NONLINEAR VIRTUAL REFERENCE FEEDBACK TUNING
Application of Neural Networks to Direct Controller Design

Keywords: Virtual Reference Feedback Tuning, neural networks, experimental controller tuning, adaptive control.

Abstract: Virtual Reference Feedback Tuning (VRFT) is a direct controller design methodology which can be applied both in the linear and nonlinear case. In this work, neural network controllers have been designed following nonlinear VRFT principles, conforming what could be considered a particular scheme of direct neural Model Reference Control. The approach has been applied to a simulated crane example, proposing alternative block diagrams with extra inputs. In this example, the neural VRFT is compared to the linear one.

1 INTRODUCTION

Most of the available controller design methodologies are based on the availability of a model of the system to be controlled. This can be a handicap, as several reasons make this step very difficult in many cases: unstability of the plant, presence of nonlinearities or time-varying parameters, high order dynamics, etc. In this scenario direct controller design methodologies seam to be appealing, as they do not need a plant model to deal with the controller design problem. Within these approaches is the so called *Virtual Reference Feedback Tuning* (VRFT).

Virtual Reference Feedback Tuning (VRFT) is a model-free direct controller tuning methodology, introduced by (Campi et al., 2002) for the linear case and extended to nonlinear systems in (Campi and Savaresi, 2006). Enhancements and remarks to the basic setting were proposed in (Sala and Esparza, 2005; Sala, 2007). VRFT has proved to be a good option to design a controller for an unknown plant. It does not need a plant model and, in addition, this methodology is simple to apply, because it only needs a single open loop experiment or, possibly, a second one if the data are collected in closed loop and corrupted by significant noise (Campi et al., 2002).

The model-free nature of VRFT makes it appealing for practical cases, even if it is, from a theoretical point of view, approximate: a plant model is needed to propagate gradients in order to achieve unbiased convergence (Campi and Savaresi, 2006) if the controller parameterisation is not powerful enough. In fact, correct application of the methodology would require simultaneous plant and controller identification (Sala, 2007).

The objective of this contribution is to adapt the recent VRFT results to controllers incorporating neural networks. Prior experiments with nonlinear VRFT and neural networks are reported in (Previdi et al., 2004). However, the correct computation of the gradients requires backpropagation through time (Werbos, 1990), which was not carried out in the cited reference, which also used a very simple linear-in-parameter neural network with least-squares fit.

This paper discusses the computation of gradients and, additionally, shows the achieved improvements over the standard 1-degree of freedom control loop when auxiliary sensors are used. The methodology is tested by simulation of a crane model. An approximate linear model of the plant will be used to improve gradient computations, identified from the same available input-output data which will later be used to identify the controller.

The organisation of the paper is as follows: Section 2 presents a brief exposition of control structures using neural networks, with the objective of comparing them to VRFT approach. Then, VRFT principles, both in the linear and nonlinear case are ex-

posed in section 3. Section 4 is the main contribution, as it presents a nonlinear VRFT approach using neural networks, offering a simulated application example of two particular neural network controller structures. This contribution ends with the conclusion drawn from this work.

2 NEURAL NETWORKS FOR CONTROL

Neural networks have been used not only for identifying nonlinear plants but also for controlling them. As a neural network is a function approximator, it can also be trained to map the loop error into the needed control action.

In the literature (see for example (Hagan et al., 2002; Kasparian and Batur, 1998; Narendra and Parthasarathy, 1990)), several are the structures used to control a nonlinear system by means of neural networks. Two of the most popular ones are the *Direct Inverse Control* (Fig. 1) and the *Model Reference Control* (Fig. 2) using neural networks. Both structures, as will be useful for the subsequent exposition, will be briefly reviewed next.

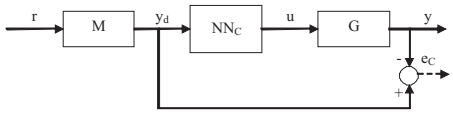

Figure 1: Direct Inverse Control using neural networks

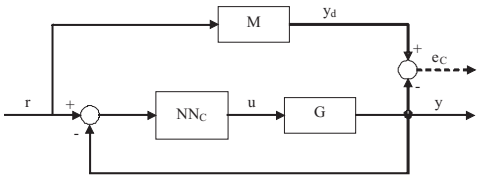

Figure 2: Model Reference Control using neural networks

2.1 Neural Direct Inverse Control

Figure 1 shows the structure of the so called Direct Inverse Control using neural networks. The neural network NN_C is trained to model the inverse plant dynamics, as seen in Fig. 3, using the error signal e_{IM}, which stands for the 'inverse model error'. Once this is done, NN_C is placed in series with the plant, being its input the desired behavior y_d (see Fig. 1). The trained network can also operate in an adaptive way to account for possible disturbances. In this case, the error signal e_C is used to adapt online the controller.

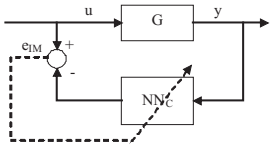

Figure 3: Training the neural network NN_C

This scheme has several disadvantages. Of course, the first one is that it can not be used with open-loop unstable plants or with non-invertible ones. Furthermore, if the nonadaptive scheme is the one to be used, as the control is carried out in open loop, perturbances or undermodelling will degrade the system's performance.

2.2 Neural Model Reference Adaptive Control

Neural Model Reference Adaptive Control is a particular case of Model Reference Adaptive Control (MRAC). Its structure corresponds to that depicted in Fig. 2, where the signal e_C is used to train or adapt online the weights of the controller NN_C. Two are the approaches used to design a MRAC control for an unknown plant: Direct and Indirect Control.

Direct Control: This procedure aims at designing a controller without having a plant model. As the knowledge of the plant is needed in order to train the neural network which corresponds to the controller (*i.e.* NN_C), until present, no method has been proposed to deal with this problem.

Indirect Control: This approach uses two neural networks: one for modelling the plant dynamics (NN_M), and another one trained to control the real plant so as its behavior is as close as possible to the reference model (NN_C). This scheme is represented in Fig. 4. As a first step, the neural network NN_M is trained to approximate the plant input/output relation using the signal e_M. This is usually done offline, using a batch of data gathered from the plant in open loop. Once the model NN_M is trained, it is used to train the network NN_C which will act as the controller. The model NN_M is used instead of the real plant's output because the real plant is unknown, so backpropagation algorithms can not be used. In this way, the control error e_C is calculated as the difference between the desired reference model output y_d and $\hat{y}$, which is the closed loop predicted output. Then, as NN_M is fixed, its derivatives with respect to any parameter are known and easy to compute.

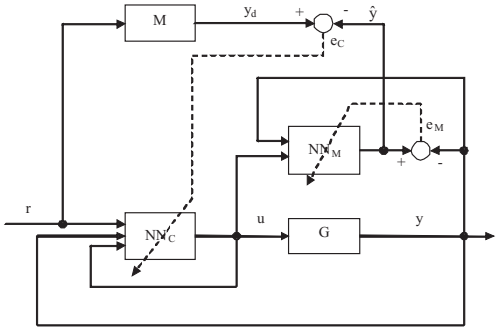

Figure 4: Indirect MRAC

3 VIRTUAL REFERENCE FEEDBACK TUNING

Let us denote by P the unknown plant to be controlled. The objective is to design a controller which attains a closed-loop behaviour as close as possible to a reference model M, chosen by the designer. A first open-loop experiment carried out on it gives a set of input/output data $\{u_{ex}, y_{ex}\}$. Now let us consider a *virtual* closed-loop (Fig. 5) with an unknown controller C whose output is precisely u_{ex} (and therefore, the plant's output will be y_{ex}). Then, the signals r_v and e_v in Fig. 5, called *virtual reference* and *virtual error*, respectively, are calculated as follows:

$$r_v = M^{-1} y_{ex} \quad (1)$$
$$e_v = (M^{-1} - 1) y_{ex} \quad (2)$$

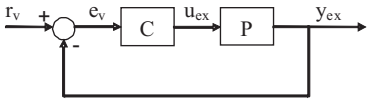

Figure 5: *Virtual* closed loop

Within this environment, the controller design reduces to an identification problem between the signals e_v and u_{ex}, considering a parameterised controller $u = C(\theta, e)$, where $\theta \in \mathbb{R}^p$ stands for the vector of parameters, being p its length. On the sequel, the parameterised controller will be denoted by $C_\theta(e)$. The ideal controller C^* fulfills $u_{ex} = C^*(e_v)$, but it will be, possibly, nonlinear and high-order.

The cost index to minimise will be the Euclidean norm (denoted by $\|\cdot\|$) of the (possibly filtered) difference between the output of the loop when a particular θ is used, y_θ, and the ideal one $Mr_v = y_{ex}$:

$$J = \frac{1}{2}\|\varepsilon\|^2 = \frac{1}{2}\|F(y_\theta - y_{ex})\|^2 \quad (3)$$

In expression above F is a frequency weighting filter chosen by the user. In a closed-loop setting, y_θ is defined as:

$$y_\theta = P(C_\theta(e)) \quad (4)$$

In this expression P is the (unknown) plant, $e = r_v - y_\theta$ is the (non-virtual) tracking error. Then, the cost index (3) can be expressed as:

$$J = \frac{1}{2}\|F(P(C_\theta(e)) - y_{ex})\|^2 \quad (5)$$

In the remainder, filter F will be considered the identity, without loss of generality.

As the plant P is not known, it is not possible to minimise (5). So, the VRFT methodology proposes the following data-based cost index:

$$J_{VRFT} = \frac{1}{2}\|L(C_\theta(e_v) - u_{ex})\|^2 \quad (6)$$

where L is a filter designed so as to make the solution to the minimization of (6) as close as possible to that of (5). Note that (6) only uses available data u_{ex} and e_v. In the linear case, (Campi et al., 2002) show that the difference between J and J_{VRFT} is minimised, under some assumptions, by choosing the filter L as:

$$L = M(I - M)T_u^{-1} \quad (7)$$

where T_u is a filter such that $|T_u|^2 = \Phi_u$ (Φ_u is the power spectral density of $u_{ex}(t)$). Linearity of plants and controllers, when applicable, will refer to the input signals and not to the parameterisations θ.

Let us now consider the nonlinear VRFT approach (Campi and Savaresi, 2006). In order to minimise a cost index, under differentiability assumptions, most optimization algorithms use its gradient with respect to its parameters. In particular, the gradients of J and J_{VRFT} (considering $F = I$ and the availability of $\{u_{ex}, y_{ex}\}$) are given by:

$$\frac{dJ}{d\theta} = < y_\theta - y_{ex}, \frac{dy_\theta}{d\theta} > \quad (8)$$
$$\frac{dJ_{VRFT}}{d\theta} = < L(C_\theta(e_v) - u_{ex}), L\frac{\partial C_\theta}{\partial \theta} > \quad (9)$$

where $< \cdot, \cdot >$ denotes the scalar product and $\frac{\partial C_\theta}{\partial \theta}$ is the derivative of the controller with respect to its parameters considering the remaining inputs constant.

On the sequel, operations over the gradient (8) will be carried out so as to express an approximation of it as a function of the available signals $\{u_{ex}, e_v, y_{ex}\}$. That is, the value of the filter L will be found out so as to make the index J_{VRFT} as close as possible to J.

Neither of both sides of the scalar product of expression (8) are computable, as the plant model P is not known. So it is necessary to express this gradient

as a function of both the actually recorded signals and the virtual ones. The derivative of y_θ with respect to the controller parameters can be expressed as:

$$\frac{dy_\theta}{d\theta} = \bar{P}\frac{du_\theta}{d\theta} \tag{10}$$

where $\bar{P} = \left.\frac{\partial y_\theta}{\partial u}\right|_u$ is a (possibly time-variant) plant linearisation and u_θ is the output of the controller to be designed, where its dependence on the vector of parameters θ has been emphasised.

If plants and controllers are recurrent, gradient propagation in time (Werbos, 1990) must be used. For instance, given, say

$$u_{\theta,k} = C(\theta, e_k, e_{k-1}, \ldots, e_{k-m}, u_{k-1}, \ldots, u_{k-n})$$

we would obtain, applying the chain rule:

$$\begin{aligned}\frac{du_{\theta,k}}{d\theta} &= \frac{\partial C_\theta}{\partial \theta} - \sum_{j=0}^{m}\frac{\partial C_\theta}{\partial e_{k-j}}\frac{dy_{\theta,k-j}}{d\theta} \\ &+ \sum_{j=1}^{n}\frac{\partial C_\theta}{\partial u_{\theta,k-j}}\frac{du_{\theta,k-j}}{d\theta}\end{aligned} \tag{11}$$

which is a recurrent equation (and time-variant if C is nonlinear in u or e). In previous expression, it has been considered that the error signal is defined as $e_i = r_i - y_{\theta,i}$ and thus its derivative with respect to the vector of parameters is $\frac{de_i}{d\theta} = -\frac{dy_{\theta,i}}{d\theta}$. The resulting sequence of $\frac{du_{\theta,k}}{d\theta}$ in (11) must be the input in (10) to the recurrence equations defining $\bar{P}$ in order to correctly compute the gradient of the closed-loop output with respect to the parameters. Hence, (10) and (11) are the sensitivity equations defining a recurrent linear time-variant system, to be denoted by L, whose input is $\frac{\partial C_\theta}{\partial \theta}$, *i.e.*

$$\frac{dy_\theta}{d\theta} = L\frac{\partial C_\theta}{\partial \theta} \tag{12}$$

Figure 6 represents such sensitivity equations computed at the instant k. In such figure, both the block diagram input $\frac{\partial C_\theta}{\partial \theta}$ and the output $\frac{dy_\theta}{d\theta}$ are vectors the size of which is the number of parameters θ. The filter L is also depicted. As it can be seen, this filter is time-variant, so it must be updated at each k ($k \in [1, N]$, being N the data length).

Going back to the problem of computing the gradient of the index J, the left part of expression (8), $(y_\theta - y_{ex})$, can also be approximated to an expression computable using the available signals. Considering the existence of θ^*, which is the vector of parameters which makes $u_{ex} = C(\theta^*, e_v, u_{ex})$, for θ close to θ^*, the following approximation can be considered:

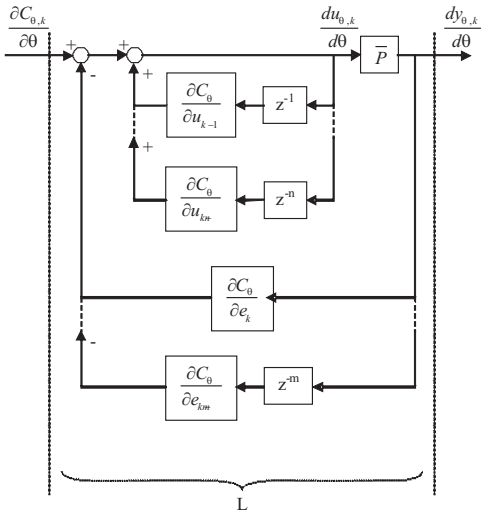

Figure 6: Sensitivity block diagram

$$\begin{aligned}y_\theta - y_{ex} &\simeq \frac{dy_\theta}{d\theta}(\theta - \theta^*) = \\ L\frac{\partial C}{\partial \theta}(\theta - \theta^*) &\simeq L(C(\theta, e_v) - u_{ex})\end{aligned} \tag{13}$$

In summary, the gradient of J in (8) can be estimated by obtaining the vectors $(C_\theta(e_v) - u_{ex})$ and $\frac{\partial C_\theta}{\partial \theta}$, filtering them by L previously defined and then computing the inner product in (9).

Before closing this section, the open loop controller implementation will be considered. In this case, the *virtual loop* is an open loop one (Fig. 7), where r_v is the virtual reference, calculated using the same expression as in (1). This structure could be called *Virtual Reference Feedforward Tuning* as there is no feedback in the loop. In this case, the controller must be directly identified from r_v and u_{ex}.

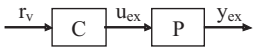

Figure 7: Virtual Reference Feedforward Tuning

In this case, when operating to compute the gradient of J defined in (8), a different expression for $\frac{du_{\theta,k}}{d\theta}$ than that in (11) is obtained, as the controller input r_v does not depend on the vector of parameters θ. The correct expression for the open loop case is:

$$\frac{du_{\theta,k}}{d\theta} = \frac{\partial C_\theta}{\partial \theta} + \sum_{j=1}^{n}\frac{\partial C_\theta}{\partial u_{\theta,k-j}}\frac{du_{\theta,k-j}}{d\theta} \tag{14}$$

Figure 8 displays the sensitivity equations in open loop case, as well as the computation of the filter L,

necessary to approximate the minimization of both indexes J and J_{VRFT}.

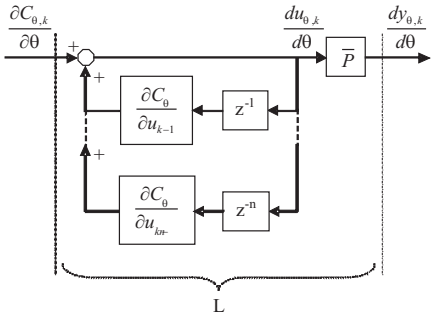

Figure 8: Sensitivity block diagram in open loop operation

4 NEURAL VIRTUAL REFERENCE FEEDBACK TUNING

In this section we will consider the nonlinear VRFT, using neural networks for the controller. The plant could be either linear or nonlinear with smooth nonlinearities. The objective is to design a controller which minimises the cost index J defined in (5) through the minimisation of the data-based index J_{VRFT} in (6). Then when computing the gradient of J_{VRFT}, if the controller is a neural network, its derivatives with respect to θ, u_{k-i} and e_{k-i} are easy to obtain. Applying VRFT principles, two are the possible schemes to be used attending to the control loop structure: open and closed loop operation.

Virtual Reference Feedforward Tuning. This is the basic scheme and involves a neural controller placed in open loop (see Fig. 7, where the controller C is a neural network). This neural network is trained to approximate the dynamics of the plant inverse multiplied by the reference model. Therefore, as it can be observed, this approach is very similar to the *Neural Direct Inverse Control* depicted in Fig. 1, suffering from the same disagvantadges pointed out in section 2.

As explained before, the sensitivity block diagram which defines the filter L which must be used in this case is the one depicted in Fig. 8. This block diagram is used both being $(C(\theta,e_v)-u_{ex})$ and $\frac{\partial C_\theta}{\partial \theta}$ its input. It should be noted that L must be updated at each time instant.

Virtual Reference Feedback Tuning. If we consider closed loop operation, then the resulting scheme can be properly called VRFT. Fig. 6 shows the sensitivity block diagram which is used to compute the time-varying filter L for the closed loop case (derived so as to calculate the gradient of J_{VRFT}). VRFT scheme could be considered comparable to the class of Direct *Neural Model Reference Control*, which is schematised in Fig. 2. In subsection 2.2 the *adaptive* scheme was presented, pointing out the two different existing approaches: direct and indirect. The proposed scheme can be considered a *Direct Non-adaptive* Model Reference Control, as it does not need an accurate and possibly non-linear and time-variant plant model to design a controller which aims at achieving a closed loop performance as close as possible to the reference model M.

Observing Figures 6 and 8, it is clear that a linearised plant model $\bar{P}$ is needed so as to build the filter L. So, it is necessary to have some information about the plant dynamics, but there is no need for this information to be very accurate. Therefore, a linear plant model obtained from either first principles modelling or linear identification should suffice.

4.1 Neural VRFT applied to a crane model. Theoretical concepts

The expressions derived in previous section can be applied to any controller being linear or nonlinear, and with any (possibly recurrent) structure agreeing with $u_k = C(\theta, e_k, \dots, u_{k-1}, \dots)$.

In (Campi et al., 2002; Campi and Savaresi, 2006; Sala, 2007), some approximations are carried out, in order to use a time-invariant transfer function L' and make the minimisation process easier. However, in order to obtain more accurate results, in this work the implemented filter has been the filter one L, as the partial derivatives required in (11) or (14) are easily computed for neural networks. As the objective of VRFT is to diminish, if possible, the need of elaborated plant models, $\bar{P}$ in (10) will, however, be considered a linear time-invariant system, identified from experimental data via standard output-error algorithms.

In this example, neural networks have been used to control a crane model under closed loop operation. This plant has two measured outputs: the hanging mass horizontal position (y_1) and its angle with respect to the vertical axis which passes through the center of the cart (y_2) (see Fig. 9). The objective is to control y_1. For the controller, two topologies have been used: $C_1 = NN(\theta, e, u)$ and $C_2 = NN(\theta, e, f(y_2), u)$. The first one fits the controller-class considered in previous section, so all the derived expressions hold. The controller C_2 is more powerful, as it has as an extra input a function of the system's output y_2. The extra input, later denoted as $s = f(y_2)$, changes the block diagram, hence it must be explic-

itly considered when computing the derivatives $\frac{du_\theta}{d\theta}$. The extra input will provide better performance, when compared to C_1, as discussed in next section.

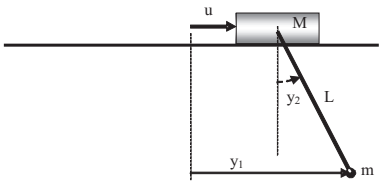

Figure 9: Structure of the crane system

To compute the gradient J_{VRFT} for the controller structure C_2, and following the same reasoning as in the previous section, equation (10) gets converted into:

$$\begin{pmatrix} \frac{dy_1}{d\theta} \\ \frac{dy_2}{d\theta} \end{pmatrix} = \begin{pmatrix} \bar{P}_1 \\ \bar{P}_2 \end{pmatrix} \frac{du}{d\theta} \tag{15}$$

where $\bar{P}_1 = \left.\frac{\partial y_1}{\partial u}\right|_u$ and $\bar{P}_2 = \left.\frac{\partial y_2}{\partial u}\right|_u$. In the same way, equation (11) turns into:

$$\frac{du_k}{d\theta} = \frac{\partial C}{\partial \theta} - \sum_{j=0}^{m} \frac{\partial C}{\partial e_{k-j}} \frac{dy_{1,k-j}}{\partial \theta} + \sum_{j=1}^{n} \frac{\partial C}{\partial u_{k-j}} \frac{du_{k-j}}{d\theta} + \sum_{j=0}^{p} \frac{\partial C}{\partial s_{k-j}} \frac{\partial s_{k-j}}{\partial y_2} \frac{dy_{2,k-j}}{d\theta} \tag{16}$$

Here the subindex θ used to emphasise the dependence of the signals on the vector of parameters has been omitted, for the sake of clearness.

The neural networks to be used are made up of two layers: the first one has four neurons (one linear and three with hyperbolic tangent activation function); the second layer has only one linear neuron. A bias input to each neuron is also present. For the controllers of class C_1, the network considered is the one depicted in Fig. 10. In this figure, the inputs $\bar{u}$ and $\bar{e}$ stand for vectors conforming a delay-line input: if the controller order considered is n, at a time instant k, $\bar{u}_k = [u_{k-1}, \ldots, u_{k-n}]$ and $\bar{e}_k = [e_k, \ldots, e_{k-n+1}]$. In the same way, the network used for the C_2-class controllers is the one in Fig. 11, where $f(y_2) = sin(y_2)$ has been added as a new network's input.

4.2 Neural VRFT applied to a crane model. Simulation results

As above mentioned, neural VRFT is applied to a crane model, depicted in Fig. 9, where the plant's input (u) and outputs (y_1 and y_2) are indicated. A first-principle model for simulation was found in (Butler et al., 1991), being the parameters values:

- Mass of the cart (M): 2 Kg.

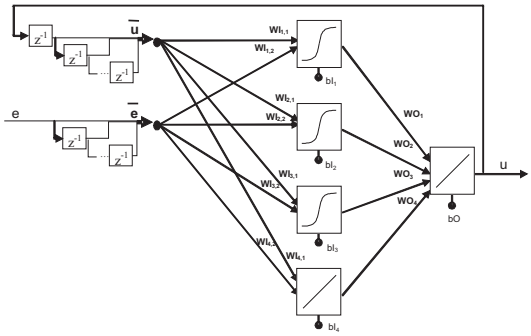

Figure 10: Neural network used for controllers of class C_1

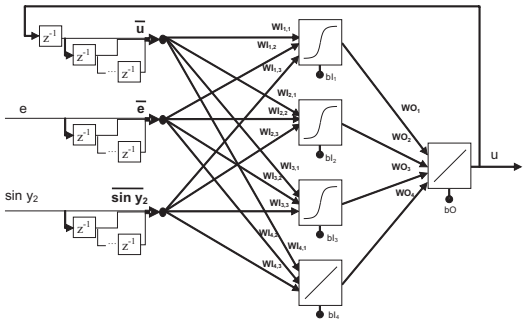

Figure 11: Neural network used for controllers of class C_2

- Hanging mass (m): 1 Kg.
- Length of the joining bar (L): 0.5 m.
- Linear friction coefficient (fc): 0.05 Ns/m.
- Angular fiction coefficient (fp): 0.01 Ns.

Using the neural network controller structure depicted either in Fig. 10 or 11, the control action at an instant k can be computed as:

$$u_k = \sum_{i=1}^{4} WO_i F_i \left(\sum_{j=1}^{n \times m} WI_{ij} x_k + bI_i \right) + bO \tag{17}$$

In the above expression, F_i is the hyperbolic tangent for $i = \{1,2,3\}$ and identity for $i = 4$ (linear neuron), n is the number of delays in the inputs (the same value for all the inputs has been considered), $m = 2$ for the neural network structure of Fig. 10 and $m = 3$ for the one in Fig. 11. In addition, x_k is the input vector, constructed as:

$$x_k = \begin{bmatrix} u_{k-1} & \ldots & u_{k-n} & e_k & \ldots & e_{k-n+1} \end{bmatrix}$$

for $m = 2$. For $m = 3$, this expression turns into:

$$x_k = \begin{bmatrix} u_{k-1} & \ldots & u_{k-n} & e_k & \ldots & e_{k-n+1} \\ & & s_k & \ldots & s_{k-n+1} & \end{bmatrix}$$

where $s_k = sin(y_{2k})$, *i.e.*, the sine of the angle y_2 at a time instant k.

Finally, in expression (17) the indexes i and j stand for the neuron number and the input position in x_k. Therefore, denoting by N the number of neurons ($N = 4$ in this example), the vector of parameters has $(n \times m + 2)N + 1$ elements, and is defined as:

$$\theta = \left[\begin{array}{cccccccccc} WI_{1,1} & \ldots & WI_{N,1} & \ldots & WI_{1,n\times m} & \ldots & WI_{N,n\times m} \\ bI_1 & \ldots & bI_N & WO_1 & \ldots & WO_N & bO \end{array} \right]^T \quad (18)$$

4.2.1 Simulation results

The objective of the application is to design a controller for the crane system in such a way that the controlled output y_1 follows as closer as possible a reference that goes from 0 to 1 m following a ramp with different slopes: from 0.04 m/s to 1 m/s.

First of all, the linear version of VRFT has been applied, as this particular system presents a smooth nonlinearity which allows linear controllers to be used for low angles (slow cart speed). Then, in order to compare and to improve the closed-loop tracking performance, both neural network structures (Figs. 10 and 11) have been considered as the controller's class to be used when applying nonlinear VRFT.

As the main nonlinearity in the system equations comes from trigonometric expressions depending on the angular position and speed, two situations have been envisaged:

1. A first simulation tries to adjust VRFT controllers (linear and neural) based on a set of open-loop data where the angular displacements are small.
2. A second simulation scenario uses open-loop data where angular displacements are significantly higher, due to a larger input amplitude.

Intuitively, it is expected that linear and nonlinear VRFT controllers behave similarly in the first case, but nonlinear ones improve in the second one. Let us discuss each of the simulations.

Low-amplitude training data. A first open-loop experiment is carried out with a white noise input in a range of ± 0.1 N. The angle variation is very low, so the nonlinear effect is not very present.

A linear 6^{th} order controller, as well as a neuronal one (type C_1), also with 6^{th} order delay-line inputs, are trained from the same data set. For the slowest reference (slope of 0.04 m/s during the transient), both linear and neural network controllers work well and produce a very similar output. The linear controller produces a better performance but a bigger oscillation of the pendulum (Fig. 12); it seems to have totally canceled the weakly damped mode. As long the reference's slope increases, the linear controller provides a better tracking performance (the more reduced number of parameters seems to make learning more effective) until a slope of 1 m/s, when it makes the loop unstable, whereas the one using a neural network provides the response in Fig. 13.

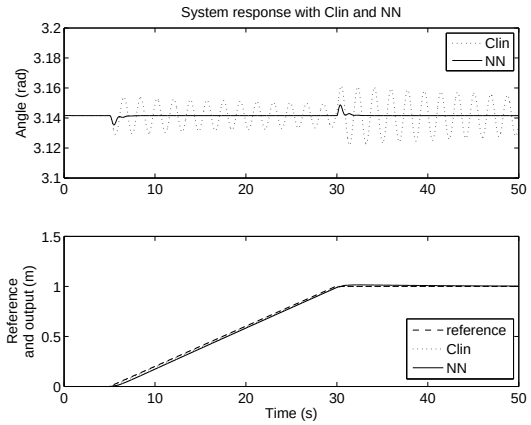

Figure 12: Comparison of plant behaviour with linear and neural network controllers (slow setpoint change speed)

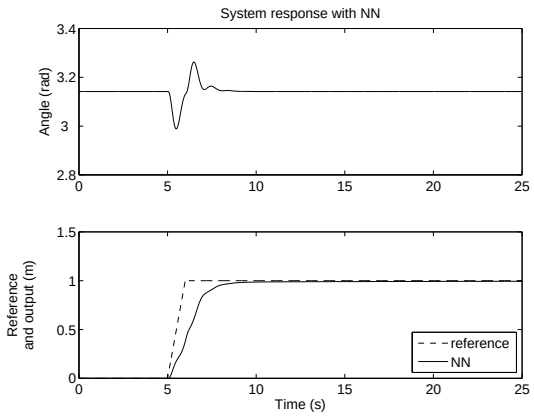

Figure 13: Plant behaviour with the neural network controller for a slope of 1 m/s in setpoint change

In order to increase the performance, two 4^{th}-order controllers are designed, using the closed-loop error and the sine of the bar's angle as inputs (the number of parameters is, hence, the same). The linear controller still produces an unstable loop, while the neural network one does not improve significantly over the one in Fig. 13 (results not shown for brevity).

Higher-amplitude training data. For comparison, the signals recorded at a second open-loop experiment, with a white noise input in a range of ± 1 N, are used to design linear and neural network controllers, using either only the closed-loop error or this error together with the sine of the pendulum's angle as inputs. All these controllers resulted in unstable loops, except the 4^{th}-order neural network one with

error and angle feedback (class C_2). Figure 14 shows the closed-loop behaviour when using this neural network controller for a fast-varying reference of slope 1.

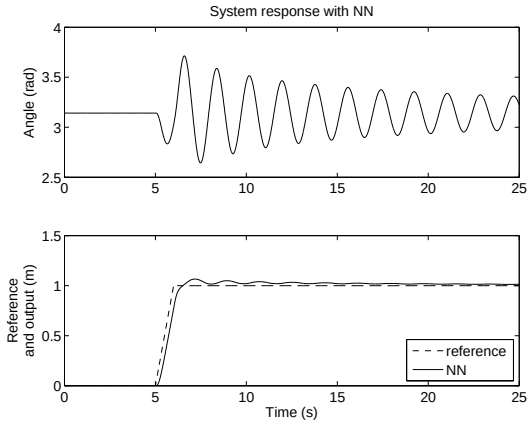

Figure 14: Closed-loop response with the neural network controller designed with 2 sensors and high-amplitude data

Note: the linear controllers have been identified using output error algorithms available in Matlab®'s Identification Toolbox. The neural network parameters have been adjusted using the Levenberg-Marquardt optimisation algorithm, applied to minimise the index J by means of the gradient expressions derived in previous sections.

5 CONCLUSIONS

This paper compares 'classical' neural network control structures with nonlinear VRFT ones. In particular, and due to its undeniable advantages, the closed loop approach is mainly tackled. Then, it is shown that successful application of the VRFT paradigm to neural controllers is possible, by propagating gradients through time using an approximate linear model of the plant. Such model is instrumental, only used in a filtering stage and its accuracy becomes less relevant as the parameterisation of the controller is more flexible allowing for a lower minimum approximation error.

Simulations show that, under demanding specifications, linear VRFT controllers yield unstable loops while nonlinear neural ones plus additional sensory feedback provide a satisfactory response. The results were achieved without the need of any nonlinear plant modelling or identification.

ACKNOWLEDGEMENTS

REFERENCES

Butler, H., Honderd, G., and Van Amerongen, J. (1991). Model reference adaptive control of a gantry crane scale model. *IEEE Control Systems Magazine*, 11(1):57–62.

Campi, M. and Savaresi, S. (2006). Direct nonlinear control design: the Virtual Reference Feedback Tuning (VRFT) approach. *IEEE Transactions on Automatic Control*, 51(1):14–27.

Campi, M. C., Lecchini, A., and Savaresi, S. M. (2002). Virtual Reference Feedback Tuning: a direct method for the design of feedback controllers. *Automatica*, 38:1337–1346.

Hagan, M., Demuth, H., and Jesús, O. D. (2002). An introduction to the use of neural networks in control systems. *International Journal of Robust and Nonlinear Control*, 12:959–985.

Kasparian, V. and Batur, C. (1998). Model reference based neural network adaptive controller. *ISA Transactions*, 37:21–39.

Narendra, S. and Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4–27.

Previdi, F., Schauer, T., Savaresi, S., and Hunt, K. (2004). Data-driven control design for neuroprotheses: a Virtual Reference Feedback Tuning (VRFT) approach. *IEEE Transactions on Control Systems Technology*, 12(1):176–182.

Sala, A. (2007). Integrating Virtual Reference Feedback Tuning into a unified closed-loop identification framework. *Automatica*, 43(1):178–183.

Sala, A. and Esparza, A. (2005). Extensions to "Virtual Reference Feedback Tunning: A Direct Method for the Design of Feedback Controllers". *Automatica*, 41(8):1473–1476.

Werbos, P. (1990). Backpropagation through time: What it does and how to do it. *Proc. of IEEE*, 78(10):1550–1560.