Pergamon

Engng Applic. Artif. Intell. Vol. 8, No. 5, pp. 499–514, 1995
Elsevier Science Ltd. Printed in Great Britain

0952–1976(95)00029–1

Contributed Paper

Training Fuzzy Systems to Perform Estimation and Identification

ERIC G. LAUKONEN
The Ohio State University, U.S.A.

KEVIN M. PASSINO
The Ohio State University, U.S.A.

(Received July 1994; in revised form April 1995)

A fuzzy system can be constructed to interpolate between input–output data to provide an approximation for the function that is implicitly defined by the input–output data-pair associations. This paper begins by explaining how function approximation techniques can be used to solve nonlinear estimation and system identification problems. Next, several fundamental issues are discussed, related to how to choose the input–output data pairs so that accurate function approximation can be achieved with fuzzy systems. Using this insight a technique called "uniform training" is proposed, in which input sequences are chosen to produce good training data sets ("uniform training data sets"). Also, a new technique for function approximation via fuzzy systems called "modified learning from examples" is outlined, where membership functions are specified and rules are added to try to achieve a pre-specified function approximation accuracy. Uniform training and the modified learning from examples technique are then illustrated on a simple pendulum example. In addition, the use of the modified learning from examples approach is demonstrated in constructing a fuzzy system which can identify actuator failures on an F-16 aircraft.

Keywords: Fuzzy systems, estimation, identification, failure identification.

1. INTRODUCTION

Fuzzy systems have been successfully applied in several areas within engineering including control, signal processing and pattern recognition. Some recent work has focused on the idea of constructing fuzzy systems from a finite set of input–output training data in order to perform function approximation.[1,2] This new focus is particularly important, due to the fact that many problems in estimation and identification can be formulated as function-approximation problems. For instance, in conventional system identification, input–output data is gathered from a physical system and a least-squares approach can be used to provide the best approximation for the linear function that maps the system inputs to its outputs. Similarly, in parameter estimation if one is given data that associates measurable system variables with an internal system parameter, a functional mapping may be constructed that approximates the process of estimation of the internal system parameter.

This paper:

(i) studies the problem of how to generate the input–output data so that good function approximation (and hence identification or estimation) can be achieved;
(ii) introduces a novel technique to generate good input–output data called the "uniform training algorithm";
(iii) introduces a new technique, called "modified learning from examples" (MLFE) to construct fuzzy systems to approximate the mapping present in a set of input–output data (so that the fuzzy system can be used as an estimator or identifier);
(iv) evaluates the performance of MLFE and the uniform training algorithm; and
(v) shows how it can be used to identify actuator failures on an F-16 aircraft.

The first step is to define precisely the function-approximation problem, where one seeks to synthesize

Correspondence should be sent to: Professor Kevin M. Passino, Department of Electrical Engineering, The Ohio State University, 2015 Neil Avenue, Columbus, OH 43210-1272, U.S.A.

a function to approximate another function that is inherently represented via a finite number of input–output associations (i.e. one only knows how the function maps a finite number of points in its domain to its range). Following this, the paper shows how the problem of how to construct nonlinear system identifiers and nonlinear estimators is a special case of the problem of how to perform function approximation. Next some theoretical issues are examined, associated with how to choose the input–output data so that good function approximation can be achieved. In particular, after explaining the relevance of the universal approximation property[1] it is shown that the accuracy with which approximation is achieved will depend on the structure of the input–output data in the sense that if the input–output data is "uniform" better approximation can be expected. Using this insight a method is introduced to achieve a uniform training data set called the "uniform training algorithm". Basically, this algorithm generates a sequence of inputs to the system so that the training data that is gathered uniformly covers (in a sense to be defined more carefully later) the training data space.

Next, a new technique is introduced to construct fuzzy systems from input–output training data. This is called "modified learning from examples". In this technique, ideas from the approaches in Refs 3 and 4 are used to modify the "learning from examples" (LFE) technique in Ref. 5. In particular, for MLFE a unique and novel way is utilized to position input and output membership functions using the input–output data, (i) so that what is learned about the mapping from one training data pair is not destroyed by using information from other training data pairs; and (ii) so that a smooth interpolation is achieved between the training data pairs. In addition, for MLFE rules are added to a fuzzy system to try to achieve a pre-specified function approximation accuracy. This is done in a way that guarantees that no matter what input is put into the constructed fuzzy system, there will be a well-defined output. Techniques, in addition to those in Refs 3–5, that are related to the MLFE technique include those in Refs 2 and 6–9. It is important to note, however, that while the approaches in Refs 3, 4 and 5, and the related ones in Refs 2, 6, 7, 8 and 9 have been used successfully for a variety of identification and estimation problems, they have not addressed the problems in choosing good training data sets (i.e. how to choose the training data to improve approximation accuracy) as is done here.

Following the introduction of the MLFE technique MLFE is evaluated, and it is shown how for a simple pendulum example (i) the uniform training algorithm can automatically generate a training data set that increases identification accuracy of *both* LFE and MLFE, and (ii) how MLFE can perform better than LFE while using fewer rules than LFE. To further investigate the MLFE approach, it is shown being used to construct a fuzzy system that can identify a class of actuator failures on an F-16 aircraft (this seems to be the first application of fuzzy identification to failure detection and identification). An early version of this paper appeared in Ref. 10.

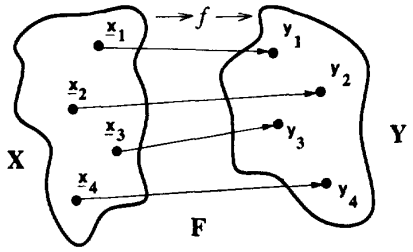

Fig. 1. Function mapping with four known input–output data pairs.

Section 2 defines the function-approximation problem, shows how identification and estimation problems are a special case of function-approximation problems, and investigates fundamental issues in the choice of the training data set. Section 3 introduces the uniform training algorithm. Section 4 introduces the MLFE approach, and Section 5 evaluates the MLFE and uniform training algorithms for a simple pendulum example. Section 6 shows how to use the MLFE for constructing a fuzzy system that can identify actuator failures on an F-16 aircraft. Section 7 contains some concluding remarks and future research directions.

2. BACKGROUND

Given some function $f: \mathscr{X} \subset \mathbb{R}^n \rightarrow \mathscr{Y} \subset \mathbb{R}$ where $\mathscr{Y}$ is a bounded set, the objective is to construct a fuzzy system $g: X \subset \mathscr{X} \rightarrow Y \subset \mathscr{Y}$, where X and Y are some domain and range of interest, by choosing a parameter vector $\boldsymbol{\theta} \in \Theta$ so that

$$f(\mathbf{x}) = g(\mathbf{x}; \boldsymbol{\theta}) + e(\mathbf{x}) \tag{1}$$

for all $\mathbf{x} \in X$ where $e(\mathbf{x})$, the error in approximation, is as small as possible. It is assumed that all that is available to choose the parameters $\boldsymbol{\theta}$ of the fuzzy system $g(\mathbf{x}; \boldsymbol{\theta})$ is some part of the function f in the form of a finite set of input–output data pairs. The ith input–output data pair for the system f is denoted by $(\mathbf{x}_i, y_i)$ where $\mathbf{x}_i \in X$, $y_i \in Y$ and $y_i = f(\mathbf{x}_i)$. The set of input–output data pairs is referred to as the *training data set*, and denoted by

$$F = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{m_F}, y_{m_F})\} \subset X \times Y \tag{2}$$

where m_F denotes the number of I/O data pairs contained in F (see Fig. 1). Therefore, the problem being considered here is how to construct a fuzzy system $g(\mathbf{x}; \boldsymbol{\theta})$ so that $f(\mathbf{x}) \approx g(\mathbf{x}; \boldsymbol{\theta})$ for all $\mathbf{x} \in X$ when only limited information is available about f in the form of the training set F.

The first step is to develop a criterion to evaluate how closely a fuzzy system $g(\mathbf{x}; \boldsymbol{\theta})$ approximates the function $f(\mathbf{x})$ for all $\mathbf{x} \in X$ for a given of $\boldsymbol{\theta}$. It is necessary to

determine a bound on the approximation error. For example,

$$\sup_{\mathbf{x}\in X}\{|f(\mathbf{x})-g(\mathbf{x};\boldsymbol{\theta})|\} \tag{3}$$

is such a bound. Such an expression requires that the function $f:\mathcal{X}\rightarrow\mathcal{Y}$ be completely known; however, as it is stated above, only a part of f, given by the finite set F is known. Therefore, it is only possible to evaluate the accuracy in approximation by evaluating the error between $f(\mathbf{x})$ and $g(\mathbf{x};\boldsymbol{\theta})$ at certain points $\mathbf{x}\in X$ given by the available I/O data. This set of I/O data is referred to as the *test set*, and denoted as Γ, where

$$\Gamma=\{(\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_{m_\Gamma},y_{m_\Gamma})\}\subset X\times Y. \tag{4}$$

Here, m_Γ denotes the number of known I/O data pairs contained within the test set. It is important to note that the I/O data pairs $(\mathbf{x}_i,y_i)$ contained in Γ may not be contained in F, or vice versa. It also might be the case that the test set is equal to the training set ($F=\Gamma$). An evaluation of the error in approximation between f and a fuzzy system $g(\mathbf{x};\boldsymbol{\theta})$ based on a test set Γ may or may not be a true measure of the error between f and g for every $\mathbf{x}\in X$, but it is the only evaluation that can be made, based on known information. Either

$$e_2=\frac{1}{2}\sum_{(\mathbf{x}_i,y_i)\in\Gamma}(f(\mathbf{x}_i)-g(\mathbf{x}_i,\boldsymbol{\theta}))^2 \tag{5}$$

or

$$e_\infty=\sup_{(\mathbf{x}_i,y_i)\in\Gamma}\{|f(\mathbf{x}_i)-g(\mathbf{x}_i,\boldsymbol{\theta})|\} \tag{6}$$

will be used to measure the approximation error. Accurate function approximation requires that some expression of this nature be small.

Many applications exist in the control and signal-processing areas which may utilize nonlinear function approximation. One such application is system identification. System identification is the process of constructing a mathematical model of a dynamic system using experimental data from that system. Let f denote the physical system to be identified. The training set F is defined by the experimental input–output data. In linear system identification an autoregressive with exogenous inputs (ARX) model is often used where

$$y(k)=\sum_{i=1}^{q}\theta_{a_i}y(k-i)+\sum_{i=0}^{p}\theta_{b_i}u(k-i) \tag{7}$$

and $u(k)$ and $y(k)$ are the system input and output at time k. In this case $g(\mathbf{x};\boldsymbol{\theta})$, which is not a fuzzy system, is defined by equation (7) where

$$\mathbf{x}=[y(k-1)\cdots y(k-q)u(k)\cdots u(k-p)]^T \tag{8}$$

$$\boldsymbol{\theta}=[\theta_{a_1}\cdots\theta_{a_q}\theta_{b_0}\cdots\theta_{b_p}]^T. \tag{9}$$

System identification amounts to adjusting $\boldsymbol{\theta}$ using information from F so that $g(\mathbf{x};\boldsymbol{\theta})\approx f(\mathbf{x})$ for all $\mathbf{x}\in X$. Clearly, restricting $g(\mathbf{x};\boldsymbol{\theta})$ to be linear may often make it difficult to achieve accurate identification (i.e. function approximation), especially if $f(\mathbf{x})$ is highly nonlinear.

This paper will investigate the possibility of constructing a fuzzy system $g(\mathbf{x};\boldsymbol{\theta})$ by choosing $\boldsymbol{\theta}$ based on available training data F so that $e(k)$ is small for all k. Similar to conventional system identification an appropriately defined "regression vector" $\mathbf{x}$, as specified in equation (8), will be utilized. Hopefully, since the fuzzy system $g(\mathbf{x};\boldsymbol{\theta})$ has more functional capabilities than the linear map defined in equation (7), it will be possible to achieve more accurate identification for highly nonlinear systems by an appropriate adjustment of its parameters $\boldsymbol{\theta}$.

A system which exhibits *universal approximation* is capable of approximating any real continuous function on a compact set to any arbitrary accuracy. Certain classes of fuzzy systems have the property of *universal approximation*. One common class of fuzzy systems considered here (for consideration of others see Ref. 2) with singleton fuzzification, product inference, Gaussian membership functions, and centroid defuzzification is governed by the parameter set

$$\boldsymbol{\theta}=[N\quad b_1\quad\ldots\quad b_N\quad c_1^1\quad\ldots\quad c_1^n\quad\ldots\quad c_N^1\quad\ldots\quad c_N^n\quad\sigma_1^1\quad\ldots\quad\sigma_1^n\quad\ldots\quad\sigma_N^1\quad\ldots\quad\sigma_N^n]^T, \tag{10}$$

and has

$$g(\mathbf{x};\boldsymbol{\theta})=\frac{\sum_{i=1}^{N}b_i\prod_{j=1}^{n}\exp\left(-\frac{1}{2}\left(\frac{x_i-c_i^j}{\sigma_i^j}\right)^2\right)}{\sum_{i=1}^{N}\prod_{j=1}^{n}\exp\left(-\frac{1}{2}\left(\frac{x_i-c_i^j}{\sigma_i^j}\right)^2\right)} \tag{11}$$

(see Ref. 2 for a more detailed explanation of fuzzy systems). The size of $\boldsymbol{\theta}$ is governed by the number of fuzzy rules N and the number of inputs to the fuzzy system n. For the fuzzy system (11) the output membership function for the ith rule is represented by the scalar point b_i (a "singleton"), the input membership function for the ith rule and jth input is a Gaussian type input membership function with a point of maximum c_i^j and a relative width term $\sigma_i^j>0$. This form for a fuzzy system has the property of universal approximation, is continuously differentiable, and has nonzero input membership values over the domain of interest X:

Theorem 1. *For any given real continuous function $f(\mathbf{x}):X\subset\mathbb{R}^n\rightarrow Y\subset\mathbb{R}$, where X and Y are compact, and an arbitrary $\varepsilon>0$, there exists a fuzzy system $g(\mathbf{x};\boldsymbol{\theta}):X\rightarrow Y$ with a fixed $\boldsymbol{\theta}$ (11) such that*

$$\sup_{\mathbf{x}\in X}|g(\mathbf{x};\boldsymbol{\theta})-f(\mathbf{x})|<\varepsilon \tag{12}$$

The proof of Theorem 1 is given in Ref. 1. Note that many other classes of fuzzy systems other than equation (11) exhibit the property of universal approximation. In order for a class of fuzzy systems to be a universal

approximator it is necessary that the class of fuzzy systems meet the conditions of the *Stone–Weierstrass Theorem*.

The property of universal approximation guarantees that there exists a way of choosing $\boldsymbol{\theta}$ for (11) so that the resulting fuzzy system can approximate any nonlinear continuous mapping to any specified accuracy. The means to choose the size of the fuzzy system, or necessary parameter set $\boldsymbol{\theta}$ are not stated, as Theorem 1 is not constructive. No knowledge of f is available for points $\mathbf{x} \in X$ such that $(\mathbf{x}, y) \notin F$. Therefore, even though a fuzzy sytem may be determined which satisfies a chosen ε for f over the training set F, this may not provide a good approximation for those points $\mathbf{x} \in X$ such that $(\mathbf{x}, y) \notin F$. Figure 2 illustrates this point for the one-dimensional case where the dotted line denotes the output of the fuzzy system $g(\mathbf{x}; \boldsymbol{\theta})$ and the solid line is $f(x)$ for all $x \in [x_1, x_2]$. Although the error in approximation at the training points x_1 and x_2 is within ε, the error in approximation within the interval is certainly not within ε. This analysis stresses the importance of obtaining an appropriate training set F, and leads to the following theorem.

Theorem 2. *Given a function* $f: X \subset \mathbb{R}^n \rightarrow Y \subset \mathbb{R}$ *and a fuzzy system* $g: X \times \rightarrow Y$ *with a fixed* $\boldsymbol{\theta}$, *both continuously differentiable, and two points* $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$ *such that* $|f(\mathbf{x}) - g(\mathbf{x}; \boldsymbol{\theta})| < \varepsilon$ *for* $\mathbf{x} = \mathbf{x}_1$ *and* $\mathbf{x} = \mathbf{x}_2$ *with* $\varepsilon > 0$ *and* $x_2 \neq x_1$. *Then for the neighborhood*

$$\beta(\mathbf{x}_1, \mathbf{x}_2) = \{\mathbf{x} \in X : \|\mathbf{x} - \mathbf{x}_1\|_2 \leq \|\mathbf{x}_2 - \mathbf{x}_1\|_2\} \tag{13}$$

where $\|\cdot\|_2$ *is the Euclidean norm, the approximation error is bounded such that*

$$\sup_{\mathbf{x} \in \beta(\mathbf{x}_1, \mathbf{x}_2)} \{|f(\mathbf{x}) - g(\mathbf{x}; \boldsymbol{\theta})| < \sup_{\mathbf{x} \in \beta(\mathbf{x}_1, \mathbf{x}_2)} \left\{ \left\| \frac{\partial f}{\partial \mathbf{x}} - \frac{\partial g}{\partial \mathbf{x}} \right\|_2 \right\} \|\mathbf{x}_2 - \mathbf{x}_1\|_2 + \varepsilon. \tag{14}$$

Proof. See Appendix.

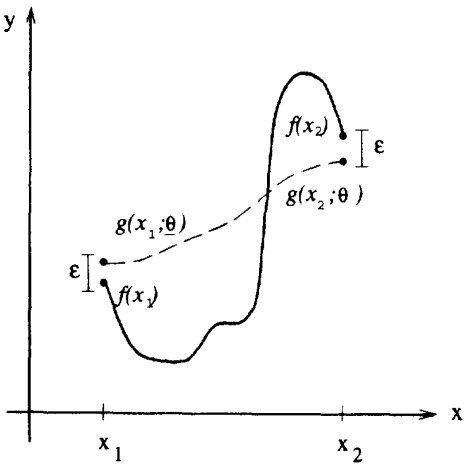

Fig. 2. Universal approximation property interpretation.

Since the case where $f(\mathbf{x})$ is not known completely is being considered, the usage of Theorem 2 is somewhat limited. Although this is the case, Theorem 2 does lead to the following observations (and it is these observations that the theorem leads to, that show the value of the theorem):

1. If the conditions of the theorem are satisfied and a fuzzy system is constructed so that good approximation is achieved at known points (ε small), then the approximation error in the neighborhood of two known points is bounded by an expression which is governed by the Euclidean distance between those two points multiplied by the magnitude of the gradient expression. Therefore, the nearness of training points contained in F may improve the approximation error within the neighborhood associated with those points (Theorem 2 quantifies the meaning of this intuitive idea).
2. This theorem helps to clarify the limitations of the universal approximation property when applied to the nonlinear function approximation problem being addressed in this paper.
3. This theorem introduces one additional constraint, namely that both $f(\mathbf{x})$ and $g(\mathbf{x}; \boldsymbol{\theta})$ be continuously differentiable, to guarantee that the error "in the neighborhood" of two points is bounded. In some cases the assumption that the functions be continuously differentiable is restrictive but the development of the ideas of the next section are not dependent on the technical conditions of the theorem.

Motivated by these observations, the ideas of *uniform training* and a uniform training algorithm are introduced.

3. UNIFORM TRAINING

The discussion of fuzzy systems and nonlinear function approximation given above illustrates a need for a *uniform* set of input–output training data F. A uniform training data set is characterized according to the following definition (note that this definition is meant to appeal to *intuition* on the type of training data set that is good for training fuzzy systems).

Uniform training data set

An input–output training data set F (2) is "uniform" if the points given by $\mathbf{x}_i \in X$, *where* $(\mathbf{x}_i, y_i) \in F$ *for* $i = 1, \ldots, m_F$:

1. *sufficiently cover the domain of interest X,*
2. *are evenly distributed over the domain of interest X, and*
3. *are sufficiently many, or* m_F *is large enough, so that for any* $\mathbf{x} \in X$ *the Euclidean distance between* $\mathbf{x}$ *and the nearest training data point is sufficiently small.*

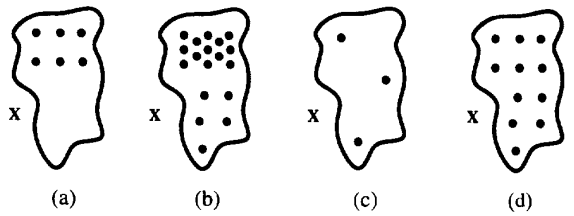

Fig. 3. Examples of not uniform (a, b, c) and uniform (d) training data sets.

The first item of the definition addresses the need to include training data which reflects the unknown functional relationship for the entire domain of interest X. The second item of the definition states that no region in the domain X contains significantly more training data points than other regions for a uniform training data set, as this might bias the construction of a fuzzy system $g(\mathbf{x};\boldsymbol{\theta})$ to approximate the unknown function $f(\mathbf{x})$ for a specific region in X. The final item in the above definition is based on Theorem 2. That is, if the conditions of Theorem 2 are satisfied, training points are sufficiently close, and good approximation is achieved at those points, then the approximation error "in the neighborhood" of those training data points may be small.

The definition of a uniform training data set can be illustrated through a graphical example for $X \subset \mathbb{R}^2$ where training points $\mathbf{x}_i$ are denoted by "•" (Fig. 3). Figure 3(a) shows a training data set which does not sufficiently cover the domain X, (b) shows a training data set which is not evenly distributed, (c) shows a training data set which may not contain a sufficient number of training data points and (d) is an example of a uniform training data set.

A uniform training data set is desirable, as it may provide for the construction of a fuzzy system $g(\mathbf{x};\boldsymbol{\theta})$ which accurately represents the functional mapping $f(\mathbf{x})$ over the domain X. If input–output training data pairs are generated experimentally by injecting input sequences $u(k)$ into an unknown system $f(\mathbf{x})$ where $u(k)$ and its delayed values are elements in the regression vector $\mathbf{x}$, then the question arises as to the proper choice of input sequences $u(k)$ which will produce a uniform training data set. Note that for practical applications the data that one would get for training a fuzzy system is not likely to be uniform in the sense described above; the focus of this section is on the development of an algorithm that will generate inputs into a system that will seek to produce this uniform set.

Most standard system-identification techniques utilize a random input sequence or some other persistently exciting signal for system excitation. Although a random input may produce sufficient information for effective identification, it may not prove to be the best excitation signal when one is concerned with the construction of fuzzy systems for nonlinear function approximation. Since fuzzy systems approximate an unknown nonlinear mapping based on known points in the space, accurate function approximation is dependent on training data which reflects the functional mapping over the domain of interest X. If an input could be constructed which systematically produced experimental training data $(\mathbf{x}_i, y_i)$ where the $\mathbf{x}_i$ sufficiently covered the domain X, then it is possible that a more accurate model may be identified when compared to a model produced with random excitation. An algorithm is proposed here for constructing input sequences which may approximate specified points for the input portion of an input–output training data for a class of unknown nonlinear systems. This is achieved by way of a "uniform training algorithm".

3.1. Uniform training algorithm

1. Class of systems

The class of systems considered for uniform training are single-input-single-output discrete-time systems described by some function $y(k)=f(\mathbf{x}(k))$ at a discrete-time k where $\mathbf{x}(k)=[y(k-1)\cdots y(k-q)\ u(k)\cdots u(k-p)]^T \in \mathbb{R}^{q+p+1}$. Moreover, the class of systems contain one and only one limit point $\mathbf{x}_e \in X$. The characteristics of the limit point $\mathbf{x}_e$ are such that when $\mathbf{x}(k_0) \in X$ and $u(k)=u_e$ for $k \geq k_e > 0$ and some known constant u_e

$$\lim_{k \geq k_e,\, k \to \infty} |\mathbf{x}(k) - \mathbf{x}_e| = 0 \tag{15}$$

where $|\mathbf{x}| = \sqrt{\mathbf{x}^T\mathbf{x}}$ (note that this condition does not imply that the systems being considered must be asymptotically stable as $\mathbf{x}$ is formed from input–output data, not necessarily the state of the system).

2. Quantized set

First, a set of regression points is chosen, based on the definition of a uniform training data set. This set is named a *quantized* set. A quantized set includes those points which are needed as the input portion of a training data set, where the "input portion" of the training data set F refers to $\mathbf{x}_i$ such that $(\mathbf{x}_i, y_i) \in F$. These quantized points are a finite subset of the domain X. Although there may be many choices for a quantized set (e.g. a uniform grid), a quantized set given by

$$\mathcal{Q} = \{\mathbf{x} \in X : \mathbf{x} = c_i(q_d \mathbf{x}_i^d) + \mathbf{x}_e \quad i = 1, \ldots, n_d,\ c_i = 1, \ldots, n_{c_i}\} \tag{16}$$

is chosen where $\mathbf{x}_i^d$ are unit direction vectors indexed by i, q_d is a scalar quantization level which represents the distance between quantized points in each direction, n_{c_i} is the number of quantized points in the ith direction, and n_d is the number of unit direction vectors. As an example, suppose a given function is only known at a finite number of points $f(\mathbf{x})$ with $X \subset \mathbb{R}^2$ and $\mathbf{x}_e = [0\ 0]^T$.

If $q_d = \frac{1}{2}$, $n_d = 8$, $n_{c_i} = 2$ for all i, and directions $\mathbf{x}_i^d$ are chosen as

$$x_1^d = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, x_2^d = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, x_3^d = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, x_4^d = \begin{bmatrix} 0 \\ -1 \end{bmatrix},$$

$$x_5^d = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \quad x_6^d = \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \quad x_7^d = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix},$$

$$x_8^d = \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$$

quantized points as shown in Fig. 4 are obtained. The information (e.g. using the relative magnitude of the gradient $\partial f/\partial \mathbf{x}$) may be known, or it may be chosen to reflect a desired number of input–output training data points.

3. Supervisory algorithm (picking desired point $\mathbf{x}_d$)

The supervisory algorithm picks a sequence of points $\mathbf{x}_d \in \mathcal{Q}$ that the trajectory control algorithm (see Step 4 below) tries to drive the system to (note that $\mathbf{x}_d$ is not a state but a regression vector). The sequence of points it picks is best explained by using the simple 2-dimensional example from above. For this example (see Fig. 4) it will first choose $\mathbf{x}_d = [0.5\ 0]^T$, then $\mathbf{x}_d = [1\ 0]^T$ (this choice is arbitrary; beginning in a different direction may be just as satisfactory). After this it will successively choose $\mathbf{x}_d$ to lie along the +45° line with increasing distance from the origin. Next, it rotates to +90° and picks points of increasing distance from the origin, and so on. The supervisory algorithm proceeds in this manner until it has selected each and every point in the quantized set $\mathcal{Q}$. In between the selection of successive $\mathbf{x}_d$ other processing occurs, as explained next.

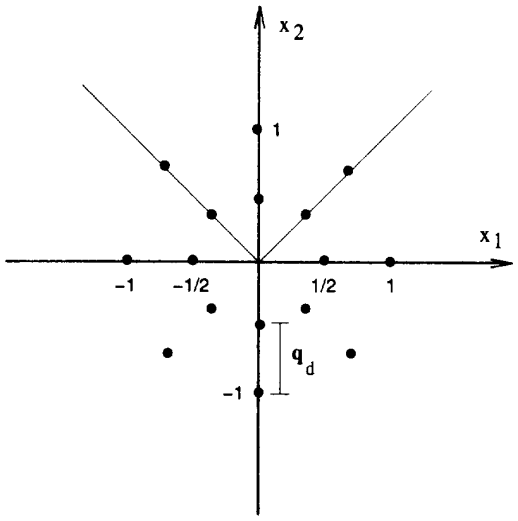

Fig. 4. Example of a quantized set.

Suppose that at some step $\mathbf{x}_d$ is chosen. The trajectory control algorithm (see Step 4 below) will generate an input sequence denoted by $u(k)$ of length T_f (where $T_a \leq T_f \leq T_a$ and T_a and T_b are design parameters) to drive the system to $\mathbf{x}_d$. $\mathbf{x}_d$ can be considered to be reached if the system can be driven to within a distance ε_d of $\mathbf{x}_d$ (i.e. within an ε_d-neighborhood of $\mathbf{x}_d$). Suppose that the system reaches $\mathbf{x}_d$. In this case the actual values of $\mathbf{x}$ and y are recorded to form a training data pair $(\mathbf{x}, y)$, which is put in F. Next, an input u_e (usually zero) is applied to the system until it reaches an ε_e-neighborhood of $\mathbf{x}_e$, and then the process is repeated by choosing the next point in the quantized space. If the point $\mathbf{x}_d$ cannot be reached by T_b seconds, then u_e is input to the system until the ε_e-neighborhood of $\mathbf{x}_e$ is reached. In either case (reaching or not reaching $\mathbf{x}_d$) as time progresses $\mathbf{x}$ is monitored. If it comes near any point in the quantized set, the corresponding values of $\mathbf{x}$ and y are used to form a training data pair, which is put in F. The algorithm allows the trajectory-control algorithm up to C tries to generate a $u(k)$ that will drive the system to $\mathbf{x}_d$. If it fails each and every time, then the supervisory control algorithm moves to the next point.

4. Trajectory control algorithm (picking inputs to drive the system to $\mathbf{x}_d$)

The trajectory control algorithm steers $\mathbf{x}(k)$ to the $\mathbf{x}_d$ chosen by the supervisory algorithm by applying an input sequence $u(k)$. The class of input sequences chosen to construct follows the form of a finite Fourier series given by

$$u(k) = \phi_0 + \sum_{i=1}^{Q} \{\phi_{2i-1} \sin(k(i\gamma)) + \phi_{2i} \cos(k(i\gamma))\} \quad (17)$$

where Q is a measure of the number of terms in the Fourier series, γ is a parameter denoting the fundamental frequency, and $\phi = [\phi_0 \cdots \phi_{2Q}]^T$ is a vector of free parameters which are the coefficients of the Fourier series. The finite Fourier series input allows a wide range of possible inputs within an expression which is easy to manipulate. Note also, that as long as $\|\phi\|_2$ is bounded then the input $u(k)$ is bounded. Therefore, to ensure a bounded input the constraint

$$\|\phi\|_2 \leq M \quad (18)$$

is imposed for some $M > 0$. The input sequence (17) is fixed, based on the values for ϕ and is applied to the system over some discrete-time window W

$$W = \{0, 1, \ldots, T_f\} \quad (19)$$

where T_f is a parameter denoting the input window duration. The training algorithm treats T_f as a free parameter which is bounded according to fixed values T_a and T_b so that $T_f \in \{T_a, T_a+1, \ldots, T_b\}$. These bounds on the free parameter T_f ensure a more tractable algorithm by allowing the input window duration to be in some specified range given by T_a and T_b. The update of T_f and $\boldsymbol{\phi}$ is explained next.

The supervisory-level algorithm injects the calculated input sequences $u(k)$ over the discrete-time window W. Parameter updates for $\boldsymbol{\phi}$ are calculated using a gradient-descent technique based on $\mathbf{x}(k)$ generated by the previous input sequence. Let

$$z(k) = \tfrac{1}{2}(\mathbf{x}(k)-\mathbf{x}_d)^T(\mathbf{x}(k)-\mathbf{x}_d) = \tfrac{1}{2}[(x_1(k)-x_{d_1})^2+\cdots+(x_n(k)-x_{d_n})^2]. \quad (20)$$

The optimization technique we utilized is based on a performance measure given by

$$J = \inf_{k\in W}\{z(k)\} = z(k_f). \quad (21)$$

To define the performance measure the ordered set of discrete time points at which the inf is achieved is denoted as $\mathcal{K}$, so that

$$\mathcal{K} = \arg\inf_{k\in W}\{z(k)\}. \quad (22)$$

A single element $k_f \in \mathcal{K} \subset W$ which is the minimum in $\mathcal{K}$ is chosen, or

$$k_f = \inf \mathcal{K}. \quad (23)$$

In other words, the minimum $z(k)$ is found over the window given by W. If more than one minimum exists then the first occurrence of the minimum and $J = z(k_f)$ is picked as the performance.

It is desired to minimize (21) based on a gradient descent method. Specifically, the idea is to compute the gradient of the cost function with respect to the free parameter set, $\boldsymbol{\phi}$. The gradient is used to iteratively construct an input sequence so that $z(k_f)$ becomes small after successive tries with the constraint that $\|\boldsymbol{\phi}\|_2 \le M$. Specifically, an input $u(k)$ is applied for all $k = 1, \ldots, T_f$ (17) ($k=1$ means the current point in time) based on the parameters $\boldsymbol{\phi}$. $\boldsymbol{\phi}$ is then updated, based on the gradient of the cost function (21), and once the system is settled near $\mathbf{x}_e$ the new input sequence based on the updated parameter set is applied.

The parameter update utilizing the gradient of J is computed with respect to the parameter set $\boldsymbol{\phi}$. The new coefficients for the finite Fourier series become $\boldsymbol{\phi} := \boldsymbol{\phi} + \Delta\boldsymbol{\phi}$ where

$$\Delta\boldsymbol{\phi} = -\eta\frac{\partial J}{\partial \boldsymbol{\phi}} = -\eta\frac{\partial z(k_f)}{\partial \boldsymbol{\phi}} \quad (24)$$

where η is the step size chosen by the designer. Using the chain rule,

$$\frac{\partial z(k_f)}{\partial \phi_i} = (x_1(k_f)-x_{d_1})\frac{\partial x_1(k_f)}{\partial u(k_f)}\frac{\partial u(k_f)}{\partial \phi_i} + \cdots + (x_n(k_f)-x_{d_n})\frac{\partial x_n(k_f)}{\partial u(k_f)}\frac{\partial u(k_f)}{\partial \phi_i} \quad i = 1, \ldots, 2Q \quad (25)$$

and

$$\frac{\partial z(k_f)}{\partial \boldsymbol{\phi}} = (\mathbf{x}(k_f)-\mathbf{x}_d)^T\frac{\partial \mathbf{x}(k_f)}{\partial u(k_f)}\frac{\partial u(k_f)}{\partial \boldsymbol{\phi}}. \quad (26)$$

The term $\mathbf{x}$ is measured and the term $\mathbf{x}_d$ is given. The term $\partial u(k_f)/\partial \phi_i$ is easily computed by finding the gradient of the finite Fourier series,

$$\frac{\partial u(k_f)}{\partial \phi_i} = \begin{cases} 1 & i=0 \\ \sin(k_f i\gamma) & i=1,3,\ldots,2Q-1 \\ \cos(k_f i\gamma) & i=2,4,\ldots,2Q. \end{cases} \quad (27)$$

The remaining term $\partial x_i(k_f)/\partial u(k_f)$ represents the change in the output vector with respect to the change in input between iterations. The expression is approximated by computing the finite difference between successive iterations in the windowing sequence.

$$\frac{\partial \mathbf{x}(k_f)}{\partial u(k_f)} = \frac{\mathbf{x}(k_f)-\mathbf{x}((k-1)_f)}{u(k_f)-u((k-1)_f)} \quad (28)$$

where $(k-1)_f$ denotes k_f from the previous input sequence. These expressions are combined together with the constraint that $\|\boldsymbol{\phi}\|_2 \le M$ to form an overall expression representing the update algorithm for the parameter set $\boldsymbol{\phi}$:

$$\Delta\boldsymbol{\phi} = \begin{cases} -\eta\dfrac{\partial z(k_f)}{\partial \boldsymbol{\phi}} & \|\boldsymbol{\phi}+\Delta\boldsymbol{\phi}\|_1 \le M \\ -M\dfrac{\dfrac{\partial z(k_f)}{\partial \boldsymbol{\phi}}}{\left\|\dfrac{\partial z(k_f)}{\partial \boldsymbol{\phi}}\right\|_2} - \boldsymbol{\phi} & \|\boldsymbol{\phi}+\Delta\boldsymbol{\phi}\|_1 > M \end{cases} \quad (29)$$

T_f, is updated based on k_f, and the bounds T_a and T_b. Since J may be minimized for k near k_f an input window duration is chosen which places k_f in the center of the discrete-time window provided certain conditions are met with k_f with regard to T_a and T_b. Specifically, the final time for the next applied input is chosen according to the equation

$$T_f = \begin{cases} T_a & 2k_f < T_a \\ 2k_f & T_a \le 2k_f \le T_b \\ T_b & 2k_f > T_b \end{cases}. \quad (30)$$

Such a choice is *ad hoc*, but it allows flexibility in the input duration which may improve efficiency of the algorithm.

In order to implement the uniform training algor-

ithm, a set of design parameters must be chosen. Although these parameters may be chosen in an *ad hoc* fashion, in many cases choices for parameters may be based on implementation concerns. Some parameter values may be chosen on the basis of knowledge about the system and the actuator dynamics. For instance, based on the actuator dynamics $Q\gamma$ is a term associated with the bandwidth of the finite Fourier series. If the bandwidth of the actuator used for excitation is known, then the quantity $Q\gamma$ can be chosen to match the actuator bandwidth. Additionally, the actuator will generally contain hard limits in signal amplitude. If these limits are known, then M can be chosen properly. Additionally, if the relative magnitude of the gradient in the system is known, then q_d can be chosen small enough and the directions $\mathbf{x}_i^d$ can be chosen to specify an appropriate quantized set. Also, if the domain of interest X is known then n_{c_i} may be chosen to specify an appropriate quantized set. The values of ε_e and ε_d can be chosen on the basis of the spacing in the quantized set. The remaining parameters, C, and η can be chosen via trial and error to achieve a uniform training data set.

4. MODIFIED LEARNING FROM EXAMPLES

This section utilizes some of the insight from the analysis performed in the previous section to propose a simple method for constructing a fuzzy system for nonlinear function approximation based on a training data set F defined by (2). The method proposed is an extension of other techniques proposed for nonlinear function approximation via fuzzy systems. Namely, a technique given in Ref. 5, entitled "learning from examples", is utilized, with some ideas from Refs 3 and 4, to develop a new method to construct a fuzzy system for nonlinear function approximation. First, the methodology is developed, and then the function-approximation algorithm is presented.

Given the parameterized fuzzy system (11) the parameters in $\boldsymbol{\theta}$ and the number of rules N are chosen to approximate a nonlinear function $f: X \subset \mathbb{R}^n \rightarrow Y \subset \mathbb{R}$ based on input–output data pairs generated via experiment or simulation.

The first question be addressed in the construction of a fuzzy system is the choice of the number of fuzzy rules N. More fuzzy rules means increased computational complexity in implementation. Therefore specific applications may limit the number of rules which may be utilized in implementation. On the other hand, the use of many fuzzy rules may produce a fuzzy system with more functional capability available for approximation. Therefore, a tradeoff exists between computational complexity and functional capability based on the number of fuzzy rules N.

First the quantity ε_f, which characterizes the desired accuracy with which the fuzzy system performs function approximation is defined. Specifically, the output of the current fuzzy system is compared to the output portion of the input–output training data point y_i. If this comparison is greater then ε_f then the current fuzzy system is augmented with an additional rule so that:

$$|g(\mathbf{x}_i, \boldsymbol{\theta}) - y_i| > \varepsilon_f \Rightarrow \text{modify}$$
$$|g(\mathbf{x}_i; \boldsymbol{\theta}) - y_i| \leq \varepsilon_f \Rightarrow \text{do not modify.} \quad (31)$$

The choice for ε_f determines the number of fuzzy rules, where a smaller value for ε_f generally means more fuzzy rules for a given training data set, and vice versa. Employment of ε_f in the function approximation algorithm is outlined below.

4.1. MLFE Algorithm to construct fuzzy system

1. Construct an initial fuzzy system

Assume some input–output training data pairs given by $(\mathbf{x}_i, y_i) \in F$, where $i = 1, \ldots, m_F$. An initial value is also chosen for the input membership associated with the first fuzzy rule given by $\sigma_1^j = \sigma_0$ for all $j = 1, \ldots, n$. The initial choice σ_0 has little effect on the resulting fuzzy system as training progesses, but it is necessary to initialize the fuzzy system. The first step in the algorithm is to form a fuzzy rule according to the first input–output training data pair $(\mathbf{x}_1, y_1)$ and the initial fuzzy system by

$$g(\mathbf{x}; \boldsymbol{\theta}) = \frac{b_1 \prod_{j=1}^{n} \exp\left[-\left(\frac{x_j - c_1^j}{\sigma_1^j}\right)^2\right]}{\prod_{j=1}^{n} \exp\left[-\left(\frac{x_j - c_1^j}{\sigma_1^j}\right)^2\right]}, \quad (32)$$

where $c_1^j = x_1^j$, $b_1 = y_1$, and $\sigma_1^j = \sigma_0$ for $j = 1, \ldots, n$. Once the initial fuzzy system is constructed, the function-approximation algorithm forms additional fuzzy rules and chooses parameters θ for the fuzzy system based on the remaining input–output training data pairs and the desired approximation accuracy ε_f in Steps 2 and 3.

2. Evaluate new training data point

For each additional training data point $(\mathbf{x}_i, y_i) \in F$, for $i = 2, \ldots, m_F$, evaluate (31) with the current fuzzy system. If an additional rule is necessary then go to Step 3, otherwise return to the start of Step 2 to evaluate the next training data point according to (31).

3. Augment current fuzzy system

Modify some of the parameters of the original fuzzy system, namely σ_i^j, to account for the new information contained in $(\mathbf{x}_i, y_i)$ that passed the test in (31). A new rule is added to the fuzzy system so that the new fuzzy parameter set $\boldsymbol{\theta}$ is given by

$$N := N + 1 \quad (33)$$
$$b_N := y_i \quad (34)$$
$$c_N^j := x_i^j, j = 1, \ldots, n \quad (35)$$

Moreover, modify σ_i^j for $j = 1, \ldots, n$ to adjust the

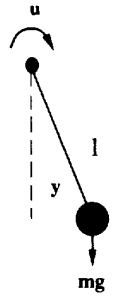

Fig. 5. Ideal pendulum.

spacing between the membership functions for the rules in the augmented fuzzy system so that: (i) the added rule does not distort what has already been learned; and (ii) a smooth interpolation betwen training points is achieved. Modification of σ_i^j is carried out by determining for each rule a nearest neighbor. Modify σ_i^j for the ith rule according to the computed nearest neighbor, denoted by the index i^*

$$i^* = \arg\min\{\|\mathbf{c}_{i'} - \mathbf{c}_i\|_2 : i' = 1, \ldots, N, i' \neq i\} \quad (36)$$

where $\mathbf{c}_i = [c_i^1, \ldots, c_i^n]^T$. Update for $j = 1, \ldots, n$:

$$\sigma_i^j = \frac{1}{W}|\mathbf{c}_i^j - \mathbf{c}_{i^*}^j| \quad (37)$$

where W is a weighting term which governs the input membership overlap between nearest neighbor rules. A larger W means less overlap of membership functions, and vice versa. Once Step 3 is complete go to Step 2 for the next training data point, until the number of training data points are exhausted.

5. EXAMPLE: SIMPLE PENDULUM

To illustrate the MLFE algorithm with and without the uniform training algorithm, system identification is performed for a simple pendulum system to estimate the pendulum angular position $y(k)$, based on previous torque inputs and angular position outputs. The dynamics of an ideal pendulum (Fig. 5) can be modeled by a differential equation written as

$$ml\ddot{y} = -mg \sin y - bl\dot{y} + \frac{1}{l}u \quad (38)$$

where y is the angle subtended from the centerline, m is the mass of the bob, l is the length of the massless connection between the pivot point and the bob, g is the acceleration due to gravity, b represents a damping coefficient, and u is an input torque at the pivot point. Additionally, the subtended angle y is constrained between $\pm\pi/2$ and the input is constrained between ± 10.

Using a backward-looking difference approximation for the dervative the continuous-time system is approximated by a discrete-time system

$$\frac{ml(y(k) - 2y(k-1) + y(k-2))}{\Delta t^2} = -mg \sin y(k-2) - \frac{bl(y(k-1) - y(k-2))}{\Delta t} + \frac{1}{l}u(k-2). \quad (39)$$

This discrete-time approximation produces an expression for $y(k)$ where

$$y(k) = f(\mathbf{x}(k)) \quad (40)$$

$$\mathbf{x}(k) = [y(k-1), y(k-2), u(k-2)]^T. \quad (41)$$

Such a system satisfies the assumptions necessary to implement the uniform training algorithm. Namely, the system contains one and only one limit point $\mathbf{x}_e = [0\ 0\ 0]^T \in X$ where the domain of interest is

$$\mathbf{x}(k) \in X = \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \times \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \times [-10, 10]. \quad (42)$$

For simulation purposes, $m = 0.50$, $g = 9.81$, $l = 1.00$ and $b = 0.6$ are chosen; also, a sampling time $\Delta t = 0.10$ s.

First, different training scenarios are tried out, to give an insight into the effects of the choice of design parameters on training. For comparison purposes, the LFE algorithm is implemented, and the approximation accuracy is compared for the cases trained with band-limited white-noise input and with training data produced by the uniform training algorithm.

5.1. MLFE algorithm with different choices in design parameters

In this section the MLFE algorithm is used to construct a fuzzy system for various training scenarios. The training scenarios investigated include:

- Training for different numbers of training data points m_F;
- Training for different input membership overlap (W); and
- Training with different specified accuracies (ε_f).

For these tests the chosen input for training is band-limited white noise, generated by injecting a Gaussian white-noise signal through a third-order Butterworth filter with a cutoff frequency of $7\pi/10$. This signal is utilized to generate a training data set F which consists of the training data pairs $(\mathbf{x}_i, y_i)$ for $i = 1, \ldots, m_F$. System identification accuracy is tested by comparing the pendulum angular position output with the output of the fuzzy system for a step input. The duration of the test chosen is 5 s. It is important to note that the step input is different from the input used in training and comparisons are done by evaluating e_2 in equation (5) and e_∞ in equation (6). The size of the resulting fuzzy systems is also evaluated by the number of rules N.

Table 1. MLFE training for the simple pendulum

Scenario	m_F	W	ε_f	N	e_2	e_∞
Varying m_F	50	2	0.20	27	0.2945	0.1583
	100	2	0.20	43	0.2725	0.1554
	500	2	0.20	147	0.0915	0.0931
Varying W	200	1	0.20	45	0.2230	0.1299
	200	3	0.20	57	0.3528	0.1686
	200	5	0.20	63	0.6804	0.1486
Varying ε_f	200	2	0.30	30	0.6736	0.2429
	200	2	0.10	85	0.3077	0.2077
	200	2	0.05	143	0.4946	0.1834

The training scenarios given in Table 1 illustrate some effects of parameter choice on the resulting fuzzy system. By increasing the number of training data points m_F the error in approximation, in this case, decreases as more information is included in training, while the number of rules N increases. When W is increased the relative amount of overlap between fuzzy sets is decreased and the accuracy, in this case, is degraded. With increased W the resulting fuzzy system contains narrower input membership functions, and accuracy with respect to the training data set is improved, but the ability of the fuzzy system to generalize for other inputs may be degraded as is illustrated in the results. The relative number of rules remains about the same for the case where W is varied. In a similar way, decreasing the desired error ε_f increases the accuracy with respect to the training data set, but accuracy in the presence of other inputs different than the training data set is degraded, as shown in Table 1. These tests also demonstrate a need to obtain a uniform training data set which may enable the MLFE algorithm to train a fuzzy system to perform accurate function approximation for a wide class of inputs.

5.2. Comparison between LFE and MLFE

For the uniform training algorithm, $M=10$, $Q=40$, $\gamma=\frac{1}{30}$, $T_a=4$, $T_b=400$, $C=10$, $q_d=0.1$, $\varepsilon=0.025$, $\varepsilon_d=0.025$ and $\eta=0.20$ were chosen. 26 evenly spaced direction vectors were chosen to represent the quantized set. A bandlimited white-noise signal is injected for 500 samples, and then supervised training is implemented. Once supervised training with noise is complete, the trajectory control algorithm is implemented to actively induce training in regions of the space that have not been trained.

For implementation of the LFE algorithm three input universes of discourse were chosen, one for each input to the fuzzy system, with 31 input fuzzy sets with evenly spaced triangular membership functions. The universes of discourse for x_1 and x_2 are defined over the interval $[-\pi/2, \pi/2]$ and the universe of discourse for x_3 is defined over the interval $[-10, 10]$. The output universe of discourse contains 31 output fuzzy sets with triangular membership functions, with a universe of discourse defined over the interval $[-\pi/2, \pi/2]$. This configuration produced good function approximation for comparison. Moreover, the design parameters chosen for the MLFE algorithm were chosen to provide good function approximation via experiment.

Table 2. Training for the simple pendulum with uniform training algorithm

Technique	Training	Figure	m_F	W	ε_f	N	e_2	e_∞
LFE	noise	6	249	—	—	208	0.1667	0.1333
MLFE	noise	7	249	2	0.10	202	0.1101	0.1009
LFE	uniform	8	249	—	—	202	0.1369	0.1169
MLFE	uniform	9	249	2	0.10	97	0.0498	0.0585

Although the LFE and the MLFE algorithms produce functionally different fuzzy systems, to make the comparisons fair an attempt was made to provide equivalent levels of training. Specifically, since the uniform training algorithm produces 249 training data points, the bandlimited white-noise cases were also trained with 249 training data points.

The accuracy in system identification was evaluated by comparing the angular position output of the pendulum to the output of the fuzzy system for a step input 5 s in duration. It is important to note that the step input is different than the input used in training, and comparisons were made by evaluating e_2 in equation (5) and e_∞ in equation (6). The size of the resulting fuzzy systems was also evaluated by the number of rules N. The results are summarized in Table 2. The results are shown graphically in Figs 6–9.

As the results illustrate, the MLFE algorithm performs better than the LFE training algorithm and MLFE uses fewer rules*. More importantly, utilizing the uniform training algorithm to produce the training data set for the MLFE algorithm increases system-identification accuracy, while at the same time the number of fuzzy rules for the fuzzy estimator is reduced

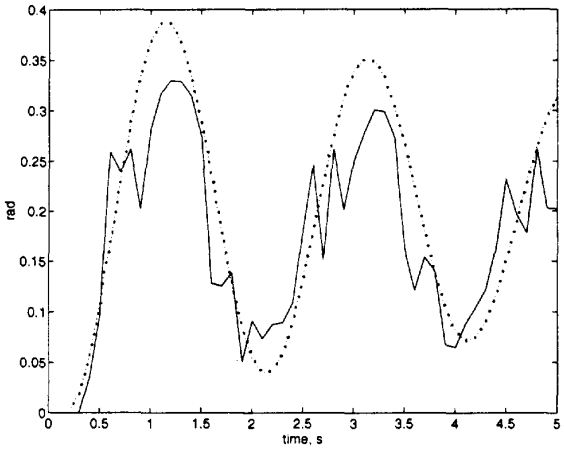

Fig. 6. LFE (dotted = real) (solid = estimate) for noise.

* This is not, however, considered to be conclusive evidence of the superiority of MLFE over LFE. It is clear that a significant amount of future work (beyond the scope of this study) on theoretical and experimental analysis would have to be performed to make such a conclusion. See the comments in the concluding remarks of this paper.

when compared to fuzzy estimators with similar accuracy trained with strictly bandlimited white noise. In a similar way, accuracy is increased for the LFE algorithm when the uniform training algorithm is utilized instead of training with bandlimited white noise. By utilizing the uniform training algorithm the fuzzy estimator has been enabled to perform accurately for some inputs in which the system has not been trained, like the step input. Notice also that the uniform training algorithm in this case improves system identification accuracy for both training methods, meaning that the uniform training algorithm may be useful for any algorithm that is used to construct the fuzzy system.

6. FUNCTION APPROXIMATION EXAMPLE: F-16 AIRCRAFT

In this example, the function approximation algorithm is used to construct an estimator that can detect and identify actuator failures by using measurable F-16 aircraft data. The presence of an actuator failure is detected by comparing the actuator's commanded position and the estimated position. A failure is detected if the commanded position and the estimated position differ by more than some threshold for some window of time.

6.1. F-16 aircraft model and failure modes

The F-16 aircraft model used in this example is based on a set of five linear perturbation models (that are extracted from a non-linear model* at the five operating conditions); (A_i, B_i, C_i, D_i), $i \in \{1, 2, 3, 4, 5\}$:

$$\dot{x}_{f16} = A_i x_{f16} + B_i u$$
$$y = C_i x_{f16} + D_i u \tag{43}$$

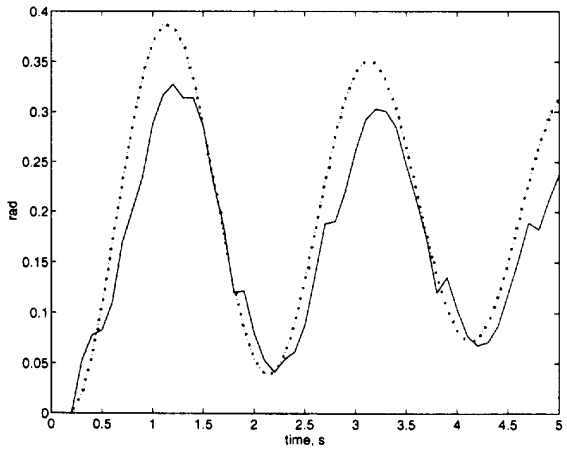

Fig. 7. MLFE (dotted = real) (solid = estimate) for noise.

* All information about the F-16 aircraft models was provided by Wright Laboratories.

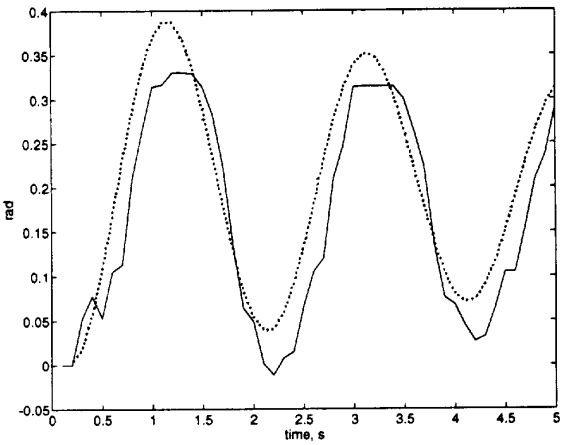

Fig. 8. LFE (dotted = real) (solid = estimate) with uniform training.

where the variables are defined as follows:

- Inputs $u = [\delta_e \ \delta_{de} \ \delta_a \ \delta_r]^T$:
  - δ_e = elevator deflection (degrees)
  - δ_{de} = differential elevator deflection (degrees)
  - δ_a = aileron deflection (degrees)
  - δ_r = rudder deflection (degrees)
- System State $x_{f16} = [\alpha \ q \ \phi \ \beta \ p \ r]^T$:
  - α = angle of attack (degrees)
  - q = body axis pitch rate (degrees/s)
  - ϕ = Euler roll angle (degrees)
  - β = sideslip angle (degrees)
  - p = body axis roll rate (degrees/s)
  - r = body axis yaw rate (degrees/s)
- Outputs $y = [x_{f16}^T A_z]^T$:
  - A_z = normal acceleration (g)
- System matrices (A_i, B_i, C_i, D_i): Provided by Wright Laboratories.

The nominal control laws for the F-16 aircraft that were provided by the Wright Laboratories consist of

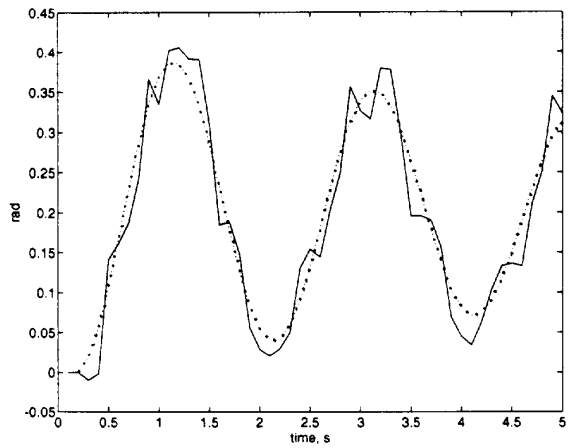

Fig. 9. MLFE (dotted = real) (solid = estimate) with uniform training.

two parts, one for the longitudinal channel and the other for the lateral channel. The inputs to the controller are the pilot commands and the F-16 system feedback signals. For the longitudinal channel, the pilot command is the desired pitch A_{zd}, and the system feedback signals are normal acceleration A_z, angle of attack α, and pitch rate q. Likewise, for the lateral channel, the pilot commands are the desired roll rate p_d as well as the desired yaw β_d, and the system feedback signals are the roll rate p, yaw angle r, and sideslip β. The controller gains for the longitudinal and for the lateral channels are scheduled as a function of different dynamic pressures. The dynamic pressure at all five perturbation models is fixed at 499.24 psf, which is based on an assumption that the F-16 aircraft will operate with constant speed and altitude. Hence, a gain schedule table, which is provided by Wright Laboratories, is used to determine the controller gains.

The transfer function $20/s+20$ is used to represent the actuator dynamics for each of the aircaft control surfaces, and the actuators have physical saturation limits so that: $-21° \leq \delta_e \leq 21°$, $-21° \leq \delta_{de} \leq 21°$, $-23° \leq \delta_a \leq 20°$, and $-30° \leq \delta_r \leq 30°$. The actuator rate saturation is $\pm 60°/s$ for all the actuators. The closed-loop system is simulated by interpolating between the five perturbation models based on the value of α. For all the simulations, a special "loaded roll command sequence", provided by Wright Laboratories, is used. For this command sequence: at time $t=0.0$, a 60°/s roll rate command (p_d) is held for 1 s; at time $t=1.0$, a 3g pitch command (A_{zd}) is held for 9 s; at time $t=4.5$, a $-60°/s$ roll rate command (p_d) is held for 1.8 s; and at time $t=11.5$, a 60°/s roll rate command (p_d) command is held for 1 s. The sideslip command β_d is held at zero throughout the sequence.

While many different failures can occur on a high-performance aircraft such as the F-16 (e.g. performance degradation or structural damage), this study will focus on FDI for aileron and differential elevator stuck failures.

6.2. Failure models and training

In order to implement a fuzzy estimator for FDI on the F-16 aircraft it is necessary to determine those signals from the aircraft which enable one to deduce the position of the aileron and differential elevator. Specifically, it is first necessary to determine the inputs $\mathbf{x}_i$ $(i=1, \ldots, n)$ so that a fuzzy system (11) may produce accurate estimates ($\hat{\delta}_a$ and $\hat{\delta}_{de}$) for the aileron and differential elevator positions (δ_a and δ_{de}). This is done by examining the structure of the F-16 aircraft model and understanding the effect of the controller on the actuator positions. Specifically, if a failure in the aileron actuator occurs, the controller compensates via the differential elevator and vice versa. Also, the roll command input p_d affects both the aileron and differential elevator positions. Therefore these signals are utilized as inputs to the fuzzy estimator to deduce the position of the aileron and differential elevator. In particular, the aircraft responses are sampled, and

$$\mathbf{x} = [\delta_a^c, \delta_{de}^c, p_d]^T \tag{44}$$

is chosen, where δ_a^c and δ_{de}^c are the signals commanded by the controller. The vector $\mathbf{x}$ $(n=3)$ is used in the estimation of both the aileron position and the position of the differential elevator. Next, f_a and f_{de} are used to denote the unknown functional mapping between the aircraft variables and the estimates of the actuator positions, so that

$$\hat{\delta}_a(k) = f_a(\mathbf{x}) \tag{45}$$

and

$$\hat{\delta}_{de}(k) = f_{de}(\mathbf{x}). \tag{46}$$

Note that the choice of the inputs in (44) intuitively represents signals which may contribute to the functional mapping f_a or f_{de} in order to construct the fuzzy estimator. For instance, during no-fault operation the aileron and differential elevator positions are related linearly to the roll command p_d. Basically, the estimator construction problem involves training the fuzzy system to approximate f_a and another to approximate f_{de}. With this, one fuzzy system will estimate the position of the aileron, and one the position of the differential elevator.

The fuzzy systems are trained for estimation by obtaining input–output training data developed through simulation with a sampling time of $T=0.02$ s. The aileron and differential elevator actuators are alternately failed at various positions (no failure, 1 s, 3 s, 5 s, 7 s) and sampled data is collected for $\mathbf{x}$. The fuzzy systems are then constructed for estimation by means of the training algorithm which chooses values for N, b_i, c_i^j and σ_i^j to achieve an accuracy of $\pm 1°$ on both the aileron and the differential elevator estimators during training (i.e. $\varepsilon_f = 1$). The width scaling term W is 3. The resulting fuzzy system for estimation of aileron position contains 108 rules, and the resulting fuzzy system for estimation of differential elevator position contains 224 rules.

6.3. Results

The fuzzy estimator provides a simple FDI system to detect and isolate failures for the aileron and differential elevator actuators. Form the absolute value of the difference between the estimated value and the value commanded for the actuator. If one differs above some threshold for some period of time, this implies that a failure has occurred, and in this situation one may want to reconfigure the control laws (see Ref. 11). In this case the median of the residual is evaluated over 21 samples. If the resulting median exceeds a threshold of 3° for the aileron and 1° for the differential elevator, then the system is flagged that a specific actuator failure has occurred (these thresholds were chosen via a simulation-based analysis of the no-fault residuals).

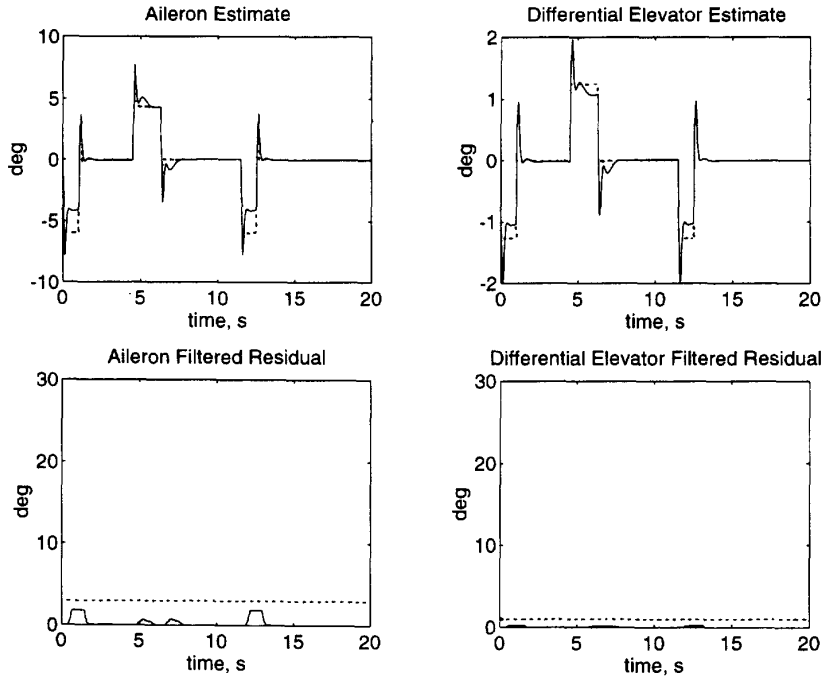

Fig. 10. No failure case (——, estimate and residual; . . . . ., true value and threshold).

Once a residual exceeds a threshold the failure flag is set for the entire test. The results are given in Figs 10–12. Notice that in Fig. 10, if there is no actuator failure then the failure estimators for both the aileron and differential elevator provide an estimate of the actuator positions that is reasonably close to the actual position. Clearly if a detection threshold of 3° had been chosen for the aileron residual and a detection threshold of 1° for the differential elevator, no failure would be indicated. Using the medial filter described above, failures

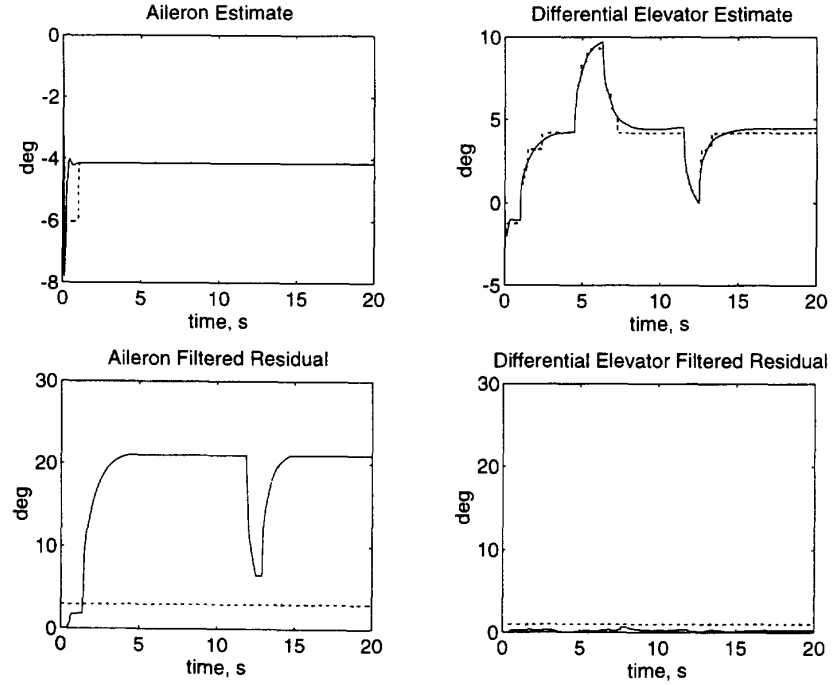

Fig. 11. Aileron failure at $t = 1$ s (——, estimate and residual; . . . . . , true value and threshold).

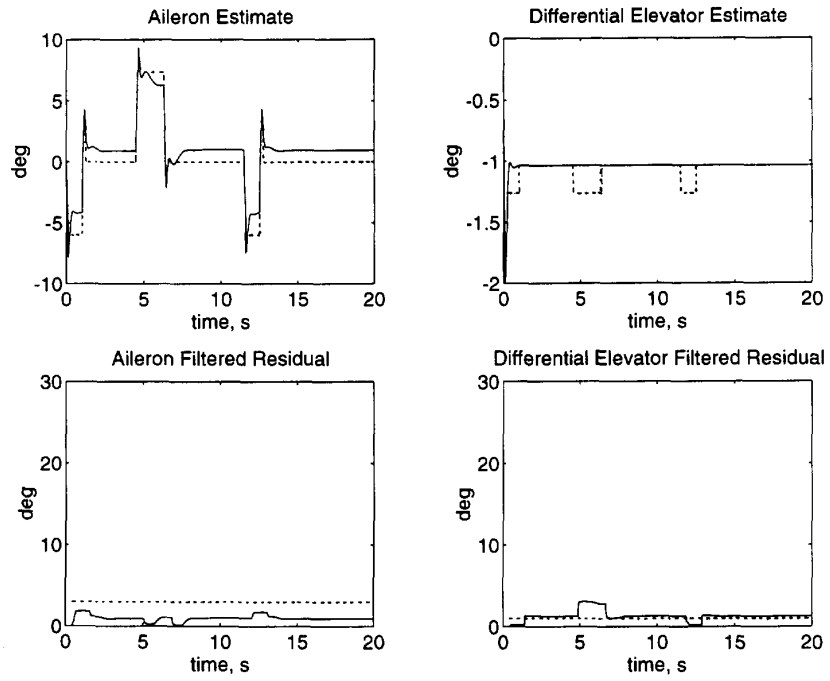

Fig. 12. Differential elevator failure at $t = 1$ s (——, estimate and residual; . . . . . , true value and threshold).

are correctly detected and determined for an aileron failure at 1 s (Fig. 11) into the loaded roll command sequence with the filtered aileron residuals exceeding the threshold and the filtered differential elevator residual remaining below the threshold. Moreover, failures are correctly detected and determined for a differential elevator failure at 1 s into the loaded roll command sequence with the filtered differential elevator residuals exceeding the 1° threshold and the filtered aileron residuals remaining below the threshold (see Fig. 12). It can be seen that the estimators provide an adequate estimate of the position at which the aileron and differential elevator are stuck, and also properly indicate which actuator has not failed. The simulation results actually show that the fuzzy estimator can accurately discriminate between an aileron and differential elevator failures for the training data set utilized. If failures occur other than the ones that have been trained for, or if a different aircraft maneuver is initiated, accurate estimation may not be achieved.

7. CONCLUDING REMARKS

This paper has concisely stated the function-approximation problem in the context of fuzzy systems, illustrated possible applications for nonlinear function approximation in the system identification and signal processing fields, discussed the functional capability of fuzzy systems in terms of the universal approximation property and discussed constraints associated with the input–output training data set. In addition, an algorithm was introduced to obtain a uniform training data set using a uniform training algorithm. A method was also outlined to construct a fuzzy system for function approximation using a method of modified learning from examples. The uniform training algorithm and the modified learning from examples were used to construct fuzzy systems to perform system identification for a simple pendulum example. It was demonstrated that the MLFE outperformed LFE, and that both LFE and MLFE performed better when the uniform training algorithm was used to generate the training data set. Finally it was shown how the MLFE technique could be used to identify actuator failures on an F-16 aircraft.

For future work it is important to address the problems encountered with the computational complexity of the uniform training algorithm (due to its complexity it was not used to generate training data for the F-16 aircraft example). In addition, there is a significant need for both a mathematical and empirical comparative analysis of the various approaches to construct fuzzy systems to perform identification or estimation (e.g. as was demonstrated in Section 5, there is a need to compare MLFE and LFE in much more detail). For the aircraft example there is a need to consider the performance of the fuzzy estimator for a wider class of failures (including simultaneous failures) and for a more complex simulation testbed.

Acknowledgements—This work was supported in part by an Ohio State University Interdisciplinary Seed Grant and by National Science Foundation Grant IRI 9210332.

REFERENCES

1. Wang L.-X. Fuzzy systems are universal approximators. In *1st IEEE Conference on Fuzzy Systems*, pp. 1163–1170, March (1992).
2. Wang L. X. *Adaptive Fuzzy Systems and Control*. Prentice Hall, New Jersey (1994).
3. Magan A. T. Fuzzy parameter estimation for failure identification. Master's thesis, Department of Electrical Engineering, Ohio State University (1992).
4. Wang L.-X. Training of fuzzy logic systems using nearest neighborhood clustering. In *2nd IEEE Conference on Fuzzy Systems*, San Francisco, California, pp. 13–17 (1993).
5. Wang L.-X. and Mendel J. M. Generating fuzzy rules by learning from examples. *IEEE Trans. Systems, Man, Cybernetics* **22**, 1414–1427 (1992).
6. Tanaka H. and Ishibuchi H. Identification of possibilistic linear systems by quadratic membership functions of fuzzy parameters. *Fuzzy Sets and Systems* **41**, 145–160 (1991).
7. Wang L.-X. and Mendel J. M. Back-propagation fuzzy system as nonlinear dynamic system identifiers. *1st IEEE Conference on Fuzzy Systems*, San Diego, California, pp. 1409–1418 (1992).
8. Wang L.-X. and Mendel J. M. Fuzzy basis functions, universal approximation and orthogonal leastsquares learning. *IEEE Trans. Neural Networks* **3**, 1–8 (1992).
9. Sin S.-K. and deFigueiredo R. J. Fuzzy system design through fuzzy clustering and optimal predefuzzification. *2nd IEEE Conference on Fuzzy Systems*, San Francisco, California, pp. 190–195 (1993).
10. Laukonen E. G. and Passino K. M. Fuzzy systems for function approximation with applications to failure estimation. *IEEE Int. Symp. on Intelligent Control*, Columbus, OH, pp. 184–189, Aug. (1994).
11. Kwong W. A., Passino K. M. and Yurkovich S. Fuzzy learning systems for aircraft control law reconfiguration. *IEEE Int. Symp. Intelligent Control*, Columbus, Ohio, pp. 333–338, Aug. (1994).

APPENDIX

Proof of Theorem 2

The set of points on the boundary of the neighborhood $\beta(\mathbf{x}_1, \mathbf{x}_2)$ is defined by

$$X_l = \{\mathbf{x} \in IR^n : \|\mathbf{x} - \mathbf{x}_1\|_2 = \|\mathbf{x}_2 - \mathbf{x}_1\|_2\} \subset \beta(\mathbf{x}_1, \mathbf{x}_2). \quad (47)$$

Next, the set of points along a line defined by some $\mathbf{x}_b \in X_b$ and $\mathbf{x}_1$, is defined:

$$X_l = \{w \in [0, \|\mathbf{x}_2 - \mathbf{x}_1\|_2] : \mathbf{x} = \frac{\mathbf{x}_b - \mathbf{x}_1}{\|\mathbf{x}_b - \mathbf{x}_1\|_2} w + \mathbf{x}_1\} \subset \beta(\mathbf{x}_1, \mathbf{x}_2). \quad (48)$$

Here

$$\frac{\mathbf{x}_b - \mathbf{x}_1}{\|\mathbf{x}_b - \mathbf{x}_1\|_2}$$

is the unit direction vector of the line.

The problem is decomposed by considering the approximation error along the line X_l. The approximation error is rewritten as

$$\sup_{\mathbf{x} \in X_l} \{|f(\mathbf{x}) - g(\mathbf{x}; \boldsymbol{\theta})|\} = \max \left\{ \sup_{\mathbf{x} \in X_l} \{f(\mathbf{x}) - g(\mathbf{x}; \boldsymbol{\theta})\}, \sup_{\mathbf{x} \in X_l} \{g(\mathbf{x}; \boldsymbol{\theta}) - f(\mathbf{x})\} \right\}. \quad (49)$$

If just the points along the line given by X_l are considered, then $f(\mathbf{x})$ and $g(\mathbf{x}; \boldsymbol{\theta})$ can be written as functions of the scalar w. So for some $\mathbf{x}_b \in X_b$ and $w \in [0, \|\mathbf{x}_2 - \mathbf{x}_1\|_2]$

$$f\left(\frac{\mathbf{x}_b - \mathbf{x}_1}{\|\mathbf{x}_b - \mathbf{x}_1\|_2} w + \mathbf{x}_1\right) = f_b(w) g\left(\frac{\mathbf{x}_b - \mathbf{x}_1}{\|\mathbf{x}_b - \mathbf{x}_1\|_2} w + \mathbf{x}_1; \boldsymbol{\theta}\right) = g_b(w). \quad (50)$$

Moreover, the directional derivative along X_l is written as

$$\frac{\mathrm{d}}{\mathrm{d}w} f_b(w) = \frac{\partial f}{\partial x} \cdot \frac{\mathbf{x}_b - \mathbf{x}_1}{\|\mathbf{x}_b - \mathbf{x}_1\|_2} \frac{\mathrm{d}}{\mathrm{d}w} g_b(w) = \frac{\partial g}{\partial x} \cdot \frac{\mathbf{x}_b - \mathbf{x}_1}{\|\mathbf{x}_b - \mathbf{x}_1\|_2}. \quad (51)$$

Now, using the fundamental theorem of calculus the first term in (49) is considered. First, the upper limit of the integral is defined as

$$w^* \in \arg \sup_{w \in [0, \|\mathbf{x}_2 - \mathbf{x}_1\|_2]} \{f_b(w) - g_b(w)\}. \quad (52)$$

Then an expression which contains the first term of (49) can be written as

$$[f_b(w^*) - g_b(w^*)] - [f_b(0)] - g_b(0)] = \int_0^{w^*} \left(\frac{\mathrm{d}}{\mathrm{d}w} f_b(t) - \frac{\mathrm{d}}{\mathrm{d}w} g_b(t)\right) \mathrm{d}t. \quad (53)$$

$$[f_b(w^*) - g_b(w^*)] = \int_0^{w^*} \left(\frac{\mathrm{d}}{\mathrm{d}t} f_b(t)\right) \mathrm{d}t + [f_b(0) - g_b(0)] \quad (54)$$

Since $w^* \leq \|\mathbf{x}_2 - \mathbf{x}_1\|_2$ it is known that

$$\int_0^{w^*} \left(\frac{\mathrm{d}}{\mathrm{d}t} f_b(t) - \frac{\mathrm{d}}{\mathrm{d}t} g_b(t)\right) \mathrm{d}t \leq \left[\frac{\mathrm{d}}{\mathrm{d}w} f_b(w^*) - \frac{\mathrm{d}}{\mathrm{d}w} g_b(w^*)\right] \|\mathbf{x}_2 - \mathbf{x}_1\|_2. \quad (55)$$

By definition, $f_b(0) - g_b(0) < \varepsilon$. Given this,

$$[f_b(w^*) - g_b(w^*)] < \left[\frac{\mathrm{d}}{\mathrm{d}w} f_b(w^*) - \frac{\mathrm{d}}{\mathrm{d}w} g_b(w^*)\right] \|\mathbf{x}_2 - \mathbf{x}_1\|_2 + \varepsilon. \quad (56)$$

Now substituting in the expression for the directional derivative,

$$\sup_{w\in[0,\|\mathbf{x}_2-\mathbf{x}_1\|_2]}\left\{\frac{\mathrm{d}}{\mathrm{d}w}f_b(w)-\frac{\mathrm{d}}{\mathrm{d}w}g_b(w)\right\}$$

$$=\sup_{x\in X_l}\left\{\left(\frac{\partial f}{\partial \mathbf{x}}-\frac{\partial g}{\partial \mathbf{x}}\right)\left(\frac{\mathbf{x}_b-\mathbf{x}_1}{\|\mathbf{x}_b-\mathbf{x}_1\|_2}\right)\right\}, \quad (57)$$

and using the Schwarz inequality

$$\sup_{\mathbf{x}\in X_l}\left\{\left(\frac{\partial f}{\partial \mathbf{x}}-\frac{\partial g}{\partial \mathbf{x}}\right)\left(\frac{\mathbf{x}_b-\mathbf{x}_1}{\|\mathbf{x}_b-\mathbf{x}_1\|_2}\right)\right\}$$

$$\leq\sup_{\mathbf{x}\in X_l}\left\{\left\|\frac{\partial f}{\partial \mathbf{x}}-\frac{\partial g}{\partial \mathbf{x}}\right\|_2\left\|\frac{\mathbf{x}_b-\mathbf{x}_1}{\|\mathbf{x}_b-\mathbf{x}_1\|_2}\right\|_2\right\}$$

$$\leq\sup_{\mathbf{x}\in X_l}\left\{\left\|\frac{\partial f}{\partial \mathbf{x}}-\frac{\partial g}{\partial \mathbf{x}}\right\|_2\right\}. \quad (58)$$

Now, considering every $\mathbf{x}_b\in X_b$, then

$$\sup_{\mathbf{x}\in\beta(\mathbf{x}_1,\mathbf{x}_2)}\{f(\mathbf{x})-g(\mathbf{x};\boldsymbol{\theta})\}<$$

$$\sup_{\mathbf{x}\in\beta(\mathbf{x}_1,\mathbf{x}_2)}\left\{\left\|\frac{\partial f}{\partial \mathbf{x}}-\frac{\partial g}{\partial \mathbf{x}}\right\|_2\right\}\|\mathbf{x}_2-\mathbf{x}_1\|_2+\varepsilon. \quad (59)$$

Similarly, considering the second term of (49) gives

$$\sup_{\mathbf{x}\in\beta(\mathbf{x}_1,\mathbf{x}_2)}\{g(\mathbf{x};\boldsymbol{\theta})-f(\mathbf{x})\}<$$

$$\sup_{\mathbf{x}\in\beta(\mathbf{x}_1,\mathbf{x}_2)}\left\{\left\|\frac{\partial g}{\partial \mathbf{x}}-\frac{\partial f}{\partial \mathbf{x}}\right\|_2\right\}\|\mathbf{x}_2-\mathbf{x}_1\|_2+\varepsilon. \quad (60)$$

Then combining these two expressions gives the result:

$$\sup_{\mathbf{x}\in\beta(\mathbf{x}_1,\mathbf{x}_2)}\{|f(\mathbf{x})-g(\mathbf{x};\boldsymbol{\theta})\}<$$

$$\sup_{\mathbf{x}\in\beta(\mathbf{x}_1,\mathbf{x}_2)}\left\{\left\|\frac{\partial f}{\partial \mathbf{x}}-\frac{\partial g}{\partial \mathbf{x}}\right\|_2\right\}\|\mathbf{x}_2-\mathbf{x}_1\|_2+\varepsilon. \quad (61)$$

AUTHORS' BIOGRAPHIES

Eric Laukonen received his M.S. and B.S. in Electrical Engineering from the Ohio State University in 1994 and 1992 respectively. He has worked for General Electric Lighting, developing systems to manufacture lightbulbs. He has also worked for Battelle Memorial Institute, developing computer simulations and computer design tools for ultrasonics and other physical systems. He is presently employed by the Motorola Government Systems and Technology Group in Phoenix, AZ, and is part of a team developing satellite-based global cellular communications.

Kevin M. Passino received his M.S. and Ph.D in Electrical Engineering from the University of Notre Dame in 1989, and his B.S.E.E. from Tri-State University in 1983. He has worked in the control systems group at Magnavox Electronic Systems Co., on research in missile control, and at McDonnell Aircraft Co., on research in flight control. He is currently an Associate Professor in the Department of Electrical Engineering at The Ohio State University. He is an Associate Editor for the *IEEE Transactions on Automatic Control* and this journal, and has served as a Guest Editor for the *IEEE Control Systems Magazine* and the *IEEE Expert* magazine. He is General Chair of the *11th IEEE International Symposium on Intelligent Control*, 1996. He is also co-editor of *An Introduction to Intelligent and Autonomous Control* (Kluwer Academic Press), and a member of the IEEE Control Systems Society Board of Governors. His research interests include intelligent and autonomous control techniques, nonlinear analysis of intelligent control systems, failure detection and identification systems, and scheduling and stability analysis of flexible manufacturing systems.