AIRCRAFT CONFIGURED BY CONTROL OF THE PREDICTIVE NEURAL TYPE

Nei Cardoso Cardenuto, Atair Rios-Neto

Abstract—**The control-configured vehicle (CCV) approach for aircraft non-typical maneuvers has usually been carried out by eigenstructure assignment in order to decouple flight dynamic modes. This eigenstructure assignment approach, with eigenvalues and eigenvectors placement is thus restricted to a known and linear systems model. This work develops and tests a predictive neural control new technique applied to the aircraft non-typical maneuvering problem. This technique allows to deal with the mode decoupling by using artificial neural network (ANN) training in order to emulate the aircraft dynamics, and by using a predictive control to constraint the maneuvers. A comprehensive presentation of the predictive control approach is made including analytic recurrent expressions of gradients needed in the optimization calculations. To demonstrate the technique performance tests are done using a literature model of a stabilized aircraft and commanded maneuvers of: pitch pointing; altitude rate; pitch pointing and altitude rate simultaneously; and stabilization.**

Index Terms— Control Configured Vehicle, Aircraft Control, Neural Network, Predictive Control.

I. Introduction

The high performance military aircrafts design advances have been focused in the avionics weapon aiming systems in conjunction with high maneuverability control in order to expand the target engagement envelope and evasive maneuvers. Back in 1972, the F-104 Phanton YRF-4C prototype already had the CCV concept tested and denominated PACT ("Precision Aircraft Control Technology"). These vehicles usually need to be hardware specially configured with canards for control augmentation and fuel systems tank exchange for allowing a gravity center displacement in order to increase control components. In terms of control all the implementation up to now have been based on the mode decoupling obtained by the eigenstructure assignment The theoretical work of Srinathkumar [1] supports the possibility of directly controlling decoupled modes in linear systems and several works (e.g., Andry et al [2], Sobel et al [3]-[6], and Garrard [7]) have focused their results on the linear systems mode decoupling by state or output feedback for eigenstructure allocation. This approach needs model knowledge and eigenstructure assignment to be implemented, and are subjected to the limitation that parametric variation due to maneuver conditions can couple the modes again.

The predictive control approach is an attractive method that provides more freedom of design. It allows an output predefined profile reference to be tracked by the aircraft, with the control variables being constrained inside the saturation levels. The decoupling is obtained by the on-line optimization of the output receding horizon errors through the minimization of a quadratic function, where output errors to prescribed references are weighted against control smoothness. The decoupling is thus obtained by restricting the output time response at the desired reference values; for example, if one desires pitch angles constant and path angle null it is enough to set the output components at these values in the horizon future steps.

The model identification is part of the predictive control design and in this work it is implemented using a NARMA neural network model [8]. This internal ANN model provides the propagated horizons and the analytic gradient for the predictive control optimization algorithm.

II. Neural Predictive Control

The neural predictive control is carried out in two stages. The first is to identify the system using the artificial neural network (ANN) and the second to include this ANN identification model as an internal model for the control calculations. The (ANN) function provides the sensitivity of output to input control in order to feed the optimization algorithm. Figure 1 shows the

This work was supported in part by EMBRAER (Empresa Brasileira de Aeronáutica).

N. C. Cardenuto was with INPE (Instituto de Pesquisas Espaciais). He is now (2003) with GET/GEH Defense Division, EMBRAER, São José dos Campos, CEP 12227-901, BR (phone: 55-12 3927-1055; fax: 55-12 3927-2614; e-mail: nei.cardenuto@embraer.com.br).

A. Rios-Neto is presently with INPE (Instituto de Pesquisas Espaciais) and is a consultant in the GDT division, EMBRAER, São José dos Campos, CEP 12227-901, BR (e.mail:atairrn@uol.com.br)

architecture of this method. The neural network as an internal model in the optimization scheme allows the analytical gradient extraction without need of system model.

Even the most recent works in this area (Sorensen et al [9], Zhao et al [10]) have considered particular situations for Jacobian and gradient calculations. Thus, for the sake of completeness in what follows, the analytical Jacobian and gradient are developed for a general feedforward neural net architecture with any number of hidden layers and type of activation function [8]. The full analytic gradient calculation allows the generalization for tapped delay times and horizon propagation specifications.

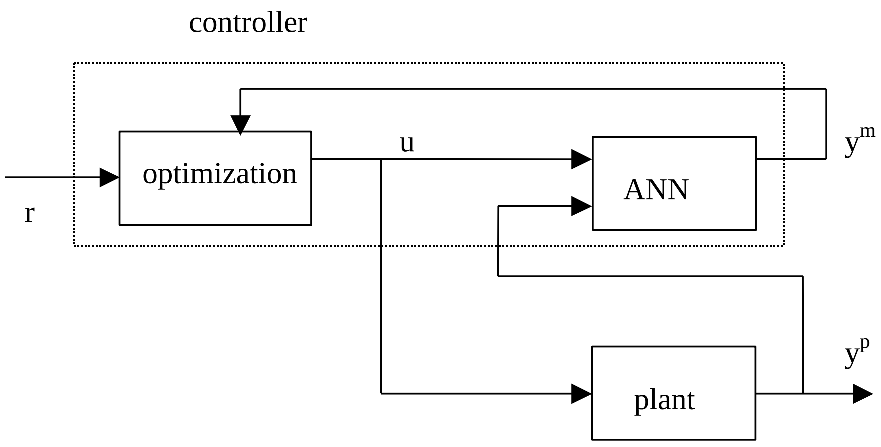

Fig. 1 – Predictive Neural Control Architecture

Optimization problem

Consider a dynamic system described by (1):

$$\dot{\mathbf{x}} = f(\mathbf{x},\mathbf{u}) \quad , \quad \mathbf{y} = h(\mathbf{x},\mathbf{u}) \tag{1}$$

where $\mathbf{x} \in R^n, \mathbf{u} \in R^m, \mathbf{y} \in R^r$ are the system state, control and output, respectively. The method consists in minimizing the following performance index (2):

$$\begin{aligned} J_i(\mathbf{u}) &= \frac{1}{2}\{ \sum_{k=i+N_1}^{i+N_2} [\,\hat{\mathbf{y}}(k) - \mathbf{r}(k)\,]^T \mathbf{R}\ [\hat{\mathbf{y}}(k) - \mathbf{r}(k)\,] + \sum_{i}^{i+N_u-1} \mathbf{\Delta u}^T\, \mathbf{R_u}\, \mathbf{\Delta u}\ \} \\ &= \frac{1}{2}\{ \sum_{k=i+N_1}^{i+N_2} \hat{\mathbf{e}}(k)^T\, \mathbf{R}\ \hat{\mathbf{e}}(k) + \sum_{i}^{i+N_u-1} \mathbf{\Delta u}^T\, \mathbf{R_u}\, \mathbf{\Delta u}\ \} \end{aligned} \tag{2}$$

Where the output errors with respect to a specified reference trajectory and control smoothing are weighted in the quadratic form, and the outputs and control are constrained such that: $|\mathbf{e}| \le \mathbf{e}_{max}$, $|\mathbf{u}| \le \mathbf{u}_{max}$, and $|\Delta\mathbf{u}| \le \Delta\mathbf{u}_{max}$.

The optimization of performance index (2) will certainly yield a local solution because the output and control constraints are attained at each sample time. The neural net emulator is trained for providing a one step ahead output predictions and arranged in cascade to extend the model propagation horizon at discrete steps. The complexity emerges when there are general delayed system outputs in the neural network inputs. Herein this problem was considered and a recursive formula was developed [8] which allows the analytic Jacobian and gradient determination for the cascaded neural nets. Figure 2 shows the neural net propagation starting at the initial instant "i" to the final instant "i+N_2": The control horizon "Nu" usually is less than the output horizon, and control is maintained constant at the value u(i+Nu) from "i+Nu" to "i+N_2". For this problem, for the sake of generalization, the delayed inputs corresponding to outputs feedback are considered of "n" order.

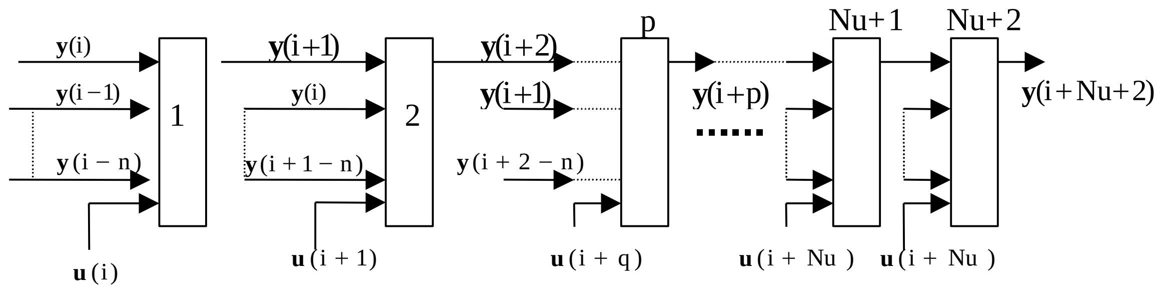

Fig. 2– Net propagation

Jacobian and gradient calculation

For $N_1 = 1$ in (2), for formulae simplification, the gradient with respect to control and only of the term corresponding to the tracking error is (3):

$$\frac{\partial I_i^E(\mathbf{u})}{\partial \mathbf{u}(j)}^T = \sum_{k=1}^{N_2}\left[\frac{\partial \hat{\mathbf{y}}(i+k)}{\partial \mathbf{u}(j)}\right]^T \mathbf{R}\,[\hat{\mathbf{y}}(i+k) - \mathbf{r}(i+k)], \qquad (3)$$
$$j = i, i+1, \ldots, i+Nu-1, \quad j < i+k$$

The Jacobian at the "i" instant is defined (4) as:

$$\mathbf{J}_{yu}(i) = \begin{bmatrix} \left.\frac{\partial \hat{\mathbf{y}}(i+1)}{\partial \mathbf{u}(i)}\right|^T & \left.\frac{\partial \hat{\mathbf{y}}(i+2)}{\partial \mathbf{u}(i)}\right|^T & \left.\frac{\partial \hat{\mathbf{y}}(i+3)}{\partial \mathbf{u}(i)}\right|^T & \ldots\ldots. & \ldots\ldots. & \left.\frac{\partial \hat{\mathbf{y}}(i+N_2-1)}{\partial \mathbf{u}(i)}\right|^T & \left.\frac{\partial \hat{\mathbf{y}}(i+N_2)}{\partial \mathbf{u}(i)}\right|^T \\ 0 & \left.\frac{\partial \hat{\mathbf{y}}(i+2)}{\partial \mathbf{u}(i+1)}\right|^T & \left.\frac{\partial \hat{\mathbf{y}}(i+3)}{\partial \mathbf{u}(i+1)}\right|^T & \ldots\ldots. & \ldots\ldots. & \left.\frac{\partial \hat{\mathbf{y}}(i+N_2-1)}{\partial \mathbf{u}(i+1)}\right|^T & \left.\frac{\partial \hat{\mathbf{y}}(i+N_2)}{\partial \mathbf{u}(i+1)}\right|^T \\ \ldots\ldots. & \ldots\ldots. & \ldots\ldots. & \ldots\ldots. & \ldots\ldots. & \ldots\ldots. & \ldots\ldots. \\ \ldots\ldots. & \ldots\ldots. & \ldots\ldots. & \ldots\ldots. & \ldots\ldots. & \ldots\ldots.. & \ldots\ldots. \\ 0 & \ldots\ldots. & \ldots\ldots. & \ldots\ldots.. & \ldots\ldots. & \left.\frac{\partial \hat{\mathbf{y}}(i+N_2-1)}{\partial \mathbf{u}(i+N_u-1)}\right|^T & \left.\frac{\partial \hat{\mathbf{y}}(i+N_2)}{\partial \mathbf{u}(i+N_u-1)}\right|^T \\ 0 & \ldots\ldots. & 0 & \left.\frac{\partial \hat{\mathbf{y}}(i+N_u)}{\partial \mathbf{u}(i+N_u-1)}\right|^T & \ldots\ldots. & \left.\frac{\partial \hat{\mathbf{y}}(i+N_2-1)}{\partial \mathbf{u}(i+N_u-1)}\right|^T & \left.\frac{\partial \hat{\mathbf{y}}(i+N_2)}{\partial \mathbf{u}(i+N_u-1)}\right|^T \end{bmatrix} \qquad (4)$$

where each matrix element is (5):

$$\mathbf{J}_{yu}(i+p, i+q) = \frac{\partial \hat{\mathbf{y}}(i+p)}{\partial \mathbf{u}(i+q)}, \quad \text{with} \qquad p > q \qquad (5)$$

corresponding to the matrix "$i+p$" line and "$i+q$" column respectively; where the diagonal terms are calculated by straight application of the backpropagation rule and the out of diagonal terms are calculated in a recursive way as will be shown in what follows; and where the control values are the same after the control horizon N_u and equal to the value at this point in the predictive control horizon.

To get the recursive calculations, start considering that the intermediary Jacobians $\mathbf{J}_{yy}$ of the output at the "p" step with respect to the delayed outputs at the "q" step (that is inputs to the neural network at the "p" step, see Figure 2), are obtained by straight application of the backpropagation rule, as (6):

$$\mathbf{J}_{yy}(i+p, i+q) = \frac{\partial \hat{\mathbf{y}}(i+p)}{\partial \hat{\mathbf{y}}(i+q)} \tag{6}$$

So, that for a given “n” generic order (delayed outputs) the Jacobian can be constructed in the recursive form (7):

$$\mathbf{J}_{yu}(i+p, i+q) = \sum_{j=p-1}^{p-n} \mathbf{J}_{yy}(i+p, i+j) * \mathbf{J}_{yu}(i+j, i+q) \tag{7}$$

Where: $j = p-1, p-2, \ldots., p-n$, $q = 0, \ldots., Nu-1$, and $p = q+1, \ldots., Nu$

For $p > Nu$, the Jacobian takes the following form (8):

$$\mathbf{J}_{yu}(i+p, i+Nu) = \sum_{j=p-1}^{p-n} \{\mathbf{J}_{yy}(i+p, i+j) * \mathbf{J}_{yu}(i+j, i+Nu)\} + \mathbf{J}_{yu}(i+p, i+p-1) \tag{8}$$

for $p = Nu+2, \ldots. N2$,

Due to causality these Jacobians are calculated as (9) and (10):

$$\mathbf{J}_{yy}(i+p, i+q) = \mathbf{0}, \quad \text{for} \quad p-q > n \tag{9}$$

$$\mathbf{J}_{yu}(i+p, i+q) = \mathbf{0}, \quad \text{for} \quad p \le q \tag{10}$$

The total gradient, including the control smoothing component is (11):

$$\nabla_{\mathbf{u}}^{T}(i) = \mathbf{J}_{yu}(i) * \mathbf{e}_R(i) \quad + \quad \frac{\partial \Delta \mathbf{u}}{\partial \mathbf{u}}(i)\, \Delta \mathbf{u}(i) \tag{11}$$

Where, the weighted output error is (12):

$$\mathbf{e}_R(i) = \mathbf{R} * \mathbf{e}(i) = \mathbf{R} * [\hat{\mathbf{y}}(i+k) - \mathbf{r}(i+k)] \tag{12}$$

and the $\Delta \mathbf{u}$ derivatives (13):

$$\frac{\partial \Delta \mathbf{u}}{\partial \mathbf{u}}(i) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ \ldots.. & \ldots.. & \ldots.. & \ldots.. & \ldots.. \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \tag{13}$$

The recurrent formula for Jacobian and gradient calculation allows a generalized approach related to the system order, output and control horizons. Once the total gradient is available the optimization problem to get the predictive control in each step can be numerically solved.

III. DEMONSTRATION EXAMPLE

The demonstration example is that of an aircraft longitudinal motion model, the one presented in the Andry et al. work [2], with the same data for analysis and results evaluation. The model is of an aircraft longitudinal motion at 3000 ft of altitude, 0.6 mach [2]. Figure 3 illustrates the state and input variables in the aircraft coordinate system:

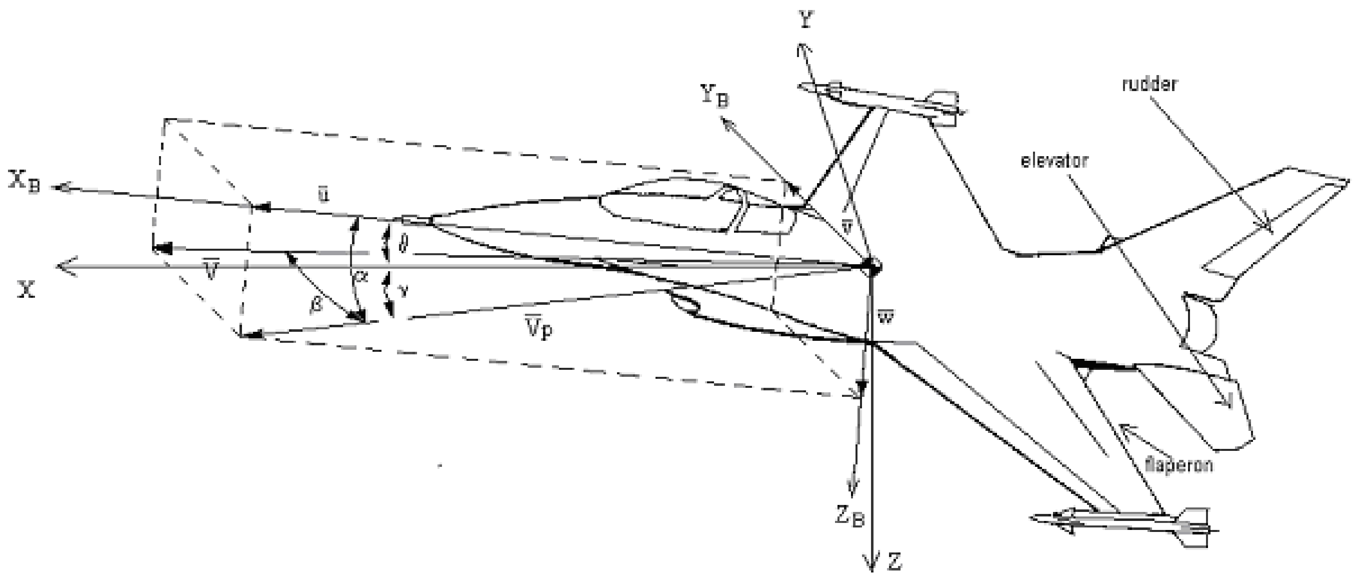

Fig. 3 – Variables Definition

The variable definitions are:

θ – pitch angle (rad)
q – "pitch rate" (rad / s)
α – attack angle (rad)
δ_e – elevator deflection (rad)
δ_f – flaperon deflection (rad)

The input control is defined as (14):

$$u = [\delta_{ec} \quad \delta_{fc}] \tag{14}$$

where:

δ_{ec} – "elevator command" (rad)
δ_{fc} – "flaperon command" (rad)

The linear system is defined as (15):

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{Ax} + \mathbf{Bu} \\ \mathbf{y} &= \mathbf{Cx} \end{aligned} \tag{15}$$

where the variables are (16), (17) and (18):

$$\mathbf{x} = [\theta \quad q \quad \alpha \quad \delta_e \quad \delta_f]^T \tag{16}$$

$$\mathbf{u} = [\delta_{ec} \quad \delta_{fc}]^T \tag{17}$$

$$\mathbf{y} = [\theta \quad q \quad \gamma \quad \delta_e \quad \delta_f] \tag{18}$$

where the pitch angle is related with the flight path and attack angles by (19):

$$\theta = \gamma + \alpha \tag{19}$$

The system (20), control (21) and output (22) matrixes are:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & -0.87 & 43.22 & -17.25 & -1.58 \\ 0 & 0.99 & -1.34 & -0.17 & -0.25 \\ 0 & 0 & 0 & -20 & 0 \\ 0 & 0 & 0 & 0 & -20 \end{bmatrix} \tag{20}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 & 20 \end{bmatrix}^T \tag{21}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{22}$$

The open loop eigenvalues are (23):

$$\boldsymbol{\lambda} = [-7.66 \quad 5.45 \quad 0.00 \quad -20.00 \quad -20.00] \tag{23}$$

The closed loop eigenvalues are taken as [2] and allocated to λ_{cl} (24) by Matlab "place" function in order to stabilize the system in a trivial procedure:

$$\boldsymbol{\lambda}_{cl} = [-5.6 + 4.2j \quad -5.6 - 4.2j \quad -1 \quad -19 \quad -19.5] \tag{24}$$

The corresponding gain matrix "K" in the stabilizing feedback control is (25):

$$K = \begin{bmatrix} 1.4839 & -0.7581 & -4.2955 & 0.4780 & 0.0886 \\ 0.8553 & 0.1136 & -0.0567 & -0.0748 & -0.0535 \end{bmatrix} \tag{25}$$

It should be noticed that there is no eigenvector assignment for this system, which means there isn't decoupling until this step.

The Neural network used for identification model is a three layer perceptron network with 12 fan out, 20 sigmoid and 5 linear neurons in the input, hidden and output layer, respectively. The net inputs are composed of input control at the one tapped delay time, and of outputs at the two tapped delay time, in the total of 12 input elements. Figure 4 shows this architecture.

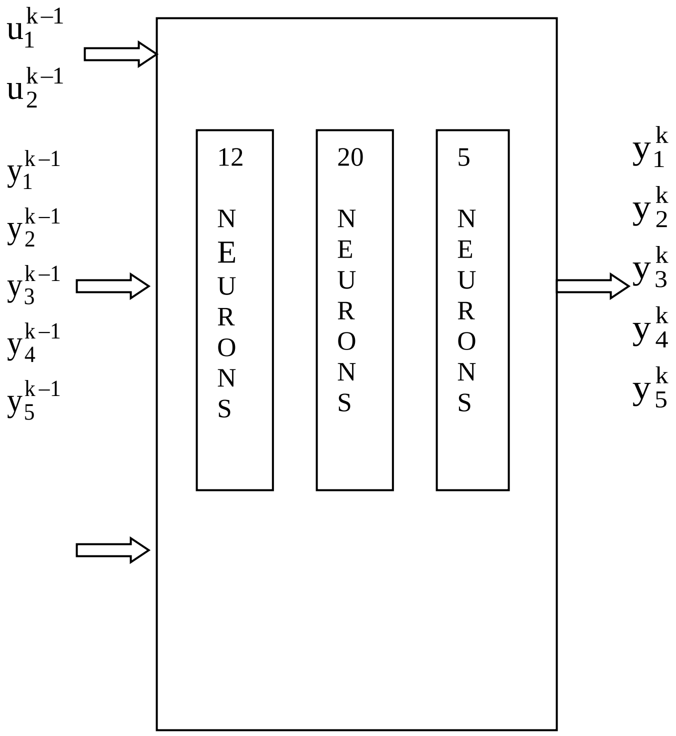

Fig. 4 – Network architecture

The input training pattern used (Figure 5), with 3 stages, yielded a better sensitivity to control inputs. The alternating command inputs components a each time step provided a control weighting training. The training points were taken at each 0,2 seconds and the first 1000 points are shown. The pulse width duration is enough for settling time transient response.

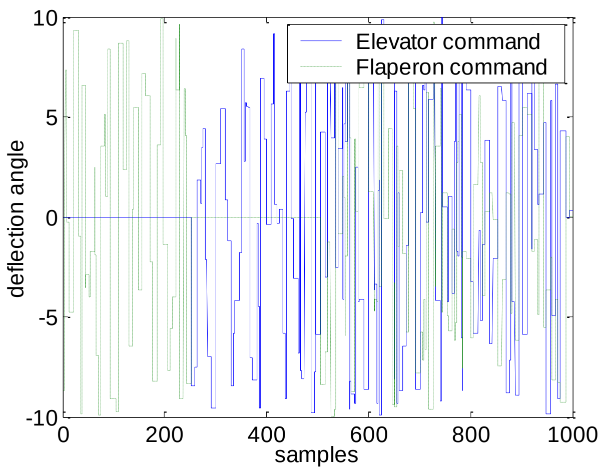

Fig. 5 – Network training pattern

The training method used was the "trainlm" Matlab function which consists of Levenberg-Marquardt algorithm; the initialization method used was the Nguyen-Widrow ("initnw"), and the performance evaluation method used was the meansquared-error (“mse”). The Figure 6 shows the net training performance achieved.

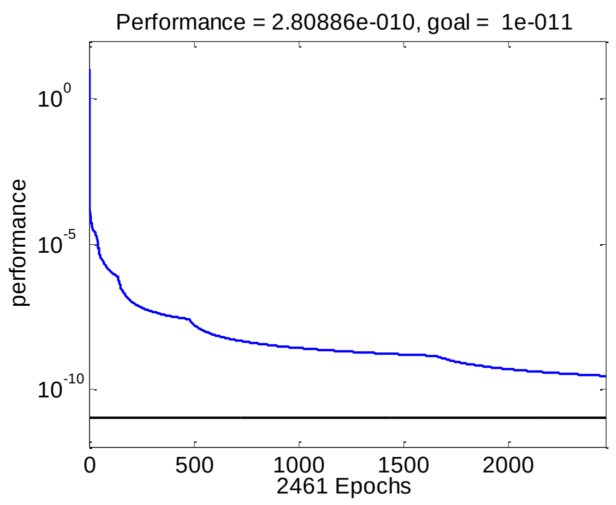

Fig. 6 – Network Training performance

The network validation set was obtained by oscillatory inputs. Figure 7 shows the error profile on the validation data set and Figure 8 shows that same for the pitch rate.

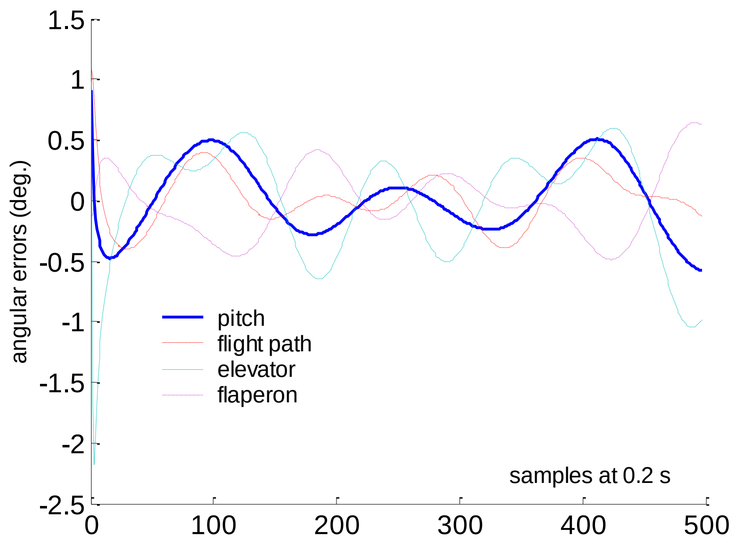

Fig. 7 – Neural net output erros

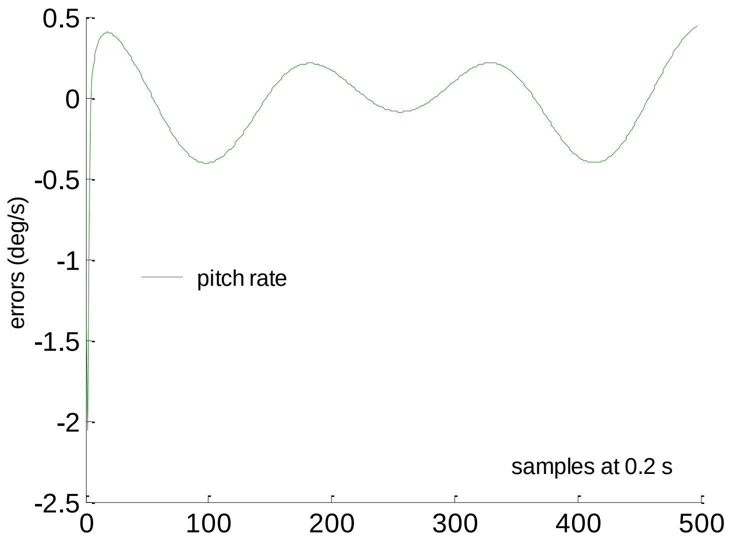

Fig. 8 – Pitch rate output error

IV. RESULTS

The diagonal matrix terms R(i,i) in (2), herein will be cited as Ri, and the control matrix Ru(i,i) diagonal terms are assumed all equal to Ru.

Figure 9 shows a case of control with Nu=1, Ru = 0.5, R1=10, R2=R3=0. This means that there is pitch demand only and the flight path and attitude profiles are typical conventional coupled flight dynamics. Figure 10 points indicate the flight path, and a line segment at each point represents the attitude. The graphic scale was dimensioned in order to yield a visibility augmentation of the attitude angle.

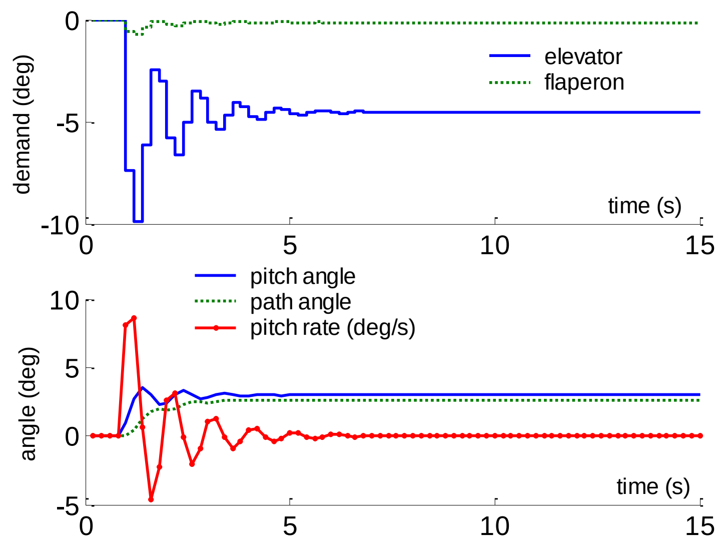

Fig. 9 – 3-degree pitch demand: R1=10, R2=R3=0, Nu=1, N2=3, Ru = 0.5.

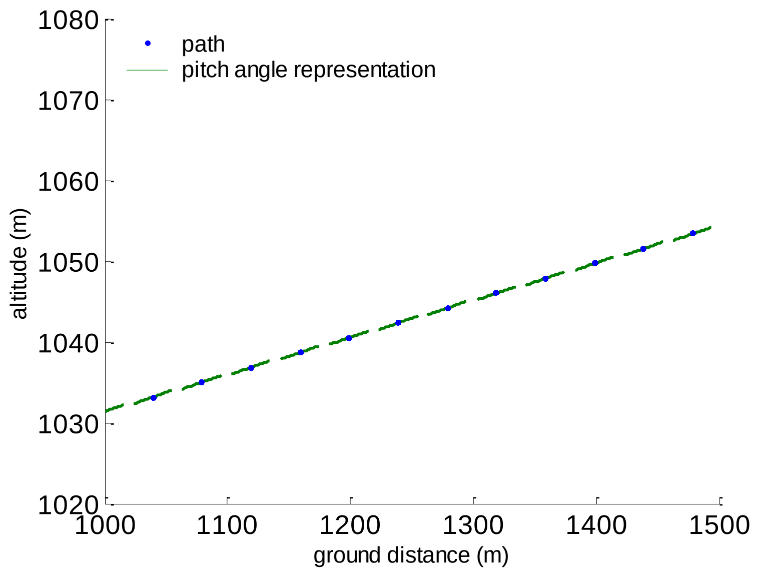

Fig. 10 – Flight path for 3-degree pitch demand: R1=10, R2=R3=0, Nu=1, N2=3, Ru = 0.5.

Figure 11 shows the pitch pointing case. Herein one sees an oscillation transient and the Figure 12 shows the altitude change prior to stabilization due to the transient.

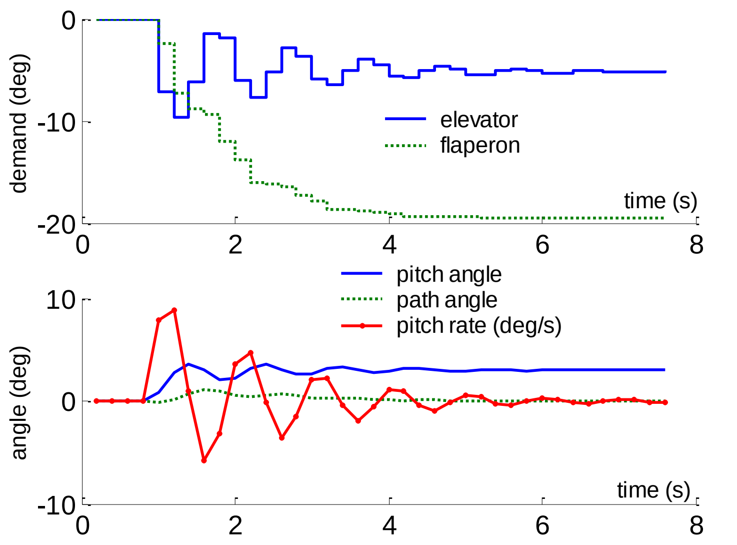

Fig. 11 – 3-degree pitch pointing: R1=10, R2=R3=0, Nu=1, N2=3, Ru = 0.5.

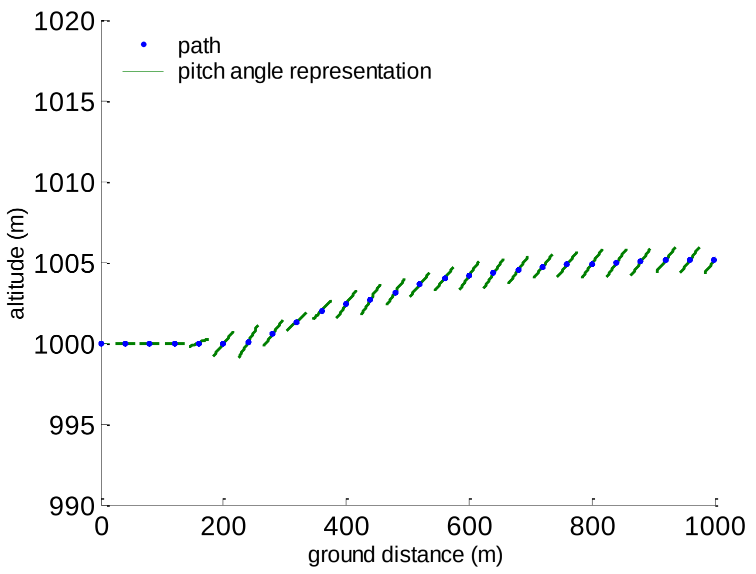

Fig. 12 – Fligth path for 3-degree pitch pointing: R1=10, R2=R3=0, Nu=1, N2=3, Ru = 0.5.

This methodology allows also a greater freedom of design for profiles demand. The next case (Figures 13 and 14) the R2 component was set nonzero, which means to add a pitch rate restriction. This setup yields a transient damping and so, a better performance.

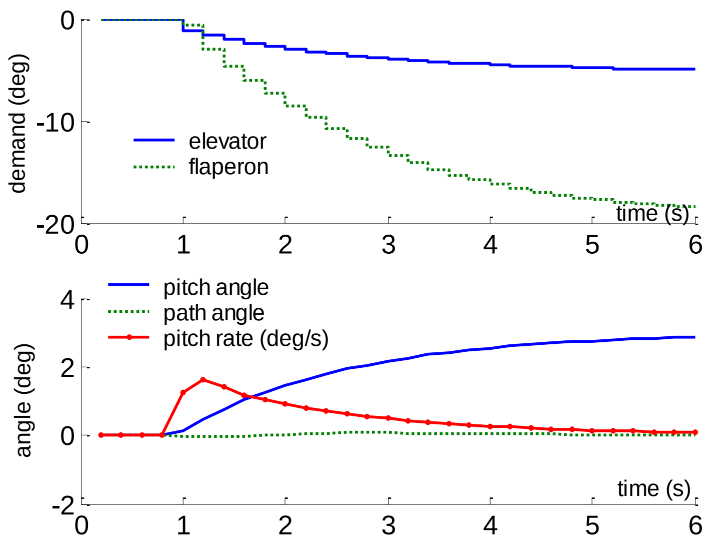

Fig. 13 – 3-degree pitch pointing: R1=R2=R3=10, Nu=1, N2=3, Ru = 0.5.

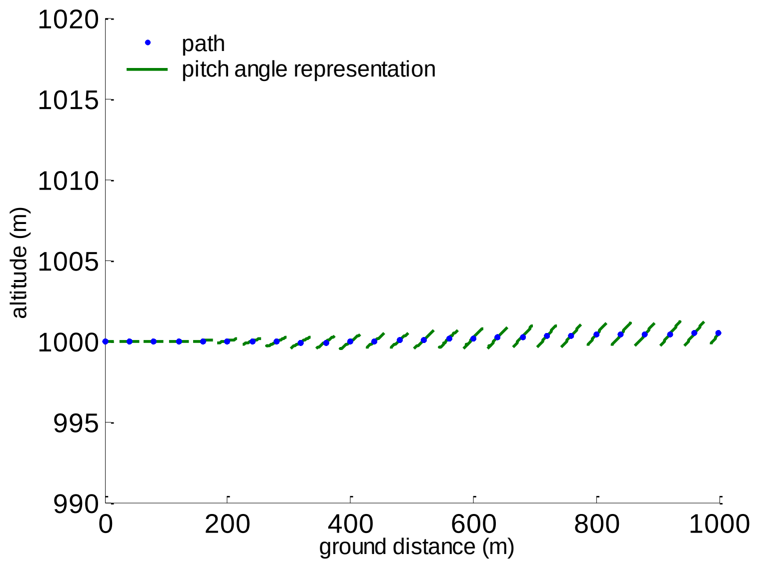

Fig. 14 – Fligth path for 3-degree pitch pointing: R1=R2=R3=10, Nu=1, N2=3, Ru = 0.5.

Figures 15 and 16 show the altitude rate mode.

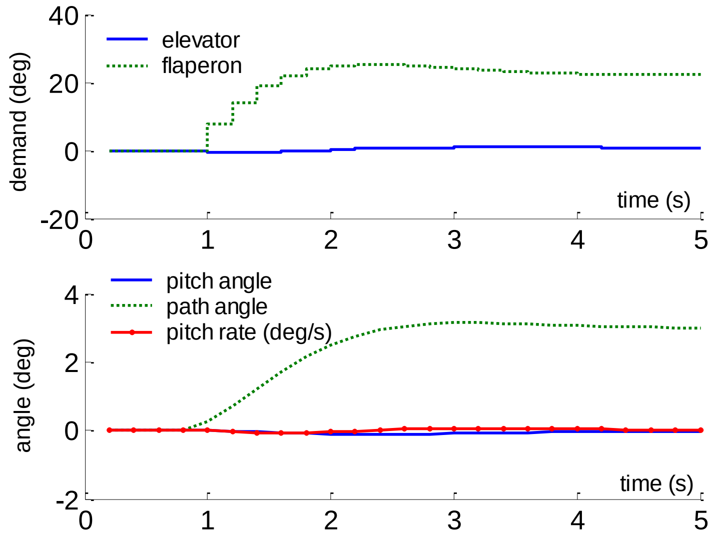

Fig. 15 – 3-degree altitude rate: R1=R2=R3=10, Nu=1, N2=3, Ru = 0.5.

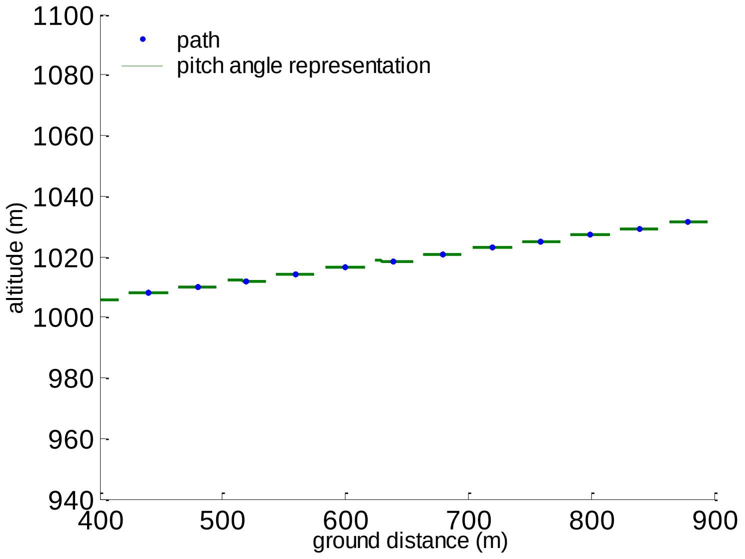

Fig. 16 – Flight path for 3-degree altitude rate: R1=R2=R3=10, Nu=1, N2=3, Ru = 0.5.

Figures 17 and 18 show the negative 3-degree flight path demand with a positive 3-degree pitch demand simultaneously. The results confirm the methodology possibilities.

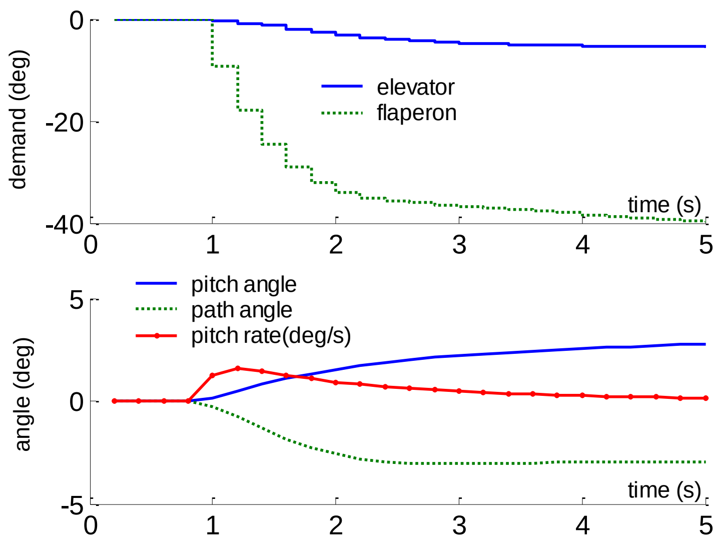

Fig. 17 – Negative 3-degree altitude rate with positive 3-degree pitch pointing: R1=R2=R3=10, Nu=1, N2=3, Ru = 0.5.

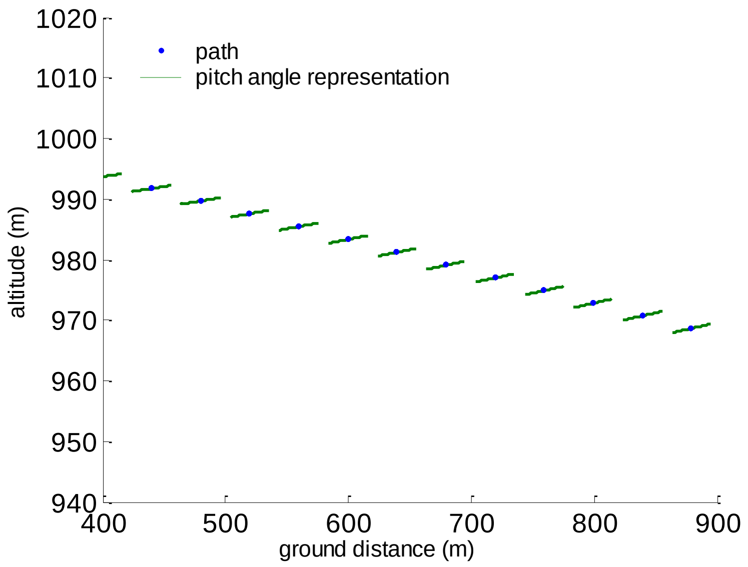

Fig. 18 – Flight path for negative 3-degree altitude rate with positive 3-degree pitch pointing: R1=R2=R3=10, Nu=1, N2=3, Ru = 0.5.

The next cases show the possibility of stabilization of an unstable system. Figure 19 shows an unstable case with R2 =1 pitch rate demand. Figure 20 shows a same case with R2 set to 10 and the system stabilization.

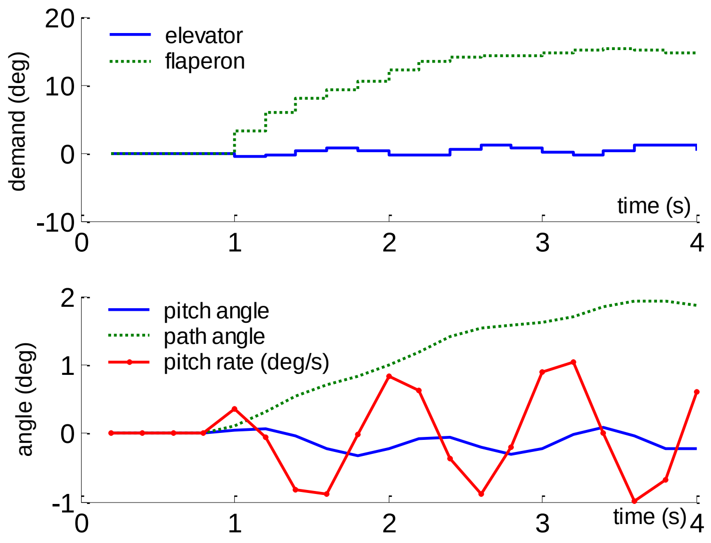

Fig. 19 – 3-degree altitude rate: R1=10, R2=1, R3=1, Nu=2, N2=4, Ru = 0.05.

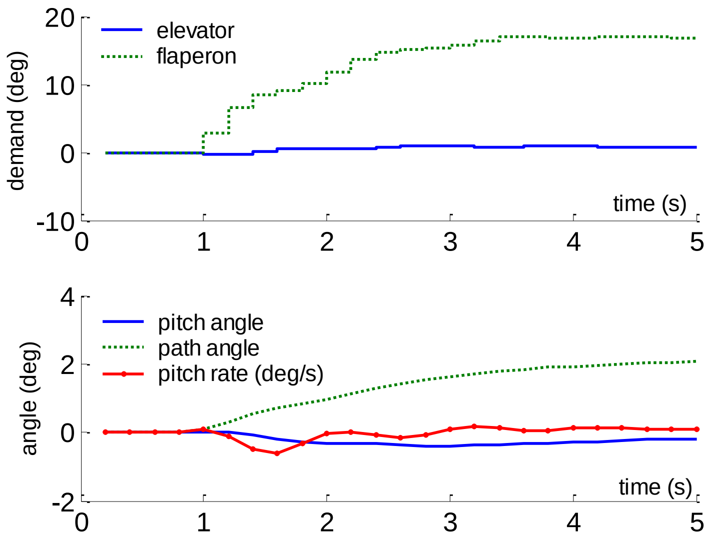

Fig. 20 – Flight path for 3-degree altitude rate: R1=10, R2=10, R3=1, Nu=2, N2=4, Ru = 0.05.

V. CONCLUSIONS

The results demonstrate that the predictive neural control technique yields the output profile requirements generalization when compared to the eigenstructure assignment. The use of a Neural Network as an internal model gives the possibility of the technique being also implemented in non-linear systems control.

Although in the demonstration example the linear region for unstable cases can have been violated, the response analysis can be considered valid in order to demonstrate only numerical performance. The stabilization by output restriction was another application possibility verified and it suggested using open loop eigenvalues for the stabilization performance.

In what concerns the modeling with the Neural Network, the training pattern choice was found to be relevant; several training sets achieved the goal precision performance, but failed as an identifier model. The net sensitivity to the inputs (delayed outputs and controls) was affected by these patterns. The pattern used with three stages, where each control was separately set at the two initial periods showed better performance in sensitivity sense. This behavior suggests a study about net training, focusing sensitivity.

The proposed technique, that allows to extend the decoupling mode design for non-typical maneuvers and stability control, opens the possibility of several applications, as flight director, trim flight parameters adjusting and other systems where the output references profile are the objective to be achieved.

VI. References

[1] Srinathkumar, S., “Eigenvalue / Eigenvector Assignment Using Output Feedback”, *IEEE Transactions on Automatic Control*, AC-23, No. 1, 1978, pp. 79-81.

[2] Andry, A.N. Jr, Shapiro, and E.Y., Chung, J.C., “Eigenstructure Assignment for Linear Systems”, *IEEE Transactions on Aerospace and Electronic Systems,* Vol AES-19, No.5, Sept. 1983, pp. 711-730.

[3] Sobel, K.M. and Shapiro, E.Y., “A Design Methodology for Pitch Pointing Flight Control Systems”, *Journal of Guidance Control and Dynamics,* Mar.-Apr. 1985, pp.181-187.

[4] Sobel, K.M. and Shapiro, E.Y., “Eigenstructure Assignment for Design of Multimode Flight Control Systems”, *IEEE Control Systems Magazine,* May 1985, pp. 9-14.

[5] Sobel, K.M., Shapiro, E.Y. and Andry, A.N. Jr., “Eigenstructure Assignment”, *International Journal of Control*, Vol. 59, No.1, 1994, pp. 13-37.

[6] Sobel, K.,M. and Lallman, F. J., “Eigenstructure Assignment for the Control of Highly Augmented Aircraft”, *Journal of Guidance*. Vol. 12, No. 3, 1988, pp. 318-324.

[7] Garrard, W.L., “Lateral Directional Aircraft Control Using Eigenstructure Assigment”, *Journal of Guidance, Engineering Notes,* Vol.21, No. 3, 1998, pp.523-525.

[8] Cardenuto, N.C. “Aeronaves configuradas por controle do tipo preditivo neural”. Ph.D. dissertation, Computation and Applied Mathematics Laboratory, INPE, São José dos Campos, SP, 2003.

[9] Sorensen, P., H.; Norgaard, M.; Ravn, O.; Poulsen, N., K. “Implementation of neural network based non-linear predictive control”, *Neurocomputing,* v.28, p. 37-51, 1999.

[10] Zhao, H.; Guiver, J.; Neelakantan, R.; Biegler, L.,T. “A Nonlinear Industrial Model Predictive Controller Using Integrated PLS and Neural Net State-Space Model”, *Control Engineering Practice*, v. 9, p. 125-133, 2001.