2018 European Control Conference (ECC)
June 12-15, 2018. Limassol, Cyprus

Event-Based Networked Model Predictive Control With Overclocked Local Nodes

Patrik Simon Berner and Martin Mönnigmann

***Abstract*— We improve a recently proposed networked MPC approach by reducing the memory and computational requirements for the local nodes and by reducing the network use. The networked MPC setup consists of a powerful central node that provides one or multiple local nodes with regionally optimal affine feedback laws. Whenever the region of optimality of a current affine law is left on one of the local nodes, this local node requests a new optimal law from the central node. Since the local node uses lean hardware, computational and memory resources are restricted. Similarly, network usage should be as small as possible. The proposed method essentially increases the sampling time and reduces the horizon of the underlying MPC, which results in the desired reductions on the local node and the network usage. We show that the feedback laws can be overclocked on the local nodes to compensate for the loss of performance due to longer sampling times and shorter horizons. Results are obtained from hardware-in-the-loop simulations with a micro controller and a PC as local and central nodes, respectively, and with a wireless network.**

I. Introduction

Many approaches to reducing the online computational cost of model predictive control (MPC) have been explored. The best-known approach is probably parametric programming (see [1]–[3]), where the solution to a parametric quadratic program is computed offline, stored and evaluated online. Since no optimization problems need to be solved online, this technique is attractive for implementations on embedded hardware (see e.g. in [4]–[6]). However, storage requirements grow rapidly with increasing system size, limiting possible applications. Another generic technique is code generation, where problem and hardware-tailored code is generated. Code generation toolboxes with attention to embedded hardware are proposed e.g. in [7] and [8]. In move blocking, the degrees of freedom of the optimal control problem are reduced by assuming the input to be constant over one ore more time steps. This reduces the computational effort as examined in [9], [10]. Finally, networked MPC approaches must be mentioned in this context, where complex calculations are carried out on a central computation node and signals or control laws are sent across networks to be executed by lean, low-power hardware located near the process under control (see e.g. [11]–[17]).

In the present paper we make use of the *event triggered networked model predictive controller* we examined in [18], [19] and propose a method that reduces the complexity of local computations and the amount of transmitted and locally stored data. The controller works as follows: Multiple local nodes are connected to a single central node via a network connection (e.g. a wireless network), as depicted in Fig. 1. Each local node controls the attached system in a local feedback loop. There is no interaction between the local nodes. The local nodes are energy efficient, because they merely evaluate affine control laws and request a new affine law whenever the region of optimality of the current one is left. Leaving the polytope is understood as the event that triggers the request of a new feedback law and polytope from the central node for the current system state. Thus, the network connection in Fig. 1 is only used on demand.

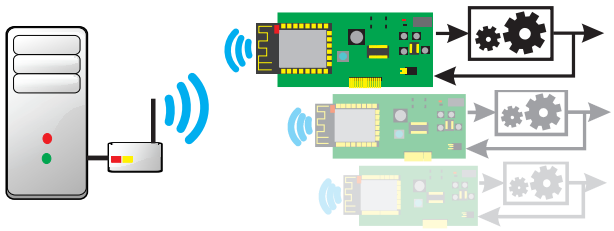

Fig. 1: Setup of central node (left) and multiple local nodes (right) with wireless connection in between. Each local node controls its connected system in a closed loop (solid black lines) while the wireless connection is only used on demand.

Since the local node is a lean hardware, the local computations, the local storage requirements and the transmitted data must be reduced to a minimum. We propose to reduce these figures by using short horizon lengths and long discretization time steps, and to compensate for the resulting loss in performance by overclocking the feedback calculations on the local node. This overclocking results in the MPC controller implemented on the central node to be used at a higher frequency on the local node than it was designed for. We note the authors of [20] already investigated changing the sampling time of a networked controller to reduce network usage for an unconstrained linear quadratic regulator. Although the idea to reduce the sampling time is the same here, the present work differs in that it considers MPC with constraints, and in that the setup of the networked controller is different.

Section II states background on MPC and the event triggered MPC scheme from [18], [19] as needed here. Section III states the new method and examines the achievable reductions. We provide results obtained with hardware-in-the-loop simulations in Section IV and conclude in Section V.

Both authors are with Automatic Control and Systems Theory, Department of Mechanical Engineering, Ruhr-Universität Bochum, Germany, E-mail: patrik-simon.berner@rub.de, martin.moennigmann@rub.de

II. Event-triggered networked MPC

We consider discrete-time linear, time-invariant systems

$$x(k+1) = Ax(k) + Bu(k), \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ result from discretizing

$$\dot{x}_c(t) = A_c x_c(t) + B_c u_c(t) \tag{2}$$

with

$$A = e^{A_c \Delta T}, \quad B = \int_0^{\Delta T} e^{A_c \tau} d\tau B_c, \tag{3}$$

where ΔT denotes the discretization time and both (A, B) and (A_c, B_c) are assumed to be stabilizable. System states $x(k)$ and inputs $u(k)$ are subject to the constraints

$$x(k) \in \mathcal{X} \text{ and } u(k) \in \mathcal{U} \text{ for all } k, \tag{4}$$

where $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{U} \subset \mathbb{R}^m$ are compact and convex polytopes containing the origin in their interior. Whenever we compare the behavior of a continuous-time system (2) to a discrete-time system (1) we assume $x(0) = x_c(0)$ and

$$u_c(t) = u(k) \text{ for } t \in [k\Delta T, (k+1)\Delta T). \tag{5}$$

We recall (3) and (5) guarantee $x_c(k\Delta T) = x(k)\ \forall k$. It is well-known, however, that the continuous-time system may violate the constraints for times $t \in (k\Delta T, (k+1)\Delta T)$ even if they are enforced for all $k\Delta T$ for the discrete-time system.

An optimal input sequence $U = \{u(0), ..., u(N-1)\}$ is determined by solving the optimal control problem

$$\min_{X,U} J(X, U) \tag{6a}$$

$$\text{s.t. } x(k+1) = Ax(k) + Bu(k),\ \forall k \in [0, N-1] \tag{6b}$$

$$x(0) \text{ given},$$

$$u(k) \in \mathcal{U}, \qquad \forall k \in [0, N-1] \tag{6c}$$

$$x(k) \in \mathcal{X}, \qquad \forall k \in [0, N-1] \tag{6d}$$

$$x(N) \in \mathcal{T}, \tag{6e}$$

with $X = \{x(0), ..., x(N)\}$, the value function $J(X, U) = x^T(N)Px(N) + \sum_{k=0}^{N-1} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} Q & V \\ V^T & R \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}$, and horizon length N in every time step. We assume the weighting matrix, $\begin{bmatrix} Q & V \\ V^T & R \end{bmatrix}$ with $Q \in \mathbb{R}^{n\times n}$, $Q \succeq 0$, $R \in \mathbb{R}^{m\times m}$, $R \succ 0$ and $V \in \mathbb{R}^{n\times m}$, is positive definite. The value function J results from discretizing the continuous value function

$$J_c = x(N\Delta T)^T P_c x(N\Delta T) + \int_0^{N\Delta T} x(\tau)^T Q_c x(\tau) + u(\tau)^T R_c u(\tau) d\tau \tag{7}$$

with continuous weighting matrices Q_c, R_c and P_c as shown below [21, pp. 411], [20]:

$$\begin{bmatrix} Q & V \\ V^T & R \end{bmatrix} = \int_0^{\Delta T} e^{\begin{bmatrix} A_c & B_c \\ 0 & 0 \end{bmatrix}^T s} \begin{bmatrix} Q_c & 0 \\ 0 & R_c \end{bmatrix} e^{\begin{bmatrix} A_c & B_c \\ 0 & 0 \end{bmatrix} s} ds. \tag{8}$$

The terminal set $\mathcal{T}$ is chosen to be the maximum positively invariant set for (1) under the terminal controller K_T which is used to compute the control input whenever $x(k)$ enters the terminal set (i.e. $u(j) = K_T x(j)\ \forall x(j) \in \mathcal{T}$). K_T and P result from solving the discrete-time algebraic Riccati equation for the dynamics (1) and $\mathcal{T}$ is calculated as stated in [22].

We write the MPC problem (6) as a quadratic program (QP) [23, Chap. 3]:

$$\min_U \frac{1}{2} U^T H U + x^T F U \quad s.t.\ GU - Ex \le w \tag{9}$$

with $U \in \mathbb{R}^{mN}$, $F \in \mathbb{R}^{n\times mN}$, $H \in \mathbb{R}^{mN\times mN}$, $G \in \mathbb{R}^{q\times mN}$, $w \in \mathbb{R}^q$, $E \in \mathbb{R}^{q\times n}$, where $\mathcal{Q} = \{1, \ldots, q\}$ denotes the set of all constraint indices. H is known to be positive definite if $\begin{bmatrix} Q & V \\ V^T & R \end{bmatrix}$ is positive definite [23, p. 76]. The set of feasible states is denoted $\mathcal{X}_f$. Since the cost function is strictly convex ($H \succ 0$) and constraints (6c-6e) are linear, there exists, for every $x \in \mathcal{X}_f$, a unique optimal solution, which we denote $U^\star(x)$. In [2], [3] it is shown that this optimal solution is piece-wise affine on state space polytopes:

$$U^\star(x) = \begin{cases} K_1 x + b_1, \text{ if } x \in \mathcal{P}_1, \\ \vdots \\ K_p x + b_p, \text{ if } x \in \mathcal{P}_p, \end{cases} \tag{10}$$

with polytopes $\mathcal{P}_1, ..., \mathcal{P}_p$, $\cup_i^p \mathcal{P}_i = \mathcal{X}_f$ and $K_i \in \mathbb{R}^{mN\times n}$, $b_i \in \mathbb{R}^{mN}$, $i = 1, ..., p$. We never actually calculate the explicit law (10) of the solution in the present paper, but exploit the structure by reusing the regional feedback law $K_i x + b_i$ as long as the system state does not leave the respective polytope $\mathcal{P}_i$. We refer to this as regional MPC [19]. The following lemma, which is an immediate corollary to the results in [2], summarizes how to determine the polytope and its regional law from the point wise solution of the MPC problem (9).

Lemma 1: Let $U^\star(x)$ be the solution to (9) for an arbitrary $x \in \mathcal{X}_f$, and $\mathcal{A}$ and $\mathcal{I}$ refer to the sets of active and inactive constraints

$$\begin{aligned} \mathcal{A} &= \{i \in \{1, ..., q\} \,|G_i\, U^\star(x) - E_i x = w_i\}, \\ \mathcal{I} &= \{i \in \{1, ..., q\} \,|G_i\, U^\star(x) - E_i x < w_i\}. \end{aligned} \tag{11}$$

Assume the matrix $G_\mathcal{A}$ has full row rank, where $X_\mathcal{A}$ and $X_\mathcal{I}$ denote the sub matrices of any matrix X with rows in $\mathcal{A}$ and $\mathcal{I}$, respectively. Then

$$\begin{aligned} K^\star &= H^{-1}(G_\mathcal{A})^T (G_\mathcal{A} H^{-1} (G_\mathcal{A})^T)^{-1} S_\mathcal{A} - H^{-1} F^T, \\ b^\star &= H^{-1}(G_\mathcal{A})^T (G_\mathcal{A} H^{-1} (G_\mathcal{A})^T)^{-1} w_\mathcal{A}, \\ T^\star &= \begin{pmatrix} G_\mathcal{I} H^{-1}(G_\mathcal{A})^T (G_\mathcal{A} H^{-1}(G_\mathcal{A})^T)^{-1} S_\mathcal{A} - S_\mathcal{I} \\ (G_\mathcal{A} H^{-1}(G_\mathcal{A})^T)^{-1} S_\mathcal{A} \end{pmatrix}, \\ d^\star &= -\begin{pmatrix} G_\mathcal{I} H^{-1}(G_\mathcal{A})^T (G_\mathcal{A} H^{-1}(G_\mathcal{A})^T)^{-1} w_\mathcal{A} - w_\mathcal{I} \\ (G_\mathcal{A} H^{-1}(G_\mathcal{A})^T)^{-1} w_\mathcal{A} \end{pmatrix}, \end{aligned} \tag{12}$$

with $S = E + GH^{-1}F^T$, $S \in \mathbb{R}^{q\times n}$ define the optimal control law and the polytope on which the control law is valid:

$$U^\star(x) = K^\star x + b^\star\ \forall\ x \in \mathcal{P}^\star, \tag{13a}$$

$$\mathcal{P}^\star = \{x \in \mathbb{R}^n \,|\, T^\star x \le d^\star\}. \tag{13b}$$

Lemma 1 implies an optimal control law $U^\star(x) = K^\star x + b^\star$ and its polytope of validity $\mathcal{P}^\star$ can be calculated with simple matrix-vector operations (12) if the current active set $\mathcal{A}$ is known. Since $\mathcal{A}$ can be encoded in only q bits, it is advantageous to send $\mathcal{A}$ instead of $K^\star$, $b^\star$ and $\mathcal{P}^\star$ from the central to the local node [18]. This idea is implemented in Algorithm 1, which is the reference to which the new algorithms stated in the present paper must be compared.

Data: $\hat{A}, \hat{B}, \hat{\mathcal{X}}_f$
Initialization: $k = 0$, $x(0) \in \hat{\mathcal{X}}_f$, $\hat{\mathcal{A}}(x(0))$, $\hat{K}^\star_{\mathcal{M}}$, $\hat{b}^\star_{\mathcal{M}}$, $\hat{T}^\star$, $\hat{d}^\star$ for $\hat{\mathcal{A}}(x(0))$;
on local node repeat at frequency $1/\Delta\hat{T}$
  measure current state $x(k)$;
  if $\hat{T}^\star x(k) \leq \hat{d}^\star$ **then**
    $u(k) = \hat{K}^\star_{\mathcal{M}} x(k) + \hat{b}^\star_{\mathcal{M}}$;
    $k \leftarrow k + 1$;
  else
    send $x(k)$ to central node and request $\hat{\mathcal{A}}(x(k))$;
    determine $\hat{K}^\star_{\mathcal{M}}, \hat{b}^\star_{\mathcal{M}}, \hat{T}^\star$, $\hat{d}^\star$ with Lemma 1;
    $u(k) = \hat{K}^\star_{\mathcal{M}} x(k) + \hat{b}^\star_{\mathcal{M}}$;
    $k \leftarrow k + 1$;
  end
end
on central node
  Solve (9), (11) for $x(k)$ to obtain $\hat{\mathcal{A}}(x(k))$;
  Transmit $\hat{\mathcal{A}}(x(k))$ from central to local node;
end

Algorithm 1: Event triggered networked MPC

It is easy to see from Algorithm 1 that the *if* condition triggers whether communication with the central node (referred as *on central node* block in Alg. 1) is necessary. We thus call $T^\star x(k) \leq d^\star$ the *triggering condition*, and use the term *event* to indicate that the triggering condition is *not* fulfilled (i.e. an event occurs if $x(k) \notin \mathcal{P}^\star$). As long as no event occurs, *no* QP must be solved, *no* data needs to be transmitted over the network, and the feedback law currently stored on the local node can be evaluated with very little computational effort [18].

III. Reducing the Complexity of Event Triggered MPC

Lemma 2 below extends a result from [18] by a statement about storage requirements. We assume that multiplications, summations and divisions require one, one and ten floating point operations (flops), respectively [24]. Floating point numbers are represented with 32 bits according to IEEE 745 [25].

Lemma 2: The following statements hold for the approach summarized in Alg. 1.

(i) The central node transmits $q = |\mathcal{Q}|$ bits to the local node whenever an optimization is triggered.

(ii) On the local node,

$$\begin{aligned}\eta = {} & (mNq_{\mathcal{A}} + mn + qq_{\mathcal{A}})(2mN - 1) \\ & + (qn + q + m + mn)(2q_{\mathcal{A}} - 1) \\ & + \frac{2}{3}q_{\mathcal{A}}^3 + 6q_{\mathcal{A}}^2 + \frac{7}{3}q_{\mathcal{A}} + qn + q - q_{\mathcal{A}}n + mn\end{aligned} \tag{14}$$

flops are required to calculate the control law and polytope (12) whenever a new $\mathcal{A}$ is received.

(iii) The number of bits that must be stored on the local node is

$$\begin{aligned}\rho = 32 \big(& m^2N^2 + q\,(mN + 2n + 2) \\ & + nmN + mn + m\big).\end{aligned} \tag{15}$$

Proof: The proof of parts (i) and (ii) is given in [18]. Statement (iii) results, because $H^{-1} \in \mathbb{R}^{mN \times mN}$, $G \in \mathbb{R}^{q \times mN}$, $S \in \mathbb{R}^{q \times n}$, $F \in \mathbb{R}^{n \times mN}$, $w \in \mathbb{R}^q$, $K^\star_{\mathcal{M}} \in \mathbb{R}^{m \times n}$, $b^\star_{\mathcal{M}} \in \mathbb{R}^m$, and $T^\star$, $d^\star$, which have the dimensions $q \times n$ and q, respectively, need to be stored. Assuming 32 bits per floating point number, (iii) results. Because we only apply the first $\mathcal{M} = \{1, \ldots, m\}$ elements of the predicted input U, it suffices to compute and store $K^\star_{\mathcal{M}}$ and $b^\star_{\mathcal{M}}$. ■

A. Oversampling MPC on the local nodes

A short sampling time ΔT and a long horizon N is obviously of interest in MPC. It is just as obvious, of course, that both decreasing ΔT and increasing N results in an increase of computational cost. Since the central node of the networked MPC variant considered here (see Fig. 1) is assumed to be arbitrarily powerful, we ignore the effect of decreasing ΔT and increasing N on this node. The memory and computation time increase on the local node as well, however. Since using lean local nodes is the main goal for the setup sketched in Fig. 1, these increases must be analyzed and mitigated.

We propose to reduce the complexity of the local computations and storage requirements by simply reducing N while maintaining the predicted time. More precisely, let T_c be an arbitrary but fixed time span of prediction and let $\check{N}$ and $\hat{N} = \lambda\check{N}$ be two horizon lengths where $\lambda \in \{2, 3, 4, \ldots\}$. The corresponding sampling times are then defined by $T_c = \hat{N}\Delta\hat{T} = \check{N}\Delta\check{T}$. Since $\check{N} = \hat{N}/\lambda < \hat{N}$, the number of flops η and bits ρ (see Lemma 2) needed on the local node will be reduced for the MPC controller with horizon $\check{N}$ and $\Delta\check{T}$.

Increasing the sampling time from $\Delta\hat{T}$ to $\Delta\check{T}$ will typically result in a worse performance of the MPC controlled system. It is the intuition behind the approach proposed here that some of this decrease in performance can be compensated for by oversampling the slow MPC controller, i.e. by using the feedback created for the slow sampling with $\Delta\check{T}$ and $\check{N}$ at a faster frequency $1/\Delta\hat{T}$. We stress that oversampling is only attractive, because it does not result in solving QPs at the faster frequence $1/\Delta\hat{T}$. Essentially, the oversampling only makes sense, because the local node only evaluates cheap affine control laws at a higher frequency than before.

We recall the states of the continuous-time system (1) may be infeasible for the slow controller at time points $t \in (k\Delta T, (k+1)\Delta T)$ in between successive sampling times even if the states of the discrete-time system (2) are feasible for k and $k+1$ and the input $u(t)$ is chosen as in (5). It is therefore not surprising that the oversampled controller may be infeasible at some time points $k\Delta\hat{T}$ that lie in between two successive sampling points $l\Delta\check{T}$ of the slow discretized system. Consequently, we have to clarify at which points in time the controller is feasible for the fast discretized system if it is controlled with an oversampled MPC. The oversampled MPC is stated precisely in Algorithm 2. We show the controller described in this algorithm results in a feasible state for the slow system at least every λ steps of the fast sampling time $\Delta\hat{T}$ in the following lemma.

Lemma 3: Let $x(k) \in \check{\mathcal{X}}_f$ be an arbitrary feasible point for the slow system. Let $u(k+l)$, $l = 0, 1, \ldots$ be computed at frequency $1/\Delta\hat{T}$ according to Algorithm 2 and let $x(k+l+1) = \hat{A}x(k+l) + \hat{B}u(k+l))$, $l = 0, 1, \ldots$ be the

resulting successor states. Then there exists a $j \in [1, \lambda]$ such that $x(k+j) \in \check{\mathcal{X}}_f$, i.e. $x(k+j)$ is feasible for the slow system.

Proof: Since $x(k)$ is feasible for the slow system by assumption, $x(k)$ is either in the current polytope $\mathcal{P} = \{\xi \in \mathbb{R}^n : \check{T}^\star \xi \leq \check{d}^\star\}$ (i.e. lines 4-6 in Algorithm 2 apply) or $x(k)$ is in another polytope (i.e. lines 7-11 apply). In both cases, $u(k)$ and thus $x(k+1) = \hat{A}x(k) + \hat{B}u(k)$ are well-defined. One of the following cases applies to the successor state $x(k+1)$:

(i) $x(k+1) \in \mathcal{P}$, i.e. the successor state is feasible and lies in the same polytope as $x(k)$.

(ii) $x(k+1) \notin \mathcal{P}$ and $x(k+1) \in \check{\mathcal{X}}_f$, i.e. the successor state is feasible but does not lie in the same polytope as $x(k)$.

(iii) $x(k+1) \notin \check{\mathcal{X}}_f$, i.e. the successor state does not lie in the feasible set of the original MPC-controlled system.

In cases (i) and (ii), $x(k+1)$ is feasible and the claim holds with $j = 1$. In case (iii) the controller switches to zero order hold and applies the input signal that was optimal for the slow system at time step k. Keeping this input signal constant and applying it for λ fast time steps $\Delta\hat{T}$ is equivalent to applying $u(k)$ to the slow system for one type time step $\Delta\check{T}$ since $\Delta\check{T} = \lambda\Delta\hat{T}$. Since $u(k)$ is the optimal input for the feasible state $x(k)$ of the slow system, this results in a feasible state of the slow system. Consequently, a feasible state for the slow system results after $j = \lambda$ steps. Note that Algorithm 2 detects if there exists a feasible step after $j < \lambda$ steps and computes a new input if such a j exists. ∎

Data: $\check{A}, \check{B}, \check{\mathcal{X}}_f$
1 **Initialization:** $k = 0$, $x(0) \in \check{\mathcal{X}}_f$, $\check{\mathcal{A}}(x(0))$, $\check{K}^\star_{\mathcal{M}}$, $\check{b}^\star_{\mathcal{M}}$, $\check{T}^\star$, $\check{d}^\star$ for $\check{\mathcal{A}}(x(0))$;
2 **on local node** repeat at frequency $1/\Delta\hat{T}$
3 measure current state $x(k)$;
4 **if** $\check{T}^\star x(k) \leq \check{d}^\star$ **then**
5 $u(k) = \check{K}^\star_{\mathcal{M}} x(k) + \check{b}^\star_{\mathcal{M}}$;
6 $k \leftarrow k+1$;
7 **else if** $x(k) \in \check{\mathcal{X}}_f$ **then**
8 send $x(k)$ to central node and request $\check{\mathcal{A}}(x(k))$;
9 determine $\check{K}^\star_{\mathcal{M}}, \check{b}^\star_{\mathcal{M}}, \check{T}^\star$, $\check{d}^\star$ with Lemma 1;
10 $u(k) = \check{K}^\star_{\mathcal{M}} x(k) + \check{b}^\star_{\mathcal{M}}$;
11 $k \leftarrow k+1$;
12 **else**
13 $u(k) = u(k-1)$;
14 $k \leftarrow k+1$;
15 **end**
16 **end**
17 **on** central node
18 Solve (9), (11) for $x(k)$ to obtain $\mathcal{A}(x(k))$;
19 Transmit $\mathcal{A}(x(k))$ from central to local node;
20 **end**

Algorithm 2: Event triggered networked MPC with different sampling and discretization times

Algorithm 2 assumes the feasible set $\check{\mathcal{X}}_f$ to be available explicitly. This is a strong assumption, since feasible sets can usually not be calculated explicitly for all but small problems. Feasibility is checked at no additional cost if the central node attempts to solve the QP, however. The set $\check{\mathcal{X}}_f$ therefore need not be known explicitly, but the central node can flag infeasibility by sending one additional bit whenever the local node expects the active set. We use $\check{\mathcal{X}}_f$ for simplicity only.

B. Achievable Reduction

We compare the amount of data transmitted across the network, the computational effort on the local node, and the memory required on the local node for Algorithms 1 and 2. We carry out these comparisons under the assumption that all states and inputs are subject to lower and upper bounds for simplicity. The number of constraints in (9) then is

$$q = 2mN + 2nN + 2n. \tag{16}$$

The following Lemma states the reductions achieved by overclocking the MPC controller designed for $\Delta\check{T}$ by running it at the frequency $1/\Delta\hat{T}$ (Algorithm 2). All comparisons are carried out with respect to a controller with discretization time $\Delta\hat{T}$ run at frequency $1/\Delta\hat{T}$ (Algorithm 1). We compare the number of bits transmitted from the central to the local node, the co mputational effort for (12) on the local node, and the number of bits stored on the local node. We do not include the number of bits transmitted from the local to the central node in our comparison, because both approaches send the current state from the local to the central node whenever a new active set needs to be determined. Similarly, we do not include any computations on the local node in our comparisons that both approaches have in common. Let $\tilde{X}$ and $\hat{X}$ refer to any value that belong to the proposed approach ($\Delta\check{T}$ and $1/\Delta\hat{T}$) and the standard approach ($\Delta\hat{T}$ and $1/\Delta\hat{T}$), respectively. Note that the decoration with a tilde refers to the new case, where the controller designed for $\Delta\check{T}$ is used at the frequency $1/\Delta\hat{T}$.

Lemma 4: Let q be as in (16). The following statements hold:

(i) Let $\tilde{q}$ and $\hat{q}$ refer to the number of bits sent from the central to the local node in the proposed and the standard approach, respectively, whenever a new active set is requested. Then

$$\frac{\tilde{q}}{\hat{q}} = \frac{1}{\lambda} + \frac{1 - \frac{1}{\lambda}}{m/n\hat{N} + \hat{N} + 1}. \tag{17}$$

(ii) Let $\tilde{\eta}$ and $\hat{\eta}$ refer to the number of flops required to evaluate (12) in the proposed and the standard approach, respectively, whenever the local node receives a new active set. Then

$$\frac{\tilde{\eta}}{\hat{\eta}} \leq \frac{1}{\lambda}. \tag{18}$$

(iii) Let $\tilde{\rho}$ and $\hat{\rho}$ refer to the number of bits required to store the current affine law on the local node for the proposed and the standard approach, respectively. Then

$$\frac{\tilde{\rho}}{\hat{\rho}} \leq \frac{1}{\lambda}. \tag{19}$$

The statements of Lemma 4 can be proved with simple calculations. We sketch these calculations in the appendix. Since $m/n\hat{N} + \hat{N} + 1 \gg 1$ in (17) for most applications, $\tilde{q}/q \approx \frac{1}{\lambda}$ in part (i) of Lemma 4.

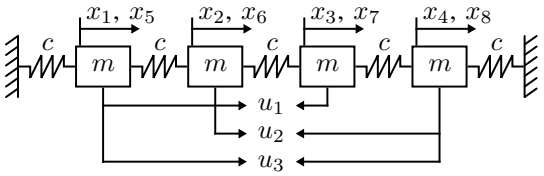

Fig. 2: Mechanical system with four oscillating masses. $m = c = 1$, no damping. States $x_i, i = 1, ..., 4$ are the position and $x_j, j = 5, ..., 8$ the velocity of the masses. There are three control inputs $u_l, l = 1, 2, 3$.

IV. Hardware-in-the-loop implementation

We use a standard desktop computer as the central node. The technical data read: Intel Core2 Duo CPU with two 3.0 GHz cores and 8 GB RAM. The central node is connected via 1Gbps Ethernet to a IEEE 802.11 b/g/n wireless LAN access point. As the local node, we use an Espressif ESP8266[1] SoC with an integrated IEEE 802.11 b/g/n WiFi controller. The SoC features a 80MHz Tensilica L106 32-bit RISC micro controller and 96 KiB data RAM. The local node is connected to a dSpace hardware-in-the-loop (HIL) simulator that simulates the behavior of the sample systems. Measurements of the state are done by analog to digital converters and analog control inputs are generated by digital to analog converters. Both feature a resolution of 12 bit.

A. Sample Systems

We use three different sample systems of different sizes. For illustration purposes, we use the double integrator (DI) that results from discretizing the continuous-time system

$$\dot{x} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x + \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} u \tag{20}$$

with the discretization time $\Delta\hat{T} = 1.0\,\mathrm{s}$ and the horizon $\hat{N} = 4$. The continuous-time weighting matrices read $Q_c = I^{n\times n}$ and $R_c = 1$. We choose the constraints $x_i \in [-3, 3], i = 1, 2,\ u \in [-2, 2]$. The second system (referred to as US) contains an unstable pole and results from discretizing

$$\dot{x} = \begin{bmatrix} -1 & 0.3 \\ 0.1 & 4 \end{bmatrix} x + \begin{bmatrix} 0.5 \\ -2 \end{bmatrix} u \tag{21}$$

with $\Delta\hat{T} = 0.5\,\mathrm{s}$ and the horizon $\hat{N} = 12$. The weighting matrices read $Q_c = I^{n\times n}$ and $R_c = 1$. The input and state constraints read $x_i \in [-3, 3], i = 1, 2,\ u \in [-2, 2]$.

Example 3, illustrated in Fig. 2, is a mechanical system with four masses connected by springs (4M for short). Masses and spring constants have the value one and there is no damping. State variables $x \in \mathbb{R}^8$ and input variables $u \in \mathbb{R}^3$ are subject to the constraints $x_i \in [-3, 3], i = 1, ..., 8$ and $u_l \in [-2, 2], l = 1, 2, 3$. Cost function matrices are chosen to be $Q_c = I^{n\times n}$ and $R_c = I^{m\times m}$ and the discretization time is $\Delta\hat{T} = 0.25\,\mathrm{s}$. All systems and their respective value functions are discretized as stated in (3) and (8). The reduced controller is calculated with $\lambda = 2$ for the DI and US system, and $\lambda = 3$ for the 4M system.

[1]See https://espressif.com/en/products/hardware/esp8266ex/overview for more information on the ESP8266 (checked on May 15, 2017).

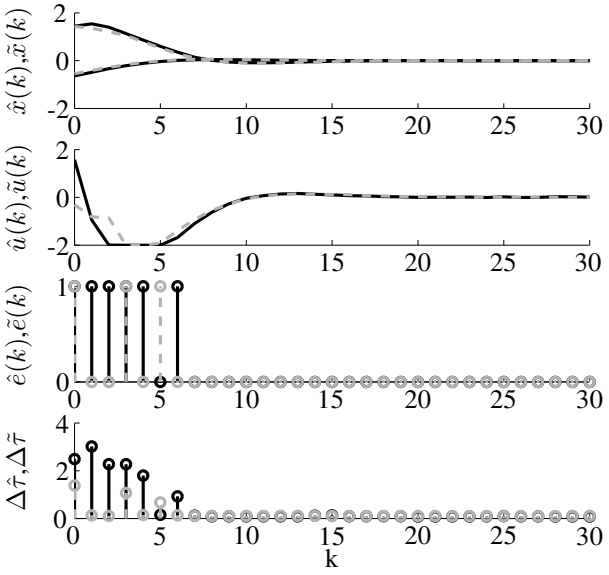

Fig. 3: DI sample system. Solid and dashed lines indicate data from the non-reduced MPC and the reduced MPC. The data is recorded with sampling time $\Delta\hat{T}$. Computation times $\Delta\tau$,$\Delta\tilde{\tau}$ are given in milliseconds.

TABLE I: Measured complexity for three different systems

Sys	m	n	$\Delta\hat{T}$	$\hat{N}$	$\hat{q}$	$\overline{\Delta\hat{\tau}}$	$\hat{\rho}$	$\tilde{q}$	$\overline{\Delta\tilde{\tau}}$	$\tilde{\rho}$	$\frac{\tilde{q}}{\hat{q}}$	$\frac{\tilde{\eta}}{\hat{\eta}}$	$\frac{\tilde{\rho}}{\hat{\rho}}$
DI	1	2	1	4	32	3.79	1.4	18	1.76	0.6	0.56	0.46	0.45
US	1	2	.5	12	76	5.4	15.1	66	2.1	4.7	0.52	0.38	0.31
4M	3	8	.25	10	-	-	85.4	206	28.45	25.7	-	-	0.30

Quantities q, $\overline{\Delta\tau}$ and ρ are measured in Bits, milliseconds and kBytes, respectively, and are stated for a single event. $\overline{\Delta\tau}$ denotes the average computation time that is measured from 1000 random start points. Note that sample system 4M can only be controlled with the reduced controller, since the non-reduced one does not fit on the local node.

B. Results

We compare Algorithm 1, where discretization and sampling time are both $\Delta\hat{T}$ to Algorithm 2 with discretization time $\Delta\tilde{T}$ and sampling time $\Delta\hat{T}$. Let $\hat{X}$ and $\tilde{X}$ refer to data that belongs to Algorithm 1 and Algorithm 2, respectively, again. We refer to the respective controllers as non-reduced and reduced MPC for short.

Since we cannot measure flops on the micro controller directly we state measurements of the computation time and assume that flops and computation time are related by a constant factor (i.e. $\Delta\tau = \alpha\eta,\ \alpha \in \mathbb{R}^+$). Computation time is denoted $\Delta\tau$ and refers to the time that is necessary to compute (12) and (13a) on the local node. We provide measurements for 1000 random feasible start points for each sample system and denote the average computation time $\overline{\Delta\tau}$.

We first discuss the time series depicted in Fig. 3 that shows the states x, inputs u, events e and local computation time $\Delta\tau$ of the DI system for both controllers. The system is regulated from a random, feasible point to the origin. Trajectories from the non-reduced and reduced MPC are shown with solid black lines and dashed grey lines, respectively. As apparent from Fig. 3 (first and second plot), state and input trajectories almost coincide. The variable $e(x)$ in Fig. 3 indicates an event ($e(x) = 1$ if an event occurred). Measurements of the computation time (bottom

plot in Fig. 3) show that the control law is evaluated at very low computational effort if no event occurs, and that the computation time is reduced by our approach in case of an event, as shown in Lemma 4.

Table I compares the number of transmitted (q) and stored (ρ) bits, and average computation times for the reduced and non-reduced MPC. The measured reductions corroborate the results stated in Lemma 4. Note that $\tilde{q}/\hat{q} \approx \frac{1}{\lambda}$ applies in all cases. The memory requirements of the third sample system (4M) exceeds the memory of the ESP8266 for the non-reduced controller. Hence, the controller can only be realized due to the proposed reduction approach.

V. Conclusions and Outlook

We proposed a method that reduces local computational effort, storage requirements and transmitted data of a networked MPC approach in which local nodes evaluate affine feedback laws that they receive from a powerful, QP-solving central node. We use the optimal control laws provided by the central node at an overclocked frequency on the local node. This results in smaller computational and memory requirements on the local node and smaller amounts of data sent across the network compared to network MPC with the same prediction horizon length without overclocking. The reduction is essentially proportional to the overclocking factor λ for all three quantities of interest. We analyzed the proposed approach with an implementation on embedded hardware and corroborated the expected reductions with three examples. In fact, the largest example considered was only tractable with the embedded hardware because of the reduction achieved with overclocking. Future work has to establish the stability properties of the proposed overclocked approach.

Appendix

Part (i) of Lemma 4 can easily be shown by computing the number of constraints $\hat{q}$ and $\tilde{q}$ in (16) with $\hat{N}$ and $\check{N}$, respectively.

Evaluating (14) for $\tilde{\eta}$ and $\hat{\eta}$ results in

$$\frac{\tilde{\eta}}{\hat{\eta}} = \frac{1}{\lambda}\left(\frac{a/\lambda^2 + b/\lambda + c + d\lambda}{e}\right) \tag{22}$$

with

$a = 20\hat{N}^3m^2 + 12\hat{N}^3mn$,
$b = 6\hat{N}^2m^2 + 24\hat{N}^2mn + 12\hat{N}^2n^2 + 21\hat{N}^2m + 6\hat{N}^2n$,
$c = 12\hat{N}mn + 12\hat{N}n^2 + 3\hat{N}m + 3\hat{N}n + 7\hat{N}$,
$d = -3n - 3$ and
$e = a+b+c+d$. Since $e = a+b+c+d$, $\lambda > 1$ and $d < 0$, the expression in the brackets in (22) is smaller than one. This proves part (ii).

Evaluating (15) for $\tilde{\rho}$ and $\hat{\rho}$ results in

$$\frac{\tilde{\rho}}{\hat{\rho}} = \frac{1}{\lambda}\left(\frac{f/\lambda + g}{l}\right) + \frac{h}{l}, \tag{23}$$

where $f = 3\,\hat{N}m^2 + 2\,\hat{N}mn$, $g = 7\,mn + 4\,n^2 + 4\,m + 4\,n$, $h = 4n^2 + 5n + 1$ and $l = f + g + h$. Since $l \gg h$, the contribution of $\frac{h}{l}$ to $\frac{\tilde{\rho}}{\hat{\rho}}$ can be neglected. Furthermore, $\lambda > 1$ by definition, which implies that the expression in the brackets in (23) is smaller than one. This proves part (iii).

References

[1] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit solution of model predictive control via multiparametric quadratic programming," in *Proceedings of the American Control Conference*, 2000, pp. 872–876.

[2] A. Bemporad, M. Morari, V. Dua and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3 – 20, 2002.

[3] M. Seron, G. Goodwin, and J. DeDona, "Characterisation of Receding Horizon Control for constrained systems," *Asian Journal of Control*, vol. 5, no. 2, pp. 271 – 286, 2003.

[4] M. Klaučo, M. Kalúz, and M. Kvasnica, "Real-time implementation of an explicit MPC-based reference governor for control of a magnetic levitation system," *Control Engineering Practice*, vol. 60, pp. 99 – 105, 2017.

[5] Marek Honek and Michal Kvasnica and Alexander Szűcs and Peter Šimončič and Miroslav Fikar and Boris Rohal'-Ilkiv, "A low-complexity explicit MPC controller for AFR control," *Control Engineering Practice*, vol. 42, pp. 118 – 127, 2015.

[6] A. G. Beccuti, S. Mariethoz, S. Cliquennois, S. Wang, and M. Morari, "Explicit Model Predictive Control of DC–DC Switched-Mode Power Supplies With Extended Kalman Filtering," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 6, pp. 1864–1874, 2009.

[7] P. Zometa, M. Kögel, and R. Findeisen, "μAO-MPC: A free code generation tool for embedded real-time linear model predictive control," in *Proceedings of the American Control Conference*, 2013, pp. 5320–5325.

[8] B. Houska, H. J. Ferreau, and M. Diehl, "ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization," *Optimal Control Appl. & Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[9] Petter Tøndel and Tor A. Johansen, "Complexity Reduction in Explicit Linear Model Predictive Control," in *Proceedings of the 15th IFAC World Congress*, vol. 35, no. 1, 2002, pp. 189 – 194.

[10] R. Cagienard and P. Grieder and E.C. Kerrigan and M. Morari, "Move blocking strategies in receding horizon control ," *Journal of Process Control* , vol. 17, no. 6, pp. 563 – 570, 2007.

[11] W. P. M. H. Heemels, J. H. Sandee, and P. P. J. Van den Bosch, "Analysis of event-driven controllers for linear systems," *International Journal Of Control*, vol. 81, no. 4, pp. 571–590, 2008.

[12] W. P. M. H. Heemels, K. H. Johansson, and P. Tabuada, "An introduction to event-triggered and self-triggered control," in *51st Annual Conference on Decision and Control (CDC)*, 2012, pp. 3270–3285.

[13] J. Hespanha, P. Naghshtabrizi, and Y. Xu, "A Survey of Recent Results in Networked Control Systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 138–162, 2007.

[14] A. Eqtami, D. Dimarogonas, and K. Kyriakopoulos, "Event-triggered strategies for decentralized model predictive control," in *Proceedings of the 18th IFAC World Congress*, 2011, pp. 10 068–10 073.

[15] P. Varutti, B. Kern, T. Faulwasser, and R. Findeisen, "Event-based model predictive control for Networked Control Systems," in *Proceedings of the 48th IEEE Conference on Decision and Control, jointly with the 28th Chinese Control Conference (CDC/CCC)*, 2009, pp. 567–572.

[16] D. E. Quevedo, P. K. Mishra, R. Findeisen, and D. Chatterjee, "A stochastic model predictive controller for systems with unreliable communications," *Preprints, 5th IFAC Conference on Nonlinear Model Predictive Control*, pp. 57–64, 2015.

[17] A. Bemporad, M. Heemels, and M. Vejdemo-Johansson, *Networked Control Systems*. Springer Verlag, 2010.

[18] P. S. Berner and M. Mönnigmann, "A Comparison of Four Variants of Event-Triggered Networked MPC," in *Proceedings of the 2016 IEEE Multi-Conference on Systems and Control*, 2016, pp. 1519–1524.

[19] M. Jost, M. Schulze Darup, and M. Mönnigmann, "Optimal and suboptimal event-triggering in linear model predictive control," in *Proceedings of the European Control Conf.*, 2015, pp. 1147–1152.

[20] J. Araújo, A. Teixeira, E. Henriksson, and K. H. Johansson, "A down-sampled controller to reduce network usage with guaranteed closed-loop performance," in *53rd IEEE Conference on Decision and Control*, Dec 2014, pp. 6849–6856.

[21] K. Åström and B. Wittermark, *Computer-Controlled Systems*, 3rd ed. Prentice-Hall, 1997.

[22] E. G. Gilbert and K. T. Tan, "Linear systems with state and control constraints: the theory and application of maximal output admissible sets," *IEEE Transactions on Automatic Control*, vol. 36, no. 9, pp. 1008–1020, Sep 1991.

[23] J. Maciejowski, *Predictive Control: With Constraints*, ser. Pearson Education. Prentice Hall, 2002.

[24] C. Ueberhuber, *Numerical Computation 1*. Springer Berlin Heidelberg, 1997.

[25] "IEEE Standard for Floating-Point Arithmetic," *IEEE Std 754-2008*, pp. 1–70, Aug 2008.