Transformation of Neural State Space Models into LFT Models for Robust Control Design

Bendtsen, Jan Dimon; Trangbæk, Klaus

Publication date:
2000

Document Version
Også kaldet Forlagets PDF

Transformation of Neural State Space Models into LFT Models for Robust Control Design

Jan Dimon Bendtsen, Klaus Trangbæk
Department of Control Engineering, Aalborg University
Fredrik Bajersvej 7C, 9220 Aalborg East, Denmark.
Email: {dimon,ktr}@control.auc.dk

Abstract

This paper considers the extraction of linear state space models and uncertainty models from neural networks trained as state estimators with direct application to robust control. A new method for writing a neural state space model in a linear fractional transformation form in a non-conservative way is proposed, and it is demonstrated how a standard robust control law can be designed for a system described by means of a multi layer perceptron.

Keywords: Neural Networks, Linear Fractional Transformation, Robust $\mathcal{H}_\infty$ Control

1 Introduction

Many systems found in real-life situations are slightly nonlinear in a restricted region of the relevant state space, but exhibit saturation and other forms of nonlinear behaviour more strongly when the state of the system gets outside this region. The classical approach to control of such systems has been to linearise the system model in some set of operating points and design one or more linear controllers for the system in said points. Modern control paradigms such as robust $\mathcal{H}_\infty$ control synthesis methods typically deal with this by requiring a linear nominal (state space) model plus some kind of uncertainty model for the control design [6]. One problem with these types of approaches, however, is that it can be difficult to obtain a suitable model to build the control design on.

With the right choice of neuron functions, artificial neural networks such as *Multi-Layer Perceptrons* (MLPs) have been shown to be able to model the kind of nonlinear systems described above accurately. If the system states cannot be measured directly, an MLP can also provide a nonlinear state estimator based only on samples of in- and output. The idea of training an MLP as a nonlinear state space model of the plant and then extracting the information about the linear part of the system behaviour from the MLP then seems rather obvious. This linear system can then be employed as the nominal model of the plant, and a controller can be designed so that it is robust to the nonlinearities represented by the 'leftover' part of the plant model.

Hence, what we wish to do in this paper is to establish a link from the MLP description to a *Linear Fractional Transformation* (LFT) description and give an example of its applicability to control design. Some work along these lines has already been presented, e.g. in [3] and [4]. The results presented in these papers tend to be somewhat conservative, however, since the nonlinearities are only considered in terms of the sector bounds of the neuron functions. That is, the full range of all possible responses from the MLP is taken into account in the analysis, and it is largely ignored that, in fact, not all possible responses can be expected to occur. Strictly speaking, the MLP can only be considered a reliable plant model in the region of state space where it has been trained, and a controller based on it should of course be designed with that particular region in mind.

In Section 2 we will therefore present a new method for transforming a nonlinear state space model parametrised via an MLP into an LFT description in a much less conservative manner than what has been done so far. After that, Section 3 gives a brief outline of how to design a linear controller which is robust to perturbations from the nonlinearities via μ-synthesis. Section 4 presents a simulation example of modelling and control of an induction motor. Finally, Section 5 sums up the conclusions of the work.

2 From Neural State Space Models to Robust $\mathcal{H}_\infty$ Framework

We consider a system of the form

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

where $\tilde{x} \in \mathbb{R}^n$ is the state vector, $\tilde{u} \in \mathbb{R}^m$ is a control signal and $\tilde{y} \in \mathbb{R}^p$ is the output vector for the system. $f(\cdot,\cdot) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is an unknown continuous function of the states describing the system dynamics.

From neural network theory—see e.g. [1]—it is known that we can approximate this function to a desired accuracy with a single hidden layer MLP with q neurons

(assuming q is chosen large enough):

$$f(\tilde{x},\tilde{u}) = W_o\sigma\left(W_x\tilde{x} + W_u\tilde{u} + \tilde{W}_b\right) + \varepsilon_x$$

where $W_o \in \mathbb{R}^{n\times q}$ and $W_x \in \mathbb{R}^{q\times n}, W_u \in \mathbb{R}^{q\times m}$ contain the output and hidden layer weights, respectively. $\sigma(\cdot) : \mathbb{R}^q \to \mathbb{R}^q$ is a continuous, diagonal, static nonlinearity. $W_b \in \mathbb{R}^q$ contains a set of biases which will allow us to model non-odd functions with odd neuron functions $\sigma(\cdot)$ such as the hyperbolic tangent. We assume it is possible to bound the modelling error: $\|\varepsilon_x\|_\infty \leq \varepsilon_{max}$ by choosing the MLP large enough and train it long enough on a sufficiently rich training set.

In other words, we will assume that the neural network can be trained to estimate the states in the system (1). In practice this can for instance be achieved by employing Narendra's Dynamic Back-propagation (see e.g. [3]), and we will for simplicity only consider offline training here; i.e. we will not consider time-varying systems.

Consider a system for which a neural state space model has been trained according to the guidelines given above, until ε_x is small enough to be ignored:

$$\dot{\tilde{x}} = W_o\sigma\left(W_x\tilde{x} + W_u\tilde{u} + \tilde{W}_b\right), \quad \tilde{y} = C\tilde{x} \qquad (2)$$

We wish to rewrite the neural model (2) as the linear fractional transformation

$$\dot{x} = Ax + Bu + B_1\Delta(W_x x + W_u u), \quad y = Cx \qquad (3)$$

where the uncertainty model Δ is diagonal and bounded with $\|\Delta(\cdot)\|_\infty < 1$, and where the coordinates (x,u) only differ from $(\tilde{x},\tilde{u})$ by the possible subtraction of an equilibrium point.

We assume that there exists an equilibrium, $(\tilde{x},\tilde{u}) = (\tilde{x}^\circ,\tilde{u}^\circ)$, i.e.

$$0 = W_o\sigma(W_x\tilde{x}^\circ + W_u\tilde{u}^\circ + \tilde{W}_b)$$

We can then change the network coordinates in such a way that instead of the arbitrary equilibrium point $(\tilde{x}^\circ,\tilde{u}^\circ)$ we have $0 = W_o\sigma'(0)$ (σ' is a new neuron function mapping which will be defined shortly). Let the new coordinates be given as $x = \tilde{x} - \tilde{x}^\circ, u = \tilde{u} - \tilde{u}^\circ$. Then (2) can be written as

$$\dot{x} = W_o\sigma\left(W_x(x+\tilde{x}^\circ) + W_u(u+\tilde{u}^\circ) + \tilde{W}_b\right)$$

Here we will define a new bias vector $W_b = W_x\tilde{x}^\circ + W_u\tilde{u}^\circ + \tilde{W}_b$ and the new neuron function

$$\sigma'(\cdot) = \sigma\left(W_x(x+\tilde{x}^\circ) + W_u(u+\tilde{u}^\circ) + \tilde{W}_b\right) - \sigma\left(W_b\right)$$

Adding and subtracting $W_o\sigma(W_b)$ in (2) then gives

$$\begin{aligned}\dot{x} &= W_o\sigma\left(W_x\tilde{x} + W_u\tilde{u} + \tilde{W}_b\right) + W_o\sigma\left(W_b\right) - W_o\sigma\left(W_b\right)\\ &= W_o\left(\sigma\left(W_x\tilde{x} + W_u\tilde{u} + \tilde{W}_b\right) - \sigma\left(W_b\right)\right) + W_o\sigma\left(W_b\right)\\ &= W_o\sigma'(W_x x + W_u u)\end{aligned}$$

$W_o\sigma(W_b) = 0$, because this is in fact the equilibrium point. Note that, apart from providing a way to shift the operation point to the origin, the main purpose of the steps given above is to remove the bias from σ instead of having to consider them as constant disturbance inputs, as suggested in [3].

It should furthermore be noted that the method given above applies equally well to sampled-data systems $\tilde{x}_{k+1} = f(\tilde{x}_k,\tilde{u}_k)$. In this case the MLP equilibrium point is of the form $\tilde{x}^\circ_{k+1} = f(\tilde{x}^\circ_k,\tilde{u}^\circ_k)$, $\forall k$, but the definition of $\sigma'(\cdot)$ turns out to be the same.

Now we can find the effective range of the input arguments to the neuron functions, denoted ξ. This is simply done by calculating

$$\xi_{j,max} = \sup_{0\geq t\geq T}\left\{|W_x^j x + W_u^j u|\right\}$$

for $1 \leq j \leq q$ where $[0;T]$ is the time interval in which the training data have been acquired and W_x^j, W_u^j denote the j'th rows in the hidden layer weight matrices. Then we have the following bounds on the active input range[1] of the j'th neuron:

$$\xi_j = W_x^j x + W_u^j u \in [-\xi_{j,max};\xi_{j,max}]$$

Hence the neuron function response to the active input range must belong to the sector $\sigma'_j \in [k_{j,min}\ ,\ k_{j,max}]$ where

$$k_{j,min} = \inf_{\xi_j\in[-\xi_{j,max};\xi_{j,max}]\setminus\{0\}}\left\{\frac{\sigma'(\xi_j)}{\xi_j}\right\} \qquad (4)$$

and

$$k_{j,max} = \sup_{\xi_j\in[-\xi_{j,max};\xi_{j,max}]\setminus\{0\}}\left\{\frac{\sigma'(\xi_j)}{\xi_j}\right\} \qquad (5)$$

In other words, the sector bounds are determined such that $k_{j,min}\xi_j^2 \leq \xi_j\sigma'(\xi_j) \leq k_{j,max}\xi_j^2$. The actual expressions for these sector bounds must be found for each neuron function individually and will in general depend on the bias, but the bounds obviously exist and are apparently the least conservative.

Once the sector bounds are found, we go back to vector notation and define the nonlinear function $\delta(\cdot) : \mathbb{R}^{n+m} \to \mathbb{R}^n$ as

$$\delta(x,u) = \sigma'(\cdot) - \frac{1}{2}\left(K_{min} + K_{max}\right)\left(W_x x + W_u u\right) \qquad (6)$$

where $K_{min} = \mathrm{diag}\{k_{j,min} - \epsilon\}$ and $K_{max} = \mathrm{diag}\{k_{j,max} + \epsilon\}, 1 \leq j \leq q$. ϵ is a small positive quantity included to make the sector bounds strict. It is observed that $\delta(\cdot)$ belongs to the sector $(-\frac{1}{2}(K_{max} - K_{min})\ ,\ \frac{1}{2}(K_{max} - K_{min}))$. Now we can write the equation for $\dot{x}$ as

$$\begin{aligned}\dot{x} &= W_o\sigma'(W_x x + W_u u)\\ &= W_o\left(\delta(x,u) + \frac{1}{2}\left(K_{min} + K_{max}\right)\left(W_x x + W_u u\right)\right)\\ &= Ax + Bu + B_1\Delta(W_x x + W_u u)\end{aligned}$$

[1] The input ranges are in general not symmetric around 0, so the bounds given here may be slightly conservative.

in which A, B, B_1 and Δ are given by

$$A = \frac{1}{2} W_o \left(K_{min} + K_{max}\right) W_x \quad (7)$$

$$B = \frac{1}{2} W_o \left(K_{min} + K_{max}\right) W_u \quad (8)$$

$$B_1 = \frac{1}{2} W_o \left(K_{max} - K_{min}\right) \quad (9)$$

$$\Delta(W_x x + W_u u) = 2 \left(K_{max} - K_{min}\right)^{-1} \delta(x, u) \quad (10)$$

Note that the scaling by $\frac{1}{2}(K_{max} - K_{min})$ is included in order to make Δ belong to the sector $(-1\ ,\ 1)$.

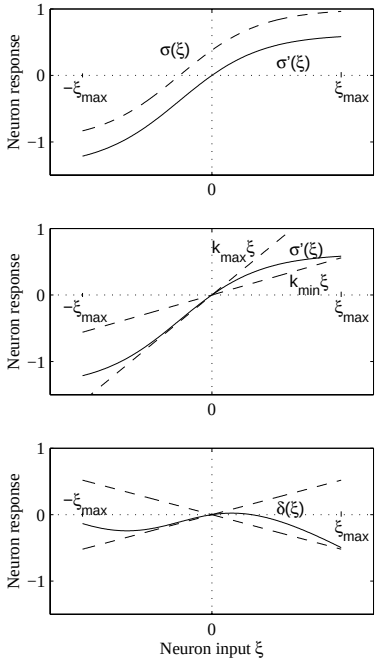

Figure 1: *Extraction of linear content from normal hyperbolic tangent neuron.*

In order to illustrate the procedure above we will provide an expression for the sector bounds (4) and (5) for the $\tanh(\cdot)$ neuron function, which is probably the most popular neuron function employed in MLPs. Refer to Figure 1, where the top plot shows the parallel translation of the original neuron function with bias W_b to the origin. We will without loss of generality assume that $W_b > 0$. Only the section of the neuron function, which corresponds to the input interval $[-\xi_{max}; \xi_{max}]$, is considered.

On the middle plot the straight lines $k_{min}\xi$ and $k_{max}\xi$ have been added. Since $d^2(\tanh(x))/dx^2 < 0$ for $x > 0$ it is immediately concluded that k_{min} is given by $k_{min} = \sigma'(\xi_{max})/\xi_{max}$. k_{max}, on the other hand, can either be given by $\sigma'(-\xi_{max})/-\xi_{max}$ if the endpoint of the input range is sufficiently close to 0, or by the slope of the tangent to the neuron function which intersects 0. The relationship between the bias and the argument ξ_b for which said tangent coincides with the neuron function has been found numerically[2] as

$$\xi_b = -0.00379 W_b^3 + 0.07274 W_b^2 - 1.5146 W_b$$

Hence, if $\xi_b > -\xi_{max}$ we have $k_{max} = \sigma'(\xi_b)/\xi_b$, otherwise $k_{max} = \sigma'(-\xi_{max})/-\xi_{max}$.

Note that there is no loss of generality in the assumption $W_b > 0$ since the fact that the (original) neuron function is odd ensures that the expressions given above hold for negative biases as well, with a simple sign change of ξ_b and ξ_{max}.

3 Controller Synthesis

This section deals with the question of designing a robust controller K for the uncertain system $\Delta \star M$ as shown in Figure 2, where $\star$ denotes the Redheffer star product.

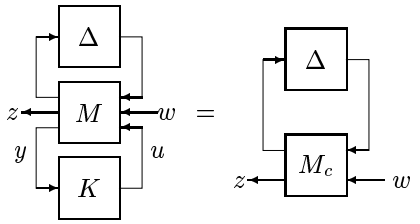

Figure 2: *The interconnection of the nominal system M, the uncertainty block Δ, and the controller K.*

y is the measurement vector, u is the control vector, z is the signal to be controlled, which may coincide with y, and w is the disturbance vector containing noise and command signals. We will for simplicity assume that $w, z \in \mathbb{R}^m$, and $\Delta \in \bar{\Delta}_d \subset \mathcal{H}_\infty^{\eta-m \times \eta-m}$. See [6] for a more thorough discussion of the topics presented here.

For a matrix $M_s \in \mathbb{C}^{\eta \times \eta}$ and an uncertainty set $\bar{\Delta}_g$ the structured singular value of M_s is defined as

$$\mu_{\bar{\Delta}_g}(M_s) = \left(\min\{\bar{\sigma}(\Delta) : \Delta \in \bar{\Delta}_g, \det(I - M_s\Delta) = 0\}\right)^{-1}$$

Assuming that $\|\Delta\|_\infty < 1$ for all $\Delta \in \bar{\Delta}_d$, and that $M_c(s) \in \mathcal{RH}_\infty^{\eta \times \eta}$, the interconnection in Figure 2 is stable and has robust performance level γ, i.e. $\|z\|_2 \leq \gamma \|w\|_2$, if

$$\mu_{\bar{\Delta}_{dp}}\left(M_c(j\omega)\begin{bmatrix} I_{\eta-m} & 0 \\ 0 & \gamma^{-1} I_m \end{bmatrix}\right) \leq 1, \forall \omega \in \mathbb{R}$$

where

$$\bar{\Delta}_{dp} = \left\{\Delta_{dp} : \Delta_{dp} = \begin{bmatrix} \Delta & 0 \\ 0 & \Delta_p \end{bmatrix}, \Delta \in \bar{\Delta}_d, \Delta_p \in \mathbb{C}^{m \times m}\right\}$$

The structured singular value usually cannot be computed by an efficient algorithm, but if Δ is a nonlinear diagonal

[2] A closed form most likely does not exist. The polynomial given here provides values of k_{max} with errors of the order of magnitude 10^{-5}.

uncertainty like the one presented in Section 2 an upper bound is given by

$$\begin{aligned}\nu_{\bar{\Delta}_{dp}}(M_s,\gamma) &= \inf_D \bar{\sigma}\left(\begin{bmatrix} D & 0 \\ 0 & I_m \end{bmatrix} M_s \begin{bmatrix} D^{-1} & 0 \\ 0 & \gamma^{-1} I_m \end{bmatrix}\right) \\ &\geq \mu_{\bar{\Delta}_{dp}}(M_s)\end{aligned}$$

where D is a diagonal matrix.

The aim is to find the controller K yielding the best robust performance, i.e. solving

$$\min_{K,D} \quad \gamma \quad s.t. \quad \nu_{\bar{\Delta}_{dp}}(M(j\omega),\gamma) \leq 1, \forall \omega \in \mathbb{R}.$$

This is a non-convex problem. One approach that usually yields good results is to perform a bisectional search over γ. The so-called DK-iteration is performed for each value of γ, where a minimisation of ν is iteratively solved for K and D.

4 Simulation Example

This section will demonstrate the usage of the methods outlined in the previous two sections on a fairly realistic simulation example. A nonlinear induction motor model simulated in continuous time provides training data for an MLP model, which is then transformed into the LFT form (3).

With widely used simplifying assumptions a simple model of the current controlled induction motor is given by (see e.g. [2])

$$\begin{aligned}\tfrac{d}{dt} i_{md}(t) &= T_r^{-1}(i_{sd}(t) - i_{md}(t)) + (\omega(t) - \omega_r(t)) i_{mq}(t) \\ \tfrac{d}{dt} i_{mq}(t) &= T_r^{-1}(i_{sq}(t) - i_{mq}(t)) - (\omega(t) - \omega_r(t)) i_{md}(t) \\ \tfrac{d}{dt} \omega_r(t) &= -\beta_1 \omega_r(t) - \beta_2 m_L(t) + \\ &\quad \beta_3 \left(i_{md}(t) i_{sq}(t) - i_{mq}(t) i_{sd}(t)\right)\end{aligned}$$

where the stator currents i_{sd} and i_{sq} are the control signals, ω_r is the angular velocity of the shaft which we wish to control, and m_L is a load torque on the shaft. i_{md} and i_{mq} are the magnetising currents in the rotor, while ω is the angular velocity of the reference frame in which the currents are expressed. For each of the currents, the subscripts 'd' and 'q' refer to parallel and perpendicular axes of decomposition relative to the rotating reference frame. Apart from i_{sd} and i_{sq} only ω_r is assumed to be measurable in the simulation.

ω is typically chosen so that the reference frame follows the magnetising currents, i.e. $i_{mq} = 0$. Since only ω_r is measured, an observer is needed. The simplest such observer, and the one which will be used here, is given by

$$\begin{aligned}\frac{d}{dt}\hat{i}_{md}(t) &= \hat{T}_r^{-1}(i_{sd}(t) - \hat{i}_{md}(t)) \\ \omega(t) &= \omega_r(t) + \hat{T}_r^{-1} \frac{i_{sq}(t)}{\hat{i}_{md}(t)}\end{aligned}$$

The observer uses an estimate of the rotor time constant $\hat{T}_r$. This parameter estimate is the only one used in the modelling and controller design. To simulate a realistic situation the estimate used will have a 3% error compared to the value used in the simulations. The modelling and controller design will be based only on $\hat{T}_r$, input/disturbance ($\tilde{u} = [i_{sd}\ i_{sq}]^T, \tilde{d} = \hat{i}_{md}$) and output ($\tilde{y} = \omega_r$) data.

The parameter values used are $T_r = 0.0949\ s$, $\hat{T}_r = 0.0921\ s$, $\beta_1 = 23.0\ s^{-1}$, $\beta_2 = 2000\ kg^{-1}m^{-2}$, and $\beta_3 = 3794\ A^{-2}s^{-2}$.

This model is simulated in continuous time with filtered pseudo-random noise as input. The input and output signals were sampled at every $T_s = 0.005\ s$ and collected in two consecutive sequences spanning 2000 training samples and 1000 test samples, respectively. In order to get good training results for the MLP and to provide a linear system identification model for the control synthesis comparison it was chosen to define the model states based on delayed input and output samples. It should be noted that it might be a more general—and better—idea to allow the MLP model to be trained as an innovation model, but this approach will for simplicity not be considered here. After some trial-and-error it was found that the following choice of input-output and state configuration gave the best modelling results: $\tilde{x}_k = [\tilde{y}_k, \tilde{y}_{k-1}, \tilde{y}_{k-2}, \tilde{y}_{k-3}]^T$, $\tilde{\zeta}_k = [\tilde{x}_k^T, \tilde{u}_k^T, \tilde{u}_{k-1}^T, \tilde{d}_k^T]^T$ and

$$\tilde{x}_{k+1} = W_o \sigma\left(W_i \tilde{\zeta}_k + \tilde{W}_b\right), \quad \tilde{y}_k = \begin{bmatrix} I & 0 \end{bmatrix} \tilde{x}_k$$

Based on the same samples a standard linear system identification gave estimates $[\bar{A}\ \bar{B}_u\ \bar{B}_d]$ of the parameters in the linear model

$$x_{k+1} = \bar{A} x_k + \bar{B}_u \begin{bmatrix} u_k \\ u_{k-1} \end{bmatrix} + \bar{B}_d d_k$$

These initial linear estimates can also be used to get a good set of starting weights for the MLP, since, if the system were linear, the gain of the hidden layer would be very close to identity and the product $W_o W_i$ would be very close to $[\bar{A}\ \bar{B}_u\ \bar{B}_d]$. Hence, we let $[\bar{A}\ \bar{B}_u\ \bar{B}_d]$ be decomposed via SVD as $U\Sigma V^T$, choose any matrices $Q, R \in \mathbb{R}^{n \times q}$ such that $QR^T = I, \|Q\| \gg \|R^T\|$, and set $W_o = UQ$ and $W_i = R^T \Sigma V^T$, thereby distributing the linear estimates to the initial weight matrices.

In the example, 8 $\tanh(\cdot)$ neurons in the hidden layer were sufficient to model the system adequately. A comparison between an open loop simulation of the test set performed by the linear model and the MLP model is shown in Figure 3. As expected the two models are equally good at simulation close to the operating point, but as the state moves away from the operating point the neural simulation is better than the linear. This can be seen on the simulation error on the lower plot.

Based on the LFT form of this MLP model, we design a stabilising discrete-time controller for the system; refer to Figure 4, where the configuration is presented. We also calculate a controller $\bar{K}$ with the exact same design, but based directly on $\bar{A}, \bar{B}_u$ and $\bar{B}_u$ (i.e. not robust to Δ). The only actual 'design' takes place in the filter F,

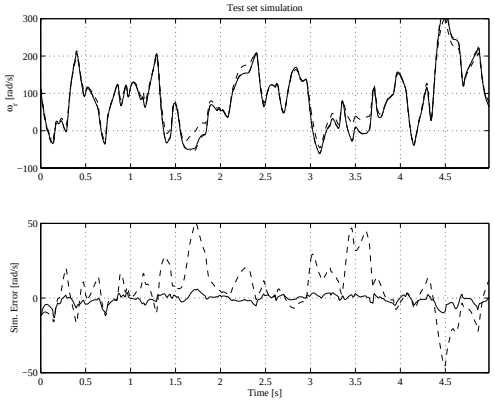

Figure 3: *Top: simulation of the test set (−·−) by the linear (−−) and the MLP model (—). Bottom: simulation errors for the linear (−−) and the MLP model (—).*

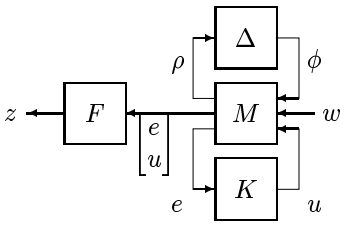

Figure 4: *The controlled system, with controller K, nominal system M, nonlinearity block Δ and a filter F on the performance channel. The output error e is defined as $e = y - w$.*

which defines a frequency weighting of the performance channel $w \rightarrow z$. F is identical for both controllers and chosen as two first order low pass filters on the outputs with cut-off frequencies of 10 rad/s for ω_r and 1 rad/s for $\hat{i}_m$, and static gains of 0.05 and 0.004 on i_{sd} and i_{sq}, respectively. The latter gain has been detuned sufficiently to avoid instability in the simulations with the controller $\bar{K}$. Figure 5 shows a simulation of the controlled system.

As can be seen, both controllers perform well near the operating point, but after a change in the magnetising current the non-robust controller $\bar{K}$ almost destabilises the system. The robust controller K, on the other hand, performs well over a wide range of states.

5 Discussion

In Section 2 a method has been developed for turning an MLP model into an LFT of a dynamic system and a nonlinear function. In Section 4 a controller was designed for a simulation example based on robust $\mathcal{H}_\infty$ theory and an LFT model obtained with the method described in Section 2. The controller designed in this manner was shown to be superior to a controller designed based on a linearisation in an operating point.

When used for linear controller design the obtained

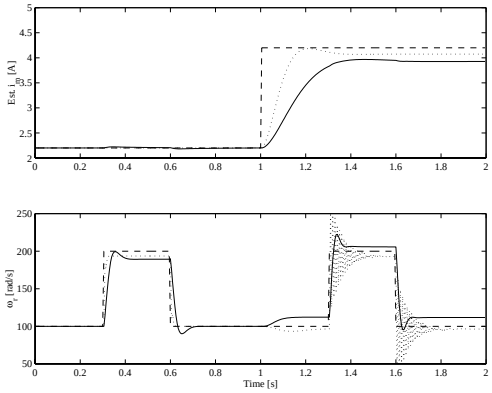

Figure 5: *Simulation of the controlled system. The plot shows the reference signal (−−), the non-robustly controlled system (···) and the robustly controlled system (—).*

model provides a far less conservative result compared to earlier methods. The advantage is obtained by considering only the region of the state space which is relevant and by taking advantage of a known equilibrium. This approach will in practically all cases be a clear improvement over a design based on the unmodified sector bounds of the neuron function and considering biases as constant disturbance inputs.

The objective of the example was to demonstrate that reasonable results could be obtained by a fairly automatic approach with the new method, even though it might be possible to obtain similar results with a standard nonlinear field oriented controller.

References

[1] S. Lu and T. Basar "Robust Nonlinear System Identification Using Neural Network Models," IEEE Transactions on Neural Networks Vol. 9, pp. 407–429, 1998

[2] H. Rasmussen, P. Vadstrup, H. Børsting, "Nonlinear Field Oriented Control of Induction Motors using the Backstepping Design," Proc. of European Control Conference, Sep. 1999

[3] J. A. K. Suykens, J. Vandewalle, B. De Moor, "Nonlinear System Identification using Neural State Space Models, Applicable in Robust Control Design," International Journal of Control Vol. 1, pp. 129–152, 1995

[4] J. A. K. Suykens, J. Vandewalle, B. De Moor, "Global Asymptotic Stability Criteria for Multilayer Recurrent Neural Networks with Applications to Modelling and Control," Proc. of the IEEE Int. Conf. on Neural Networks, Nov.-Dec. 1995

[5] J. A. K. Suykens, J. Vandewalle, B. De Moor, "Lur'e Systems with Multilayer Perceptron and Recurrent Neural Networks: Absolute Stability and Dissipativity," IEEE Transactions on Automatic Control Vol. 4, pp. 770–774, 1999

[6] K. Zhou, J. Doyle, K. Glover, "Robust And Optimal Control," Upper Saddle River, NJ: Prentice-Hall, 1996