Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Series-Parallel Approach to On-line Observer based Neural Control of a Helicopter System

Louw vS. Hager * **Kenneth R. Uren** *, **George van Schoor** **

* *School of Electrical, Electronic and Computer Engineering, North-West University, Potchefstroom, South Africa (e-mail: 23911999@nwu.ac.za; kenny.uren@nwu.ac.za).*
** *Unit for Energy Systems, North-West University, Potchefstroom, South Africa (e-mail: george.vanschoor@nwu.ac.za).*

Abstract: This paper is concerned with the control of an under-actuated, uncertain, delayed non-linear system through the implementation of artificial neural networks(ANNs). The aim is the development of a series-parallel training scheme for the on-line observer to ensure faster convergence and more accurate estimations. Reinforcement learning is used to improve future performance and maintain stability while an estimated tracking error is minimised. Lyapunov stability measures are employed to guarantee the uniform ultimate boundedness of the closed-loop tracking error. Real-time learning algorithms are derived for the individual components (observer, actor, critic). Final performance is tested on a mathematical helicopter model and real-world helicopter flight data.

Keywords: Neural Network, Adaptive Control, Helicopter Systems

1. INTRODUCTION

Rotary wing aircraft are challenging dynamic systems since they are non-linear, open-loop unstable and under-actuated MIMO-systems. The purpose of this paper is to illustrate the implementation of an artificial neural network (ANN) controller on a helicopter system by employing an on-line observer of which the design is motivated by off-line procedures.

Neural networks are ideally suited for highly coupled, highly non-linear problems Panchal et al. (2010). The design of adaptive observers have been actively studied in recent years, Sharma and Verma (2010), while the successful implementation of ANNs on helicopter platforms remain fairly new, Nodland et al. (2013). Abbeel et al. (2010) successfully applied machine learning to train (Artificial Intelligence) AI-based systems on aerobatic manoeuvres, but overall ANNs are not given full authority and simply augment PID-type control methodologies, Raimúndez et al. (2006). The most common control method remains PID, Kendoul (2012).

ANN-based control has the ability to approximate dynamic programming equations. Many variants exist, but they typically include an off-line training phase to ensure convergence, He and Jagannathan (2007). The use of reinforcement learning is also common in adaptive-critic schemes. An actor-critic structure, where one structure is responsible for perception (critic) and the other for action (actor), is normally employed. The actor decides on a control scheme while the critic evaluates the outcome of the actor's decisions.

In order to find a suitable observer structure, various ANNs were tested on actual flight data. The NARX-network emerged as the most promising. Its mathematical form also closely represents that of the reduced Brunovsky canonical observer given by (1). This strict-feedback observer has been successfully used for control purposes by the recursive application of backstepping Khalil (2001).

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ &\vdots \\ x_n(k+1) &= f(x(k)) + g(x(k))u(k) + d'(k) \\ y(k) &= x_1(k) \end{aligned} \tag{1}$$

In (1), $x(k) = [x_1^T(k), x_2^T(k), \ldots x_n^T(k)]^T, u, y$ are the respective states, inputs and system observer outputs.

The aim of the work presented in this paper is to combine an off-line series-parallel training structure with the identified NARX network through subtle alterations to the update-law and structure. Supplying the observer network with actual response values augments the estimations and improves overall convergence rates. Final application will be illustrated on a helicopter platform.

In this paper, section 2 provides a description of the states used for control as well as a model summary. The research layout can be found in section 3. Section 4 details the observer while section 5 focusses on the controller. Stability requirements are stated in section 6, with simulation results given in section 7.

2. PLATFORM DESCRIPTION

Controller design and testing will mainly be done on a linear model presented in state-space form. Mettler et al.

(1999) presents a detailed overview. Final validation is based on actual flight data. The state-space representation is

$$x(k+1) = Ax(k) + Bu(k) \tag{2}$$

where

$$x(k) = [u\ v\ p\ q\ \phi\ \theta\ a\ b\ w\ r\ r_{fb}]^T \tag{3}$$

and

$$u(k) = [\delta_{lat}\ \delta_{lon}\ \delta_{col}\ \delta_{ped}]^T. \tag{4}$$

Translational velocities, u, v, w are measured in m/s, angular rates, ϕ and θ in rad/s with roll- and pitch-angles being radians. The rotor-tip path-plane angles are represented by a and b with r_{fb} being an active yaw-damper. Of these 11 states, only the 4 which completely describe the system and exhibit the strongest coupling to specific inputs will be used for stabilisation control. State selection for the helicopter system is based on the singular value decomposition (SVD) method presented in Valavanis (2007). The recommended input-state pairings, Table 1, will be used for estimation and control.

Table 1. Recommended input-output pairing

Input	Pairing
Longitudinal (u_{lon})	Pitch (θ)
Lateral (u_{lat})	Roll (ϕ)
Pedals (u_{ped})	Yaw rate (r)
Collective (u_{col})	Heave velocity (z)

3. RESEARCH APPROACH

After modifying the NARX-network it can be implemented as an on-line observer. Its estimated states will be used for control in a critic-based control scheme as shown in Fig. 1. The state vector is represented by x, the estimated state by $\hat{x}$, and the estimation error by $\tilde{x} = \hat{x} - x$. The number of outputs and the number of states are represented by the indices m and n, respectively.

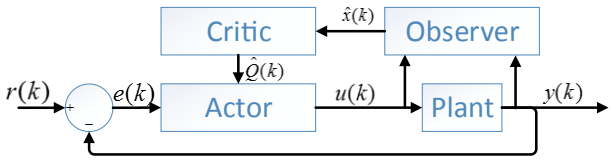

Fig. 1. Subsystem interaction

State estimations, $\hat{x}(k)$, are evaluated by a critic which computes system performance with a modified Bellman equation. The critique signal $\hat{Q}(k)$ in turn adjusts the control policy, $u(k)$, to minimize the tracking error $e(k)$.

The Sarangapani (2006) design approach is followed to demonstrate the uniform ultimate boundedness (UUB) of the system. Boundedness holds in the presence of bounded disturbances and ANN approximation errors. The control-system overcomes the persistent excitation (PE) condition without an off-line training phase. Linear in the parameter (LIP) assumptions are also not required.

In this paper, a novel alteration is made to the observer design, combining the idea implemented in classic, series-parallel off-line NARX-network training, Demuth (2013) with the on-line structure. Additional alterations to the update-law decouple internal parameters which allow for faster optimization with genetic algorithms. Overall the system exhibits faster convergence rates with less oscillatory behaviour on more complex systems. Final validation is done on the model, through the introduction of non-linearities such as input saturation, rate-limiting and artificial noise. The system is also tested on a wind-tunnel validated helicopter model, as well as actual helicopter flight data.

4. ON-LINE OBSERVER

In the case when a sufficient number of states are available, and these states accurately portray the system under observation, a state-feedback controller can be designed for the system, Landau et al. (2011). The aim of this section is to combine the modelling capabilities of the NARX-type neural network with state representation of the reduced Brunovsky canonical form. In doing so, the Brunovksy structure can be used to ensure that an adequate number of states are available while the NARX-network ensures the quality of state information.

4.1 Architecture

Using the universal function approximation capability of the neural network Abdollahi (2011), (1) can be redefined to include an artificial neural network.

$$\begin{aligned} \hat{x}_1(k) &= \hat{x}_2(k-1) \\ \hat{x}_2(k) &= \hat{x}_3(k-1) \\ &\vdots \\ \hat{x}_n(k) &= \hat{w}_1^T(k-1)\phi_1(v_1^T(k-1)\hat{z}_1(k-1)) \end{aligned} \tag{5}$$

The state observer is used to estimate the entire state vector $\hat{x}(k) \in R^{nm}$ through the determination of $\hat{x}_n \in R^m$ which propagates through $\hat{x}_i \in R^m$,for $i = 1, \ldots, n$. Representing the neural network in matrix form, the output- and hidden-layer weight matrices are $\hat{w}_1(k-1) \in R^{(n_1 \times m)}$ and $v_1 \in R^{(n+1)m \times n_1}$ respectively. The hidden-layer activation function is chosen to be of the tan-sigmoid form with a linear activation function on the output-layer. An input vector, $\hat{z}_1(k)$, still has to be defined before the neural network can be implemented as an observer. It will be selected according to the NARX-network definition where the network output depends on past-estimates (outputs) and a delayed input. By defining, $\hat{z}_1(k-1) = [\hat{x}_1^T(k-1), \ldots, \hat{x}_n^T(k-1), u^T(k-1)]^T \in R^{(n+1)m}$, the structure definition of the observer neural network completely takes the form of the NARX-network illustrated in Demuth (2013).

4.2 Weight update law

Sarangapani (2006) recommends a subtle change to the NARX-network's standard update-law, (6), allowing the neural network to estimate the future through output-feedback tuning, (7). This change is the final alteration required for the network to completely assimilate the Brunovsky form where $\hat{y}(k) = \hat{x}_1(k)$. The suggested update law

$$\hat{w}_1(k+1) = \hat{w}_1(k) - \Delta\hat{w}_1(k), \tag{6}$$

with

$$\Delta\hat{w}_1(k) = \alpha_1\phi_1(\hat{z}_1(k))(\hat{x}_n(k) + l_2\tilde{x}_1(k))^T, \quad (7)$$

represents the starting point for a newly suggested observer structure. The inclusion of the n^{th} state-estimation, $\hat{x}_n(k)$, in the update-law ensures that the estimated signal, $\hat{x}_1(k)$, is bounded to the zero-side of the actual system signal. Forcing the magnitude of the estimated signal to be less than the actual signal contributes to system stability and the overall Lyapunov proof, Sarangapani (2006). Neural network update-laws are either error- or reinforcement-driven. This holds true for (7) where the design matrix $l_2 \in R^{m\times m}$ is selected to govern the ratio between error contribution, $\hat{x}_1(k)$, and estimator boundedness, $\hat{x}_n(k)$. The elements need to be selected, $l_2 >> 1$, such that the update-law is primarily error-driven. Even though the approach binds the estimation, the position of the l_2 matrix not only governs the contribution ratio, it also influences the overall adaptation gain and herein lies the problem. To illustrate the effect, consider the SISO case where l_2 is treated as a constant:

$$\Delta\hat{w}_1(k) = \alpha_1 l_2\phi_1(\hat{z}_1(k))\left(\frac{1}{l_2}\hat{x}_n(k) + \tilde{x}_1(k)\right)^T, \quad (8)$$

$$\alpha_{new} = \alpha_1 l_2. \quad (9)$$

For adequate tracking performance of the SISO-system, the design constant should be chosen $l_2 > 1$, resulting in $\alpha_{new} > \alpha_1$. According to Sarangapani (2006) the recommend learning rate for stable weight-updates is $\alpha_1 < \frac{1}{n_1}$. Clearly the choice of l_2 can violate the stability directive resulting, in highly oscillatory behaviour of the estimated signal. Adaptation gain, α_1, is generally selected near the upper bound to ensure fast convergence. The proposed update-law utilizes the $\hat{x}_n$-bound as well as the l_2 ratio without complicating the selection of α_1 through the inclusion of the l_2 contribution. Selection of the final adaptation gain can be done independently from ratio selection if:

$$\Delta\hat{w}_1(k) = \alpha_1\phi_1(\hat{z}_1(k))(l_2\hat{x}_n(k) + \tilde{x}_1(k))^T, \quad (10)$$

where $l_2 << 1$. The contribution of l_2 towards the adaptation gain is minimal, and if it is selected to be small enough, the update error can be approximated as

$$l_2\hat{x}_n(k) + \tilde{x}_1(k) \approx \tilde{x}_1(k). \quad (11)$$

Using the altered update-law facilitates the selection of α_1 by not placing an additional constraint on the maximum adaptation gain. The altered update-law represents only a small part of the proposed changes. By altering the input-vector it is possible to enhance the approximation capabilities of the neural network. In essence, through the Brunovsky form and output definition $\hat{y}(k) = \hat{x}_1(k)$, the observer estimates the future system output. Graphically the altered observer is defined in Fig. 2 with the shift register representing the input to the neural network.

Generally at least two states are required per degree-of freedom. More state information can result in a more accurate control signal, however, in the current design state, information is coupled to the prediction horizon. If the prediction horizon is made too large, the quality of state information decreases due to the delay between network output and error-comparison of that output as shown in Fig. 2. To overcome this constraint it is suggested that past values be included in the shift register. The prediction horizon can be kept small enough thereby preventing estimation degradation, while the amount of state information can be maximized for optimal controller performance. The position of the error computation shifts as indicated in Fig. 2.

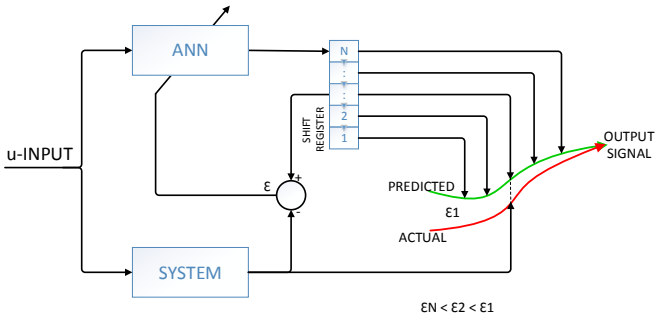

Fig. 2. Past and future states with altered ANN observer

Instead of using the network estimation, $\hat{y}(t)$, for training purposes, the actual off-line output data, $y(t)$, are presented to the network. If the observer structure is altered to include past estimates, as in Fig. 2, it should also be possible to replace these past estimates with the actual past system outputs. In doing so the same advantages gained in the off-line training scheme for the NARX-network can be enjoyed in the on-line observer. Using the actual system output as an input to the neural network, provides a stable anchor-point for future estimations. It increases system stability and accuracy by not only relying on its own estimations but also implementing real system values. The final input vector for the observer neural network is defined as:

$$\hat{z}_1(k) = [y_1^T(k), \ldots, y_{actual}^T(k), \hat{x}_1^T(k), \ldots, \hat{x}_{n-actual}^T(k), u^T(k)]^T \in R^{(n+1)m} \quad (12)$$

where the subscript n still represents the number of states and *actual*, the number of actual historic values from the system. The update-law changes slightly to include the shifted error position:

$$\hat{w}_1(k+1) = \hat{w}_1(k) - \Delta\hat{w}_1(k), \quad (13)$$

$$\Delta\hat{w}_1(k) = \alpha_1\phi_1(\hat{z}_1(k))(l_2\hat{x}_n(k) + \tilde{x}_e(k))^T, \quad (14)$$

$$\tilde{x}_e(k) = x_0(k) - y_{actual}(k). \quad (15)$$

5. ADAPTIVE ANN CONTROLLER

With actor-critic based ANN control it is possible to optimize a long-term performance measure together with the minimization of a short-term error, as opposed to classical adaptive and ANN control. The additional signal used to evaluate the performance of the selected actions (actor) is generated by the critic. It is the role of the actor to map the environmental states to control signals, while it is the role of the critic to change that mapping in order to guarantee long-term stability and improve performance. To minimise the computational overhead, single hidden-layer networks will be used with output-layer only update-laws.

5.1 Strategic Utility Function

The strategic utility function, $Q(k)$, is defined as the long term performance measure of the system. It represents

the sum of all of the future system performance indices, Sarangapani (2006), where each index is evaluated once per control signal and determined by the binary utility function, $p(k) = [p_i(k)]_{i=1}^{m}$. As the tracking error $e_n(k)$ is unavailable at the k^{th} time instant, the modified tracking error $\hat{e}_n(k)$ will be used. The utility function is defined as

$$p_i(k) = \begin{cases} 0, & \text{if } \left|\hat{e}_n^i(k)\right| \leqslant c \\ 1, & \text{otherwise} \end{cases} \quad i = 1, 2, \ldots, m \tag{16}$$

with $c \in R^+$ the predefined performance threshold and $\hat{e}_n^i(k) \in R$ the i^{th} estimation error of the MIMO-vector. A binary reinforcement of "1" represents poor performance while a "0" indicates adequate tracking performance. Using the standard Bellman equation for dynamic programming and the alterations suggested by Sarangapani (2006), the long-term system-performance measure becomes

$$Q^T(k) = \alpha^N p(k+1) + \alpha^{N-1} p(k+2) + \cdots + \alpha^{k+1} p(N). \tag{17}$$

Using mathematical induction (17) reduces to

$$Q^T(k) = min_{u(k)} \left\{ \alpha Q(k-1) - \alpha^{N+1} p(k) \right\}. \tag{18}$$

In (18), $\alpha \in R$ is a design variable indicating the signal decay rate with $0 < \alpha < 1$. The decay rate forces the reinforcement signal back to zero if no additional penalties are incurred. N represents the stage number. Since the measure is similar to the Bellman equation, Si and Wang (2001), for dynamic programming it can be used as the only signal to tune the controller.

5.2 Critic Neural Network

Since $Q(k)$ cannot be determined at the k^{th} instant, an estimator is required to approximate the strategic utility function, known as the critic. By implementing a neural network it is possible to estimate the function $Q(k)$ if (18) is rewritten in the prediction error form

$$e_c(k) = \hat{Q}(k) - \alpha(\hat{Q}(k-1) - \alpha^N p(k)). \tag{19}$$

The estimation of the strategic utility function becomes

$$\hat{Q}(k) = \hat{w}_3^T(k)\phi_3(v_3^T x(k)) = \hat{w}_3^k \phi_3(k). \tag{20}$$

In these equations n_3 represents the number of hidden-layer nodes with the hidden- and output-layer weight matrices $v_3(k) \in R^{nm \times n_3}$ and $\hat{w}_3(k) \in R^{n_3 \times m}$ respectively. The subscript "c" indicates the error associated with the critic estimation. If the basis vector $\phi_3(v_3^T x(k))$ is selected large enough, the function approximation error approaches zero. Since no hidden-layer tuning is done the basis vector is simply written as $\phi_3(\hat{x}k)$. The choice of activation function, constrains the hidden-layer output to $\phi_3 \in (-1, 1)$ if it is of the tan-sigmoid form.

The objective function to be minimised by the critic is given in quadratic form, Sarangapani (2006),

$$E_c(k) = \frac{1}{2} e_c^T(k) e_c(k). \tag{21}$$

Using gradient based back-propagation, the weight adaptation rule becomes

$$\hat{w}_3(k+1) = \hat{w}_3(k) + \Delta\hat{w}_3(k), \tag{22}$$

where

$$\Delta\hat{w}_3(k) = \alpha_3 \left[-\frac{\partial E_c(k)}{\partial \hat{w}_3(k)} \right]. \tag{23}$$

The final update-law for the critic's output-layer weights is given by

$$\begin{aligned} \hat{w}_3(k+1) = \hat{w}_3(k) - \alpha_3 \phi_3(\hat{x}(k)) \\ \times (\hat{Q}(k) + \alpha^{N+1} p(k) - \alpha \hat{Q}(k-1))^T, \end{aligned} \tag{24}$$

where $\alpha_3 \in R$ is the adaptation gain for the critic ANN. If need be, the adaptation gain can be selected individually for each output-node to increase the overall performance. The critic network is tuned by its estimation of the reinforcement signal and the discounted values of its past output.

5.3 Actor Neural Network

In action-based adaptive control, the general purpose of the actor is the generation of control inputs based on the performance analysis supplied by the critic and/or other error metrics. When using neural networks, the actor learns a mapping policy from the environmental states to the control signals. Changes to the policy are made by the error metric, which increases controller performance. The tracking error is defined as the error between the actual model states $x_i(k)$ and the desired trajectory $y_d(k)$

$$e_i(k+1) = x_i(k+1) - y_d(k+i) \qquad i = 1, \ldots, n. \tag{25}$$

Combining the Brunovsky canonical form of (1) with the tracking error definition in (25), the n^{th} state error becomes

$$\begin{aligned} e_n(k+1) = f(x(k)) + g(x(k))u(k) + d'(k) \\ - y_d(k+n). \end{aligned} \tag{26}$$

Using (25), it is possible to define the desired auxiliary control variable in such a way that the smooth functions $f(x(k))$ and $g(x(k))$ describes the system fall away. If the input is selected as

$$v_d(k) = g^{-1}(x(k))(-f(x(k)) + y_d(k+n) + l_1 e_n(k)), \tag{27}$$

the state error (25) reduces to

$$e_n(k+1) = l_1 e_n(k) + d'(k), \tag{28}$$

which can be manipulated by the choice of l_1. The design matrix should be chosen such, that consecutive errors keep decreasing. External disturbances are represented by $d'(k)$ and will be assumed negligible during the design of the update law, but not during the stability proof. In theory it should be possible to implement the desired control action. Practically the smooth functions $f(x(k))$ and $g(x(k))$ are unknown and should be estimated. The desired auxiliary control signal $v_d(k)$ is approximated by the action neural network

$$\begin{aligned} v_d(k) &= w_2^T \phi_2(v_2^T s(k)) + \varepsilon_2(s(k)) \\ &= w_2^T \phi_2(s(k)) + \varepsilon_2(s(k)). \end{aligned} \tag{29}$$

With the input vector $s = [x^T(k), e_n(k)]^T \in R^{(n+1)m}$ and tan-sigmoid activation function, ϕ_2, serves as network basis $w_2^T \phi_2(s(k)) \in R^{n_2}$. Again, the approximation error can be made arbitrarily small by selecting a large enough basis. The ideal hidden- and output-layer weights are defined as $v_2 \in R^{(n+1)\times n_2}$ and $w_2 \in R^{n_2 \times m}$ respectively. Using the tracking error definition in (25) together with the error between the desired auxiliary control signal, $v_d(k)$, and the actual control signal, $v(k)$, it is possible to derive an update-law for the actor in terms of the reduced state error (28). The quadratic objective function to be minimised is defined in Sarangapani (2006) as:

$$E_a(k) = \frac{1}{2} e_a^T(k) e_a(k) \tag{30}$$

where $e_a(k)$ is a function similar to $e_c(k)$ including both the instantaneous error and the critic signal. By replacing the desired control signal, $v_d(k)$, with the actual control signal,$v(k)$ and the model states, $x(k)$ with their estimated states, $\hat{x}(k)$, it is possible to derive a tuning-law for the following actor neural network, Sarangapani (2006):

$$\begin{aligned} v(k) &= \hat{w}_2^T \phi_2(v_2^T \hat{s}(k)), \\ &= \hat{w}_2^T \phi_2(\hat{s}(k)). \end{aligned}$$

Using gradient based back-propagation as the tuning method, the weight adaptation rule is specified as:

$$\hat{w}_2(k+1) = \hat{w}_2(k) + \Delta\hat{w}_2(k), \tag{31}$$

where

$$\Delta\hat{w}_2(k) = \alpha_2 \left[-\frac{\partial E_a(k)}{\partial \hat{w}_2(k)} \right], \tag{32}$$

which in turn results

$$\begin{aligned} \hat{w}_2(k+1) = \hat{w}_2(k) - \alpha_2 \phi_2(\hat{s}(k))(e_n(k+1) \\ - l_1 e_n(k) - d_2(k) + \hat{Q}(k))^T. \end{aligned}$$

Replacing the states with their estimates also changes the tracking error $e_n(k)$ to the modified tracking error $\hat{e}_n$, which can be measured at the k^{th} time instant. In the ideal case the disturbance signal $d_2(k)$ is taken as zero to obtain the weight-update rule for the action neural network:

$$\begin{aligned} \hat{w}_2(k+1) = \hat{w}_2(k) - \alpha_2 \phi_2(\hat{s}(k))(\hat{e}_n(k+1) \\ - l_1 \hat{e}_n(k) + \hat{Q}(k))^T \end{aligned} \tag{33}$$

The activation function for the output-layer is selected to be linear. This presents a problem for real-world systems as the control signal can grow beyond the actual input limitations of the system. Actuator constraints need to be taken into account. The final actor output is defined as

$$u(k) = \begin{cases} v(k), & \text{if } |v(k)| \leqslant u_{max} \\ u_{max} sgn(v(k)), & \text{if } |v(k)| \geqslant u_{max} \end{cases} \tag{34}$$

where saturation limitations, u_{max} are accounted for. The addition of the magnitude-bound does not affect the update-law or the final Lyapunov theorem.

6. STABILITY

According to Igelnik and Pao (1995) the reconstruction error $\varepsilon(k)$ approaches zero if the hidden-layer is selected large enough. Assuming that the helicopter-system can be modelled by the Brunovksy canonical form, (5), the universal function approximation ability implies that an ideal network, with a minimum reconstruction error, exists. The ideal network can approximate the n^{th} state function. If it is possible to prove that the weights of the actual ANN will always converge to those of the ideal ANN, the system is considered stable. This is the purpose of the Lypunov function definition.

The Lypunov function includes the weight estimation errors $\tilde{w}_i(k), i = 1, 2, 3$ (of the observer, critic and actor), the state-estimation errors, $\tilde{x}_i(k)$, and the tracking errors $e_i(k)$ and $e_n(k)$. By proving that all errors always converge to zero, it is possible to prove system stability. The Lypunov function is selected to be positive-definite with its derivative being negative semi-definite if the following conditions, stated in (Sarangapani, 2006) are met:

$$0 < \alpha_1 \|\phi_1(\hat{z}_1(k))\|^2 < 1, \tag{35}$$

$$0 < \alpha_2 \|\phi_2(k)\|^2 < min\left(\frac{g_{min}}{g_{max}^2}, \frac{1}{g_{\min}} \right), \tag{36}$$

$$0 < \alpha_3 \|\phi_3(\hat{x}_1(k))\|^2 < 1, \tag{37}$$

$$0 < \alpha < \frac{\sqrt{2}}{2}. \tag{38}$$

The function derivative will always be non-positive if the approach presented in Sarangapani (2006) is followed. This concludes Lyapunov stability proof with the state estimation error, $\tilde{x}_i(k)$, the weight estimates, $\hat{w}_1(k), \hat{w}_2(k), \hat{w}_3(k)$ and the tracking error, $e_i(k)$, Uniformly Ultimate Bounded (UUB).

7. SIMULATION RESULTS

The following section provides results for the control-system. Both the observer and controller performance is illustrated independently. Parameter selection was done with a multi-objective genetic algorithm optimising tracking-performance. Selected parameters include: $\alpha, \alpha_1, \alpha_2, \alpha_3, l_1, l_2, n, n_1, n_2, n_3, N$, and c.

7.1 State-space model

Only a single channel is illustrated in Fig. 3 to show the learning capabilities of the ANN control structure. Initial oscillations are caused by poor estimations from the observer and not poor controller performance. As the observer error decreases exponentially, the system performance improves with time. The system remains stable for the duration of the simulation. Notice a reduction in overshoot magnitude as the actor learns how to control the system behaviour. Successive execution of the same manoeuvres are done with increased accuracy.

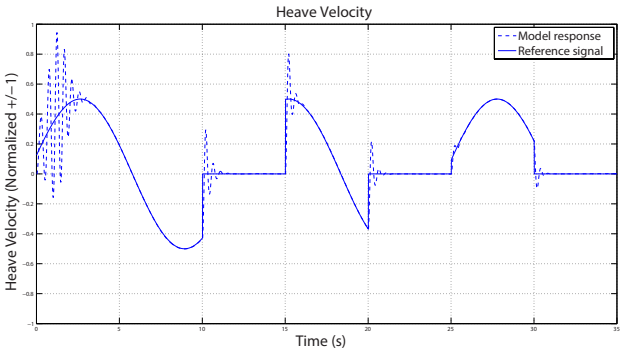

Fig. 3. Control-system performance plot

7.2 Actual helicopter data

Experimental results show that the controller performance is highly dependent on the observation signal. In order to illustrate the disturbance rejection capabilities of the ANN observer, it is implemented on actual flight data. For the purpose of the study, recorded data from the Rooivalk attack helicopter is used. Fig. 4 shows the tracking capabilities and disturbance rejection characteristics of the modified observer structure. No additional filters are applied to the data before use. It does however undergo

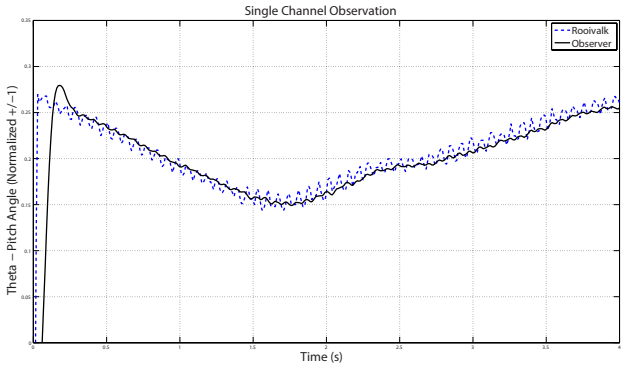

Fig. 4. State prediction and filtering of Rooivalk data

a normilisation procedure. When predicting 5 time-steps into the future, the estimated signal shows a 93.18% correlation (NRMSE) to the actual signal. Considering the presence of noise, the match is closer than the computed value.

8. CONCLUSION AND RECOMMENDATIONS

It is found that the series-parallel, on-line, observer structure demonstrates superior prediction capabilities when estimating over the same horizon. The estimated signal converges faster and predicts more accurately than previous observers of similar structure. Through the subtle alteration made to the training-law, genetic algorithm optimization can compute an optimal solution in fewer iterations.

It is possible to control a helicopter if the system can be represented in the reduced Brunovsky canonical form. Control was successfully implemented on a linear-model by Mettler et al. (1999), with the addition of non-linear properties in the form of a rate-limiter and artificial noise. It was found that the controller is highly dependent on the quality of observations and that the observer exhibits disturbance rejection capabilities. Through the Lypunov proof, it is shown that the system remains stable even if initialized at random with stability and performance increasing with time. The single-layer update-law allows the controller to operate in real-time.

If faster sampling rates are required, the neural networks can be operated on a parallel core system. This will supply the computational power needed for multiple hidden-layers and multi-layer tuning. An off-line training phase can also be utilized to increase initial convergence time. Future work should focus on more advanced stability proofs with the ultimate aim of providing satisfactory mathematical guarantees for industry implementation.

ACKNOWLEDGEMENTS

This work is based on the research supported in part by the National Research Foundation of South Africa (Grant specific unique reference number (UID: 80020); The Grant-holder acknowledges that opinions, findings and conclusions or recommendations expressed in any publication generated by the NRF supported research are that of the author(s), and that the NRF accepts no liability whatsoever in this regard. A special thanks is also extended to Denel Aviation and the Marenco Engineering Technologies Africa group for their technical support and use of their facilities.

REFERENCES

Abbeel, P., Coates, A., and Ng, a.Y. (2010). Autonomous Helicopter Aerobatics through Apprenticeship Learning. *The International Journal of Robotics Research*, 29(13), 1608–1639.

Abdollahi, F. (2011). Neural Networks Lecture 8 : Identification Using Neural Networks Identification Model Direct modeling Inverse Modeling. 1–32.

Demuth, H. Beale, M.H.M. (2013). *Neural Network Toolbox TM 7 User's Guide*. Natick, MA: The MathWorks, INC., 2013.

He, P. and Jagannathan, S. (2007). Reinforcement Learning Neural-Network-Based Controller for Nonlinear Discrete-Time Systems With Input Constraints. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 37(2), 425–436.

Igelnik, B. and Pao, Y.H. (1995). Stochastic choice of basis functions in adaptive function approximation and the functional-link net. *Neural Networks, IEEE Transactions on*, 6(6), 1320–1329.

Kendoul, F. (2012). Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. *Journal of Field Robotics*, 29(2), 315–378.

Khalil, H.K. (2001). *Nonlinear Systems (3rd Edition)*. Prentice Hall, 3 edition.

Landau, I.D., Lozano, R., M'Saad, M., and Karimi, A. (2011). *Adaptive Control*. Communications and Control Engineering. Springer London, London.

Mettler, B., Tischler, M.B., and Kanade, T. (1999). System identification of small-size unmanned helicopter dynamics. In *Annual Forum Proceedings- American Helicopter Society*, volume 2, 1706–1717.

Nodland, D., Zargarzadeh, H., and Jagannathan, S. (2013). Neural network-based optimal adaptive output feedback control of a helicopter uav. *IEEE Trans. Neural Netw. Learning Syst.*, 24(7), 1061–1073.

Panchal, G., Ganatra, A., and Kosta, Y. (2010). Searching Most Efficient Neural Network Architecture Using Akaike's Information Criterion (AIC). *International Journal of*, 1(5), 41–44.

Raimúndez, C., Camano, J., and Béjar, M. (2006). Application of Adaptive Neural-Network Control to a Scale 6DoF Helicopter.

Sarangapani, J. (2006). *Neural Network Control of Nonlinear Discrete-Time Systems*. Automation and Control Engineering. Taylor & Francis.

Sharma, M. and Verma, A. (2010). Adaptive observer based tracking control for a class of uncertain nonlinear systems with delayed states and input using self recurrent wavelet neural network. In *Advances in Computing, Control and Telecommunication Technologies (ACT), 2010 Second International Conference on*, 27–31.

Si, J. and Wang, Y.T. (2001). Online learning control by association and reinforcement. *Neural Networks, IEEE Transactions on*, 12(2), 264–276.

Valavanis, K.P. (2007). *Advances in Unmanned Aerial Vehicles*, volume 33 of *Intelligent Systems, Control and Automation: Science and Engineering*. Springer Netherlands, Dordrecht.