A Distributed Active Set Method for Model Predictive Control

Gösta Stomberg∗ Alexander Engelmann∗ Timm Faulwasser∗

∗ Institute for Energy Systems, Energy Efficiency and Energy Economics, Faculty of Electrical Engineering and Information Technology, TU Dortmund University, 44227 Dortmund, Germany
(e-mail: {goesta.stomberg,alexander.engelmann}@tu-dortmund.de, timm.faulwasser@ieee.org)

Abstract: This paper presents a novel distributed active set method for model predictive control of linear systems. The method combines a primal active set strategy with a decentralized conjugate gradient method to solve convex quadratic programs. An advantage of the proposed method compared to existing distributed model predictive algorithms is the primal feasibility of the iterates. Numerical results show that the proposed method can compete with the alternating direction method of multipliers in terms of communication requirements for a chain of masses example.

Keywords: distributed control, model predictive control, active set methods, conjugate gradient methods, ADMM

1. INTRODUCTION

In many control applications such as process control, distributed model predictive control (DMPC) is promising due to the absence of central coordination and a limited information exchange. DMPC schemes are often realized by solving optimal control problems (OCPs)—respectively, appropriate discretizations thereof—by means of distributed optimization algorithms. A challenge is that many distributed optimization algorithms guarantee consensus constraint satisfaction only asymptotically. This implies that per sampling instant usually many distributed optimization iterations have to be executed in order to ensure closed-loop properties such as stability and recursive feasibility (Kögel and Findeisen, 2012; Mota et al., 2012; Pu et al., 2014; Rostami et al., 2017). To overcome this limitation and to enable early termination of the optimization, constraint-tightening approaches have been proposed in Doan et al. (2011); Giselsson and Rantzer (2014). Wang and Ong (2017) extend these approaches to state constraints. However, constraint tightening potentially leads to a loss in control performance, or to a reduced domain of attraction resulting from the tightened constraint set.

A second line of research enables decomposition via coordinate descent methods, where each subsystem solves its own OCP while keeping the influence of the neighboring systems fixed. Such methods guarantee feasible iterates (Necoara, 2012). A Jacobi-type method has been proposed by Stewart et al. (2010), where coupling variables are averaged between subsystems and a linear rate of convergence has been shown by Groß and Stursberg (2013). This approach was extended to event-based communication in Groß and Stursberg (2016). A condition for verifying convergence a posteriori is presented in (Dang Doan et al., 2017), where the authors also provide convergence guarantees for the special case of chain-linked systems.

However, a general shortcoming of Jacobi-type iterations is that—although they decrease the objective in every iteration—convergence to a minimizer is often not guaranteed. Moreover, for fast convergence they require carefully chosen weights. For a comprehensive overview on DMPC approaches we refer to Müller and Allgöwer (2017).

This paper aims to overcome the conservatism introduced by constraint tightening. We introduce a novel distributed active set method (ASM) that builds upon a decentralized conjugate gradient (DCG) method. The method converges to the exact solution of an OCP in a finite number of iterations and thus avoids constraint tightening. Moreover, the proposed method has the advantage that the iterates remain feasible after a feasible initial point has been computed. This makes early termination possible without losing recursive feasibility guarantees. Furthermore, our method is tuning-free, has a practically super-linear convergence rate and numerical examples indicate a low communication footprint.

The paper is structured as follows. Section 2 formulates the DMPC scheme for the control of a linear system network. Section 3 presents the proposed distributed ASM for solving the arising QPs in the DMPC scheme. Section 4 presents numerical results on an example chain of masses system where our method is compared to ADMM. Appendix A briefly discusses the ADMM variant we use as a benchmark.

2. PROBLEM STATEMENT

We consider a network of linear time-invariant systems\
i ∈ M = {1, . . . , M} with dynamics\

\begin{equation}
x_i^+ = A_{ii} x_i + B_i u_i + \sum_{j \in M^+} A_{ij} x_j, \quad x_i(0) = x_{i,0}.
\end{equation}
where \( x_i \in \mathbb{R}^{n_i} \) and \( u_i \in \mathcal{U}_i \subset \mathbb{R}^{m_i} \) denote the state and input of agent \( i \). We use a subscript \((\cdot)_i\) throughout the paper to mark a variable’s association to agent \( i \). The matrices \( A_{ij} \in \mathbb{R}^{n_i \times n_j} \) create state couplings between neighboring agents. We distinguish between in-neighbors \( M_i^\text{in} = \{ j \in \mathcal{M} | A_{ij} \neq 0 \} \) and out-neighbors \( M_i^\text{out} = \{ j \in \mathcal{M} | A_{ji} \neq 0 \} \) and we denote their union by \( M_i = M_i^\text{in} \cup M_i^\text{out} \). The dynamics (1) can be rewritten as

\[
x_i^+ = A_{ii} x_i + B_i u_i + \sum_{j \in M_i^\text{in}} A_{ij} v_{ji}, \quad x_i(0) = x_{i,0} \tag{2}
\]

where \( v_{ji} \equiv x_j \) are state copies. We gather the state copies of agent \( i \)’s in-neighbors in

\[
v_i \equiv \text{col}_{j \in M_i^\text{in}}(v_{ji}) \in \mathbb{R}^{n_i}.
\]

**Assumption 1. (Input Constraints).** The sets \( \mathcal{U}_i \) are compact convex polytopes and contain the origin in their interior. \( \square \)

**Remark 1. (State Constraints).** For the sake of simplicity, we consider only input constraints in the present paper. Our approach can be extended to include state constraints as well as terminal constraints. \( \square \)

Let \( x \equiv \text{col}_{i \in \mathcal{M}}(x_i) \in \mathbb{R}^N \) denote the overall system state and \( u \equiv \text{col}_{i \in \mathcal{M}}(u_i) \in \mathbb{R}^m \) the overall input. The overall system dynamics are \( x^+ = Ax + Bu \), where \( A \in \mathbb{R}^{N \times N} \) is a block matrix with entries \( A_{ij} \) and \( B = \text{diag}_{i \in \mathcal{M}}(B_i) \in \mathbb{R}^{N \times m} \). The Assumption 2. (Controllability). The overall system \((A, B)\) is controllable. \( \square \)

### 2.1 MPC Formulation

Let \( x_i, u_i \) and \( v_i \) denote the state, input and copy sequences \( x_i^\text{T} = [x_{i,0}^T, \ldots, x_{i,N-1}^T] \), \( u_i^\text{T} = [u_{i,0}^T, \ldots, u_{i,N-1}^T] \) and \( v_i^\text{T} = [v_{i,0}^T, \ldots, v_{i,N-1}^T] \) for agent \( i \) with horizon \( N \). The OCP reads

\[
\min_{x_i, u_i, v_i} \sum_{i \in \mathcal{M}} J_i(x_i, x_i^N, u_i) \tag{3a}
\]

subject to for all \( i \in \mathcal{M} : \)

\[
x_{i}^+ = A_{ii} x_{i} + B_i u_i + \sum_{j \in M_i^\text{in}} A_{ij} v_{ji}, \quad x_{i,0} = x_{i,0}, \tag{3b}
\]

\[
u_i \in \mathcal{U}_i, \tag{3c}
\]

\[
v_{ji} = x_j \quad \forall j \in M_i^\text{in}, \tag{3d}
\]

where the objective for agent \( i \in \mathcal{M} \) is given by

\[
J_i(x_i, x_i^N, u_i) = \frac{1}{2} x_i^N P_i x_i^N + \frac{1}{2} \sum_{k=1}^{N-1} (x_i^{kT}Q_ix_i^k + u_i^{kT} R_i u_i^k).
\]

**Assumption 3.** The weighting matrices \( Q_i \) and \( R_i \) are positive definite. The matrices \( P_i \) are positive semi-definite. \( \square \)

Given measured or observed states \( x_{i,0} \), the MPC scheme solves this OCP for the optimal input sequences \( u_i^* \) and then applies the first part of the input sequence \( u_{i,0}^* \) as a control input to each agent. This MPC scheme asymptotically stabilizes the origin given Assumptions 1–3 for sufficiently long horizons \( N \) on a set of initial conditions \( \mathcal{X}_0 \). The set \( \mathcal{X}_0 \) on which closed-loop stability can be guaranteed depends on the interplay of the input constraints \( \mathcal{U} \) and the stability of \( A \), cf. (Boccia et al., 2014).

### 2.2 Distributed OCP Formulation

We next aggregate the state, input and copy trajectories into decision variables for each agent into \( z_i^T \equiv [x_i^T, x_i^T, v_i^T, v_i^T] \in \mathbb{R}^{Nn_i} \) where \( x_i^T \equiv [x_{i,0}^T, \ldots, x_{i,N-1}^T] \in \mathbb{R}^{n_i} \) and \( z_{ji} = (N+1)i + N(n_j + 1) \). The decision variables \( z_i \) are not typeset bold to simplify the notation even though they contain trajectories. With this we rewrite OCP (3) as the partially separable quadratic program (QP)

\[
\begin{align}
\min_{z_i} & \quad \sum_{i \in \mathcal{M}} \frac{1}{2} z_i^T H_i z_i \tag{4a} \\
\text{s.t.} & \quad C_i^E z_i = b_i^E \quad \forall i \in \mathcal{M}, \tag{4b} \\
& \quad C_i^T z_i \leq b_i^T \quad \forall i \in \mathcal{M}, \tag{4c} \\
& \quad \sum_{i \in \mathcal{M}} C_i^F z_i = 0. \tag{4d}
\end{align}
\]

The Hessian matrix is given by

\[
H_i = \text{diag}(\bar{Q}_i, P_i, R_i, \bar{Q}_i), \quad \bar{Q}_i \in \mathbb{R}^{n_i \times n_i}, \tag{5}
\]

where \( \bar{Q}_i \equiv 1/(|M_i^\text{out}| + 1) \cdot \text{diag}(Q_i, \ldots, Q_i) \in \mathbb{R}^{Nn_i \times Nn_i} \) and \( R_i = \text{diag}(R_i, \ldots, R_i) \in \mathbb{R}^{Nn_i \times Nn_i} \). This choice of \( \bar{Q}_i \) spreads the cost associated with state \( x_i \) evenly among agent \( i \) and its outgoing neighbors. The equality constraints \((4b)\) with \( C_i^E \in \mathbb{R}^{n_i \times n_{zi}} \) include the initial condition and dynamics \((2)\); the inequality constraints \((4c)\) include the input constraints \((3c)\); and the coupling constraints \((4d)\) with \( C_i^F \in \mathbb{R}^{n_{zi} \times n_{zi}} \) include the coupling between states and copies \((3d)\). We propose to solve OCP \((4)\) with the following distributed ASM.

### 3. DISTRIBUTED ACTIVE SET METHOD

Active set methods (ASMs) are well-known for solving inequality-constrained convex QPs. In particular, primal ASMs produce primal feasible iterates (Nocedal and Wright, 2006), which is advantageous for MPC. We next present a distributed primal ASM for solving OCP \((4)\).

#### 3.1 Distributed Active Set Method

Let \( c_i^T \) denote the \( j \)th row of \( C_i^E \) and \( b_i^T \) the \( j \)th element of \( b_i^T \). The active set of agent \( i \) is given by

\[
A(z_i) = \{ j | c_i^T z_i = b_i^T \}.
\]

The proposed method first chooses an initial active set \( A(z_i^0) \) and a corresponding iterate \( z_i^0 \) for each agent. Afterwards, the method takes steps \( z_i^{k+1} = z_i^0 + \alpha^k \Delta z_i^0 \) until it has converged. The step directions \( \Delta z_i^0 \) are obtained by solving the equality constrained QP

\[
\begin{align}
\min_{z_i} & \quad \sum_{i \in \mathcal{M}} \frac{1}{2} (\Delta z_i^T H_i \Delta z_i + g_i^T \Delta z_i) \tag{6a} \\
\text{s.t.} & \quad C_i^0 \Delta z_i = d_i \quad | \gamma_i \forall i \in \mathcal{M}, \tag{6b} \\
& \quad \sum_{i \in \mathcal{M}} C_i^0 \Delta z_i = 0 \quad | \lambda C \tag{6c}
\end{align}
\]

where \( C_i^0 \equiv [\text{col}_{j \in A(z_i)} c_i^T] \), \( g_i^0 = H_i z_i^0 \) and \( d_i = 0 \). Here, \( \gamma_i \) and \( \lambda C \) are the Lagrange multipliers associated with the constraints \((6b)\) and \((6c)\). Then, each agent
computes the largest step length $\alpha^n_i \in (0,1]$ such that primal feasibility is maintained (Nocedal and Wright, 2006):

$$o^n_i = \min \left\{ 1, \min_{j \not\in A(z^n_i)} \frac{b^T_j - c^T_j z^n_i}{c_i^T p^n_i} \right\}$$

(7)

The step length $\alpha^n = \min\{\alpha^n_1, \ldots, \alpha^n_M\}$ is then chosen for all agents to obtain $z^{n+1}$. This choice ensures that $z^{n+1}$ remains feasible for all agents. If $\alpha < 1$, then an inactive constraint of one agent is blocking and the constraint is added to the respective agent’s active set. This process is repeated until $\|\Delta z^n\| < \varepsilon$ for all $i \in \mathcal{M}$ where $\varepsilon$ is small. Then, each agent checks for dual feasibility

$$\gamma_i^n = 0, \quad \forall j \in A(z^n_i).$$

If dual feasibility is attained for all agents, then $z^n$ is returned as solution to problem (6). Else the constraint corresponding to the smallest Lagrange multiplier among all agents is removed from the respective agent’s active set.

Algorithm 1 summarizes the ASM and it further contains the condensing and DCG steps that we subsequently describe in Subsections 3.2 and 3.3.

**Remark 2. (ASM initialization).** Different ways of initializing ASMs with $A(z^n_i)$ and $z^n_i$ have been reported in the literature. Ferreau et al. (2008) present an ASM that is initialized on a homotopy path between the current and the previous sample. Klaučo et al. (2019) propose to warm-start ASMs with classification methods from supervised learning. Here, we initialize with the optimal active set from the previous MPC iteration as a warm-start or with $A(z^n_i) = \emptyset$ if no previous MPC iteration is available. We then find $z^n_i$ by solving a QP similar to (4) but with (4c) replaced by the active set. If the obtained $z^n_i$ violates an inequality constraint, then this constraint is added to the active set. This is repeated until a feasible initialization is obtained.

**Remark 3. (Early termination).** The active set method produces primal feasible iterates that satisfy (4b)-(4d). It can therefore be terminated early and still guarantee stability for MPC schemes with terminal constraints where feasibility implies stability (Stoakert et al., 1999).

**Algorithm 1 Distributed ASM**

**Initialization:** $A(z^n_i)$ and $z^n_i$.

**Repeat until convergence:**

1. Condense QP (6) into (8).
2. Solve (8) with DCG for $\lambda_C$.
3. Obtain $\Delta z_i$ via backsubstitution (12) for all $i \in \mathcal{M}$.
4. If $\|\Delta z_i\| < \varepsilon, \forall i \in \mathcal{M}$,
   - $\gamma^n_i = (C_i^T C_i)^{-1} C_i^T (-g_i - C_i^T \lambda_C^n) \forall i \in \mathcal{M}$.
   - If $\gamma_i^n \geq 0, \forall j \in A(z^n_i), \forall i \in \mathcal{M}$,
     - Return $z^n_i$ and terminate.
5. Else,
   - Find the smallest $\gamma_i^n$ among all $i \in \mathcal{M}$.
   - Remove the respective constraint $j$ from $A(z^n_i)$.
   - Compute $\alpha^n_i$ locally according to (7) for all $i \in \mathcal{M}$.
   - Determine agent with the smallest $\alpha_i$ and set $\alpha = \alpha_i$.
   - $A(z^n_i) \leftarrow A(z^n_i) \cup \{j\}$ where $j$ is the blocking constraint.
   - Local step: $z^{n+1}_i \leftarrow z^n_i + \alpha^n_i \Delta z^n_i$ for all $i \in \mathcal{M}$.

**3.2 Condensing the QP**

The core idea of the distributed ASM is to solve QP (6) with a decentralized conjugate gradient method (DCG) from (Engelmann and Faulwasser, 2021). To apply DCG, we condense and rewrite QP (6) as

$$\sum_{i \in \mathcal{M}} S_i \lambda_C = \sum_{i \in \mathcal{M}} s_i,$$

(8)

where $S_i$ and $s_i$ are yet to be defined. In particular, $\sum_{i \in \mathcal{M}} S_i$ is positive definite and the matrices $S_i$ have favorable sparsity properties that motivate DCG.

To arrive at (8), we first apply the nullspace method (Nocedal and Wright, 2006) to eliminate the equality constraints $C_i \in \mathbb{R}^{n_i \times n_n}$. Let

$$\Delta z_i = Z_i v_i + Y_i w_i,$$

(9)

where the columns of $Z_i \in \mathbb{R}^{n_i \times (n_n-n_i)}$ form a nullspace of $C_i$, and $Y_i \in \mathbb{R}^{n_i \times n_n}$ is chosen such that $[Z_i, Y_i]$ is invertible. This choice of $Y_i$ together with $Z_i, Z_i = 0$ result in $C_i Y_i$ being nonsingular and inserting into the equality constraints yields $w_i = (C_i Y_i)^{-1} d_i$. We insert (9) into (6) and obtain

$$\min_{v_i} \sum_{i \in \mathcal{M}} \left( \frac{1}{2} v_i^T \bar{H}_i v_i + \bar{g}_i^T v_i \right)$$

(10a)

s.t.

$$\sum_{i \in \mathcal{M}} (\bar{C}_i^T v_i + b_i) = 0 \quad \lambda_C$$

(10b)

where $\bar{H}_i = Z_i^T H_i Z_i, \bar{g}_i = Z_i^T g_i + Z_i^T H_i Y_i w_i, \bar{C}_i = C_i Z_i$ and $b_i = C_i Y_i w_i$. The KKT conditions of (10) read

$$\begin{bmatrix} \bar{H}_1 & \bar{C}_1^T \\ \vdots & \vdots \\ \bar{H}_M & \bar{C}_M^T \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_M \end{bmatrix} = \begin{bmatrix} -\bar{g}_1 \\ \vdots \\ -\bar{g}_M \end{bmatrix} \quad \lambda_C$$

(11)

Assumption 4. The matrices $\bar{H}_i$ are positive definite. \(\Box\)

Utilizing that $\bar{H}_i$ is positive definite for all $i \in \mathcal{M}$, we solve (11) for $v_i = \bar{H}_i^{-1} (-\bar{g}_i - \bar{C}_i^T \lambda_C)$. Inserting back into the KKT conditions yields (8) when $S_i = \bar{C}_i^T \bar{C}_i^{-1} \bar{C}_i^T$ and $s_i = b_i - \bar{C}_i^T \bar{H}_i^{-1} \bar{g}_i$. The QP (6) has hence been reduced to a set of $n_c$ linear equations. The solution to QP (6) can be obtained via backsubstitution:

$$\Delta z_i = Z_i \bar{H}_i^{-1} (-\bar{g}_i - \bar{C}_i^T \lambda_C) + Y_i (C_i Y_i)^{-1} d_i$$

(12)

**Remark 4.** The above derivation allows for $d_i \neq 0$ in (6b). This is needed to perform the ASM initialization with DCG as described in Remark 2. Once the ASM initialization is complete, we use DCG to find $\Delta z_i$ and in this case $d_i = 0$.

**3.3 The Decentralized Conjugate Gradient Method**

Next, we recall the decentralized conjugate gradient (DCG) method from (Engelmann and Faulwasser, 2021) to solve the positive definite linear system of equations (8) in decentralized fashion. Consider the system of equations

$$S \lambda_C = s,$$

where $S \triangleq \sum_{i} S_i$ is positive definite and $s \triangleq \sum_{i} s_i$. The centralized conjugate gradient method (Nocedal and Wright, 2006) solves this system iteratively for $\lambda_C$ until the
residual at iteration $n$ defined as $r^n = s - S\lambda^n_C$ vanishes. The iterations read
\[
\alpha^n = \frac{r^{nT}r^n}{p^{nT}Sp^n}, 
\lambda^n_\mathcal{C} = \lambda^n_C + \alpha^n p^n, 
\begin{align*}
\epsilon^n &= r^{nT}r^n - \alpha^n p^n, 
\beta^n &= \frac{\epsilon^n + r^{nT}r^n}{p^{nT}p^n}, 
\end{align*}
\]
with the initialization $r^0 = p^0 = s - S\lambda^0_C$. The idea of DCG is to introduce local versions of the CG variables $\lambda_C$, $r$ and $p$ and to decompose the CG updates into local updates. Since DCG is a reformulation of the centralized CG method, DCG is guaranteed to solve the QP (6) in a finite number of iterations and exhibits a practically superlinear convergence rate.

We exploit sparsity properties of the $S_i$ matrices that result from zero rows in $C^c_i$. The rows of $C^c_i$ couple variables between two agents: the states of one agent and $\alpha(I)$. We obtain by taking the identity matrix $I$ that map from global CG variables to local variables. The matrices $I_{C(i)}$ are obtained by taking the identity matrix $I \in \mathbb{R}^n \otimes n$ and by subsequently eliminating all rows that do not belong to any constraint in $\mathcal{C}(i)$. With this we introduce the local variables
\[
\lambda_{\mathcal{C}, i} \triangleq I_{C(i)} \lambda_C, 
\end{align*}
\]
\[
\begin{align*}
r_i &= I_{C(i)} r, 
\sigma_i &= I_{C(i)} p, 
\end{align*}
\]
\[
\text{Let } \Lambda \triangleq \sum_{i \in \mathcal{M}} I_{C(i)}^T I_{C(i)} \text{ and } \Lambda_i \triangleq I_{C(i)} \Lambda I_{C(i)}^T. \text{ We rewrite (13a) as } \alpha^n = \frac{\eta^n}{\sigma^n} \text{ where } \eta^n = r^{nT}r^n \text{ and } \sigma^n = p^{nT}Sp^n. \text{ This can be decomposed into }
\end{align*}
\]
\[
\begin{align*}
\eta^n &= \sum_{i \in \mathcal{M}} \eta_i^n, 
\sigma^n &= \sum_{i \in \mathcal{M}} \sigma_i^n, 
\end{align*}
\]
\[
\text{where } \hat{S}_i \triangleq I_{C(i)} S_i I_{C(i)}^T. \text{ The computation of } \alpha^n \text{ therefore requires the local computation of } \eta_i \text{ and } \sigma_i \text{ for each agent and two subsequent scalar global sums. We next decentralize the updates of the Lagrange multiplier } \lambda_C \text{ and the step direction } p. \text{ We multiply (13b) and (13c) by } I_{C(i)} \text{ from the left and get }
\end{align*}
\]
\[
\begin{align*}
\lambda_{\mathcal{C}, i}^{n+1} &= \lambda_{\mathcal{C}, i}^n + \eta_i^n \sigma_i^n p_i^n, 
p_i^{n+1} &= r_i^{n+1} + \beta^n p_i^n, 
\end{align*}
\]
\[
\text{with residual update }
\end{align*}
\]
\[
\begin{align*}
r_i^{n+1} &= r_i^n - \frac{\eta^n}{\sigma^n} \sum_{j \in \mathcal{M}(i) \cup i} I_{ij} \hat{S}_{ij} p_j^n, 
\end{align*}
\]
\[
\text{where } I_{ij} \triangleq I_{C(i)} I_{C(j)}^T. \text{ This requires local communication among neighboring agents. We have now decomposed all CG updates and can summarize DCG in Algorithm 2. The initialization of the residual and step direction must satisfy } r^0 = p^0 = s - S\lambda^0_C \text{ where } \lambda^0_C \text{ can be chosen. This can be decomposed into }
\end{align*}
\]
\[
\begin{align*}
r_i^0 &= p_i^0 = \sum_{j \in \mathcal{M}(i) \cup i} I_{ij} \lambda^0_{\mathcal{C}, j} - \sum_{j \in \mathcal{M}(i) \cup i} I_{ij} \lambda^0_{\mathcal{C}, j} \text{ and hence requires neighbor-to-neighbor communication.}
\end{align*}
\]

### 3.4 Communication analysis

We rely on three measures to analyse the communication requirements of ASM/DCG: the number of float that are sent to and from a central coordinator (global floats), the number of booleans that are sent to and from a central coordinator (global booleans), and the number of floats that are sent on a neighbor-to-neighbor basis (local floats).

In each DCG iteration, each agent sends $\eta_i^n$ and $\sigma_i^n$ to a central coordinator. The coordinator then computes the sums $\sigma^n$ and $\eta^n$ and returns them to each agent. This totals in 4M global floats per DCG iteration. In addition, each agent sends a convergence flag to the coordinator and the coordinator returns a global convergence flag to each agent, which gives 2M global booleans per DCG iteration. In step 4. of the DCG iteration, agent $j$ sends those elements of $\hat{S}_{ij} p_j^n$ to neighbor $i$, that correspond to non-zero elements in $I_{ij}$. Hence, agents $i$ and $j$ exchange one float per coupled variable. This results in $2n_c$ local floats per DCG iteration. Within each ASM iteration, either the smallest Lagrange multiplier or the largest feasible step size have to be determined and convergence is checked. This requires $2M$ global floats and $2M$ global booleans.

| Algorithm | global fl. | global bo. | local fl. |
|-----------|------------|------------|-----------|
| ASM       | 4M         | 2M         | 0         |
| DCG       | 4M         | 2M         | 2n_c      |
| ADMM      | 0          | 2M         | 2n_c      |

Table 1. Communication per iteration.
4. NUMERICAL RESULTS

We compare the presented ASM with an ADMM implementation (cf. Appendix A) on a chain of masses system (Conte et al., 2016). The baseline example consists of 10 masses with $m = 1$ kg that are coupled by springs with stiffness $k = 3$ N/m and dampers with coefficient $d = 3$ Ns/m. Each mass is actuated by a force $-1$ N $\leq u_i \leq 1$ N. We do not consider constraints on the states $x_i(t) = [y_i(t) \; v_i(t)]^T$, where $y_i(t)$ and $v_i(t)$ denote the position and velocity of mass $i$. The equations of motion are discretized using the Euler forward discretization with step size $T = 0.2$ s. We choose the parameters of the MPC controller as $N = 12$, $Q_i = \text{diag}([10,10])$, $P_i = 0$ and $R = 1$. The resulting OCP (4) has the dimension $n_z = 812$ with 260 equality constraints for the dynamics, 240 input constraints and 432 coupling constraints. We choose the following tolerances for the stopping criteria: $\|v_i\|_\infty < 10^{-7}$ for DCG and $\|\Delta z_i\|_\infty < 10^{-6}$ for ASM. We choose $\lambda_{C,i} = 0$ to initialize DCG and warm-start the ASM with the optimal active set from the previous MPC iteration. For ADMM, the stopping criteria $\|C_i^T(z_i - \bar{z}_i)\|_\infty \leq \varepsilon_r \min\{\max\{\|C_i^T z_i\|_\infty, \|C_i^T \bar{z}_i\|_\infty\}, 1\}$ and $\|\rho C_i(z_i^t - z_i)\|_\infty \leq \varepsilon_d \min\{\|\lambda_{C,i}\|_\infty, 1\}$ are used. Two different tolerance levels are chosen:

- ADMM1: $\varepsilon_r = 10^{-6}$ and $\varepsilon_d = 10^{-3}$
- ADMM2: $\varepsilon_r = 10^{-4}$ and $\varepsilon_d = 10^{-2}$

Five case studies that each simulate the closed-loop behavior for 25 MPC iterations starting at 30 random initial positions are conducted. For the baseline case study, the initial position and velocity for each agent are chosen from a uniform distribution in the intervals $-1 \text{m} \leq y_{i,0} \leq 1 \text{m}$ and $-0.5 \text{m/s} \leq v_{i,0} \leq 0.5 \text{m/s}$. Figure 4 shows simulation results with the closed-loop trajectories of the agents and the number of floats per sample interval that is communicated locally from neighbor-to-neighbor in the network. The output and input trajectories of the fifth agent are displayed in black for better visualization and the trajectories of the remaining agents are shown in grey.

This paper has proposed a novel distributed active set method that can be used for the distributed MPC of linear systems. The method combines a usual active set approach.

Table 3. Iterations (baseline case).

|        | mean | max | mean | max | mean | max |
|--------|------|-----|------|-----|------|-----|
| DCG    | 30   | 98  | 27   | 97  | 1    | 1   |
| ASM    | 1    | 1   | -    | -   | 1    | 1   |
| ADMM1  | 117  | 185 | -    | -   | -    | -   |
| ADMM2  | 41   | 78  | -    | -   | -    | -   |

Table 4. Iterations and communication footprint (modified problem parameters).

|        | mean | max | mean | max | mean | max |
|--------|------|-----|------|-----|------|-----|
| ADM/DCG| 37   | 283 | 29   | 108 | 34k  | 251k|
| ADMM1  | 133  | 303 | -    | -   | 109k | 262k|
| ADMM2  | 154  | 226 | -    | -   | 56k  | 82k |
| 5 Masses|      |     |      |     |      |     |
| ADM/DCG| 26   | 68  | 25   | 67  | 11k  | 28k |
| ADMM1  | 106  | 171 | -    | -   | 41k  | 66k |
| ADMM2  | 127  | 237 | -    | -   | 231k | 433k|

The results show that ASM/DCG requires less iterations than ADMM1 for all analyzed scenarios.

5. SUMMARY AND OUTLOOK

Fig. 1. Closed-loop trajectories and locally communicated floats per sampling instance.

Table 2. Communication footprint (baseline case).

|        | global floats | global bool. | local floats |
|--------|---------------|--------------|-------------|
| ASM/DCG| 1.3k          | 3.9k         | 0.7k  2.1k  | 27k  88k  |
| ADMM1  | 0             | 0            | 2.4k  3.7k  | 102k 160k|
| ADMM2  | 0             | 0            | 0.8k  1.6k  | 35k  68k  |

Future work may improve the initialization procedure (cf. Remark 2). Four additional case studies were conducted in addition to the baseline case. A single parameter compared to the baseline case has been changed for each of the additional cases and the results are given in Table 4. The iterations reported for ASM/DCG are inner (i.e. DCG) iterations. The results show that ASM/DCG requires less iterations than ADMM1 for all analyzed scenarios.
REFERENCES

Boccia, A., Grüne, L., and Worthmann, K. (2014). Stability and feasibility of state constrained MPC without stabilizing terminal constraints. Systems & Control Letters, 72, 14–21.

Boyd, S., Parikh, N., and Chu, E. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. Now Publishers Inc.

Conte, C., Jones, C.N., Morari, M., and Zeilinger, M.N. (2016). Distributed synthesis and stability of cooperative distributed model predictive control for linear systems. Automatica, 69, 117–135.

Dang, D., Diehl, M., Keviczky, T., and De Schutter, B. (2017). A Jacobi decomposition algorithm for distributed convex optimization in distributed model predictive control. IFAC-PapersOnLine, 50(1), 4905–4911.

Doan, M.D., Keviczky, T., and De Schutter, B. (2011). A dual decomposition-based optimization method with guaranteed primal feasibility for hierarchical MPC problems. IFAC Proceedings Volumes, 44(1), 392–397.

Engelmann, A. and Faulwasser, T. (2021). Decentralized conjugate gradients with finite-step convergence. Submitted, arXiv:2102.12311.

Ferreau, H.J., Bock, H.G., and Diehl, M. (2008). An online active set strategy to overcome the limitations of explicit MPC. International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal, 18(8), 870–895.

Giselsson, P. and Rantzer, A. (2014). On feasibility, stability and guarantee stability because the scheme ensures primal feasibility. This aspect and improved initialization procedures may also reduce the communication requirement further.

Algorithm 3 ADMM

Initialization: $\lambda_0$ and $\bar{z}_0$.

Repeat until convergence:

1. $z_{i}^{n+1} = \underset{z_i}{\text{argmin}} z_i^T H_i z_i + \lambda_{C,i}^n C_i^T z_i + \frac{\rho}{2} \|C_i(z_i - z_i^n)\|^2_2$ subject to $C_i^T z_i = b_i^n$, $C_i^T z_i \leq b_i^n$

2. Receive $[v_{ij}]_{j \in M^{out}}$.

3. Compute $x_i = \sum_{j \in M^{out}} (x_i + v_{ij}) / |M^{out}|$.

4. Receive $[\bar{v}]_{j \in M^{out}}$ and form $\bar{z}_i$.

5. $\lambda_{C,i}^{n+1} = \lambda_{C,i}^n + \rho C_i (z_i^{n+1} - z_i^n)$

We briefly comment on the ADMM variant that we use here for the comparison with ASM/DCG. We refer the reader to (Boyd et al., 2011) for an extensive overview of ADMM and to (Rostami et al., 2017) for the application of ADMM to distributed MPC. To apply ADMM, we introduce the trajectory average $\bar{x}_i \in \mathbb{R}^{N_{ni}}$ and the decision variable $\bar{z}_i^T = [x_i^T, u_i^T, [x_j^T]_{j \in M^{in}}] \in \mathbb{R}^{n_{si}}$, which satisfy the coupling constraints (4d) by design. ADMM alternately updates $\bar{z}_i$, which satisfies (4b) and (4c), and $\bar{z}_i$ until $\bar{z}_i$ satisfies (4d) to a chosen accuracy. The $\bar{z}_i$ update requires local communication and convergence flags are sent to a global coordinator as for ASM/DCG.