4

Extended Kalman Filter Based Fuzzy Adaptive Filter

Wai Kit Wong and Heng Siong Lim
Faculty of Engineering and Technology, Multimedia University, Jln Ayer Keroh Lama, 75450 Melaka, Malaysia

1. Introduction

In classical logic, every statement is either true or false, i.e., it has a truth value of 1 or 0. Classical sets impose rigid membership requirements. Fuzzy logic, which is the principle of imprecise knowledge, was introduced by Lofti A. Zadeh in 1965 (Zadeh, L. A., 1965). It is an extension of classical logic dealing with the partial truth concept. Every statement in fuzzy logic is a matter of degree and exact reasoning is viewed as a limiting case of approximate reasoning. In fuzzy logic, classical/Boolean truth value is replaced with degree of truth. Degree of truth denotes the extent to which a proposition is true. In fuzzy logic, the degree of truth of a proposition may be any real number between 0 and 1, inclusive. This fuzzy truth represents membership in vaguely defined sets, not likelihood of some event or condition.

Fuzzy logic allows for set membership values between and including 0 and 1, shades of grey as well as black and white, and in its linguistic form, imprecise concepts like "slightly", "quite" and "very". Specifically it allows partial membership in a set. It is related to fuzzy sets and possibility theory. Fuzzy sets are an extension of classical set theory and are used in fuzzy logic (Zadeh, L. A., 1975). In classical set theory, the membership of elements in relation to a set is assessed in a crisp condition: either belongs to or not. In contrast, fuzzy set theory allows the gradual assessment of the membership of elements in relation to a set, with the aid of a membership function μ. A membership function may act as an indicator function, mapping all elements of fuzzy sets to real numbered value in the interval 0 and 1: $\mu \rightarrow [0,1]$. In general, there are 6 types of membership functions as depicted in Fig. 1.

2. Fuzzy logic system

A fuzzy logic system is composed of four principal components: a fuzzifier, a fuzzy rule base, a fuzzy inference engine and a defuzzifier (Mendel, J. M., 1995). It is an information processing system. Fig. 2 depicts a fuzzy logic system that is widely used in fuzzy logic controllers and signal processing applications.

The crisp inputs s, are first converted into fuzzy quantities u. This process is known as fuzzification, where a fuzzifier transforms crisp input values into linguistic values. Input values are translated into linguistic concepts, which are represented by fuzzy sets. Rules may be provided by experts or can be extracted from numerical data. In either case, engineering rules are expressed as a collection of IF-THEN statements. The inference engine

Source: Kalman Filter: Recent Advances and Applications, Book edited by: Victor M. Moreno and Alberto Pigazo, ISBN 978-953-307-000-1, pp. 584, April 2009, I-Tech, Vienna, Austria

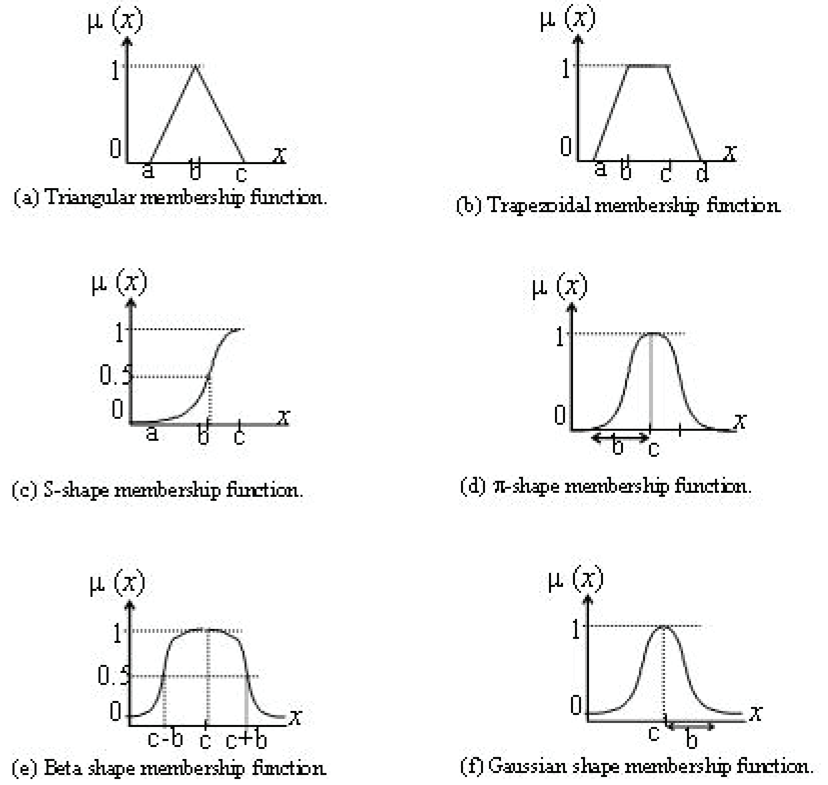

Fig. 1. Types of membership functions

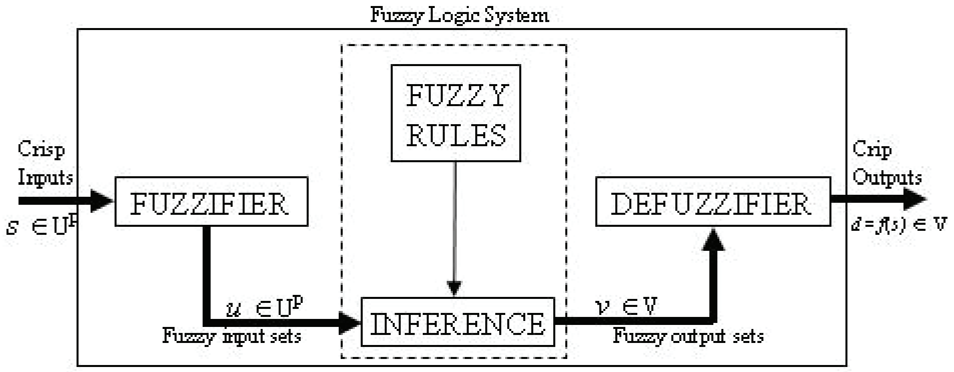

Fig. 2. Fuzzy logic system (Mendel, J.M., 1995)

of the fuzzy logic system maps fuzzy sets *u* into fuzzy sets *v*. It handles the way in which rules are combined. The defuzzifier maps output sets into crisp numbers. This mapping can be expressed quantitatively as $d = f(s)$. We shall consider an example of a simple temperature regulator that uses a fan. Fig. 3 shows the fuzzy sets and the membership functions for the input temperature.

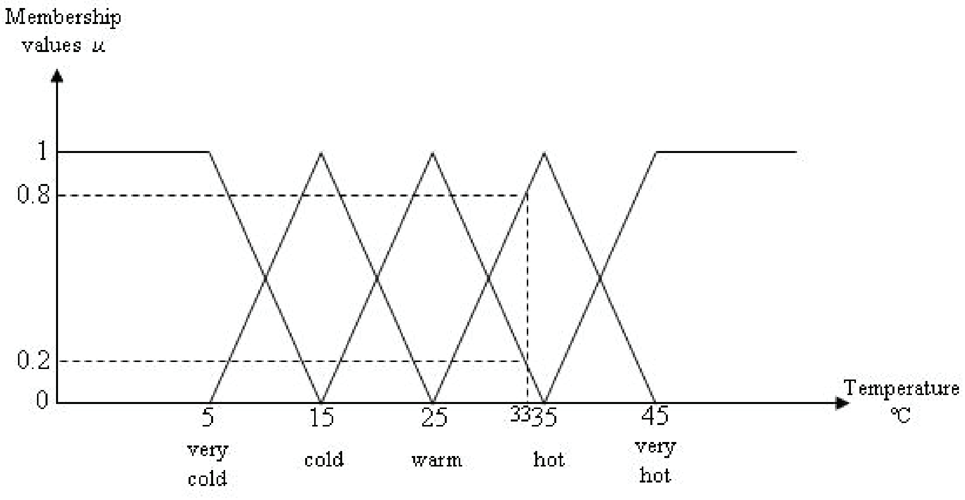

Fig. 3. Fuzzy sets of a simple temperature regulator that use a fan

A membership function μ_X, for $x \in X$, quantifies the grade of membership of the elements x to the fundamental set X. An element mapping to the value 0 means that the member is not included in the given set, 1 describes a fully included member. Values strictly between 0 and 1 characterize the fuzzy members. Membership functions are applied to the measurement and the degree of truth in each determined premise. According to Fig.3, if the input temperature s is 33ºC, after fuzzification, the membership value that s belongs to warm and hot temperature are $\mu_{warm} = 0.2$ and $\mu_{hot} = 0.8$ respectively.

The fuzzy rules may be provided by a human expert, or can be extracted from numerical input-output data pairs. In either case, engineering rules are expressed as a collection of IF-THEN statements, i.e.,

"IF temperature is very hot THEN speed maximum fan."
"IF temperature is hot THEN speed medium high fan."
"IF temperature is warm THEN maintain medium fan."
"IF temperature is cold THEN turn down medium low fan."
"IF temperature is very cold THEN stop fan."

These rules reveal that we will need an understanding of (Mendel, J.M., 1995):

1. Linguistic variables versus numerical values of the variable, e.g., very hot versus 45ºC.
2. Quantifying linguistic variables. E.g., u may have a finite number of linguistic terms associated with it, ranging from very hot to very cold, which is fuzzifying using fuzzy membership functions. There is no unique membership function in a situation and it is primarily subjective in nature. But this does not mean that membership function can be assigned arbitrarily, it is rather on the basis of application-specific criteria. Some of the commonly used membership functions are shown in Fig. 1 (Chen, S., 1990).
3. Implications, which is the relationship between two statements where the truth of one suggests the truth of the other, e.g., "IF temperature is warm THEN maintain medium

fan." Here, the truth of temperature is warm suggests the fan to maintain in medium speed.

4. Logical connections for linguistic variables, e.g., "and", "or", etc. "IF temperature is very hot and humidity is high, THEN speed medium high fan." Humidity is another fuzzy set. The fan speeds to medium high only when the two conditions are fulfilled.

Inference is the act or process of drawing a conclusion based solely on the fuzzy rules, e.g., *u* has $\mu_{warm} = 0.2$ and $\mu_{hot} = 0.8$, inference machine will draw a conclusion that the temperature is hot since $\mu_{hot} > \mu_{warm}$. Defuzzifier converts the fuzzy value into a "crisp" value. The defuzzifier maps output sets *v* into crisp numbers *d*. In the example mentioned above, *v* is speed medium high fan, when map into crisp number, *d* is the supply voltage value to the fan (e.g., 80 volts).

In summary, once the fuzzy rules have been established, the fuzzy logic system could map crisp inputs *s* into crisp outputs *d*. The mapping can be expressed quantitatively as $d = f(s)$. The fuzzifier maps crisp numbers *s* into fuzzy sets *u* in order to activate fuzzy rules that are in terms of linguistic variables, which have fuzzy sets associated with them. The inference machine maps fuzzy sets *u* into fuzzy sets *v*. It deals with the process in which rules are combined. The defuzzifier maps output sets *v* into crisp numbers *d*. In the sub-section below, we will discuss a well-known fuzzy logic system, which is the Takagi-Sugeno Kang (TSK) fuzzy logic system.

2.1 Takagi Sugeno Kang (TSK) fuzzy logic system

The Takagi-Sugeno Kang (TSK) fuzzy model is a universal approximator of the continuous real functions that are defined in a closed and bounded subset of *n*-dimensional real number $\Re^n$. This strong property of the TSK model finds several applications in modeling dynamical systems (Mastorakis, N.E., 2004). A TSK fuzzy logic system is described by fuzzy IF-THEN rules which represent input/output relations of a system. The most widely used TSK fuzzy logic system is the first-order TSK fuzzy logic system. It has a rule base of *M* rules, each having *p* antecedents, where the *l*-th rule is expressed as:

R^l: IF x_1 is F_1^l and x_2 is F_2^l and …and x_p is F_p^l

THEN $y^l = c_0^l + c_1^l x_1 + c_2^l x_2 + \ldots + c_p^l x_p$

in which l=1, 2,…, *M*; c_j^l is the consequent parameter, for j=0,1,…,p; x_j is the input to the fuzzy logic system; y^l is the output of the *l*-th IF-THEN rule; and F_j^l is the fuzzy sets, for j=0,1,…,p. The final output of the unnormalized first order TSK model is inferred as (Tanaka, K., 1998):

$$r = \sum_{l=1}^{M} f^l y^l \tag{1.1}$$

where f^l are rule firing strengths defined as:

$$f^l = \mathcal{T}_{j=1}^{p} \mu_{F_j^l}(x_j) \tag{1.2}$$

and $\mathcal{T}$ denotes a *t*-norm. *t*-norm is the short form of triangular norm. It is a kind of function used in multi-valued logic, especially in fuzzy logic. *t*-norm generalizes intersection in a lattice and AND in logic. The most often used *t*-norms are:

minimum *t*-norm : $\mathcal{T}_{min}(a,b) = \min\{a,b\}$

product *t*-norm : $\mathcal{T}_{prod}(a,b) = a.b$

drastic *t*-norm : $\mathcal{T}_{-1}(a,b) = \begin{cases} a \ , & \text{if } b = 1 \\ b \ , & \text{if } a = 1 \\ 0 & else \end{cases}$

When Gaussian membership functions

$$\mu_{F_j^l}(x_j) = \exp\left[-\frac{1}{2}\left(\frac{x_j - m_j^l}{\sigma_j^l}\right)^2\right] \tag{1.3}$$

and product *t*-norm are used, (1.1) can be expressed as:

$$r = \sum_{l=1}^{M} y^l \prod_{j=1}^{p} \exp\left[-\frac{1}{2}\left(\frac{x_j - m_j^l}{\sigma_j^l}\right)^2\right] \tag{1.4}$$

where m_j^l and σ_j^l are the centre and width of the *l*-th fuzzy set F_j^l, respectively.

2.2 Application of fuzzy logic system

Fuzzy logic systems deal with reasoning with fuzzy sets, fuzzy rules, and estimated sampled functions from linguistic input to linguistic output. Fuzzy logic systems are successful in the field of control especially the feed back control of some physical and chemical processes like electric current, temperature, motion of machines and flow of liquids or gas. Fuzzy logic principles are also be applied in fuzzy software engineering that incorporate fuzziness in data and programs and fuzzy database systems in the field of economics, management and medicines (Gupta, M.M., 1994[a], Gupta, M.M., 1994[b], Jang, J.S.R., 1993, Kaufmann, A., 1988). Recently, some automotive industry products and consumer electronics in the market have moved into fuzzy logic technology, and the outcome of the products has significant performance improvement (Al-Holou, N., 2002, Eichfeld, H., 1996).

Fuzzy logic systems are nonlinear systems and they are capable of inferring complex nonlinear relationships between input and output variables (Mendel, J.M., 1997). The non-linearity property is particularly important when the underlying physical mechanism to be modeled is inherently nonlinear. The system can 'learn' the non-linear mapping by being presented with a sequence of input signals and desired response pairs, which are used in conjunction with an optimization algorithm to determine the values of the system parameters. This is one of the most commonly used learning paradigms, called *supervised learning*. Even if the process to be modeled is non-stationary, the system can be updated to reflect the changing statistics of the process. Unlike conventional stochastic models used to model such processes, fuzzy logic systems do not make any assumptions regarding the structure of the process, nor do they invoke any kind of probabilistic distribution model, i.e., they belong to the general family of model free, data driven, non-parametric methods.

However, conventional fuzzy systems have limitation of low capabilities for learning and adaptation. An improvement done in (Gupta, M.M.,1991) combines conventional fuzzy technology with neural network technology to form an innovative technological field, so called fuzzy neural networks (FNNs). Fuzzy mathematics gives an inference mechanism for approximate reasoning under cognitive uncertainty, while neural networks have the abilities of pattern recognition, optimization and decision making (Bhatti, S.S., 2002). A combination of these two technological innovations give birth to a new technology in which the explicit knowledge representation of fuzzy logic is improved by the learning power of simulated neural networks (Bhatti, S.S., 2002). Fuzzy basis function network (FBFN) is one of the FNNs that are used for information processing. It rediscovers some interesting advantages of a fuzzy system. One is the universal approximation capability, and the other is learning and adaptation, which have not been dealt with in fuzzy systems. Fuzzy adaptive equalizers are based on such technology. This technology is also capable of dealing with nonlinearity and uncertainty.

3. Equalizer

Consider, for example, a communication system which consists of a transmitter, communication channel, receiver, and equalizer connected together as shown in Fig. 4. The term equalizer in communication system is commonly refer to a device that place after the receiver designed to equalize the channel characteristics and extract out information send by transmitter side from noise and distortion.

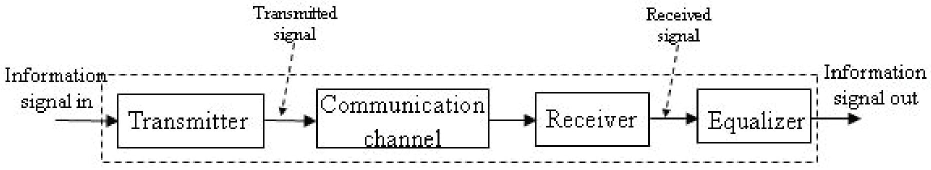

Fig. 4. Block diagram of a communication system

The main problem that has to be considered in communication system involves the fading and intersymbol interference (ISI), which is generated by multipath propagation effects and its resulting delay spread. Sometimes, there are obstacles and reflectors in the communication channel. The transmitted signal arrives at the receiver from various directions over a multiplicity of paths. Such a phenomenon is called multipath. It is an unpredictable set of reflections from direct waves and each with its own degree of attenuation and delay.
In communication channel, multiple reflections of the transmitted signal may arrive at the receiver at different times, this can result in intersymbol interference (or bits "crashing" into one another) which the receiver cannot sort out. This time dispersion of the channel is called multipath delay spread which is an important parameter to degrade the performance of communication systems.Besides signal distortion due to multipath propagation, noise is the most crucial factor that degrades the performance of communication systems. Since communication systems have such a lot of undesired distortions, equalizers are designed to compensate these distortions.

3.1 Optimal equalizer (MAP equalizer)

Maximum a-posteriori (MAP) equalizer is the optimal equalizer based on maximum a posteriori probability estimation. MAP equalization requires the knowledge of the conditional probability density function (PDF) of the received signal given the transmitted signal pattern. Assume that we want to estimate an unobservable population θ on the basis of observations y. Let f be the sampling distribution of y, so that $f(y \mid \theta)$ is a conditional PDF, summarizing our knowledge provided by the data y conditioned on knowing θ. Then the function:

$$\theta \mapsto f(y \mid \theta) \tag{1.5}$$

is known as the likelihood function and the estimate (Kay, S.M., 1993):

$$\hat{\theta}_{ML}(y) = \arg \max_{\theta} f(y \mid \theta) \tag{1.6}$$

as the maximum likelihood estimate of θ. Now, assume that a prior distribution g over θ exists. Applying Bayesian statistics, we may treat θ as a random variable with posterior distribution as:

$$\theta \mapsto \frac{f(y \mid \theta) g(\theta)}{\int_{\theta} f(y \mid \theta) g(\theta) d\theta} \tag{1.7}$$

This is an application of Bayes' theorem. The method of maximum a-posteriori estimation is then estimates θ as the mode of the posterior distribution with:

$$\hat{\theta}_{MAP}(y) = \arg \max_{\theta} \frac{f(y \mid \theta) g(\theta)}{\int_{\theta} f(y \mid \theta) g(\theta) d\theta} = \arg \max_{\theta} f(y \mid \theta) g(\theta) \tag{1.8}$$

MAP estimate of θ coincides with maximum likelihood estimate when the prior distribution function g is uniform. The geometric formulation of MAP equalizer is given below.

3.2 Geometric formulation for MAP equalizer

The geometric formulation of the MAP equalizer is shown in (Chen, S., 1990) and (Chen, S., 1991). We are using the same notation as in those studies and we define:

$$P_{\eta,\tau}(1) = \{\hat{\mathrm{x}}(k) \in \Re^{\eta} \mid s(k-\tau) = 1\} \tag{1.9}$$

and

$$P_{\eta,\tau}(-1) = \{\hat{\mathrm{x}}(k) \in \Re^{\eta} \mid s(k-\tau) = -1\} \tag{1.10}$$

where η is the order, τ is the lag of the equalizer, $\Re$ is any real number,

$$\hat{\mathrm{x}}(k) = [\hat{x}(k) \quad \hat{x}(k-1) \quad \ldots \quad \hat{x}(k-\eta+1)]^T \tag{1.11}$$

$\hat{x}(k)$ is the noise-free output of the rake receiver, and $P_{\eta,\tau}(1)$ and $P_{\eta,\tau}(-1)$ are two sets of possible noise free output vectors $\hat{\mathrm{x}}(k)$ that can be produced from input sequences containing $s(k\text{-}\tau) = 1$ and $s(k\text{-}\tau) = -1$, respectively. Let:

$$\mathrm{x}(k) = [x(k) \quad x(k-1) \quad \ldots \quad x(k-\eta+1)]^T \tag{1.12}$$

be the observed output vector with noise, $p_1[\mathrm{x}(k) \mid \hat{\mathrm{x}}(k) \in P_{\eta,\tau}(1)]$ and $p_{-1}[\mathrm{x}(k) \mid \hat{\mathrm{x}}(k) \in P_{\eta,\tau}(-1)]$ be the conditional probability density functions of $x(k)$ given $\hat{\mathrm{x}}(k) \in P_{\eta,\tau}(1)$ and $\hat{\mathrm{x}}(k) \in P_{\eta,\tau}(-1)$ respectively. It was shown in (Chen, S., 1990) and (Chen, S., 1991), which the MAP equalizer is defined by:

$$f_{opt}(\mathrm{x}(k)) = \mathrm{sgn}[p_1(\mathrm{x}(k) \mid \hat{\mathrm{x}}(k) \in P_{\eta,\tau}(1)) - p_{-1}(\mathrm{x}(k) \mid \hat{\mathrm{x}}(k) \in P_{\eta,\tau}(-1))] \tag{1.13}$$

This optimal equalizer achieves the minimum bit error rate for the given order η and τ, where sgn(q) = 1 (-1), if $q \geq 0$ ($q<0$). If the noise is Gaussian and with covariance matrix:

$$Q = E\{[(n(k) \quad \ldots \quad n(k-\eta+1)][n(k) \quad \ldots \quad n(k-\eta+1)]^T\} \tag{1.14}$$

Then from $x(k) = \hat{x}(k) + n(k)$, we have:

$$p_1(\mathrm{x}(k) \mid \hat{\mathrm{x}}(k) \in P_{\eta,\tau}(1)) - p_{-1}(\mathrm{x}(k) \mid \hat{\mathrm{x}}(k) \in P_{\eta,\tau}(-1))$$

$$= \sum \exp\left[-\frac{1}{2}(\mathrm{x}(k) - \hat{\mathrm{x}}_+)^T Q^{-1} (\mathrm{x}(k) - \hat{\mathrm{x}}_+)\right] - \sum \exp\left[-\frac{1}{2}(\mathrm{x}(k) - \hat{\mathrm{x}}_-)^T Q^{-1} (\mathrm{x}(k) - \hat{\mathrm{x}}_-)\right] \tag{1.15}$$

where the first summation is over all the positive noise free points $\hat{\mathrm{x}}_+ \in P_{\eta,\tau}(1)$ whereas the second summation is over all the negative noise free points $\hat{\mathrm{x}}_- \in P_{\eta,\tau}(-1)$.

In practice, it is difficult to know or predict the PDF of the transmitted or received signals, and MAP equalizer is too complex for practical use. Therefore, attention has been given to the design of sub-optimum equalizers such as adaptive equalizers that are practical and have near optimal performance.

4. Adaptive equalizer

Adaptive equalizers are sub-optimum equalizers that rely on a recursive algorithm to perform satisfactory information extraction in a communication system in which the statistical characteristics about the input-output signal are not known. The algorithm will start from some predetermined set of initial conditions, representing whatever the system knows about the communication channel. In a stationary communication channel, after successive iterations of the algorithm, the equalizer will converges to the optimum solution in some statistical sense. In a non-stationary communication channel, the algorithm will also have the ability to track time variations in the statistic of the input data (Haykin, S. 2002). Some adaptive equalizers examples are shown below:

4.1 Transversal equalizer

Digital equalizer is well known and was already described in 1940 by H. J. Kallman in (Kallman, H.J., 1940). The invention relates to a simple and linear transversal equalizer for processing an analog signal with a number of stages to which level control devices are connected by uniformly spaced taps in a non-dispersive delay line. The transversal equalizer is typically implemented using digital circuitry, charged-coupled devices, or surface-acoustic wave devices. Due to its versatility and ease of implementation, the transversal equalizer has emerged as an essential signal processing structure in a wide variety of applications.

In signal processing, transversal equalizer is also known as tapped-delay line equalizer or finite–duration impulse response (FIR) equalizer (Proakis, J.G.,1995). The structure of an adaptive transversal equalizer is depicted in Fig. 5:

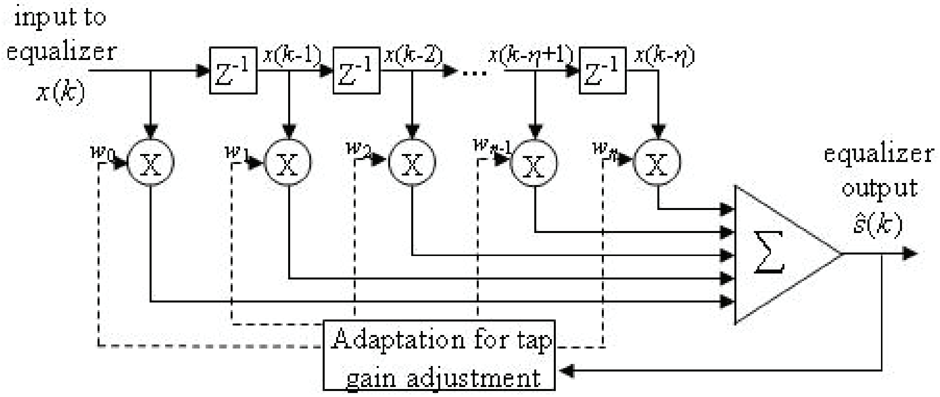

Fig. 5. Structure of transversal equalizer

The output of the transversal equalizer is given by:

$$\hat{s}(k) = \sum_{l=0}^{\eta} w_l x(k-l) . \tag{1.16}$$

where w_l is the l-th tap weight and η is the order of the equalizer. The weights of the equalizer can be optimized by minimizing some criterion functions. Two popular choices for the adaptation algorithm are the RLS and LMS algorithm.

4.2 Decision feedback equalizer

Another sub-optimum adaptive equalizer is the decision feedback equalizer (DFE). DFE utilizes two transversal equalizers: a feedforward transversal equalizer and a feedback transversal equalizer (Sailer, T.M., 2001). The DFE uses previous symbol estimates for interference cancellation of ISI corrupted data transmission. When the symbol estimates are correct, they have the advantage of not being corrupted by additive channel noise, giving the DFE a performance advantage over other linear structures. However, if the symbol estimates are incorrect, there is the danger of error propagation leading to catastrophic performance. The structure of DFE is shown in Fig. 6:

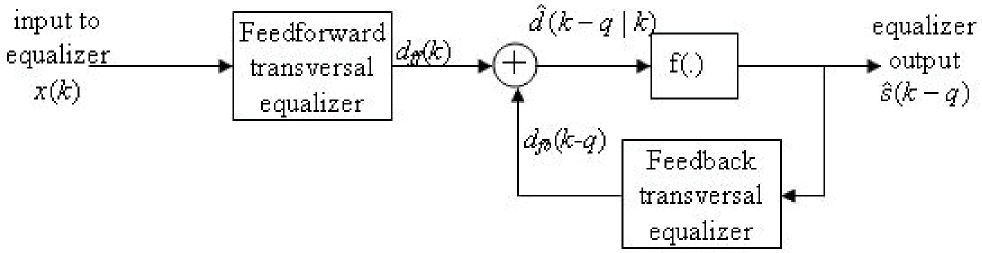

Fig. 6. Structure of decision feedback equalizer

where f(.) is a decision device and $x(k)$ is the input to the feedforward transversal equalizer. From the output of the feedforward transversal equalizer, $d_{ff}(k)$, the interference from previously detected symbols are removed via the output of the feedback transversal equalizer, $d_{fb}(k\text{-}q)$. The difference between these two transversal equalizer outputs constitutes an estimate of the transmitted symbol, $\hat{d}(k-q\,|\,k)$. This estimate is sometimes called *soft estimate*, since it is not yet quantized. The decision device quantizes the soft estimate and the resulting *hard estimate*, $\hat{s}(k-q)$ becomes the input of the feedback transversal equalizer. The constant q is known as the decision delay or the smoothing lag. It specifies how many future measurements are being processed before a decision is made on the present symbol.

4.3 Volterra series expansion equalizer

Since the optimal decision boundary is normally nonlinear for communication channel equalization problem, linear equalization methods are no longer adequate for the task. In this case, nonlinear equalizers that have the ability to perform nonlinear input-output mapping can be applied to minimize the error probability. In this and the following two subsections, we discuss three nonlinear channel equalization methods, namely the Volterra series expansion, the radial basis function and the multilayer perceptrons. The Volterra series expansion equalizer is a nonlinear equalizer based on the Volterra series functional representation from mathematics (Kong, X., 2004).

Let $x[n]$ and $y[n]$ represent the input and output signals, respectively, of a discrete time and causal nonlinear system. The Volterra series expansion for $y[n]$ using $x[n]$ is given by (Mathews, V.J., 1991):

$$y[n] = h_0 + \sum_{m_1=0}^{\infty} h_1[m_1]x[n-m_1] + \sum_{m_1=0}^{\infty}\sum_{m_2=0}^{\infty} h_2[m_1,m_2]x[n-m_1]x[n-m_2] + \ldots$$

$$+ \sum_{m_1=0}^{\infty}\sum_{m_2=0}^{\infty}\ldots\sum_{m_p=0}^{\infty} h_p[m_1,m_2,\ldots,m_p]x[n-m_1]x[n-m_2]\ldots x[n-m_p] + \ldots \tag{1.17}$$

where $h_p[m_1,m_2,\ldots,m_p]$ is known as the p-th order Volterra kernel of the system. $h_p[m_1,m_2,\ldots,m_p]$ can be optimized by minimizing some criterion functions. Among the most commonly used algorithm are the recursive least squares (RLS) and least mean squares (LMS) algorithms. Without any loss of generality, it is assumed that the Volterra kernels are symmetric, in which $h_p[m_1,m_2,\ldots,m_p]$ is left unchanged for any of the possible p! permutations of the indices $m_1,m_2,\ldots,m_p$ (Mathews, V.J., 1991).

Since an infinite series expansion like (1.17) is not useful in channel equalization, we may work with truncated Volterra series expansion (Mathews, V.J., 1991):

$$y[n] = \sum_{m_1=0}^{N-1} h_1[m_1]x[n-m_1] + \sum_{m_1=0}^{N-1}\sum_{m_2=0}^{N-1} h_2[m_1,m_2]x[n-m_1]x[n-m_2] + \ldots$$

$$+ \sum_{m_1=0}^{N-1}\sum_{m_2=0}^{N-1}\ldots\sum_{m_p=0}^{N-1} h_p[m_1,m_2,\ldots,m_p]x[n-m_1]x[n-m_2]\ldots x[n-m_p] \tag{1.18}$$

where $h_0 = 0$ (without loss of generality (Mathews, V.J., 1991)). Note that there are $O(N^p)$ coefficients in this polynomial expansion. One big disadvantage for the model as in (1.18) is that the complexity of implementing equalizers using this model can be very large even for moderately large values of N and P. Consequently, most Volterra series expansions system involve low order model.

4.4 Radial basis function equalizer

A radial basis function (RBF) equalizer is a neural network equalizer whose outputs are a linear combination of the hidden layer functions (Mulgrew, B., 1996). It is trained to perform a mapping from an m-dimensional input space to an n-dimensional output space. The structure of RBF equalizer is shown in Fig. 7 below.

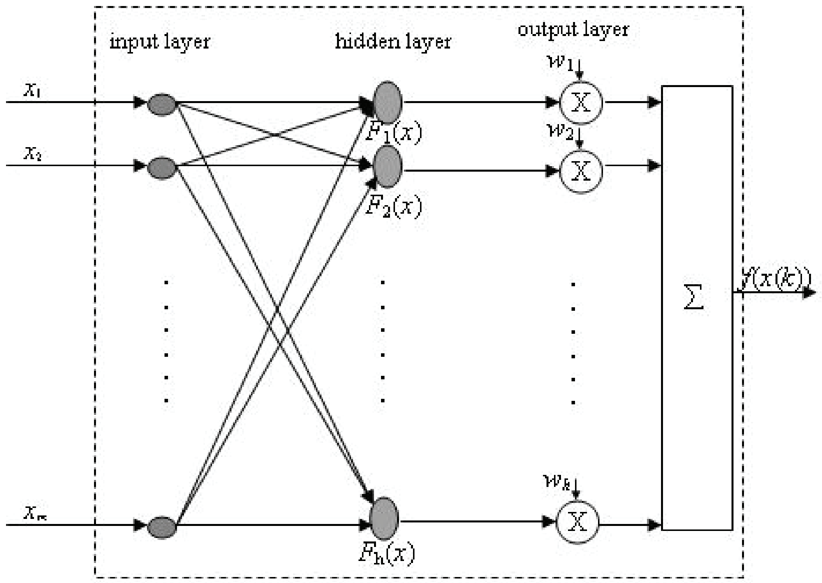

Fig. 7. Structure of radial basis function equalizer

Mathematically, the output of a RBF equalizer is,

$$f(x(k)) = \sum_{l=1}^{h} w_l F_l(\boldsymbol{x}) \tag{1.19}$$

where the basis function F is a sigmoidal function

$$F(\boldsymbol{x}) = \exp\left(-\frac{1}{\sigma_l^2}\left\|\boldsymbol{x}-\mu_l\right\|^2\right) \tag{1.20}$$

$\boldsymbol{x}$ is the input vector of the equalizer $[x_1 \ \ x_2 \ \dots \ \ x_m]$, h indicates the total number of hidden neurons, μ_l and σ_l refer to the center and width of the l-th hidden neuron. $||.||$ is the Euclidean norm. The coefficient w_l is the weight of the l-th hidden neuron to the output neuron.

4.5 Multilayer perceptrons equalizer

Another approach for nonlinear channel equalization using neural network is the multilayer perceptrons (MLP) equalizer. A typical MLP equalizer consists of a set of source nodes forming the input layer, one or more hidden layers of computation nodes, and an output layer of nodes. The input signal propagates through the MLP equalizer layer-by-layer. The signal-flow of such an equalizer is shown in Fig. 9. In between the input layer and the output layer are the hidden layers of the MLP equalizer. The MLP equalizer has L layers of synaptic connections and *L*+1 layers of neurons.

In (Seung, S., 2002), back propagation algorithm is used in multilayer perceptrons network. Assume there are no biases, the network is diagrammed as:

$$x^0 \xrightarrow{W^1} x^1 \xrightarrow{W^2} \cdots \xrightarrow{W^L} x^L \tag{1.21}$$

where $x^l \in \Re^{n_l}$ for all *l* = 0,1, …, *L*; $x^l = [x_1^l \; x_2^l \ldots x_h^l]$ and W^l is an $n_l \times n_{l-1}$ matrix for all *l* = 0,1, …, *L*. n_l is the number of hidden neurons in *l*-th layer. There are *L* + 1 layers of neurons. The input vector x^0 is transformed into the output vector x^L by evaluating the equation:

$$x_i^l = \sum_{j=1}^{n_l-1} W_{ij}^l x_j^{l-1} \text{ for } l = 1 \text{ to } L. \tag{1.22}$$

The actual output of the equalizer is

$$f(x(k)) = \sum_{i=1}^{h} x_i^L \tag{1.23}$$

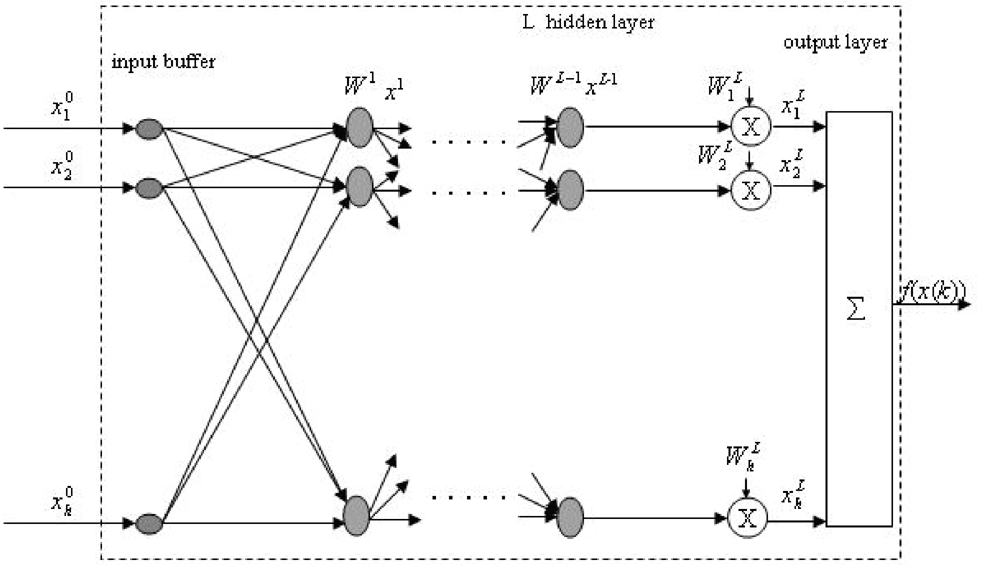

Fig. 9. Structure of multilayer perceptron equalizer

Two types of linear adaptive equalizer had been discussed, namely transversal equalizer and decision feedback equalizer; at the meantime, three types of nonlinear channel are also discussed, namely Volterra series expansion equalizer, radial basis function equalizer and multilayer perceptron equalizer. Normally, complex communication channels are nonlinear channel. Therefore most of the adaptive equalizers used today are nonlinear adaptive equalizer, because they not only dealing well with linear channel characteristic, but nonlinear channel as well. Since nonlinear channels include a very broad spectrum of nonlinear distortion, it is difficult to comment on which nonlinear adaptive equalizer is dominantly better than the others. So, it is good to try out new nonlinear adaptive equalizer for communication channel equalization. Fuzzy adaptive equalizer is such a new nonlinear equalizer.

5. Fuzzy adaptive equalizer

Fuzzy adaptive equalizers are adaptive equalizers that apply the concepts of fuzzy logic. Fuzzy adaptive equalizers are information processors that make use of both linguistic (in the form of fuzzy IF-THEN rules) and numerical information (in the form of input-output pairs). The main merits of using fuzzy adaptive equalizers are nonlinear and simple in design, which linguistic information from human experts can be directly incorporated into the equalizer. If no linguistic information is available, the fuzzy adaptive equalizers become well-defined nonlinear adaptive equalizers (similar to the polynomial, neural nets, or radial basis function adaptive equalizers). The adaptive algorithms adjust the parameters of the membership functions which characterize the fuzzy concepts in the IF-THEN rules, by minimizing some criterion function.
Fuzzy adaptive equalizer, as a fuzzy basis function expansions system, can be represented as two-layered feedforward network structure (Wang, L.X., 1992[b]). On the basis of this idea, the fuzzy adaptive equalizer can be trained to realize the desired input-output relationship using various learning algorithms such as least mean squares (LMS), recursive least squares (RLS) and extended Kalman filter (EKF) adaptation algorithms. Fig. 10 shows the schematic diagram of fuzzy adaptive equalizer. The inputs to the fuzzy adaptive equalizer [$x(k)$, $x(k-1)$,…, $x(k-n+1)$] are the receiver's outputs. The task of the fuzzy adaptive equalizer at the sampling instant k is to produce an estimate of the transmitted symbol $\hat{s}(k-d)$, using the information contained in [$x(k)$, $x(k-1)$,…, $x(k-n+1)$] where the integer n and d are the order and lag of the equalizer respectively.
A fuzzy adaptive equalizer is functionally equivalent to a fuzzy basis function network (FBFN) with the form given in (1.24) to (1.27). It can be shown that the input-output equations of a fuzzy adaptive equalizer with singleton fuzzifier, product inference and centroid defuzzifier can be expressed as:

$$f(x(k)) = \frac{\sum_{l=1}^{M} \theta^l \left(\prod_{i=1}^{n} \mu_{F_i^l}(x_i) \right)}{\sum_{l=1}^{M} \left(\prod_{i=1}^{n} \mu_{F_i^l}(x_i) \right)} = \sum_{l=1}^{M} \theta^l \phi^l(\mathrm{x}) \tag{1.24}$$

$$\text{where} \quad \phi^l(\mathrm{x}) = \frac{\prod_{i=1}^{n} \mu_{F_i^l}(x_i)}{\sum_{l=1}^{M}\left(\prod_{i=1}^{n} \mu_{F_i^l}(x_i)\right)} \tag{1.25}$$

$\phi^l(\mathrm{x})$ are called the fuzzy basis functions. Equation (1.24) gives an expression for fuzzy adaptive equalizer as shown in Fig.10. $\boldsymbol{x}$ =[x_1 x_2 …$x_{n\text{-}1}$]=[$x(k)$ $x(k\text{-}1)$ … $x(k\text{-}n\text{+}1)$] is the vector of inputs to the fuzzy adaptive equalizer. In particular, if a Gaussian radial basis function is chosen as the membership function, then

$$\phi^l(\mathrm{x}) = \frac{u_l(\mathrm{x})}{\sum_{l=1}^{M} u_l(\mathrm{x})} \tag{1.26}$$

$$\text{where} \quad u_l(\boldsymbol{x}) = \left(\prod_{i=1}^{n} \mu_{F_i^l}(x_i)\right) = \prod_{i=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_i - \tilde{x}_i^l}{\sigma_i^l}\right)\right) \tag{1.27}$$

$u_l(\boldsymbol{x})$ is the membership function after product inference, $\tilde{x}_i^l$ is the center of the i-th membership function and σ_i^l represents the width of the i-th membership function. In the next two sections, we will discuss two examples of fuzzy adaptive equalizers that were proposed in (Wang, L.X., 1993) for nonlinear channel equalization.

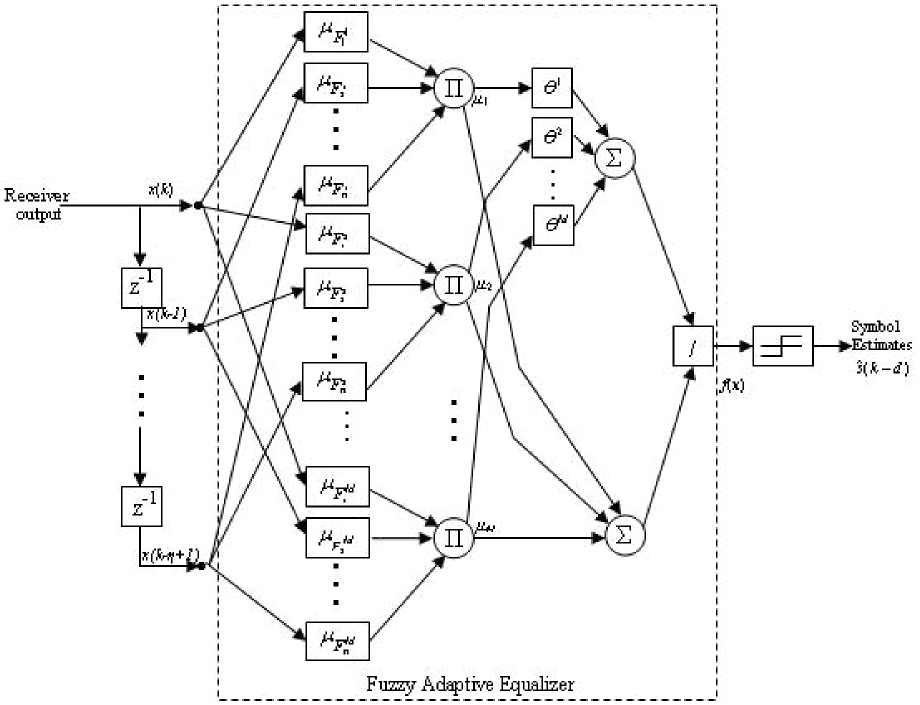

Fig. 10. Schematic diagram of fuzzy adaptive equalizer.

6. RLS based fuzzy adaptive equalizer

In (Wang, L.X., 1993), the RLS based fuzzy adaptive equalizer is used to solve the following problem. Consider a real-valued vector input sequence [*x*(*k*)] and a real-valued scalar sequence [*d*(*k*)], where *k*=0,1,2,… is the time index, and $x(k) \in U \equiv [C_1^-, C_1^+] \times [C_2^-, C_2^+] \times \ldots \times [C_n^-, C_n^+] \subset \Re^n$. U and $\Re$ are the input and output spaces of the equalizer respectively. At each time point *k*, the values of *x*(*k*) and *d*(*k*) are given. The problem to be solved is to determine a fuzzy adaptive equalizer $f_k : U \subset \Re^n \to \Re$ such that:

$$J(k) = \sum_{i=0}^{k} \lambda^{k-i} [d(i) - f_k(\boldsymbol{x}(i))]^2 \tag{1.28}$$

is minimized, where $\lambda \in (0,1]$ is a forgetting factor.

6.1 Design procedure of the RLS based fuzzy adaptive equalizer

Step 1: m_i fuzzy sets are defined in each interval $\left[C_i^-, C_i^+\right]$ of the input space U, which are labeled as F_i^{ji} (*i* = 1, 2, …, *n*; *ji* = 1, 2, …, m_i; *ji* is a single index, i.e., *j*1 is an index which takes values from 1 to m_1 for *i*=1), in the following way: the m_i membership functions $\mu_{F_i^{ji}}$ cover the interval $\left[C_i^-, C_i^+\right]$ in the sense that for each $x_i \in \left[C_i^-, C_i^+\right]$ there exist at least one $\mu_{F_i^{ji}}(x_i) \neq 0$. These membership functions are fixed and will not change during the adaptation procedure.

Step 2: A set of $\prod_{i=1}^{n} m_i$ fuzzy IF-THEN rules is constructed in the following form:

$$R^{(j1,\ldots,jn)} : \text{IF } x_1 \text{ is } F_1^{j1} \text{ and } \ldots \text{ and } x_n \text{ is } F_n^{jn}, \text{ THEN } d \text{ is } G^{(j1,\ldots,jn)}. \tag{1.29}$$

where $\mathrm{x} = \left[x_1, \ldots, x_n\right]^T = \left[x(k), \ldots, x(k-n+1)\right]^T \in U$ is the equalizer input, $d \in \Re$ is the equalizer output, *ji* = 1, 2, …, m_i with *i* =1, 2, …, *n*, F_i^{ji}'s are the same labels of the fuzzy sets defined in Step 1, and the $G^{(j1,\ldots,jn)}$'s are labels of fuzzy sets defined in the output space which are determined in the following way: if there are linguistic rules from human experts in the form of (1.29), $G^{(j1,\ldots,jn)}$ is set to the corresponding linguistic terms of these rules; otherwise, $\mu_{G^{(j1,\ldots,jn)}}$ is set to an arbitrary membership function over the output space $\Re$. It is in this way that linguistic rules are incorporated into the fuzzy adaptive equalizer.

Step 3: The filter output f_k is calculated based on the $\prod_{i=1}^{n} m_i$ rules in step 2 as follows:

$$f_k(\mathrm{x}) = \frac{\sum_{j1=1}^{m_1} \cdots \sum_{jn=1}^{m_n} \theta^{(j1,\ldots,jn)} \left(\mu_{F_1^{j1}}(x_1) \ldots \mu_{F_n^{jn}}(x_n)\right)}{\sum_{j1=1}^{m_1} \cdots \sum_{jn=1}^{m_n} \left(\mu_{F_1^{j1}}(x_1) \ldots \mu_{F_n^{jn}}(x_n)\right)} \tag{1.30}$$

where $\mathrm{x}=[x_1,...,x_n]^T=[x(k),...,x(k-n+1)]^T \in U$, $\mu_{F_i^{ji}}$'s are membership functions defined in Step 1, and $\theta^{(j1,...,jn)} \in \Re$ is the point at which $\mu_{G^{(j1,...,jn)}}$ achieves its maximum value. Due to the way in which the $\mu_{F_i^{ji}}$'s are defined in Step 1, the denominator of (1.30) is nonzero for all the points of *U*; therefore the filter f_k of (1.30) is well defined. For a given input $\mathrm{x} \in U$, the equalizer output is determined as a weighted average of the $\prod_{i=1}^{n} m_i$ points $\theta^{(j1,...,jn)}$ in the output space at which the fuzzy sets $G^{(j1,...,jn)}$ of the THEN parts of the $\prod_{i=1}^{n} m_i$ rules have maximum membership values; and, the weight $\mu_{F_1^{j1}}(x_1)...\mu_{F_n^{jn}}(x_n)$ for $\theta^{(j1,...,jn)}$ is proportional to the membership values of which *x* satisfies the IF part of $R^{(j1,...,jn)}$.

6.2 Parameter adaptation of the RLS based fuzzy adaptive equalizer

In (1.30), the weights $\mu_{F_1^{j1}}(x_1)...\mu_{F_n^{jn}}(x_n)$ are fixed functions of *x*; therefore the free design parameters of the fuzzy adaptive equalizer are the $\theta^{(j1,...,jn)}$'s which are collected as a $\prod_{i=1}^{n} m_i$ -dimensional vector (Wang, L.X., 1993):

$$\theta=[\theta^{(1,1,...,1)},...,\theta^{(m_1,1,...,1)},\theta^{(1,2,...,1)},...,\theta^{(m_1,2,1,...,1)},...,\theta^{(1,m_2,1,...,1)},...,\theta^{(m_1,m_2,1,...,1)},...,\theta^{(1,m_2,...m_n)},...,\theta^{(m_1,m_2,...m_n)}]^T \quad (1.31)$$

The fuzzy basis function is defined by

$$p^{(j1,...,jn)}(\mathrm{x})=\frac{\mu_{F_1^{j1}}(x_1)...\mu_{F_n^{jn}}(x_n)}{\sum_{j1=1}^{m_1}...\sum_{jn=1}^{m_n}\left(\mu_{F_1^{j1}}(x_1)...\mu_{F_n^{jn}}(x_n)\right)} \quad (1.32)$$

and they are collected as a $\prod_{i=1}^{n} m_i$ -dimensional vector ***p***(*x*) in the same ordering as the $\boldsymbol{\theta}$ of (1.31), i.e.,

$$\mathrm{p}(\mathrm{x})=(p^{(1,1,...,1)}(\mathrm{x}),...,p^{(m_1,1,...,1)}(\mathrm{x}),p^{(1,2,1,...,1)}(\mathrm{x}),...,p^{(m_1,2,1,...,1)}(\mathrm{x}),...,p^{(1,m_1,1,...,1)}(\mathrm{x})$$
$$,...,p^{(m_1,m_2,1,...,1)}(\mathrm{x}),...,p^{(1,m_2,...,m_n)}(\mathrm{x}),...,p^{(m_1,m_2,...,m_n)}(\mathrm{x}))^T \quad (1.33)$$

Based on (1.31) and (1.33), (1.30) can be rewritten as:

$$f_k(\boldsymbol{x})=\mathrm{p}^T(\mathrm{x})\theta \quad (1.34)$$

The following RLS algorithm is used to update θ. The initial estimate of $\boldsymbol{\theta}$, $\boldsymbol{\theta}(0)$, is determined as in Step 2, and *P*(0)=*σI*, where *σ* is a small positive constant, and *I* is the $\prod_{i=1}^{n} m_i$ -by- $\prod_{i=1}^{n} m_i$ identity matrix. At each time point *k*=1,2, ..., the following equation are computed:

$$\phi(k) = \mathrm{p}(\mathrm{x}(k)) \tag{1.35}$$

$$P(k) = \frac{1}{\lambda}\Big[P(k-1) - P(k-1)\phi(k)[\lambda + \phi^T(k)P(k-1)\phi(k)]^{-1}\phi^T(k)P(k-1)\Big] \tag{1.36}$$

$$K(k) = P(k-1)\phi(k)[\lambda + \phi^T(k)P(k-1)\phi(k)]^{-1} \tag{1.37}$$

$$\theta(k) = \theta(k-1) + K(k)(d(k) - \phi^T(k)\theta(k-1)) \tag{1.38}$$

where $\boldsymbol{x}(k)$ is the real-valued input vector and $d(k)$ is the real-valued desired output scalar sequence. $\mathrm{p}(\mathrm{x}(k))$ is defined in (1.33), and λ is the forgetting factor. Some comments on the RLS based fuzzy adaptive equalizer are given in the next sub-section.

6.3 Comments on RLS based fuzzy adaptive equalizer

The RLS algorithm, (1.36) – (1.38) are obtained by minimizing the recursive least squares criterion, $J(k) = \sum_{i=0}^{k} \lambda^{k-i}[d(i) - f_k(\mathrm{x}(i))]^2$, with f_k constrained to be the form of (1.34). Because f_k is linear in the parameter, the derivation of (1.36) – (1.38) is the same as that for the FIR linear adaptive filter (Cowan, C.F.N., 1985).

Equations (1.36) – (1.38) can be viewed as updating the $\prod_{i=1}^{n} m_i$ rules in the form of (1.29) by changing the centers $\theta^{(j1,\dots,jn)}$ of the THEN parts of these rules in the direction that minimizing the criterion function $J(k)$ as in (1.28). Only these centers are allowed to change. The membership functions $\mu_{F_i^{ji}}$ of the IF parts of the rules are fixed at the very beginning and are not allowed to change. Hence, a good choice of the membership functions is very important to the success of the entire equalizer. However, in the next section, LMS based fuzzy adaptive equalizer will lighten this constraint by allowing $\mu_{F_i^{ji}}$'s also to change during the adaptation process.

It was proven in (Wang, L.X., 1992[a]) and (Wang, L.X., 1992[b]) that the equalizers (1.30) are universal approximators; i.e. for any real continuous function g on the compact set U, there exists a function in the form of (1.30) such that it can uniformly approximate g over U to an arbitrary accuracy. Therefore, this fuzzy adaptive equalizer is a powerful nonlinear adaptive equalizer in the sense that it has the capability of performing very difficult nonlinear equalization operation. This type of fuzzy adaptive equalizer performs two operations on the input vector $\boldsymbol{x}$: first, it performs a nonlinear transformation $\boldsymbol{p}(.)$ on $\boldsymbol{x}$; then the equalizer output is obtained as a linear combination of these transformed signals. The fuzzy adaptive equalizer is similar to the radial basis function (Chen, S., 1991), (Powell, M.J.D., 1987) and potential function (Meisel, W.S., 1969) approaches.

Linguistic information (in the form of fuzzy IF-THEN rules of (1.30)) and numerical information (in the form of desired input-output pairs $\boldsymbol{x}(k)$, $d(k)$) are combined into the equalizer in the following way: In Step 2-3, linguistic IF-THEN rules are directly incorporated into the equalizer (1.30) by constructing the initial equalizer based on the linguistic rules. During the parameter adaptation, numerical pairs ($\boldsymbol{x}(k)$, $d(k)$) are