Model Predicitve Position Control of Electrical Drives on an Industrial PC

Fabian Karau, Michael Leuer
Bielefeld University of Applied Science
Schulstraße 10
Gütersloh, Germany
Email: fabian.karau1@fh-bielefeld.de, michael.leuer@fh-bielefeld.de

Keywords

≪MPC (Model-based Predictive Control)≫, ≪Motion control≫, ≪Dead-time≫, ≪Servo-drive≫, ≪State-space model≫.

Abstract

Dynamic position control of electric drives is of high importance in many industrial applications. In addition to the classic cascade control based on P and PI controllers for this task, advanced controls such as model predictive control (MPC) are receiving increasing attention. However, one drawback of MPC is the high computational power required, which makes it difficult to implement, especially for older servo drives. In this paper, we show how an industrial PC (IPC) that is available at many plants anyway can be used to implement MPC for position control of a simple drive system. Occurring dead times can be considered directly by the MPC.

Introduction

Electric drives are used for highly dynamic, precise positioning tasks in many industrial applications. Examples include machine tools, robotics applications and printed circuit board assembly. The quality of the drive control has a significant effect on the quality of the manufactured workpieces and also influences the wear, energy consumption and productivity of the system. The control concept is therefore highly important and has been subject of research for years.

The foundation for many of today's industrially used drive control structures is a cascade control based on linear P and PI controllers. This consists of the current or torque control loop with the superimposed velocity and position control loops. It is characterized by simple implementation and parameterization. In addition, it can be easily extended with feedforward controls and setpoint as well as actual value filters. However, complex drive systems with non-linear influences such as gear backlash, friction or elasticities in the drive train and manipulated variable or state variable constraints can only be handled to a limited extent by PI-based cascade control [1, 2].

Model predictive control (MPC) is a universal control method capable of solving the above problems. With the help of a mathematical model of the controlled system, the reaction to certain manipulated variables can be predicted. By solving an optimization problem, the optimal manipulated variables in terms of a freely formulable cost function are determined. The model and the optimization problem can include nonlinearities and constraints. This potentially enables MPC to achieve a superior control performance compared to PI-based cascade control.

The high computational power required by MPC to solve the optimization problem is a major drawback compared to P and PI controllers. Thanks to the generally increasing computational power, the concept of model-based predictive control has also become more of a focus in drive technology. Numerous MPC algorithms have already been published for current and torque control [3, 4, 5] as well as for velocity and position control [6, 7, 8, 9, 10, 11]. In many publications, powerful dSPACE rapid control prototyping systems are used for this purpose [6, 9, 8]. But also the implementation on modern SoC FPGAs [10] and state of the art digital signal processors (DSP)[11] has already been realized. Industrial PCs (IPCs) represent another exciting target platform. Newer IPCs have standard extensive memory and computing power. In addition, almost every industrial plant has a programmable logic controller (PLC), which is increasingly being implemented as an IPC. The goal of this paper is therefore to implement a first approach of a model predictive position control on an IPC for a simple drive system. In this way, older servo drives without sufficient computing power can benefit from advanced control methods as well.

The upper part of Fig. 1 displays the classical, widespread cascade control in a simplified way. Here, control takes place entirely within the servo drive and setpoint generation runs on the IPC. The lower part shows the desired control structure, in which both the position/velocity MPC and the setpoint generation are implemented on the IPC. The servo drive receives a torque reference from the IPC and is used only for torque control and setting the motor voltages. Strictly speaking, current control takes place, but with the permanent magnet synchronous motors used, the current can be converted into a torque.

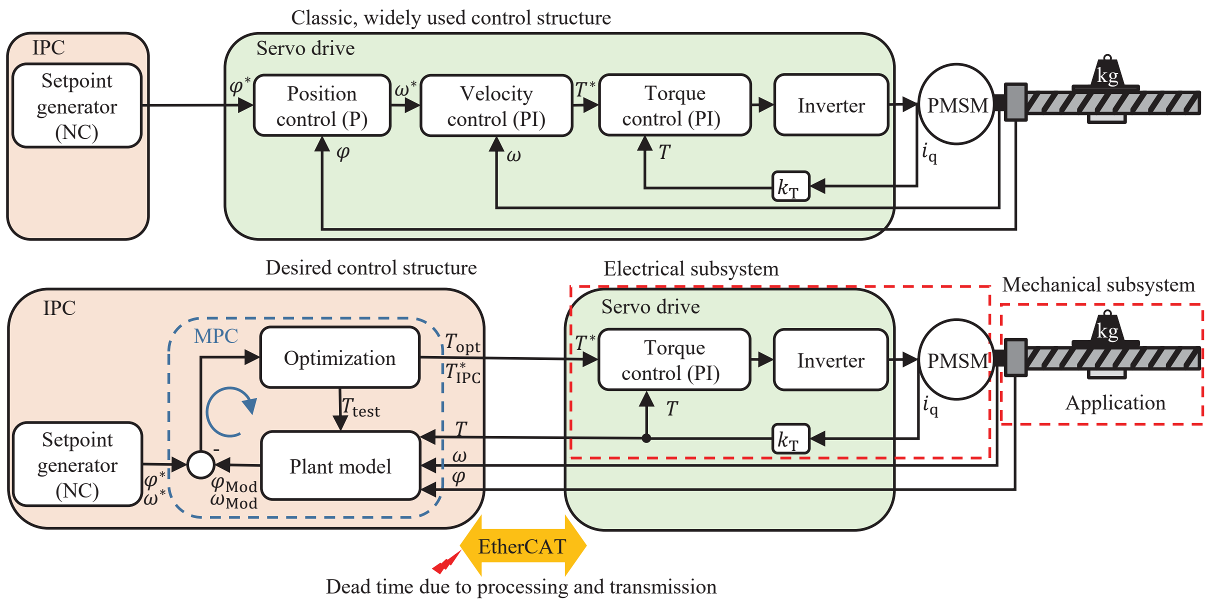

Fig. 1: Simplified overview of classic cascade control and the desired MPC control structure

Model Predictive Control

This section describes the applied MPC approach. The basis of the control is the model of the controlled system, which is explained in the following.

Prediction Model

The controlled system of the position control of a drive system can be divided into an electrical and a mechanical subsystem. The electrical subsystem comprises the closed torque control loop including the inverter and motor windings. Since there is no direct access to the inverter from the IPC in the form of a switching state or voltage reference, the torque control structure is not changed and the preset PI controller is used. Furthermore, the exchange of reference and actual values between the IPC and servo drive via the EtherCAT bus system and the processing time in the PLC introduces additional dead time into the system. The closed torque control loop and the total dead time are modeled by the transfer function (1)

$$G_{\mathrm{Trq}}(s) = \frac{T_{\mathrm{M}}(s)}{T^*_{\mathrm{M\,IPC}}(s)} = e^{-\tau_{\mathrm{Delay}}\cdot s} \cdot \frac{1}{\tau_{\mathrm{Trq}}^2 \cdot s^2 + 2D_{\mathrm{Trq}}\tau_{\mathrm{Trq}} \cdot s + 1}\,. \tag{1}$$

Fig. 2 shows the measured frequency response with a sampling time $T_{\mathrm{S}} = 250\,\mu\mathrm{s}$ and the model fitted according to (1). The parameters determined are $\tau_{\text{Trq}} = 154.64\,\mu\text{s}$, $\tau_{\text{Delay}} = 843.75\,\mu\text{s}$ and $D_{\text{Trq}} = 0.7071$.

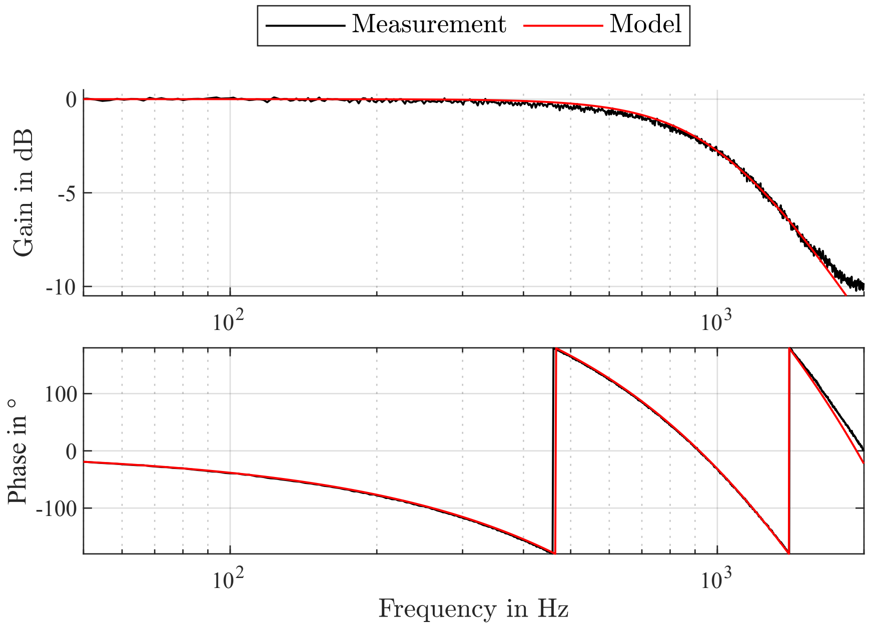

Fig. 2: Measured frequency response and estimated model

Here, the PT2 element mainly describes the torque control loop and τ_{Delay} the accumulated dead time in the system as well as the remaining model deviations of the torque control loop. For the hardware and configuration used, the total system dead time is thus $\tau_{\text{Delay}} = 3.375 \cdot T_{\text{S}}$. By optimizing the configuration, a further reduction of the dead time is probably possible, but a dead time will always remain in the described experimental setup.

The mechanical subsystem is composed of the motor moment of inertia and the driven application. For rigid mechanics, modeling as a single-mass system with the total moment of inertia Θ_{T} and the viscous friction μ_{FV} is permissible. The disturbance torques caused by the application and the coloumb friction are summarized as load torque T_{L}. This load torque is usually not measurable and must be reconstructed by an observer [7], Kalman filter [11] or moving horizon estimator [8, 9]. Here a Luenberger disturbance observer is used, whose poles are placed according to Butterworth with a cutoff frequency of 400 Hz. The single mass system is described by (2)

$$\dot{\omega}_{\text{M}} = \frac{T_{\text{M}} - \mu_{\text{FV}} \cdot \omega_{\text{M}} - T_{\text{L}}}{\Theta_{\text{T}}}, \tag{2}$$

where ω_{M} is the motor angular velocity. The dead-time free part of (1) and (2) are combined and brought into the continuous-time state space representation according to (3) and (4). The integration of the angular velocity leads to the angular position

$$\underbrace{\begin{pmatrix} \ddot{T}_\mathrm{M} \\ \dot{T}_\mathrm{M} \\ \dot{\omega}_\mathrm{M} \\ \dot{\varphi}_\mathrm{M} \\ \dot{T}_\mathrm{L} \end{pmatrix}}_{\dot{x}(t)} = \underbrace{\begin{pmatrix} \frac{-2D_\mathrm{Trq}}{\tau_\mathrm{Trq}} & \frac{-1}{\tau^2_\mathrm{Trq}} & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{\Theta_\mathrm{G}} & \frac{-\mu_\mathrm{RV}}{\Theta_\mathrm{G}} & 0 & \frac{-1}{\Theta_\mathrm{G}} \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}}_{\mathbf{A}_\mathrm{c}} \cdot \underbrace{\begin{pmatrix} \dot{T}_\mathrm{M} \\ T_\mathrm{M} \\ \omega_\mathrm{M} \\ \varphi_\mathrm{M} \\ T_\mathrm{L} \end{pmatrix}}_{x(t)} + \underbrace{\begin{pmatrix} \frac{1}{\tau^2_\mathrm{Trq}} \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}}_{b_\mathrm{c}} \cdot \underbrace{T^*_\mathrm{M}}_{u(t)}, \tag{3}$$

$$\underbrace{\varphi_\mathrm{M}}_{y(t)} = \underbrace{\begin{pmatrix} 0 & 0 & 0 & 1 & 0 \end{pmatrix}}_{c_\mathrm{c}^\mathrm{T}} \cdot \underbrace{\begin{pmatrix} \dot{T}_\mathrm{M} \\ T_\mathrm{M} \\ \omega_\mathrm{M} \\ \varphi_\mathrm{M} \\ T_\mathrm{L} \end{pmatrix}}_{x(t)}. \tag{4}$$

For the implementation in a PLC program the model must be discretized. This is done with the known methods as described for example in [12]. To represent the dead time, an augmentation of the state space model is done. The discrete-time state space model finally has the form of (5)

$$x(k+1) = \mathbf{A}x(k) + bu(k), \qquad y(k) = c^\mathrm{T}x(k). \tag{5}$$

Control Approach

Over the years, numerous MPC algorithms have been developed. Nevertheless, there are some common features that essentially all algorithms share. Two important variables of MPC are the prediction horizon n_p and the control horizon n_c, where $n_\mathrm{p} \geq n_\mathrm{c}$. Starting from the current state $x(k)$ and a certain manipulated variable sequence $u(k+i)$, the course of the controlled variable $y(k+i)$ is predicted for n_p sampling steps, with $i = 1,2,...,n_\mathrm{p}$. The manipulated variable can be varied for n_c times by the optimization and remains constant afterwards. Similarly, the cost function is a central element of all algorithms. Often the predicted quadratic control deviation $(y^*(k+i) - y(k+i))^2$ and the manipulated variable changes $\Delta u(k+i-1)^2$ are evaluated, resulting in the cost function for the SISO case according to (6)

$$J(u,x) = \sum_{i=1}^{n_\mathrm{p}} (y^*(k+i) - y(k+i))^2 q + \Delta u(k+i-1)^2 r. \tag{6}$$

The aggressiveness/dynamics of the control can be set via the weights q and r. Instead of the sum equation, a compact representation in the matrix-vector notation according to (7) is also possible

$$J(\Delta\bar{u}) = (\bar{y}^* - \bar{y})^\mathrm{T} \cdot \bar{\mathbf{Q}} \cdot (\bar{y}^* - \bar{y}) + \Delta\bar{u}^\mathrm{T} \cdot \bar{\mathbf{R}} \cdot \Delta\bar{u} \tag{7}$$

with the vectors and matrices

$$\bar{y}^* = \begin{pmatrix} y^*(k+1) \\ y^*(k+2) \\ \vdots \\ y^*(k+n_\mathrm{p}) \end{pmatrix}, \Delta\bar{u} = \begin{pmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+n_\mathrm{c}-1) \end{pmatrix}, \bar{y} = \begin{pmatrix} y(k+1) \\ y(k+2) \\ \vdots \\ y(k+n_\mathrm{p}) \end{pmatrix}, \\ \bar{\mathbf{Q}} = \mathrm{diag}(q), \bar{\mathbf{R}} = \mathrm{diag}(r). \tag{8}$$

In many industrial motion control applications, the reference trajectory is known a priori, since it is given by the workpiece contour, for example. The MPC can take advantage of this by previewing the future position references and using them in the reference vector $\bar{y}^*$. The optimal sequence of manipulated variables $\Delta\bar{u}_{\mathrm{opt}}$ is the one that minimizes the cost function while complying with the constraints

$$\Delta\bar{u}_{\mathrm{opt}} = \underset{\Delta\bar{u}}{\operatorname{argmin}} J(\Delta\bar{u})\,. \tag{9}$$

Only the first value $u_{\mathrm{opt}}(k) = \Delta u_{\mathrm{opt}}(k) + u(k-1)$ is set. In the next sampling step the process starts again.

This principle underlies all MPC algorithms. In addition, a fundamental distinction can be made between whether the optimization problem is solved at runtime (online) or before commissioning (offline). The offline variant is also called explicit MPC. Here, the state space is decomposed into regions and the optimal control law is determined for each region. At runtime, only the current state has to be assigned to a region and the corresponding control law has to be applied. This approach avoids the computationally intensive online solution of the optimization problem, but requires a lot of memory to store the control laws. This is where the advantages of a modern IPC come into play, for which memory space is not an obstacle, making explicit MPC an attractive approach. Nevertheless, this paper is limited to the online solution. Investigations on explicit MPC are planned for further work.

The Finite Control Set MPC (FCS-MPC) and the Continuous Control Set (CCS-MPC) are the two main categories of MPC algorithms. With the FCS-MPC, the manipulated variable can only assume a limited number of discrete values, such as the switching states of the inverter. Therefore, the FCS-MPC is often used for current or torque control [3, 4, 5]. However, the computational complexity of the FCS-MPC increases exponentially to the prediction horizon. Since the time constants of the position control loop are usually much larger than the electrical time constants in the torque control loop, a long prediction horizon is required making FCS-MPC uncommon for direct position control.

For this purpose, the concept of CCS-MPC is usually used, where the manipulated variables can assume any value within the constraints. To limit the numerical effort and to avoid a non-convex optimization problem, simplified linear models are often aimed at. As long as no constraints have to be considered, a computationally efficient analytical solution is possible. If this solution exceeds a manipulated variable limit, it is simply limited to this value. This procedure has been successfully implemented in [9, 10], for example. If constraints have to be considered in the optimization problem, methods of quadratic programming (QP) can be used as in [6, 8].

Since the IPC is also to be used for further control tasks, the analytical solution is selected as the solution approach despite its high computing power. This is described briefly below; a detailed description can be found in [13], for example.

The state space model (5) can also be formulated as $x(k+1) = \mathbf{A}x(k) + bu(k-1) + b\Delta u(k)$. Using the output equations, all predicted control variables can now be calculated up to the prediction horizon. Thereby predicted states $x(k+1)$, $x(k+2)$, ..., $x(k+n_{\mathrm{p}})$ have to be formulated depending on the current state $x(k)$. This is shown exemplarily for $y(k+1)$ and $y(k+2)$ in (10) and (11)

$$\begin{aligned} y(k+1) &= c^{\mathrm{T}}x(k+1) \\ &= c^{\mathrm{T}}\mathbf{A}x(k) + c^{\mathrm{T}}bu(k-1) + c^{\mathrm{T}}b\Delta u(k) \end{aligned} \tag{10}$$

$$\begin{aligned} y(k+2) &= c^{\mathrm{T}}x(k+2) \\ &= c^{\mathrm{T}}\mathbf{A}^2x(k) + c^{\mathrm{T}}(\mathbf{A}+\mathbf{I})bu(k-1) + c^{\mathrm{T}}(\mathbf{A}+\mathbf{I})b\Delta u(k) + c^{\mathrm{T}}b\Delta u(k+1) \end{aligned} \tag{11}$$

By continuing this scheme up to the prediction horizon, the predicted controlled variables can be computed in compact form according to (12)

$$\bar{y} = \mathbf{F}x(k) + \mathbf{G}u(k-1) + \mathbf{H}\Delta\bar{u}\,, \tag{12}$$

with the vectors and the matrices already introduced in (8) and the following

$$\mathbf{F} = \begin{pmatrix} c^{\mathrm{T}}\mathbf{A} \\ c^{\mathrm{T}}\mathbf{A}^2 \\ c^{\mathrm{T}}\mathbf{A}^3 \\ \vdots \\ c^{\mathrm{T}}\mathbf{A}^{n_\mathrm{p}} \end{pmatrix}, \mathbf{G} = \begin{pmatrix} c^{\mathrm{T}}b \\ c^{\mathrm{T}}(\mathbf{A}+\mathbf{I})b \\ c^{\mathrm{T}}(\mathbf{A}^2+\mathbf{A}+\mathbf{I})b \\ \vdots \\ c^{\mathrm{T}}(\mathbf{A}^{n_\mathrm{p}-1}+\ldots+\mathbf{I})b \end{pmatrix},$$

$$\mathbf{H} = \begin{pmatrix} c^{\mathrm{T}}b & 0 & \ldots & 0 \\ c^{\mathrm{T}}(\mathbf{A}+\mathbf{I})b & c^{\mathrm{T}}b & \ldots & 0 \\ c^{\mathrm{T}}(\mathbf{A}^2+\mathbf{A}+\mathbf{I})b & c^{\mathrm{T}}(\mathbf{A}+\mathbf{I})b & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ c^{\mathrm{T}}(\mathbf{A}^{n_\mathrm{c}-1}+\ldots+\mathbf{I})b & c^{\mathrm{T}}(\mathbf{A}^{n_\mathrm{c}-2}+\ldots+\mathbf{I})b & \ldots & c^{\mathrm{T}}b \\ c^{\mathrm{T}}(\mathbf{A}^{n_\mathrm{c}}+\ldots+\mathbf{I})b & c^{\mathrm{T}}(\mathbf{A}^{n_\mathrm{c}-1}+\ldots+\mathbf{I})b & \ldots & c^{\mathrm{T}}(\mathbf{A}+\mathbf{I})b \\ \vdots & \vdots & \ddots & \vdots \\ c^{\mathrm{T}}(\mathbf{A}^{n_\mathrm{p}-1}+\ldots+\mathbf{I})b & c^{\mathrm{T}}(\mathbf{A}^{n_\mathrm{p}-2}+\ldots+\mathbf{I})b & \ldots & c^{\mathrm{T}}(\mathbf{A}^{n_\mathrm{p}-n_\mathrm{c}}+\ldots+\mathbf{I})b \end{pmatrix} \quad (13)$$

The predicted control error that would occur without a change in the manipulated variable is now denoted by $\tilde{e}$

$$\tilde{e} = \bar{y}^* - \mathbf{F}x(k) - \mathbf{G}u(k-1)\,. \quad (14)$$

By substituting (12) and (14) into the cost function (7) it follows

$$J(\Delta\bar{u}) = (\tilde{e} - \mathbf{H}\Delta\bar{u})^{\mathrm{T}} \cdot \bar{\mathbf{Q}} \cdot (\tilde{e} - \mathbf{H}\Delta\bar{u}) + \Delta\bar{u}^{\mathrm{T}} \cdot \bar{\mathbf{R}} \cdot \Delta\bar{u}\,. \quad (15)$$

Setting the gradient of the cost function to zero corresponds to the necessary condition for a minimum, from which the analytical solution is determined

$$\frac{\partial J(\Delta\bar{u})}{\partial \Delta\bar{u}} = 2(\mathbf{H}^{\mathrm{T}}\bar{\mathbf{Q}}\mathbf{H} + \bar{\mathbf{R}})\Delta\bar{u} - 2\mathbf{H}^{\mathrm{T}}\bar{\mathbf{Q}}\tilde{e} = 0\,, \quad (16)$$

$$\Delta\bar{u} = (\mathbf{H}^{\mathrm{T}}\bar{\mathbf{Q}}\mathbf{H} + \bar{\mathbf{R}})^{-1}\mathbf{H}^{\mathrm{T}}\bar{\mathbf{Q}}\tilde{e}\,. \quad (17)$$

Experimental results

The described algorithm is implemented on a Beckhoff IPC CP6930-0050 with an i7-4700EQ 2.4GHz processor and 2 times 4096 MB DDR3L-RAM SO memory and tested with the laboratory setup shown in Fig. 3. Two PMSM of type Beckhoff AM3021 are connected via a rigid coupling. One motor acts as the test motor, the other serves as the load motor. Torque control is performed on a Beckhoff AX5203 servo drive. The configured maximum torque is $T_\mathrm{L} = 0.9\,\mathrm{Nm}$.

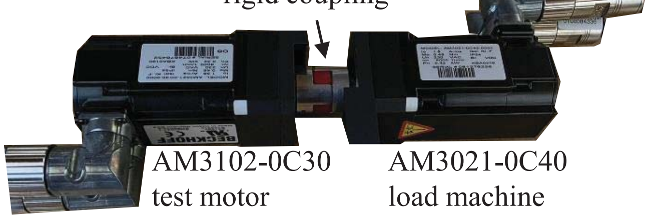

Fig. 3: Laboratory setup

The MPC parameters were set experimentally. Here $n_\mathrm{p} = 20$, $n_\mathrm{c} = 10$, $q = 1$ and $r = 0.2$ are selected. Moreover, the MPC controller is implemented with reference previewing. The cascade control was also parameterized manually and operates with active velocity feedforward. The sampling time of the

MPC and the position controller is $T_S = 250\,\mu s$, that of the subordinate velocity controller is $125\,\mu s$. It is emphasized again that for the comparative representation the cascade control is completely closed within the servo drive and is therefore not influenced by the dead time.

Two reference trajectories are compared. On the left side in Fig. 4, a typical trajectory with limited jerk is shown. Due to the dead time that acts between the numerical control (NC on the IPC) and the servo drive with the classic cascade control, the actual position follows the NC setpoint position delayed with this control, as can be seen in the detailed view in the upper left figure. However, in order to evaluate the path accuracy when tracking contours, the tracking error measured in the servo drive is relevant. Therefore this tracking error is shown. For the MPC, the tracking error is calculated in the IPC. Without disturbances, the MPC has a slightly higher maximum tracking error, which is, however, compensated for faster than with classic cascade control. This is especially the case for velocity transitions. At the time 100 ms a load step is switched on. Both controllers are able to compensate for the disturbance, although a higher tracking error is observed with the MPC due to the delayed response caused by the dead time. Furthermore, small oscillations in the tracking error are to be noted with the MPC, which can be explained by the high noise sensitivity of the MPC.

On the right side in Fig. 4, the results of a step-shaped excitation are illustrated. This excitation leads to an operation at the manipulated variable limit resulting in anti-windup procedures becoming active. Therefore, the tracking error is compensated faster with the MPC, albeit with an overshoot. In addition, a load step is switched on again. The influence of the dead time can be clearly seen in the detailed picture, which is why the MPC reacts later and a larger tracking error is formed. This is still compensated faster than with the cascade control. To improve the disturbance behavior, the dead time must be reduced by decreasing the sampling time.

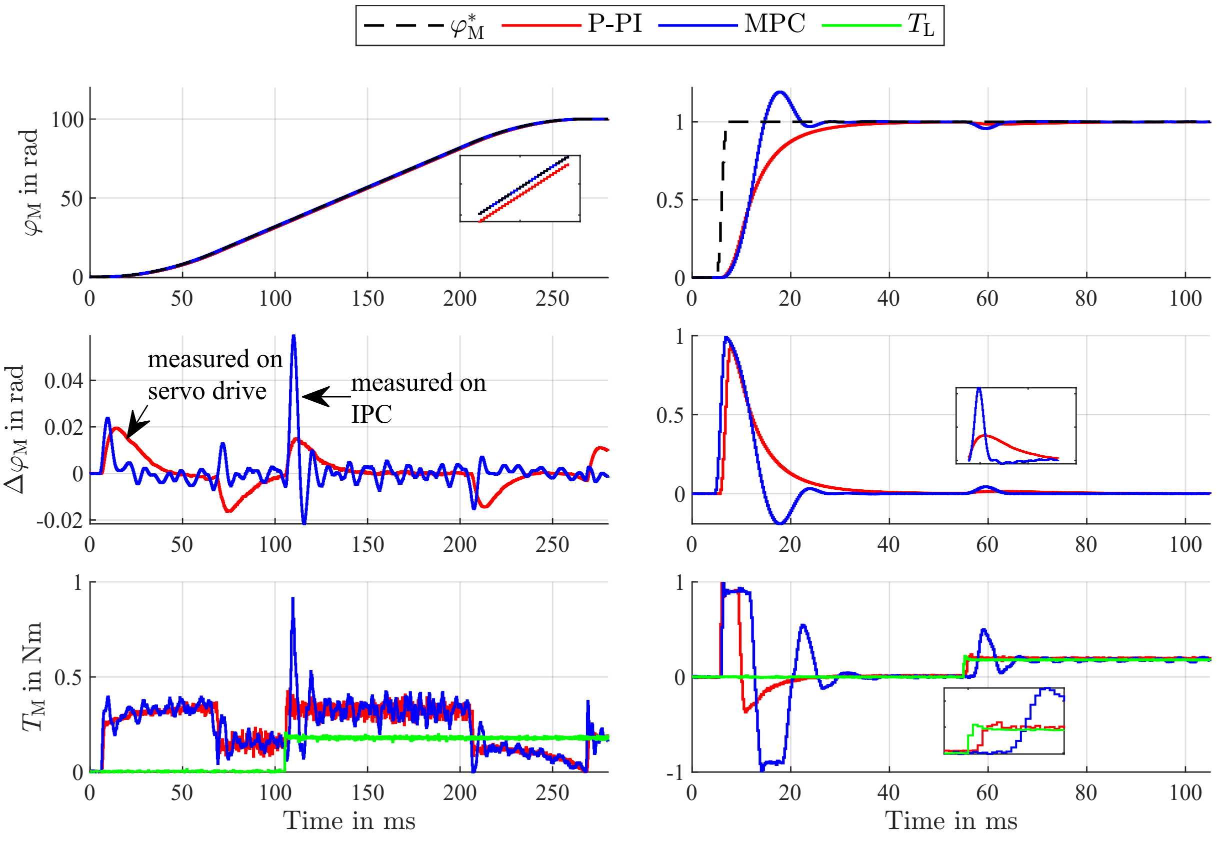

Fig. 4: Comparison of MPC and P-PI cascade control: Left typical position trajectory, Right step-like excitation. Disturbance $T_L = 0.18\,\mathrm{Nm}$

Furthermore, during the test the CPU utilization of the IPC was measured with the function block TC_CpuUsage. The resulting CPU usage or required computing power strongly depends on the particular implementation. The code of the MPC algorithm was generated with the standard Simulink PLC coder as well as with a special Beckhoff TE1400 + Simulink coder [14]. The results are shown in Table I.

Table I: CPU usage of the IPC

	only PLC programm	PLC programm with MPC (Simulink Coder with TE1400 (C++))	PLC programm with MPC (Simulink PLC Coder)
CPU usage	$\approx 8\,\%$	$\approx 12\,\%$	$\approx 20\,\%$

About 8 % are required as base load for the TwinCAT system and the test program. The MPC increases the utilization to 12 % or 20 %, depending on the implementation. It should be noted that in the extended state space it is already a ninth-order system and quite long horizons were used. Model simplifications and optimization of the MPC parameters can potentially save further computation time. Nevertheless, it can be stated that the computational load on the CPU is moderate and further control tasks can definitely be executed. With the IPC used, it is even possible to increase the complexity of the MPC and thus achieve further improvements.

Finally, the significance of the results on the path accuracy in the interaction of several axes is to be investigated simulatively, since no real plant was available. For this purpose, a diamond-shaped trajectory in the XY-plane is given. The results are illustrated in Fig. 5. In the cascade control considered, the velocity feedforward is not balanced by a position setpoint filter. The result is a strong overshoot at contour corners. With the MPC, the path accuracy is significantly improved. A similar result is achieved if the position setpoint filter is implemented correctly. For this purpose, the filter time constant must be set to the equivalent time constant of the velocity control loop, which therefore means additional tuning effort.

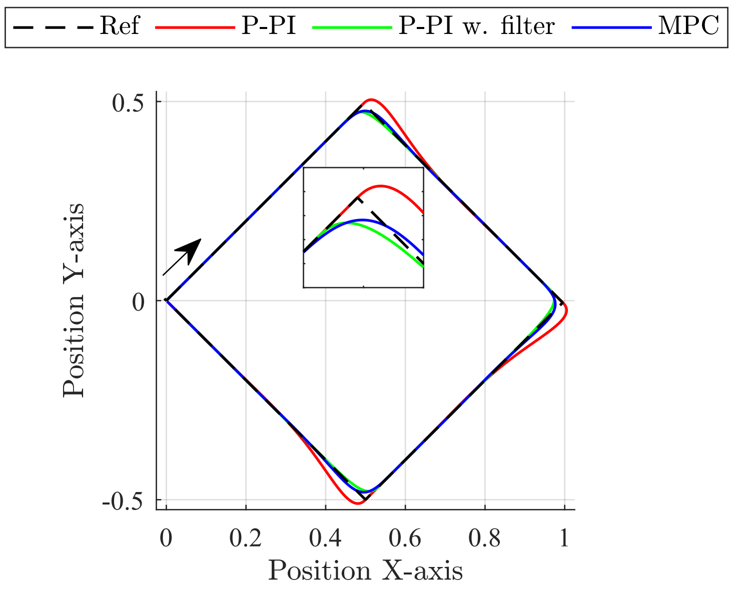

Fig. 5: Simulation: Tracking performance for a diamond reference trajectory

Conclusion

Model predictive control is well known for its superior control performance compared to PI control. However, to take advantage of this in practice, a target platform with sufficient computing power is required. Industrial PCs, in contrast to standard servo drives, have sufficient computing power and are available in many industrial plants. This paper gives evidence that they are suitable for the implementation of high dynamic model predictive position control. A main problem with the implementation on the IPC is the occurring dead time due to the communication between the IPC and the servo drive. This leads to a delayed response to disturbances compared to the internal cascaded control implemented on the servo drive. Since the IPC has sufficient computing capacity for further improvements subsequent work will investigate how to improve the disturbance behavior. Another exciting research topic is explicit MPC, for which the IPC is an ideal target platform thanks to its large storage capacity.

References

[1] S. Thomsen and F. W. Fuchs, ”Speed control of torsional drive systems with backlash,” 2009 13th European Conference on Power Electronics and Applications, 2009, pp. 1-10

[2] S. Thomsen, N. Hoffmann and F. W. Fuchs: PI Control, PI-Based State Space Control, and Model-Based Predictive Control for Drive Systems With Elastically Coupled Loads—A Comparative Study, in IEEE Transactions on Industrial Electronics, vol. 58, no. 8, pp. 3647-3657, Aug. 2011

[3] T. Geyer, G. Papafotiou and M. Morari, ”Model Predictive Direct Torque Control—Part I: Concept, Algorithm, and Analysis,” in IEEE Transactions on Industrial Electronics, vol. 56, no. 6, pp. 1894-1905, 2009

[4] M. Leuer and J. Böcker: Fast online model predictive control of IPMSM using parallel computing on FPGA, 2013 International Electric Machines Drives Conference, 2013, pp. 1017-1022

[5] H. A. Young, M. A. Perez, J. Rodriguez and H. Abu-Rub, ”Assessing Finite-Control-Set Model Predictive Control: A Comparison with a Linear Current Controller in Two-Level Voltage Source Inverters,” in IEEE Industrial Electronics Magazine, vol. 8, no. 1, pp. 44-52, March 2014

[6] P. Serkies and K. Szabat: Predictive Control of the Two-Mass Drive with an Induction Motor for a Wide Speed Range, 2018 IEEE 18th International Power Electronics and Motion Control Conference (PEMC), 2018, pp. 750-755

[7] P. Serkies and K. Szabat: Predictive position control of the induction two-mass system drive, 2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE), 2014, pp. 871-876

[8] O. Wallscheid, E. F. Bouna Ngoumtsa and J. Böcker: Hierarchical Model Predictive Speed and Current Control of an Induction Machine Drive with Moving-Horizon Load Torque Estimator, 2019 IEEE International Electric Machines Drives Conference (IEMDC), 2019, pp. 2188-2195

[9] P. G. Carlet, F. Toso, A. Favato and S. Bolognani: A speed and current cascade Continuous Control Set Model Predictive Control architecture for synchronous motor drives, 2019 IEEE Energy Conversion Congress and Exposition (ECCE), 2019, pp. 5682-5688

[10] S. Wendel, B. Haucke-Korber, A. Dietz and R. Kennel: Cascaded Continuous and Finite Model Predictive Speed Control for Electrical Drives, 2018 20th European Conference on Power Electronics and Applications (EPE’18 ECCE Europe), 2018, pp. P.1-P.10

[11] S. Wendel, P. Löhdefink, M. Hoerner, A. Dietz and R. Kennel: Dynamic model predictive position control for linear actuators in automotive applications, 2018 Thirteenth International Conference on Ecological Vehicles and Renewable Energies (EVER), 2018, pp. 1-6

[12] J. Lunze: Regelungstechnik 2, 12th ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2020

[13] J. Adamy: Nichtlineare Systeme und Regelungen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018

[14] Beckhoff Automation GmbH & Co. KG: Manual Target for Simulink, 2022