

ORIGINAL RESEARCH

Open access

Optimization Algorithms for Distributed Model Predictive Control: Benchmarking Dual Decomposition Approaches

Patrick O'Connor^{1*}, Sean Murphy¹

Abstract

This paper introduces a benchmark comparison of distributed optimization algorithms based on dual decomposition for model predictive control problems with constraint coupling. Such problems can be viewed as multiple subsystems linked through constraints on shared, limited resources. In a dual decomposition approach, the allocation and use of these resources are coordinated by repeatedly calculating their associated prices and distributing those values to the participating subsystems. After a short overview of model predictive control, various architectures and communication structures suitable for distributed implementations are outlined. The centralized control problem is then decomposed into several subproblems by introducing dual variables. A range of distributed optimization methods is examined, including the recently developed quasi-Newton dual ascent algorithm. Additionally, an epigraph reformulation of the bundle, combined with a line search technique, is proposed for the quasi-Newton dual ascent method; these enhancements improve its numerical stability and accelerate convergence relative to the earlier trust-region approach. Lastly, the quasi-Newton dual ascent algorithm is benchmarked against the subgradient method, the bundle trust method, and the alternating direction method of multipliers across a broad set of test problems. All benchmark instances are openly accessible on GitHub.

Keywords Distributed model predictive control, Dual decomposition, Subgradient method, ADMM, Bundle method, Quasi-Newton

*Correspondence:

Patrick O'Connor

patrick.oconnor@tcd.ie

¹ Department of Power Systems Engineering, Faculty of Engineering, Trinity College Dublin, Dublin, Ireland

Introduction

The growing complexity of industrial systems has driven the adoption of sophisticated control techniques. For many years, decentralized control structures relying on numerous independent PID loops have been widely implemented in practice [1]. These approaches typically presume minimal or no interaction among the control loops within the overall system. Nevertheless, the pursuit of greater efficiency in large-scale systems has resulted in stronger interconnections among components, thereby weakening the effectiveness of purely decentralized strategies. Moreover, contemporary systems—and their corresponding control systems—are required to meet a variety of operational and safety constraints while simultaneously optimizing multiple performance objectives. Model predictive control (MPC) is an optimization-based strategy that uses a plant model to forecast future system behavior from the current state and prospective control actions. Formulating the control task as an optimization problem enables direct minimization or maximization of performance criteria while explicitly enforcing all relevant constraints. Today, problems involving hundreds or even thousands of variables can be solved efficiently, largely thanks to major advances in the performance of optimization solvers. Despite these developments, certain scenarios still call for more decentralized control strategies. A prominent example involves large-scale systems composed of multiple autonomous subsystems, each having its own goals and limitations. Although

these subsystems can make independent decisions, they may remain coupled—for instance, through shared energy or material flows. In such cases, global constraints must be upheld even when local control problems are addressed in a distributed fashion. One key motivation for this architecture is the need to preserve confidentiality among subsystems, particularly when they belong to separate organizations unwilling or unable to share details of their internal dynamics and local constraints. Distributed optimization solves the problem of coordinating individual subsystem controllers, leading to distributed model predictive control (DMPC) problems.

A class of distributed optimization algorithms particularly well-suited for coordinating resource-coupled DMPC problems is dual-decomposition-based methods. These approaches are frequently called price-based coordination schemes, in which the introduced dual variables are interpreted as prices associated with shared, limited resources; these prices guide the subsystems toward solutions that satisfy the overall system-wide constraints.

The present paper makes two main contributions: First, the recently introduced quasi-Newton dual ascent algorithm [2, 3] is enhanced by reformulating its cutting planes as linear constraints through an epigraph representation. In addition, a line search procedure with constraint handling is incorporated, which accelerates convergence, especially near the optimal solution. Second, a comprehensive set of DMPC benchmark problem instances has been created and released for public use on GitHub. Although several prior studies have benchmarked DMPC algorithms, many are limited to single instances [4–6] or do not provide open access to the benchmark problems [7].

Preliminaries

In this work, vectors are denoted by bold lowercase letters ($\mathbf{x}$) and matrices by bold uppercase letters ($\mathbf{A}$). In model predictive control, time is typically discretized into uniform sampling intervals of length T_s , resulting in discrete time points $t_k = t_0 + k \cdot T_s$, where t_0 denotes the current time. The symbols $\mathbf{x}^k$ and $\mathbf{u}^k$ denote the plant states and control inputs, respectively, at time t_k . The value of a variable $\mathbf{z}$ at iteration i of a dual decomposition-based algorithm is indicated by (i) . The Euclidean norm is written as $\|\cdot\|_2$.

MPC methods generally rely on a discrete-time model to forecast the future evolution of the plant states $\mathbf{x} \in \mathbb{R}^{n_x}$ based on the current states and control inputs $\mathbf{u} \in \mathbb{R}^{n_u}$. A standard MPC optimization problem is formulated as shown in Eq. 1.

$$\min_{\mathbf{x}^{0:N_p}, \mathbf{u}^{0:N_p-1}} J^f(\mathbf{x}^{N_p}) + \sum_{k=0}^{N_p-1} J(\mathbf{x}^k, \mathbf{u}^k), \quad (1a)$$

$$\text{s. t. } \mathbf{x}^{k+1} = \mathbf{A}\mathbf{x}^k + \mathbf{B}\mathbf{u}^k, \quad k = 0, \dots, N_p - 1, \quad (1b)$$

$$\mathbf{x}^0 = \tilde{\mathbf{x}}(t_0), \quad (1c)$$

$$\mathbf{x}^k \in \mathcal{X} \subset \mathbb{R}^{n_x}, \quad k = 0, \dots, N_p, \quad (1d)$$

$$\mathbf{u}^k \in \mathcal{U} \subset \mathbb{R}^{n_u}, \quad k = 0, \dots, N_p - 1. \quad (1e)$$

Constraint (Eq. 1b) describes a discrete-time dynamic model that predicts the plant state $\mathbf{x}_{k+1}$ at the subsequent sampling instant from the current state $\mathbf{x}^k$ and applied input $\mathbf{u}^k$, with system matrices $\mathbf{A} \in \mathbb{R}^{n_x \times n_x}$ and $\mathbf{B} \in \mathbb{R}^{n_x \times n_u}$. Predictions of future states and inputs are generated over a prediction horizon of length N_p . It should be noted that control inputs need not be optimized over the full prediction horizon, particularly when computational cost is a concern. In such cases, inputs may be optimized only over a shorter control horizon $N_c < N_p$, with the final computed input held constant

for the remaining steps (i.e., $\mathbf{u}_{N_c} = \mathbf{u}_{N_c+1} = \dots = \mathbf{u}_{N_p-1}$). For this study, it is assumed that $N_p = N_c$. Constraint (Eq. 1c) specifies the initial conditions of the plant, where $\hat{\mathbf{x}}(t_0)$ represents the measured states at the current time t_0 . In situations where not all states are directly measurable, state estimation techniques may be employed to reconstruct them from available outputs; however, state estimation lies beyond the scope of this paper. An overview of state estimation techniques in distributed frameworks is provided by Necoara et al. [8].

One of the primary strengths of MPC over alternative control strategies is its ability to explicitly incorporate constraints on both states and inputs directly into the optimization formulation [9]. These constraints are collectively expressed in (Eq. 1d) for the states and in (Eq. 1e) for the inputs.

At every sampling instant, the controller calculates a sequence of future inputs across the prediction horizon N_p by solving the open-loop optimization problem defined in (Eq. 1).

A frequent goal in model predictive control is to make the system follow a specified reference signal throughout the horizon while keeping control actions as moderate as possible, which leads to the cost function.

$$J(\mathbf{x}^k, \mathbf{u}^k) = (\mathbf{x}^k - \mathbf{x}^{\text{ref},k})^T \mathbf{H}_x (\mathbf{x}^k - \mathbf{x}^{\text{ref},k}) + \mathbf{u}^{k,T} \mathbf{H}_u \mathbf{u}^k, \quad (2)$$

Where $\mathbf{H}_x \in \mathbb{R}^{n_x \times n_x}$ and $\mathbf{H}_u \in \mathbb{R}^{n_u \times n_u}$ are symmetric positive (semi-)definite weighting matrices. The expression in Eq. 2 is usually called the stage cost, in contrast to the terminal cost ($\mathbf{x}_{N_p}$). One standard form of the terminal penalty penalizes the squared distance between the states at the final time step and their desired reference values, namely.

$$J^f(\mathbf{x}^{N_p}) = (\mathbf{x}^{N_p} - \mathbf{x}^{\text{ref},N_p})^T \mathbf{H}_x^f (\mathbf{x}^{N_p} - \mathbf{x}^{\text{ref},N_p}). \quad (3)$$

Careful tuning of the terminal cost term is essential to ensure satisfactory closed-loop behavior and stability [9]. Discussion of these design issues, however, is beyond the objectives of the current study. A key strength of MPC lies in its ability to handle more advanced control objectives than mere setpoint tracking by embedding them directly into the optimization setup [10]. For example, one can optimize process economics directly, leading to the field known as economic model predictive control (EMPC). For a thorough treatment of EMPC, the reader is referred to the review by Ellis et al. [11].

After the open-loop optimization has been completed, only the initial control move $\mathbf{u}_0$ is actually implemented. Then, at the next sampling time, the plant's new state is measured (or estimated), and the optimization cycle restarts.

Distributed Model Predictive Control

Because MPC requires an optimization problem to be solved repeatedly at each discrete time step, difficulties arise when that problem becomes excessively large—either exceeding available computation time or preventing critical data from being gathered at a single central location due to a multi-subsystem architecture. One effective way to overcome these limitations is to break the global MPC task into smaller pieces and solve them in a distributed way.

Research on distributed MPC (DMPC) has primarily focused on two distinct problem classes.

The first class involves a collection of N_s dynamically interconnected subsystems indexed by the set $I = \{1, \dots, N_s\}$. The corresponding optimization problems take the form

$$\min_{\mathbf{x}^{0:N_p}, \mathbf{u}^{0:N_p-1}} \sum_{i \in \mathcal{I}} \left[J_i^f(\mathbf{x}_i^{N_p}) + \sum_{k=0}^{N_p-1} J_i(\mathbf{x}_i^k, \mathbf{u}_i^k) \right], \quad (4a)$$

$$\text{s. t. } \mathbf{x}_i^{k+1} = \sum_{j \in \mathcal{I}} \mathbf{A}_{ij} \mathbf{x}_j^k + \mathbf{B}_{ij} \mathbf{u}_j^k, \quad \forall i \in \mathcal{I}, k = 0, \dots, N_p - 1, \quad (4b)$$

$$\mathbf{x}_i^0 = \tilde{\mathbf{x}}(t_0), \quad \forall i \in \mathcal{I}, \quad (4c)$$

$$\mathbf{x}_i^k \in \mathcal{X}_i \subset \mathbb{R}^{n_{x_i}}, \quad \forall i \in \mathcal{I}, k = 0, \dots, N_p, \quad (4d)$$

$$\mathbf{u}_i^k \in \mathcal{U}_i \subset \mathbb{R}^{n_{u_i}}, \quad \forall i \in \mathcal{I}, k = 0, \dots, N_p - 1, \quad (4e)$$

With $\mathbf{x}_i$ and $\mathbf{u}_i$ standing for the state and input vectors of the i -th subsystem. The complete vectors of all subsystem states and inputs are assembled as $\mathbf{x} = [\mathbf{x}_1^\top \dots \mathbf{x}_{|\mathcal{I}|}^\top]^\top$ and $\mathbf{u} = [\mathbf{u}_1^\top \dots \mathbf{u}_{|\mathcal{I}|}^\top]^\top$. Each subsystem is equipped with its private terminal cost $J_i(\mathbf{x}_i^{N_p})$ and per-step running cost $J_i(\mathbf{x}_i^k, \mathbf{u}_i^k)$. The matrices $\mathbf{A}_{ij} \in \mathbb{R}^{n_{x_i} \times n_{x_j}}$ and $\mathbf{B}_{ij} \in \mathbb{R}^{n_{x_i} \times n_{u_j}}$ quantify how the states and inputs of subsystem j affect the evolution of subsystem i .

The second class features subsystems that interact solely via inequality constraints.

$$\min_{\mathbf{x}^{0:N_p}, \mathbf{u}^{0:N_p-1}} \sum_{i \in \mathcal{I}} \left[J_i^f(\mathbf{x}_i^{N_p}) + \sum_{k=0}^{N_p-1} J_i(\mathbf{x}_i^k, \mathbf{u}_i^k) \right], \quad (5a)$$

$$\text{s. t. } \mathbf{x}_i^{k+1} = \mathbf{A}_i \mathbf{x}_i^k + \mathbf{B}_i \mathbf{u}_i^k, \quad \forall i \in \mathcal{I}, k = 0, \dots, N_p - 1, \quad (5b)$$

$$\mathbf{x}_i^0 = \tilde{\mathbf{x}}(t_0), \quad \forall i \in \mathcal{I}, \quad (5c)$$

$$\mathbf{x}_i^k \in \mathcal{X}_i \subset \mathbb{R}^{n_{x_i}}, \quad \forall i \in \mathcal{I}, k = 0, \dots, N_p, \quad (5d)$$

$$\mathbf{u}_i^k \in \mathcal{U}_i \subset \mathbb{R}^{n_{u_i}}, \quad \forall i \in \mathcal{I}, k = 0, \dots, N_p - 1, \quad (5e)$$

$$\sum_{i \in \mathcal{I}} \mathbf{R}_i \mathbf{u}_i^k \leq \mathbf{r}_{\max}^k, \quad k = 0, \dots, N_p - 1. \quad (5f)$$

In this setting, the individual subsystem dynamics (Eq. 5b) are completely decoupled. The only source of coupling is the collection of global resource constraints (Eq. 5f), which enforce limits on shared scarce resources. Each matrix $\mathbf{R}_i \in \mathbb{R}^{n_r \times n_{u_i}}$ converts the inputs of subsystem i into the net consumption or generation of the various resources, while the vector $\mathbf{r} \in \mathbb{R}^{n_r}$ defines the upper bound on resource availability. Within these constraints, a positive entry indicates resource use, and a negative entry indicates resource supply.

The focus of the present article is exclusively on the second category—DMPC problems of type (5)—which lend themselves naturally to resolution through dual decomposition techniques in a distributed optimization setting. For completeness, it is worth noting that problems in the first category (4) can also be tackled using dual decomposition methods.

DMPC architectures

Besides the specific form of interaction among the subsystems, the selected distributed model predictive control (DMPC) strategy is largely determined by the permitted communication constraints. Several DMPC schemes have been examined

in detail by authors such as Scattolini [12] and Christofides et al. [13]; several representative examples appear in **Figure 1**.

To keep the presentation straightforward, we temporarily neglect any influence that one subsystem's control output exerts on others, which corresponds to setting $\mathbf{B}_{ij} = \mathbf{0}$ whenever $i \neq j$ (**Eq. 4b**).

The layout in **Figure 1(a)** represents the case in which DMPC controllers completely disregard interconnections among subsystems and therefore do not exchange any data. This arrangement is commonly referred to as decentralized MPC. Here, any coupling effects are treated as unknown disturbances that each local controller must reject on its own. Strong inter-subsystem coupling frequently leads to degraded closed-loop behavior under this scheme. In addition, purely decentralized DMPC cannot reliably enforce globally defined constraints because there is no mechanism for coordinating constraint satisfaction across subsystems. Further discussion of decentralized MPC approaches can be found in Christofides et al. [13].

Whenever the subsystems are coupled in a way that requires explicit handling inside the local optimizers, at least limited information sharing is essential. **Figure 1(b)** illustrates the scenario that allows unrestricted, direct communication links among all DMPC controllers. Through these links, the controllers can transmit predicted state and input trajectories, cost function values, active constraints, and other relevant quantities.

Algorithms that adopt this fully connected communication pattern are commonly distinguished by the number of information-exchange steps executed during each MPC time step—i.e., the count of negotiation cycles before the computed inputs are actually implemented. Both single-round and multi-round communication strategies are systematically reviewed by Camponogara et al. [14].

The main practical consideration when choosing between one or several communication rounds per sampling interval is the CPU time needed to solve a single local DMPC problem. Multi-round methods—often described as negotiation among agents [15]—generally yield improved overall performance and, in many cases, converge toward the globally optimal solution that a centralized MPC would produce. The drawback is a significantly higher computational load, since each subsystem must resolve its optimization multiple times within a single control cycle. For applications that demand very short sampling periods, repeated negotiations become infeasible. A typical example is collision-free control of robotic arms, where the forecasted state sequence is usually forwarded to adjacent controllers [16].

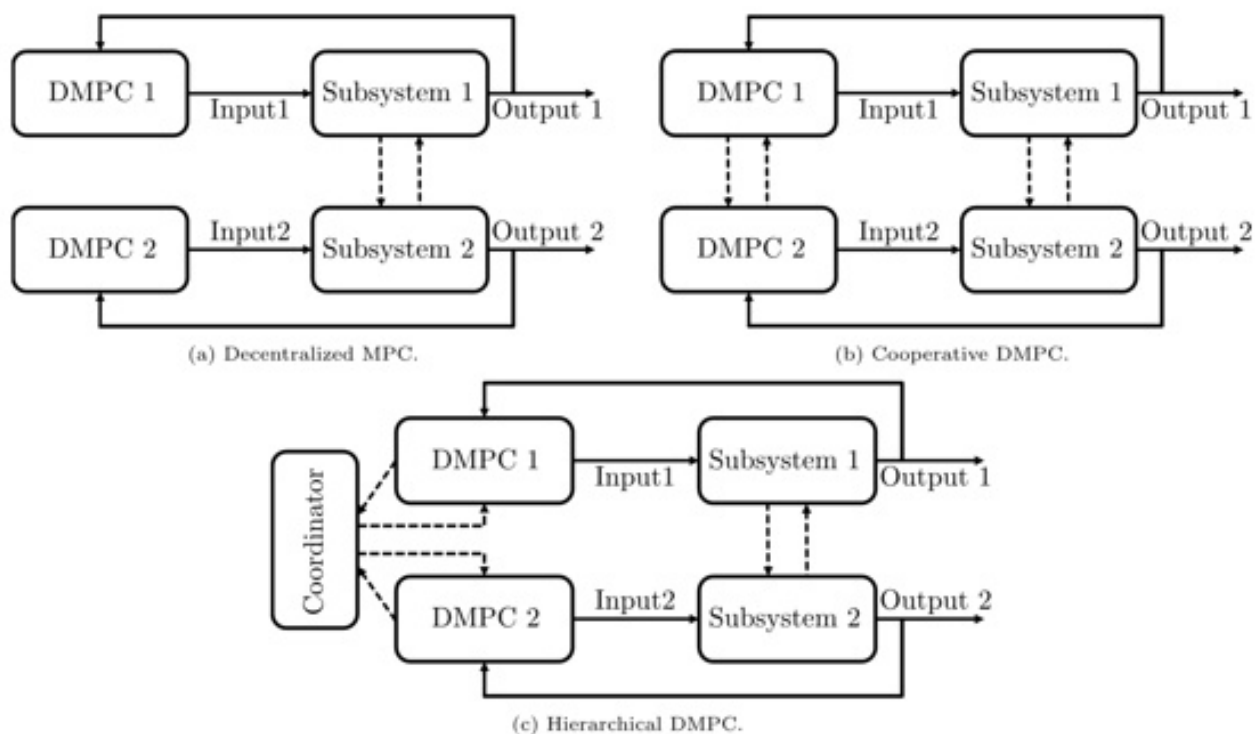


Figure 1. Different distributed model predictive control structures for two subsystems with two controllers—source: Adapted from Scattolini [12].

In contrast, the configuration shown in **Figure 1(c)** forbids any peer-to-peer communication between the local DMPC controllers. Instead, all local optimizers report to and receive directives from a single coordinating entity, resulting in a hierarchical architecture. The coordinator gathers the local solutions, computes appropriate price (or Lagrange multiplier) updates, and broadcasts these signals back to the subsystems. Dual decomposition techniques rely precisely on this kind of information flow, with the coordinator transmitting updated dual variables at every iteration.

Hierarchical DMPC proves especially advantageous for systems whose constraints couple multiple subsystems yet where direct controller-to-controller data sharing is restricted or prohibited—for reasons such as data privacy or organizational boundaries. One important characteristic of hierarchical schemes is that they invariably involve several rounds of communication between the local units and the coordinator during each sampling interval. For this reason, the method is realistic only when the plant exhibits slow dynamics and the sampling time is long enough to accommodate repeated solution of the optimization subproblems until convergence or feasibility is reached. Representative application domains include large-scale chemical processes and electric power networks.

The subsequent section briefly surveys selected recent contributions to DMPC, focusing primarily on cooperative and hierarchical formulations.

DMPC literature review

Research into distributed control strategies for large-scale processes has remained vibrant since the 1970s [17]. In-depth surveys of distributed model predictive control (DMPC) appear in works by, among others, Scattolini [12], Necoara et al. [8], Christofides et al. [13], Negenborn and Maestre [18], and Müller and Allgöwer [19].

Stewart et al. introduce a cooperative DMPC technique that functions as an approximate centralized MPC and thereby establishes closed-loop stability [20]. Maestre et al. describe a cooperative DMPC method tailored to two agents involving

two information-exchange steps per control interval [21]. The same group also develops a cooperative DMPC framework for multiple agents, in which, during each communication cycle, a randomly chosen subset of agents proposes control moves that neighboring agents adopt only if those moves improve the global performance index [15]. Zheng *et al.* outline a DMPC procedure designed for networks of interconnected systems featuring direct controller-to-controller links [22].

Approaches based on dual decomposition have addressed systems with coupling in both the dynamics and the constraints. Giselsson and Rantzer apply dual decomposition to dynamically coupled systems and introduce a termination condition that ensures limited suboptimality together with asymptotic stability [23]. In related work, Giselsson *et al.* propose an accelerated gradient technique for dual decomposition DMPC that incorporates L_1 -norm regularization in the cost [24]. Giselsson and Rantzer further describe an early-stopping rule based on adaptive constraint relaxation, enabling faster convergence of dual decomposition iterations in dynamically coupled setups [25]. Köhler *et al.* investigate conditions for recursive feasibility when halting a dual decomposition distributed optimizer ahead of full convergence [26]. Doan *et al.* utilize dual decomposition combined with primal averaging and constraint adjustment to secure convergence in DMPC scenarios involving coupled dynamics and constraints alike [27].

Biegel *et al.* demonstrate the application of MPC to congestion control within an electricity network populated by smart consumers equipped with storage units [28]. This concept is later adapted to a fully distributed DMPC framework [29]. Biegel *et al.* also illustrate the broad applicability of dual decomposition DMPC to systems linked solely through constraints [30]. Pflaum *et al.* compare primal versus dual decomposition DMPC strategies for distributed energy management in smart neighborhoods [31]. In a separate study [32], they assess how well dual decomposition DMPC scales—using a bundle-based solver—when varying the count of subsystems, the quantity of constraints, and the largest bundle dimension.

Razzanelli *et al.* implement DMPC with a subgradient algorithm to orchestrate energy exchanges across a cluster of microgrids [33]. Eser *et al.* adopt the alternating direction method of multipliers (ADMM) for decentralized regulation of building climate and energy systems [34]. Maxeiner and Engell employ ADMM to handle DMPC of semi-batch chemical reactors that share scarce input resources [35]. That effort is broadened in the work by Maxeiner and Engell [5], where the authors conduct a thorough benchmark of the subgradient approach, ADMM, and the augmented Lagrangian alternating direction inexact Newton (ALADIN) method for resource-constrained coupled problems. Houska and Shi offer an instructional overview of applying ALADIN to DMPC tasks [36]. Yfantis *et al.* introduce a quadratic model-based solver for DMPC under shared resource limits and compare its efficiency with the subgradient technique [37].

Conte *et al.* explore numerical properties of dual decomposition DMPC when accelerated gradient or ADMM solvers are used [7]. Their analysis specifically quantifies how factors such as coupling intensity, system stability, starting conditions, network scale, and topology affect runtime performance. Stomberg *et al.* benchmark six different DMPC solvers for dynamically interacting subsystems: the basic subgradient method, an accelerated subgradient variant, ADMM, a distributed active-set procedure, a nearly decentralized interior-point algorithm, and Jacobi-style iterations [6].

Beyond ADMM, DMPC also uses additional proximal splitting methods. Halvgaard *et al.* apply the Douglas-Rachford splitting scheme to coordinate distributed smart energy infrastructures [38]. A comprehensive compilation of DMPC methods, along with practical case studies, is available in the textbook edited by Maestre and Negenborn [39].

Dual Decomposition-Based Distributed Model Predictive Control

The present section focuses on DMPC grounded in dual decomposition principles. By decomposing an overarching constraint-coupled MPC formulation through the introduction of dual (Lagrange) multipliers, one obtains a dual optimization problem amenable to a fully distributed solution. Various iterative solvers for this dual problem are outlined,

among them the quasi-Newton dual ascent procedure. For completeness, the section also illustrates how dual decomposition extends naturally to DMPC problems featuring dynamic coupling between subsystems.

Dual decomposition of constraint-coupled DMPC problems

This section explains how dual decomposition is applied to separate the overall constraint-coupled model predictive control (MPC) problem into independent subproblems. By associating dual variables $\lambda k \in \mathbb{R}^{nr}$ with each sampling instant k , the Lagrangian of the centralized problem can be expressed as follows.

$$\begin{aligned} \mathcal{L}(\mathbf{x}^{0:N_p}, \mathbf{u}^{0:N_p-1}, \boldsymbol{\lambda}^{0:N_p-1}) \\ = \sum_{i \in \mathcal{S}} \underbrace{\left[J_i^f(\mathbf{x}_i^{N_p}) + \sum_{k=0}^{N_p-1} [J_i(\mathbf{x}_i^k, \mathbf{u}_i^k) + \boldsymbol{\lambda}^{k,T} \mathbf{R}_i \mathbf{u}_i^k] \right]}_{= \mathcal{L}_i(\mathbf{x}_i^{0:N_p}, \mathbf{u}_i^{0:N_p-1}, \boldsymbol{\lambda}^{0:N_p-1})} - \sum_{k=0}^{N_p-1} \boldsymbol{\lambda}^{k,T} \mathbf{r}_{\max}^k \end{aligned} \quad (6)$$

The full Lagrangian then splits additively into a sum of local Lagrangian terms, one for each subsystem.

$$\begin{aligned} \mathcal{L}(\mathbf{x}^{0:N_p}, \mathbf{u}^{0:N_p-1}, \boldsymbol{\lambda}^{0:N_p-1}) &= \sum_{i \in \mathcal{S}} \mathcal{L}_i(\mathbf{x}_i^{0:N_p}, \mathbf{u}_i^{0:N_p-1}, \boldsymbol{\lambda}^{0:N_p-1}) \\ &- \sum_{k=0}^{N_p-1} \boldsymbol{\lambda}^{k,T} \mathbf{r}_{\max}^k \end{aligned} \quad (7)$$

As a result, the dual function

$$\begin{aligned} d(\boldsymbol{\lambda}^{0:N_p-1}) &= \min_{\mathbf{x}^{0:N_p}, \mathbf{u}^{0:N_p-1}} \mathcal{L}(\mathbf{x}^{0:N_p}, \mathbf{u}^{0:N_p-1}, \boldsymbol{\lambda}^{0:N_p-1}), \\ \text{s.t. (5b)–(5e)} \end{aligned} \quad (8)$$

Becomes computable in a fully distributed fashion by independently solving the local DMPC optimization problems

for every subsystem i .

$$\begin{aligned} \min_{\substack{0:N_p \\ \mathbf{x}_i, \mathbf{u}_i}} \mathcal{L}_i(\mathbf{x}_i^{0:N_p}, \mathbf{u}_i^{0:N_p-1}, \boldsymbol{\lambda}^{0:N_p-1}), \\ \text{s.t. } \mathbf{x}_i^{k+1} = \mathbf{A}_i \mathbf{x}_i^k + \mathbf{B}_i \mathbf{u}_i^k, \end{aligned} \quad (9a)$$

$$k = 0, \dots, N_p - 1, \quad (9b)$$

$$\mathbf{x}_i^0 = \tilde{\mathbf{x}}(t_0), \quad (9c)$$

$$\mathbf{x}_i^k \in \mathcal{X}_i \subset \mathbb{R}^{n_{x_i}}, \quad k = 0, \dots, N_p, \quad (9d)$$

$$\mathbf{u}_i^k \in \mathcal{U}_i \subset \mathbb{R}^{n_{u_i}}, \quad k = 0, \dots, N_p - 1 \quad (9e)$$

The core idea of dual decomposition-based optimization is to tackle the dual problem

$$\max_{\lambda^{0:N_p-1}} d(\lambda^{0:N_p-1}), \quad (10a)$$

$$\text{s.t. } \lambda^k \geq \mathbf{0}, \quad k = 0, \dots, N_p - 1, \quad (10b)$$

This naturally lends itself to parallel and distributed implementation, thanks to the dual function's separability. The equality constraint (Eq. 10b) arises directly from the Karush–Kuhn–Tucker (KKT) stationarity conditions applied to the inequality-coupled constraints [40].

Several key theoretical features characterize both the dual function and the associated dual optimization problem. First, the optimal value attained by the dual problem (10) invariably serves as a lower bound to the optimal value of the original (primal) problem (5)—a relation known as weak duality. When the primal problem is convex and satisfies appropriate constraint qualifications, the primal and dual optimal values coincide exactly; this is termed strong duality [40]. Moreover, once the optimal dual variables are known, solving the local subproblems (9) at those dual values yields a globally optimal solution to the primal problem.

Second, irrespective of the convexity properties of the primal problem, the dual function is always concave. That said, it is generally nonsmooth: the gradient with respect to the dual variables may fail to exist at certain points [3]. Consequently, the dual problem (10) constitutes a nonsmooth convex optimization task. The next section surveys numerical methods specifically designed to solve such nonsmooth dual problems.

Dual decomposition-based algorithms

The current section describes the distributed optimization techniques based on dual decomposition that form the backbone of the methods presented in this work. It begins with a short overview of the classical subgradient method, followed by the bundle trust-region approach and the alternating direction method of multipliers (ADMM). Particular attention is then devoted to the more recent quasi-Newton dual ascent algorithm introduced by Yfantis *et al.* [2, 3]. Compared with the version detailed by Yfantis *et al.* [3], the algorithm described here incorporates two important enhancements: a reformulation of the inner update subproblem using an epigraph representation and the addition of a line-search procedure applied in the neighborhood of the solution.

Subgradient method

The most straightforward technique for addressing the nonsmooth dual optimization problem is the subgradient method [41]. This approach extends the notion of a gradient to functions that are not differentiable everywhere. For convenience, we represent the complete vector of dual variables across the entire prediction horizon as a single column vector.

$$\lambda_{N_p} = [\lambda^{0,T}, \dots, \lambda^{N_p-1,T}]^T \in \mathbb{R}^{n_{\lambda}(N_p-1)} \quad (11)$$

A subgradient of the dual function (Eq. 8) at iteration t is obtained by evaluating the coupling constraints on the shared limited resources (Eq. 5f), yielding:

$$\mathbf{g}(\lambda^{0:N_p-1,(t)}) = \begin{pmatrix} \sum_{i \in \mathcal{J}} \mathbf{R}_i \mathbf{u}_i^{0,(t+1)} - \mathbf{r}_{\max}^0 \\ \vdots \\ \sum_{i \in \mathcal{J}} \mathbf{R}_i \mathbf{u}_i^{N_p-1,(t+1)} - \mathbf{r}_{\max}^{N_p-1} \end{pmatrix}, \quad (12)$$

Where $\mathbf{u}^k, (t+1)$ refers to the control sequences determined by each subsystem through solution of the local problem (9) during iteration t of the distributed procedure.

In the subgradient method, the dual variables are then adjusted at every step by moving in the direction of this subgradient, according to

$$\lambda_{N_p}^{(t+1)} = \left[\lambda_{N_p}^{(t)} + \alpha^{(t)} \mathbf{g} \left(\lambda_{N_p}^{(t)} \right) \right]^+, \quad (13)$$

Where $[\cdot]^+$ indicates projection onto the nonnegative orthant. The step-size coefficient (t) is typically adjusted dynamically as iterations progress. The procedure stops once both the primal residual norm

$$\left[\mathbf{w}_p^{(t)} \right]_l = \max \left\{ \left[\mathbf{g} \left(\lambda_{N_p}^{(t)} \right) \right]_l, 0 \right\}, \quad l = 1, \dots, n_b \quad (14)$$

And the dual residual norm

$$\mathbf{w}_d^{(t)} = \lambda_{N_p}^{(t+1)} - \lambda_{N_p}^{(t)}, \quad (15)$$

Falls below a user-specified tolerance, indicating that the original primal problem has become feasible and the dual variables have stabilized at a stationary point.

Bundle trust method

Unlike the subgradient method, which relies solely on the most recent subgradient to update the dual variables, bundle methods form a broader, often more effective family of solvers for nonsmooth optimization problems [42]. These algorithms accumulate subgradient information from several past iterations to guide the updates.

$$\mathcal{B}^{(t)} = \left\{ \left(\lambda_{N_p}^{(l)}, \mathbf{g} \left(\lambda_{N_p}^{(l)} \right), d \left(\lambda_{N_p}^{(l)} \right) \right) \in \mathbb{R}^{n_\lambda} \times \mathbb{R}^{n_\lambda} \times \mathbb{R} \mid l = t - \tau + 1, \dots, t \right\} \quad (16)$$

Specifically, in each iteration, the following information is retained:

Where $n_\lambda = n_r \cdot (N_p - 1)$ gives the total dimension of the dual variable vector. The set $\mathcal{B}^{(t)}$ is called the bundle; it comprises the dual variable values, corresponding subgradients, and dual function evaluations collected from earlier steps. To prevent excessive memory usage, only the most recent τ iterations' data are kept in the bundle.

Using this accumulated bundle, a piecewise linear outer approximation of the dual function—known as a cutting-plane model—is constructed and then maximized. Because this model remains nonsmooth, its optimization is reformulated as an epigraph problem, and a search direction for the dual variables is computed under an additional trust-region constraint. This results in the bundle trust method (BTM):

$$\max_{v \in \mathbb{R}, \mathbf{s} \in \mathbb{R}^{n_\lambda}} v, \quad (17a)$$

$$\text{s. t. } \|\mathbf{s}\|_2^2 \leq \alpha^{(t)}, \quad (17b)$$

$$\mathbf{g}^T \left(\boldsymbol{\lambda}_{N_p}^{(l)} \right) \mathbf{s} - \beta^{(l,t)} \geq v, \quad \forall l \in \{t - \tau + 1, \dots, t\}, \quad (17c)$$

$$\boldsymbol{\lambda}_{N_p}^{(t)} + \mathbf{s} \geq \mathbf{0}. \quad (17d)$$

With the linearization error defined as

$$\beta^{(l,t)} = d \left(\boldsymbol{\lambda}_{N_p}^{(t)} \right) - d \left(\boldsymbol{\lambda}_{N_p}^{(l)} \right) - \mathbf{g}^T \left(\boldsymbol{\lambda}_{N_p}^{(l)} \right) \left(\boldsymbol{\lambda}_{N_p}^{(t)} - \boldsymbol{\lambda}_{N_p}^{(l)} \right), \quad \forall l \in \{t - \tau + 1, \dots, t\}. \quad (18)$$

Once a promising direction has been determined, the dual variables are updated via

$$\boldsymbol{\lambda}_{N_p}^{(t+1)} = \boldsymbol{\lambda}_{N_p}^{(t)} + \mathbf{s}^{(t)}. \quad (19)$$

Alternating direction method of multipliers

ADMM introduces quadratic penalty terms that effectively smooth the nonsmooth dual function and promote convexity in the response surface [43]. When multiple subsystems are interconnected solely through shared, scarce resources, the optimal-exchange variant of ADMM is particularly suitable [44].

Under this formulation, auxiliary variables $\mathbf{z}_{ik}$ are introduced for every subsystem and time instant, allowing the construction of an augmented Lagrangian as follows:

$$\begin{aligned} \mathcal{L}_{\rho,i} \left(\mathbf{x}_i^{0:N_p}, \mathbf{u}_i^{0:N_p-1}, \boldsymbol{\lambda}^{0:N_p-1}, \mathbf{z}_i^{0:N_p-1} \right) = \\ J_i^f \left(\mathbf{x}_i^{N_p} \right) + \sum_{k=0}^{N_p-1} \left[J_i \left(\mathbf{x}_i^k, \mathbf{u}_i^k \right) + \boldsymbol{\lambda}^{k,T} \left(\mathbf{R}_i \mathbf{u}_i^k - \mathbf{z}_i^k \right) \right] + \frac{\rho}{2} \sum_{k=0}^{N_p-1} \left\| \mathbf{R}_i \mathbf{u}_i^k - \mathbf{z}_i^k \right\|_2^2. \end{aligned} \quad (20)$$

Theoretical analysis confirms that ADMM converges reliably when the penalty parameter ρ is held constant. Nevertheless, practical implementations frequently achieve faster and more stable progress by adjusting ρ dynamically across iterations. The adaptation scheme implemented here follows the strategy outlined by He et al. [45] and Wang and Liao [46]:

$$\rho^{(t+1)} = \begin{cases} \tau^{\text{incr}} \rho^{(t)}, & \text{if } \left\| \mathbf{w}_p^{(t)} \right\|_2 > \mu \left\| \mathbf{w}_d^{(t)} \right\|_2, \\ \rho^{(t)} / \tau^{\text{decr}}, & \text{if } \left\| \mathbf{w}_d^{(t)} \right\|_2 > \mu \left\| \mathbf{w}_p^{(t)} \right\|_2, \\ \rho^{(t)}, & \text{otherwise.} \end{cases} \quad (21)$$

The constants, τ^{incr} and τ^{decr} (each greater than 1), act as tunable hyperparameters. Once this rule is applied, the algorithm proceeds by sequentially updating the primal variables, dual variables, and auxiliary variables:

$$\mathbf{z}_i^{k,(t+1)} = \mathbf{R}_i \mathbf{u}_i^{(k,t+1)} - \frac{1}{N_e} \sum_{i \in \mathcal{J}} \left(\mathbf{R}_i \mathbf{u}_i^{(t+1)} - \mathbf{r}_{\max}^k \right), \quad k = 0, \dots, N_p - 1, \quad (22a)$$

$$\lambda^{k,(t+1)} = \left[\lambda^{k,(t)} + \rho^{(t)} \frac{1}{N_s} \sum_{i \in \mathcal{S}} \left(\mathbf{R}_i \mathbf{u}_i^{(t+1)} - \mathbf{r}_{\max}^k \right) \right]^+, \quad k = 0, \dots, N_p - 1. \quad (22b)$$

One characteristic advantage of ADMM is that the augmentation term actively encourages the subsystem solutions to satisfy the primal constraints. In problems with inequality coupling, however, the algorithm may reach a point at which the augmented constraints are satisfied while a residual penalty contribution persists. At true optimality, this residual should disappear, so that the final dual variables produce a feasible point even when evaluated with the original (unaugmented) Lagrangian.

To overcome this limitation, Maxeiner [47] recommended replacing the standard auxiliary update step (Eq. 22b) with a dedicated optimization subproblem:

$$\mathbf{z}^{(t+1)} = \arg \min_{\mathbf{z}} \sum_{k=0}^{N_p-1} \left\| \sum_{i \in \mathcal{S}} \mathbf{R}_i \mathbf{u}_i^{k,(t)} - \mathbf{z}^k \right\|_2^2, \quad (23a)$$

$$\text{s. t. } \sum_{i \in \mathcal{S}} \mathbf{z}_i^k \leq \mathbf{r}_{\max}^k, \quad k = 0, \dots, N_p - 1. \quad (23b)$$

The consequences of this change are visualized in Figure 2. Employing the original update rule (Eq. 22b) typically causes ADMM to settle at dual variables that enforce resource limits only when the penalty is active; removing the augmentation then reveals violations of the shared constraints. This behavior appears in the left panel of Figure 2 for a pair of representative resources.

In contrast, applying the proposed update (Eq. 23) drives ADMM toward the genuine dual optimum, delivering dual variables that guarantee feasibility in both the augmented and the unaugmented settings. The right panel of Figure 2 demonstrates this superior outcome. Consequently, all further discussion and results in this work rely on updating the auxiliary variables via (Eq. 23).

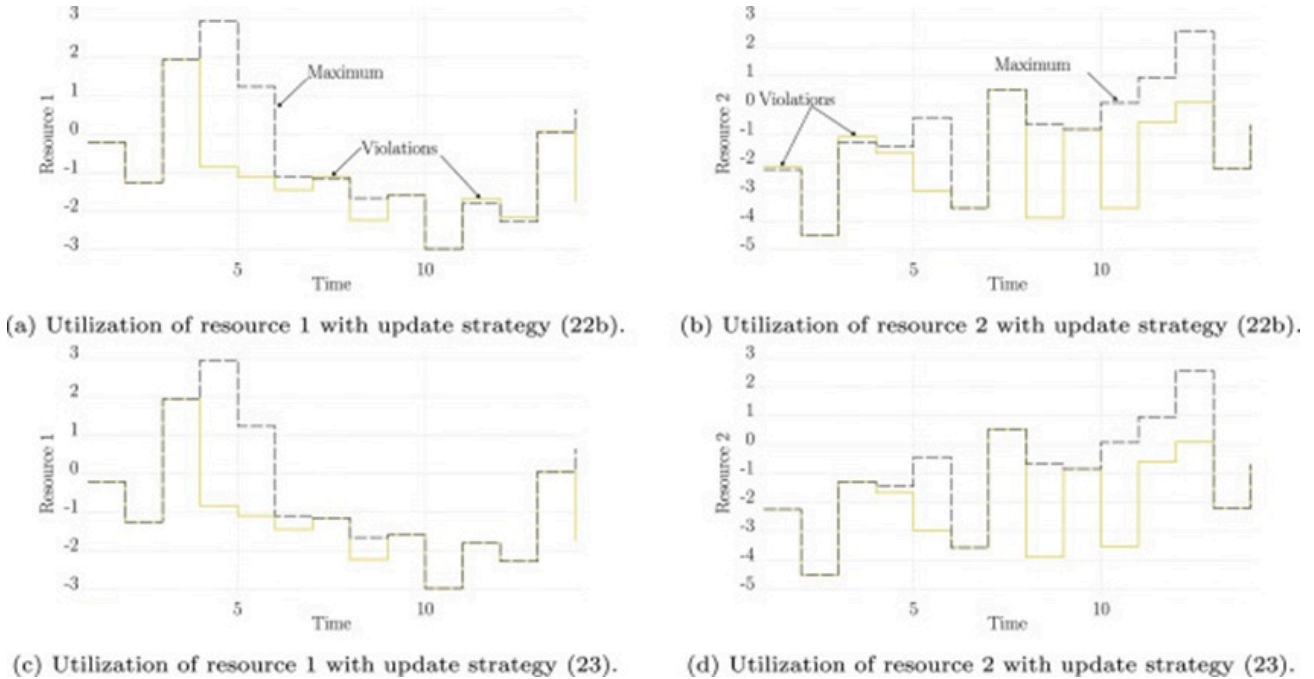


Figure 2. Comparison of the resource utilization upon convergence of the ADMM algorithm for the DMPC benchmark problem with different update strategies of the auxiliary variables. The plots show the aggregated resource utilization for a DMPC problem with $N_s = 20$, $n_x = 3$, $n_u = 2$, $n_v = 2$, $N_p = 15$. The constraints on the shared resources are violated when using the update strategy (Eq. 22b).

Quasi-Newton dual ascent

Yfantis *et al.* recently introduced the quasi-Newton dual ascent (QNDA) method [2, 3]. The approach relies on constructing a quadratic model to approximate the concave dual function.

$$d_B^{(t)}(\lambda_{N_p}) = \frac{1}{2}(\lambda_{N_p} - \lambda_{N_p}^{(t)})^T \mathbf{B}^{(t)} (\lambda_{N_p} - \lambda_{N_p}^{(t)}) + \mathbf{g}^T(\lambda_{N_p}^{(t)}) (\lambda_{N_p} - \lambda_{N_p}^{(t)}) + d(\lambda_{N_p}^{(t)}), \quad (24)$$

This strategy draws inspiration from classical Newton-type methods. Since the dual function is nonsmooth, however, exact gradients and Hessians cannot be computed. The algorithm therefore employs subgradients and updates an approximate Hessian at each step using the Broyden–Goldfarb–Shanno–Fletcher (BFGS) formula:

$$\mathbf{B}^{(t)} = \mathbf{B}^{(t-1)} + \frac{\mathbf{y}^{(t)} \mathbf{y}^{(t)T}}{\mathbf{y}^{(t)T} \mathbf{s}^{(t)}} - \frac{\mathbf{B}^{(t-1)} \mathbf{s}^{(t)} \mathbf{s}^{(t)T} \mathbf{B}^{(t-1)}}{\mathbf{s}^{(t)T} \mathbf{B}^{(t-1)} \mathbf{s}^{(t)}}, \quad (25)$$

Where,

$$\mathbf{s}^{(t)} = \lambda_{N_p}^{(t)} - \lambda_{N_p}^{(t-1)} \quad (26)$$

Denotes the change in the dual variables between consecutive iterations and:

$$\mathbf{y}^{(t)} = \mathbf{g}(\lambda_{N_p}^{(t)}) - \mathbf{g}(\lambda_{N_p}^{(t-1)}) \quad (27)$$

Represents the corresponding change in the subgradients. Once the quadratic model has been updated, it is maximized to obtain the next iterate of the dual variables.

Although the model $d_B^{(t)}(\lambda_{N_p})$ provides a smooth quadratic surrogate, the true dual function remains nonsmooth, leading to substantial modeling inaccuracies—particularly near or at points where differentiability fails. To mitigate this drawback, the algorithm incorporates information from the bundle defined in (Eq. 16). While bundle trust methods (BTM) exploit the bundle to form a piecewise linear outer approximation of the dual function, the QNDA scheme instead uses the collected data to generate supporting hyperplanes (cutting planes) that are enforced as linear inequalities within the update subproblem. Because any subgradient defines the normal to a supporting hyperplane of the concave dual function, each such plane gives rise to the linear constraint:

$$d_B^{(t)}(\lambda_{N_p}^{(t+1)}) \leq d(\lambda_{N_p}^{(t)}) + \mathbf{g}^T(\lambda_{N_p}^{(t)}) (\lambda_{N_p}^{(t+1)} - \lambda_{N_p}^{(t)}), \quad \forall l \in \{t - \tau + 1, \dots, t\}. \quad (28)$$

It is added to the update problem. This prevents the update from leaving the validity range of the approximation. Constraints (Eq. 28) are referred to as bundle cuts and summarized as

$$\mathcal{B}_{d_B}^{(t)} = \left\{ \lambda_{N_p} \in \mathbb{R}^{n_\lambda} \mid d_B^{(t)}(\lambda_{N_p}) \leq d(\lambda_{N_p}^{(l)}) + \mathbf{g}^T(\lambda_{N_p}^{(l)}) (\lambda_{N_p} - \lambda_{N_p}^{(l)}), \forall l \in \{t - \tau + 1, \dots, t\} \right\}. \quad (29)$$

In the remainder of the text, Yfantis *et al.* suggested computing the dual-variable update at every iteration by solving the following optimization problem [2, 3]:

$$\lambda_{N_p}^{(t+1)} = \arg \max_{\lambda_{N_p}} d_B^{(t)}(\lambda_{N_p}), \quad (30a)$$

$$\text{s. t. } \|\lambda_{N_p} - \lambda_{N_p}^{(t)}\|_2^2 \leq \alpha^{(t)}, \quad (30b)$$

$$\lambda_{N_p} \in \mathcal{B}_{d_B}^{(t)}, \quad (30c)$$

$$\lambda_{N_p} \geq \mathbf{0}. \quad (30d)$$

The constraint (Eq. 30b) imposes the identical trust-region bound employed in the BTM formulation (Eq. 17b). Enforcing all bundle cuts (Eq. 30c) from the outset can sometimes cause sluggish progress in the early stages of the algorithm. For this reason, the cuts are activated only when the current iterate lies within a user-specified distance from the anticipated optimum, controlled by a threshold.

$$\|\mathbf{w}_p(\lambda_{N_p}^{(t)})\|_2 \leq \epsilon_b \cdot \|\mathbf{w}_p(\lambda_{N_p}^{(0)})\|_2. \quad (31)$$

The resulting problem (Eq. 30) constitutes a quadratically constrained quadratic program (QCQP). Extensive numerical experiments indicate, however, that greater efficiency and robustness can be achieved by reformulating the bundle cuts using an epigraph representation:

$$\lambda_{N_p}^{(t+1)} = \arg \max_{\lambda_{N_p}, v} d_B^{(t)}(\lambda_{N_p}), \quad (32a)$$

$$\text{s. t. } \|\lambda_{N_p} - \lambda_{N_p}^{(t)}\|_2^2 \leq \alpha^{(t)}, \quad (32b)$$

$$d_B^{(t)}(\lambda_{N_p}) \leq v \quad (32c)$$

$$v \leq d(\lambda_{N_p}^{(l)}) + \mathbf{g}^T(\lambda_{N_p}^{(l)}) (\lambda_{N_p}^{(t+1)} - \lambda_{N_p}^{(l)}), \forall l \in \{t - \tau + 1, \dots, t\}, \quad (32d)$$

$$\lambda_{N_p} \geq \mathbf{0}. \quad (32e)$$

Here, the auxiliary variable v serves as an upper bound on the current quadratic approximation $d_B^{(t)}(\lambda_{N_p})$ at the candidate point (Eq. 32c). Thanks to this reformulation, the bundle cuts become simple linear inequalities (Eq. 32d), thereby improving numerical stability and accelerating overall convergence.

Preliminary numerical trials also indicated that the QNDA procedure generally reaches the region near the optimal point quite rapidly when applied to typical DMPC instances. Close to the solution, however, it frequently displays persistent oscillations that hinder satisfactory overall efficiency. To suppress this unwanted behavior, the mechanism for advancing the dual variables is changed—once the current iterate is sufficiently near the optimum—to a line-search procedure subject to constraints, rather than continuing with the trust-region approach:

$$\mathbf{s}^{(t)} = \arg \max_{\mathbf{s} \in \mathbb{R}^{n_\lambda}, \hat{\alpha} \in \mathbb{R}} \hat{\alpha}, \quad (33a)$$

$$\text{s. t. } \lambda^{(t)} + \mathbf{s} \in \mathcal{B}_{d_B}^{(t)}, \quad (33b)$$

$$\mathbf{s} = \hat{\alpha} \nabla d_B^{(t)} \left(\lambda^{(t)} \right), \quad (33c)$$

$$\hat{\alpha} \leq \alpha^{(t)}, \quad (33d)$$

$$\lambda^{(t+1)} = \left[\lambda^{(t)} + \mathbf{s}^{(t)} \right]^+. \quad (33e)$$

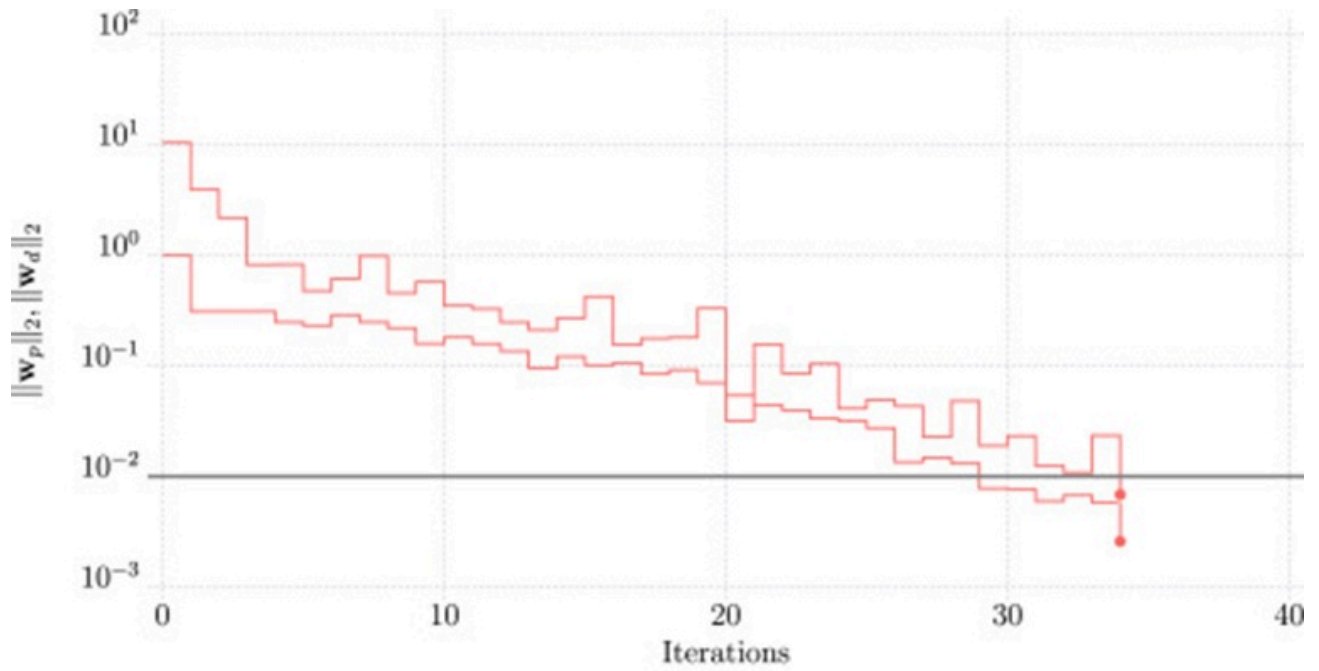
The revised update rule (Eq. 33) essentially performs a step along the gradient direction computed from the quadratic model of the dual function. Constraint (Eq. 33b) guarantees adherence to the bundle cuts by adopting the epigraph-style representation from (Eq. 32c) to (Eq. 32d). The actual search direction appears in (Eq. 33c), and the allowable step length is restricted by (Eq. 33d), which takes the place of the previous trust-region bound. After determining this direction, the dual variables are incremented and then projected back into the nonnegative orthant to satisfy the nonnegativity condition imposed on them.

Activation of this constrained line-search update (Eq. 33) occurs only when the following condition is met:

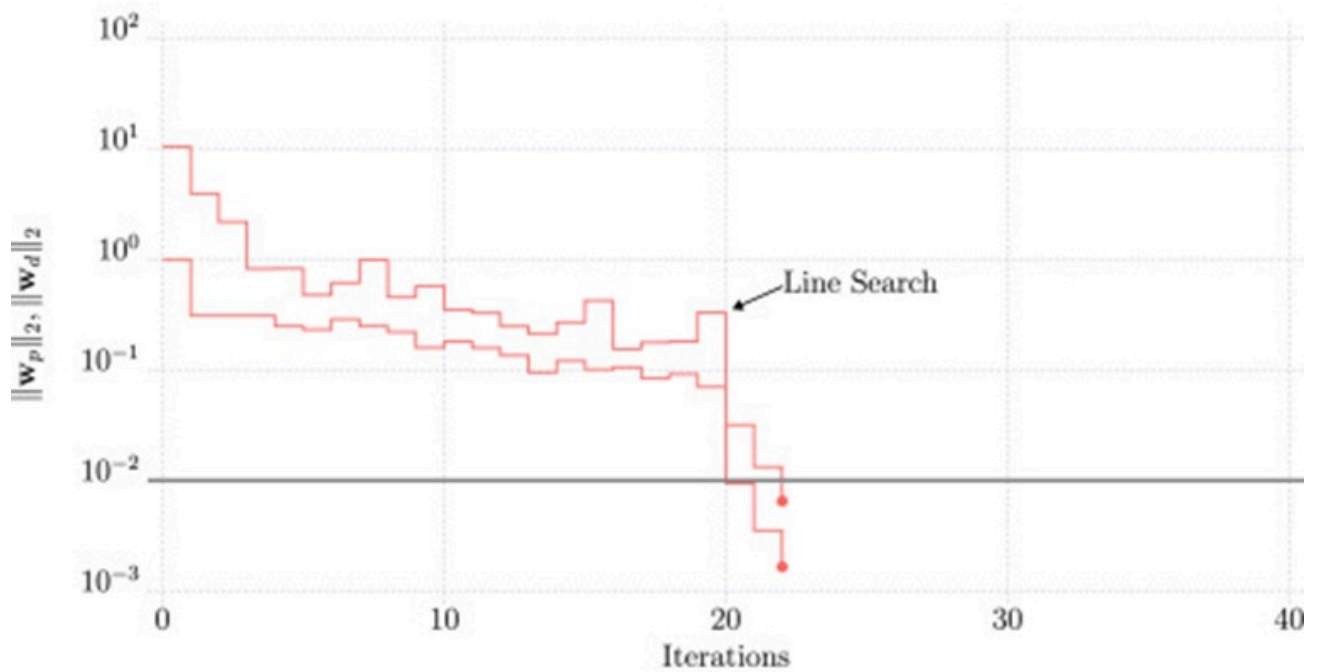
$$\left\| \mathbf{w}_p \left(\lambda^{(t)} \right) \right\|_2 \leq \epsilon_l \cdot \left\| \mathbf{w}_p \left(\lambda^{(0)} \right) \right\|_2, \quad (34)$$

With ϵ_l selected small enough to confirm that the iterates have indeed entered the immediate neighborhood of the optimal dual solution.

Figure 3 highlights the advantage offered by this line-search modification. The progression of primal and dual residual norms without line search is shown in Figure 3a. For comparison, Figure 3b shows the much smoother, faster decay of both residuals once the line-search update (Eq. 33) is engaged near the optimum.



(a) Evolution of the primal and dual residuals without the line search strategy.



(b) Evolution of the primal and dual residuals with the line search strategy.

Figure 3. Comparison of the convergence of the QNDA algorithm for a DMPC benchmark problem with and without the line search update (Eq. 33). The plots show the evolutions of the primal and dual residuals for a DMPC problem with $N_s = 20$, $n_x = 3$, $n_u = 2$, $n_v = 2$, $N_p = 15$. The line search is used once the algorithm is near the optimum.

Dual decomposition applied to dynamically coupled DMPC problems

In the interest of completeness, this section briefly outlines the extension of dual decomposition to enable the distributed solution of DMPC problems with dynamic interconnections, as expressed in (4). To maintain brevity, coupling is restricted to the state equations alone, i.e., $\mathbf{B}_{ij} = \mathbf{0}$ for all $i \neq j$.

Begin by defining, for each subsystem i , the set of its direct neighbors $\mathcal{N}_i \subset \mathcal{I}$, consisting of all indices j for which $\mathbf{A}_{ij} \neq \mathbf{0}$. Next, introduce auxiliary decision variables $\mathbf{v}_{ij}$ within subsystem i ; these act as local replicas of the state trajectories from neighboring subsystem j . With this augmentation, the global centralized problem (4) can be recast in the equivalent form.

$$\begin{aligned} \min_{\mathbf{x}^{0:N_p}, \mathbf{u}^{0:N_p-1}, \mathbf{v}^{0:N_p}} \quad & \sum_{i \in \mathcal{I}} \left[J_i^f \left(\mathbf{x}_i^{N_p} \right) + \sum_{k=0}^{N_p-1} J_i \left(\mathbf{x}_i^k, \mathbf{u}_i^k \right) \right], \\ \text{s. t. } \quad & \mathbf{x}_i^{k+1} = \mathbf{A}_{ii} \mathbf{x}_i^k + \sum_{j \in \mathcal{N}_i} \mathbf{A}_{ij} \mathbf{v}_{ij}^k + \mathbf{B}_i \mathbf{u}_i^k, \quad \forall i \in \mathcal{I}, \end{aligned} \quad (35a)$$

$$k = 0, \dots, N_p - 1, \quad (35b)$$

$$\mathbf{x}_i^0 = \tilde{\mathbf{x}}(t_0), \quad \forall i \in \mathcal{I}, \quad (35c)$$

$$\mathbf{x}_i^k \in \mathcal{X}_i \subset \mathbb{R}^{n_{x_i}}, \quad \forall i \in \mathcal{I}, k = 0, \dots, N_p, \quad (35d)$$

$$\mathbf{u}_i^k \in \mathcal{U}_i \subset \mathbb{R}^{n_{u_i}}, \quad \forall i \in \mathcal{I}, k = 0, \dots, N_p - 1, \quad (35e)$$

$$\mathbf{v}_{ij}^k - \mathbf{x}_j^k = \mathbf{0}, \quad \forall i \in \mathcal{I}, j \in \mathcal{N}_i, k = 0, \dots, N_p. \quad (35f)$$

Under this rewritten structure (Eq. 35), the only remaining inter-subsystem links are the consistency (copy) constraints (Eq. 35f). Dual variables λ_{kij} ($i \neq j$) are then associated with each of these equality constraints. The resulting local Lagrangian for subsystem i becomes

$$\begin{aligned} \mathcal{L}_i \left(\mathbf{x}_i^{k:N_p}, \mathbf{u}_i^{0:N_p-1}, \mathbf{v}_{ij}^{0:N_p}, \lambda_{ij}^{0:N_p} \right) &= J_i^f \left(\mathbf{x}_i^{N_p} \right) + \sum_{k=0}^{N_p-1} J_i \left(\mathbf{x}_i^k, \mathbf{u}_i^k \right) \\ &+ \sum_{k=0}^{N_p} \sum_{j \in \mathcal{N}_i} \left[\lambda_{ij}^{k,T} \mathbf{v}_{ij}^k - \lambda_{ji}^{k,T} \mathbf{x}_i^k \right]. \end{aligned} \quad (36)$$

Thereafter, the complete MPC optimization task (Eq. 35) admits a fully distributed resolution: every subsystem independently minimizes its own Lagrangian. At the same time, the dual variables provide the necessary coordination across the network to enforce global consistency.

Numerical Analysis for DMPC Problems

This section evaluates the performance of four distributed optimization algorithms—the subgradient method, the bundle trust method (BTM), ADMM, and the quasi-Newton dual ascent (QNDA)—by applying them to several carefully constructed DMPC benchmark instances. All test problems follow the general structure of problem (5), featuring a reference-tracking cost functional as in (2) supplemented by a terminal penalty term (3).

The weighting matrices appearing in the objective were created randomly as symmetric positive definite matrices using the following procedure:

$$\mathbf{H}_{\mathbf{x}_i} = \mathbf{H}_{\mathbf{x}_i}^f = \mathbf{N}_{\mathbf{x}_i}^T \mathbf{N}_{\mathbf{x}_i} \in \mathbb{R}^{n_{\mathbf{x}_i} \times n_{\mathbf{x}_i}}, \quad (37)$$

$$\mathbf{H}_{\mathbf{u}_i} = \mathbf{N}_{\mathbf{u}_i}^T \mathbf{N}_{\mathbf{u}_i} \in \mathbb{R}^{n_{\mathbf{u}_i} \times n_{\mathbf{u}_i}}, \quad (38)$$

Where the entries of $\mathbf{N}_{\mathbf{x}_i}$ and $\mathbf{N}_{\mathbf{u}_i}$ were independently sampled from the standard normal distribution, i.e., $[\mathbf{N}_{\mathbf{x}_i}]_{l,j}, [\mathbf{N}_{\mathbf{u}_i}]_{l,j} \in (\mu = 0, \sigma = 1)$.

System matrices $\mathbf{A}_i$ were designed so that each resulting subsystem is asymptotically stable. For discrete-time linear systems, stability requires that every eigenvalue of $\mathbf{A}_i$ lies strictly inside the unit disk [48]. Accordingly, each $\mathbf{A}_i$ was formed via

$$\mathbf{A}_i = \mathbf{S}_i \mathbf{D}_i \mathbf{S}_i^{-1}, \quad (39)$$

Here, $\mathbf{D}_i$ is a diagonal matrix whose diagonal entries—the eigenvalues of $\mathbf{A}_i$ —were drawn from the continuous uniform distribution on $(-1, 1)$, i.e., $[\mathbf{D}_i]_{l,l} \in (-1, 1)$. The elements of the orthogonal matrix $\mathbf{S}_i$ were independently sampled from $\mathcal{U}_c(-3, 3)$. The input-coupling matrices $\mathbf{B}_i$ had entries drawn independently from the continuous uniform distribution $\mathcal{U}_c(-2, 2)$.

State and input constraints for each subsystem were defined as the intersection of several ellipsoids centered at the origin:

$$\mathcal{X}_i = \left\{ \mathbf{x}_i \in \mathbb{R}^{n_{\mathbf{x}_i}} \mid \mathbf{x}_i^T \mathbf{G}_{\mathbf{x}_i, l} \mathbf{x}_i \leq p_{\mathbf{x}_i, l}^2, l = 1, \dots, n_c \right\}, \quad (40)$$

$$\mathcal{U}_i = \left\{ \mathbf{u}_i \in \mathbb{R}^{n_{\mathbf{u}_i}} \mid \mathbf{u}_i^T \mathbf{G}_{\mathbf{u}_i, l} \mathbf{u}_i \leq p_{\mathbf{u}_i, l}^2, l = 1, \dots, n_c \right\}. \quad (41)$$

The shaping matrices $\mathbf{G}$, and $\mathbf{G}_{\mathbf{u}}$ were generated identically to the objective matrices (positive definite, random via the normal-based procedure above). The right-hand-side scalars $p_{\mathbf{x}_i}$, and $p_{\mathbf{u}_i}$, were independently drawn from $(1, 5)$.

To ensure meaningful test cases, the desired reference trajectory $\mathbf{x}_i^{\text{ref}, 0:N_p}$ should itself be feasible. Its components were therefore sampled pointwise from $(-2, 2)$ at every time step until a feasible state (i.e., $\mathbf{x}_i^{\text{ref}} \in \mathcal{X}_i$ for all k) was obtained. The initial state $\tilde{\mathbf{x}}(t_0)$ was generated following the same rejection-sampling procedure.

Entries of the resource-interaction matrix $\mathbf{R}_i$ were initially drawn from the uniform distribution $(1, 2)$. Subsequently, each entry was independently modified by multiplying it with a factor sampled from the discrete uniform distribution over $\{-1, 0, 1\}$:

$$\mathbf{R}_i = \mathbf{D}_i \circ \mathbf{C}_i \in \mathbb{R}^{n_r \times n_{\mathbf{u}_i}}, [\mathbf{D}_i]_{l,j} \in \mathcal{U}_c(1, 2), [\mathbf{C}_i]_{l,j} \in \mathcal{U}_d[-1, 0, 1]. \quad (42)$$

This random sign flip or zeroing guarantees that not every subsystem participates in every resource network and that some subsystems act as net producers. In contrast, others act as net consumers for any given resource.

The upper bounds on resource usage were chosen deliberately so that the purely decentralized solution (corresponding to $\lambda = \mathbf{0}$) becomes infeasible, whereas the fully coordinated (system-wide) problem remains feasible. To achieve this, the unconstrained centralized MPC problem (resource limits removed) was first solved to obtain the optimal input sequence. From this unconstrained optimum, the corresponding resource consumption profile was calculated:

$$\mathbf{u}^{0:N_p-1,*} = \arg \min_{\mathbf{x}^{0:N_p}, \mathbf{u}^{0:N_p-1}} \sum_{i \in \mathcal{I}} \left[J_i^f(\mathbf{x}_i^{N_p}) + \sum_{k=0}^{N_p-1} J_i(\mathbf{x}_i^k, \mathbf{u}_i^k) \right], \quad (43a)$$

$$\text{s. t. } \mathbf{x}_i^{k+1} = \mathbf{A}_i \mathbf{x}_i^k + \mathbf{B}_i \mathbf{u}_i^k, \quad \forall i \in \mathcal{I}, \quad k = 0, \dots, N_p - 1,$$

$$\mathbf{x}_i^0 = \tilde{\mathbf{x}}(t_0), \quad \forall i \in \mathcal{I}, \quad (43b)$$

$$\begin{aligned} \mathbf{x}_i^k &\in \mathcal{X}_i \subset \mathbb{R}^{n_{x_i}}, \quad \forall i \in \mathcal{I}, k = 0, \dots, N_p, \\ \mathbf{u}_i^k &\in \mathcal{U}_i \subset \mathbb{R}^{n_{u_i}}, \quad \forall i \in \mathcal{I}, \end{aligned} \quad (43c)$$

$$k = 0, \dots, N_p - 1. \quad (43d)$$

The resource bounds were then set by scaling down this unconstrained usage profile:

$$\mathbf{r}^{k,*} = \sum_{i \in \mathcal{I}} \mathbf{R}_i \mathbf{u}_i^{k,*}, \quad k = 0, \dots, N_p - 1. \quad (44)$$

After defining these tightened resource limits, the feasibility of both the decentralized problem ($\lambda = \mathbf{0}$) and the centralized (coupled) problem was explicitly verified to confirm the desired property.

$$\begin{aligned} [\mathbf{r}_{\max}^k]_l &= [\mathbf{r}^{k,*}]_l - \beta_l^k |[\mathbf{r}^{k,*}]_l|, \quad k = 0, \dots, N_p - 1, \quad l = 1, \dots, n_r, \quad \beta_l^k \\ &\in \mathcal{U}_c(0, 0.2). \end{aligned} \quad (45)$$

The dimensions and counts of the subsystems were systematically varied across these ranges:

- Total subsystems: $N_s \in \{5, 10, 20, 50\}$
- States per subsystem: $n_x \in \{2, 3, 4, 5\}$
- Inputs per subsystem: $n_u \in \{2, 3, 4, 5\}$, subject to $n_x \geq n_u$
- Shared resource channels: $n_r \in \{2, 3, 4, 5\}$, subject to $n_u \geq n_r$
- Length of prediction horizon: $N_p \in \{10, 15, 20\}$

All subsystems were given uniform internal complexity, so $n_{xi} = n_x$ and $n_{ui} = n_u$ held for every $\forall i \in I$. The quantity of local state constraints and input constraints per subsystem was deliberately matched to the number of states, giving $n_c = n_x$.

Ten separate benchmark instances were randomly created for every possible combination of the above parameters, producing an overall collection of 2400 different DMPC test cases.

Both distributed algorithms and the individual DMPC subproblems were implemented in Julia [49], leveraging the JuMP algebraic modeling framework [50]. Every local DMPC optimization instance was processed by the commercial MIP solver Gurobi [51]. The trust-region update subproblem within BTM—linearly objective with affine equality constraints and convex quadratic inequality constraints—was also sent to Gurobi. Conversely, the quadratic update subproblems that arise in QNDA were solved using the nonlinear interior-point solver IPOPT [52].

Every simulation run was executed on a conventional laptop computer featuring an Intel(R) Core(TM) i5-6200U processor clocked at 2.30 GHz and equipped with 8 GB of main memory.

Algorithm efficiency was quantified by estimating the total time required to complete each distributed optimization task according to the formula introduced in the study by Pflaum et al. [32]:

$$T_{\text{comp}} = N_{\text{iter}} \cdot T_{\text{comm}} + \sum_{t=1}^{N_{\text{iter}}} \left(T_{\text{update}}^{(t)} + \max_{i \in \mathcal{I}} T_{\text{sub},i}^{(t)} \right), \quad (46)$$

Where N_{iter} stands for the total iteration count, T_{comm} is the assumed constant per-iteration communication latency between coordinator and subsystems, $T_{\text{update}}^{(t)}$ is the coordinator's CPU time for revising the dual variables in iteration t , and $T_{\text{sub},i}^{(t)}$ denotes the solve time of local subproblem i during iteration t .

In a parallel distributed execution, the subproblems can run concurrently; however, the coordinator must wait for the last subsystem to finish before proceeding, so the slowest local solve bounds the effective per-iteration primal collection time. For consistency across all experiments reported below, the fixed communication delay was set to $T_{\text{comm}} = 800$ ms.

Table 1. Parameter settings of the distributed optimization algorithms for the DMPC benchmark problems.

Value		Description	Algorithms
$\lambda(0)$	0	initial dual variables	All
$\alpha(0)$	1	initial step size/trust region parameter	SG, BTM, QNDA
i_{max}	500	maximum number of iterations	All
ϵ_p	10 ^{−2}	primal residual convergence tolerance	All
ϵ_d	10 ^{−2}	dual residual convergence tolerance	All
ϵ_b	0.6	bundle cuts threshold	QNDA
ϵ_l	10 ^{−2}	line search threshold	QNDA
$\varrho(0)$	10 ^{−3}	initial regularization parameter	ADMM
τ_{incr}	1.25	see (21)	ADMM
τ_{decr}	1.1	see (21)	ADMM

μ	10	see (21)	ADMM
$\mathbf{z}(0)$	$\mathbf{0}$	initial auxiliary variables	ADMM
τ	150	allowed age of data points	BTM, QNDA
$\mathbf{B}(0)$	$-\mathbf{I}$	Initial approximated Hessian	QNDA

Parameter settings for DMPC problems

The tuning parameters for the DMPC test cases were determined empirically through iterative experimentation to achieve an effective balance between numerical stability and convergence speed.

For the subgradient method (SG), the initial step-size factor was fixed at $\alpha(0) = 1$; the same starting value was used for both the initial trust-region radius in BTM and QNDA. In all three cases, this parameter was subsequently adjusted following the rule.

$$\alpha^{(t)} = \frac{\alpha^{(0)}}{\max \left\{ \left\| \frac{\mathbf{w}^{(0)}}{p} \right\|_2, \dots, \left\| \mathbf{w}_p^{(t)} \right\|_2 \right\}}. \quad (47)$$

The ADMM scheme proved sensitive to overly aggressive tuning and frequently diverged under such conditions. Consequently, its starting penalty parameter was conservatively set to $\beta(0) = 10^{-3}$. The associated adaptation constants were chosen as $\tau_{\text{incr}} = 1.25$, $\tau_{\text{decr}} = 1.1$, and $\mu = 10$.

The threshold for activating bundle cuts in QNDA was placed at $\epsilon_b = 0.6$ (Eq. 31), while the trigger for switching to the constrained line-search update (33) was set to $\epsilon_l = 10^{-2}$ (see Eq. (34)). For both BTM and QNDA, the bundle retention depth (age parameter) was fixed at $\tau = 150$.

Every algorithm began with dual variables initialized to $\lambda(0) = \mathbf{0}$; for ADMM, the auxiliary variables were likewise started at $\mathbf{z}(0) = \mathbf{0} \forall i \in \mathcal{I}$. The approximate Hessian matrix required by QNDA was initialized as the negative identity matrix.

Termination occurred after a maximum of $\text{imax} = 500$ iterations or whenever both the primal residual and the dual residual dropped below the common tolerance $\epsilon_p = \epsilon_d = 10^{-2}$. A complete overview of all selected parameter values appears in Table 1.

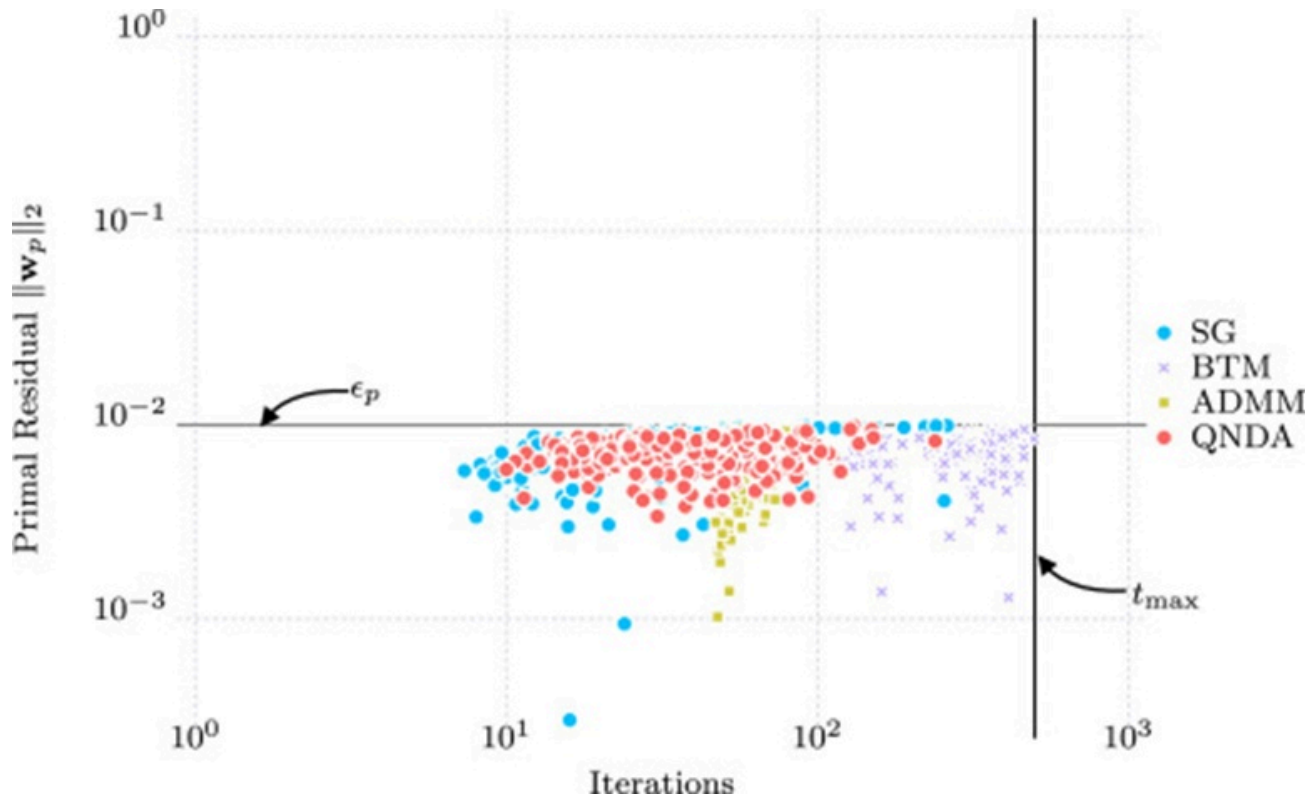


Figure 4. Values of the primal residuals upon termination for the distributed MPC problems. Each data point represents the mean values of the converged problem instances for a tuple $(N_s, n_x, n_u, n_r, N_p)$

Results for DMPC problems

Performance outcomes for the distributed solution of the DMPC benchmark suite are visualized in **Figure 4** and condensed in **Table 2**. Both the ADMM and QNDA algorithms successfully solved nearly all test instances. ADMM converged to a marginally higher proportion of problems. In contrast, QNDA demonstrated clear superiority in three key metrics: fewer iterations, shorter total computation time, and consistently lower residual values (better solution accuracy) among the converged cases.

The classical subgradient method required a comparable number of iterations to QNDA for the problems it could solve, and its simpler update rule translated into lower per-iteration CPU cost, yielding better computational times. Nevertheless, it failed to converge to a substantially larger fraction of the benchmark set than ADMM and QNDA. The bundle trust method (BTM) delivered the weakest overall results across the DMPC test suite.

Table 2. Summary of the results for the distributed optimization of the DMPC benchmark problems (mean values of the converged instances only): mean number of iterations until convergence, $\bar{T}_{comp}$ mean computation time of converged runs (in s), $\bar{\|w_p\|_2}$ mean primal residual of converged runs ($\times 10^{-3}$), $\%_c$: percentage of converged runs within iterations.

Algorithm	$\bar{t}$	$\bar{T}_{comp}$	$\bar{\ w_p\ _2}$	$\%_c$
SG	42.83	35.24	8.18	80.96
BTM	273.83	228.17	7.09	57.58

ADMM	62.24	51.92	7.65	99.21
QNDA	42.6	38.31	7.07	98.21

Since the DMPC problems examined here are strictly convex, strong duality holds, and the optimal duality gap vanishes—the primal and dual objective values coincide at optimality. In contrast, nonconvex formulations (e.g., mixed-integer programs) typically exhibit a shrinking duality gap as the number of subsystems increases [53]. At the same time, a larger number of subsystems usually lessens the nonsmoothness of the dual function [3, 54].

For the convex benchmark problems considered in this study, these characteristics enable distributed solvers to deliver solutions that are noticeably more accurate overall, as shown in Figure 5.

Importantly, the primal feasibility tolerance φ used to terminate the algorithms was intentionally kept fairly loose (Table 1). In most real-world control applications, extremely tight satisfaction of the coupling constraints is unnecessary. Moreover, techniques such as ADMM often converge to high-quality points quickly under moderate tolerances, whereas further tightening the tolerances demands a disproportionately large number of additional iterations [44].

Figure 5 shows that, across the converged runs, the average primal residual steadily decreases as more subsystems are included. Only successfully converged instances contribute to these averages.

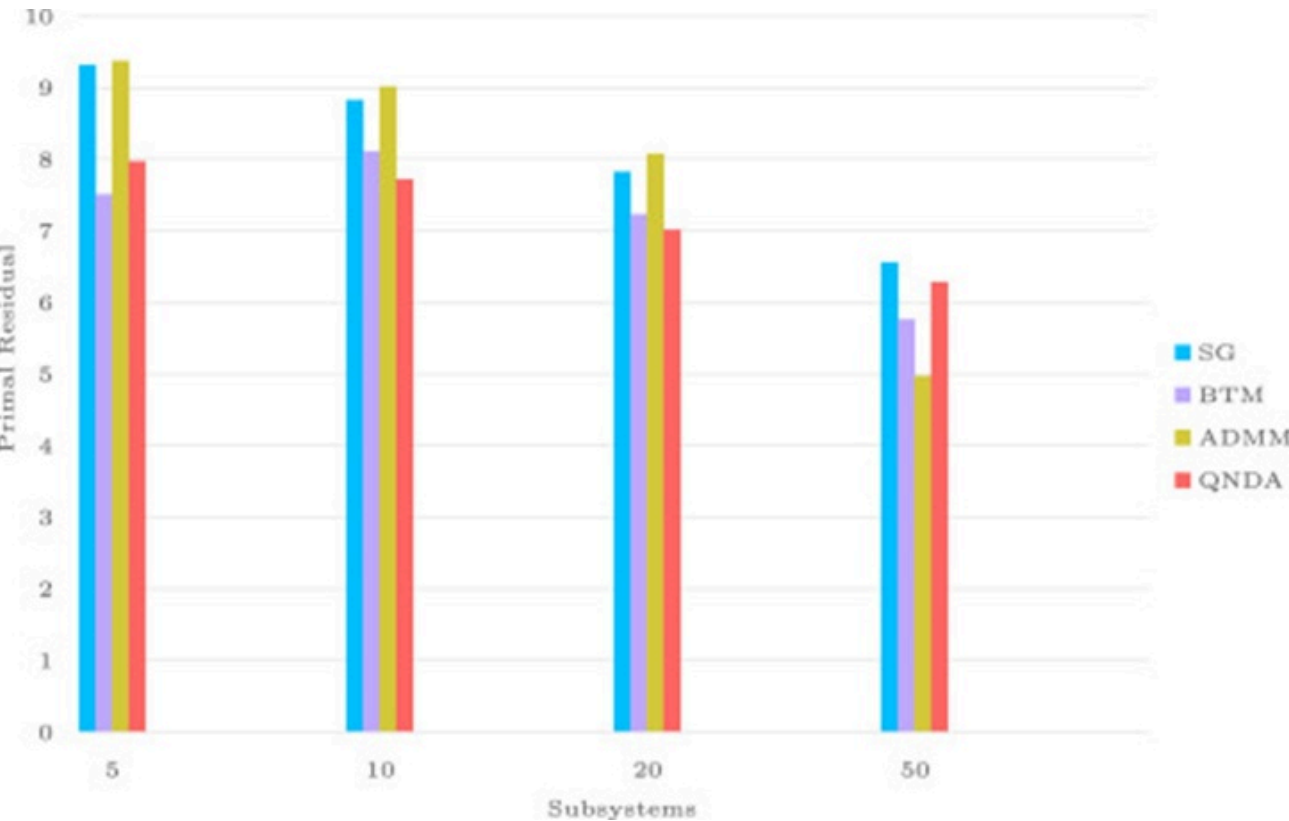


Figure 5. Comparison of the mean primal residuals of converged runs for different numbers of subsystems.

Figure 6 compares the average iteration count as a function of the number of subsystems for each algorithm. The subgradient method and BTM both exhibit a clear upward trend in the number of iterations required as the subsystem count increases. Conversely, ADMM, and especially QNDA, maintain low (and often decreasing) iteration counts, confirming their markedly better scalability and computational efficiency.

Figure 7, on the other hand, indicates that varying the number of shared resources has comparatively little impact on the iteration counts of any algorithm.

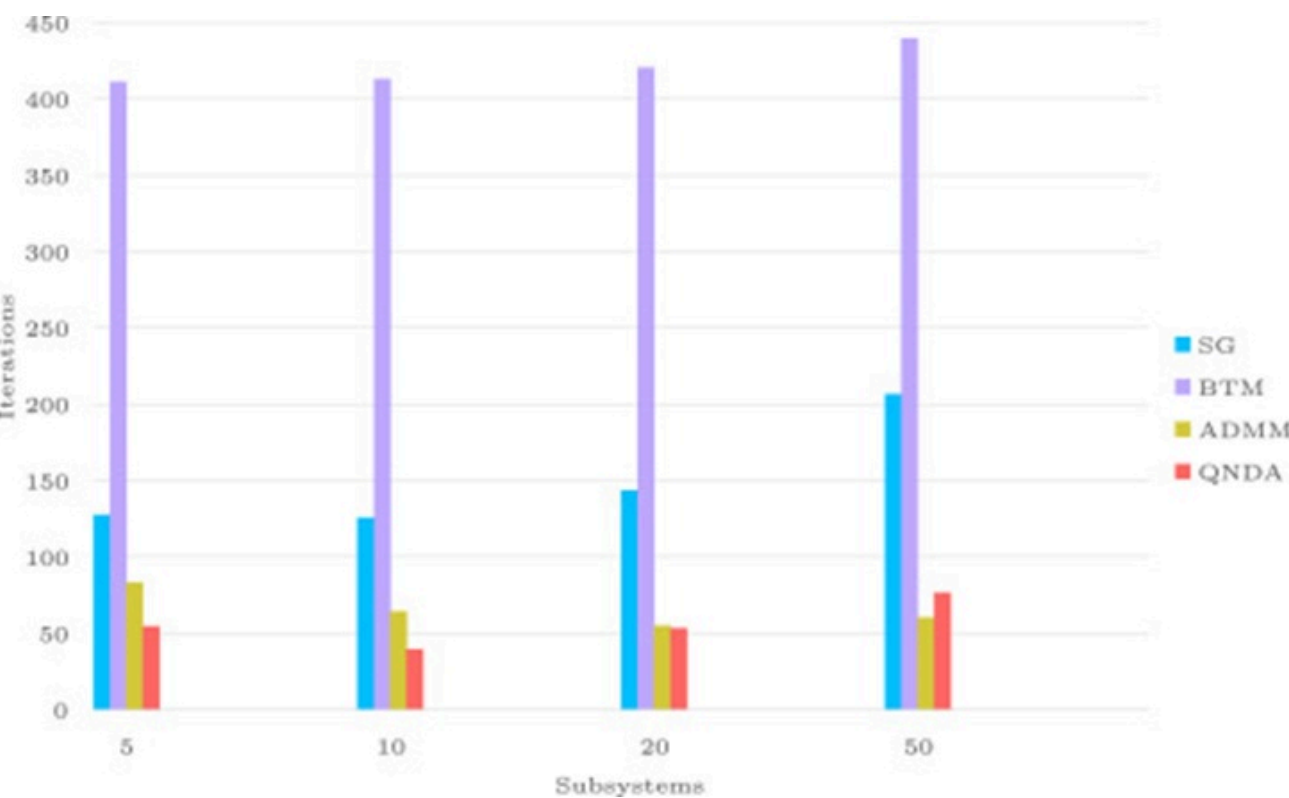


Figure 6. Comparison of the mean number of iterations performed by the different algorithms for different numbers of subsystems.

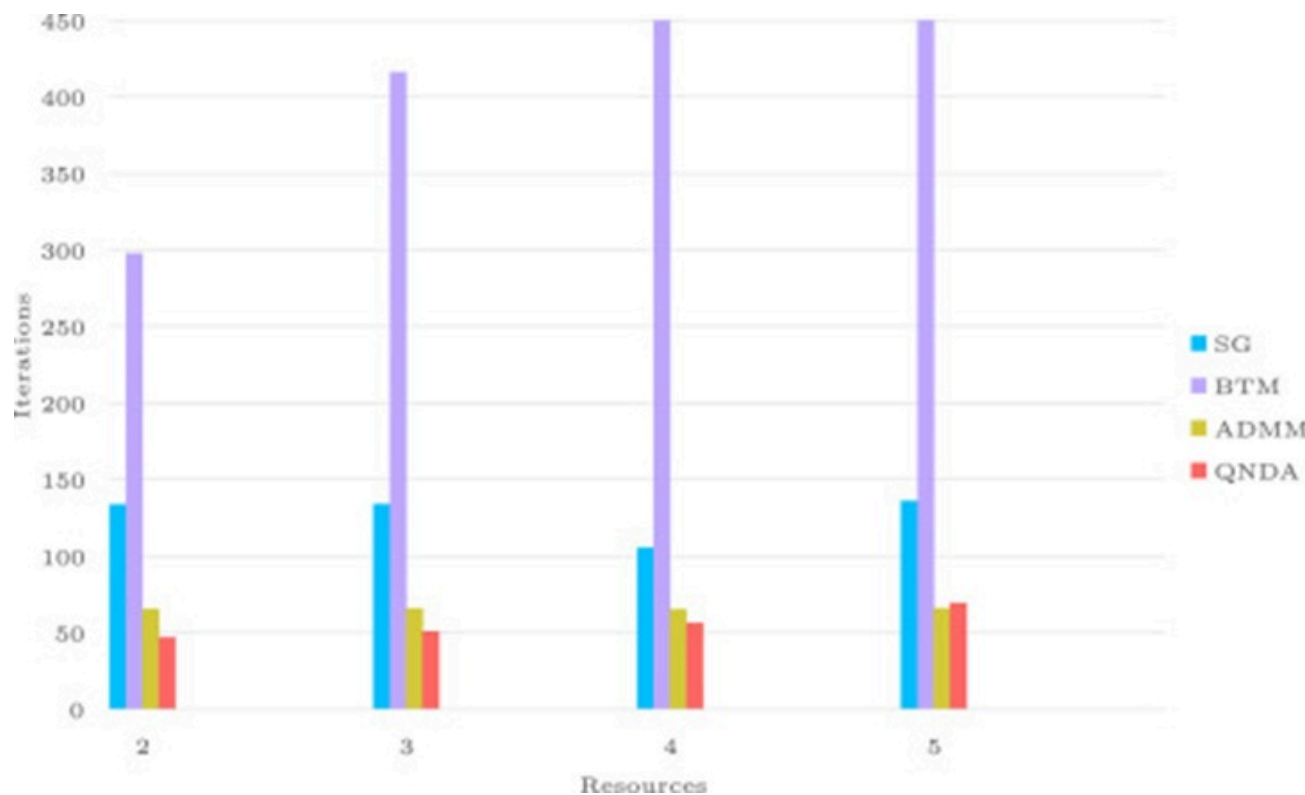
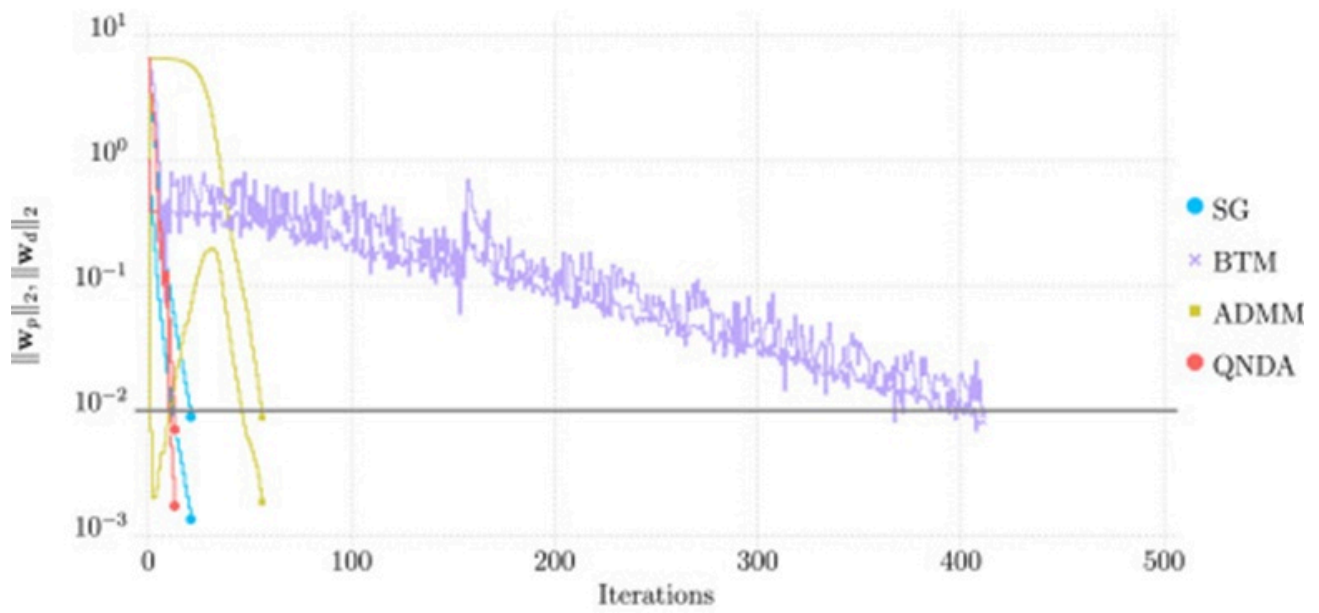
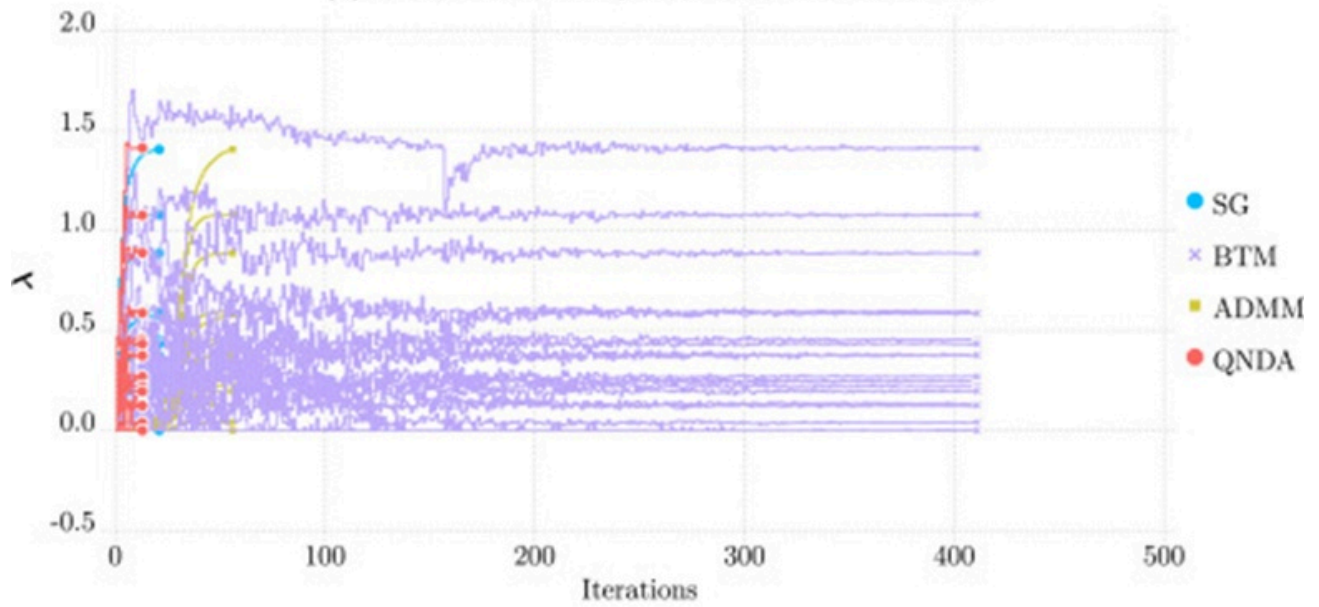


Figure 7. Comparison of the mean number of iterations performed by the different algorithms for different numbers of resources.

Figure 8 traces the convergence history of one representative test case involving three shared resources and a 15-step prediction horizon (yielding dual variables in $\mathbb{R}^{42}$). Here, the subgradient method and QNDA reach convergence most rapidly. ADMM displays minor oscillations but still settles quickly. BTM, by contrast, suffers from large-amplitude oscillations and converges much more slowly relative to the other methods. The magnified view in **Figure 8(b)** reveals that these oscillations are concentrated near the optimal dual point, indicating that resource allocations are particularly sensitive to small adjustments in the dual variables.



(a) Evolution of the primal and dual residuals.



(b) Evolution of the dual variables.

Figure 8. Results for a DMPC benchmark problem with $N_s = 10$, $n_x = 5$, $n_u = 5$, $n_r = 3$, $N_p = 15$.

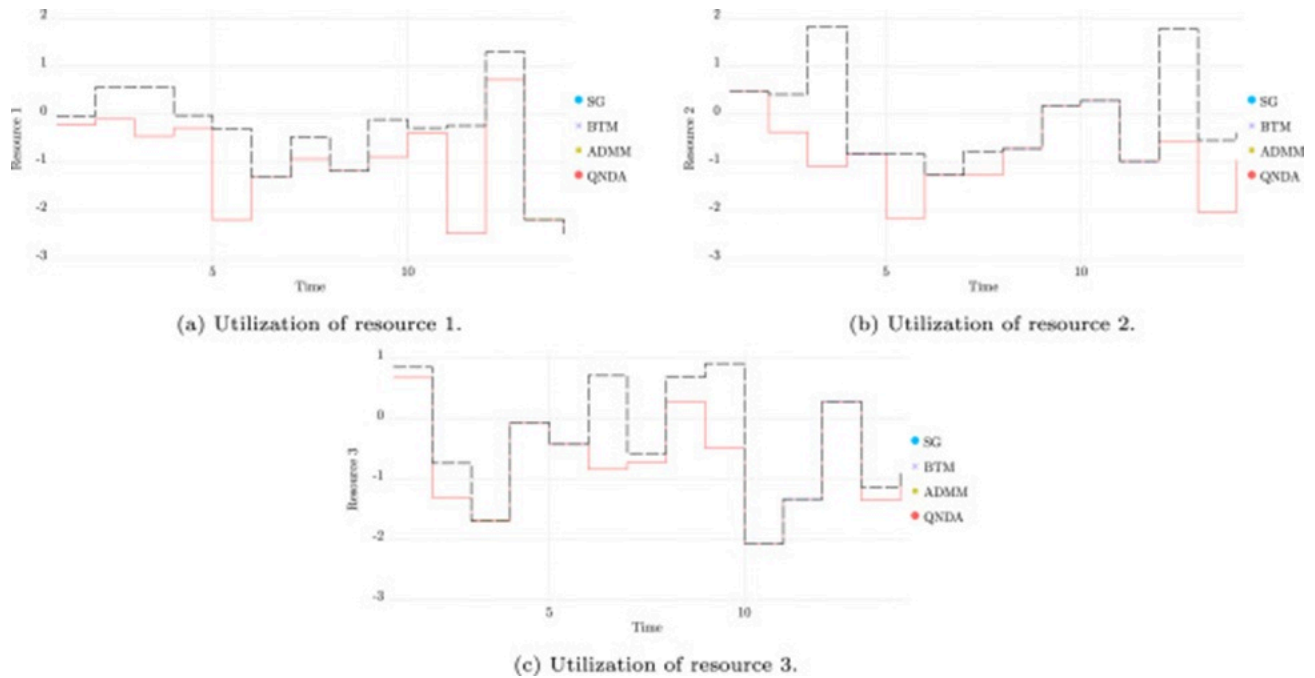


Figure 9. compares the final resource consumption profiles obtained by the four algorithms for the same test problem shown in **Figure 8**. Despite differing convergence paths, all methods terminate at essentially the same (optimal) resource utilization levels, confirming consistency at the dual optimum.

As reported in **Table 2**, the subgradient method achieves performance on par with QNDA across the cases it solves. This level of efficiency, however, depends on quite aggressive step-size choices that significantly reduce robustness.

Figure 10 illustrates convergence curves for a more challenging benchmark with only two shared resources and a shorter 10-step horizon ($\lambda \in \mathbb{R}^{18}$). Under these conditions, the subgradient method diverges wildly due to excessive oscillations triggered by the aggressive step size. All three algorithms converge successfully, with QNDA achieving the best speed. Notably, QNDA and BTM inherit the same trust-region radius as the subgradient method (and QNDA uses the same parameter to control its line-search steps).

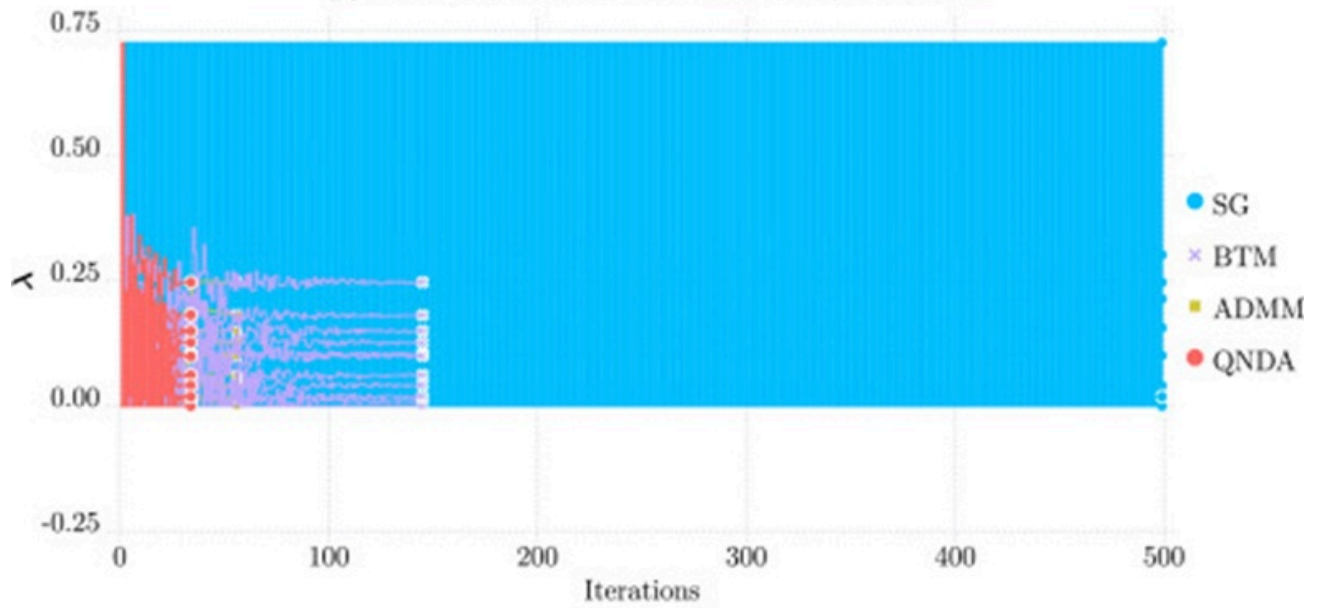
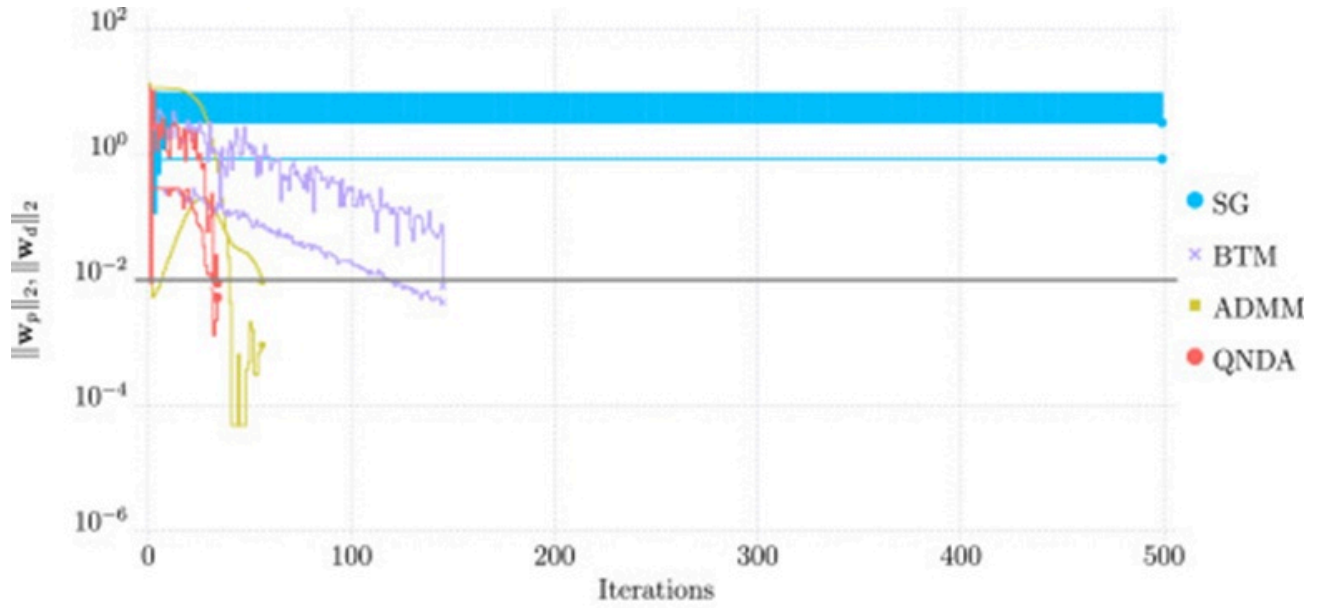


Figure 10. Results for a DMPC benchmark problem with $N_s = 50$, $n_x = 4$, $n_u = 4$, $n_r = 2$, $N_p = 10$.

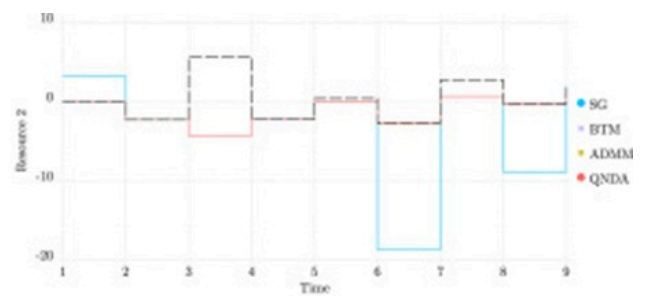
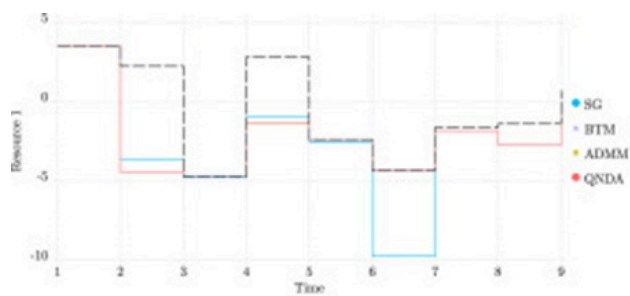


Figure 11. Resource utilization upon termination for a DMPC benchmark problem with $N_S = 50$, $n_x = 4$, $n_u = 4$, $n_r = 2$, $N_p = 10$.

Figure 11 presents the terminal resource utilization for the problem in **Figure 10**. BTM, ADMM, and QNDA all achieve the optimal allocation. **Figure 11(b)** highlights that the subgradient method, however, stops at an infeasible point—over-allocating resource 2.

Conclusion

This work has shown how dual decomposition techniques can be effectively employed to tackle distributed model predictive control (DMPC) problems in which subsystems are interconnected exclusively via shared scarce resources. The recently introduced quasi-Newton dual ascent (QNDA) algorithm was enhanced through two key modifications: an epigraph-based reformulation of the bundle cuts and the incorporation of a constrained line-search step applied once the iterates approach the optimal region. Extensive numerical experiments on a broad collection of benchmark instances confirmed the strong performance of this improved QNDA method.

Although the alternating direction method of multipliers (ADMM) solved the largest proportion of test cases, the classical subgradient method delivered comparable efficiency to QNDA on the problems it solved, and QNDA consistently offered the most favorable trade-off between solution quality, iteration count, computational time, and numerical robustness. The benchmark results further revealed excellent scaling behavior for both ADMM and QNDA. As the number of subsystems increased, both the accuracy of the obtained solutions and the required number of iterations remained favorable. In contrast, the number of shared resources had only a marginal influence on iteration counts across all tested algorithms.

Dual decomposition extends naturally to DMPC problems featuring dynamic coupling between subsystems. Future research could therefore focus on constructing a comprehensive suite of benchmark problems for dynamically coupled DMPC and on systematically comparing the performance of different distributed solvers in that setting.

Linear dynamics governed all systems studied in the present paper. Because dual decomposition is an iterative, hierarchical procedure that requires multiple communication rounds before control inputs can be implemented, it is particularly well suited to processes characterized by slow dynamics—processes that can typically be approximated by linear models with acceptable accuracy over the prediction horizon. However, ongoing improvements in the speed and reliability of nonlinear optimization solvers are making nonlinear MPC increasingly attractive. Consequently, extending dual decomposition-based DMPC to systems described by nonlinear dynamics and nonlinear constraints represents a promising direction for further work.

Nonlinear MPC problems are generally nonconvex, so convergence to an optimal dual solution does not automatically guarantee primal feasibility. In such cases, additional heuristic strategies are usually required to recover a feasible primal solution when applying dual decomposition to nonconvex subproblems. Nevertheless, even in the nonconvex regime, the dual function value remains a rigorous lower bound on the optimal primal objective.

Finally, the literature still lacks realistic, industry-scale DMPC benchmark suites—particularly those involving a very large number of subsystems. While laboratory-scale examples are valuable for illustrating many theoretical and algorithmic properties, only application to truly large-scale industrial systems can fully reveal the practical benefits and limitations of a given coordination strategy.

Acknowledgements

None

Conflict of interest

None

Financial support

None

Ethics statement

None

Received: 25 Mar 2024

Revised: 06 May 2024

Accepted: 19 Jun 2024

Published online: 20 September 2024

Rights and permissions

Open Access The author(s) retain copyright. This article is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. It may be shared and adapted for non-commercial purposes with appropriate attribution, an indication of changes, and distribution of adaptations under the same license. Third-party material may be subject to separate terms identified in its credit line. View the license at <https://creativecommons.org/licenses/by-nc-sa/4.0/>.

References

- Bakule L. Decentralized control: An overview. *Ann Rev Control*. 2008;32(1):87-98.
- Yfantis V, Ruskowski M. A hierarchical dual decomposition-based distributed optimization algorithm combining Quasi-Newton steps and bundle methods. In: 2022 30th Mediterranean Conference on Control and Automation (MED). IEEE; 2022:31-6.
- Yfantis V, Wenzel S, Wagner A, Ruskowski M, Engell S. Hierarchical distributed optimization of constraint-coupled convex and mixed-integer programs using approximations of the dual function. *EURO J Comput Optim*. 2023;11:100058.
- Alvarado I, Limon D, De La Peña DM, Maestre JM, Ridao M, Scheu H. A comparative analysis of distributed MPC techniques applied to the HD-MPC four-tank benchmark. *J Process Control*. 2011;21(5):800-15.
- Maxeiner LS, Engell S. Comparison of dual based optimization methods for distributed trajectory optimization of coupled semi-batch processes. *Optim Eng*. 2020;21(3):761-802.
- Stomberg G, Engelmann A, Faulwasser T. A compendium of optimization algorithms for distributed linear-quadratic MPC. *at-Automatisierungstechnik*. 2022;70(4):317-30.
- Conte C, Summers T, Zeilinger M, Morari M, Jones C. Computational aspects of distributed optimization in model predictive control. In: 2012 IEEE 51st IEEE Conference on Decision and Control (CDC). IEEE; 2012:6819-24.
- Necoara I, Nedelcu V, Dumitrache I. Parallel and distributed optimization methods for estimation and control in networks. *J Process Control*. 2011;21(5):756-66.

- Mayne D, Rawlings J, Rao C, Scokaert P. Constrained model predictive control: Stability and optimality. *Automatica*. 2000;36(6):789-814.
- Engell S. Feedback control for optimal process operation. *J Process Control*. 2007;17(3):203-19.
- Ellis M, Durand H, Christofides P. A tutorial review of economic model predictive control methods. *J Process Control*. 2014;24(8):1156-78.
- Scattolini R. Architectures for distributed and hierarchical model predictive control—a review. *J Process Control*. 2009;19(5):723-31.
- Christofides P, Scattolini R, de la Peña DM, Liu J. Distributed model predictive control: A tutorial review and future research directions. *Comput Chem Eng*. 2013;51:21-41.
- Camponogara E, Jia D, Krogh B, Talukdar S. Distributed model predictive control. *IEEE Control Syst Mag*. 2002;22(1):44-52.
- Maestre J, de la Peña D, Camacho E, Alamo T. Distributed model predictive control based on agent negotiation. *J Process Control*. 2011;21(5):685-97.
- Gafur N, Kanagalingam G, Wagner A, Ruskowski M. Dynamic collision and deadlock avoidance for multiple robotic manipulators. *IEEE Access*. 2022;10:[placeholders].
- Sandell N, Varaiya P, Athans M. A survey of decentralized control methods for large scale systems. *Syst Eng Power*, US Dept of Commerce. 1975:334-5.
- Negenborn R, Maestre J. Distributed model predictive control: An overview and roadmap of future research opportunities. *IEEE Control Syst Mag*. 2014;34(4):87-97.
- Müller MA, Allgöwer F. Economic and distributed model predictive control: Recent developments in optimization-based control. *SICE J Control Meas Syst Integr*. 2017;10(2):39-52.
- Stewart B, Venkat A, Rawlings J, Wright S, Pannocchia G. Cooperative distributed model predictive control. *Syst Control Lett*. 2010;59(8):460-9.
- Maestre J, de la Peña D, Camacho EF. Distributed model predictive control based on a cooperative game. *Optim Control Appl Methods*. 2011;32(2):153-76.
- Zheng Y, Li S, Qiu H. Networked coordination-based distributed model predictive control for large-scale system. *IEEE Trans Control Syst Technol*. 2012;21(3):991-8.
- Giselsson P, Rantzer A. Distributed model predictive control with suboptimality and stability guarantees. In: 2010 49th IEEE Conference on Decision and Control (CDC). IEEE; 2010:7272-7.
- Giselsson P, Doan M, Keviczky T, De Schutter B, Rantzer A. Accelerated gradient methods and dual decomposition in distributed model predictive control. *Automatica*. 2013;49(3):829-33.
- Giselsson P, Rantzer A. On feasibility, stability and performance in distributed model predictive control. *IEEE Trans Autom Control*. 2013;59(4):1031-6.
- Köhler J, Müller MA, Allgöwer F. Distributed model predictive control—Recursive feasibility under inexact dual optimization. *Automatica*. 2019;102:1-9.
- Doan M, Keviczky T, De Schutter B. A distributed optimization-based approach for hierarchical MPC of large-scale systems with coupled dynamics and constraints. In: 2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC). IEEE; 2011:5236-41.

Biegel B, Stoustrup J, Bendtsen J, Andersen P. Model predictive control for power flows in networks with limited capacity. In: 2012 American Control Conference (ACC). IEEE; 2012:2959-64.

Biegel B, Andersen P, Stoustrup J, Bendtsen J. Congestion management in a smart grid via shadow prices. *IFAC Proc Vol.* 2012;45(21):518-23.

Biegel B, Stoustrup J, Andersen P. Distributed MPC via dual decomposition. In: *Distributed Model Predictive Control Made Easy*. Springer; 2014:179-92.

Pflaum P, Alamir M, Lamoudi M. Comparison of a primal and a dual decomposition for distributed MPC in smart districts. In: 2014 IEEE International Conference on Smart Grid Communications (SmartGridComm). IEEE; 2014:55-60.

Pflaum P, Alamir M, Lamoudi M. Scalability study for a hierarchical NMPC scheme for resource sharing problems. In: 2015 IEEE European Control Conference (ECC). IEEE; 2015:1468-73.

Razzanelli M, Crisostomi E, Pallottino L, Pannocchia G. Distributed model predictive control for energy management in a network of microgrids using the dual decomposition method. *Optim Control Appl Methods*. 2020;41(1):25-41.

Eser S, Stoffel P, Kümpel A, Müller D. Distributed model predictive control of a nonlinear building energy system using consensus ADMM. In: 2022 30th Mediterranean Conference on Control and Automation (MED). IEEE; 2022:902-7.

Maxeiner L, Engell S. Hierarchical MPC of batch reactors with shared resources. *IFAC-PapersOnLine*. 2017;50(1):12041-6.

Houska B, Shi J. Distributed MPC with ALADIN—a tutorial. *arXiv preprint arXiv:2204.01654*. 2022.

Yfantis V, Gafur N, Wagner A, Ruskowski M. Hierarchical distributed model predictive control based on dual decomposition and quadratic approximation. In: 2022 30th Mediterranean Conference on Control and Automation (MED). IEEE; 2022:914-9.

Halvgaard R, Vandenberghe L, Poulsen N, Madsen H, Jørgensen J. Distributed model predictive control for smart energy systems. *IEEE Trans Smart Grid*. 2016;7(3):1675-82.

Maestre J, Negenborn R. *Distributed Model Predictive Control Made Easy*. Springer; 2014.

Nocedal J, Wright S. *Numerical Optimization*. Springer Science & Business Media; 2006.

Shor N, Kiwiel K, Ruszcayski A. *Minimization Methods for Non-differentiable Functions*. Springer-Verlag; 1985.

Mäkelä M. Survey of bundle methods for nonsmooth optimization. *Optim Methods Softw*. 2002;17(1):1-29.

Maxeiner L, Engell S. An accelerated dual method based on analytical extrapolation for distributed quadratic optimization of large-scale production complexes. *Comput Chem Eng*. 2020;135(1):106728.

Boyd S, Parikh N, Chu E, Peleato B, Eckstein J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found Trends Mach Learn*. 2011;3(1):1-122.

He B, Yang H, Wang S. Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities. *J Optim Theory Appl*. 2000;106(2):337-56.

Wang S, Liao L. Decomposition method with a variable parameter for a class of monotone variational inequality problems. *J Optim Theory Appl*. 2001;109(2):415-29.

Maxeiner L. *Dual-based Methods for Distributed Optimization of Interconnected Systems*. Shaker Verlag; 2021.

Lunze J. *Regelungstechnik 2: Mehrgrößensysteme, digitale Regelung*. Springer-Verlag; 2014.

Bezanson J, Edelman A, Karpinski S, Shah V. Julia: A fresh approach to numerical computing. *SIAM Rev.* 2017;59(1):65-98.

Lubin M, Dowson O, Garcia JD, Huchette J, Legat B, Vielma JP. JuMP 1.0: recent improvements to a modeling language for mathematical optimization. *Math Program Comput.* 2023;1-9.

Gurobi Optimization L. Gurobi Optimizer Reference Manual. 2023. Available from: <https://www.gurobi.com>.

Wächter A, Biegler L. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math Program.* 2006;106(1):25-57.

Vujanic R, Esfahani P, Goulart P, Mariéthoz S, Morari M. A decomposition method for large scale MILPs, with performance guarantees and a power system application. *Automatica.* 2016;67:144-56.

Wenzel S, Riedl F, Engell S. An efficient hierarchical market-like coordination algorithm for coupled production systems based on quadratic approximation. *Comput Chem Eng.* 2020;134(8):106704.