Recent Development and Computational Solution for Fuzzy Filtering

Sukhvir Singh, Yogesh Mohan & Kishori Lal Bansal

Department of Computer Science, Himachal Pradesh University, Shimla (HP) - 171005
E-mail : sukhvir_thakur@rediffmail.com, yogeshmohan.edu@gmail.com, kishorilalbansal@yahoo.co.in

***Abstract* – This study present recent developments in the field of fuzzy modeling and also provide their detailed MATLAB software implementation. The objective of the study is to provide a general overview of fuzzy modelling and to show how computational intelligence based models contribute to the methodology of constructing models of software processes and products. This study also helps in software developments of some computational optimization methods for fuzzy modeling and data modeling algorithm using a stochastic fuzzy system in the presence of uncertainties. The developed MATLAB software can be used in the field of image processing, machine learning, pattern recognition, signal processing.**

Keywords* – *Fuzzy filtering, data modeling, uncertainties, clustering.

I. INTRODUCTION

The use of fuzzy systems in data driven modelling is a widely studied topic. A real-world modelling problem typically involves the uncertainties and the fuzzy systems based on fuzzy set theory [1], [2] are considered suitable tools for dealing with the uncertainties. This manuscript proposes a method of data modelling in the presence of uncertainty that performs better than the standard neuro/fuzzy modelling techniques. The main contribution of this text is to present a modelling algorithm and their matlab code providing an integrated framework to develop a data model based on different fuzzy filtering algorithms.

The paper is organized as follows. Section 2 presents some background of the method. Section 3 and section 4 suggest computational optimization algorithms and their matlab implementation. Section 5 deal with estimation of parameter of proposed fuzzy system and their matlab code.

II. PRELIMINARY

2.1 Fuzzy clustering

Clustering is main task of explorative data mining and a common technique for statistical data analysis, pattern recognition, image analysis, information retrieval and bioinformatics. Clustering involves the task of dividing data point into homogeneous classes or clusters so that items in the same class are as similar as possible and items in different classes are as dissimilar as possible. Depending on data and application, different type of similarity measure may be used to identify classes, where the similarity measure controls how the clusters are formed. In fuzzy clustering, which is a well recognized paradigm to generate the initial fuzzy model, each point has a degree of belonging to clusters, as in fuzzy logic. Thus points on the edge of a cluster may be in the cluster to a lesser degree then points in the centre of cluster. On the other hand in non fuzzy clustering, data point belongs exactly to one cluster, which is opposite to fuzzy clustering.

Many methods have been proposed in literature to determine the relevant number of clusters in a clustering problem. In fuzzy clustering, the objective is to partition the identification data in a specified number of clusters, no matter whether the clusters are meaningful or not. The validity of clusters must be evaluated separately, after the clustering take place. The number of clusters should ideally correspond to the number of sub-structure naturally present in the data. Numerous clustering algorithms have been developed. The most widely used is fuzzy C-means (FCM) clustering algorithm (Dunn, 1974; Bezdek, 1974; Bezdek et al., 1987), due to its efficiency and simplicity. The FCM algorithm partition a collection of n data points $(X = \{x_1, x_2, -----, x_n\})$ into c fuzzy clusters such that the following objective function is minimized:

$$J_m = \sum_{k=1}^{n}\sum_{i=1}^{c} \mu_{ik}^{m}(x)\left\|x_k - v_i\right\|^2, \quad 1<m<\infty,$$

Where v_i is the ith cluster centre, μ_{ik} is the degree of membership of the k^{th} data in the i^{th} cluster and m is a constant greater then one. $\mu_{ik} \in U$, U is a $c \times n$ fuzzy partition matrix which satisfies the constraints:

$$0 < \sum_{k=1}^{n} \mu_{ik} < n \quad \text{for } i = 1, 2, -----c,$$

$$\text{and } \sum_{i=1}^{c} \mu_{ik} = 1 \text{ for } k = 1, 2, -------n.$$

Cluster validity is the problem of finding the best value for c subject to minimization of J_m.

2.2 Robustness Issues in Fuzzy Identification

The fuzzy modelling is based on the assumption that there exist an ideal set of model parameters that model output to input through a mapping in an approximate manner. But, the identification of fuzzy models can sometimes be a difficult problem, often characterized by lack of data in some regions, collinearities and other data deficiencies, or a suboptimal choice of model structure. The part of the input-output mapping that cannot be modelled, for a given type and structure of the model, is refer to as uncertainty. Many real-world physical processes are also generally characterized by the presence of non-linearity, complexity and uncertainty. These processes cannot be represented by linear model used in conventional system identification [3]. One particular useful way to deal with uncertainties is to make use of flexible strategies, which if robust enough can handle different future conditions. Therefore, a robust identification of model parameters using available input-output data pairs is obviously a straightforward approach to handle uncertainty. Generally, robustness and interpretability are essential prerequisites for a model to be used. By model robustness means both generalization capability and numerical stability of the identified model. On the other hand interpretability concerns its transparency and intelligibility, as well as, the clear and sound physical meaning of the estimated parameters.

The identification problem for Sugeno fuzzy systems comes out to be linear in consequent parameters but non linear in antecedent parameters. For the tuning of linear in parameters, numerous adaptive estimation algorithm such as least-squares algorithms and gradient algorithms are available. Also, for non linear models, Regularization was suggested as a method for improving the robustness of fuzzy identification scheme leading to more accurate and well behaved fuzzy models in [4]-[6].

2.3 Stochastic fuzzy filtering

The study as in [16] define the problems of mixed fuzzy filter in which antecedents are deterministic where as consequents are stochastic. The stochastic fuzzy filtering extends the mixed stochastic / deterministic fuzzy filter of [16] to a purely stochastic fuzzy filter whose consequents as well as antecedents are random variable too. The parameters of mixed fuzzy filter were inferred under the Variational Bayes framework as in [16]. Most of the stochastic problems becomes high dimensional non-linear at last. The goal of such stochastic search algorithm is the convergence to global minima.

III. COMPUTATIONAL OPTIMIZATION

This study presents two optimization methods for solving non-linear problems. The first is Stationary Fuzzy Fokker Planck learning algorithm and second one is $L1$ optimization method. The software development of the presented algorithm is done in MATLAB 6.5.

3.1 Stationary Fuzzy Fokker-Planck Learning (SFFPL) [7]

SFFPL is a computational optimization technique to calculate minimum point of a continuous multivariate function. Generally, in the optimization process we set derivative equal to zero but sometimes derivative does not exist or it is too difficult to compute derivatives. In these types of problems we cannot apply finite difference methods. So, to deal with such type of problem there is a necessity of some derivative free method. The SFFPL is one such type of optimization method which gives idea how fuzzy modelling can be used for derivative free optimization process.

This method is based on the fuzzy approximation of stationary probability density of the stochastic search process. For the fuzzy approximation of stationary density function, C different fuzzy models $(m^1, ..., m^C)$ with $(K^1, ..., K^C)$ number of fuzzy rules, respectively, are considered. The number of fuzzy rules and uncertainty are chosen in such a way that

$$K^1 < K^2 < ..... < K^C$$

$$\Delta_j^1 > \Delta_j^2 > .... > \Delta_j^C.$$

The approach follows that by the use of m^1 for an objective function $U(x_1, x_2, ..., x_n)$, calculate a point x_i where U is minimum while keeping points other then x_i, fixed as an initial random guess. This linear one dimensional problem can be easily solved. In the next cycle the estimated minima serves as the initial guess for m^2 based evaluation. This process is continued till the estimation of global minima by the use of m^C. The detailed approach is presented in [7].

3.2 A Fuzzy Model with Triangular Membership Functions

This section introduces a fuzzy model with triangular membership function that maps an input variable to the output variable. The mathematical detail of this section has been taken from section II (A) of [7].

Consider a Zero-order Takagi-Sugeno Fuzzy Model with Uniformly Distributed Triangular Membership Functions mapping an input variable $x_i \in [a_i, b_i]$ to the output variable y_i through K different fuzzy rules:

R_1 : If x_i is A_1, then $y_i = S_1$

$\vdots \qquad\qquad \vdots$

R_K : If x_i is A_K, then $y_i = S_K$

Here, $(A_1, ..., A_K)$ are fuzzy sets and $(S_1, ..., S_K)$ are real scalars. To define the membership functions, consider K equally distanced points on the considered range of x_i :

$\{t^0 = a_i, t^1 = a_i + \delta_i, t^2 = a_i + 2\delta_i, ..., t^{K-2} = a_i + (K-2)\delta_i, t^{K-1} = b_i\}$, where

$$\delta_i = \frac{b_i - a_i}{K-1}.$$

Now, K triangular membership functions $(\mu_{A_1}, ...\mu_{A_K})$ can be defined as

$$\mu_{A_1}(x_i) = \max(0, \min(1, \frac{t^1 - x_i}{t^1 - t^0}))$$

$$\mu_{A_j}(x_i) = \max\left(0, \min\left(\frac{x_i - t^{j-2}}{t^{j-1} - t^{j-2}}, \frac{t^j - x_i}{t^j - t^{j-1}}\right)\right),$$

for all j=2,....,K-1

$$\mu_{A_K}(x_i) = \max\left(0, \min\left(\frac{x_i - t^{K-2}}{t^{K-1} - t^{K-2}}, 1\right)\right).$$

The output of fuzzy model is computed by taking the weighted average of the output provided by each rule. That is,

$$y_i = \frac{S_1\mu_{A_1}(x_i) + S_2\mu_{A_2}(x_i) + ... + S_K\mu_{A_K}(x_i)}{\mu_{A_1}(x_i) + \mu_{A_2}(x_i) + ... + \mu_{A_K}(x_i)}$$

$$= S_1\mu_{A_1}(x_i) + S_2\mu_{A_2}(x_i) + ... + S_K\mu_{A_K}(x_i).$$

For the defined triangular membership function, it is easy to observe that for any $x_i \in \left[a_i, b_i\right]$,

$$\frac{dy_i}{dx_i} = \frac{S_{z(x_i)+1} - S_{z(x_i)}}{t^{z(x_i)} - t^{z(x_i)-1}}$$

where $z(x_i) \in \{1, 2, ..., K-1\}$ is an integer such that

$$t^{z(x_i)-1} \le x_i \le t^{z(x_i)}.$$

Since $t^{z(x_i)} - t^{z(x_i)-1} = \delta_i$, we have

$$\frac{dy_i}{dx_i} = \frac{S_{z(x_i)+1} - S_{z(x_i)}}{\delta_i}.$$

3.3 Fuzzy Modeling of Stationary Cumulative Distribution

This section explains the concept of fuzzy approximation problem in which parameter of fuzzy models are estimated in such a way so that they approximate density function. Basically, the parameters of models are estimated by least square method. The mathematical detail of this section has been taken from section II (B) of [7].

Consider a set of $K-1$ points $(x_i^1, x_i^2, ..., x_i^{K-1})$ used for an identification of fuzzy model parameters $(S_2, ..., S_{K-1})$ in approximating the functional relation between x_i and $y(x_i \mid \{x_{j\neq i} = x_j^*\}))$.

An identification points x_i^k could be chosen randomly from a uniform distribution on (t^{k-1}, t^k), i.e.,

$$x_i^k \in (t^{k-1}, t^k), \text{ where } k \in \{1, 2, ..., K-1\}.$$

This results $\begin{bmatrix} S_2 \\ S_3 \\ \vdots \\ S_{K-1} \end{bmatrix} \in R^{(K-2)\times 1}$ in $\left(\frac{dy_i}{dx_i}\right)_{x_i = x_i^k} = \frac{S_{k+1} - S_k}{\delta_i}$.

The fuzzy approximation problem is to estimate the parameters $(S_2,....,S_{K-1})$, which satisfy $0 \le S_2 \le ..... \le S_{K-1} \le 1,$ such that

$$\left(\frac{dy_i}{dx_i}\right)_{x_i=x_i^k} \approx \frac{\exp(-U(x_i^k,\{x_{j\neq i}=x_j^*\})/T)}{\int_{a_i}^{b_i}\exp(-U(x_i,\{x_{j\neq i}=x_j^*\})/T)dx_i}, \text{ i.e.,}$$

$$S_{k+1}-S_k \approx \delta_i \frac{\exp(-U(x_i^k,\{x_{j\neq i}=x_j^*\})/T)}{\int_{a_i}^{b_i}\exp(-U(x_i,\{x_{j\neq i}=x_j^*\})/T)dx_i}.$$

The least squares estimation of fuzzy model parameters follows as:

$$\hat{S} = \arg \min_{S} [\|Y-BS\|^2, BS \ge h]$$

$$\text{Where } S = \begin{bmatrix} S_2 \\ S_3 \\ \vdots \\ S_{K-1} \end{bmatrix} \in R^{(K-2)\times 1}$$

$$B = \begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & -1 & 1 & \\ \vdots & \vdots & \vdots & \vdots \\ & & -1 & 1 \\ & & & -1 \end{bmatrix} \in R^{(K-1)\times(K-2)} \quad (1)$$

$$Y = \begin{bmatrix} \delta_i \frac{\exp(-U(x_i^1,\{x_{j\neq i}=x_j^*\})/T)}{\int_{a_i}^{b_i}\exp(-U(x_i,\{x_{j\neq i}=x_j^*\})/T)dx_i} \\ \vdots \\ \delta_i \frac{\exp(-U(x_i^{K-2},\{x_{j\neq i}=x_j^*\})/T)}{\int_{a_i}^{b_i}\exp(-U(x_i,\{x_{j\neq i}=x_j^*\})/T)dx_i} \\ \delta_i \frac{\exp(-U(x_i^{K-1},\{x_{j\neq i}=x_j^*\})/T)}{\int_{a_i}^{b_i}\exp(-U(x_i,\{x_{j\neq i}=x_j^*\})/T)dx_i} - 1 \end{bmatrix} \quad (2)$$

$$h = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ -1 \end{bmatrix} \in R^{(K-1)\times 1}. \quad (3)$$

3.4. Issue of Modeling Errors

Maximum or minimum point estimated for model generated density may contain error or uncertainty. These modelling errors arise because of the chosen triangular shape of membership function. If $\hat{S} = \left[\hat{S}_2\ \hat{S}_3.....\hat{S}_{K-1}\right]^T$ are the estimated parameters, then the problem to locate the point of maximum density can be mathematically expressed as

$$\arg \max_{x_i} \frac{\hat{S}_{z(x_i)+1}-\hat{S}_{z(x_i)}}{\delta_i}$$

Where $z(x_i) \in \{1,2,.....,K-1\}$ is an integer such that $t^{z(x_i)-1} < x_i < t^{z(x_i)}$.

The solution of this problem is nonunique. So, consider the center of the solution interval as follows:

$$x_i^p = \frac{t^{z^*-1}+t^{z^*}}{2}, \text{ where } z^* = \arg \max_{z} \frac{\hat{S}_{z+1}-\hat{S}_z}{\delta_i}.$$

Let Δ_i be a quantity that reflects an uncertainty in the maximum density point (x_i^p). One possible choice of Δ_i is as follows:

$$\Delta_i = \underbrace{\text{width of solution interval on either side of } x_i^p}_{=0.5\delta_i} + \underbrace{\text{uncertainty due to model prediction error}}_{=k_\Delta\delta_i}$$

Where k_Δ is a constant of proportionality. The term $(k_\Delta\delta_i)$ reflects the uncertainty because of a difference between the model-predicted density and the true density. Thus

$$\Delta_i = (0.5+k_\Delta)\delta_i.$$

The mathematical detail of this section has been taken from section II (C) of [7].

3.5 Stationary Fuzzy Fokker-Planck Learning Algorithm [7]

Algorithm considers the computation of the maximum density point x_i^p, which is based on the model parameters' values that are averaged over N^{mc} iterations. Algorithm requires to evaluate the objective function. The computational requirement of the algorithm can be controlled by choosing C different fuzzy model $(K^1 < K^2 < ..... < K^C)$, and $(N^1 > N^2 > ..... > N^C)$ appropriately. The choice

$N^1 > N^2 > ..... > N^C$ is justified as the smaller models have larger Δ_i values, and thus, the larger number of samples should be drawn to estimate the average density. The algorithm has been taken from section II (D) of [7].

Algorithm: *An algorithm for stationary fuzzy Fokker-Planck learning*

Require: objective function $(U(x_1, x_2, ..., x_n))$; $[a_i, b_i]$ *such that* $a_i \le x_i \le b_i$ for all i=1,2,.....,n .

1) *Choose* C *different fuzzy models* $(m^1, ..., m^C)$ *of the type described in section II (A) with* $(K^1, ..., K^C)$ *number of fuzzy rules respectively where* $K^1 < K^2 < ..... < K^C$.

   Let $(N^1, N^2, ..., N^C)$ *be the number of sample drawn from the distribution approximated by fuzzy models* $(m^1, ..., m^c)$ *respectively. Choose* $(N^1, N^2, ..., N^C)$ *in such a way that* $N^1 > N^2 > ..... > N^C$ *. Choose initial guess:*

   $x_i^p = 0.5(a_i + b_i), x_i^* = x_i^p, x_i^r = x_i^p$

   for all $i = 1, 2, ..., n$.

   Choose the constant k_Δ *e.g. equal to 1.*

2) *for model count:* $mc = 1$ to C *do*

3) *define* $\forall i = 1, 2, ..., n$, $\delta_i = (b_i - a_i)/(K^{mc} - 1)$ and $\Delta_i = (0.5 + k_\Delta)\delta_i$

4) *for sample count:* $sc = 1$ to N^{mc} do

5) *for* $i = 1$ *to* n *do*

6) *Define*

$$K = K^{mc}, \{t^0 = a_i, t^1 = a_i + \delta_i,\ t^2 = a_i + 2\delta_i, ..., t^{K-2} = a_i + (K-2)\delta_i, t^{K-1} = b_i\},$$

   and identification points $(x_i^1, x_i^2, ..., x_i^{K-1})$ *such that* x_i^k *, where* $k \in \{1, 2, ..., K-1\}$ *, is drawn randomly from a uniform distribution on* (t^{k-1}, t^k) .

7) *Set temperature equal to the variance of objective function values calculated at identification points, i.e.,*

$$T = \mathrm{var}\begin{pmatrix} U(x_i^1, \{x_{j\neq i} = x_j^*\}), U(x_i^2, \{x_{j\neq i} = x_j^*\}), ... \\ ..., U(x_i^{K-1}, \{x_{j\neq i} = x_j^*\}) \end{pmatrix}.$$

8) *The trapezoidal numerical integration method, with the points*

$$\begin{Bmatrix} U(x_i^1, \{x_{j\neq i} = x_j^*\}), U(x_i^2, \{x_{j\neq i} = x_j^*\}), ... \\ ..., U(x_i^{K-1}, \{x_{j\neq i} = x_j^*\}) \end{Bmatrix},$$

   can be used to approximate the integral

$$\int_{a_i}^{b_i} \exp(-U(x_i, \{x_{j\neq i} = x_j^*\})/T)dx_i .$$

9) *Solve the constrained linear least square problem* $\hat{S}|_{(i,sc)} = \arg\min_S [\|Y - BS\|^2, BS \ge h]$ *by transforming it first to a least distance programming [8]. Here, B,Y, and* h *are defined by (1-3).*

10) *For the convergence of estimated parameters, compute the average values:*

$$\hat{S}|_{(i)} = \left\langle \hat{S}|_{(i,sc)} \right\rangle_{sc} .$$

11) Compute a point of maximum density (x_i^p) as follows

$$x_i^p = \frac{t^{z^*-1} + t^{z^*}}{2},\ z^* = \arg\max_z \frac{\hat{S}_{z+1}|_{(i)} - \hat{S}_z|_{(i)}}{\delta_i} .$$

12) *Generate a point* (x_i^*) *from a uniform distribution on* $\left[x_i^p - \Delta_i, x_i^p + \Delta_i\right]$.

13) if $\left|x_i^p - x_i^r\right| > 0.5\delta_i$, then

14) *Since the model predicted density value remain same in an interval of length* $0.5\delta_i$ *on either side of* x_i^p , *we can take a chance to generate randomly a point* x_i^r *from a uniform distribution on* $\left[x_i^p - 0.5\delta_i, x_i^p + 0.5\delta_i\right]$ *and then to check in the next step if* x_i^r *is better than* x_i^p .

15) end if

16) *If* $U(x^r) < U(x^p)$, set $x_i^p = x_i^r$, *otherwise take a new chance to generate* x_i^r *for the next iteration.*

17) end for

18) end for

19) end for

20) return x^p .

Remark : *Algorithm 1 requires to evaluate the objective function following number of times:*

$$n(K^1+1)N^1 + n(K^2+1)N^2 + ... + n(K^C+1)N^C .$$

Software Implementation

Algorithms explained have been implemented in MATLAB 6.5. This section presents some pieces of software code developed during the computational optimization process of algorithms.

```
function [B] = AntecedentMatrix (inputs_data_matrix,FM)
[N,n] = size(inputs_data_matrix);
if FM.order == 0
   K = length(FM.consequents.mean);
elseif FM.order == 1
   K_f = length(FM.consequents.mean);
   K = K_f/(n+1);
   B_f = zeros(N,K_f);
else
   disp('The order must be either 0 or 1!');
   return
end
B = zeros(N,K);
k1 = 1;
while (k1 <= N)
   if strcmp(FM.memberships.type,'GaussianClustering')
  B(k1,:) =
GaussianMembershipClustering((inputs_data_matrix(k1
,:))',FM.memberships.cluster_matrix,FM.memberships.v
ar_matrix);
   else
      %%%%%%%%%%%%%%%%
      k2 = 1;
      while (k2 <= n)
         ts.membership_values{k2} =
MembershipValues(inputs_data_matrix(k1,k2),FM.mem
berships.knots{k2},FM.memberships.type);
         number_memberships(k2) =
length(ts.membership_values{k2});
         k2 = k2 +1;
      end
      %%%%%%%%%%%%%%%%
      if n==1,
         B(k1,:) = ts.membership_values{1};
      end
      if n == 2
         j = 0;
         for i1 = 1:number_memberships(1),
            for i2 = 1:number_memberships(2),
               B(k1,j+i2) =
(ts.membership_values{1}(i1))*(ts.membership_values{
2}(i2));
            end
            j = j+number_memberships(2);
         end
      end
      if n == 3
         j = 0;
         for i1 = 1:number_memberships(1),
            for i2 = 1:number_memberships(2),
               for i3 = 1:number_memberships(3),
                  B(k1,j+i3)
=(ts.membership_values{1}(i1))*(ts.membership_val
ues{2}(i2))*(ts.membership_values{3}(i3));
               end
               j = j+number_memberships(3);
            end
         end
      end
      if n == 4
         j = 0;
         for i1 = 1:number_memberships(1),
            for i2 = 1:number_memberships(2),
               for i3 = 1:number_memberships(3),
                  for i4 = 1:number_memberships(4),
                     B(k1,j+i4) =
(ts.membership_values{1}(i1))*(ts.membership_valu
es{2}(i2))*(ts.membership_values{3}(i3))*(ts.memb
ership_values{4}(i4));
                  end
                  j = j+number_memberships(4);
```

```
            end
        end
    end
  end
  %%%%%%%%%%%%%%%%
  B(k1,:) = (1/sum(B(k1,:)))*B(k1,:);
  %%%%%%%%%%%%%%%%
  if FM.order == 1
    for i = 1:K
      tmpvrb1 = n*(i-1) + i;
      tmpvrb2 = (n+1)*i;
      B_f(k1,tmpvrb1:tmpvrb2) =
(B(k1,i))*[inputs_data_matrix(k1,:) 1];
    end
  end
  k1 = k1+1;
end
if FM.order == 1
  B = B_f;
end
return

 function [mv] =
MembershipValues(x,t,membership_type)
switch lower(membership_type)
  case 'trapezoidal'
    [mv] = TrapezoidalMembership(x,t);
  case 'gaussian'
    [mv] = GaussianMembership(x,t);
  case 'clustering'
    [mv] = ClusteringMembership(x,t);
  case 'triangular'
    [mv] = TriangularMembership(x,t);
end
return

%%%%%%%%%%%%%%%%
function [p] = AntecedentsDensityFunction(theta,FM)
n = length(FM.memberships.knots);
p = [];
k = 1;
for i = 1:n,
  t = FM.memberships.knots{i};
  number_knots = length(t);
  p(k) = Trinagular(theta(k),t(1),t(1),0.5*(t(1)+t(2)));
  k = k+1;
  for j = 2:(number_knots-1),
    p(k) = Trinagular(theta(k),0.5*(t(j-
1)+t(j)),t(j),0.5*(t(j)+t(j+1)));
    k = k+1;
  end
  p(k) = Trinagular(theta(k),0.5*(t(number_knots-
1)+t(number_knots)),t(number_knots),t(number_knots))
;
  k = k+1;
end
return


function [y] = Trinagular(x,a,b,c)
if (x >= a) & (x <= b) & (b > a)
  y = 2*(x-a)/((c-a)*(b-a));
elseif (x >= b) & (x <= c) & (c > b)
  y = 2*(c-x)/((c-a)*(c-b));
else
  y = 0;
end
return
%%%%%%%%%%%%%%%%
function [mv] = ClusteringMembership(x,t)
n1 = length(t);
if x <= t(1),
  mv(1) = 1;
elseif x <= t(2),
  mv(1) = ((x-t(2))^2)/((x-t(2))^2+(x-t(1))^2);
else
  mv(1) = 0;
```

```
end
if x >= t(n1-1) & x <= t(n1),
   mv(n1) = ((x-t(n1-1))^2)/((x-t(n1-1))^2+(x-t(n1))^2);
elseif x > t(n1),
   mv(n1) = 1;
else mv(n1) = 0;
end
for i = 2:(n1-1),
   if x >= t(i-1) & x <= t(i)
      mv(i) = ((x-t(i-1))^2)/((x-t(i-1))^2+(x-t(i))^2);
   elseif x >= t(i) & x <= t(i+1)
      mv(i) = ((x-t(i+1))^2)/((x-t(i+1))^2+(x-t(i))^2);
   else mv(i) = 0;
   end
end
return
%%%%%%%%%%%%%%%%
function [c] = constraint(n)
for i =1:n+1,
   for j= 1:n,
      if j ==i
         c(i,j) = 1;
      elseif j ==(i-1)
         c(i,j)=-1;
      else
         c(i,j) =0;
      end
   end
end
%%%%%%%%%%%%%%%%
function [mv] = TrapezoidalMembership(x,t)
n1 = length(t);
a = t(1);
b = t(n1);
nmb = n1/2;
% calculate the membership values for given input
varable x
if x < t(1)
   mv(1) = 1;
elseif x <= t(2) & x >= t(1)
   mv(1) = 1;
elseif x <= t(3) & x >= t(2)
   mv(1) = (t(3)-x)/(t(3)-t(2));
else
   mv(1) = 0;
end
for j = 2:nmb-1,
   if x <= t(2*j-1) & x >= t(2*j-2)
      mv(j) = (x-t(2*j-2))/(t(2*j-1)-t(2*j-2));
   elseif x <= t(2*j) & x >= t(2*j-1)
      mv(j) = 1;
   elseif x <= t(2*j+1) & x >= t(2*j)
      mv(j) =(t(2*j+1)-x)/(t(2*j+1)-t(2*j));
   else
      mv(j) = 0;
   end
end
if x <= t(2*nmb-1) & x >= t(2*nmb-2)
   mv(nmb) = (x-t(2*nmb-2))/(t(2*nmb-1)-t(2*nmb-2));
elseif x <= t(2*nmb) & x >= t(2*nmb-1)
   mv(nmb) = 1;
elseif x > t(2*nmb)
   mv(nmb) = 1;
else
   mv(nmb)=0;
end
return
      %%%%%%%%%%%%%%%%
 function [bestpp,t1,t2,cov_val1,cov_val2] =
StochasticModelling(trndata,alph,cluster_matrix,show_f
ig)
 m = size(trndata,1);
 d = size(trndata,2)-1;
 out_trn = zeros(1,m);
```

```
 est_uncrtn = zeros(1,m);

 for j = 1:m,

 out_trn(j)   =mbership_clusters((trndata(j,1:d))', cluster_matrix)) *alph;

    est_uncrtn(j) = trndata(j,d+1)-out_trn(j);

end

 if show_fig == 1,

    disp('Performing Gaussian Mixture Modelling of the uncertainties...');

 end

y = [out_trn' est_uncrtn']';

[bestk,bestpp,bestmu,bestcov,dl,countf] = GaussianMixtureModelling_Modified(y,1,min(50,round(2.5*sqrt(m))),0,1e-4,0);

for j = 1:bestk,

    U(j,:) = multinorm([out_trn' est_uncrtn']',bestmu(:,j),bestcov(:,:,j));

end

color_code = rand(bestk,3);

maxU = max(U,[],1);

if show_fig == 1,

    h = figure;

end

for j = 1:bestk,

    index = find(U(j,:) == maxU);

    if show_fig == 1,

        figure(h); subplot(311), plot(y(1,index),y(2,index),'.','color',color_code(j,1:3)), xlabel('filtered output'), ylabel('estimated uncertainty'), hold on, elipsnorm_color(bestmu([1,2],j),bestcov([1,2],[1,2],j),2,1,color_code(j,1:3));

    end

end

if show_fig == 1,

    hold off;

end

t1 = bestmu(1,:);

t2 = bestmu(2,:);

for j = 1:bestk,

    cov_val1(j) = bestcov(1,1,j);

    cov_val2(j) = bestcov(2,2,j);

            end

s_out_trn = sort(out_trn);

s_est_uncrtn = sort(est_uncrtn);

for i = 1:m,

    for j = 1:bestk,

        s_mv1(i,j) = (bestpp(j))*((exp(-((s_out_trn(i)-t1(j))^2)/(2*cov_val1(j))))/(sqrt(2*pi*cov_val1(j))));

        s_mv2(i,j) = (bestpp(j))*(exp(-((s_est_uncrtn(i)-t2(j))^2)/(2*cov_val2(j)))/(sqrt(2*pi*cov_val2(j))));

    end

end

if show_fig == 1,

    figure(h); subplot(312), plot(s_out_trn,s_mv1), xlabel('filtered output'), ylabel('pdf'), hold on, plot(s_out_trn,sum(s_mv1,2),'k'), hold off;

    figure(h); subplot(313), plot(s_est_uncrtn,s_mv2), xlabel('uncertainty'), ylabel('pdf'), hold on, plot(s_est_uncrtn,sum(s_mv2,2),'k'), hold off;

end

return

function [R_square] = Calculate_R_Square(measured,predicted)

mean_meas = mean(measured);

mean_pred = mean(predicted);

for i = 1:length(measured),

    temp1(i) = (measured(i)-mean_meas)*(predicted(i)-mean_pred);

end

temp2 = measured-mean_meas;

temp3 = predicted-mean_pred;

temp4 = (sqrt(sum(temp2.^2)))*(sqrt(sum(temp3.^2)));

R_square = ((sum(temp1))/temp4)^2;

return
```

IV. A NEURAL NETWORK FOR CONSTRAINED *L*1 OPTIMIZATION [15]

Lemma 1 *the constrained* L_1 *estimation problem*

$$\min_x \ \left\{ \|Dx-d\|_1, Ax \le p \right\} \qquad (4)$$

can be solved by the following cooperative recurrent neural network model:

$$\frac{dx(t)}{dt} = \lambda \left\{ F_1(t) - D^T F_2(t) - A^T F_3(t) \right\} \qquad (5)$$

$$\frac{ds(t)}{dt} = \lambda \left\{ DF_1(t) + F_2(t) \right\} \qquad (6)$$

$$\frac{dz_{II}(t)}{dt} = \lambda \left\{ AF_1(t) + F_3(t) \right\} \qquad (7)$$

Where $\lambda > 0$ is a designing constant and

$$F_1(t) = -D^T s(t) - A^T z_{II}(t),$$

$$F_2(t) = gX_1(s(t) + Dx(t) - d) - s(t),$$

$$F_3(t) = gX_2(z_{II}(t) + Ax(t) - p) - z_{II}(t).$$

Here, $x(t), s(t), z_{II}(t)$ are state vectors of suitable dimensions and

$$gX_1(a) = \begin{cases} -1, & a < -1 \\ a, & -1 \le a \le 1 \\ 1, & a > 1 \end{cases}, \quad gX_2(a) = \begin{cases} a, & a \ge 0 \\ 0, & a < 0 \end{cases}$$

Proof : *The result has been directly taken from* [15].

Lemma 2 *The cooperative recurrent neural network (5-7) is stable in the sense of Lyapunov and is globally convergent to an optimal solution of the estimation problem (4) within a finite time.*

Proof: *The result has been directly taken from* [15].

***Matlab Implementation*:**

```
function [sol] =
L1_Estimation(D,d,B,b,A,p,lambda,xo,st,et)

options = simset('SrcWorkspace','current');

sim('L1_Neural',[st et],options);

load simout_file;

sol = ans(2:end,end)';

return
```

V. A FUZZY SYSTEM FOR DATA MODELING IN PRESENCE OF UNCERTAINTIES

A model with the following K fuzzy rules is considered:

R_1) If x belongs to a cluster having center c_1 then filtered output value $y_f = \alpha^1$,

lower limit on output value $y_L = \alpha_L^1$,

upper limit on output value $y_U = \alpha_U^1$,

for $x \in IG_i$, the output value

$$y = \frac{\sum_{j=1}^{C} a_j p(y_f \mid m_j^1, \Sigma_j^1) w_j^{i,1}}{\sum_{j=1}^{C} a_j p(y_f \mid m_j^1, \Sigma_j^1)};$$

$\vdots$

R_K) If x belongs to a cluster having center c_K then filtered output value $y_f = \alpha^K$,

lower limit on output value $y_L = \alpha_L^K$,

upper limit on output value $y_U = \alpha_U^K$,

for $x \in IG_i$, the output value

$$y = \frac{\sum_{j=1}^{C} a_j p(y_f \mid m_j^1, \Sigma_j^1) w_j^{i,K}}{\sum_{j=1}^{C} a_j p(y_f \mid m_j^1, \Sigma_j^1)}. \qquad (8)$$

Such a fuzzy model has been successfully applied to real-world problem in [9]-[11].

Introduce the notations:

$$\alpha_L = \left[\alpha_L^1 \alpha_L^2 ..... \alpha_L^K \right]^T, \alpha_U = \left[\alpha_U^1 \alpha_U^2 ..... \alpha_U^K \right]^T.$$

By aggregating these rules, we have $y_f = G^T(x,\theta)\alpha$, $y_L = G^T(x,\theta)\alpha_L$, $y_U = G^T(x,\theta)\alpha_U$,

$$y = G_1(x,\theta) \frac{\sum_{j=1}^{C} a_j p(y_f \mid m_j^1, \Sigma_j^1) w_j^{i,1}}{\sum_{j=1}^{C} a_j p(y_f \mid m_j^1, \Sigma_j^1)} + ...$$

$$.... + G_K(x,\theta) \frac{\sum_{j=1}^{C} a_j p(y_f \mid m_j^1, \Sigma_j^1) w_j^{i,K}}{\sum_{j=1}^{C} a_j p(y_f \mid m_j^1, \Sigma_j^1)}.$$

In other words,

$$y=\frac{\sum_{j=1}^{C} a_j\, p(y_f \mid m_j^1,\Sigma_j^1)G^T(x,\theta)w_j^i}{\sum_{j=1}^{C} a_j\, p(y_f \mid m_j^1,\Sigma_j^1)}. \qquad (9)$$

The motivation of considering such a mathematical expression for y in (8) is derived from the following fact:

Remarks 1: *The fuzzy rule based system (8) for any input x combine the outputs of C different local models $(M_1^i(x),.....,M_C^i(x))$ valid in the predefined operating regions represented by fuzzy sets $(A_1(y_f),.....,A_C(y_f))$ respectively based on the following fuzzy rule base:*

For input x, if the filtered value $y_f=G^T(x,\theta)\alpha$ is A_1, then $y=M_1^i(x)$;

$$\vdots$$

For input x, if the filtered value $y_f=G^T(x,\theta)\alpha$ is A_C, then $y=M_C^i(x)$. (10)

Here $M_j^i(x)$ and $A_j(y_f)$, for $j=1,.....,C$, are defined as

$$M_j^i(x)=G^T(x,\theta)w_j^i,\ A_j(y_f)=p(y_f \mid m_j^1,\Sigma_j^1). \qquad (11)$$

The value $a_j\in[0,1]$ is the weight of the rule that represents the belief in the accuracy of the j^{th} rule. To see the equivalent of (8) and (10), note that the weighted average of the output provided by each rule of (10) is equal to (9).

5.1 Estimation Of Fuzzy System Parameters

For a data driven construction of the fuzzy model (10), following parameters must be estimated using given input-output data pairs $\{x(j),y(j)\}_{j=0,1,.....,N.}$

5.2 Estimation of Filtering Parameters (α,θ)

Lemma 3 *A class of algorithms for estimating the parameters of Takagi-Sugeno type fuzzy filter recursively using input-output data pairs $\{x(j),y(j)\}_{j=0,1,.....}$ is given by the following recursions:*

$$\theta_j=\arg\min_{\theta}\left[\Psi_j(\theta)\right] \qquad (12)$$

$$\alpha_j=\alpha_{j-1}+\frac{P_jG(x(j),\theta_j)[y(j)-G^T(x(j),\theta_j)\alpha_{j-1}]}{1+G^T(x(j),\theta_j)P_jG(x(j),\theta_j)}, \qquad (13)$$

$$\Psi_j(\theta)=\frac{\left|y(j)-G^T(x(j),\theta)\alpha_{j-1}\right|^2}{1+G^T(x(j),\theta)P_jG(x(j),\theta)}+\mu_\theta^{-1}\left\|\theta-\theta_{j-1}\right\|^2 \qquad (14)$$

$$P_{j+1}=\left[P_j^{-1}+(1+\gamma)G(x(j),\theta_j)G^T(x(j),\theta_j)\right]^{-1}$$

For all $j=0,1,....$ with $\alpha_{-1}=0$, $P_0=\mu I$, and θ_{-1} is an initial guess about antecedents. Here, $\gamma\geq -1$ is a scalar whose different choices, as illustrated in table 1, solve the different filtering problems.

Table 1 The value of γ for different filtering criteria

H^∞-optimal like filtering criterion [12]	$\gamma=-1$
risk-averse like filtering criterion [12]	$-1\leq\gamma<0$
risk-seeking like filtering criterion [12]	$\gamma>0$

Proof: The result has been directly taken from [12].

The positive constant μ_θ in (14) is the learning rate for θ. The elements of vector θ (being the co-ordinates of clusters' centres in n-dimensional input space), if assumed as random variables, may have different variances depending upon the distribution functions of different inputs. Therefore, estimating the elements of θ with different learning rates make a sense. To do this, define a diagonal matrix Σ (with positive entries on its main diagonal):

$$\Sigma=\begin{bmatrix}\mu_{\theta(1)} & 0 & \cdots & 0\\ 0 & \mu_{\theta(2)} & \cdots & 0\\ \vdots & & \ddots & \vdots\\ 0 & \cdots & \cdots & \mu_{\theta(Kn)}\end{bmatrix}$$

To reformulate (14) as

$$\Psi_j(\theta)=\frac{\left|y(j)-G^T(x(j),\theta)\alpha_{j-1}\right|^2}{1+G^T(x(j),\theta)P_jG(x(j),\theta)}+\left\|\Sigma^{-1/2}(\theta-\theta_{j-1})\right\|^2 \qquad (15)$$

***5.2** An algorithm for fuzzy modeling of uncertain data*

Require: *Training data pairs $\{x(j),y(j)\}_{j=0,1,.....,N}$ (input and output variables have been normalized to have zero means and unity variances).*

1) *Choose the number of maximum epochs* $E_{\max}$. *A clustering on the input-output data of training set, via Gaussian mixture models of [13] with a common diagonal covariance for all clusters, can be performed to choose the*
   - *Number of clusters (i.e. rules) K ;*
   - *Initial guess about the clusters' centres* θ_{-1}*;*
   - *Learning rate* μ *for* α *equal to the variances of the clusters on output dimensions;*
   - *Learning rates* $(\mu_{\theta(1)}, \mu_{\theta(2)}, ..., \mu_{\theta(Kn)})$ *equal to the variances of different elements of* θ_{-1}.
2) *Identify fuzzy filter parameters* (α^I, θ^I) *using algorithm 2 of [14]*
3) *Compute* $\underline{\alpha}_L$ *and* $\bar{\alpha}_U$ *using Lemma 1*
4) *Compute* $\{z_j\}_{j=0,1,.....,N}$
5) ***for*** $i = 1$ *to* S ***do***
6) *for* $j = 1$ *to* C ***do***
7) *Obtain the data set* DP_j^i . *The parameters* ε *can be chosen in such a way that all data pairs for* $x \in IG_i$, *whose filtered output value* y_f *lies at* $\pm 3\sqrt{\sum_j^1}$ *from mean* m_j^1, *are included in* DP_j^i.
8) *Compute* $\underline{w}_j^i$ *using recursive least-squares (RLS) algorithm. Perform several epochs of the RLS algorithm till either the parameters converge or the maximum number of epochs* $E_{\max}$ *is reached.*
9) *end for*
10) *end for*
11) *return*

(α^I, θ^I); $(\underline{\alpha}_L, \bar{\alpha}_U)$;

$\{a_j, m_j^1, \sum_j^1\}_{j=1,2,.....,C}$; $\{\underline{w}_j^i\}_{i=1,2,.....,S;\, j=1,2,.....,C}$.

Software Implementation

Algorithms explained have been implemented in MATLAB 6.5. Algorithm was used for data modeling with the filtering criteria of Lemma 3. This section includes some pieces of software code developed during the computational optimization process of algorithms.

```
function [alph,cluster_matrix,mu,mu_theta_vector] =
FuzzyFilterIdentification(trndata,testdata,gama,max_it,s
how_fig,selection,p,a,b)

if show_fig == 1,

disp('**********************************************
************************');

disp('** This method of fuzzy modelling in presence of
uncertainties **');

disp('** has been developed by sukhvir singh and
Yogesh Mohan) ***************');

disp('**********************************************
************************');

   disp('Performing clustering in input space...');

   end

   m = size(trndata,1);

%k_max = min(50,round(2.5*sqrt(m)));

k_max = 100;

d = size(trndata,2)-1;

[bestk,bestpp_t,bestmu_t,bestcov,dl,countf] =
GaussianMixtureModelling_Modified((trndata(:,1:d+1))
',1,k_max,0,1e-4,3);

%--- remove the clusters with same centres in input
space -----

temp_matx = bestmu_t(1:d,:);

bestk =

   size((unique(temp_matx','rows')),1);

rmv_inx = [];

for i = 1:bestk,

var1 =
DistMatrix((temp_matx(:,i))',(temp_matx(:,(i+1):end))');

var2 = i+find(var1 == 0);

var3 = DistMatrix((temp_matx(:,i))',(bestmu_t(1:d,:))');

var4 = find(var3 == 0);

   var_ext = var4(1);

   var4(1) = [];

   rmv_inx = [rmv_inx var4];

temp_matx(:,var2) = [];

bestpp(i) = bestpp_t(var_ext)+sum(bestpp_t(var4));

var5(i) = mean([bestmu_t(d+1,var_ext)
bestmu_t(d+1,var4)]);
```

```
end
bestcov(:,:,rmv_inx) = [];
bestmu = [temp_matx;var5];
%-------------------------------------------------------------
if bestk == 1,
   if show_fig == 1,
      disp(sprintf('The number of clusters originally
choosen = %d out of %d,',bestk,k_max));
      disp('however two additional clusters lying far (+/-
3 sqrt(cov)) from mean have been added.');
   end
  bestk = 3;
  bestmu(:,2) = bestmu(:,1)+3*sqrt(diag(bestcov(:,:,1)));
bestmu(:,3) = bestmu(:,1)-3*sqrt(diag(bestcov(:,:,1)));
   bestcov(:,:,2) = bestcov(:,:,1);
   bestcov(:,:,3) = bestcov(:,:,1);
end
cluster_matrix = bestmu(1:d,:);
for i = 1:bestk,
   temp_v = diag(bestcov(:,:,i));
   in_indx = 1+((i-1)*d);
   mu_theta_vector(in_indx:i*d,1) = temp_v(1:d);
   temp_mu(i) = bestcov(d+1,d+1,i);
end
mu = mean(temp_mu);
%%% if we take initial guess a null vector, then
alph = zeros(bestk,1);
%%% otherwise
%alph = bestmu(d+1,:)';
if show_fig == 1,
   disp(sprintf('Total number of clusters choosen = %d
out of %d.',bestk,k_max));
end
clear bestk bestpp bestmu bestcov dl countf k_max;
if show_fig == 1,
   if nargin > 5,
      disp(strcat(sprintf('Identifying the parameters of the
fuzzy filter for p = %d ...',p),'for selection:', ' ',
selection));
   else
      disp(sprintf('Identifying the parameters of the fuzzy
filter for gamma = %d ...',gama));
   end
end
P = mu*eye(length(alph));
Tot_trndata = trndata;
for i = 1:(max_it-1),
   Tot_trndata = [Tot_trndata;trndata];
end
if show_fig == 1,
   h = waitbar(0,'Please wait...','Name','Identifying the
fuzzy filter parameters');
end
out_trn = zeros(1,m);
out_test = zeros(1,size(testdata,1));
            for k = 1:size(Tot_trndata,1),
   %i = round(1+(size(Tot_trndata,1)-1)*rand(1,1));
   i = k;
   if nargin > 5,
      [alph,cluster_matrix] =
DeterministicFuzzyLearningClustering(alph,Tot_trndata
(i,1:d),Tot_trndata(i,d+1),selection,cluster_matrix,mu_t
heta_vector,p,a,b);
   else
      [alph,cluster_matrix,P] =
ExponentialFuzzyLearningClustering(Tot_trndata(i,1:d)
,cluster_matrix,alph,Tot_trndata(i,d+1),P,mu_theta_vect
or,gama);
   end
   if rem(i,size(trndata,1)) == 0
      for j = 1:m,
         out_trn(j) =
(membership_clusters((trndata(j,1:d))',cluster_matrix))*
alph;
      end
      for j = 1:size(testdata,1),
```

```
        out_test(j) =
(membership_clusters((testdata(j,1:d))',cluster_matrix))*
alph;
      end
      R_square_trn =
Calculate_R_Square(trndata(:,d+1),out_trn');
      R_square_test =
Calculate_R_Square(testdata(:,d+1),out_test');
      mse_trn = sum((trndata(:,d+1)-out_trn').^2)/m;
      mse_test = sum((testdata(:,d+1)-
out_test').^2)/size(testdata,1);
      if show_fig == 1,
         disp(sprintf('%d. R_square_trn = %d,
R_square_test = %d, mse_trn = %d, mse_test =
%d',i/m,R_square_trn,R_square_test,mse_trn,mse_test))
;
      end
   end
   if show_fig == 1,
      waitbar(k/size(Tot_trndata,1),h);
   end
end
if show_fig == 1,
   close(h);
end
return
%%%%%%%%%%%%%%%%
This is file for updating fuzzy parameters. Please Note
that
% x = 1 x n
function [alph,cluster_matrix,P] =
ExponentialFuzzyLearningClustering(x,cluster_matrix,a
lph,y,P,mu_theta_vector,gama)
% convert cluster_matrix to theta_v
[n,no_clusters] = size(cluster_matrix);
for i = 1:no_clusters,
  theta_v((n*(i-1)+1):n*i) = (cluster_matrix(1:n,i))';
end
[theta_v] =
Deterministic3_clustering(x',theta_v,alph,y,P,mu_theta_
vector);
% convert again theta_v to cluster_matrix
for i = 1:no_clusters,
   cluster_matrix(1:n,i) = (theta_v((n*(i-1)+1):n*i))';
end
B = membership_clusters(x',cluster_matrix);
s2 = 1+(B*P*B');
s3 = y-B*alph;
alph = alph+(s3/s2)*P*B';
if gama == -1,
   P = P;
else
   gama_T = 1/(1+gama);
   P = P -((P*B'*B*P)/(gama_T+B*P*B'));
end
return
```

VI. CONCLUDING REMARKS

- A detailed presentation of recent developments in the field.
- Study and software development of some computational optimization algorithms for implementing
  1. A variation Bayesian fuzzy modelling algorithm, and
  2. A data modelling algorithm using a stochastic fuzzy system

VII. REFERENCES

[1] L.A. Zadeh.: Outline of a New Approach to the Analysis of Complex Systems and Decision Processes, IEEE Trans. on Systems, Man, and Cybernetics **3**, 28–44 (1973).

[2] L.A. Zadeh.: The role of fuzzy logic in the management of uncertainty in expert systems. Fuzzy Sets systems 11, 199-227, 1983.

[3] L. Ljung.: System Identification, Theory for the User. New Jersey: Prentice-Hall, 1987.

[4] T. Johansen.: "Robust identification of takagi-sugeno-kang models using regularization," in Proc. IEEE conf. Fuzzy Systems, New Orleans, USA, 1996, pp. 180-186.

[5] M. Burger, H. Engl, J. Haslinger, and U. Bodenhofer.: "Regularized data-driven construction of fuzzy controllers," J. Inverse and Ill-posed problems, vol. 10, pp. 319-344, 2002.

[6] M. Kumar, R. Stoll, and N. Stoll.: "Regularized adaptation of fuzzy inference systems. Modelling the opinion of a medical expert about physical fitness: An application," Fuzzy Optimization and Decision Making, vol. 2, pp. 317-336, Dec. 2003.

[7] Mohit Kumar, Norbert Stoll, Regina Stoll: "Stationary Fuzzy Fokker – Planck Learning and Stochastic Fuzzy Filtering". IEEE T. Fuzzy Systems 19(5): 873-889 (2011).

[8] C. L. Lawson and R. J. Hanson.: Solving Least Squares Problems. Philadelphia, PA: SIAM, 1995.

[9] M. Kumar, D. Arndt, S. Kreuzfeld, K. Thurow, N. Stoll, and R. Stoll.: "Fuzzy techniques for subjective workload score modelling under uncertainties," IEEE Transactions on Systems, Man, and Cybernetics-PartB: Cybernetics, vol. 38, no. 6, pp. 1449-1464, 2008.

[10] Kumar, M., Thurow, K., Stoll, N., Stoll, R.: A fuzzy system for modelling the structure-activity relationships in presence of uncertainties. In: Proc. IEEE International Conference on Automation Science and Engineering (CASE 2008), pp. 1025-1030. Washington DC, USA (2008)

[11] Kumar, M., Weippert, M., Kreuzfeld, S., Stoll, N., Stoll, R.: A fuzzy filtering based system for maximal oxygen uptake prediction using heart rate variability analysis. In: Proc. IEEE International Conference on Automation Science and Engineering (CASE 2009), pp. 604-608. Banglore, India (2009)

[12] Kumar, M., Stoll, N., Stoll, R.: On the estimation of parameters of takagi-sugeno fuzzy filters. IEEE Transaction on Fuzzy Systems 17(1), 150-166 (2009)

[13] Figueiredo, M.A.T., Jain, A.K.: Unsupervised Learning of Finite Mixture Models. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(3), 381-396 (2002)

[14] Kumar, M., Thurow, K., Stoll, N., Stoll, R.: Fuzzy filtering: A mathematical theory and application in life science. In: A.T. Azar (ed.) Fuzzy Systems, pp. 129-146. Intech, Olajnica, Croatia (2010)

[15] Xia, Y., Kamel, M.S.: A cooperative recurrent neural network for solving L_1 estimation problems with general linear constraints Neural Comput. 20(3), 844-872 (2008).

[16] M.Kumar, N. Stoll, and R. Stoll.: "Variational Bayes for a Mixed Stochastic/Determinstic Fuzzy Filter," IEEE Transactions on Fuzzy Systems, vol. 18, no.4, pp. 787-801, 2010.

◈◈◈