Proceedings of the 2006 IEEE International Conference on Robotics and Automation
Orlando, Florida - May 2006

Unmanned Aerial Vehicle (UAV) Modelling Based on Supervised Neural Networks

R. San Martin, A. Barrientos, P.Gutierrez, J. del Cerro
Dpto. Ing. Sistemas y Automática (DISAM), E.T.S.I.I.
Universidad Politécnica de Madrid (U.P.M.)
José Gutierrez Abascal 2, 28006 Madrid, Spain
rsan, barrientos,pgutierrez,jcerro@etsii.upm.es

***Abstract*- This paper proposes the utilization of hybrid models of supervised neural networks for the modelling of dynamic systems. Particularly, as an example of a system, a autonomous helicopter or UAV is identified in both attitude and position systems. The evaluation of the model is done by comparing the radial basis and multilayer perceptron with the real system.**

Index Terms – Supervised Neural Networks, Modelling, UAV.

I. INTRODUCTION

An Unmanned Aerial Vehicle (UAV) is: A powered, aerial vehicle that does not carry a human operator, uses aerodynamic forces to provide vehicle lift, can fly autonomously or be piloted remotely, can be expendable or recoverable [1].

This implies serious issues both in the model identification and its controlling.

In this particular work, the UAV used is based on the BENZIN trainer Radio Controlled helicopter from Vario, with a 5Kg payload, a 26cc engine and a on board embedded system developed in our department. Basically, the embedded system consists of: A Mircroinfinity A3350M IMU and a Honeywell HMR3000 compass for the attitude data; Novatel RT2 GPS for the position data and a OctagonPC/770 PC-104 featuring a 1GHz PentiumIII with Linux RT that coordinates the action of the peripherals.

The helicopter's flight is described by its attitude as well as its position and considering that the inputs and outputs are coupled, the system can be qualified as dynamic, multivariable and unstable.

The modelling can be performed in several ways, with mathematical and empirical models, as well as hybrid models (mathematical-empirical), each one having advantages and disadvantages.

For a helicopter's model the mathematical description is very complex [2], and this requires a large computing power. Any hybrid model simplifies the analysis, however, it reduces its accuracy in relation to the real behaviour.

This work considers neural networks as an alternative to empirical models. The main advantages of a neural network are the low computing power required and the ability to store the model once the network has been trained. Besides, it does not need any previous knowledge since the network learns.

The identification process is critical for the controlling that provides the autonomy of a UAV. Control models are first tested using simulators and in case the identification is accurate, they will have a better performance under actual conditions. Besides, if used as a simulator the model can be an excellent output predictor, in case these can not be collected by the embedded system due to any kind of error or transmission delay.

The following sections deal with the next topics: The section II discusses some existing architectures for neural network used for the identification of a dynamic model. The section III describes the algorithms generated for the selected neural architectures for our particular system. Finally, the results comparison between a multilayer perceptron and a radial basis neural network are analyzed and the conclusion presented.

II. DYNAMIC MODELS' ARCHITECTURES

A. Supervised Neural Network

This work focuses in supervised neural networks, which are those networks that need training patterns for their learning process in order to adjust their parameters, so that the networks' outputs become as similar as possible to the training patterns. Once the network has been adjusted according to the training patterns the network is ready to be used and the parameters will no longer be modified. Therefore, it is very important to select the training patterns carefully, using the most representative patterns for the system's states [3].

Among supervised neural networks there is another classification that depends on their ability to analyze previous system status and divides them into non-recurrent and recurrent networks.

Non recurrent networks: are networks whose inputs propagate to obtain an output, without been fed back to the inputs.

Consider an input vector (Fig.1) $\mathbf{X = [x_1\ x_2\ ...\ x_n]}$, and the weights $\mathbf{W_{igm}}$, where **i** = number of layer, **g** = number of neurons and **m** = number of neurons of the previous layer, $\mathbf{F_1}$ and $\mathbf{F_2}$ are transfer functions represented as blocks and a bias vector $\boldsymbol{\theta}_\mathbf{i} = [\boldsymbol{\theta}_{\mathbf{i1}}\ \boldsymbol{\theta}_{\mathbf{i2}}\ ...\ \boldsymbol{\theta}_{\mathbf{ig}}]$. The equations are (1) and (2)

$$a_{1i} = F_1\left(\sum_{k=1}^{n} x_k \cdot w_{1ik} + \theta_{1i}\right) \tag{1}$$

$$y_i = a_{2i} = F_2\left(\sum_{k=1}^{g} a_{1k} \cdot w_{2ik} + \theta_{2i}\right) \tag{2}$$

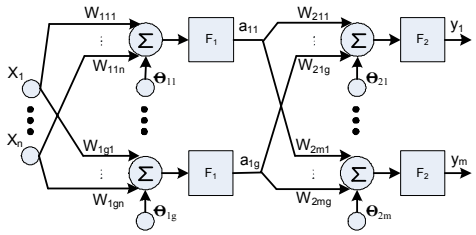

Fig.1. Non recurrent neural network
One input layer, one hidden layer and one output layer

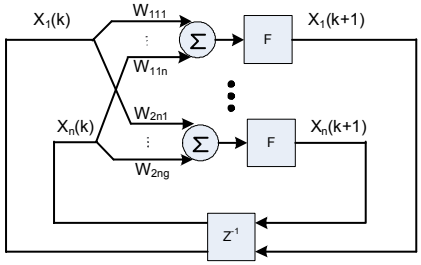

Fig.2. Hopfield memory a recurrent network

The output obtained in (2) is compared to the desired output pattern, using a minimum squared error algorithm.

During the training process, the error is used to correct both the weights and the bias.

A radial basis neural network [4] [5] is considered a particular case of a multilayer perceptron (MLP). Figure 1 shows a general diagram of a non-recurrent neural network.

Recurrent networks: are those which have a memory that works like a state machine and the outputs act as another input.

This networks are described in a discrete space, for an initial input vector $\mathbf{X_0 = X(0)}$ which originates a state loop. In many cases another external input might be considered to produce state changes once the system has achieved stability. Equation (3) shows the recurrent form, with an external input vector ***I***. Figure 2 shows a Hopfield memory [6], which is an example of a recurrent network.

$$X_j(k+1) = F_j\left(\sum_{k=1}^{n} w_{jk} X_k(k) + I_j\right) \qquad (3)$$

Clearly, a recurrent network is a memory device that tends to have the value predetermined by its weights.

B. System identification and neural network

System Identification is a critical problem in systems' theory. Basically it consists of: a system with an input space ***X***, an output space ***Y*** and an operator **H** that transforms the space ***X*** into ***Y***, where **H** belongs to an operators' class **C**. In case the operator **H** cannot be obtained, due to its complexity or to any other reasons, the system identification is used, which consists of finding an $\mathbf{H' \in C \ \therefore \ H'}(X) = Y'$, trying to make $Y' \approx Y$. The difference between ***Y*** and ***Y'*** is called error (4), obviously the smaller the error the better the approximation, in this manner a threshold is defined and all approximation with errors smaller than this threshold '***e***' are considered as valid.

The type of modelling depends on the actual system's characteristics [7]. In this work systems are qualified as static or dynamic.

$$\left\|Y' - Y\right\| = \left\|H'(X) - H(X)\right\| \le e \qquad (4)$$

Considering the system identifications by means of state variables, a dynamic system can be described in a discrete space as in (5). Where ***v*** is the state variable, ***x*** the input, and ***y*** the output.

$$\begin{aligned} v(k+1) &= \Phi[v(k), x(k)] \\ y(k) &= \Psi[v(k), x(k)] \end{aligned} \qquad (5)$$

For the identification of dynamic systems it is necessary to consider not only the system's inputs but also the previous system's states. Therefore, when determining the most convenient neural network architecture for a dynamic system, it is clear that a recurrent architecture cannot be selected, nor a purely non recurrent one. This is due to the fact that a non recurrent network is able to perform an excellent output tracking, but only considering the control inputs, without taking into account the previous states. On the other hand, a recurrent network is able to perform changes according to the previous states, but it does not consider tracking any of its outputs.

The solution is a complementary network, that features the characteristics of both variations (recurrent and non recurrent) in a single system.

There are several defined networks, but the ones known as Jordan [8] and Elman [9] stand out among the others [10].

C. Hybrid Network

A hybrid network is defined through modules that may be modified depending on the flexibility assigned to the block. Figure 3 shows **block A**, being a non-recurrent network, and **block B**, as a recurrent network or a set of contextual neurons.

The external inputs are represented by vector $\mathbf{X=[x_1 \ldots x_n]}$, generating the contextual neurons. The outputs are defined by vector $\mathbf{Y=[y_1 \ldots y_m]}$ which also generates contextual neurons.

Contextual neurons (6) define previous states to be memorized. In the case of the inputs, the d previous samples are used, and the contextual neurons vector for the i-th input takes the form $\mathbf{C_{xi}=[x_i(k-1) \ldots x_i(k-d)]}$. For the outputs the h previous samples are used, yielding a contextual neurons vector $\mathbf{C_{yi}=[y_i(k-1) \ldots y_i(k-h)]}$ for the i-th output. Thus, the input vector for the non recurrent network in block A is $\mathbf{U=[X \ C_{x1} \ldots C_{xn} \ C_{y1} \ C_{ym}]}$.

As mentioned before, **block B** corresponds to the recurrent stage and the contextual neurons, where the previous states are memorized and it is defined together with the network's architecture. In the case of a system with a large inertia, the order of the delays will increase. At the same time, **block A** corresponds to the non recurrent stage, which performs the

system output signals tracking and is able to operate internally both with a MLP [3] or radial basis networks [5].

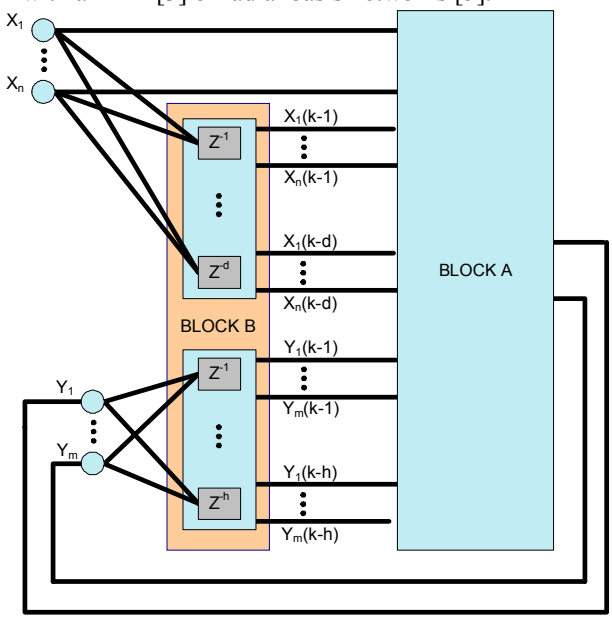

Fig.3. Hybrid Network

The hybrid network can be converted to the form (7), where functions **F** and **G** will depend on the architecture selected for **block A**, either radial basis network or MLP, as well as the internal transfer functions of these recurrent networks.

$$\begin{aligned} &Cout_i^r(k) = Y_i(k-r)\ \forall i = 1,...,n; r = 1,...,d \\ &Cin_j^q(k) = X_i(k-q)\ \forall\ j = 1,...,m; q = 1,...,h \\ &C(k) = [Cin_1^1(k)...Cin_n^d(k)\ Cout_1^1(k)...Cout_m^h(k)] \end{aligned} \tag{6}$$

Rewriting (6) in the propagation of a multilayer network (2) a form similar to the one necessary in (5) is obtained, where the contextual neurons will react depending on the previous external input and the previous state of the contextual neurons, that have defined the output (7).

$$\begin{aligned} C(k+1) &= F[C(k), X(k)] \\ Y(k) &= G[C(k), X(k)] \end{aligned} \tag{7}$$

The following section describes the hybrid network used for the UAV system and justifies the form that defines it.

III. Hybrid Network Arqchitecture For The Identification of a UAV

For the identification of a system, like the helicopter described previously, there are some adjustments that need to be performed on the hybrid network, and the simulation strategy has to be planned.

The flight of a helicopter [2] is based on the orientation and velocity of its blades. First it defines an attitude which results in a change in the position of the aircraft. Therefore, two simulation stages are considered: the first one consists of simulating the attitude based on the commands sent to the helicopter and the second one consists of simulating the position based on the attitude and the previous states.

In regards to the neuronal system that has to be trained, since it is a supervised network, two different training methods can be adopted. In the first one, both systems are connected in cascade and the output of the attitude system's training is used for the training of the position system. In the second one, a parallel training of both systems is carried out, but this architecture does not consider the error propagated from the output of the first network to the input of the next one. For this reason the first method is used.

A. Training Architectures

The data obtained from the AHRS, Attitude and Heading Reference System, (attitude: angles ***roll***, ***pitch***, ***yaw***) from the GPS (position: ***X***, ***Y***, ***Z***) and from the radio transmitter (control: ***Croll*** and ***Cpitch***, controlling the slate of the main rotor's plane; ***Cyaw***, controlling the tail rotor's revolutions and the ***Ccole*** which is a mixture of the slate of the blades and the rotor's throttle) are the patterns used for the training.

The natural operation of a helicopter's flight system is depicted in Figure 4, with its position being defined by its attitude.

The first system, that is, the attitude system, is trained as a single and isolated system, which is possible thanks to the previous knowledge of the real attitude data (***roll, pitch*** and ***yaw***), for the next system, the values obtained from the simulated attitude network (***roll', pitch'*** and ***yaw'***) are used for training the position network.

The training pattern for **block A** is vector **P** (8), with an external input vector **X** consisting of the different commands (***Croll, Cpitch, Cyaw*** and ***Ccolle***), another vector name $\mathbf{C_{in}}$, that corresponds to the input contextual neurons and finally $\mathbf{C_{out}}$, which represents the attitude system's feedback loop contextual neurons. The training pattern is represented by vector **T** (9), which is the real data provided by the AHRS.

Once the patterns are obtained, the training of **block A** starts by comparing the desired system output **T** (9) with the current network output **Y** (10). All necessary adjustments are performed with the difference, according to the training rules of a recurrent network. Basically, if the network is correctly trained, a minimum error is expected in comparison to the desired output **T**.

$$\begin{aligned} &P = [X, C_{in}, C_{out}] \\ &X = [Croll(t), Cpitch(t), Cyaw(t), Cole(t)] \\ &C_{in} = [C_{Croll}, C_{Cpitch}, C_{Cyaw}, C_{Colectivo}] \\ &C_{out} = [C_{roll}, C_{pitch}, C_{yaw}] \\ &C_i = [i(t-1),...,i(t-d)]\ \forall i = Croll, Cpitch, Cyaw, Cole \\ &C_j = [j(t-1),...,j(t-h)]\ \forall j = roll, pitch, yaw \end{aligned} \tag{8}$$

$$T = [roll(t), pitch(t), yaw(t)] \tag{9}$$

$$Y = [roll', pitch', yaw'] \tag{10}$$

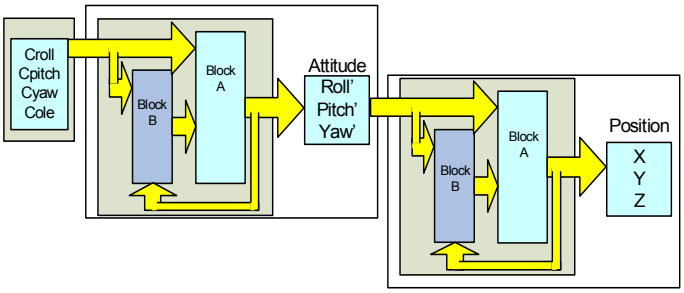

Fig.4. Cascade Training Architecture

This **Y** vector (10) obtained from simulating the first network defines the input pattern for the position system. This pattern will be very similar to the one used in the previous network, the main difference being the input vector **X** (11), which does not have actual values but instead the simulated values from the previous network. The output pattern for the training is vector **T** (12), which shows the position obtained from the GPS.

$$
\begin{aligned}
&P = [X, C_{in}, C_{out}] \\
&X = [roll'(t), pitch'(t), yaw'(t)] \\
&C_{in} = [C_{roll'}, C_{pitch'}, C_{yaw'}] \\
&C_{out} = [C_X, C_Y, C_Z] \\
&C_i = [i(t-1), \ldots, i(t-d)] \;\forall\; i = roll', pitch', yaw' \\
&C_j = [j(t-1), \ldots, j(t-h)] \;\forall\; j = x, y, z
\end{aligned}
\tag{11}
$$

$$T = [x(t), y(t), z(t)] \tag{12}$$

Once the system is trained, the simulation works in series, defining the position from the attitude, that is why it is advantageous to work with a cascade architecture. The needs of using the output of on a system as the input for the next one suggests the possibility of an eventual error propagation from the attitude network to the position network. This training method takes this error into account and corrects the attitude system error in the position system.

When comparing **T** (12) and **Y** (13) for correcting the recurrent network parameters, the error produced at the attitude output is considered in the input parameter **X** (12).

$$Y = [x'(t), y'(t), z'(t)] \tag{13}$$

B. Hybrid network Architecture

The hybrid network has the possibility of exchanging **block A**, between a MLP or a radial basis network. Elman contextual neurons are used for **block B**. Jordan contextual neurons are discarded, since they consider an accumulation of the network's input value. This affects the model identification with inputs that can be different, but the accumulation in time may produce a similarity not present in the actual system.

This system has large inertia and dynamics. Therefore, in order to define the order of the contextual neurons for both the input commands and the outputs, a correlation between the command signals and the attitude, as well as attitude and position was carried out, obtaining the largest delay value with regards to the current output. In general, the delay obtained lies between the 5th and the 9th sample, at a sample rate of 100ms. For this reason, and considering the worst case, a delay of 10 samples is adopted for the output contextual neurons $\mathbf{C_{out}}$ and a delay of 5 samples for the input contextual neurons $\mathbf{C_{in}}$, both for the attitude and the position system. This decision may seem somewhat arbitrary at a first glance, but it is indeed validated by tests that define this numbers as the best candidates.

This decision affects the training patterns and its conformation, as well as the amount of neurons in the only hidden layer of the MLP, which is defined also based on the amount of inputs. According to [11], a single layer is enough for modelling any system with a single MLP.

The inputs to **block A** of the attitude system inputs are divided into: 4 direct inputs from the radio transmitter **X** (8), which generate vector $\mathbf{C_{in}}$ (8) with an order **d** = 5, producing a total of 20 contextual neurons, the outputs **Y** (10), which have a delay **h** = 10 and generate a contextual neurons vector $\mathbf{C_{out}}$ (8) with 30 entries. These yield a total input vector **P** (8) to **block A** with 54 input neurons. This number is kept fixed both for the radial basis networks and the MLP.

For the position system, the inputs for **block A** are: 3 direct inputs, corresponding to the attitude outputs **X**; these inputs generate the contextual neurons $\mathbf{C_{in}}$, of order **d** = 5 and yielding a total of 15 elements; the contextual neurons for the outputs defining vector $\mathbf{C_{out}}$ has an order **h** = 10, generating 30 entries and yielding a total vector **P** to **block A** with 48 inputs.

The minimum error threshold used for the trainings was 10-5 in 40000 epochs for the MLP and the same error for the radial basis networks, although the epochs are limited to the amount of neurons present in its hidden layer. Therefore, this amount will change depending on the case.

The algorithm developed for the training and the simulation was created in Matlab. Although Matlab provides a large amount of tools [12], not all of them could be used due to the dynamic characteristics of the system. For this reason new algorithms have been developed.

C. Training Pattern Generation

Whether an empirical system identification, as this work, is successful or not depends on a good experimental method and, considering the neural networks as the basis of the identification, the experimental method becomes even more important.

Besides the theoretical problems in the system, practical issues have to be considered, for example, flights under windy conditions, GPS quality of signal loss, hardware damage, system vibrations and many others. The only solution for these issues is performing many flights and periodical maintenance on the helicopter in order to guarantee a representative sample.

After the data acquisition process, certain criteria for the selection of the obtained data are used. It has been mentioned before the importance of the data base that will generate the network's training pattern.

Two main criteria were established: First, in terms of the signal quality either due to the environmental conditions or the state of the equipment. This will be defined as quality criteria. On the other hand, the form criteria will consist of the rules that define the type of flight performed and its use

Quality Criteria: the idea is to separate all samples in suitable and unsuitable, and also, suitable samples may be useful for certain tasks, for example, a sample may be suitable for simulation, or training or observation, depending on its quality and significance

Form criteria: these criteria define the experiment or type o flight. Flights are not always planned for obtaining trainin samples, for example, in many cases it is necessary to carr out flights in order to verify trainings or test the systen operation.

D. Pattern transformation

In order to achieve an efficient system identification b means of a neural network, it is recommendable to manipulat the data obtained and in order to get a better performance.

Normalization: In general, the transfer functions of a neura network operate in a range of [0 1] or [-1 1]. Therefore, th pattern data has to be normalized. The standardization tes mentioned among the form criteria is used to obtain th maximum and minimum values.

Equations (14) and (15) normalize each entry X_k of vecto $\mathbf{X=[X_1 \ldots X_k \ldots X_n]}$ using the maximum and minimum values **X** (min and max) in ranges [-1 1] and [0 1] respectively.

$$norm = \frac{2 \cdot \left(X_k - \min(X)\right)}{\left(\max(X) - \min(X)\right)} - 1 \qquad (14)$$

$$norm = \frac{\left(X_k - \min(X)\right)}{\left(\max(X) - \min(X)\right)} \qquad (15)$$

Periodicity: The training of a network is considered a discrete time process. Therefore, it is necessary that the samples are obtained periodically (timing quality factor). Due to several reasons, the samples may be affected by delays. However, in order to ensure a proper operation of the system it is important to perform the sampling as periodically as possible. Otherwise, to satisfy the periodicity criteria, interpolation and extrapolation might be applied.

Yaw reference: The helicopter's attitude, while standing on the ground on a horizontal surface, has to be very close to zero. Since the yaw is referred to the north, it is not necessary that each flight starts with a yaw value of zero. The only consideration to be taken into account is to use the initial yaw angle as the reference. This simplifies the network training by ignoring the initial state of the yaw angle.

IV. TESTS AND RESULTS

The simulation will be conducted with a set of training patterns consisting of 9 flight sessions of approximately 3 minutes each, and complying with the required criteria.

The radial basis network and the MLP are compared with the same flight stage. Figure 5 shows the attitude simulations for both networks vs. the real attitude. Figure 6 shows the position simulations in takeoff for each network vs. real position.

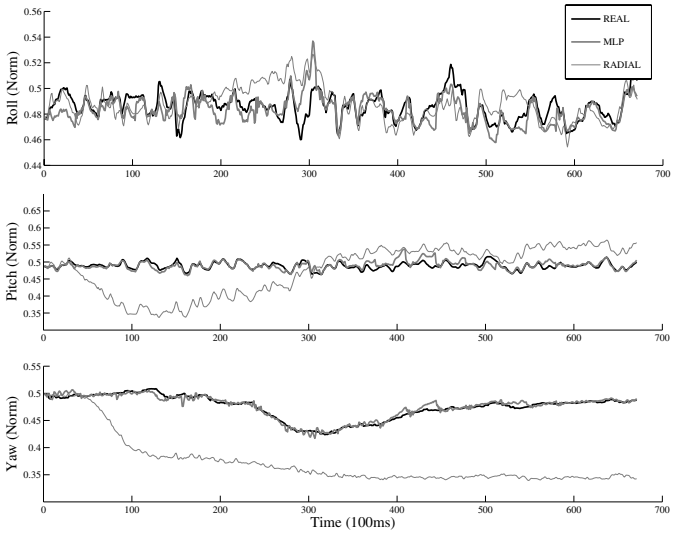

Fig.5. Real flight in attitude vs. simulated flight in attitude with MLP (60 neurons in hidden layer) and Radial Basis (500 radial neurons and 0.08 SPREAD)

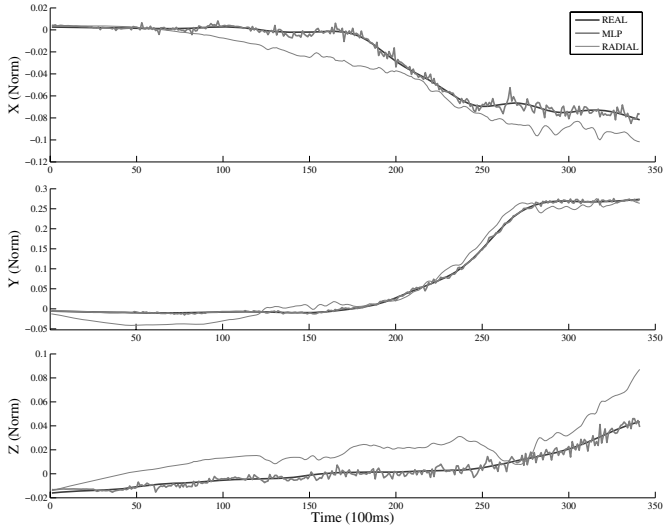

Fig.6. Real takeoff in position vs. simulated takeoff in position with MLP (60 neurons in hidden layer) and Radial Basis (5 radial neurons and 100 SPREAD)

Finally, table I summarizes the errors for the degrees of freedom of attitude and position, respectively.

TABLE I
MINIMUM SQUARED ERROR FOR RADIAL BASIS AND MLP

	Takeoff		Flight		Landing	
Attitud	MLP	Radial	MLP	Radial	MLP	Radial
Roll	4,17E-06	3,47E-07	8,53E-05	1,72E-04	3,13E-05	2,34E-05
Pitch	2,82E-06	3,00E-07	9,57E-05	5,01E-03	4,76E-05	2,27E-05
Yaw	3,51E-06	4,23E-07	2,63E-05	1,18E-02	2,81E-05	6,14E-06
Position	MLP	Radial	MLP	Radial	MLP	Radial
X	8,13E-06	2,08E-04	2,06E-04	2,53E-01	8,66E-05	6,76E-05
Y	6,69E-06	3,39E-04	1,13E-04	6,37E-01	6,25E-05	9,25E-05
Z	7,33E-06	3,20E-04	1,09E-04	7,16E-03	8,43E-05	4,17E-05

Depending on the type of simulation, the errors are higher for one network or the other.

In general, a better performance was observed on the MLP, although the signal tracking achieved by the radial base network is also considerably. The main difference lies in the error, which in this case was better for the MLP, within the required training time. While a radial basis network takes a few minutes on its training to yield the results mentioned before, a MLP takes 18 hours to achieve the results shown. It is important to note that after the network has been trained a simulation step is practically instantaneous as it represents the multiplication of the input vector and neural weights matrix.

It has to be taken into account that radial basis networks have a better performance for local approximations [4][5]. This is confirmed on the Table I in the column corresponding to position at takeoff and landing.

On the other hand, global approximations are better achieved by the MLP, which can be observed in the larger figures corresponding to the flights, where the MLP exceeds the error rates of the radial basis networks.

V. CONCLUSIONS

First of all, the main objective is to open a new method for dynamic systems identification, like the UAV, by means of supervised neural networks. The methods described and the results obtained confirm the feasibility of this new method.

With the networks behaviour, it is proved empirically that the flight of a helicopter has stages that are better to be treated separately in little phenomena. Table I shows better results for little stages with takeoff or landing were the ground effect exist. This observation is critical to realize the needs of using not only specific networks for the takeoff or landing, but also for some particular fly manoeuvres, under different physical and weather conditions.

UAV simulation based on neural networks is an excellent alternative to other simulations based on mathematical methods, because their results are similar, however, neural networks are faster and do not require large computing power or complicated calculations. At first glance, the main disadvantage is the needs to have separate neural networks for each flight stage, yet mathematical models are also specific for each one of the stages, one potential solution for this issue might be the usage of another neural network that activates each specific network at the appropriate moment.

In general, the identification models use attitude and XYZ velocity, to model the system. In this work a model was chosen in attitude and position due to the presence of noise in the velocity for slow and short motions. In future works the model attitude - velocity will be studied.

The importance of this work is the new lines of investigation it suggests and the confirmation of neural networks as a valid tool for system identification. Although this is a first approach, the results obtained with this identification are very encouraging for further works, although its utility may be considered doubtful compared to other methods. On the other hand it is clear the requirement of more knowledge in this field and more improvements in its technique

ACKNOWLEDGMENT

Authors would like to thank Spanish Administration (Ministerio de Educación y Ciencia), for financing the Project "*Guiado, Navegación y Control de Vehiculo Aéreo Autónomo (VAMPIRA)"*, (DIP 2003-01767)

REFERENCE

[1] Office of the Secretary of Defense, "Unmanned aircraft systems (UAS) roadmap, 2005 – 2030" 2005

[2] López Ruiz, José Luis "Helicópteros: Teoría y diseño conceptual". ETSI Aeronáuticos, Madrid, 1993

[3] Freeman, J.A. and Skapura, D.M. "Neural Networks. Algorithms, Applications, and Programming Techniques". Addison-Wesley, Massachusetts, 1991

[4] Chen, S.; Cowan, C.F.N. and Grant, P.M. "Orthogonal Least Squares Learning Algorithm, for Radial Basis Function Networks". IEEE Transactions on neural networks, Vol.2, nº2, March 1991

[5] Craddock, R.J. and Warwick, K. "Multi-Layer Radial Basis Function Networks An Extension to the Radial Basis Function". IEEE International Conference on Neural Networks, 1996

[6] Hopfield, J. "Neural networks and phisycal systems with emergent collective computational abilities". Proceedings of the National Academy of Science, Vol.81, pages 3088-3092. National Academy of Sciences, 1982

[7] Norgaard, M.; Ravn, O.; Poulsen, N.K. and Hansen, L.K. "Neural networks for Modelling and Control of Dynamic Systems". Springer-Verlag, London, 2001

[8] Jordan, M. "Serial order: A parallel distributed processing approach". Technical report, Institute for Cognitive Science. University of California, 1986

[9] Elman, J. "Finding structure in time". Cognitive Science, 14:179-211, 1990

[10] Narendra, K.S. and Parthasarathy, K. "Identification and Control of Dynamical Systems Using Neural Networks". IEEE Transactions on neural networks, Vol.1 Nº 1, March 1990

[11] Li, J.H.; Michel, A.N. and Porod, W. "*Qualitative Analysis and Synthesis of a Class of Neural Netwroks*". IEEE Trans. Circuits Syst., Vol.35. Nº 8, Aug. 1988

[12] Demuth, Howard and Beale, Mark "*Neural Networks TOOLBOX@ Version 4.0*" The Math Works, 2004