Neural Network Contour Error Predictor in CNC Control Systems

Krystian Erwinski,
Marcin Paprocki, Andrzej Wawrzak
Faculty of Physics Astronomy and Informatics
Nicolaus Copernicus University, Torun, Poland
erwin@fizyka.umk.pl, zwirek@fizyka.umk.pl,
awawrzak@fizyka.umk.pl

Lech M. Grzesiak
Faculty of Electrical Engineering
Institute of Control and Power Electronics
Warsaw University of Technology
Warsaw, Poland
L.Grzesiak@isep.pw.edu.pl

***Abstract*—This article presents a method for predicting contour error using artificial neural networks. Contour error is defined as the minimum distance between actual position and reference toolpath and is commonly used to measure machining precision of Computerized Numerically Controlled (CNC) machine tools. Off-line trained Nonlinear Autoregressive networks with exogenous inputs (NARX) are used to predict following error in each axis. These values and information about toolpath geometry obtained from the interpolator are then used to compute the contour error. The method used for effective off-line training of the dynamic recurrent NARX neural networks is presented. Tests are performed that verify the contour error prediction accuracy using a biaxial CNC machine in a real-time CNC control system. The presented neural network based contour error predictor was used in a predictive feedrate optimization algorithm with constrained contour error.**

I. Introduction

Computerized Numerical Control (CNC) machines are widely used in manufacturing. In recent years much work has been done to improve motion control algorithms used in CNC machine control systems in order to decrease machining errors and machining time. To implement advanced motion control algorithms in CNC machine control systems accurate models of the machine's feed drive dynamics are often required. Models are usually used in the design phase in order to test the algorithms' performance. Some algorithms such as predictive control utilize the machine model directly in order to compute optimal values of the reference signal that minimize machining errors. Predictive models can also be used to generate time-optimal feedrate profiles without violating maximum machining error tolerance. Machining errors are often defined as contour errors which are the minimum distances between the reference toolpath and actual tool positions (fig. 1). Contour error is computed using following errors in each axis and local toolpath geometry. Obtaining accurate contour error predictions therefore requires accurate predictions of following errors in each axis. This can be performed by using a dynamical model of each axis.

Machine axis feed drive can be simply modelled as a first order model with inertia and viscous friction coefficient as it's parameters [1]. This basic model can be extended in order to model elastic coupling between axis elements [2], nonlinear friction [3], [4], backlash [5], thermal deformation [6]. Including all these phenomena leads to accurate but very complex models which are hard to identify due to large number of parameters.

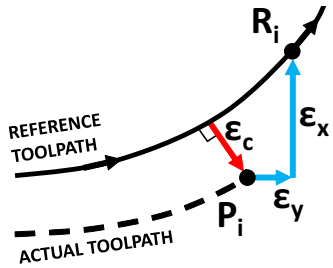

Fig. 1: Definition of contour error (ϵ_c) compared to axial following errors (ϵ_x, ϵ_y) computed between reference position (R_i) and actual position (P_i).

An alternative approach is to use a black-box input-output model. It is assumed that there is no knowledge about the inner workings of the system and its dynamics are modelled using only input and output data [7]. Such models can be identified using data collected during normal machine operation. Linear models are usually used such as ARX [8] or ARMAX [9]. Such models are easy to identify but cannot model nonlinearities in the feed drive.

Nonlinear relations are often modelled using artificial neural networks because of their universal approximation capabilities [10], [11]. Traditional feedforward neural network cannot model dynamical systems. Dynamic neural networks have to be used instead. One type that is commonly used to predict the outputs of nonlinear dynamical systems is the Nonlinear Autoregressive network with exogenous inputs (NARX) [12]–[15].

In this work NARX neural networks are used to model the dynamics of each axis' feed drive and predict each axis following error in response to axis velocity commands. Each NARX network is trained off-line using MATLAB software and then implemented in a real-time operating system (Linux RTAI) to test its predictive performance in an actual CNC control system. All networks are then combined to predict

contour error for a Non-Uniform Rational B-Spline (NURBS) toolpath in a biaxial system. Experimental results are then presented that show the performance of the neural network contour error predictor.

II. NARX NEURAL NETWORK FEED DRIVE MODEL

NARX neural networks are similar to traditional multi-layered perceptron feedforward networks (MLP). Both have layers of fully connected neurons with sigmoid activation functions in the hidden layers and linear activation function in the output layer. The main difference is the feedback path between network outputs and inputs and delay blocks in the input layer. This structure is very similar to the linear ARX model. A schematic of a NARX network used to predict axis following error is presented in figure 2. The network predicts the axis' following error using axial reference velocity as input.

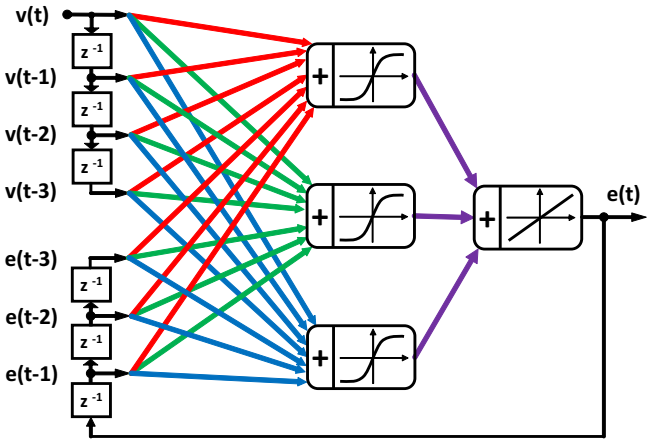

Fig. 2: Schematic of a NARX neural network with one hidden layer used to predict axis following error. $v(t)$ is the current axis velocity command and $e(t)$ is the predicted following error. Coloured arrows represent weights.

The predicted following error output of a single hidden layer NARX neural network can be computed using the following formula:

$$\begin{aligned} e(t) &= \sum_{i=1}^{H} \omega_i^o \tanh\left(h_i(t)\right) + b_o \\ h_i(t) &= \sum_{j}^{DV} \omega_{ij}^v v(t-j) + \sum_{k=1}^{DE} \omega_{ik}^e e(t-k) + b_i \end{aligned} \tag{1}$$

where: $v(t-j)$ - input axial velocities, $e(t-k)$ - feedback following errors, ω_i^o - output neuron weights, $\omega_{ij}^v, \omega_{ij}^e$ - hidden neuron weights, H - number of hidden neurons, DV - number of velocity input delays, DE - number of error feedback delays.

Artificial neural networks are usually trained by using two sets of data. One is the modelled system's input and the other is its output. The goal of the training is to iteratively adjust the networks weights and biases so that the network's output matches the systems actual output. A properly trained network has the ability of generalization which means that it can accurately predict the system's performance for any input data not just the training set. To test neural network generalization a validation input/output set is used that was not used for training.

NARX neural networks can be trained in two different ways in the series-parallel training architecture and parallel architecture [16]. In the series-parallel architecture (fig. 3a) the dynamic recurrent NARX network is trained as if it was a traditional static network [17]. The feedback path is disabled and each time delay is treated as an independent input. Feedback inputs are fed with the actual system's output data instead of data obtained from the network's output. When training is finished the trained network is reconfigured to work as a NARX network. Because the network used actual data for training and not predicted data which are used during operation the prediction error is larger with the feedback loop closed. Long term multi-step-ahead prediction which is often required for predictive control can exhibit large prediction errors.

In the parallel training architecture (fig. 3b) dynamic training algorithms have to be used such as back-propagation through time which are much more computationally demanding than ordinary algorithms for static networks. Training in the parallel architecture is also much more susceptible to premature convergence to a local minimum of the error function and training results are highly dependant on initial weights [18].

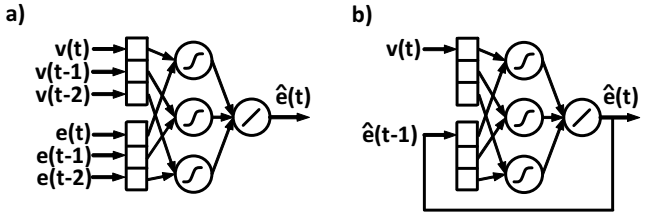

Fig. 3: Series-parallel training (a) and parallel training (b).

In this work a SISO NARX neural network was used to independently model each axis' feed drive and predict it's following error. Commanded axis velocity was used as the input value and axis following error was used as network's target output value. In order to generate the training data set was created from a randomly generated NURBS toolpath with optimized feedrate profile that ensured axial velocity profiles with maximum velocity (300 mm/s), acceleration (2000 mm/s^2) and jerk (50000mm/s^3). The NURBS toolpath is shown on figure 4 and axial reference velocity profiles are shown on figure 5.

This trajectory was executed on a biaxial CNC machine controlled by a PC based CNC controller developed by the authors [19]. The NURBS trajectory generator and interpolator were implemented in Linux RTAI real-time operating system along with data logging software. The experimental setup used is shown on figure 6.

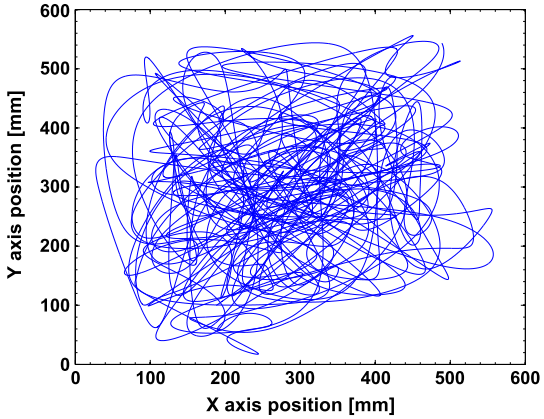

Fig. 4: Randomly generated NURBS toolpath used to generate training and validation trajectories

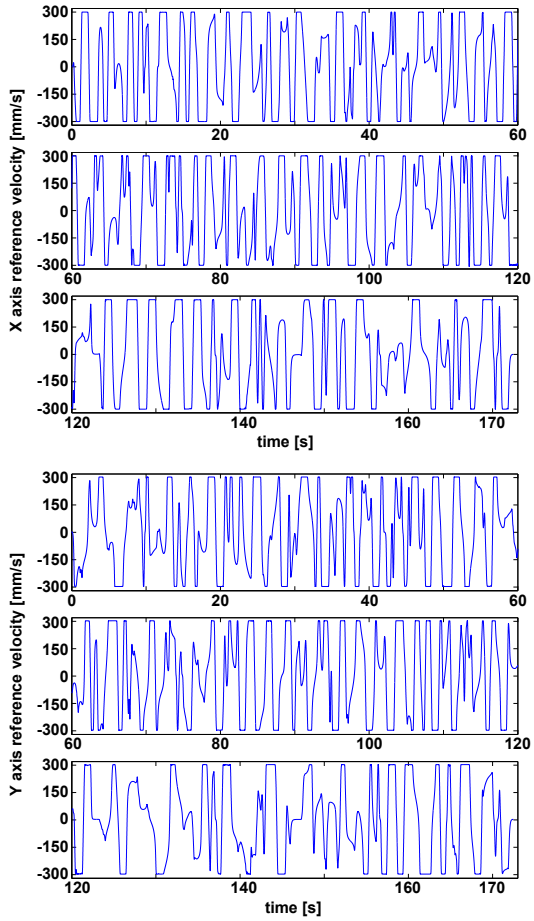

Fig. 5: Axial velocity profiles used as NARX network inputs for training and validation.

The reference trajectory was interpolated with 1ms sampling time. Following errors in each axis were saved to files and exported to MATLAB. The interpolation process was repeated 10 times and 10 output data sets with over 170000 samples were obtained. Following error in each set was slightly different due to machine repeatability errors so data sets were averaged. The

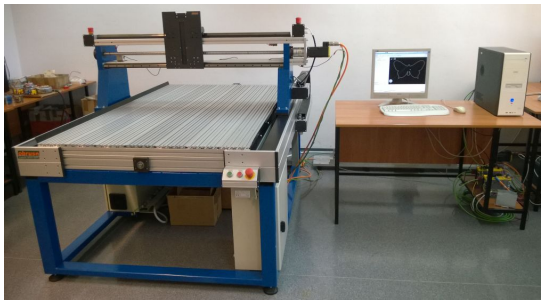

Fig. 6: Experimental setup with a biaxial CNC router used for testing

averaged data set was smoothed using MATLAB's "csaps" function with a smoothing factor of $5e^{-2}$ in order to remove measurement noise. Input velocity and output following error data sets were normalized so that their values were contained between -1 and 1. Preprocessed data sets for both axes were divided in half. First half was used for training the NARX neural networks and the other was used for validation.

Training was performed using MATLAB's "trainbr" algorithm which is the Levenberg-Marquardt training method with Bayesian Regularization [20], [21]. This algorithm minimizes an objective function that consists of a sum of squares of prediction errors and sum of squares of networks weights:

$$\epsilon_{BR} = \beta \sum_{i=1}^{N_d} (d_i - \hat{y}_i)^2 + \alpha \sum_{j=1}^{N_w} w_j^2 \tag{2}$$

where: N_d - number of training data samples, d_i - target data samples, $\hat{y}_i$ - network output data samples, N_w - number of network weights, w_j - network weights, α,β - weighting factors automatically adjusted by the training algorithm.
The aim of second part of the objective function is to prevent weights from increasing to large values which has a negative effect on generalization. Training with regularization usually takes longer than with standard LM algorithm but it's usually more robust to premature convergence to a local minimum and overtraining.

Training was performed in two stages. In the first stage a series-parallel configuration was used with randomly initialized weights. The network was trained until minimum gradient of $1e^{-8}$ was reached or the training did not show significant progress for a longer period of time. The network's feedback loop was then closed and training was continued in the parallel configuration without reinitializing the weights. Thanks to this method training resulted in a well trained network in every attempt. When attempting to train the network in the parallel configuration without pre-training many training runs resulted in networks that were unstable after several thousand time steps or had large prediction errors. Networks trained using only the series-parallel configuration usually had worse performance that those trained using the two stage method.

Different network architectures were tested by repeated training and verification using the validation set. During tests it was found that lowest following error prediction error for

each axis was obtained using a NARX neural network with a single hidden layer with 12 sigmoid neurons and 4 input and feedback delays. Graphical representation of the chosen NARX neural network architecture is shown on figure 7.

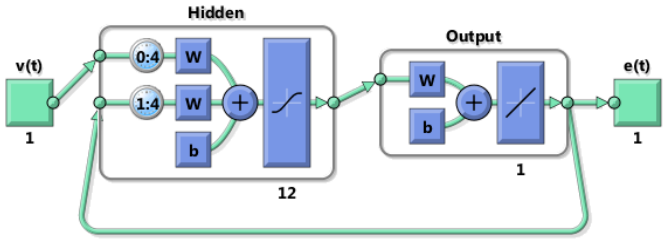

Fig. 7: Graphical representation of the chosen NARX following error predictor generated by MATLAB

NARX neural network training results were verified by comparing the predicted following error with actual following error for the validation set. Figures 8 and 9 show the validation set predicted (green) and actual following (blue) error for X and Y axes respectively. Prediction error which is the difference between predicted and actual following error values is shown on figures 10 and 11. There is a very close match between the two data sets. Main source of mismatch is due to ballscrew runout which is periodic in nature and is hard to model with any least squares training method. It is worth noting that the validation set prediction error does not increase with the number of samples predicted despite performing the prediction only in closed loop and predicting over 80000 samples. The NARX neural networks model the feed drive dynamics very well and are excellent long term predictors of axial following errors.

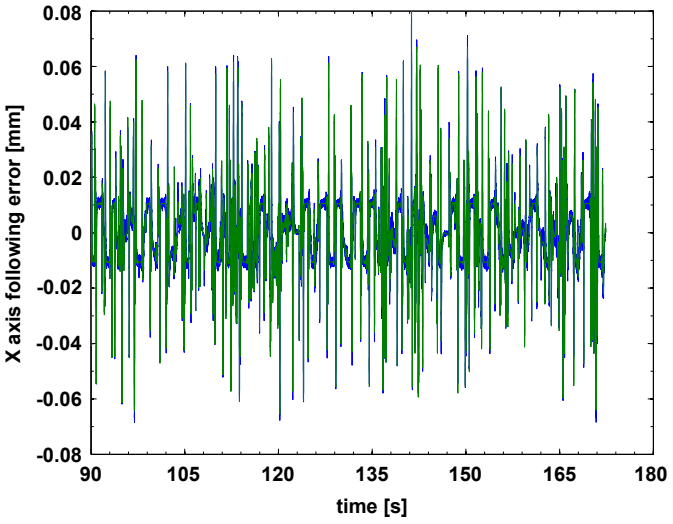

Fig. 8: Predicted (green) and actual (blue) following error in the X axis.

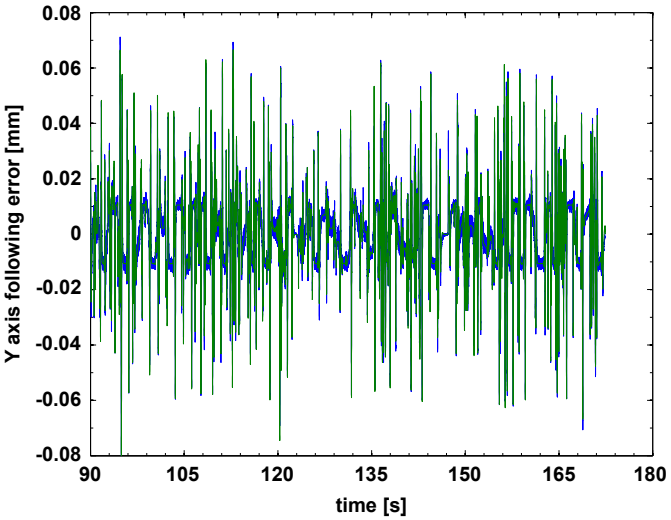

Fig. 9: Predicted (green) and actual (blue) following error in the Y axis.

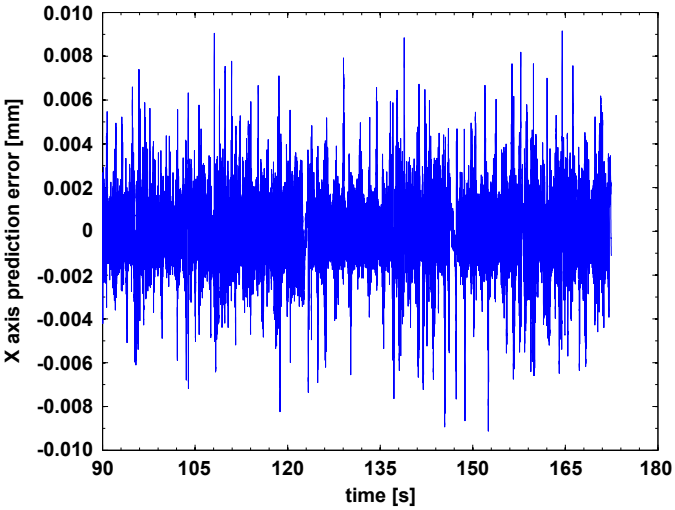

Fig. 10: Following error prediction error in the X axis.

III. Contour error prediction using NARX feed drive model

NARX neural networks trained using the method described above are excellent predictors of following error of each machine axis. Using knowledge about the toolpath geometry and predicted axial following errors contour error can be predicted. The block schematic of the contour error predictor is shown in figure 12.

When estimating contour error choosing the proper estimation method is as important as obtaining accurate following error predictions. For toolpaths defined as linear segments or circles computing contour error is simple and exact. In recent years toolpaths are commonly defined as Non-Uniform Rational B-Splines (NURBS) or other polynomial curves. This allows for representing complex shapes with relatively few data points. When using such toolpaths the contour error

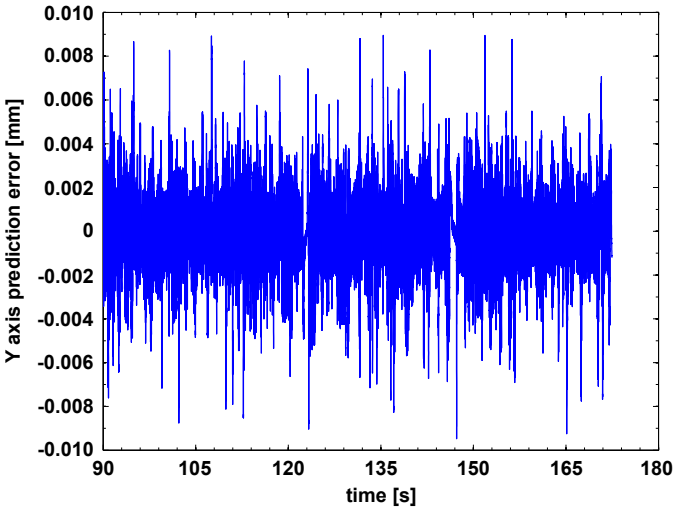

Fig. 11: Following error prediction error in the Y axis.

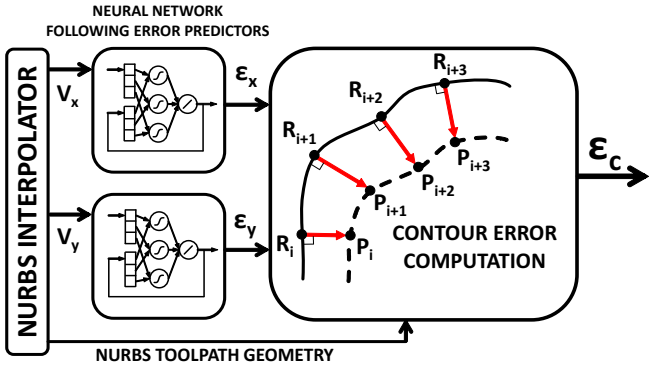

Fig. 12: Block schematic of the neural contour error predictor

cannot be computed exactly using any closed form formula [22]. Several contour error estimation techniques for free-form toolpaths have been proposed. A simple solution is to approximate the toolpath locally with a line [23] or circle [24]. Such approximation is usually only valid for toolpaths with low curvature and low feedrates. Other more advanced techniques were proposed [25]–[27] but are usually more complex and require additional data input. In one interesting algorithm proposed in [28] the contour error vector is approximated by a Taylor series which yields the following closed form formula:

$$\vec{\epsilon_c} = \left[-\vec{c} - \frac{1}{2} \frac{\kappa \left(\hat{c} \cdot \hat{n} \right) \left(\hat{t} \cdot \vec{\epsilon_t} \right) \hat{t}}{1 - \kappa \left(\hat{c} \cdot \hat{n} \right)} \right] \cdot \vec{\epsilon_t} \tag{3}$$

$$\hat{c} = -\frac{\vec{\epsilon_t} \cdot \hat{t}}{\sqrt{||\vec{\epsilon_t}||^2 - \vec{\epsilon_t} \cdot \hat{t}}} \hat{t} + \frac{1}{\sqrt{||\vec{\epsilon_t}||^2 - \vec{\epsilon_t} \cdot \hat{t}}} \vec{\epsilon_t} \tag{4}$$

where: κ - toolpath curvature at the reference point, $\hat{t}, \hat{n}$ - tangent and normal unit vectors at the reference point, $\vec{\epsilon_t}$ - following error vector. Curvature, tangent and normal vectors can be computed using the following formulas:

$$\kappa = \frac{||C'(u) \times C''(u)||}{||C'(u)||^3} \tag{5}$$

$$\begin{aligned} \hat{t} &= \frac{C'(u)}{||C'(u)||} \\ \hat{b} &= \frac{C'(u) \times C''(u)}{||C'(u) \times C''(u)||} \\ \hat{n} &= \frac{\hat{b}(u) \times C'(u)}{||\hat{b}(u) \times C'(u)||} \end{aligned} \tag{6}$$

where: $C'(u), C''(u)$ - are first and second derivatives of the NURBS toolpath position vector with respect to the toolpath parameter u obtained from the NURBS interpolator.

This method offers superior estimation accuracy compared to most other methods. Computation of the closed form formula requires only data obtained during interpolation of the NURBS curve and requires only a single iteration. Because of those reasons the method was chosen as the contour error estimation algorithm in the contour error predictor.

The contour error predictor was implemented in the real-time CNC control system and tested with the random NURBS toolpath previously used for training and validation. Predicted contour error was compared with actual contour error estimated using an iterative approximation method [29]. Actual and predicted contour error is presented in figure 13. The difference between predicted and actual values (prediction error) is presented in figure 14.

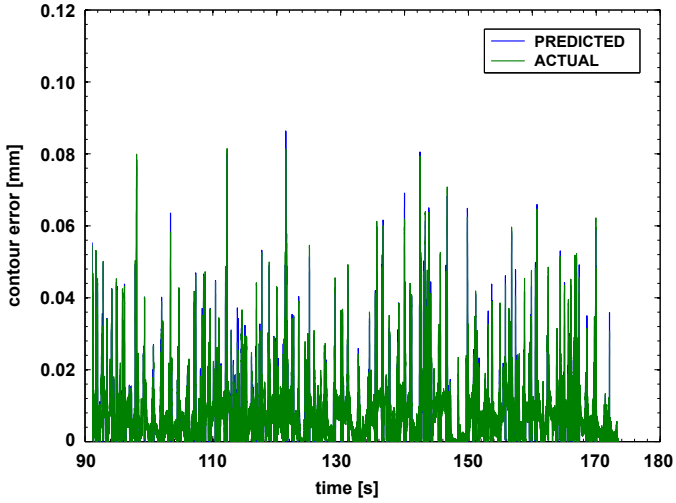

Fig. 13: Predicted (blue) and actual (green) contour error for the validation set

The contour error predictor predicts the actual contour error very well. Mismatch is mostly due to ball-screw runout which could not be modelled by the axial NARX neural networks.

IV. Conclusion

In this work a neural-network based contour error predictor for NURBS toolpaths was presented. Contour error was estimated from predicted following errors in each of the machine's

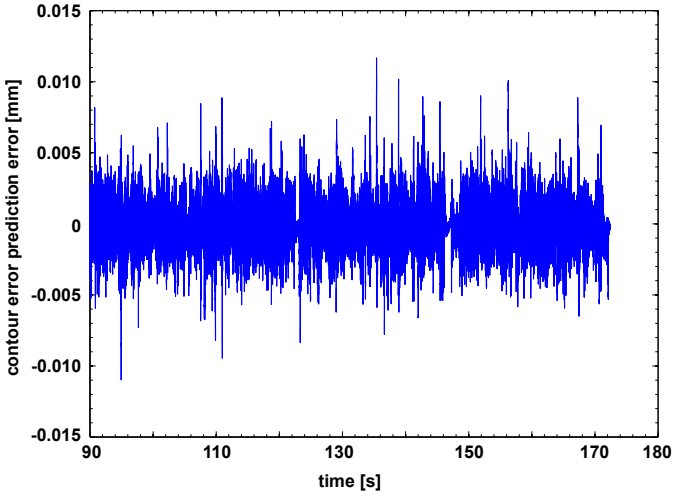

Fig. 14: Contour error prediction error for the validation set

axes. Axial following error was predicted using NARX neural networks. A two-stage NARX training method was presented that allowed to obtain useful nonlinear models of the machines feed drive. Tests were performed using a biaxial CNC machine. Presented results prove that the proposed solution can successfully predict following errors and contour error for long prediction horizons of over 80000 samples. This is crucial in many advanced motion control algorithms such as predictive control. The contour error estimation algorithm was used by the authors to develop a novel predictive feedrate optimization technique with contour error constraints presented in other works. Further work will be carried out to improve the neural network predictor in order to decrease the prediction error.

REFERENCES

[1] Y. Koren, *Computer control of manufacturing systems*. McGraw-Hill New York et al., 1983.

[2] M. Ebrahimi and R. Whalley, "Analysis, modeling and simulation of stiffness in machine tool drives," *Computers & industrial engineering*, vol. 38, no. 1, pp. 93–105, 2000.

[3] B. Armstrong-Hélouvry, P. Dupont, and C. De Wit, "A survey of models, analysis tools and compensation methods for the control of machines with friction," *Automatica*, vol. 30, no. 7, pp. 1083–1138, 1994.

[4] Z. Jamaludin, H. Van Brussel, G. Pipeleers, and J. Swevers, "Accurate motion control of xy high-speed linear drives using friction model feedforward and cutting forces estimation," *CIRP Annals-Manufacturing Technology*, vol. 57, no. 1, pp. 403–406, 2008.

[5] G. Younkin, "Modeling machine tool feed servo drives using simulation techniques to predict performance," *IEEE Transactions on Industry Applications*, vol. 27, no. 2, pp. 268–274, 1991.

[6] H. Dehnavi, M. Movahhedy, A. Naebi, and S. Pasban, "Prediction of the effect of heat generation in ballscrew on the accuracy of cnc milling machine," in *14th International Conference on Computer Modelling and Simulation (UKSim)*. IEEE, 2012, pp. 278–282.

[7] L. Ljung, *System Identification: Theory for the User,*. Prentice Hall, New Jersey,, 1999.

[8] D. Kim, D. H. Son, and D. Jeon, "Feed-system autotuning of a cnc machining center: Rapid system identification and fine gain tuning based on optimal search," *Precision Engineering*, vol. 36, no. 2, pp. 339–348, 2012.

[9] N. Tounsi, T. Bailey, and M. Elbestawi, "Identification of acceleration deceleration profile of feed drive systems in cnc machines," *International Journal of Machine Tools and Manufacture*, vol. 43, no. 5, pp. 441–451, 2003.

[10] A. Zain, H. Haron, and S. Sharif, "Prediction of surface roughness in the end milling machining using artificial neural network," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1755–1768, 2010.

[11] M. Nassef, C. Schenck, and B. Kuhfuss, "Simulation-based parameter identification of a reduced model using neural networks," in *9th IEEE International Conference on Control and Automation (ICCA)*. IEEE, 2011, pp. 974–978.

[12] A. Dzielinski, "Neural network-based narx models in non-linear adaptive control," *International Journal of Applied Mathematics and Computer Science*, vol. 12, no. 2, pp. 235–240, 2002.

[13] M. Correa, C. Bielza, and J. Pamies-Teixeira, "Comparison of bayesian networks and artificial neural networks for quality detection in a machining process," *Expert systems with applications*, vol. 36, no. 3, pp. 7270–7279, 2009.

[14] F. Huo and A.-N. Poo, "Nonlinear autoregressive network with exogenous inputs based contour error reduction in cnc machines," *International Journal of Machine Tools and Manufacture*, vol. 67, pp. 45–52, 2013.

[15] G. Kant and K. S. Sangwan, "Predictive modelling and optimization of machining parameters to minimize surface roughness using artificial neural network coupled with genetic algorithm," *Procedia CIRP*, vol. 31, pp. 453–458, 2015.

[16] K. Narendra and K. Parthasarathy, "Learning automata approach to hierarchical multiobjective analysis," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 21, no. 1, pp. 263–272, 1991.

[17] H. Xie, H. Tang, and Y. Liao, "Time series prediction based on narx neural networks: An advanced approach," in *2009 International Conference on Machine Learning and Cybernetics*, vol. 3. IEEE, 2009, pp. 1275–1279.

[18] B. Horne and C. Giles, "An experimental comparison of recurrent neural networks," *Advances in neural information processing systems*, pp. 697–704, 1995.

[19] K. Erwinski, M. Paprocki, L. Grzesiak, K. Karwowski, and A. Wawrzak, "Application of ethernet powerlink for communication in a linux rtai open cnc control system," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 2, pp. 628–636, 2013.

[20] D. MacKay, "A practical bayesian framework for backpropagation networks," *Neural computation*, vol. 4, no. 3, pp. 448–472, 1992.

[21] D. Foresee and M. Hagan, "Gauss-newton approximation to bayesian learning," in *Neural Networks, 1997., International Conference on*, vol. 3. IEEE, 1997, pp. 1930–1935.

[22] N. Uchiyama *et al.*, "Contouring controller design based on iterative contour error estimation for three-dimensional machining," *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 4, pp. 802–807, 2011.

[23] S. Yeh and P. Hsu, "Estimation of the contouring error vector for the cross-coupled control design," *IEEE/ASME Transactions on Mechatronics*, vol. 7, no. 1, pp. 44–51, 2002.

[24] Y. Koren and C. Lo, "Variable-gain cross-coupling controller for contouring," *CIRP Annals-Manufacturing Technology*, vol. 40, no. 1, pp. 371–374, 1991.

[25] F. Huo, X.-C. Xi, and A.-N. Poo, "Generalized taylor series expansion for free-form two-dimensional contour error compensation," *International Journal of Machine Tools and Manufacture*, vol. 53, no. 1, pp. 91–99, 2012.

[26] B. Sencer, Y. Altintas, and E. Croft, "Modeling and control of contouring errors for five-axis machine tools. part i: Modeling," *Journal of manufacturing science and engineering*, vol. 131, no. 3, 2009.

[27] X. Chen, J. Yong, G. Wang, J. Paul, and G. Xu, "Computing the minimum distance between a point and a nurbs curve," *Computer-Aided Design*, vol. 40, no. 10, pp. 1051–1054, 2008.

[28] L. Zhu, H. Zhao, and H. Ding, "Real-time contouring error estimation for multi-axis motion systems using the second-order approximation," *International Journal of Machine Tools and Manufacture*, vol. 68, no. 5, pp. 75–80, 2013.

[29] H. Song, X. Xu, K. Shi, and J. Yong, "Projecting points onto planar parametric curves by local biarc approximation," *Computers & Graphics*, vol. 38, pp. 183–190, 2014.