a New Data Point to the Training Set

Given a solution in the form of (2), suppose it is desired to add an additional data point $\{z_{n+1}, y_{n+1}\}$ to the training set and calculate the new solution with a minimum of additional computation. In this case, the regression matrix becomes $R' = [R^T G^T]^T$, where

$$G = [f_1(z_{n+1}) \quad \dots \quad f_m(z_{n+1})]$$

and the output vector becomes

$$y' = [y \quad y_{n+1}]^T$$

Let $P = R^T R$, and $P' = P + G^T G$. Starting with (2) and utilizing the matrix inversion lemma (Haykin, 1991), we reduce the solution to

$$\hat{b}' = \hat{b} + K(y_{n+1} - G\hat{b}) \quad (7)$$

$$P'^{-1} = [I - KG]P^{-1} \quad (8)$$

where

$$K = P^{-1} G^T [GP^{-1} G^T + 1]^{-1} \quad (9)$$

Equations (7), and (8) may be used to find the new solution heights and the new inverse matrix from the old, all without matrix inversion.

Finding Centers and Widths

In our application, the data is a function of time. That is, $z_i = z(t_i)$ and $y_i = y(t_i)$ for some sequence of time values $\{t_i\}$. Therefore, the data points form a one-dimensional *trajectory* in an $\ell + 1$ space of parameter values. In order to give approximately equal treatment to each of the dimensions in parameter space, the input is scaled so that each parameter varies between -1 and +1. Also we assume the basis functions have equal widths in all dimensions, which we define as the number σ .

The basis functions must be placed near the trajectory in order to be responsive to the incoming data stream, and they should be approximately equally spaced in order to give uniform coverage. With each new data point, the Euclidean distance from the last data point is computed. We add a series of these distances to approximate the arc length along the trajectory. When the total distance exceeds the intended RBF spacing, Δ , a new basis function is centered at the current z_i and incorporated into the solution.

Given Δ , the intended RBF spacing in parameter space, we compute the width, σ , according to a heuristic: roughly four basis functions should be non-zero for each data point in order to form accurate approximations. (This heuristic results from experiments with RBF approximations to multi-dimensional polynomials.) Ensuring such a condition requires the width of the basis functions to satisfy $\sigma \geq (3/2)\Delta$ which is easily verified in the single dimensional case.

Segmenting the Data

Equation (4) shows that the new P matrix grows by one row and one column when a new basis function is added to the model. If steps are not taken to restrict this growth, P will eventually grow too large for real-time implementation of the algorithm.

Our approach is to allow the window to grow to a fixed size, *e.g.*, 10 basis functions, store the entire window, and begin a new window. This approach inherently segments the data trajectory, with each segment approximated by 10 basis functions. To evaluate this approximation at a point on a trajectory, the segment containing this point is determined and evaluated. The problem is that for a given point, it is not obvious which segment it belongs to. The solution we adopt is to find the basis function which is closest to the given point and use the segment containing that basis function. With this approach, the only time that minimum least-squares accuracy is not obtained is when the wrong segment is determined for a point, and then a close approximation is still obtained.

APPLICATION TO ONLINE LEARNING OF AIRCRAFT PARAMETERS

We now consider the problem of learning aircraft parameters online using an RBF network trained with RLS. We wish to approximate a portion of the surface of an aerodynamic derivative in the least-squares sense, by placing the centers of the basis functions along the trajectory and adjusting their heights to provide an optimum fit.

The variables of interest are aeronautical state inputs such as the angle of attack, α , the angle of sideslip, β , the speed in mach units, and the deflection angles of the stabilator and the canard, δ_S and δ_C . These inputs are used with the aircraft equations of motion in order to estimate the aerodynamic derivatives $C_{m\alpha}$ and $C_{m\delta_S}$, which are the rates of change of the coefficient of pitching moment with respect to α and δ_S ,

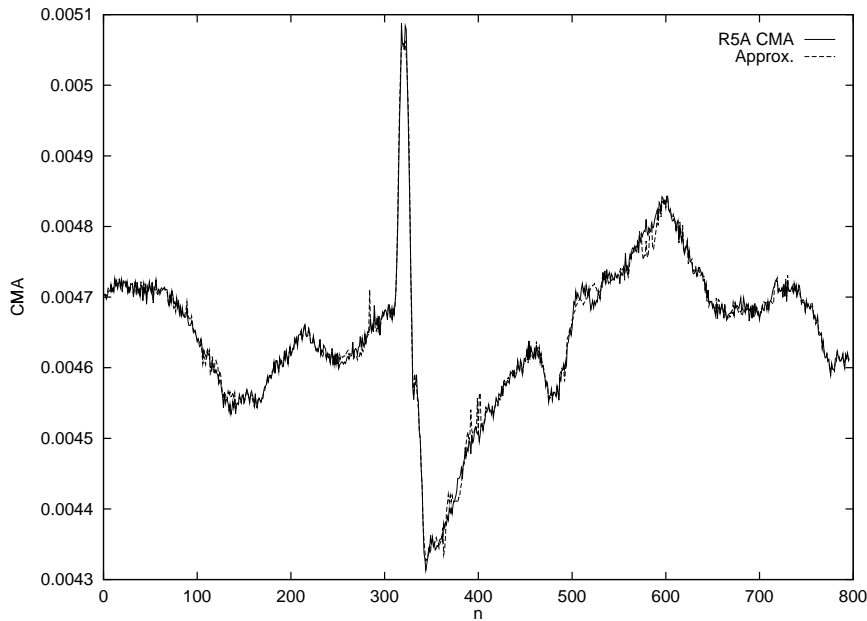


Figure 1: Segmented RLS approximation to C_{m_α} with $\Delta = 2$ and 10 RBFs per window, 42 RBFs total.

respectively. It is required to save C_{m_α} and $C_{m_{\delta_S}}$ as functions of α , β , mach, δ_S and δ_C using a neural network.

A data set from F-15 flight tests is used to demonstrate the approach. The data is sampled at 40 Hz, and the duration of our example flight is 20 seconds. Maneuvers are being performed during this flight, but the type of aircraft movement is not important to the approach. Centers are placed along the trajectory in $(\alpha, \beta, \text{mach}, \delta_S, \delta_C)$ -space using the above heuristics with $\Delta = 2$ (in normalized coordinates), and RLS is used to determine the heights so that the RBF approximates C_{m_α} and $C_{m_{\delta_S}}$ in the least-squares sense. The segmenting approach is used to limit the required computation time. The true value of C_{m_α} and its approximation in time for the flight data are shown in Figure 1. Similarly, the approximation for $C_{m_{\delta_S}}$ is shown in Figure 2.

The total range of C_{m_α} is the interval $[-0.035, 0.019]$ and the range of $C_{m_{\delta_S}}$ is $[-0.0136, 0.0014]$. Dividing the error by the size of these intervals shows that our approximations are accurate to less than 1.5% of the range of the approximated parameter. Evidence that these estimates are accurate enough for use by the controller comes from internal studies at McDonnell Douglas, which have shown that stability is maintained if all the derivatives are accurate to within 12% of range.

The compression ratio results are as follows. Our data set contains 797 data points, and 42 basis functions are used to approximate it as Figures 1 and 2 indicate. Thus, the compression ratio for this dataset is roughly 19:1. From these results, we expect compression ratios of 20:1 to be achievable and effective.

The average time to add a point using actual flight data on a SPARCstation 2 is 1.34 ms, and the average time to evaluate a point is 1.02 ms. Since the update rate of the aircraft controller is expected to be 12.5 ms, there is reason to believe that the RBF neural network can be trained and evaluated quickly enough to operate in real time.

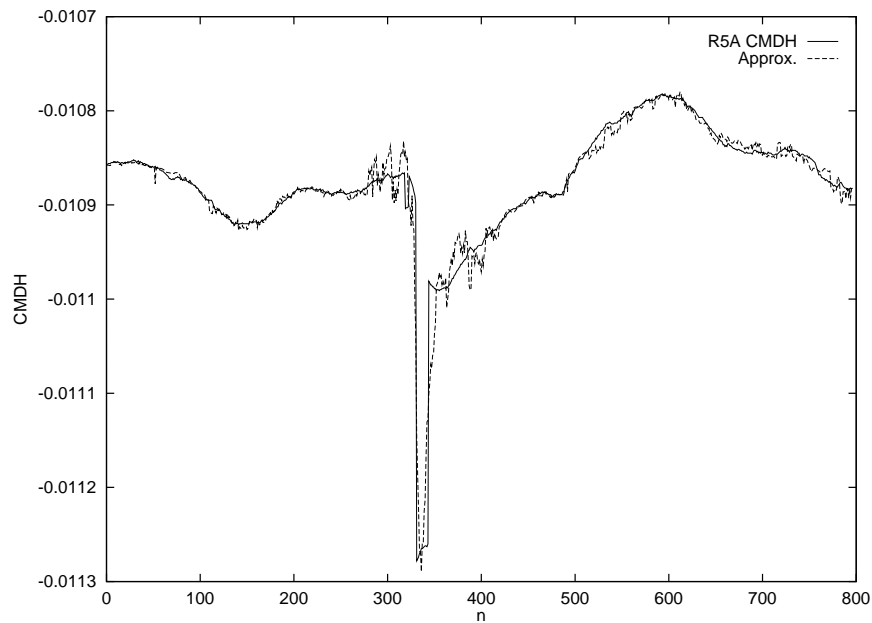


Figure 2: Segmented RLS approximation to $C_{m\delta_s}$ with $\Delta = 2$ and 10 RBFs per window, 41 RBFs total.

CONCLUSIONS

Adaptive aircraft controllers are of interest because the maintenance of excellent control is desired when the aircraft structure changes or unexpected conditions occur in flight. A quickly trained localized neural network consisting of radial basis functions can be used to provide an adaptive aircraft model for such a controller. Simulations have shown that the algorithm works in real time. Flights for the purpose of testing this concept are expected to occur in 1997.

REFERENCES

- Åström, Karl J. and Björn Wittenmark, (1989). *Adaptive Control*. Reading, MA: Addison-Wesley Publishing Company.
- Bishop, Christopher M., (1995). *Neural Networks for Pattern Recognition*, Chapter 5. Oxford, UK: Clarendon Press.
- Haykin, Simon, (1991). *Adaptive Filter Theory*, Second Edition. Englewood Cliffs, NJ: Prentice-Hall, Inc.
- Strang, Gilbert, (1980). *Linear Algebra and Its Applications*, Second Edition. New York, NY: Academic Press.