The 3rd IEEE Conference on Control Applications
24.-26. August 1994, The University of Strathclyde, Glasgow, Scotland, UK

Real-Time Application of Neural Model Predictive Control for an Induction Servo Drive

D. Neumerkel, J. Franz
Daimler-Benz AG, Forschung Systemtechnik
Alt-Moabit 91b, D-10559 Berlin
E-mail: `neumerk@DBresearch-berlin.de`

L. Krüger, A. Hidiroglu
TU-Berlin, Inst. f. Elektronik
Einsteinufer 17, D-10587 Berlin
E-mail: `krueger@tubife.ee.tu-berlin.de`

Keywords:
Neuro-control, Model Predictive Control, real-time implementation

Abstract

This paper deals with a real-time implementation of a neural net controller and its application to an induction machine. The controller used within this framework is the Model-based Predictive Control structure (MPC) with a neural network model of the process dynamics.

The hardware platform, a multi-transputer system, is described as are the implemented algorithms. Some results show the benefits of the neural network controller compared to the field oriented control structure which is the most common structure used to control induction machines.

1. Introduction

Daimler-Benz is engaged in a research project for the subsidiary company *AEG Daimler-Benz Industry.* The project concerns neural networks for nonlinear control of industrial processes such as steel manufacturing [Sba93]. In this paper a real-time implementation of a Model Predictive Controller (MPC) using Radial Basis Function neural networks (RBF) will be described.

The conventional method for controlling an induction machine is to transform the system equations into a coordinate system rotating together with the flux. By identifying the flux current and calculation of two nonlinear feedback terms it is easy to decouple and linearize the two bilinear voltage differential equations. This allows one to design two linear control paths, the first to control the flux, and the second for controlling torque and speed.

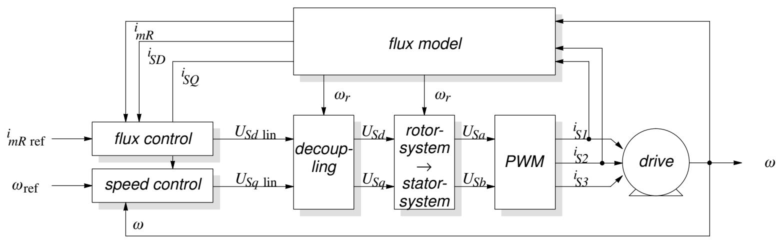

Fig. 1: Speed control using Field Oriented Control

In this project it has been our goal to figure out the demands of a real-time neuro-control implementation. For this purpose, the speed control path, including the nonlinear decoupling term, was replaced with an RBF-based predictive controller.

The next section gives a brief description of Model Predictive Control (MPC). The process model used within that control structure is implemented by a Radial Basis Function Neural Net (RBF) which is described in section 2.2.

The real time implementation of the optimization procedure, being the basic part of the predictive controller, is obtained by an iterative interval search algorithm (section 2.1). After a discussion of hardware and software aspects (section 3), section 4 gives some interesting results based on both simulations and practical measurements taken from the induction machine. The paper closes with conclusions.

2. Model Predictive Control

Model Predictive Control is a well known control technique which is often used for slow processes found in chemical applications [Psi90].

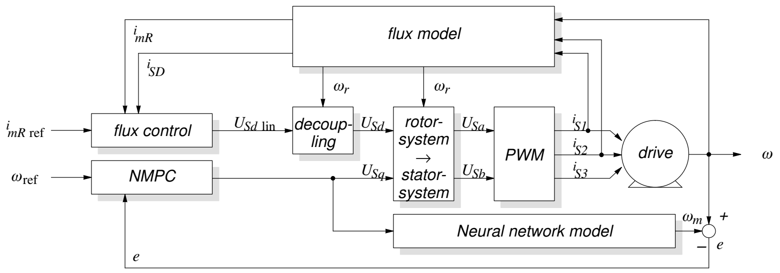

Fig. 2: Modifications for Neural Model Predictive Control

MPC works as an online optimal control procedure based on a performance index which has to be defined according to the process characteristics and the desired control dynamics [Gar89]. This index J contains predicted values of the controlled process variable supplied by a system model ω_m, and the control action to be found for a certain optimization horizon T_h.

$$J(k) = \sum_{i=1}^{T_h} w_i \cdot \left(\omega_{ref}(k) - \omega_m\left\{U_{Sq}(k+i)\right\} + e(k)\right)^2 \tag{1}$$

Minimizing this index, a sequence of control values $U_{Sq}(k+i),\ i \in \{1\ ..\ T_h\}$ is calculated, but only the first control term $U_{Sq}(k+1)$ is applied. In the following time instants the optimization problem has to be solved again, taking into account new measurements and deviations between model and process.

The sampling period determines the real time bounds for the MPC optimization procedure, which has to be solved online at each time step. The next subsection gives a description of a computationally effective optimization algorithm.

2.1 Optimization Using Interval Search

In order to match the real time conditions, the goal of any optimization algorithm should be to use as few iterations as possible. This is achieved by optimizing only the control term to be applied next using an iterative minimum search algorithm. Subsequent time steps within the optimization horizon were set to be the same

$$U_{Sq}(k+j) = U_{Sq}(k+1),\ j \in \{2 \,..\, T_h\} \quad . \tag{2}$$

The interval search algorithm (IS) divides the searched space recursively into nested intervals, each being half of the interval of the previous iteration. The algorithm can be stopped at a prespecified iteration number according to the calculation requirements of the model used and the sampling rate needed. This algorithm is not as general as standard optimization procedures, but it works efficiently, even if the number of iterations is limited to three or four.

The maximal number of iterations allowed within a certain sampling time will determine the resolution of a binary discretization of the control value, since each additional iteration increases the resolution of the optimal control value by a factor of two. In order to overcome this limitation, the initial interval size determining the maximum of change in the control variable is adapted according to the control error. Small control errors will lead to a fine resolution, whereas large deviations will give coarse control variable resolution.

2.2 System Model Based on a Radial Basis Function Neural Net

Radial Basis Function neural nets (RBF) have been described in various papers e.g. [Pog90][Neu93]. The main advantages of RBF nets are good approximation capability at reasonable expense. In addition, they can easily be implemented on parallel hardware. In contrast, the multilayer perceptron, the most familiar neural network architecture, is not well suited for a parallel implementation. Fig. 3 shows the RBF network structure.

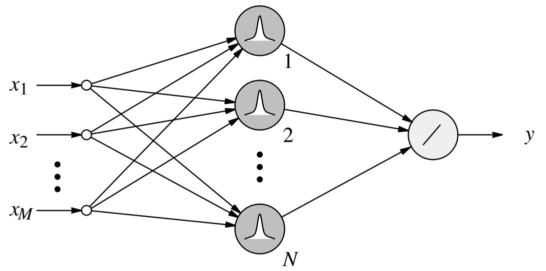

Fig. 3: Radial Basis Function Network (RBF)

The RBF network performs a static, nonlinear mapping of an input vector $\vec{x}$ to a scalar output y. The mapping is given by

$$y = \sum_{i=1}^{N} w_i \, \phi_i \quad , \tag{3}$$

with w_i being weighting factors and the basis functions ϕ. These functions can be described by Gaussians

$$\phi_i = e^{-\frac{1}{2}\left(\frac{\vec{x}-\vec{c}_i}{\sigma_i}\right)^2} \quad , \tag{4}$$

with centers c_i and widths σ_i. For this application, the nonlinear modeling task was to capture the induction machine's dynamics in terms of the voltage to speed relationship.

The real machine showed a slightly different output characteristic than the simulated machine when driven by the same input sequence. This is due to unmodeled effects, such as friction and disturbances, as well as uncertainties in the parameters. The impact of these effects on the machine's dynamics can be seen in Fig. 4, upper curves, showing different speed responses of the simulated and the real machine with respect to the same input U_{Sq}.

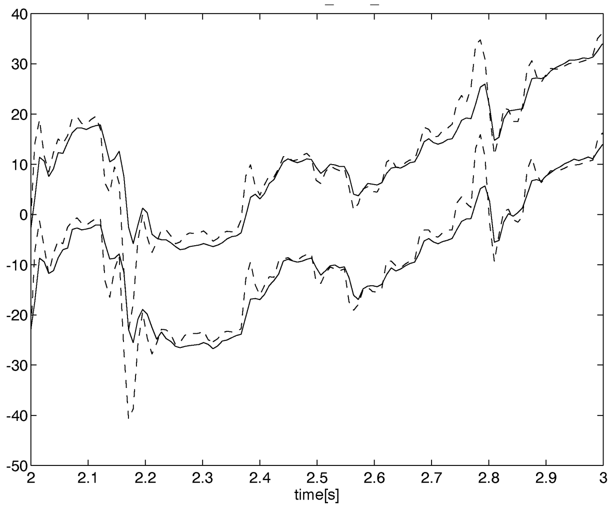

Fig. 4: Comparison: Simulation vs. real machine
Machine (shifted +10): —— ω_{real} – – ω_{Sim}
RBF-models (shifted –10): —— ω_{real} – – ω_{Sim}

Thus, two models had to be trained, one for the simulated and one for the real machine. Fig. 5 shows the input U_{Sq} (being the same for both models), and the output of the simulated machine. To receive training data for neural network training, the data has to contain information about the dynamics at each operating point. The behavior of the two models is shown in Fig. 4, lower curves. It can be seen that the models perform well on the training data. They were also verified on a different test set.

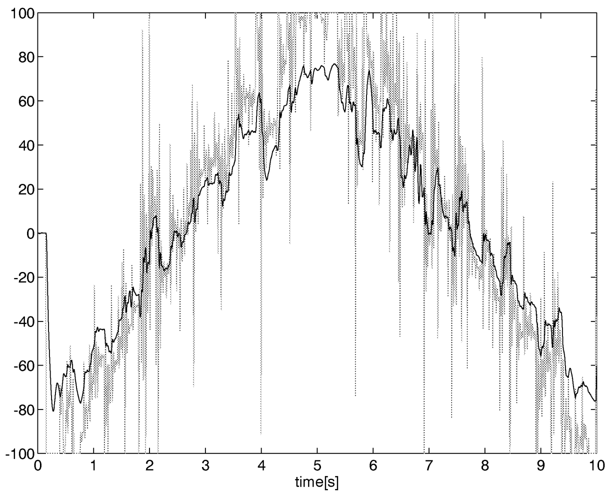

Fig. 5: Excitation used for modeling
—— ω —— U_{Sq}

Both models were trained offline using a Gaussian RBF with about 100 units in the hidden layer. The input vector was built containing past values of both speed and voltage. The centers were chosen by a clustering algorithm described in [Neu93].

3. Real Time Implementation Using a Transputer System

The implementation of the neural controller was done on Multi-Transputer hardware which has been developed for various research projects, e.g. [Bei92].

3.1 Hardware Structure

Fig. 6 shows the hardware system structure. In order to achieve a powerful and flexible system a software configurable Transputer system including 7 Transputers T800 and one T222 was used. The Transputer's serial communication links were connected according to the desired process communication scheme.

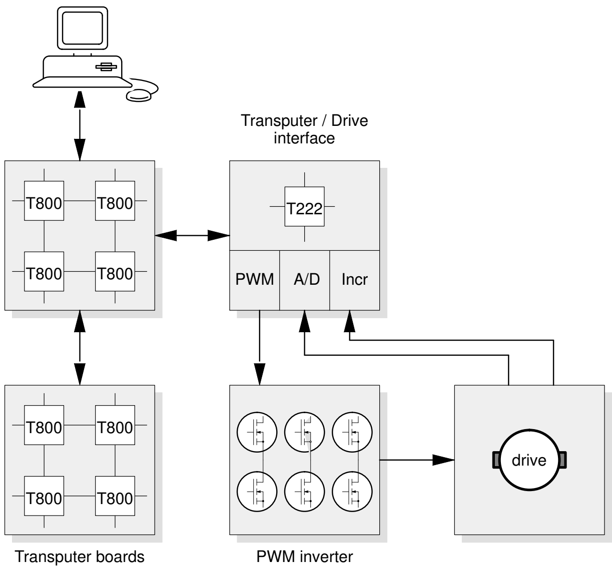

Fig. 6: The Multi-Transputer Hardware

The acquisition of analog and digital machine data is done by the T222-based I/O-board. It can be equipped with application specific I/O-modules. For this particular project the induction machine's speed is observed by an incremental encoder delivering a measure for the angular rotor position. By using a fourfold interpolation, a resolution of 10,000 increments per revolution is achieved. The speed is then calculated by differentiating the position with respect to time. Two 12-bit A/D-converters with a conversion time of 5µs are used to determine the two stator currents. The PWM signal is generated by two 16-bit timers and a programmable logic device. In order to achieve an almost sinusoidal shaped signal a PWM frequency of 15,625kHz was used.

The voltage source inverter is realized with six MOSFETs. They form a full-bridge that is connected to the three windings of the star connected machine. The MOSFET inverter is controlled by a digital pulse pattern which is generated according to the value of U_S using a three space-phasor method.

The machine is a 1.5kW three phase AC machine with four poles. The AC servo drive is coupled with a DC machine to be able to apply loads. The machine is characterized by the following parameters:

Nominal Power	P_N	1.5 [kW]
Nominal Voltage	U_N	380 [V]
Nominal Current	I_N	3.7 [A]
Nominal Torque	M_N	10 [Nm]
Number of pole pairs	z_p	2

3.2 Software Structure and Parallel Processing

The software graph of the complete neural control system is shown in Fig. 7. The system is booted and controlled by a PC. A window-based screen interface ensures comfortable user interaction. The window shell related processes are run on the "Shell"-node. The main process handling all the other parts of the software is called "Control". It is linked with the two control loops (Fig. 2), named "Flux" and "MPC". "MPC" works together with four parallel "RBF"-processes, each of them being a quarter of the total neural network.

The flux control path and other maintenance tasks were already implemented on this multi-transputer hardware as described in the work of Beierke et al. [Bei92]. The computation time for the RBF process on a single T800 transputer was 2 ms. Different simulations have shown that a prediction horizon of two to three time steps and at least four optimization iterations would be necessary in order to achieve satisfying control results. Under this assumption, the calculations of the predicted speed values would need up to 24 ms, which is far too long since a sampling period of 8 ms has to be obtained.

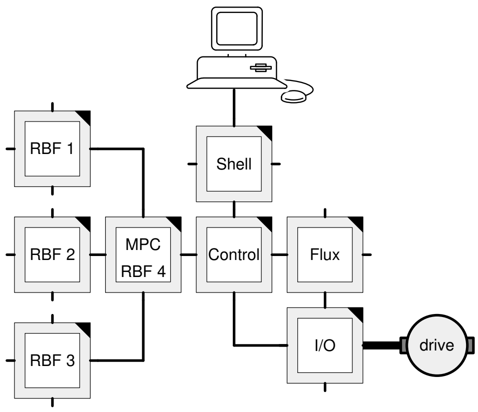

Fig. 7: Process graph showing the parallel computation of the RBF-network

The problem was solved by utilizing 4 transputers for the calculation of the 120 exponential functions in the Gaussian layer. It is obvious that this parallelization can be scaled down to one processor calculating one hidden unit output, since every hidden unit has the same input vector which can be supplied at the same time. The resulting calculation time would then be dominated by data exchange times.

In the implementation described here, each of the four T800 processors had to calculate 30 hidden Gaussians. The fourth processor was also running the whole MPC control process. This implementation resulted in an execution time of 7.8 ms allowing for a speed path control rate of 8 ms.

4. Results

Two results will illustrate the strength of the neural control approach. A comparison of NMPC with a field oriented control implementation is given in the following subsection. In 4.2 results of the real-time implementation will show that an exact model of the particular machine can further improve the control accuracy.

4.1 Simulations

The two control structures FOC and NMPC were compared on the same speed reference profile. For the use within the NMPC structure a model of the simulated machine had been trained. The sampling time T_s of the speed control loop was chosen to be 2 ms. Fig. 8 shows the step response of both controllers.

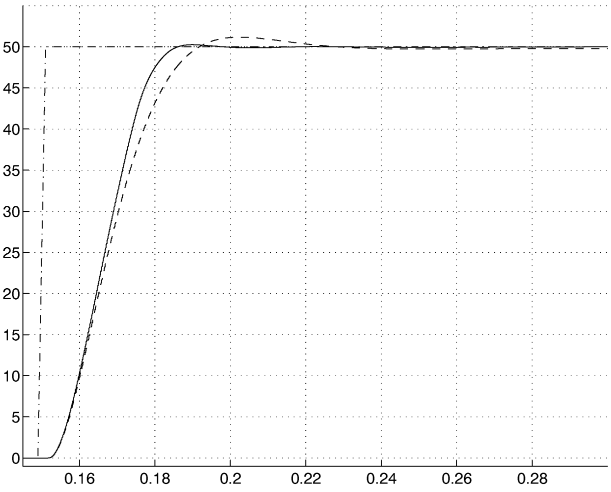

Fig. 8: Control results of NMPC vs. FOC
−·− ω_{ref} —— ω_{NMPC} − − ω_{FOC}

The NMPC shows clearly the behavior of a nonlinear controller. The risetime is shorter due to an almost constant slope of the trajectory. Although the controlled speed reaches the final value much faster, the overshoot compared to the linear controller is reduced.

Fig. 9 and Fig. 10 make the differences much clearer. Fig. 9 shows a zoomed part of Fig. 8. The shaded area indicates the band of 1% accuracy with respect to the step of speed change. NMPC reaches this band faster than FOC and settles within that accuracy almost immediately.

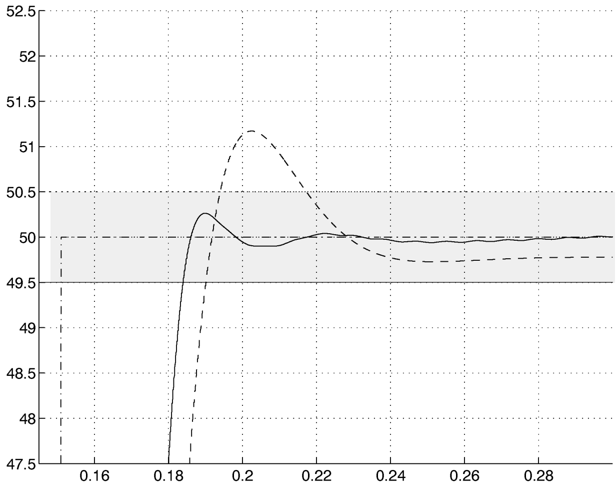

Fig. 9: Zoomed results with 1% band (grey)
−·− ω_{ref} —— ω_{NMPC} − - ω_{FOC}

Fig. 10 points out the reason for the fast but precise behavior of the neural controller. In the case of FOC, the control variable U_{Sq} decreases proportionally with the deviation between ω_{ref} and ω. NMPC holds U_{Sq} at the maximum value until the prediction indicates that the acceleration should be stopped. Thus, the predictive controller reduces U_{Sq} below the final value in order to prevent the drive from over shooting.

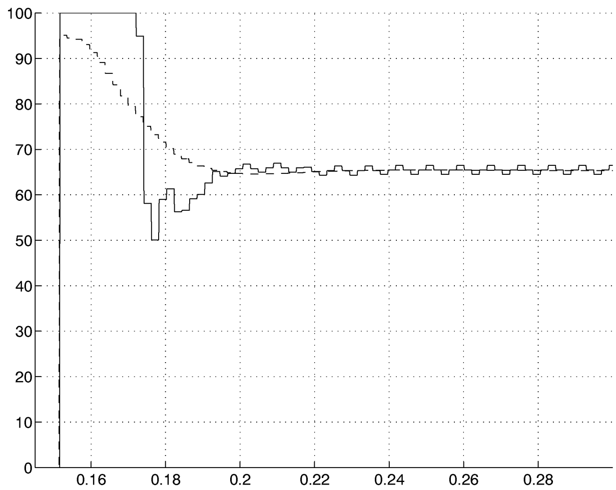

Fig. 10: Control results: control effort (simulation)
—— $U_{Sq\mathrm{NMPC}}$ − - $U_{Sq\mathrm{FOC}}$

Using simulations, it is easy to show the potential of a new control algorithm. The results presented here should show the differences between the linear field oriented controller and the nonlinear neural controller. In the following subsection some practical results will be given.

4.2 Real Machine

As mentioned earlier, the real-time implementation was targeted to achieve a control period of 8 ms. In addition to this limitation, the maximum number of iterations allowed for the optimization is limited as well. We investigated two different values for the prediction horizon length T_h, in particular two and three steps. For the two step version NMPC could interpolate up to six times. With a horizon of three steps only four iterations were possible.

The following plot shows the result of real-time NMPC with $T_h = 2$.

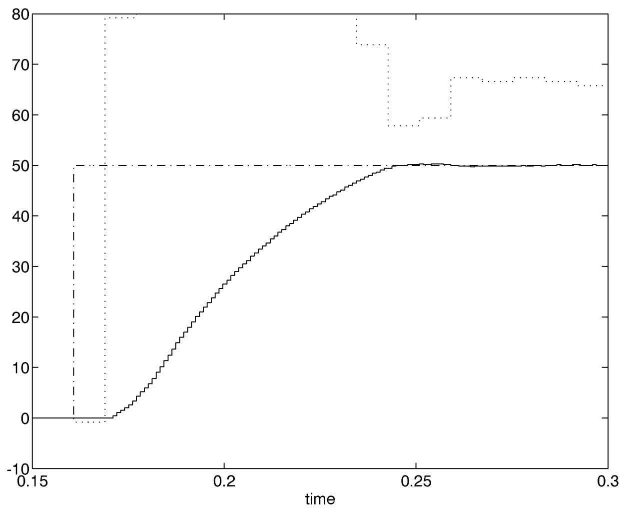

Fig. 11: Control results of real machine
−·− ω_{ref} —— ω_{NMPC} ···· $U_{Sq\mathrm{NMPC}}$

The result is very similar to that of Fig. 8 to Fig. 10, but due to the limitation of U_{Sq} of 80 V (instead of 100 V) the response is slower.

The next few plots show an interesting result indicating another advantage of the neural model based approach. As described in section 2.2, the behavior of the simulated and the real machine differ slightly, although the parameters were chosen the same as the real drive (compare [Bei92]). These results were obtained using a horizon of three steps. It should be noted that the following plots show revolutions per minute n instead of the speed ω.

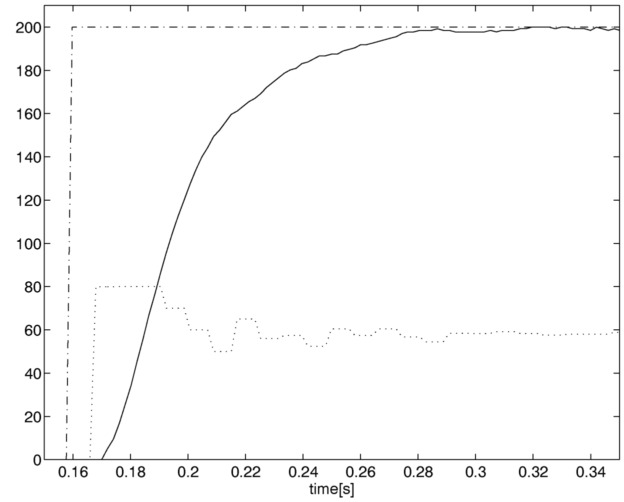

Fig. 12: NMPC with model trained on real machine data
−·− n_{ref} —— n - - U_{Sq}

Compared to the previous plot, the neural controller works more carefully. This is due to the fact that the prediction horizon has been extended. The speed reaches the desired value asymptotically without overshoot.

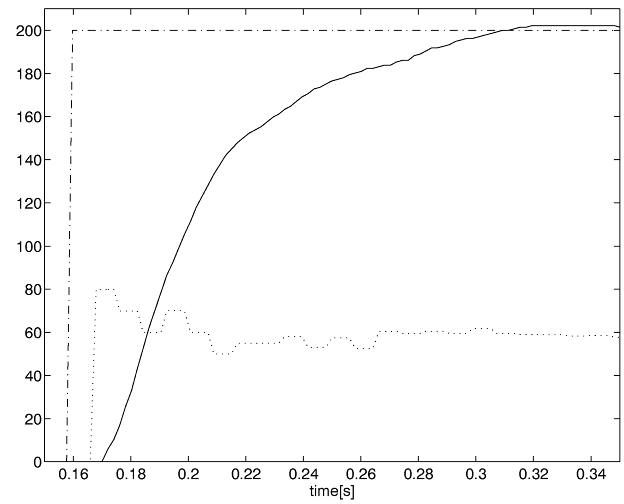

Fig. 13: NMPC with model trained on simulation data
−·− n_{ref} —— n - - U_{Sq}

The same controller with identical settings, but using a model trained on data from the simulated machine gave the performance shown in Fig. 13. The response is slower and also a small overshoot can be observed.

Since the simulated machine and the corresponding neural model were tested to be very similar (Fig. 4), one can expect that a predictive controller based on an analytical model of the machine would end up with the same accuracy. This indicates that both nonlinear MPC and neural network modeling contributed to the increased control precision.

Finally, NMPC has been tested on a different profile. The plot of Fig. 14 shows that the characteristic of the neural controller remains the same even for different operating points and step sizes.

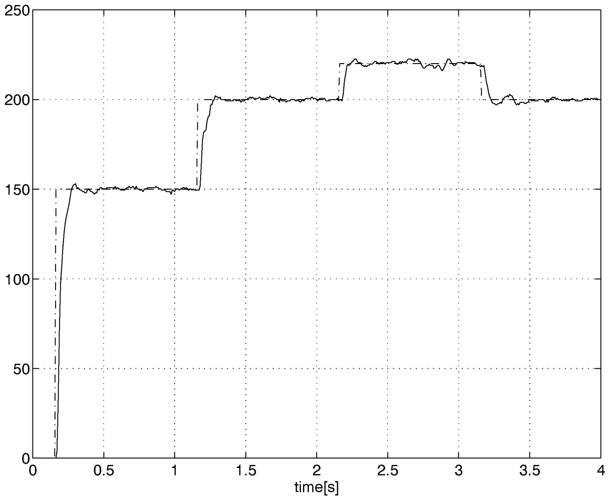

Fig. 14: MPC with different reference profile
$-\cdot-$ n_{ref} —— n - - U_{Sq}

5. Conclusions

Combining the techniques of neural network modeling with RBF neural nets and parallel processing hardware can provide fast, real-time nonlinear system models. This work shows that control strategies like MPC, which are usually limited to slow control applications, can now be exploited even for highly dynamic system control.

Further potential for faster control cycles can be exploited using improved versions of the simple RBF network which is smaller and therefore more efficient. Examples are Time-Delay RFB networks [Neu93] and Local Model Nets [Joh92].

The proposed control structure has been implemented and proven under real-world real-time conditions controlling an induction servo drive. The results achieved are very promising. Compared to field oriented control, MPC offered an optimal rise time paired with a minimized overshoot error.

The proposed optimization based on an interval search (IS) is limited to single-value optimization problems. The more general and state-of-the-art method, the Sequential Quadratic Programming algorithm (SQP) has been investigated. It has at least four times of the computing effort of the IS algorithm. But with future developments of micro processors, these more sophisticated methods can also be applied.

6. References

[Bar90] A.G. Barto: *Connectionist Learning for Control: An Overview*, 1990, Neural Networks for Control, edited by W.T. Miller, III, R.S. Sutton, and P.J. Werbos, The MIT Press

[Bei92] S. Beierke: *Vergleichende Untersuchungen von unterschiedlichen feldorientierten Lageregler-strukturen für Asynchron-Servomotoren mit einem Multi-Transputer-System*, PhD-thesis, TU-Berlin, 1992.

[Fra94] J. Franz: *Neural Model Predictive Control für die Asynchronmaschine*, Studienarbeit, TU-Berlin, 1994.

[Gar89] C.E. García, D.M. Prett, M. Morari: *Model Predictive Control: Theory and Practice – a Survey*, 1989, Automatica, vol. 25, no. 3

[Hid93] A. Hidiroglu: *Implementierung eines Model-Predictive-Controllers (MPC) zur Regelung einer Asynchronmaschine*, Studienarbeit, TU-Berlin, 1993.

[Joh92] T.A. Johansen, B.A. Foss: *A NARMAX Model Representation for Adaptive Control Based on Local Models*, 1992, Modeling, Identification and Control, vol. 13, no. 1

[Mur92] R. Murray-Smith, D. Neumerkel, D. Sbarbaro-Hofer: *Neural Networks for Modelling and Control of a Non-Linear Dynamic System*, 1992, Proceedings 7th IEEE International Symposium on Intelligent Control

[Neu93] D. Neumerkel, R. Murray-Smith, H. Gollee: *Modelling Dynamic Processes with Clustered Time-Delay Neurons*, 1993, Proceedings IJCNN-93-Nagoya International Joint Conference on Neural Networks

[Pog90] T. Poggio F. Girosi: *Networks for Approximation and Learning*, 1990, IEEE Proceedings, vol. 78, no. 9

[Psi90] D.C. Psichogios, L.H. Ungar: *Nonlinear Internal Model Control and Model Predictive Control Using Neural Networks*, 1990, Proceedings 5th IEEE International Symposium on Intelligent Control, vol. 2

[Sba93] D. Sbarbaro-Hofer, D. Neumerkel, K. Hunt: *Neural Control of a Steel Rolling Mill*, 1993, IEEE Control Systems, vol. 13, no. 3