Journal of Maritime Research, Vol. III. No. 3, pp. 69–86, 2006

ISSN: 1697–4840

THE ANFIS BASED ROUTE PREFERENCE ESTIMATION IN SEA NAVIGATION

Natasa Kovac[1] and Sanja Bauk[1]

Abstract

This paper considers developing the ANFIS system based upon functional equivalence between the two-input first-order Takagi-Sugeno-Kang (TSK) fuzzy inference system and radial basis network (RBN). The input values are the linguistic qualifications related to the criteria: the wind strength and the traveling distance, in the long-ocean sailing zone, treated here since they are of upmost importance from the aspects of the traveling safety and efficiency. The output value is the degree of particular traveling direction preference. In the paper considered, hybrid neuro-fuzzy system, could be used as a theoretical support for developing the system with power of assistance in preferential sailing direction selection in long-ocean navigation.

Key words: adaptive neuro-fuzzy inference system (ANFIS), hybrid learning, Takagi-Sugeno-Kang (TSK) fuzzy inference system (FIS), radial basis network (RBN), preferential sailing direction selection.

INTRODUCTION

The input data in most of real world selection processes are not always precise. The same is in the case of preferential sailing direction selection in long-ocean navigation. Let's suppose that the mariner needs to navigate the vessel from one port to another one, and that he has a few different sailing directions from which he must choose one. His choice will depend of a large number of parameters, such as the wind strength, the total traveling distance, the traffic density, the fuel consumption,

[1] Faculty of Maritime Studies, University of Montenegro, Dobrota 36, 85330 Kotor, Serbia & Montenegro.

the demand for transport, etc. In aim to simplify this problem, since our work is attended to be rather theoretical then practical, we will be interested only in two criteria: the wind strength and the traveling distance. The parameters related to these two criteria will be in fact the input values for the hybrid neuro-fuzzy system, while the desired output is a numerical value that corresponds to the degree of considered sailing direction preference.

The first step is to define the decision selection rules based on inputs, using experts' knowledge for a certain sailing zone. Here are the two most important rules:

— If the wind strength and the traveling distance are small, than the decision selection will have high value, and
— If the wind strength and the traveling distance are high, than the decision selection will have small value.

Then, an ANFIS is to be designed and trained, in aim to simulate experts' knowledge in the observed sailing area. Finally, the adequate example with numerical and graphical results will be given.

THE MOTIVATION

The question is: where the motivation for the problem previously defined can actually be found? An explanation could be the following: now-a-day in order to navigate the vessel from one to another point, in the most efficient way, the mariner uses the actual digital technology of the electronic chart in the same manner as the traditional paper chart. The elements of route planning in electronic chart technology are waypoints and the line legs, which connect the waypoints. While waypoints may stand alone or may be connected to legs, legs are always bounded by waypoints. The leg lines can be constructed for a preplanned speed, while waypoints can carry information about the turning radius of the course change, etc. The leg lines and the waypoints are created in the electronic chart-planning mode. The alternate routes are connected to waypoints to form a network of leg lines that represent all the routes a vessel could use on her voyage. After the alternate routes are planned, the actual route, that is the route which the vessel will use, must be selected. The electronic chart supports this operation with its auto selection function. If the way through the possible routes net is unambiguous, this function will trace the preferential route automatically. But, if more than one leg departs from a waypoint (Figure 1), then electronic chart's auto selection function asks for the mariner's decision. That is why various decision support methods in preferential sailing direction selection are required

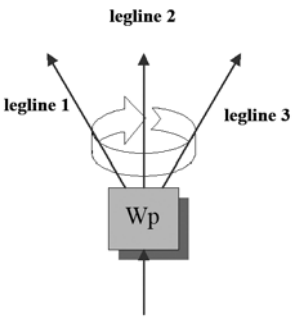

Figure 1. The case when more then one leg line departure a waypoint.

and are to be conceived, developed and implemented into up-to-date navigational equipment. This was in a way the motivation for developing the TSK and RBN based ANFIS devoted to the route, that is, its segments preference estimation (Bauk and Avramović, 2004; Bauk, 2003, 2005).

LINGUISTIC VARIABLES AND MEMBERSHIP FUNCTIONS

In the cases similar to the previously mentioned one, we often relay on qualitative features, since it is not convenient to consider numerical values in the system input segment. Namely, in such cases the input numerical data are not sufficiently precise. The main problem appears because of the different system's users estimations: someone will characterize the wind as a strength one with number 6 in Beaufort's scale, but someone else will characterize the same wind with number 7 in Beaufort's scale. In aim to resolve this problem and avoid numerical values at the system input part, the linguistics variables are to be introduced into the neuro-fuzzy modeling process.

Thus, the words or the sentences in natural or artificial languages are values of linguistic variables. According to the basic fuzzy rules mentioned in the section one, it is obvious that the inputs should be characterized with at least two linguistic constants: *small* and *high*, but we will incorporate *very small, medium and very high* as new categories in aim to increase the system precision. Each linguistics constant has its own membership function.

Let X be a set of objects, called the universe, whose elements are denoted as x. The membership in a subset A of X is the membership function μ_A from X to the real interval [0,1]. The universe consists of various elements which will be of concern in the particular context. A is called a fuzzy set and is a subset of X that has no sharp boundary. The function μ_A is the grade of membership x in A. The closer the value of μ_A is to 1, the more x belongs to A. The total allowable universe of values is called the domain of the fuzzy set. The domain is a set of real numbers, increasing monotonically from left to right where the values can be both positive and negative. A is completely characterized by the set of pairs $A=\{(x, \mu_A(x)), x \in X\}$. The degree of membership is known as the membership or truth function since it establishes a one-to-one correspondence between an element in the domain and a truth value indicating its degree of membership in the set. It takes the form $\mu_A(x) \leftarrow f(x \in A)$ (Fuller, 1999). There are few commonly used membership functions. In this work we prefer that each input has five membership functions (MF) represented by Gaussian functions given in the following form:

$$f(x) = e^{\frac{-(x-\mu)^2}{\sigma^2}} \tag{1}$$

where μ is the location parameter and represents a MF center and σ is the scale parameter which determines the MF width. The function (1) has maximum equal to one and minimum equal to zero.

FUZZY INFERENCE SYSTEM

The concept of a fuzzy set has been proposed by Zadeh (1965). Namely, the fuzzy sets were introduced as a generalization of conventional set theory. The fuzzy logic, based upon the fuzzy sets, implements human experiences and preferences via membership functions and fuzzy rules. The fuzzy sets are functions that map a value that might be a member of the set to a number between zero and one indicating its actual degree of membership. A degree of zero means that the value is not in the set, while a degree of one means that the value is completely representative of the set. This produces a curve across the members of the set. The fuzzy models manipulate linguistic variables. A linguistic variable is the representation of a fuzzy space which is essentially a fuzzy set derived from the evaluation of the linguistic variable. It encapsulates the properties of approximate or imprecise concepts in a systematic and computationally useful way. Although a linguistic variable may consist of many separate terms, it is considered as a single entity in the fuzzy proposition. The fuzzy if-then rules or the fuzzy conditional statements are expressions of the form IF *A* THEN *B* where *A* and *B* are labels of fuzzy sets characterized by appropriate membership functions (Zadeh, 1965). The if-then rules determine the system behavior. A knowledge base, decision making unit, fuzzification interface and a defuzzification interface construct a fuzzy inference system. The knowledge base comprises if-then rules and the appropriate MF associated to the linguistic constants. A decision-making unit performs the inference operations on the rules. The fuzzification interface determines the membership degree of crisp inputs to the linguistic constants, while defuzzification interface transforms the out coming fuzzy result into the crisp output. The basic structure of a fuzzy inference system is presented schematically in Figure 2.

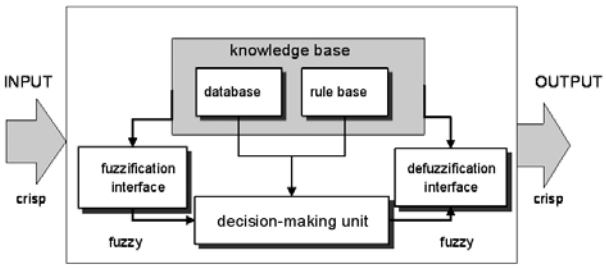

Figure 2. Fuzzy inference system.

The fuzzy inference system performs a fuzzy reasoning (inference operations upon fuzzy if-then rules) through next steps:

1. Comparing the input variables with the membership functions on the premise part to obtain the membership values of each linguistic label;

2. Combining through a specific T-norm the membership values on the premise part to get firing strength (weight) of each rule;
3. Generating the qualified consequent of each rule depending on the firing strength and
4. Aggregating the qualified consequents to produce a crisp output.

Several types of fuzzy reasoning have been proposed in the literature (Lee, 1990). A mathematical model which in some way uses fuzzy sets is called a fuzzy model. Here, we shall use Takagi–Sugeno model with incorporates fuzzy if-then rules where the output of each rule is a linear combination of input variables plus a constant term, and the final output is the weighted average of each rule's output (Takagi and Sugeno, 1983). The first order TSK fuzzy model is outlined in the Figure 3.

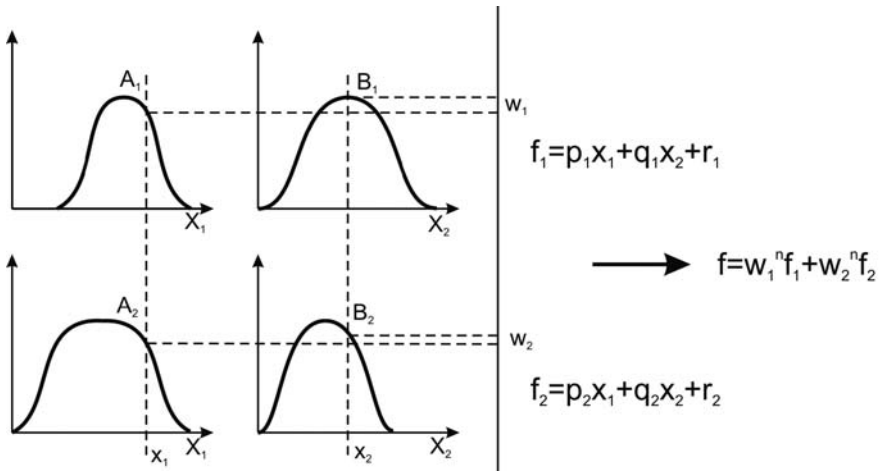

Figure 3. The Takagi-Sugeno-Kang (TSK) fuzzy inference system

THE ADAPTIVE NEURO-FUZZY INFERENCE SYSTEM - ANFIS

An adaptive network is a multi-layer, feed-forward network in which each node performs a particular node function based on incoming signals, as well as, on the set of parameters related to this node. Namely, each node function depends on the set of parameters which are associated to it. In the training process these parameters are being corrected in the aim of overall error minimization at the output of the network.

The ANFIS is the hybrid neural-fuzzy system. It has the capability of simulating the expert knowledge in the domain for which it has been constructed, on the basis of the expert knowledge represented by the if-then rules, and after the hybrid learning procedure has been realized. Thus, if an unknown input has been represent-

ed to the ANFIS, after the hybrid learning process, it can successfully predict the adequate output for the given input.

Now, let's consider our primer problem of route, or its part, preference estimation, and mark the wind strength with x_1, the traveling distance with x_2, and MFs attached to inputs with A_i and B_i respectively, where i=1,2,3,4,5. Our ANFIS is based on Takagi-Sugeno fuzzy inference system (Jang, 1993) and employs fuzzy rules of the next kind: if x_1 is A_i and x_2 is B_i then $f_1 = p_i x_1 + q_i x_2 + r_i$, where p, q and r is parameter set of function f. The number of these rules is 25. The ANFIS has five layers where all nodes from one layer have the same functions and its structure is represented in Figure 4.

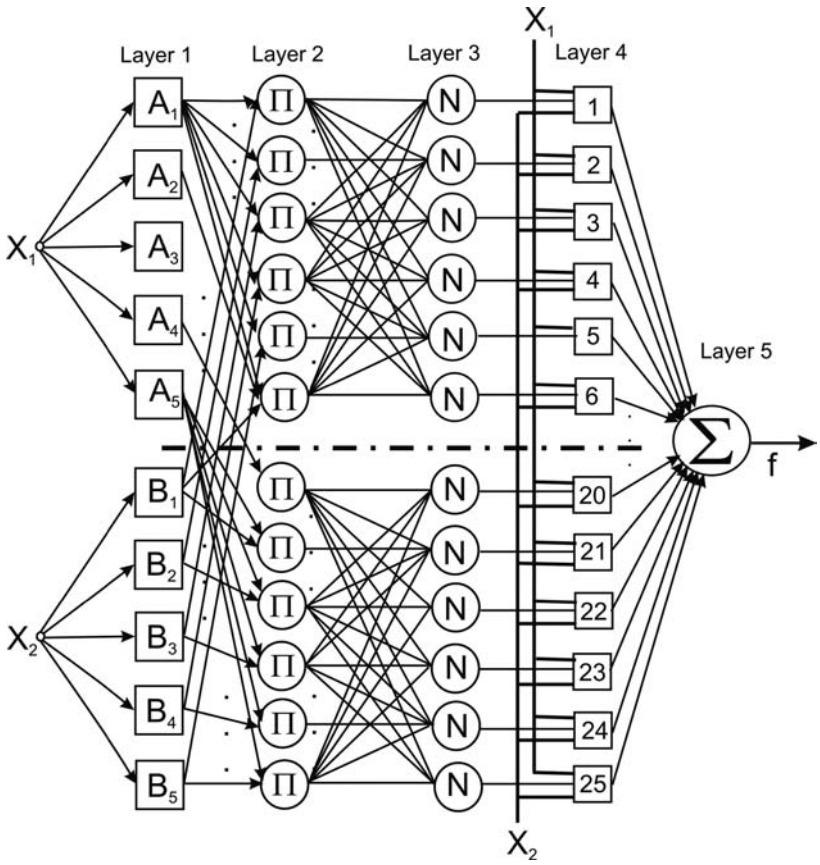

Figure 4. The ANFIS structure, here used as a navigation decision support tool

Each node represented with a square has a parameter (adaptive nodes), while nodes represented with a ring are fixed (fixed nodes). The parameters related to the adaptive nodes are presented by the set of training data in the learning process.

— Layer 1: The inputs to this layer are crisp x_1 values and x_2. The nodes are implemented as MFs and will determine the degree in which the inputs satisfy the MF. Each Gaussian function depends on its center and spread constant and these parameters are referred to as *premise parameters*.
— Layer 2: The number of the nodes is determined with the number of the fuzzy rules: with 2 inputs and 5 MFs, this layer consists of 25 nodes. The overall output of the nodes is the product of all incoming signals. Any T-norm operator can be used as fuzzy AND operator in this layer.
— Layer 3: The number of nodes in this layer is 25, too. Every node is a fix node and calculates the ratio of the *j*-th fuzzy rule firing strength (RFS) to the sum of RFS according to formula:

$$\overline{w_j} = \frac{w_j}{\sum_{i=1}^{25} w_i}, \text{ where } j = 1,2,...,25 \tag{2}$$

— Layer 4: Every node in this layer is adaptive node with node function:

$$\overline{w_i} f_i = \overline{w_i}(p_i x_1 + q_i x_2 + r_i), \text{ where } i = 1,2,...,25 \tag{3}$$

and p_i, q_i, r_i are parameters named *consequent parameters*.
— Layer 5: There is only one node that computes the output f of the ANFIS as the sum of incoming signals:

$$f = \sum_{i=1}^{25} \overline{w_i} f_i \tag{4}$$

The premise part of a rule defines a fuzzy region, while the consequent part determines the output within this region. The total number of these regions in case of here considered ANFIS is 25 (Figure 5).

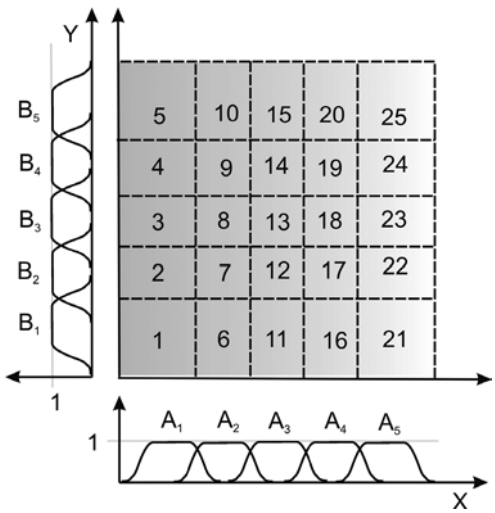

Figure 5. The partition of the input space into 25 fuzzy regions.

THE HYBRID LEARNING ALGORITHM

The gradient method can identify a parameter in an adaptive network, but its disadvantages are slowness and huge possibility to be troubled in local minimum. This is the main reason why the hybrid learning method, which combines gradient descent and least squares estimate (LSE) methods, has been exploited here.

The ANFIS uses back propagation to learn the premise parameters and least mean square estimation to determine the consequent parameters. In batch learning mode, which we have used in this paper, backward pass and premise parameters update, take place after the whole training data set has been presented to the ANFIS. Namely, premise parameters have been tuned after each epoch.

Generally, the hybrid learning algorithm in each epoch is developed through two phases: forward pass and backward pass, as it is summarized in the Table 1.

	Forward pass	**Backward pass**
Premise parameters	Fixed	Gradient descent
Consequent parameters	LSE	Fixed
Signals	Node output	Error rate

Table 1. The hybrid learning procedure

In the forward pass a set of training data formed from input values and desired corresponding output value is presented to the system. Layer by layer, every node output is calculated. According to described ANFIS structure, the overall output from the ANFIS can be represented as:

$$f = \sum_{i=1}^{25} \overline{w_i} f_i = \left(\overline{w_1} x_1\right) p_1 + \left(\overline{w_1} x_2\right) q_1 + \left(\overline{w_1}\right) r_1 + \ldots + \left(\overline{w_{25}} x_1\right) p_{25} + \left(\overline{w_{25}} x_2\right) q_{25} + \left(\overline{w_{25}}\right) r_{25} \tag{5}$$

This means that the output is a function of the input variables and the set of parameters. More formally, output can be rewritten as:

$$f = F(I, S) \tag{6}$$

where I is the vector of input variables and S is the set of parameters. This set of parameters can be decomposed in two subsets from the perspective of output: linear parameters and nonlinear parameters. From equation (5) it is obvious that the output is linear in the consequent parameters $\{p_i, q_i, r_i\}$ $i = 1,2,\ldots,25$, so we can apply least squares method to identify these parameters. Also, after presenting a set of training data to the system, the ANFIS output can be described as matrix equation,

$A\theta = B$ where θ is an unknown vector whose elements are consequent parameters. The best values for these parameters can be calculated by the least-squares estimator $\theta^* = (A^T A)^{-1} A^T B$ where A^T is the transpose matrix of A. This method can be used if $A^T A$ is nonsingular matrix.

In the backward pass, all consequent parameters are fixed, since their best values are identified in the forward pass, and the error is calculated for each training pair. Let's assume that there are P training data pairs and that the ANFIS has L layers. With $N(L)$ we denoted the number of nodes in the layer L. The node in the i-th position of the k-th layer, as well as this node output, is symbolized with O_i^k. For the p-th training data, where $1 \le p \le P$, we can define the error, as a sum of the squared errors:

$$E_p = \sum_{m=1}^{N(L)} \left(T_{m,p} - O_{m,p}^L\right)^2 \tag{7}$$

where $T_{m,p}$ is the m-th component of p-th target output vector, and $O_{m,p}^L$ is the m-th component of actual output vector obtained by the presentation of the p-th input vector to the network. The overall error measure is:

$$E = \sum_{p=1}^{P} E_p \tag{8}$$

and needs to be minimized. Now, the error rate $\frac{\partial E_p}{\partial O}$ for p-th training data and for all node output O has to be calculated. According to the formula (7) the error rate for the output node in the i-th position is

$$\frac{\partial E_p}{\partial O_{i,p}^L} = -2\left(T_{i,p} - O_{i,p}^L\right) \tag{9}$$

while the error rate for the node in the i-th position of the k-th layer, where $1 \le k \le L-1$, can be derived by the chain rule:

$$\frac{\partial E_p}{\partial O_{i,p}^k} = \sum_{m=1}^{N(k+1)} \frac{\partial E_p}{\partial O_{m,p}^{k+1}} \frac{\partial O_{m,p}^{k+1}}{\partial O_{i,p}^k} \tag{10}$$

The equation (10) means that the error rate for the node from k-th layer can be calculated as a linear combination of the error rates of the nodes in the $(k+1)$-th

layer. Consequently, the formula (9) needs to be applied first. After that, backward layer by layer, the formula (10) can be applied, until all error rates are calculated. The error signals propagate from the output to the input end, so this learning paradigm is called the *back-propagation* (Rumelhart *et al,* 1986). We need to apply the chain rule again, to find the *gradient vector*. Gradient vector is the derivative of the error measure with respect to each parameter. If q is a parameter of the network, we have:

$$\frac{\partial E}{\partial q} = \sum_{p=1}^{P} \frac{\partial E_p}{\partial q} = \sum_{p=1}^{P} \sum_{O^* \in S} \frac{\partial E_p}{\partial O^*} \frac{\partial O^*}{\partial q} \tag{11}$$

where S is the set of nodes containing q as a parameter. The update formula for the generic parameter q is:

$$\Delta q = -\eta \frac{\partial E}{\partial q} \tag{12}$$

in which η is a learning rate which can be expressed as:

$$\eta = \frac{k}{\sqrt{\sum_{q} \left(\frac{\partial E}{\partial q} \right)^2}} \tag{13}$$

where k is the step size. In the batch learning, the update formula for parameter q is (11), since update action occurs after each epoch, i. e. when all training data are presented to the network. At the end of the backward pass all the premise (nonlinear) parameters are updated by the gradient descent.

The hybrid learning algorithm pseudo – code

This subsection contains the applied pseudo-code for the hybrid learning method and its technical explanation in some more details.

The values necessary for creating the ANFIS are defined in the main program. The procedures ***Set*** set the value of the variable being denoted as the first argument to the numerical value representing the second argument of the procedure. Here are given the concrete numerical values being used in the paper. The first one ***for statement*** determines the initial values for MF, that is its centers and σ values. This has been achieved in a way that for each input – Gaussian functions are uniformly distributed in the given range of inputs. After entering the data being used

for the network training, the set of rules is to be formed and the procedure of training can start through *Learning* procedure. In aim to test in this manner trained network, the appropriate testing procedure is to be realized by the procedure *Testing*.

```
Set (NumberOfInputs,2);
Set (NumberOfOutputs,1);
Set (NumberOfMF,5);
For input := 1 to NumberOfInputs do
     Begin
     Read(minValue[input], maxValue[input]);
     EvenlyAllocateMFs(maxValue[input]-minValue[input], NumberOfMF);
     For k:= 1 to NumberOfMF do
         Remember(Center[input,k], Sigma[input,k])
     end;
Read(TrainingData);
Determine(TrainingDataNumber);
Calculate(NumberOfRules);
CreateRules;
Set (EpochNumber,100);
Learning;
Testing;
```

The procedure *Learning*, as we shall see later, calls the procedure *UpdateStepSize* to realize possible correction of the variable *StepSize* values. The correction of this variable is to be done after each fourth epoch, counting from the epoch in which the previous correction has been done. It is realized on the base of the rules listed below:

1. If the error undergoes four reductions, then increase *StepSize* by 10%. This condition is checked with the function *Increase*.
2. If the error goes through combination of increase and decrease, then decrease *StepSize* by 10%. This condition is confirmed with the function *Decrease*.

The variable *Epoch* contains the index of the current epoch, while the variable *lastChanged* stores the index of the epoch in which the variable *StepSize* has been changed previously.

```
procedure UpdateStepSize(var lastChanged : integer; epoch : integer;
                     var StepSize : integer;)
begin
     if Decrease(lastChanged,epoch) then
        DecreaseStepSize(StepSize, 10);
     if Increase(lastChanged, epoch) then
        IncreaseStepSize(StepSize, 10);
     lastChanged := epoch;
end;
```

The pseudo-code for the key procedure *Learning* containing all the theoretical statements from the section 6, follows:

```
procedure Learning;
     var epoch, lastChanged, i, j, k, ErrorMeasure : integer;
        y, grad : real;
begin
     set MinError to huge value;
     lastChanged := 1;
     for epoch := 1 to EpochNumber do
        begin
           ErrorMeasure := 0;
           for j := 1 to TrainingDataNumber do
           begin
              for k := 1 to NumberOfInputs do
                  ReadInputValue(TrainingData[j]);
                  DesiredOutput[j]:= exstract last value from
                      TrainingData[j];
              { forward pass for all layers }
              for i := 1 to NumberOfNodes do
                      Calculate Node[i] output based on InputValues;
              LSE;
              { calculate error measure }
              ErrorMeasure := ErrorMeasure +
                   sqr(DesiredOutput[j] - ObtainedOutput[j]);
              OutputNode.de_do := -2*(DesiredOutput[j]-
                                   obtainedOutput[j]);

              Perform backward calculation for all inner layers;

              y := exp(-sqr((x-c)/sigma)/2);
              for i := FirstNodeFromLayer1 to LastNodeFromLayer1 do
                    for j := 1 to 2 do
                       begin
                         do_dp[1] :=
                            y*sqr(x-c)/(sqr(sigma)*sigma);
                         do_dp[2] := y*(x-c)/sqr(sigma);
                         node[i].de_dp[j] := node[i].de_dp[j] +
                            node[i].de_do*do_dp[j];
                       end;
           end;
           Calculate (Error[epoch]);
           if Error[epoch] < MinError then
                 begin
                        MinError = Error[epoch];
                        remember all current parameters as best
                                                          parameters;
                 end;
           { update parameters: sigma and center }
           grad := 0;
           for i := FirstNodeFromLayer1 to LastNodeFromLayer1 do
                 for j : = 1 to 2 do
```

```
                grad := grad + sqr(node[i].de_dp[j]);
        grad := sqrt(grad);
        for i := FirstNodeFromLayer1 to LastNodeFromLayer1 do
              for j : = 1 to 2 do
                  node[i].para[j] := node[i].para[j] -
                      StepSize*node[i].de_dp[j]/grad;
        UpdateStepSize(lastChanged,epoch,StepSize);
     end
end;
```

The ending of this procedure, means that the network has been trained.

The last phase in ANFIS creation process is its testing. This procedure has been realized upon the set of test data, by the procedure ***Testing***, and after the testing, we have clear picture of the network efficiency.

```
procedure Testing;
var   i :  integer;
     TestingError : real;
begin
     Read(TestingData);
     Determine(TestingDataNumber);
     for i:= 1 to TestingDataNumber do
        Determine(inputs[i], DesiredOutput[i]);
     for i:= 1 to TestingDataNumber do
        begin
              ObtainedOutput[i] := CalculateOutput(inputs[i]);
              Error[i]:= DesiredOutput[i]-ObtainedOutput[i];
        end;
     TestingError := 0;
     for i := 1 to TestingDataNumber do
        TestingError := TestingError + Error[i];
     TestingError := TestingError/ TestingDataNumber
end;
```

This is the summarized presentation of the original code in Pascal being applied for the appropriated calculations in the paper, but it contains all cardinal elements of the proposed methodology for creating, training and testing the ANFIS based upon TSK and RFN functional equivalence. The obtained results, which will be presented within the next section, confirm its applicability in solving the real problem, as well as, its validity.

THE SIMULATION RESULTS

According to the problem formulation in section 2, we have to determine the preference of the route in long-ocean navigation, or even the preference of its seg-

ment, upon two criteria: the wind strength and the traveling distance in the sailing zone. Only two criteria have been considered in aim to simplify the problem, though the very similar procedure based on TSK and RFN equivalence could be applied to more complex problems of this kind with three, four or more criteria relevant for the degree of the route preference estimation.

Let's us suppose that we have the elementary fuzzy rules, formed on the base of expert knowledge related to the wind strength and the traffic density (which usually implies traveling distance between departure and arrival port, or between two waypoints) for the observed sailing zone (Table 2).

Rule	**Traveling distance**		**Wind strength**		**Preference**
No.	Linguistics value (fuzzy)	Num. value, e. g. (crisp)	Linguistics value (fuzzy)	Num. value, e. g. (crisp)	Value (crisp)
1	very small	500	very small	0.20	1.00
2	very small	500	small	2.70	0.75
3	very small	750	medium	4.00	0.50
4	very small	550	high	6.00	0.25
5	very small	500	very high	7.60	0.00
6	small	1550	very small	0.00	0.75
7	small	1630	small	2.75	0.75
8	small	1600	medium	4.75	0.50
9	small	1630	high	5.75	0.25
10	small	1500	very high	8.00	0.00
11	medium	2750	very small	0.60	0.75
12	medium	2100	small	2.00	0.75
13	medium	2700	medium	4.50	0.50
14	medium	2500	high	6.00	0.25
15	medium	2750	very high	7.80	0.00
16	high	4100	very small	0.00	0.25
17	high	4250	small	2.00	0.25
18	high	4000	medium	4.00	0.25
19	high	4100	high	6.10	0.25
20	high	3600	very high	7.90	0.00
21	very high	4900	very small	0.10	0.25
22	very high	4600	small	1.80	0.25
23	very high	5000	medium	4.75	0.00
24	very high	4900	high	5.50	0.00
25	very high	4900	very high	7.60	0.00

Table 2. The basic fuzzy rules for the sailing zone.

The exact numerical values in Table 2 (columns: 3rd and 5th) are given as some "crisp samples" for linguistics values *small*, *medium*, *high*, etc. These are the basic or

"rough" fuzzy rules used for defining training and testing sets. Namely, the example being considered here is based upon the MFs of the wind strength and the traveling distance presented in Figures 6 and 7. The wind strength in the sailing area is usually between numbers 0 and 8 of Beaufort scale, while traveling distance is a number from the interval between 500 Nm (nautical mile) and 5000 Nm.

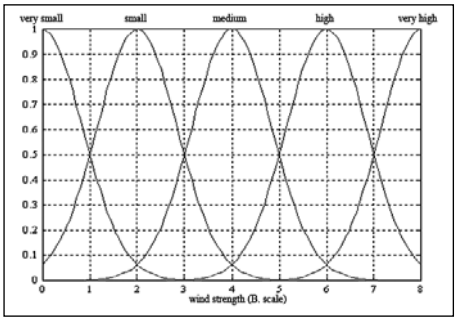

Figure 6. The MF of the wind strength.

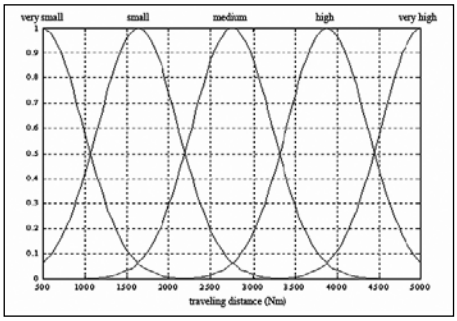

Figure 7. The MF of the traveling distance.

The values of the MFs centers and spread constants can be easily read from the presentation given above. The shapes of the wind strength MF before learning procedure are given in Figure 6, while the case after parameter regularization produced with hybrid learning algorithm (section 5) is presented below, in Figure 8. It is obvious that the centers of MFs have been shifted in a way, i.e. they have been tuned properly in aim to reduce errors between obtained and target outputs (the degrees of the route preference). Thus, this MFs tuning implies the least level of the output error in the model.

It is to be summarized that the total number of nodes in the ANFIS is 75. Though, the applied ANFIS model has 25 fuzzy rules, 25 linear and 20 nonlinear parameters. The 436 data pairs have been used in its training process. The average error after training process was $E = 0.038793$. The training error minimization, which is a result of the described hybrid learning completed after 100 epochs, is presented in the Figure 9. Obviously, the error curve decreases permanently and approaches to its minimal obtained value.

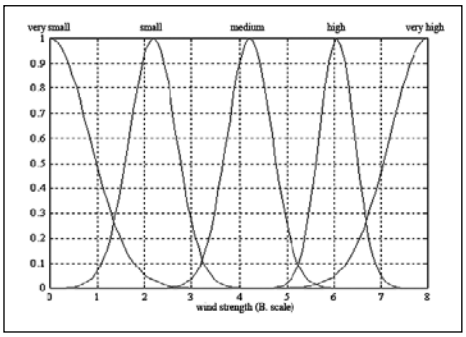

Figure 8. The wind strength MFs' tuning (the MFs after learning).

The route preference surface produced on the base of the 25 fuzzy rules previously defined and the corre-

sponding outputs within corresponding fuzzy regions are presented graphically in 3D space in Figure 10.

Within the ANFIS creation procedure, the testing phase follows after the training process. We used 50 data pairs as testing data to calculate the ANFIS output.

The proposed ANFIS efficiency has been presented in Figure 11. Here, 50 training data pairs have been used and the system output has been shown with "*", while the expected value has been represented with "·". (Figure 11). The average testing error was 0.014781. It is clear that in the most cases they are at the same position or next to each other. This confirms validity of the proposed ANFIS and the applied hybrid learning procedure.

Some of the illustrational samples, chosen from these 50 tested cases, are given in Table 3.

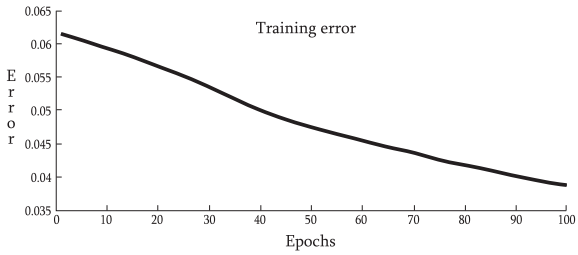

Figure 9. The error curve for the ANFIS.

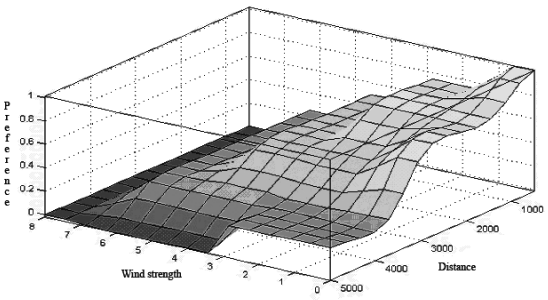

Figure 10. The output regions and the corresponding output values.

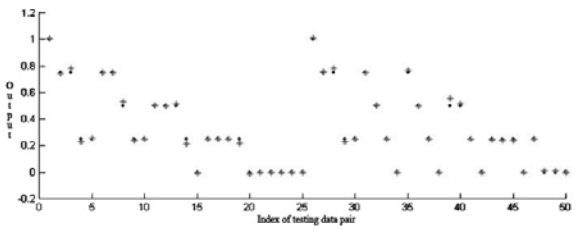

Figure 11. The desired outputs judging against the obtained ones.

On the base of the numerical data given in Table 3, it is obvious that the error is of E-02 or E-03 order, which is satisfying accuracy for a decision support method, like this of route preference degree estimation in long-ocean navigation.

How this model can be practically used in navigation? It could be used in the following manner: mariner gives the inputs to the computer, i.e. its own estimations for the wind strength and the traveling distance, and according to that how do they "fit" into the expert knowledge base – the degree of the route preference (or its segment) will be obtained automatically by the computer. This could be, in a way, a great help in ship maneuvering, particularly when larger number of criteria is to be considered and when it is not so easy to estimate a certain route (or its segment)

preference. Surely, this is still more theoretical then practical approach to the optimal long-ocean route selection, but it can be, undoubtedly, used when the appropriate knowledge base for some sailing zones become available.

Inputs		**Outputs**		**Error**
Wind strength [B. scale]	Traveling distance [Nm]	Desired output	Obtained output	
1.00	555	1.00	0.9919	0.0080968
0.30	510	1.00	0.9863	0.0136330
0.10	1500	0.75	0.7573	0.0073257
0.90	1700	0.75	0.7582	0.0082239
4.00	1728	0.50	0.4988	0.0012556
3.90	2500	0.50	0.5075	0.0075517
5.80	1500	0.25	0.2501	0.0001391
3.00	4128	0.25	0.2485	0.0014879
4.00	4921	0.00	0.0076	0.0076976
7.50	4800	0.00	0.0083	0.0083899

Table 3. The calculated data for some sample testing data.

REFERENCES

Bauk S., Avramović Z. (2004): Navigational decisions support method based upon tsk and rbfn function equivalence. *4th ICMTIR*, Barcelona, 255-264.

Bauk S. (2003): Fazi viekrišterijumsko odlučivanje u izboru preferentnog pravca plovljenja. *SYM-OP-IS*, Serbia and Montenegro, 30(1), 707-710.

Bauk S. (2005): Intelligent information systems in the ship route optimization. *PhD thesis*, University of Belgrade.

Fuller R. (1999): *Advances in soft computing – Introduction to neuro-fuzzy systems.* Physica-Verlang, A Springer Verlang Company.

Jang J.-S. R. (1993): ANFIS: Adaptive network based fuzzy inference system. *IEEE Transactions on systems man and cybernetics*, 23(3), 665-685.

Lee C.-C. (1990): Fuzzy logic in control systems: fuzzy logic controller - part 1. *IEEE Transactions on systems, man and cybernetics*, 20(2), 404-418.

Lee C.-C. (1990): Fuzzy logic in control systems: fuzzy logic controller - part 2. *IEEE Transactions on systems, man, and cybernetics*, 20(2), 419-435.

Rumelhart D. E., Hinton G. E., Williams R. J. (1986): *Learning internal representation by error propagation. Parallel distributed processing: Explorations in the microstructure of cognition*, Vol. 1 Eds. Cambridge, MA: MIT Press, 318-362.

Takagi T., Sugeno M. (1983): Derivation of fuzzy control rules from human operator's control actions. *IFAC Symposium on fuzzy information knowledge representation and decision analysis*, 55-60.

Zadeh L. A. (1965): Fuzzy sets. *Information and Control*, New York: Academic Press, **8**, 338-353.