Neural Adaptive Control of a Robot Joint Using Secondary Encoders

Jonas Weigand, Magnus Volkmann, and Martin Ruskowski

Technische Universität Kaiserslautern,
Lehrstuhl für Werkzeugmaschinen und Steuerungen, Germany,
jonas.weigand@mv.uni-kl.de,
https://www.mv.uni-kl.de/en/wskl/

Abstract. Using industrial robots for machining applications in flexible manufacturing processes lacks a high accuracy. The main reason for the deviation is the flexibility of the gearbox. Secondary Encoders (SE) as an additional, high precision angle sensor offer a huge potential of detecting gearbox deviations. This paper aims to use SE to reduce gearbox compliances with a feed forward, adaptive neural control. The control network is trained with a second network for system identification. The presented algorithm is capable of online application and optimizes the robot accuracy in a nonlinear simulation.

Keywords: Robot control, secondary encoders, neural adaptive control

1 Introduction

Using industrial robots in machining processes offers great potential regarding process flexibility while remaining low cost [1], [2], [3]. The main drawback of robots compared to machining tools is the lack of Tool-Center-Point (TCP) accuracy. One way of improving this accuracy is using secondary encoders (SE) on the main axes of the robot in order to reduce gearbox compliances [4], [5]. [6] has shown, that SE are able to improve robot accuracy in a cascade control using stiff control parameters and optimizing SE for disturbance compensation only. In order to overcome the drawbacks of a stiff cascade control configuration, the guiding behaviour of the axes needs to be improved by a feed forward control. However, a model based control needs good model knowledge, which is difficult to obtain. [2] shows that especially the friction model of the gearbox is complicated, e.g. because the gearbox temperature changes more than $80^{\circ}C$ during machining. Furthermore, a good control needs additional sensor information, which are usually not available on standard robots and imply a costly update. Therefore, this paper aims to overcome the drawbacks of bad model knowledge by using black-box system identification and adjusting the control accordingly. Due to nonlinear physical effects of the robot, e.g. gearbox friction and backlash, the algorithm has to be capable of nonlinear system identification. In addition, the algorithm has to be ready for online usage.

This paper is organized as follows. First, a concept - neural adaptive control - is introduced and explained. Next, system identification using Runge-Kutta Neural Networks (RKNN) for nonlinear modelling is presented. Afterwards, a Multi-Layer-Perceptron (MLP)

as feed forward controller is shown. Finally, a nonlinear simulation model of the robot axis is presented and simulation results using the proposed algorithm are demonstrated.

2 Concept

An overview of neural adaptive control is given in [7], [8]. The core idea used in this paper is to separate the tasks of system identification and controller design. The signal concept used in this paper is shown in Fig. 1. A controller network $c(\cdot)$ is used to modify the feed forward reference trajectory r, which in our case is the arm reference angle φ_R. Depending on the reference, the controller creates a modified reference arm angle $u \mathrel{\widehat{=}} \varphi_U$. Before the controller is trained φ_R is equal to φ_U. Using an unmodified reference as input is possible because a closed loop system is utilized. For training the controller in a simulation environment, a model of the measured system is needed. In a comparative study [9] of multi-axis robot system identification, RKNN achieve a high prediction accuracy while remaining a good generalization. RKNN are explained in [10] and combine neural networks with the Runge-Kutta differential equation solver. The RKNN is trained to approximate the measured arm angle $y \mathrel{\widehat{=}} \varphi_A$ with the simulated arm angle $\hat{y} \mathrel{\widehat{=}} \hat{\varphi}_A$ as good as possible.

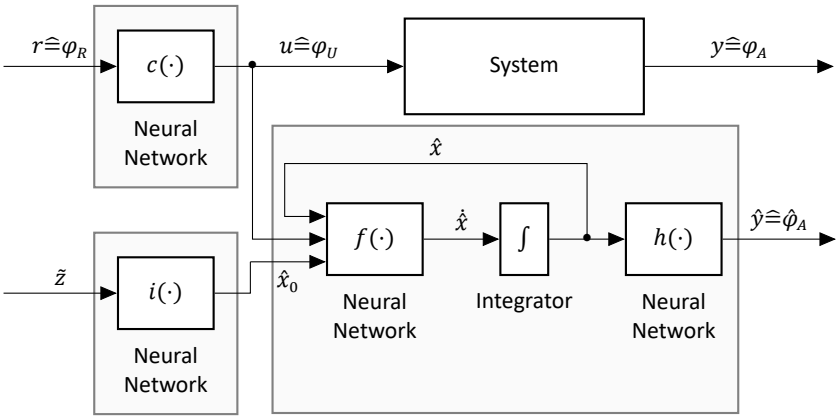

Fig. 1. Complete control structure. A RKNN is used for online black box system identification. Using the RKNN, a feed forward controller network is trained and used online.

3 Runge-Kutta-Neural-Networks

For the state network $f(\cdot)$, the output network $h(\cdot)$ and the initial network $i(\cdot)$, MLPs with a single hidden layer are applied [11]. Using a MLP, we obtain the network equations

$$\dot{\hat{x}}(t) = f(\cdot) = W_F^o \sigma(W_{FX}^h \hat{x}(t) + W_{FU}^h u(t) + b_F^h) + b_F^o \tag{1}$$

$$\hat{y}_k = h(\cdot) = W_H^o \sigma(W_{HX}^h \hat{x}_k + b_H^h) + b_H^o \tag{2}$$

$$\hat{x}_0 = i(\cdot) = W_I^o \sigma(W_{IZ}^h \tilde{z}_k + b_I^h) + b_I^o \tag{3}$$

with the continuous state vector $x(t)$, the estimated discrete state vector $\hat{x}_k$ at time step k, input u_k and estimated output $\hat{y}_k$. The activation function $\sigma(\cdot)$ is a hyperbolic tangent. Weight matrices and bias terms use the following index convention. Low-indices F, H and I are used for $f(\cdot)$, $h(\cdot)$ and $i(\cdot)$ network respectively. Low-indices X, U and Z represent state $\hat{x}$ signal, input u signal and regression vector $\tilde{z}_k$ respectively. The regression vector consists of a measured input and output sequence $\tilde{z}_k = [u_{k-N_1}, u_{k-N_1+1}, \ldots, u_{k-1}, y_{k-N_2}, y_{k-N_2+1}, \ldots, y_{k-1}]$ with $N_1 \in \mathbb{N}^+$ and $N_2 \in \mathbb{N}^+$. High-indices o and h denote output and hidden layer respectively. For a compact notation, all model network weights and bias terms are summarized in $\Theta_M = [W_I^o, W_{IR}^h, b_I^h, b_I^o, W_F^o, W_{FX}^h, W_{FU}^h, b_F^h, b_F^o, W_H^o, W_{HX}^h, b_H^h, b_H^o]$. Note that (1) is continous time, while (2) and (3) are discrete time equations. Calculating (1) requires a solution of the differential equation. This is done by using the 4-stage Runge-Kutta algorithm [10]. Incorporating the RK algorithm leads to

$$\begin{aligned}
v_1 &= W_F^o \sigma\left(W_{FX}^h \hat{x}_k + W_{FU}^h u_k + b_F^h\right) + b_F^o \\
v_2 &= W_F^o \sigma\left(W_{FX}^h \left(\hat{x}_k + \frac{h}{2} v_1\right) + W_{FU}^h u_k + b_F^h\right) + b_F^o \\
v_3 &= W_F^o \sigma\left(W_{FX}^h \left(\hat{x}_k + \frac{h}{2} v_2\right) + W_{FU}^h u_k + b_F^h\right) + b_F^o \\
v_4 &= W_F^o \sigma\left(W_{FX}^h \left(\hat{x}_k + h v_3\right) + W_{FU}^h u_k + b_F^h\right) + b_F^o \\
\hat{x}_{k+1} &= \hat{x}_k + \frac{h}{6} v_1 + \frac{h}{3} v_2 + \frac{h}{3} v_3 + \frac{h}{6} v_4
\end{aligned} \tag{4}$$

with the time step length h. The input u_k is assumed to be constant during one time interval. The RKNN is trained using *fmincon* from the *MATLAB optimization toolbox* with the SQP algorithm. The cost function for training is

$$\arg\min_{\Theta_M} \left(\left(\sum_{j=k-N_3+\max(N_1,N_2)}^{k} (y_j - \hat{y}_j)^2 \right) + \lambda_1 \cdot \Theta_M^2 \right) \tag{5}$$

using (2), (3) and (4) for calculating $\hat{y}_k$. The regularisation parameter is λ_1. The number of time steps used for training the model is $N_3 \in \mathbb{N}^+$ with $N_3 \gg N_1$ and $N_3 \gg N_2$.

4 MLP Controller

In [12], a chaotic Lur'e Problem is solved using a similar concept. Based on that work, a MLP is chosen feed forward control network $c(\cdot)$ and is defined by

$$u_k = c(\cdot) = W_C^o \sigma(W_{CS}^h \tilde{s}_k + b_C^h) + b_C^o \tag{6}$$

with the partial reference trajectory $\tilde{s}_k = [r_k, r_{k+1}, ..., r_{k+N_4}]$ with an arbitrary number of future time steps $N_4 \in \mathbb{N}^+$. For a compact notation, all weights and bias terms of network $c(\cdot)$ are summarized in $\Theta_C = [W_C^o, W_{CS}^h, b_C^h, b_C^o]$. The optimization is also done by the *MATLAB optimization toolbox*, with the cost function

$$\arg\min_{\Theta_C} \left(\left(\sum_{j=k}^{k+N_5-N_4} (r_j - \hat{y}_j)^2 \right) + \lambda_2 \cdot \Theta_C^2 + \left(\lambda_3 \cdot \sum_{j=k}^{k+N_5-1-N_4} (u_{j+1} - u_j)^2 \right) \right) \tag{7}$$

and the weighting factors λ_2 and λ_3. The simulated output $\hat{y}_k$ is calculated using (2), (3) and (4). The input u_k for the RKNN is calculated with (6). The number of time steps used for training the controller is $N_5 \in \mathbb{N}^+$ with $N_5 \gg N_4$.

5 Simulation Model

The presented algorithm is tested with a closed loop nonlinear model of a single robot joint. Due to the longest lever, axis 2 of a *KUKA Quantec Ultra SE* is chosen. The complete model is presented in [6] and this paper will cover a summary only. The open loop robot joint model is a two-mass-oscillator including nonlinear stiffness and nonlinear friction. Fig. 2 shows the model with the physical relations described in Tab. 1. The real robot is pictured in Fig. 3.

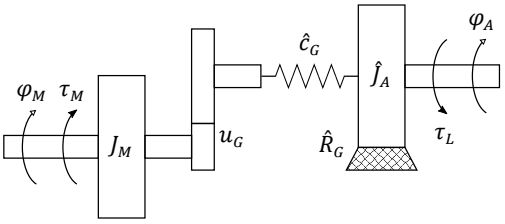

Fig. 2. Model of the robot joint as two-mass-oscillator

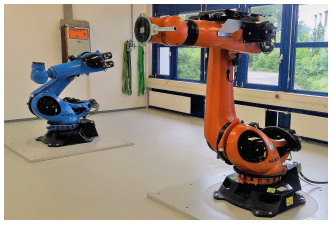

Fig. 3. KUKA Quantec Ultra SE

The motor torque serves as actuating variable and generates an acceleration of the motor inertia which rotates the gearbox. The gearbox, acting as a nonlinear spring, encounters the motor torque, overcomes the gearbox friction and accelerates the arm inertia. Using the model in Fig. 2, the nonlinear model can be obtained by calculating the sum of torques around motor and arm inertia with the friction torque $\hat{\tau}_F$ and the elastic spring torque $\hat{\tau}_E$.

$$J_M \cdot \ddot{\varphi}_M = \tau_M - \frac{\hat{\tau}_E}{u_G} \tag{8}$$

$$J_A \cdot \ddot{\varphi}_A = -\tau_L - \hat{\tau}_F + \hat{\tau}_E \tag{9}$$

Table 1. Robot model variables and parameters

description	type	symbol	value	unit
motor torque	state variable	τ_M		Nm
motor angle	state variable	φ_M		rad
motor speed	state and output variable	$\dot{\varphi}_M$		rad/s
arm angle	state and output variable	φ_A		rad
arm speed	state variable	$\dot{\varphi}_A$		rad/s
load torque	distortion variable	τ_L		Nm
modified reference arm angle	input variable	φ_U		rad
reference arm angle	guiding variable	φ_R		rad
motor inertia	parameter	J_M	0.0145	kgm^2
arm inertia	parameter	J_A	1092	kgm^2
gearbox ratio	parameter	u_G	203.52	$-$
backlash angle	parameter	ϕ_B	$0.15 \cdot 10^{-3}$	rad
lost-motion angle	parameter	ϕ_{LM}	$0.29 \cdot 10^{-3}$	rad
offset torque	parameter	$\tau_{E,0}$	210	Nm
lost-motion stiffness	parameter	c_{LM}	$1.44 \cdot 10^6$	Nm/rad
linear stiffness	parameter	c_G	$8.94 \cdot 10^6$	Nm/rad
coulomb friction	parameter	$\tau_{C,0}$	500	Nm
viscous friction	parameter	R_G	764	Nms/rad
proportional speed gain	parameter	P_1	2	Nms/rad
integral speed gain	parameter	I_1	16	Nms^2/rad
proportional position gain	parameter	P_2	8192	1/s

Motor friction as well as all electro-mechanical effects are neglected. Since the *KUKA Quantec* has a hydraulic counterweight acting on axis 2, both counterweight and gravity effects are neglected in this work. Temperature effects that have an influence on the friction are neglected, too, and require further research.
The stiffness is modelled with backlash, lost-motion, and linear elasticity. Lost-motion describes an effect in between backlash and linear elasticity, where not all tooth flanks are yet in full contact. During backlash τ_E equals 0. In the lost-motion range, the stiffness is modelled linear with a lower slope and an offset. In the full-contact range, the stiffness is modelled as a linear function with an offset. The stiffness coefficients are defined as c_{LM} and c_G, for lost-motion and full-contact respectively. We further define the angular ranges ϕ_B and ϕ_{LM}, for backlash and lost-motion respectively. With the torsion angle

$$\Delta\varphi = \varphi_M/u_G - \varphi_A \tag{10}$$

the following condition is defined.

$$\hat{\tau}_E = \begin{cases} 0 & \text{for } |\Delta\varphi| \leq \phi_B \\ c_{LM}\,\Delta\varphi - c_{LM}\,\dfrac{\phi_B}{2}\,\text{sign}(\Delta\varphi) & \text{for } \phi_B < |\Delta\varphi| \leq \phi_{LM} \\ c_G\,\Delta\varphi + \tau_{E,0}\,\text{sign}(\Delta\varphi) & \text{for } \phi_{LM} < |\Delta\varphi| \end{cases} \tag{11}$$

The gearbox friction is modelled as coulomb and viscous friction. According to [2], [13], and [14] the Stribeck effect can be neglected. With the linear friction constant R_G and the coulomb friction torque $\tau_{C,0}$ the friction conditions for the model are defined.

$$\hat{\tau}_F = \begin{cases} 0 & \text{for} \quad \dot{\varphi}_A = 0 \\ R_G\,\dot{\varphi}_A + \tau_{C,0}\,\text{sign}(\dot{\varphi}_A) & \text{otherwise} \end{cases} \tag{12}$$

In order to augment the open loop model of the axis to a closed loop model, a feed back controller is introduced. This cascade controller is configured as P-PI-controller

$$\tau_M = \left(P_1 + \frac{I_1}{s}\right) \cdot \left(P_2 \cdot (-\varphi_A + \varphi_R) - \dot{\varphi}_M\right). \tag{13}$$

with the Laplace transform variable s and the control parameters P_1, I_1 and P_2. The controller uses both, the measured arm angle φ_A obtained by the SE and the motor speed $\dot{\varphi}_M$. The modified arm reference angle φ_R is the input whereas the measured arm angle φ_A is the output of the closed loop system.

6 Simulations Results

The main result of this paper is presented in Fig. 6. The simulation is divided into three subsections. First, from 0 to $1\,s$, the input-output relation of the system is only measured and the unmodified reference is used as input. Then, a RKNN is trained for 500 iterations. Using the RKNN, a controller network is trained for 100 iterations. Finally, from 1.5 to 2.5 seconds, the controller network is used online.

The training result of the RKNN is shown in Fig. 6. With a Maximum-Error of $0.016\,°$, the RKNN achieves $96\,\%$ accuracy with respect to the angle amplitude of $0.52\,°$. Unlike other training algorithms, the data set is not split into a training and validation set. This is for simplicity only. A second data set for validation would require the effort of measuring a second, physical trajectory. However, a validation data set implies the advantages of a better evaluation of the trained network. Since the training data set of the RKNN needs to cover the complete system dynamics, a rather long data set is chosen. To shorten calculation time, the number of time steps in the training data set are reduced by an interpolation. Therefore, the step length for training can be unequal to the step length of data logging. The training of the control network is done on both, random created trajectories and the unmodified reference trajectory. Using different trajectories improves the generality of the controller network.

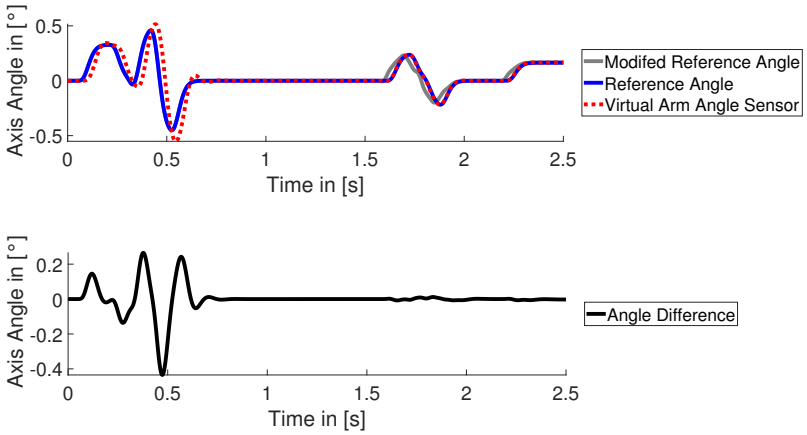

Fig. 4. Training and test of proposed algorithm

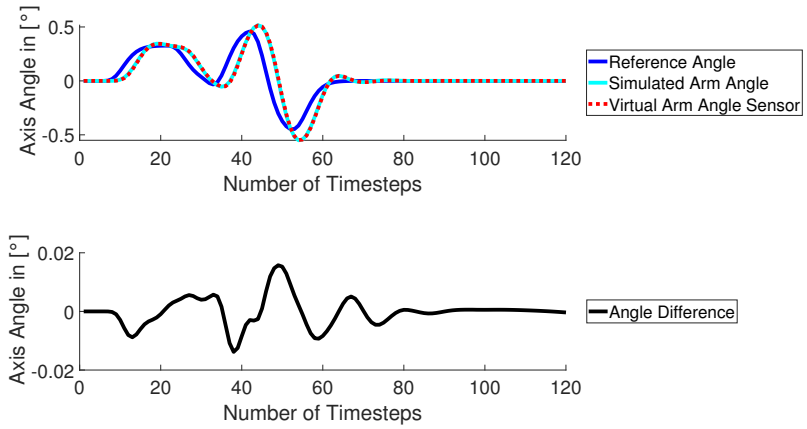

Fig. 5. Reference and system identification

7 Conclusion

All in all, the proposed algorithm achieves a Root-Mean-Square-Error between the reference and actual arm angle of $0.0026\,°$ and a Maximum-Error of $0.012\,°$. With respect to the Maximum-Error without this algorithm of $0.121\,°$, this is an improvement of $90\,\%$. The simulation results prove that it is possible to define the robot joint as a black-box and to optimize it with the help of RKKN without physical model knowledge. The algorithm achieves to overcome the challenges of backlash, lost-motion and nonlinear stiffness in simulation.

Further research is required how stability constrains can be incorporated. Only with this advancement, the network controller could be designed as feed back controller and an open loop model could be applied. As a starting point, the NLq Theory [15] presents a unified framework for neural adaptive control with global asymptotic stability criteria. Furthermore, the continuous online update of both, the model and the control network, need further research. Finally, the algorithm needs to be implemented and tested on the real robot.

Bibliography

[1] Eberhard Abele, editor. *Spanende Bearbeitung mit Industrierobotern: Forschungsprojekt ADVOCUT - Entwicklungen und Industrietransfer; Abschlußbericht BMBF-Verbundprojekt*. Meisenbach, Bamberg, 2007.

[2] Micheal Thümmel. *Modellbasierte Regelung mit nichtlinearen inversen Systemen*. Dissertation, Technische Universität München, 2006.

[3] Oliver Rösch. *Steigerung der Arbeitsgenauigkeit bei der Fräsbearbeitung metallischer Werkstoffe mit Industrierobotern*. Dissertation, Technische Universität München, 2015.

[4] Eckart Uhlmann, Sascha Reinkober, and Tobias Hollerbach. Erhöhung der genauigkeit beim fräsen mit industrierobotern. *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb*, 113(9):576–579, 2018.

[5] Russell Devlieg. High-accuracy robotic drilling/milling of 737 inboard flaps. *SAE International Journal of Aerospace*, 4(2):1373–1379, 2011.

[6] Stephan Belz. *Nichtlineare Modellierung, flachheitsbasierte Vorsteuerung und abtriebsseitige Regelung eines Robotergelenkes*. Studienarbeit, Technische Universität Kaiserslautern, 2018.

[7] Martin Hagan and Howard Demuth. Neural networks for control. *Proceedings of the American Control Conference*, 1999.

[8] M. M. Polycarpou. Stable adaptive neural control scheme for nonlinear systems. *IEEE Transactions on Automatic Control*, 41(3):447–451, 1996.

[9] M. Efe and O. Kaynak. A comparative study of neural network structures in identification of nonlinear systems. *Mechatronics*, 1998.

[10] Y. J. Wang and C. T. Lin. Runge-kutta neural network for identification of dynamical systems in high accuracy. *IEEE transactions on neural networks*, 9(2):294–307, 1998.

[11] Michael Deflorian. *Versuchsplanung und Methoden zur Identifikation zeitkontinuierlicher Zustandsraummodelle am Beispiel des Verbrennungsmotors*. Dissertation, Technische Universität München, 2011.

[12] J. Suykens and J. Vandewalle. Absolute stability criterion for a lur'e problem with multilayer perceptron nonlinearity. *International Workshop on Nonlinear Dynamics of Electronic Systems*, 1996.

[13] Matthias Reiner. *Modellierung und Steuerung von strukturelastischen Robotern*. Dissertation, Technische Universität München, 2010.

[14] Matthias Kurze. *Modellbasierte Regelung von Robotern mit elastischen Gelenken ohne abtriebsseitige Sensorik*. Dissertation, Technische Universität München, 2008.

[15] J. Suykens, B.L.R. de Moor, and J. Vandewalle. Nlq theory: a neural control framework with global asymptotic stability criteria. *Journal Neural Networks Volume 10 Issue 4*, 1997.