Esparza Peidro, A.; Sala, A.; Albertos Pérez, P. (2011). Neural networks in virtual reference tuning. Engineering Applications of Artificial Intelligence. 24(6):983-995. doi:10.1016/j.engappai.2011.04.003.

Neural Networks in Virtual Reference Tuning

Alicia Esparza*, Antonio Sala, Pedro Albertos

Dept. Systems Engineering and Control. Universidad Politécnica de Valencia. Camino de Vera s/n, 46022 Valencia, SPAIN

Abstract

This paper discusses the application of the Virtual Reference Tuning (VRT) techniques to tune neural controllers from experimental input-ouput data, by particularising nonlinear VRT and suitably computing gradients backpropagating in time. The flexibility of gradient computation with neural networks also allows alternative block diagrams with extra inputs to be considered. The neural approach to VRT in a closed loop setup is compared to the linear VRFT one in a simulated crane example.

Key words:
Neural networks, Virtual Reference Feedback Tuning, back propagation through time, model reference control, data-based controller tuning, direct controller design

1. Introduction

Current industrial development imposes an increasing demand of advanced control techniques which guarantee processes optimal working. Most of the available controlling algorithms are based on the existence of a model of the plant to be controlled. In these cases, the accuracy of this model is a deciding factor in the final performance. Many industrial processes are nonlinear and data are corrupted by significant noise; these issues usually hinder the modelling process both in data-based identification and in first-principle modelling.

The delivery of a reliable mathematical representation of the plant behaviour, which accurately models the important dynamics around the operating range, is a difficult task. In some cases, the system can be represented by a linear model, as complex as the controller design methodology needs. The term *Identification for Control* (Gevers, 1993; Van den Hof and Schrama, 1995) arose to deal with several issues concerning identification as a tool for control design: which is the frequency range of interest, what is the optimal complexity of the model, which is the optimal experiment (open/closed loop, on-line/off-line, selection of plant input), etc. This is also a field of active research at present (see, for example (Gevers, 2005; Hjalmarsson, 2005; Bombois et al., 2006; Hildebrand and Solari, 2007)). However, most of the above contributions are focused on linear systems, so they may fail when the issue is to obtain a model for a nonlinear plant.

Apart from the above difficulties in plain identification, using arbitrary nonlinear models to compute controllers might also encounter difficulties unless they are expressed in particular canonical forms.

Neural networks (Narendra and Parthasarathy, 1990; Hagan et al., 2002; Hunt et al., 1992) are widely used in industrial applications both for identification (Gregorcic and Lightbody, 2008) and control purposes (Kumar et al., 2009; Uraikul et al., 2007; Yue et al., 2007; Li and Deng, 2006; Haber and Alique, 2004). Neural networks have the ability to approximate complex non-linear relationships without prior knowledge of the model structure (black-box models), what makes them an attractive alternative to the classical modeling and control techniques.

Given the above-mentioned difficulties in obtaining a suitable process model, plus the difficulties of finding a nonlinear controller for the found model, a tempting alternative is to use the available experimental data to *directly* tune some controller parameters. This gives rise to the *Direct Data-Based Controller Design* approaches. Some of the most popular ones are the *Iterative Feedback Tuning* (IFT) for linear systems (Hjalmarsson et al., 1998) or nonlinear ones (Sjöberg et al., 2003), the *Correlation-based tuning* (Miskovic et al., 2003; Karimi et al., 2007) and the *Virtual Reference Feedback Tuning* (VRFT).

Virtual Reference Feedback Tuning (VRFT) is a model-free one-shot direct controller tuning methodology, introduced by Campi et al. (2002) for the linear case and extended to nonlinear systems in (Campi and Savaresi, 2006). It is 'model free' because it does not need any mathematical description of the plant to be controlled (under some assumptions and approximations). It is 'one-shot' because it can be applied using a single set of data generated by the plant (a second one is needed if the data are collected in closed loop and corrupted by significant noise, see (Campi et al., 2002)), with no need for specific experiments or iterations (which makes VRFT different and easier to apply than IFT). Enhancements and remarks to the basic setting were proposed in (Sala and Esparza, 2005a; Sala, 2007). If several experiments are possible, performance-improving iterations can be set up (Campi and Savaresi, 2006; Sala, 2007).

*Corresponding author.
Email addresses: alespei@isa.upv.es (Alicia Esparza), asala@isa.upv.es (Antonio Sala), pedro@aii.upv.es (Pedro Albertos)

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). However, as models are only an instrument to compute gradients, if the nonlinear controller parameterisation is flexible enough, large modelling errors will still lead to almost-optimal controller parameters.

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 considers a generic *Virtual Reference Tuning* (VRT) methodology, which includes open loop (*Virtual Reference Feedforward Tuning* or VRFFT) and closed loop (*Virtual Reference Feedback Tuning* or VRFT) setups. It discusses the computation of some required gradients via backpropagation throught time 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 on 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 structure of the paper is as follows: Section 2 presents a brief exposition of control structures using neural networks, with the objective of comparing them with VRT approach. Then, VRT principles, both in the open and closed loop setups are exposed in section 3. Sections 4 and 5 are the main contributions, as they present a nonlinear VRT approach using neural networks, developing a simulated application example using two particular neural network controller structures, one of them with an additional sensor. The contribution ends with the conclusions drawn from this work.

2. Preliminaries: Neural Networks for control

The main property of neural networks is their ability to approximate complex nonlinear relationships without prior knowledge of the model structure. Fig. 1 shows the use of neural networks as function approximators. The objective is to adjust the network's parameters in such a way that, using the same input, its output (predicted output) is as close as possible to the output of the unknown function to be modelled. Neural networks have been used not only for identifying nonlinear plants but also for controlling them.

There are two basic network architectures, namely the feedforward networks (FFNN), also called multilayer networks, and the recurrent ones (RNN) (Narendra and Parthasarathy, 1990; Liu, 2001; Hagan et al., 2002). In the former the signals are

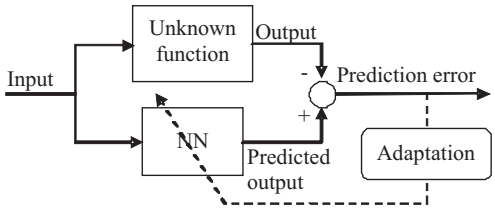


Figure 1: Neural network as a function approximator

transmitted in one direction throughout all the layers, from the network input towards its output. The latter have some feedback connections between its layers. FFNN represent static nonlinear maps between signals. They are easy to train, as standard backpropagation algorithms can be used (Werbos, 1990). FFNN have been successful in many applications as, for example, pattern recognition (Bishop, 1995).

RNN (Hopfield, 1982) are inherently dynamic networks and therefore they can be thought to approximate in a more natural way the relationship among signals in dynamic feedback systems. However, their main drawback is the need of dynamic backpropagation adaptation algorithms (Narendra and Parthasarathy, 1990; Hagan et al., 1999; Wan and Beaufays, 1996), which are more complex than simple backpropagation ones and have a high computational cost. Note, however, that in many cases, such as the VRT one in this paper, dynamic backpropagation may be needed even with FFNN when they are part of a larger dynamic system.

In the literature (see for example (Hagan et al., 2002; Narendra and Parthasarathy, 1990; Kasparian and Batur, 1998)), several are the structures used to control a nonlinear system by means of neural networks. The neural controller can either act as a feedforward controller or as a feedback one. Some of the most popular proposals on both situations will be briefly reviewed next.

2.1. Feedforward structures: Neural Inverse Control

The main idea of *Inverse Control* is to determine the inverse of the plant and then use it as a controller. Two are the approaches that can be followed to account for this: *Direct Inverse Control* and *Feedforward Direct Control* (Narendra, 1996).

Figure 2 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. 2).

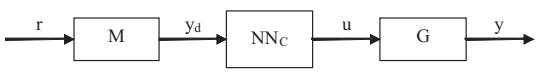


Figure 2: Direct Inverse Control using neural networks

The Feedforward Inverse Control scheme is depicted in Fig. 4. In this case, two neural networks are used. NN_1 is first trained to mimic the plant behaviour, using the prediction error e_M for this. Then the neural controller NN_2 is trained so

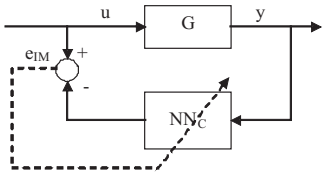


Figure 3: Training the neural network NN_C

that $(NN_2)(NN_1)$ approximates the identity. Note that the error signal e_C can be used to adapt online the controller NN_2.

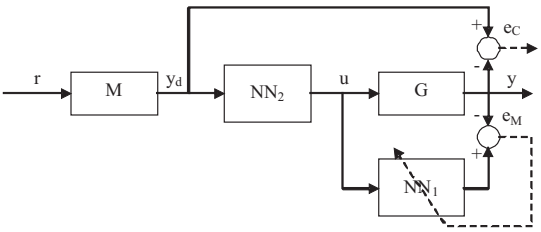


Figure 4: Feedforward Inverse Control using neural networks

The feedforward scheme has several disadvantages. Of course, the first one is that it can not be used with open-loop unstable plants. Furthermore, if the nonadaptive scheme is the one to be used, as the control is carried out in open loop, disturbances and modelling errors will degrade the system's performance.

2.2. Feedback Neural Model Reference Control

Neural Model Reference Control structure is depicted in Fig. 5. Its *adaptive* version (*i.e.*, updating the controller at every sample) is a particular case of Model Reference Adaptive Control (MRAC). In Fig. 5, the signal e_C is used either to train or adapt online the weights of the neural controller NN_C. From a theoretical point of view, two are the approaches used to design a MRAC control for an unknown plant: Direct and Indirect Control.

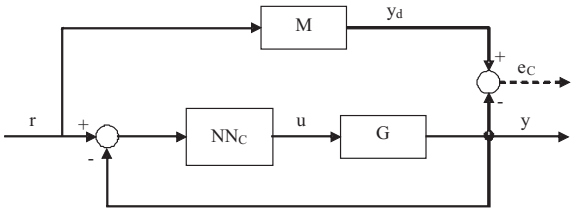


Figure 5: Model Reference Control using neural networks

Direct Control. This procedure aims at designing a controller without having a plant model. As the knowledge of the plant (or at least of an approximation of its mathematical description) is needed in order to train the neural network which corresponds to the controller (NN_C), the problem of directly training NN_C is a complex one. Indeed, the closed-loop stabilization of the (unknown) plant must be first assured. In (Lightbody and Irwin, 1995), for example, this is carried out by placing a linear fixed-gain controller in parallel with the neural network to be trained. More recently, Kumar et al. (2009) have dealt with this problem, developing a particular direct neural model reference adaptive control, the structure of which is depicted in Fig. 6. In this particular case, the stabilization of the plant has been carried out by training off-line the neural network, using Lyapunov-based synthesis and the data collected from the reference model and the unknown plant. Once trained, the neural controller NN_C is adapted on-line to improve performance.

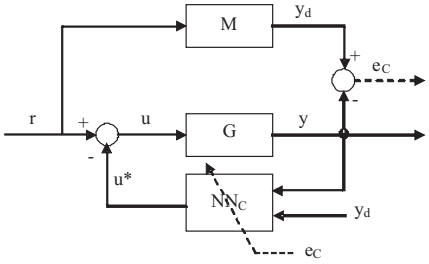


Figure 6: Direct Model Reference Adaptive Neural Control

Indirect Control. This approach uses two neural networks: one for modelling the plant dynamics (NN_M), and another one (NN_C) adjusted to control the real plant so as its behavior is as close as possible to the reference model M. This scheme is represented in Fig. 7. As a first step, the neural network NN_M is trained to approximate the plant input/output relation using the signal e_M. It is habitual to use a FFNN architecture to model the plant dynamics, as simple backpropagation algorithms can be used to train the network. This is usually done off-line, using a batch of data collected from the plant in open loop. Once the model network NN_M is trained, it is used to train the network NN_C which will act as the controller. NN_C can be either FFNN or RNN, as in both cases dynamic backpropagation is needed. The output predicted by the model NN_M $\hat{y}$ is used instead of the actual output y because the real plant dynamics is unknown, and therefore the loop's error e_C could not be backpropagated. In this way, the control error e_C is calculated as the difference between the desired reference model output y_d and $\hat{y}$. Then, as NN_M is fixed, its derivatives with respect to any parameter are known and easy to compute when training NN_C.

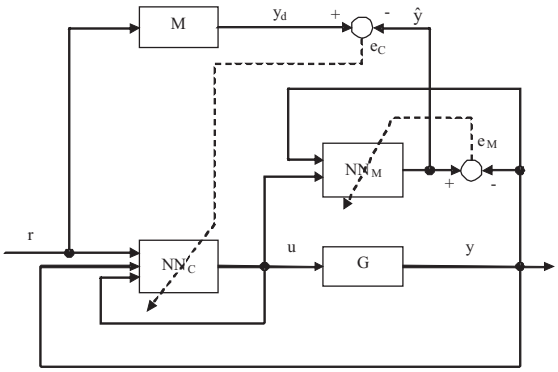


Figure 7: Indirect MRAC

There are other neuro-control paradigms, such as reinforcement learning (Sutton and Barto, 1998), sliding controllers (Baruch, 2007; Kar and Behera, 2009), neural feedback-linearisation, internal model control, model predictive control (Hagan and Demuth, 1999; Hagan et al., 2002), etc. Neural networks appear as well as elements in higher-level learning frameworks. There are also other NN topologies (CMAC, neuro-fuzzy, Elman, etc.). Liu (2001), for example, provides a more extensive description of the different structures and applications of neural networks and a larger amount of references. As the aforementioned options are unrelated to the proposals in this paper, the reader is referred to the cited references for further details.

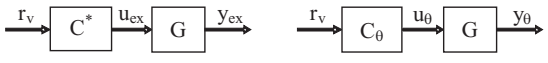
3. Virtual Reference Tuning

Virtual Reference Tuning (VRT) is a data-based direct controller design methodology, which, under certain assumptions, does not need a plant model to obtain a controller which aims at achieving a control loop's performance as close as possible to a reference model. The tuning may be for open-loop, closed-loop or even other block-diagram structures. The bibliography is quite extensive concerning a closed-loop setup (known as VRFT methodology), both for the linear and nonlinear cases. The reader is referred to the references provided in section 1 (some by the authors) for full detail on the methodology. In this section, a generic framework of VRT will be outlined.

Let us denote by G the unknown plant to be controlled and let us consider that a first open-loop experiment carried out on it gives a set of input/output data named $\{u_{ex}, y_{ex}\}$. The "virtual reference" trick consists in, given a (usually linear) invertible reference model M, generating the following "virtual" signal:

$$r_v = M^{-1} y_{ex} \tag{1}$$

Using this signal, called *virtual reference*, a virtual "perfect" loop can be constructed[1]. The idea can be applied both in feedforward and feedback loops. Figures 8a and 9a show its application in open and in closed loop, respectively. These two particular structures will be named *Virtual Reference Feedforward Tuning* (VRFFT) and *Virtual Reference Feedback Tuning* (VRFT), respectively.



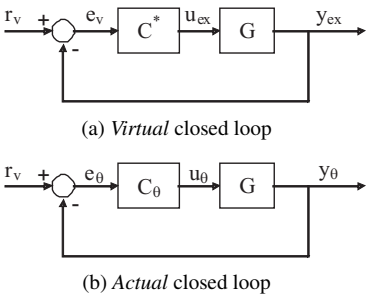
(a) *Virtual* open loop (b) *Actual* open loop

Figure 8: *Virtual* and *actual* open loops in VRFFT

In both cases, C^* stands for an unknown controller, the output of which is u_{ex} itself (and therefore, the plant's output will be y_{ex}). In the virtual closed loop (Fig. 9a) appears an additional signal, called *virtual error*, which is defined as:

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

[1] If M were not invertible, due to non-minimum-phase components, some refinements are needed Sala and Esparza (2005b). If the non-invertibility is due to delay, trivial forward-shifting some sequences will solve the issue.



(a) *Virtual* closed loop

(b) *Actual* closed loop

Figure 9: *Virtual* and *actual* closed loops in VRFT

These loops are called *virtual* because they do not exist and therefore they are not used to generate the data $\{u_{ex}, y_{ex}\}$. Obviously, both virtual loops achieve $y_{ex} = Mr_v$, *i.e.*, they work "perfectly as desired", and the input signals to the controller (r_v in open loop and e_v in closed loop) and the plant (u_{ex}) are known. Within this environment, the controller design reduces to an identification problem between the signals r_v and u_{ex} – in VRFFT – or e_v and u_{ex} – in a VRFT setup –. Identification is usually carried out considering a parameterised controller $u = C(\theta, \xi)$, where $\theta \in \mathbb{R}^p$ stands for the parameter vector (being p its length) and ξ is the controller's input (r_v in a feedforward loop or e_v in a feedback one).

On the sequel, the parameterised controller will be denoted by $C_\theta(\xi)$. The ideal controller C^* fulfills $u_{ex} = C^*(\xi)$, with $\xi = r_v$ in VRFFT and $\xi = e_v$ in VRFT, but it will be, possibly, nonlinear and high-order. As $C_\theta \neq C^*$ (actually, at startup C_θ will likely be very different from C^*), the actual output of the loop in Fig. 8b or 9b when a particular θ is used, y_θ, will differ from the ideal one $Mr_v = y_{ex}$.

The cost index to minimise in the identification procedure is, then, chosen to be the squared Euclidean norm (denoted by $\|\cdot\|^2$) of the (possibly filtered) difference between the actual output of the loop y_θ, and the ideal y_{ex}:

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

In expression above F is a frequency-weighting filter chosen by the designer (which, in the remainder, will be considered the identity, for simplicity) and y_θ is the actual plant output which, in an open loop setting (see Fig. 8b) is defined as:

$$y_\theta = G(C_\theta(r_v)) \tag{4}$$

and in a closed-loop setting (Fig. 9b):

$$y_\theta = G(C_\theta(e_\theta)) \tag{5}$$

In this latter expression $e_\theta = r_v - y_\theta$ is the (non-virtual) tracking error.

As the plant G is not known, it is not possible to minimise (3). So, the VRT methodology proposes the following data-based cost indexes for VRFFT and VRFT, respectively:

$$VRFFT: \qquad J_{VRFFT} = \frac{1}{2} \|L(C_\theta(r_v) - u_{ex})\|^2 \tag{6}$$

$$VRFT: \qquad J_{VRFT} = \frac{1}{2} \|L(C_\theta(e_v) - u_{ex})\|^2 \tag{7}$$

where L is a linear time-variant filter designed so as to make the solution to the minimization of (6) and (7) as close as possible to that of (3). Note that indexes (6) and (7) only use available data u_{ex}, r_v and e_v, contratily to (3) which uses the non-computable y_θ.

In a general framework, which accounts both for linear and nonlinear cases, 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, J_{VRFFT} and J_{VRFT} with respect to the vector of parameters θ (considering $F = I$ and the availability of $\{u_{ex}, y_{ex}\}$) are given by:

$$\frac{dJ}{d\theta} =< y_\theta - y_{ex}, \frac{dy_\theta}{d\theta} > \tag{8}$$

$$\frac{dJ_{VRFFT}}{d\theta} =< L(C_\theta(r_v) - u_{ex}), L\left(\frac{\partial C_\theta}{\partial \theta}\right) > \tag{9}$$

$$\frac{dJ_{VRFT}}{d\theta} =< L(C_\theta(e_v) - u_{ex}), L\left(\frac{\partial C_\theta}{\partial \theta}\right) > \tag{10}$$

where $< \cdot, \cdot >$ represents the scalar product and $\frac{\partial C_\theta}{\partial \theta}$ is the partial derivative of the controller with respect to its parameters. Then, the minimisation of J_{VRFFT} (in an open loop setting) or J_{VRFT} (in a closed loop setup) will be closer to that of J as the gradient in (9) or (10), respectively, approximates more accurately the gradient in (8). As it was carried out for the linear case in (Campi et al., 2002), an expression for the filter L in (9) or (10) should be derived so as to make the gradient of J_{VRFFT} or J_{VRFT} a good approximation of the gradient of J.

Neither of both sides of the scalar product of expression (8) are computable, as the plant model G is not known. So it is necessary to express this gradient as a function of both the actually recorded signals (u_{ex}, y_{ex}) and the virtual ones (r_v, e_v). Assuming the existence of an ideal θ^*, which is the vector of parameters which makes $u_{ex} = C(\theta^*, r_v)$ in VRFFT, or $u_{ex} = C(\theta^*, e_v)$, in VRFT, for θ close to θ^*, the following approximations of the left part of expressions (8), (9) and (10) can be considered:

$$y_\theta - y_{ex} \simeq \frac{dy_\theta}{d\theta}(\theta - \theta^*) \tag{11}$$

$$VRFFT: \; L(C_\theta(r_v) - u_{ex}) \simeq L\left(\frac{\partial C_\theta}{\partial \theta}\right)(\theta - \theta^*) \tag{12}$$

$$VRFT: \; L(C_\theta(e_v) - u_{ex}) \simeq L\left(\frac{\partial C_\theta}{\partial \theta}\right)(\theta - \theta^*) \tag{13}$$

Then, the problem will be solved if we find a filter L which approximates

$$\frac{dy_\theta}{d\theta} \simeq L\left(\frac{\partial C_\theta}{\partial \theta}\right) \tag{14}$$

Considering a system G, which can be linear or nonlinear, its output, when the plant's input is u_θ, can be calculated as $y_\theta = G(u_\theta)$. Then, the derivative of y_θ with respect to the controller parameters can be expressed as:

$$\frac{dy_\theta}{d\theta} = \bar{G}\frac{du_\theta}{d\theta} \tag{15}$$

where $\bar{G}$ is an operator used to express the (possibly time-variant) plant model partial derivative with respect its input, which in this particular case is u_θ, *i.e.*:

$$\bar{G} = \frac{\partial G}{\partial u_\theta} \tag{16}$$

Usually, plants and controllers are recurrent, so dynamic backpropagation (Narendra and Parthasarathy, 1990) must be used to compute both $\bar{G}$ and $\frac{du_\theta}{d\theta}$. As backpropagation through time requires the knowledge of both the plant model and the controller structure, its derivation will be tackled in next section, where we will particularise it to a neural network controller in a pre-defined block-diagram.

4. Neural Virtual Reference Tuning

VRT applications have been largely in linear cases (under the assumption of a linear plant and controller, an approximation to the filter L in (9) or (10) can be found which does not depend on the model) and in closed loop setups. Successful applications of purely linear VRFT are reported, for instance, in (Campi et al., 2003). Furthermore, linear VRFT has proven to work even when the plant is nonlinear, whenever the nonlinear plant's dynamics is not strongly relevant (see, for example (Previdi et al., 2004, 2005)).

Obviously, when significantly nonlinear behaviour of the plant is strongly excited, a nonlinear controller can likely achieve a better performance.

In this section we will consider the nonlinear VRT setup, using neural networks for the controller. The reasons for the choice of neural networks are well-known: their capability to approximate nonlinear functions (universal function approximation property) along with the ease of computing its derivatives for parameter adjustment.

Contributions (Previdi et al., 2004) and (Previdi et al., 2005) have also used neural networks along with VRFT. In these cases, a particular feedforward neural structure, using radial basis neurons was used and compared to a PID controller, the parameters of which were tuned using linear VRFT methodology. Taking these previous results into account, the contribution proposed in present paper is double: on the one hand, the usage of recurrent neural networks trained using dynamic backpropagation of the gradients and, on the other hand, the proposal of using additional sensors which improve substantially the closed-loop performance. Indeed, any feedback block diagram is amenable to dynamic backpropagation, so once the virtual reference is generated, other options apart from the pure feedforward and pure feedback are possible, as this paper exemplifies.

In this section, VRFFT and VRFT will be tackled separately, to obtain for each case a particular expression of the linear time-variant filter L which approximates the gradient of J_{VRFFT} and J_{VRFT} in (9) and (10), respectively, to the gradient of J in (8). Next, a discussion about the obtention of $\bar{G}$ will follow, finishing the section with an extension of the methodology to other block diagrams, as, for example, when additional sensors are available.

4.1. Virtual Reference Feedforward Tuning

The open loop controller implementation will now be considered. In this case, the *virtual loop* is an open loop one (Fig. 8a), where r_v is the virtual reference, calculated using expression (1). In this case, the controller must be directly identified from r_v and u_{ex}. To do this, the gradient in (9) must be approximated to the gradient in (8). In previous section it was shown that this problem will be solved if we find a linear time-variant filter L which makes the following expression true:

$$\frac{dy_\theta}{d\theta} \simeq L\left(\frac{\partial C_\theta}{\partial \theta}\right) \tag{17}$$

Let us consider as the starting point expression (15), where $\bar{G}$ is defined in (16). There are several ways of obtaining $\bar{G}$, depending on the plant model's structure. Usually, both plants and controllers are recurrent mappings, *i.e.*, the actual output depends not only on the present inputs but also on an amount of past input and output values:

$$y_k = G(u_k, u_{k-1}, \ldots, u_{k-q}, y_{k-1}, \ldots, y_{k-v}) \tag{18}$$

$$u_k = C(\theta, r_k, r_{k-1}, \ldots, r_{k-m}, u_{k-1}, \ldots, u_{k-n}) \tag{19}$$

where u, y are the plant's input and output signals, θ is the controller's vector of parameters and r, u are the controller's input and output signals, respectively. Note that the argument θ has been removed from u and y to improve clarity.

Considering the plant model in expression (18), to compute $\bar{G}$, it is necessary to apply the chain rule:

$$\frac{dy_k}{d\theta} = \sum_{j=0}^{q} \frac{\partial G}{\partial u_{k-j}} \frac{du_{k-j}}{d\theta} + \sum_{j=1}^{v} \frac{\partial G}{\partial y_{k-j}} \frac{dy_{k-j}}{d\theta} \tag{20}$$

which is a linear recurrent equation (and time-variant if G is nonlinear in y or u). This expression can be represented in a block diagram, as in Fig. 10.

As $\bar{G}$ is a linear time-variant system, the derivative in (15) must be computed at each time instant, *i.e.*:

$$\frac{dy_k}{d\theta} = \bar{G}\left(\frac{du_k}{d_\theta}\right) \tag{21}$$

where, as before, the argument θ has been omitted. Using this expression, where $\bar{G}$ is defined by the recursive equations in (20) and represented graphically in Fig. 10, to compute the gradient of J it is then necessary to obtain an expression which relates $\frac{du_k}{d_\theta}$ with $\frac{\partial C_\theta}{\partial \theta}$. For a recurrent controller (defined in expression (19)), applying the chain rule (and noting that the controller's input r_v does not depend on θ), we obtain:

$$\frac{du_k}{d\theta} = \frac{\partial C_\theta}{\partial \theta} + \sum_{j=1}^{n} \frac{\partial C_\theta}{\partial u_{k-j}} \frac{du_{k-j}}{d\theta} \tag{22}$$

Expressions (20) and (22) are the sensitivity equations defining a recurrent linear time-variant system, being $\frac{\partial C_\theta}{\partial \theta}$ and $\frac{dy_\theta}{d_\theta}$ its input and output, respectively. Fig. 11 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 $\frac{dy_k}{d\theta}$ are vectors of size p (the number of parameters θ) and $\bar{G}$ is defined in Fig. 10.

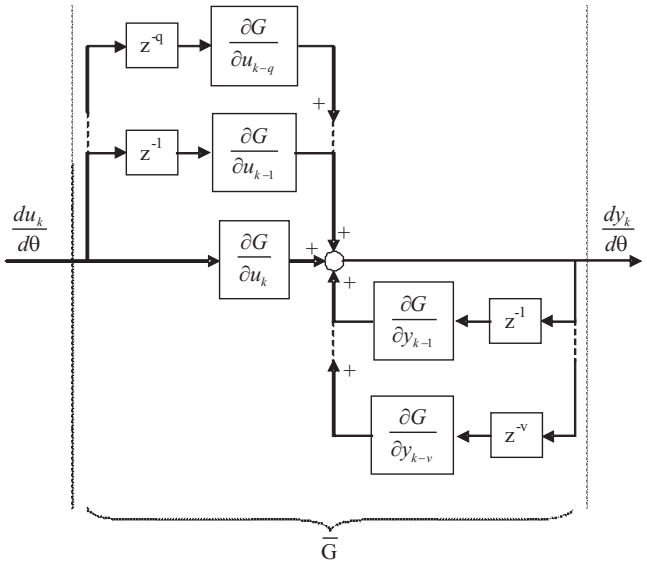


Figure 10: Block diagram of gradient propagation through the plant

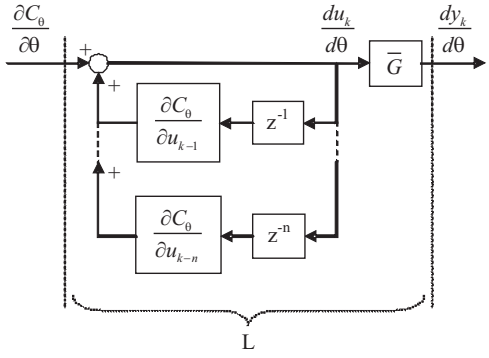


Figure 11: Sensitivity block diagram in open loop operation

4.2. Virtual Reference Feedback Tuning

In this subsection, a closed loop setup will be considered. The virtual closed loop built using the reference model M and the experimental data $\{u_{ex}, y_{ex}\}$ is depicted in Fig. 9a. The difference between this setting and the previous one is the existence of feedback, and hence the controller's input is the closed-loop error. In the virtual loop this input is the virtual error e_v and in the actual one (Fig. 9b), it is the actual closed-loop error, defined as $e_\theta = r_v - y_\theta$. The plant structure is the same described in (18), so $\bar{G}$, represented by expression (20) and in Fig. 10, is the same both for VRFFT and VRFT. However, as the controller's input is different, its output is defined as:

$$u_k = C(\theta, e_k, e_{k-1}, \ldots, e_{k-m}, u_{k-1}, \ldots, u_{k-n}) \tag{23}$$

where the argument θ has also been omitted for u and e.

The objective is still finding the linear time-variant filter L which approximates the gradient of J_{VRFT} in (10) to the gradient of J in (8). This is the same thing as, starting from expression (21), expressing $\frac{du_k}{d\theta}$ as a function of $\frac{\partial C_\theta}{\partial \theta}$. Applying the chain

rule to (23):

$$\frac{du_k}{d\theta} = \frac{\partial C_\theta}{\partial \theta} - \sum_{j=0}^{m} \frac{\partial C_\theta}{\partial e_{k-j}} \frac{dy_{k-j}}{d\theta} + \sum_{j=1}^{n} \frac{\partial C_\theta}{\partial u_{k-j}} \frac{du_{k-j}}{d\theta} \tag{24}$$

which are linear recurrent equations (and time-variant if C is nonlinear in u or e). In expression (24), it has been considered that the error signal is defined as $e_i = r_{v,i} - y_i$ and therefore its derivative with respect to the vector of parameters is $\frac{de_i}{d\theta} = -\frac{dy_i}{d\theta}$, as r_v does not depend on θ.

Expressions (20) and (24) are therefore the sensitivity equations defining a recurrent linear time-variant system, whose input is $\frac{\partial C_\theta}{\partial \theta}$. Fig. 12 represents such sensitivity equations computed at the instant k. In such figure, both the block diagram's input $\frac{\partial C_\theta}{\partial \theta}$ and output $\frac{dy_\theta}{d\theta}$ are vectors of size p (the number of parameters θ). Evidently, such a system should be the filter L in (10).

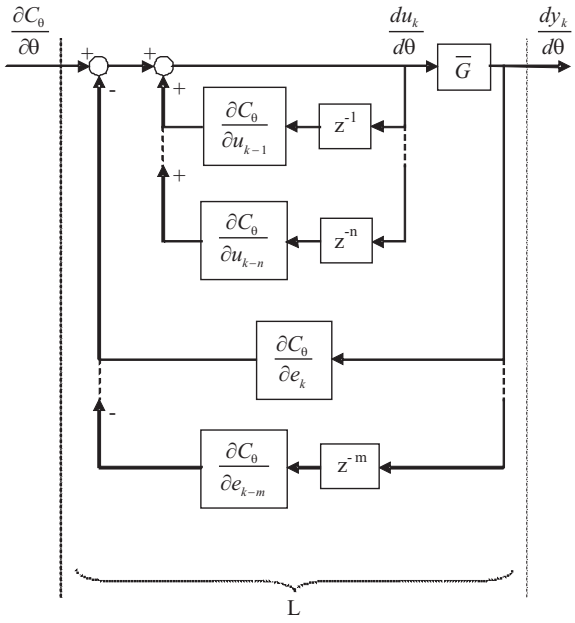


Figure 12: Sensitivity block diagram

As it can be seen, this filter is linear time-variant, so its coefficients must be updated at each k ($k \in [1, N]$, being N the data length). Expression (20) shows the explicit dependence of the filter L on $\bar{G}$, which represents the plant model linearised at each time instant around its operating point.

4.3. Obtention of $\bar{G}$

The dependence of L on $\bar{G}$ (both in VRFFT or in VRFT) shows the necessity of a plant model to propagate the gradients. And, indeed, the computation of $\bar{G}$ or an approximation of it ($\hat{\bar{G}}$) is needed. A discussion on how to solve this problem will be provided in this subsection.

Let us consider a plant with unknown dynamics G, which could be either linear or nonlinear with smooth nonlinearities. The objective is to design a controller which minimises the cost index J defined in (3) through the minimisation of the data-based one J_{VRFFT} in (6) or J_{VRFT} in (7).

As highlighted in previous subsections, a linear time-variant filter L is needed both in J_{VRFFT} and J_{VRFT} in order to approximate their gradients to the gradient of J. Such a filter requires obtaining a time-variant linearization of the plant model at each time instant and at each input value, *i.e.*, to obtain its Jacobian, which we have called $\bar{G}$.

As the plant model is not known, the instrumental $\bar{G}$ should be approximated. There are several ways of handling the issue:

- The first approach could be carried out, for example, by training (off-line) a neural network to accurately approximate the plant dynamics, which would then adapt the Indirect Model Reference Control scheme in section 2 to the virtual reference framework. In this case, the approximation $\hat{\bar{G}}$ of $\bar{G}$ can be obtained with relative ease, as the needed derivatives of the neural network with respect to its inputs are easy to compute.

- A second possibility involves identification of (purposedly) only an approximate reduced-complexity model of the plant. An appealing choice is a linearised plant model, identified using standard linear ID techniques.

- A last possibility would be disregarding the model and assuming $\bar{G}$ equal to the identity.

Note that $\hat{\bar{G}}$ is only needed as part of the filter L, which is used to make the minimization of J_{VRFT} (J_{VRFFT}) close to that of J. If the controller parameterisation is flexible enough and the real controller is within the controller class (*i.e*, there exists a θ^* such that $C(\theta^*, e_v) = u_{ex}$), the gradient of J_{VRFT} (J_{VRFFT}) will be zero at θ^* (no bias) irrespective of L: a rich enough controller does not need a model in VRFT (nor it does in VRFFT). The lower the controller class flexibility is, the more important role $\hat{\bar{G}}$ plays in the accuracy of the identified controller. Given the above considerations, in practice, some successful applications have been reported with $L = I$ or simple models of the plant.

In the proposal of this paper, the second option will be chosen, *i.e.*, an identified LTI model $\hat{G}$ of the plant to be controlled will be used. In this case, $\hat{\bar{G}} = \hat{G}$, and then expression (21) turns into:

$$\frac{dy_k}{d\theta} = \hat{G}(z^{-1}) \frac{du_k}{d\theta} \tag{25}$$

4.4. Using additional measurements in VRFT

The last contribution in this section is to develop a generic algorithm to backpropagate the gradients through the closed loop when the information from additional sensors is used to improve performance. Let us consider the closed loop depicted in Fig. 13, where a single-input-multiple-output (SIMO) plant has been considered, being u its input and $\{y_1, \cdots, y_{Ny}\}$ its outputs, where the subindex Ny stands for the number of plant's outputs. In this figure, the controller's inputs are a function (either linear or nonlinear, time-variant or time-invariant) of the plant's outputs. To simplify the development, time-invariant functions $f_i(y_i)$ of the plant's outputs will be considered.

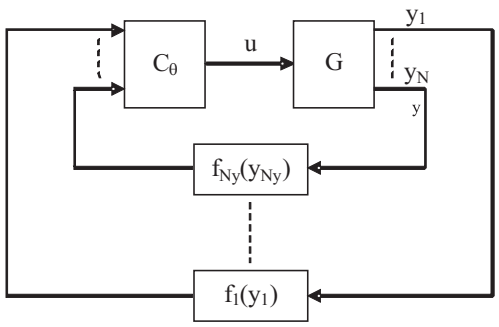


Figure 13: Generic closed loop with additional measurements

Considering a recurrent controller, its output at a time instant k is:

$$u_k = C(\theta, f_1(y_{1,k}), \cdots, f_1(y_{1,k-m_1}), \cdots, f_{Ny}(y_{Ny,k}), \cdots, f_{Ny}(y_{Ny,k-m_{Ny}}), u_{k-1}, \cdots, u_{k-n}) \quad (26)$$

The derivative of the controller's output with respect to the vector of parameters is in this case:

$$\frac{du_k}{d\theta} = \frac{\partial C_\theta}{\partial \theta} + \sum_{i=1}^{Ny} \sum_{j=0}^{m_i} \frac{\partial C_\theta}{\partial f_i} \frac{\partial f_i}{\partial y_{i,k-j}} \frac{dy_{i,k-j}}{d\theta} + \sum_{j=1}^{n} \frac{\partial C_\theta}{\partial u_{k-j}} \frac{du_{k-j}}{d\theta} \quad (27)$$

These linear (possibly) time-variant recurrent equations are the generalization of (24). Indeed, expression (24) is obtained from (27) when $Ny = 1$, $y = y_1$ and $f_1(y_1) = r_v - y_1$. Using this generalization, and the linear approximation of $\bar{G}$ discussed in previous subsection, expression (20) can be expressed as:

$$\begin{pmatrix} \frac{dy_{1,k}}{d\theta} \\ \vdots \\ \frac{dy_{Ny,k}}{d\theta} \end{pmatrix} = \bar{G} \frac{du_k}{d\theta} = \begin{pmatrix} \hat{G}_1 \\ \vdots \\ \hat{G}_{Ny} \end{pmatrix} \frac{du_k}{d\theta} \quad (28)$$

where $\hat{G}_i$ is the linearised relationship between the plant's input u and output y_i obtained using common identification algorithms (or, in a general case, the differential map of a nonlinear model, as discussed before).

5. Case study: Neural VRFT applied to a simulated crane model

In this subsection, a particularisation of the above explained methodology will be used to design a controller for a crane model operating under closed loop.

5.1. Preliminaries

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. 14). The objective is to control y_1. The choices made in this particular application are the following ones:

- The controller will be operating in closed loop, so the VRFT methodology will be applied.
- 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. In this work, the implemented filter L will be the time-variant one defined by equations (28) and (27), as the required partial derivatives are easily computed for neural networks.
- $\bar{G}$ will be approximated by a linear plant model $\hat{G}$, identified using standard OE prediction error algorithms. It will only be used as an instrument to build the time-variant filter L.
- The controller will be implemented by a neural network, the weights of which have to be adjusted conveniently.
- For the controller, two topologies will be used: $C_1 = NN(\theta, e, u)$ and $C_2 = NN(\theta, e, f(y_2), u)$ (Fig. 15 and 17, respectively). 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 will provide better performance, when compared to C_1, as discussed next.

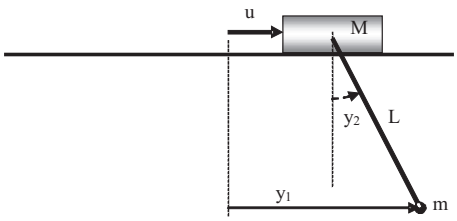


Figure 14: Structure of the crane system

In this particular application, for the controller C_1, $Ny = 1$ and $f_1(y_1) = e = r_v - y_1$. On the other hand, for the controller C_2, $Ny = 2$, $f_1(y_1) = e = r_v - y_1$ and $f_2(y_2) = s = sin(y_2)$. To compute the gradient of J_{VRFT}, for the controller's structure C_1 the filter L is depicted in Fig. 12 and defined in equations (25) and (24). For the controller's structure C_2, expression (28) gets converted into:

$$\begin{pmatrix} \frac{dy_{1,k}}{d\theta} \\ \frac{dy_{2,k}}{d\theta} \end{pmatrix} = \begin{pmatrix} \hat{G}_1 \\ \hat{G}_2 \end{pmatrix} \frac{du_k}{d\theta} \quad (29)$$

where $\hat{G}_1$ and $\hat{G}_2$ are the LTI identified plant models which relate the outputs y_1 and y_2 with the input u, respectively. In the same way, equation (27) turns into:

$$\frac{du_k}{d\theta} = \frac{\partial C_\theta}{\partial \theta} - \sum_{j=0}^{m_1} \frac{\partial C_\theta}{\partial e_{k-j}} \frac{dy_{1,k-j}}{\partial \theta} + + \sum_{j=0}^{m_2} \frac{\partial C_\theta}{\partial s} \frac{\partial s}{\partial y_{2,k-j}} \frac{dy_{2,k-j}}{d\theta} + \sum_{j=1}^{n} \frac{\partial C_\theta}{\partial u_{k-j}} \frac{du_{k-j}}{d\theta} \quad (30)$$

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. 15. 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 and the number of delayed inputs is m, at a time instant k, $\bar{u}_k = [u_{k-1}, \ldots, u_{k-n}]$ and $\bar{e}_k = [e_k, \ldots, e_{k-m+1}]$. The block TDL_x stands for the tapped delay line of length x. The input to this block is the present value of the signal and its output is a vector of x elements, made up of the present value and the past ones, as shown in Fig. 16. In the same way, the network used for the C_2-class controllers is the one in Fig. 17, where $f(y_2) = sin(y_2)$ has been added as a new network's input.

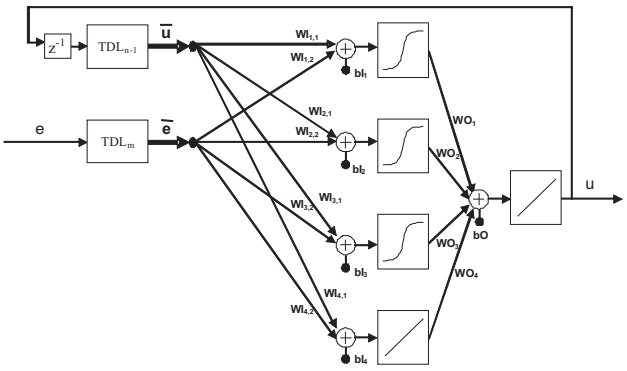


Figure 15: Neural network used for controllers of class C_1

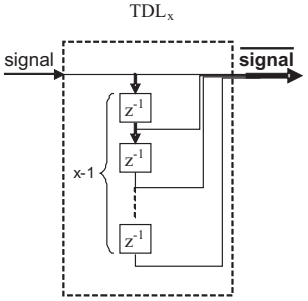


Figure 16: Tapped delay line of length x

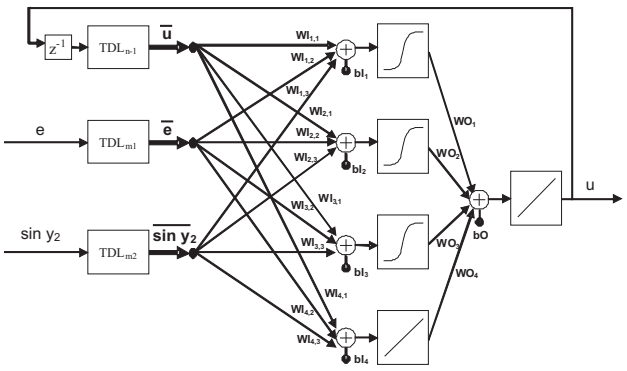


Figure 17: Neural network used for controllers of class C_2

5.2. Simulation results

As above mentioned, neural VRFT is applied to a crane model, depicted in Fig. 14, 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.
- 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.

As the actual model equations are not relevant to the data-based controller tuning procedures here demonstrated, the reader is referred to the above reference for details.

Using the neural network controller structure depicted either in Fig. 15 or 17, the control action at an instant k can be computed as:

$$u_k = \sum_{i=1}^{4} WO_i F_i \left(\sum_{j=1}^{n\times(Ny+1)} WI_{ij} x_k + bI_i \right) + bO \qquad (31)$$

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), $Ny = 1$ for the neural network structure of Fig. 15 and $Ny = 2$ for the one in Fig. 17. In addition, x_k is the input vector, constructed as:

$$x_k = \left[\begin{array}{cccccc} u_{k-1} & \ldots & u_{k-n} & e_k & \ldots & e_{k-n+1} \end{array} \right]$$

for $Ny = 1$. For $Ny = 2$, this expression turns into:

$$x_k = \left[\begin{array}{cccccccc} u_{k-1} & \ldots & u_{k-n} & e_k & \ldots & e_{k-n+1} & s_k & \ldots & s_{k-n+1} \end{array} \right]$$

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

Finally, in expression (31) 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 $(n\times(Ny+1)+2)N+1$ elements, and is defined as:

$$\theta = \left[\begin{array}{cccccccc} W^I_{1,1} & \ldots & W^I_{N,1} & \ldots & W^I_{1,n\times(Ny+1)} & \ldots & W^I_{N,n\times(Ny+1)} & \\ b^I_1 & \ldots & b^I_N & W^O_1 & \ldots & W^O_N & bO & \end{array} \right]^T \qquad (32)$$

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. 15 and 17) 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 (the nonlinearity is barely excited), but nonlinear ones improve in the second one where the nonlinearity is significant. Let us discuss each of the simulations.

Low-amplitude training data. A first open-loop experiment with a white noise input in a range of ± 0.1 N produces the signals depicted in Fig. 18. The observed 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. It is foreseeable that, as long the reference's slope increases, both controllers work worse, as the higher speeds of movements will demand a larger control action, and this will produce bigger angle variations. This, of course, will increase the nonlinear effect, beyond that learnt by the controllers identified using the weakly exciting signals in Fig. 18.

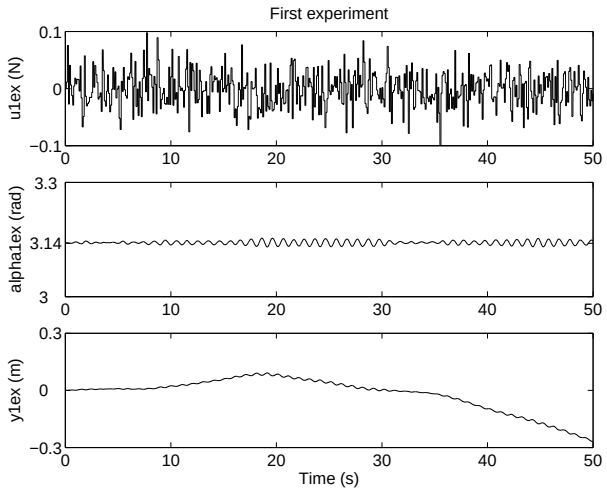


Figure 18: First simulated experiment carried out over the plant

Indeed, this is exactly what happens. 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 (in the position of the bottom tip) but a bigger oscillation of the pendulum (Fig. 19); 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. 20. The stability of the neural controller might come from pure chance (likely) or from the fact that it might have learned a somehow relevant representation of the nonlinearity.

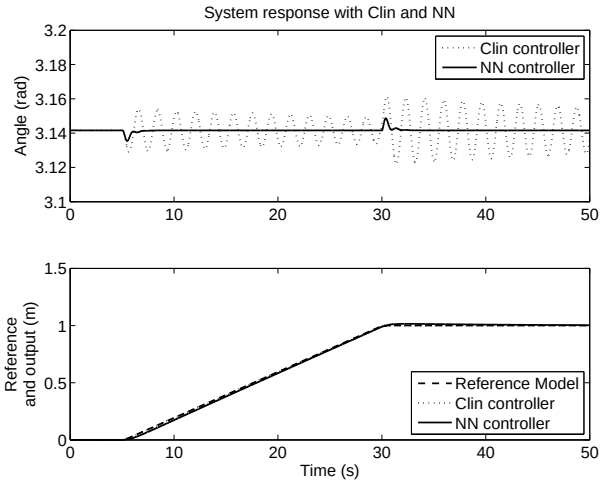


Figure 19: Comparison of plant behaviour with linear and neural network controllers (slow setpoint change speed)

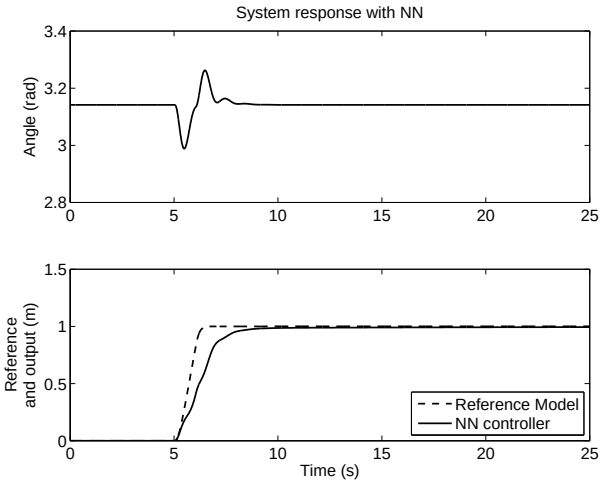


Figure 20: Plant behaviour with the neural network controller for a slope of 1 m/s in setpoint change

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, intentionally kept the same as in the previous example). The linear controller still produces an unstable loop, while the neural network one does not improve significantly over the one in Fig. 20 (results not shown for brevity).

High-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* (Fig. 21), are used to design linear and neural network controllers, using either (*a*) only the closed-loop error or, (*b*) 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 22 shows the closed-loop behaviour when using this neural network controller for a fast-varying reference of slope 1. It is clearly better than the one obtained with the controllers from low-amplitude data.

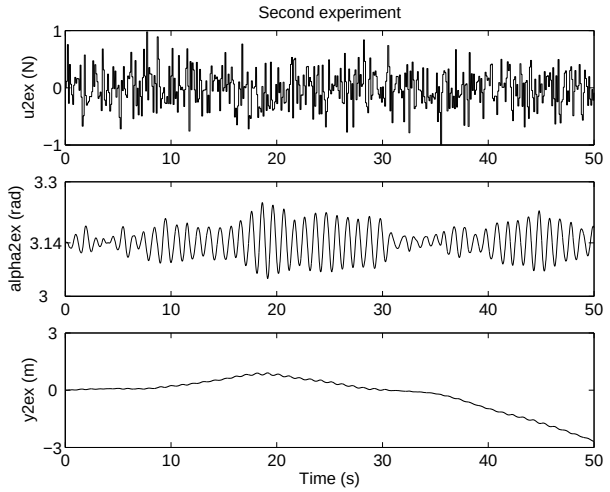


Figure 21: Signals collected from the second open-loop experiment

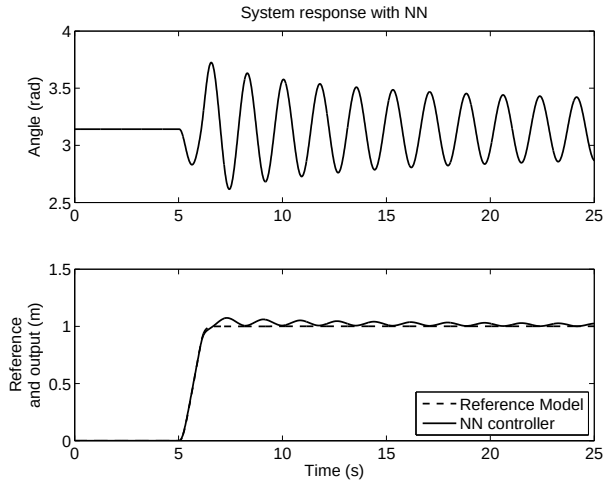


Figure 22: Closed-loop response with the neural network controller designed with 2 sensors and high-amplitude data

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 time-varying gradient expressions derived in previous sections.

6. Conclusions

This paper compares 'classical' neural network control structures with nonlinear VRT ones, both in open and closed loop setups (VRFFT and VRFT, respectively). It is shown that successful application of the VRT 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.

In simulations, and due to its clear advantages, the closed loop approach has the only been considered. They 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: only a linear output-error estimated model was used.

Acknowledgements

A. Esparza is grateful to the project GVPRE/2008/116 financed by Generalitat Valenciana. The authors are also grateful to the financial support of grant dpi2008-06731-c02-01 (Spanish Government) and Generalitat Valenciana grant prometeo/2008/088.

References

Baruch, I., 2007. Direct and indirect adaptive neural control of nonlinear systems. In: O. Castillo, P. Melin, J. K. W. P. (Ed.), Hybrid Intelligent Systems. Design and Analysis. Springer-Verlag, Berlin Heidelberg, pp. 95–114.

Bishop, C. M., 1995. Neural Networks for Pattern Recognition. Oxford University Press.

Bombois, X., Scorletti, G., Gevers, M., Hof, P. V. d., Hildebrand, R., 2006. Least costly identification experiment for control. Automatica 42, 1651–1662.

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

Campi, M., 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., Savaresi, S. M., 2002. Virtual Reference Feedback Tuning: a direct method for the design of feedback controllers. Automatica 38, 1337–1346.

Campi, M. C., Lecchini, A., Savaresi, S. M., 2003. An application of the virtual reference feedback tuning method to a benchmark problem. European Journal of Control 9, 66–76.

Gevers, M., 1993. Towards a Joint Design of Identification and Control. Birkhäuser, Ch. Essays on Control: perspectives in the theory and its applications, pp. 111–151.

Gevers, M., 2005. Identification for control: From de early achievements to the revival of experiment design. European Journal of Control 11, 1–18.

Gregorcic, G., Lightbody, G., 2008. Nonlinear system identification: From multiple-model networks to gaussian processes. Engineering Applications of Artificial Intelligence 21, 1035–1055.

Haber, R., Alique, J., 2004. Nonlinear internal model control using neural networks: an application for machining processes. Neural Computing & Applications 13 (1), 47–55.

Hagan, M., Demuth, H., 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.

Hagan, M., Jesús, O. D., Schultz, R., 1999. Training Recurrent Networks for Filtering and Control. CRC Press, Raton, FL, Ch. Chapter 12, pp. 311–340.
Hagan, M. T., Demuth, H. B., 1999. Neural networks for control. In: Proceedings of the American Control Conference (ACC). California, US, pp. 1642–1656.
Hildebrand, R., Solari, G., 2007. Identification for control: Optimal input intended to identify a minimum variance controller. Automatica 43, 758–767.
Hjalmarsson, H., 2005. From experiment design to closed-loop control. Automatica 41, 393–438.
Hjalmarsson, H., Gevers, M., Gunnarson, S., Lequin, O., august 1998. Iterative feedback tuning: Theory and applications. IEEE Control Systems Magazine 18 (4), 26–41.
Hopfield, J. J., 1982. Neural networks and physical systems with emergent collective computational abilities. In: Proc. Nat. Acad. Sci. Vol. 79. U.S., pp. 2554–2558.
Hunt, K., Sbarbaro, D., Zbikowski, R., Gawthrop, P., 1992. Neural network for control systems - a survey. Automatica 28 (6), 1083–1112.
Kar, I., Behera, L., 2009. Direct adaptive neural control for affine nonlinear systems. Applied Soft Computing 9, 756–764.
Karimi, A., Heusden, K. v., Bonvin, D., 2007. Noniterative data-driven controller tuning using the Correlation Approach. In: Proceedings of the European Control Conference. Kos Island, Greece.
Kasparian, V., Batur, C., 1998. Model reference based neural network adaptive controller. ISA Transactions 37, 21–39.
Kumar, M., Suresh, S., Omkar, S., Ganguli, R., Sampath, P., 2009. A direct adaptive neural command controller design for an unstable helicopter. Engineering Applications of Artificial Intelligence 22, 181–191.
Li, H., Deng, H., 2006. An approximate internal model-based neural control for unknown nonlinear discrete processes. Neural Networks, IEEE Transactions on 17 (3), 659–670.
Lightbody, G., Irwin, G. W., January 1995. Direct neural model reference adaptive control. In: IEE Proceedings on Control Theory and Applications. Vol. 142. IEE, pp. 31–43.
Liu, G. P., 2001. Nonlinear Identification and Control. A Neural Network Approach. Springer, London.
Miskovic, L., Karimi, A., Bonvin, D., 2003. Correlation-based tuning of a restricted complexity controller for an active suspension system. European Journal of Control 9 (1), 77–83.
Narendra, K. S., october 1996. Neural networks for control: Theory and practice. In: Proceedings of the IEEE. Vol. 84. IEEE, pp. 1385–1406.
Narendra, K. S., Parthasarathy, K., 1990. Identification and control of dynamical systems using neural networks. IEEE Transactions on Neural Networks 1 (1), 4–27.
Previdi, F., Ferrarin, M., Savaresi, S., Bittanti, S., 2005. Closed-loop control of FES supported standing up and sitting down using Virtual Reference Feedback Tuning. Control Engineering Practice 13, 1173–1182.
Previdi, F., Schauer, T., Savaresi, S., 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., Esparza, A., 2005a. Extensions to "Virtual Reference Feedback Tunning: A Direct Method for the Design of Feedback Controllers". Automatica 41 (8), 1473–1476.
Sala, A., Esparza, A., 2005b. Virtual reference feedback tuning in restricted complexity controller design of non-minimum phase systems. In: Proc. Of the $16'h$ IFAC World Congress. Elsevier IFAC Publications, Prague, Czech Republic, pp. 235–240.
Sjöberg, J., Bruyne, F. D., Agarwal, M., Anderson, B., Gevers, M., Kraus, F., Linard, N., 2003. Iterative controller optimization for nonlinear systems. Control Engineering Practice 11, 1079–1086.
Sutton, R. S., Barto, A. G., 1998. Reinforcement Learning: An Introduction. The MIT Press, Cambridge, MA.
Uraikul, V., Chan, C., Tontiwachwuthikul, P., 2007. Artificial intelligence for monitoring and supervisory control of process systems. Engineering Applications of Artificial Intelligence 20, 115–131.
Van den Hof, P., Schrama, R., 1995. Identification and control - closed-loop issues. Automatica 31, 1751–1770.
Wan, E., Beaufays, F., 1996. Diagrammatic method for deriving and relating temporal neural networks algorithms. Neural Computations 8, 182–201.
Werbos, P., 1990. Backpropagation through time: What it does and how to do it. Proc. of IEEE 78 (10), 1550–1560.
Yue, H., Li, H., Chai, T., 2007. Direct neural network-based self-tuning control for a class of nonlinear systems. International Journal of System Science 38 (8), 623–641.