Optimization and Engineering (2020) 21:761–802
https://doi.org/10.1007/s11081-020-09499-7

RESEARCH ARTICLE

Comparison of dual based optimization methods for distributed trajectory optimization of coupled semi-batch processes

Lukas Samuel Maxeiner[1] · **Sebastian Engell**[1]

Received: 3 September 2019 / Revised: 23 March 2020 / Accepted: 24 March 2020 /
Published online: 24 April 2020

Abstract

The physical and virtual connectivity of systems via flows of energy, material, information, etc., steadily increases. This paper deals with systems of sub-systems that are connected by networks of shared resources that have to be balanced. For the optimal operation of the overall system, the couplings between the sub-systems must be taken into account, and the overall optimum will usually deviate from the local optima of the sub-systems. However, for reasons, such as problem size, confidentiality, resilience to breakdowns, or generally when dealing with autonomous systems, monolithic optimization is often infeasible. In this contribution, iterative distributed optimization methods based on dual decomposition where the values of the objective functions of the different sub-systems do not have to be shared are investigated. We consider connected dynamic systems that share resources. This situation arises for continuous processes in transient conditions between different steady states and in inherently discontinuous processes, such as batch production processes. This problem is challenging since small changes during the iterations towards the satisfaction of the overarching constraints can lead to significant changes in the arc structures of the optimal solutions for the sub-systems. Moreover, meeting endpoint constraints at free final times complicates the problem. We propose a solution strategy for coupled semi-batch processes and compare different numerical approaches, the sub-gradient method, ADMM, and ALADIN, and show that convexification of the sub-systems around feasible points increases the speed of convergence while using second-order information does not necessarily do so. Since sharing of resources has an influence on whether trajectory dependent terminal constraints can be satisfied, we propose a heuristic for the computation of free final times of the sub-systems that allows the dynamic sub-processes to meet the constraints. For the example of several semi-batch reactors which are coupled via a bound on the total feed flow rate, we demonstrate that the distributed methods converge to (local) optima and

The project leading to this publication has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 723575 (CoPro, spire2030.eu/copro) in the framework of the SPIRE PPP.

Extended author information available on the last page of the article

highlight the strengths and the weaknesses of the different distributed optimization methods.

Keywords Distributed optimization · Dual methods · Dual decomposition · Trajectory optimization · Semi-batch reactors · ADMM · ALADIN · Sharing of resources

1 Introduction

The physical and virtual connectivity of systems steadily increases in order to increase throughput, performance, and safety (Engell et al. 2016). The resources that connect the different sub-systems can be flows of energy, flows of material, flows of information, or more specific constraints, as quantitative restrictions for hazardous substances, or certificates, as e.g. a given amount of CO_2-certificates that cannot be exceeded (Rius-Sorolla et al. 2020).

Of special interest are systems where the resources are not only shared bilaterally but among several sub-systems. Such system arise in the process industry, where several units are connected via networks of energy and material (Jose and Ungar 2000; Wenzel et al. 2017), in power systems, where the economic dispatch between different electricity generation facilities has to be coordinated to satisfy the network demands (Wang et al. 1995; Zhang et al. 2013) or where demand response of small loads such as residential smart appliances are integrated into the network (Gatsis and Giannakis 2013; Safdarian et al. 2014), in information technology, for instance when coordinating bandwidth between different agents (Hasan et al. 2014; Koutsopoulos and Iosifidis 2010), or when coordinating different autonomous systems such as robots or vehicles to maintain connectivity constraints (Cortés et al. 2004; Galceran and Carreras 2013).

The challenge with such systems is that often a monolithic optimization is not possible. The optimization of a whole chemical site with different ownership of the plants, for instance, requires a limited flow of information to maintain the confidentiality of business data or technical information such as prices, production targets, capacities, efficiencies, etc. Another reason is that in many cases the autonomy and decision-making power should remain with the respective sub-systems, which is not the case if a third party explicitly determines the operating conditions and the coordination of the resources between the sub-systems. If the scale of the resulting optimization problem is large, transparency of the results can be limited, as root causes are difficult to trace across sub-system boundaries (Tang et al. 2018). Furthermore, the implementation of monolithic solutions for resource allocation requires the separation of the models from the sub-systems, which constitutes an additional source of error if the models have to be maintained in different locations. Lastly, if the optimization is carried out for control purposes, solutions that reflect the modularity of the system and provide redundancy are often more desirable than a monolithic optimization (Camponogara

et al. 2002; Cheng et al. 2007; Christofides et al. 2013; Farokhi et al. 2014; Maestre and Negenborn 2014; Scattolini 2009; Van Parys and Pipeleers 2017).

As a consequence, interconnected systems are often optimized on two levels: On the sub-system level, each sub-system tries to maximize its performance according to given boundary conditions, and on the system-wide or coordinator level, where the optimal allocation of shared resources is performed (Mesarovic et al. 1970). Such bilevel problems can be solved via iterative distributed optimization approaches, i.e. primal-based and dual-based decomposition methods which are compared for instance in Conejo et al. (2006), Palomar and Chiang (2006, 2007). The most suitable methods to tackle the aforementioned challenges, in particular the confidentiality issue, are dual-based decomposition methods where the coordinator broadcasts information such as prices to the different sub-systems, whereupon these respond with their estimated usage of the shared resources, and the coordinator iteratively adapts the prices until the resource balances are met.

While methods for the distributed optimization of steady state problems have been thoroughly investigated, dynamic systems have mostly been considered in the context of distributed model predictive control. Most of the work in this area considers the control of continuous processes with linear dynamics, cf. the survey in Negenborn and Maestre (2014). To the authors' knowledge, little work has been done to compare different distributed optimization methods for coordinating resources among distributed dynamic systems.

The application that we discuss here is the dynamic resource allocation for coupled chemical reactors that are operated in semi-batch mode, i.e., some substances are filled into the reactor at the start while others are dosed during the batch run. At the end of the batch run, the reaction mixture which contains the desired product is withdrawn. We focus on how to share a limited amount of a resource, in this case, the feed flow to the reactors between the semi-batch reactors, in order to determine an optimal overall operation for given initial conditions and a given production schedule. This is a common challenge in the process industry when large quantities of products are produced in multi-product multi-batch plants and suitable production sequences have to be defined and executed (Nie et al. 2015).

1.1 Contribution

The goal of this contribution is to provide a distributed optimization strategy for semi-batch processes with end-point constraints that are subject to overarching constraints and to compare different algorithms for the distributed allocation of shared resources between dynamic systems. We restrict ourselves to dual based methods that do not require the knowledge of the objective values of the sub-systems. Additionally to the sub-gradient method and ADMM, we apply the augmented Lagrangian based algorithm for distributed non-convex optimization (ALADIN) and adapt it so that it can also handle overarching inequality constraints as they arise for systems that share finite amounts of resources.

The challenge in solving these type of problems is that significant change in the arc structure, i.e. changes in the active sets of the constraints of the sub-systems for small

changes in the dual variables occur during the iterations of the dual-based distributed optimization methods.

We present a heuristic approach for the solution of such resource allocation problems for dynamic systems with terminal constraints and free final times, such that the different sub-systems meet the constraints in the presence of overarching resource constraints.

1.2 Notation

Sub-system related vectors $x_i \in \mathbb{R}^{n_i},\ i \in 1, \dots, N$, are stacked into one large vector $x = [x_1^T, x_2^T, \dots, x_N^T]^T \in \mathbb{R}^n$ of variables on the system level. The subscript i is used as indicator for the different sub-systems throughout the paper. The superscript (k) indicates the current iteration k. To index the j-th element of a vector x, the notation $x[j]$ is used.

The Eucledian norm of a vector x is indicated by $\|x\|_2$. The infinity norm of a vector $x \in \mathbb{R}^m$ is defined by $\|x\|_\infty = \max\{|x[1]|, \dots, |x[m]|\}$, where $|\cdot|$ is the absolute value of a scalar and the max operator followed by $\{\cdot\}$ indicates the element-wise maximum of the set.

2 Theoretical background

First, a general mathematical problem formulation for a single dynamic system is presented, which is then extended to several systems that share a common resource. The extended problem is put into a standard form and algorithms for the distributed solution of the problem are introduced.

2.1 Problem formulation for a sub-system trajectory

A general dynamic or trajectory optimization problem over a fixed time interval $t \in [t_0, t_f]$ can be written as follows:

$$\min_{\substack{u(t) \\ \forall t \in [t_0, t_f]}} \quad \left(Y\big(\chi(t_f)\big) + \int_{t_0}^{t_f} \Theta(\chi(t), u(t), t) dt \right), \tag{1a}$$

$$\text{s.t.} \quad \dot{\chi}(t) = F(\chi(t), u(t), t), \quad t \in [t_0, t_f], \tag{1b}$$

$$\chi(t_0) = \chi_0, \tag{1c}$$

$$P(\chi(t), u(t), t) \leq 0, \quad t \in [t_0, t_f], \tag{1d}$$

$$T\big(\chi(t_f)\big) \leq 0. \tag{1e}$$

In dynamic optimization, there is a distinction between state variables $\chi(t)$ and inputs $u(t)$ to the system. From the inputs $u(t)$ and the initial condition, $\chi(t_0) = \chi_0$, the state variables can be computed using the model equation Eq. 1b. Thus $u(t)$ are the degree of freedom, while $\chi(t)$ are the dependent variables. The goal is to find the best inputs such that the constraints are satisfied and some performance measure of the resulting trajectory is maximized or minimized.

The objective is given in the Bolza form which consists of a scalar performance measure at the end of the horizon, Y, and an integral part that tracks some scalar performance measure over the whole path of the trajectory, Θ. Similar to the objective, the constraints are defined as terminal constraints (T_i) and path constraints (P_i) (Sargent 2000).

2.2 Problem formulation for multiple trajectories with shared inputs

The problem of interest in this paper is to optimize N trajectories for N sub-systems that share resources and start as well as end their operation possibly at different times. This can be formulated as the following dynamic optimization problem:

$$\min_{\substack{u_i(t) \\ \forall(t,\ i)\ \in\ ([t_{min}, t_{max}],\ \{1,\dots,N\})}} \quad \sum_{i\in\{1,\dots,N\}} \left(Y_i\big(\chi_i(t_{f,i})\big) + \int_{t_{0,i}}^{t_{f,i}} \Theta_i\big(\chi_i(t), u_i(t), t\big)dt \right), \tag{2a}$$

$$\text{s.t.} \quad \sum_{i\in\{1,\dots,N\}} u_i(t) \le u_{shared,max}(t), \quad t \in [t_{min}, t_{max}], \tag{2b}$$

$$\dot{\chi}_i(t) = F_i\big(\chi_i(t), u_i(t), t\big), \quad t \in [t_{0,i}, t_{f,i}],\ \forall i \in \{1,\dots,N\}, \tag{2c}$$

$$\chi_i(t_{0,i}) = \chi_{0,i}, \quad \forall i \in \{1,\dots,N\}, \tag{2d}$$

$$P_i\big(\chi_i(t), u_i(t), t\big) \le 0, \quad t \in [t_{0,i}, t_{f,i}],\ \forall i \in \{1,\dots,N\}, \tag{2e}$$

$$T_i\big(\chi_i(t_{f,i})\big) \le 0, \quad \forall i \in \{1,\dots,N\}, \tag{2f}$$

$$u_i(t) = 0, \quad t \notin [t_{0,i}, t_{f,i}], \quad \forall i \in \{1,\dots,N\}. \tag{2g}$$

The considered time interval is given by $t_{min} = \min\{t_{0,1},\dots,t_{0,N}\}$ and $t_{max} = \max\{t_{f,1},\dots,t_{f,N}\}$. The objective is to minimize the sum of the individual objectives. The variables χ_i and u_i belong to sub-system i exclusively, can only be manipulated by the respective sub-system, and, except for coupling via the overarching constraints Eq. 2b, have no impact on the other sub-systems. Due to the different

starting and final times of the trajectories, when the trajectory of sub-system i is not active, its use of the resource is fixed at 0 via Eq. 2g.

2.3 Numerical solution methods for trajectory optimization

The problems given in Eqs. 1 and 2 can be solved using different approaches: Direct optimization methods, methods based on Pontryagin's minimum principle, and methods that are based on the Hamilton–Jacobi–Bellman equations (Bellman 1957; Bertsekas 1995; Pontryagin 2018; von Stryk and Bulirsch 1992). Depending on the selected method, the level of discretization can be chosen: All quantities can be considered infinite-dimensional in time, only the inputs can be discretized, or additionally to the inputs also the states can be fully discretized. If the inputs are discretized, they are usually considered to be piece-wise constant or piece-wise linear within the discretization elements. An overview of the solution methods as well as discretization levels is given in Betts (1998) and Srinivasan et al. (2003).

While there are also other efficient methods, as, e.g., parsimonious input parametrization (Rodrigues and Bonvin 2019), direct methods are used here, since they are best suited to handle the overarching constraints Eq. 2b as well as to get reliable numerical solutions (Srinivasan et al. 2003). When the inputs are discretized into equidistant intervals of duration Δt, there are three options to solve such a problem using the direct method: The first option is control vector parametrization, where the states remain continuously defined for every point in time and are determined by integration. The degrees of freedom for the optimization are the values of the inputs. The second is the simultaneous approach, where, additionally to the inputs, also the states are fully discretized such that a sparse large non-linear program (NLP) results (Biegler 2007). Since also the states are degrees of freedom, only when the NLP has been solved, the resulting trajectory satisfies the model equations. On the other hand, this method often shows to be more robust. The third option is multiple shooting, in which the time is divided into several intervals, where in each interval control vector parametrization is used to determine the solution. Matching of the states at the ends of the intervals is forced by an additional boundary condition (Bock and Plitt 1985). In this contribution, control vector parametrization is applied and a constant stepsize 4th-order Runge-Kutta method is used for integration, because this renders all derivatives, of objective as well as of the constraints, dependent only on the inputs.

2.4 Discretized problem formulation

The discretization of the problem in Eq. 1 can be done as described in the previous subsection, however, for the problem formulation with multiple trajectories and shared inputs, synchronization of time between the different trajectories needs to be assured. Only if the starting and ending times are on the same time grid as the discretization of the inputs, the overarching constraints can be enforced at every point in time. The starting and ending times are thus expressed as multiples of the shared minimum discretization duration Δt via the following relationship:

$$t_{i,0} = N_{i,0}\ \Delta t, \tag{3}$$

$$t_{i,f} = N_{i,f}\ \Delta t. \tag{4}$$

For each sub-system i, the sets of points $\Psi_i = \{N_{i,0}, \dots, N_{i,f} - 1\}$ are defined as counterparts to the continuous time intervals $[t_{0,i}, t_{f,i}]$.

Similar to the continuous case, N_{min} and N_{max} are defined as the minimum and maximum over all sub-systems i of $N_{i,0}$ and $N_{i,f}$. This leads to the following problem formulation:

$$\min_{\substack{\chi_{i,p},\ u_{i,p},\\ \forall (p,\ i)\ \in\ (\{N_{min}, \dots, N_{max}\},\ \{1, \dots, N\})}} \sum_{i \in \{1,\dots,N\}} \left(Y_i(\chi_{i,N_{i,f}}) + \sum_{p \in \Psi_i} \Theta_i\big(\chi_{i,p}, u_{i,p}, \Delta t, p\big) \right), \tag{5a}$$

$$\text{s.t.} \quad \sum_{i \in \{1,\dots,N\}} u_{i,p} \le u_{shared,max,p}, \quad \forall p \in \{N_{min}, \dots, N_{max}\}, \tag{5b}$$

$$\tilde{F}_i\big(\chi_{i,p}, \chi_{i,p+1}, u_{i,p}, \Delta t, p\big) = 0, \quad \forall p \in \Psi_i, i \in \{1, \dots, N\}, \tag{5c}$$

$$\chi_{i,N_{i,0}} = \chi_{0,i}, \quad \forall i \in \{1, \dots, N\}, \tag{5d}$$

$$P_i\big(\chi_{i,p}, u_{i,p}, \Delta t, p\big) \le 0, \quad \forall p \in \Psi_i, i \in \{1, \dots, N\}, \tag{5e}$$

$$T_i\Big(\chi_{i,N_{i,f}}\Big) \le 0, \quad \forall i \in \{1, \dots, N\}, \tag{5f}$$

$$u_{i,p} = 0, \quad \forall p \notin \Psi_i, i \in \{1, \dots, N\}. \tag{5g}$$

In this formulation, p is the discrete time index. Note that in this case the models $\tilde{F}_i$ are defined implicitly, connecting the old and the new states. This is used to express numerical integration or full discretization. Furthermore, it should be noted that the path constraints P_i are only defined at the grid points.

The resulting optimization problem is non-convex, which is in general a $\mathcal{NP}$-hard problem that can have multiple local minima (Esposito and Floudas 2000; Papamichail and Adjiman 2002).

2.5 Problem formulation in the standard form of distributed optimization

The problem of optimizing trajectories with shared resources across system boundaries from Eq. 5 can be written in the standard form of a general sharing problem, cf. Boyd et al. (2010),

$$\min_{x_i,\ \forall i\in\{1,\dots,N\}} \quad \sum_{i\in\{1,\dots,N\}} f_i(x_i), \tag{6a}$$

$$\text{s.t.} \quad \sum_{i\in\{1,\dots,N\}} A_i x_i \leq b, \tag{6b}$$

$$g_i(x_i) \leq 0, \quad i \in \{1,\dots,N\}. \tag{6c}$$

The variables x_i comprise the inputs and the states of the sub-systems, $dim(x_i) = dim(\chi_i) + dim(u_i)$. The inputs of the trajectory optimization problem described in the previous subsection are given by the linear mapping $u_i = A_i x_i$, with $u_i = [u_{i,N_{i,0}}, u_{i,N_{i,1}}, \dots, u_{i,N_{i,f}-1}]^T$. The state variables are given by $\chi_i = B_i x_i$, with $\chi_i = [\chi_{i,N_{i,0}}, \chi_{i,N_{i,1}}, \dots, \chi_{i,N_{i,f}}]^T$. The initial conditions, model equations, and system-specific constraints are described by the n_{g_i}-dimensional inequality constraint function g_i. The dimension of the overarching constraints is m, i.e., $dim(b) = dim(A_i x_i) = dim(u_i) = dim(u_{shared,max}) = m$.

2.6 Necessary conditions of optimality for distributed optimization

For this problem in standard form, the Lagrangian of the problem is given by Bertsekas (1999):

$$\begin{aligned}
\mathscr{L}(x,\lambda,\mu) &:= \sum_{i\in\{1,\dots,N\}} \left(f_i(x_i)\right) + \lambda^T \left(\sum_{i\in\{1,\dots,N\}} A_i x_i - b \right) + \sum_{i\in\{1,\dots,N\}} \left(\mu_i^T (g_i(x_i))\right), \\
&= \sum_{i\in\{1,\dots,N\}} \left(f_i(x_i) + \lambda^T A_i x_i - \frac{1}{N}\lambda^T b + \mu_i^T (g_i(x_i)) \right), \\
&= \sum_{i\in\{1,\dots,N\}} \mathscr{L}_i(x_i,\lambda,\mu_i),
\end{aligned} \tag{7}$$

where λ are the Lagrange multipliers corresponding to the overarching constraints in Eq. 6b and μ_i are the Lagrange multipliers for the sub-system specific constraints in Eq. 6c. Using the Lagrangian, the first-order necessary conditions of optimality (Karush-Kuhn-Tucker conditions) can be expressed as:

$$\nabla_{x_i} \mathscr{L}_i(x_i,\lambda,\mu_i) = 0, \quad \forall i \in \{1,\dots,N\}, \tag{8a}$$

$$g_i(x_i) \leq 0, \quad \forall i \in \{1,\dots,N\}, \tag{8b}$$

$$\mu_i \geq 0, \quad \forall i \in \{1, \ldots, N\}, \tag{8c}$$

$$\sum_{i=1}^{N} A_i x_i - b \leq 0, \tag{8d}$$

$$\lambda \geq 0. \tag{8e}$$

The interesting property of these conditions is that Eqs. 8a–8c can be evaluated independently for each sub-system i and only Eqs. 8d and 8e require coordination between the different sub-systems.

2.7 Distributed solution algorithms based on the dual problem

While the Eqs. 8 can be solved monolithically via state of the art solvers, in this contribution methods that exploit the distributed structure of the problem are investigated.

We focus on hierarchical methods, where all sub-system specific decisions are taken in a distributed fashion and only on the coordination layer, the satisfaction of the overarching constraints Eqs. 8d and 8e is enforced. These methods are also known as dual methods, which make use of the dual variables or Lagrange multipliers. In our case, the dual variables of interest are the ones corresponding to the overarching constraints, i.e. λ.

Using the solution to Eqs. 8a–8c, written as $\inf_{x_i,\ \mu_i \geq 0} \mathscr{L}_i(x_i, \lambda, \mu_i)$, the dual function can then be defined:

$$d(\lambda) = \inf_{x,\ \mu \geq 0} \mathscr{L}(x, \lambda, \mu) = \sum_{i \in \{1,\ldots,N\}} \inf_{x_i,\ \mu_i \geq 0} \mathscr{L}_i(x_i, \lambda, \mu_i). \tag{9}$$

Using this dual function of λ, the optimality condition can be expressed as finding the maximum of $d(\lambda)$ with $\lambda \geq 0$, which is called the dual problem:

$$\max_{\lambda \geq 0} d(\lambda). \tag{10}$$

Due to the infimum in Eq. 9, the dual function is in general not known explicitly. However, according to Danskin's theorem (Bertsekas 1999), a sub-gradient can be determined at λ via:

$$\partial d(\lambda) = \partial \sum_{i \in \{1,\ldots,N\}} \inf_{x_i,\ \mu_i \geq 0} \mathscr{L}_i(x_i, \lambda, \mu_i) = \sum_{i \in \{1,\ldots,N\}} A_i x_i(\lambda) - b. \tag{11}$$

In this contribution, we compare iterative methods for the maximization of the dual which do not require the explicit knowledge of the value of the dual function and thus of the different individual objectives. As measures of convergence, we define two criteria. The primal feasibility, Φ_{Primal}, is a measure of the satisfaction of the overarching constraints of the original problem. At the same time, due to Eq. 11 and

the fact that the dual is always concave, it is also a measure of the vanishing of the gradient (Boyd and Vandenberghe 2004).

$$\boldsymbol{\Phi}_{Primal}[j] = \max\left\{0, \left(\sum_{i\in\{1,\dots,N\}} A_i x_i(\lambda) - b\right)[j]\right\}, \quad j \in \{1,\dots,m\}. \tag{12}$$

Here we highlight that x_i is a function of λ. If all elements of $\boldsymbol{\Phi}_{Primal}$ are equal to 0, the overarching constraints are satisfied. Additionally to the primal feasibility, which measures the satisfaction of the constraints, also a measure of convergence is required, in order to prevent termination when a solution is primal feasible, but not optimal yet. The dual feasibility, $\boldsymbol{\Phi}_{Dual}$, can be interpreted as the satisfaction of the optimality criterion for the dual problem, i.e., the gradient approaching the 0 vector. Thus we define the dual feasibility as the finite difference approximation of the gradient of the dual function.

$$\boldsymbol{\Phi}_{Dual}[j] = \frac{|\lambda^+[j] - \lambda[j]|}{\alpha[j]}, \tag{13}$$

This second feasibility criterion is a measure of how far the solution can deviate from active overarching constraints and is essential for inequality constrained problems, since primal feasibility can always be achieved by sufficiently large Lagrange multipliers. This ensures that the solution not only satisfies $\sum_{i=1}^{N} A_i x_i^{(k)} - b \leq \epsilon$ but also $\sum_{i=1}^{N} A_i x_i^{(k)}[j] - b[j] \geq -\epsilon$ for all active overarching constraints j. Only if a solution is primal and dual feasible, a saddle point to the Lagrangian that satisfies the conditions of optimality is found.

Since we can optimize numerically only with a certain numerical error, we define a set of x^*, λ^*, and μ^* to be optimal if the following holds:

$$\{x^*, \lambda^*, \mu^*\} = \{x, \lambda, \mu \mid \left\|\boldsymbol{\Phi}_{Primal}\right\|_\infty \leq \epsilon_{Feas,Primal} \wedge \left\|\boldsymbol{\Phi}_{Dual}\right\|_\infty \leq \epsilon_{Feas,Dual}\},$$

where $\epsilon_{Feas,Primal}$ and $\epsilon_{Feas,Dual}$ are the desired numerical tolerances and x_i minimizes the objective of sub-system i.

2.7.1 Sub-gradient method

The simplest method for the maximization of the dual is to follow the direction of the steepest ascent, i.e., to use the direction of the sub-gradient (Shor 2012). Here the challenge is the selection of a suitable stepsize. Since the dual function may be a non-smooth function, which depends on the solution structure of the different trajectories, the stepsize selection criteria for non-smooth optimization derived in Nesterov (2004, p. 142) should be satisfied.

$$\alpha^k > 0, \tag{14a}$$

$$\alpha^k \to 0, \tag{14b}$$

$$\sum_{k=0}^{\infty} \alpha^k = \infty. \tag{14c}$$

Since the impact of small changes in the dual variables is not the same for all variables $\lambda[j]$, $j \in \{1, \ldots, m\}$, a scheme is needed that adapts the stepsizes individually.

While there exist methods to evaluate the optimal stepsize at the current point using the Lipschitz constant, as explained in Bertsekas and Tsitsiklis (1989) and Kozma et al. (2014), determining the constant is difficult since the dual cannot be evaluated explicitly. Furthermore, since this constant is not valid globally, the stepsizes either have to be adapted during the maximization of the dual or be chosen conservatively enough to be valid throughout the domain of the dual function.

The sub-gradient method can be considered as alternating local optimization of the sub-systems and adaptation of the Lagrange multipliers on the coordination layer. We propose to decrease the stepsize of a specific overarching constraint every time the sign of the corresponding element of $\sum_{i=1}^{N} A_i x_i(\lambda) - b$ changes. This is equivalent to the inequality constraint becoming active or inactive respectively. Specifically the following adaptation of α is used (cf. Algorithm 1, line 8):

$$\alpha^{(k)}[j] = \begin{cases} \gamma_{Decrease}\, \alpha^{(k-1)}[j], & \text{if, } \left(\sum_{i=1}^{N} A_i x_i^{(k)} - b\right)[j] \left(\sum_{i=1}^{N} A_i x_i^{(k-1)} - b\right)[j] \leq 0, \\ \alpha^{(k-1)}[j], & \text{otherwise.} \end{cases} \tag{15}$$

If one of the stepsizes is too large and it has an influence on the Lagrange multiplier (note that if a constraint is inactive, the multiplier is fixed at 0), then this leads inevitably to a diminishing of the stepsize in this direction. The factor β in Algorithm 1 prevents the stepsize from decreasing too fast due to oscillating responses while maintaining the stepsize as large as possible if consecutive steps have the same direction.

Algorithm 1 Sub-gradient (SG) method with adaptive stepsize α

Require: $\lambda^{(0)}$, $\alpha^{(0)}$, β, $\varepsilon_{Feas,Primal}$, $\varepsilon_{Feas,Primal}$
1: $k \leftarrow 0$
2: **repeat**
3: $\quad k \leftarrow k+1$
4: $\quad$ **for all** $i = 1, \ldots, N$ **do**
5: $\quad\quad A_i x_i^{(k)} \leftarrow A_i \arg\min_{x_i,\, \mu_i \geq 0} \mathscr{L}_i \left(x_i, \lambda^{(k-1)}, \mu_i\right)$
6: $\quad$ **end for**
7: $\quad \lambda^{(k)} \leftarrow \max\left\{0, \lambda^{(k-1)} + \alpha^{(k)} \left(\beta \left(\sum_{i=1}^{N} A_i x_i^{(k)} - b\right) + (1-\beta) \left(\sum_{i=1}^{N} A_i x_i^{(k-1)} - b\right)\right)\right\}$
8: $\quad \alpha^{(k)} \leftarrow$ Eq. 15
9: **until** $\Phi_{Primal} \leq \varepsilon_{Feas,Primal} \wedge \Phi_{Dual} \leq \varepsilon_{Feas,Dual}$
10: **return** $\lambda^{(k)}$

Other possible selections of α can be found in Nesterov (2004). Regardless of the selection of the stepsize, the provable convergence rate for strictly convex problems is at best $\mathcal{O}(1/k)$.

2.7.2 ADMM

A more robust method is the *alternating direction method of multipliers* (ADMM) Boyd et al. (2010). It uses the augmented Lagrangian:

$$\mathscr{L}_{\rho,i}(x_i, \lambda, \mu_i, z_i) = \mathscr{L}_i(x_i, \lambda, \mu_i) + \frac{\rho}{2}\left\|A_i x_i - z_i\right\|_2^2. \tag{16}$$

Additionally to the linear penalty term, the deviation from a feasible use of the shared resources z_i is penalized quadratically. Essentially, the penalty terms convexify the problems around points that satisfy the overarching constraints, which accelerates the initial convergence. The variables z_i are determined on the coordinator level and are a projection of the current responses of the different sub-systems onto the feasible region. The stepsize α for the update of the Lagrange multipliers is $\frac{\rho}{N}$ in the case of ADMM. However, since additionally to the prices also the z_i determine the response of the systems, the dual feasibility is redefined as:

$$\boldsymbol{\Phi}_{Dual}[j] = \rho\left(\sum_{i=1}^{N}\left|A_i x_i^{(k)}[j] - z_i^{(k)}[j]\right|\right). \tag{17}$$

The convergence rate for convex problems is also $\mathcal{O}(1/k)$ (Hong and Luo 2017; Kozma et al. 2014).

Algorithm 2 Alternating direction method of multipliers (ADMM) for equality constrained problems as shown in [9, p. 59]

Require: $\lambda^{(0)}, \rho, \varepsilon_{Feas,Primal}, \varepsilon_{Feas,Dual}$
$k \leftarrow 0$
$z_i^{(k)} \leftarrow 0$
repeat
  $k \leftarrow k+1$
  for all $i = 1, \ldots, N$ **do**
    $A_i x_i^{(k)} \leftarrow A_i \arg\min_{x_i,\ \mu_i \geq 0} \mathscr{L}_{\rho,i}\left(x_i, \lambda^{(k-1)}, \mu_i, z_i^{(k-1)}\right)$
  end for
  $\lambda^{(k)} \leftarrow \lambda^{(k-1)} + \frac{\rho}{N}\left(\sum_{i=1}^{N} A_i x_i^{(k)} - b\right)$
  $z_i^{(k)} \leftarrow A_i x_i^{(k)} - \frac{1}{N}\left(\sum_{i=1}^{N} A_i x_i^{(k)} - b\right)$
until $\Phi_{Primal} \leq \varepsilon_{Feas,Primal} \wedge \Phi_{Dual} \leq \varepsilon_{Feas,Dual}$
return $\lambda^{(k)}$

Algorithm 3 Alternating direction method of multipliers (ADMM) adapted for inequality constrained problems

Require: $\lambda^{(0)}$, $\rho^{(0)}$, $\varepsilon_{Feas,Primal}$, $\varepsilon_{Feas,Dual}$, $\tau_{Increase}$, $\tau_{Decrease}$, δ

1: $k \leftarrow 0$
2: $z_i^{(k)} \leftarrow 0$
3: **repeat**
4: $\quad k \leftarrow k+1$
5: $\quad$ **for all** $i = 1, \ldots, N$ **do**
6: $\quad\quad A_i x_i^{(k)} \leftarrow A_i \ \arg\min_{x_i,\ \mu_i \geq 0} \mathscr{L}_{\rho^{(k-1)},i}\left(x_i, \lambda^{(k-1)}, \mu_i, z_i^{(k-1)}\right)$
7: $\quad$ **end for**
8: $\quad \lambda^{(k)} \leftarrow \max\left(0, \lambda^{(k-1)} + \frac{1}{N}\mathrm{diag}(\rho^{(k-1)})\left(\sum_{i=1}^{N} A_i x_i^{(k)} - b\right)\right)$
9: $\quad$ Solve coordinator level QP:

$$\min_{z_i,} \sum_{i \in \{1,\ldots,N\}} \left\| A_i x_i^{(k)} - z_i \right\|_2^2 \tag{18a}$$

$$\text{s.t.} \sum_{i \in \{1,\ldots,N\}} z_i \leq b, \tag{18b}$$

10: $\quad \rho^{(k)} \leftarrow$ Eq. 19
11: **until** $\Phi_{Primal} \leq \varepsilon_{Feas,Primal} \wedge \Phi_{Dual} \leq \varepsilon_{Feas,Dual}$
12: **return** $\lambda^{(k)}$

In Algorithm 2 the unscaled version of ADMM for equality constrained sharing problems is shown, cf. Boyd et al. (2010, p. 59). When adapting the algorithm to the inequality constraint problem considered here, the dual variables need to satisfy $\lambda \geq 0$ and the z-variables need to be adjusted depending on whether the overarching constraints are active or not. When they are active, the same update as in Algorithm 2 applies, however, if the constraints are not active, the references z_i are based on the previous solutions of the sub-systems. This penalty is required since otherwise only some variables are quadratically penalized possibly leading to cyclic solution changes in subsequent iterations. We present ADMM including a new and efficient update step to compute the z-variables for inequality constrained problems in Algorithm 3. To improve convergence, different and variable penalty parameters ρ are used for each constraint. The penalty parameters are adapted at step 10 of Algorithm 3 using the scheme in Wang and Liao (2001):

$$\rho^{(k)}[j] = \begin{cases} \tau_{Increase}\ \rho^{(k-1)}[j], & \text{if } \boldsymbol{\Phi}_{Primal}^{(k)}[j] \geq \delta\ \boldsymbol{\Phi}_{Dual}^{(k)}[j], \\ \tau_{Decrease}\ \rho^{(k-1)}[j], & \text{if } \delta\ \boldsymbol{\Phi}_{Primal}^{(k)}[j] \leq \boldsymbol{\Phi}_{Dual}^{(k)}[j], \\ \rho^{(k-1)}[j], & \text{otherwise.} \end{cases} \tag{19}$$

The factor δ is the maximally allowed difference between primal and dual feasibility before the penalty parameter is adapted to rebalance the proportion. The parameters $\tau_{Increase} > 1$ and $\tau_{Decrease} < 1$ adjust the penalty parameter ρ if necessary. The update ensures that primal and dual feasibility are kept in balance or, in other terms, that far away from the optimum large changes in λ are made. Close to the optimum,

the changes are reduced and additionally the quadratic penalization is decreased, to ensure that the solution without the quadratic penalty also satisfies the overarching constraints.

2.7.3 ALADIN

Another method that uses the augmented Lagrangian is the *augmented Lagrangian based algorithm for distributed non-convex optimization* in Houska et al. (2016). Different to ADMM, here the reference values for all variables of sub-system i are penalized for deviating from the reference z_i:

$$\mathscr{L}_{\rho,i}(x_i, \lambda, \mu_i, z_i) = \mathscr{L}_i(x_i, \lambda, \mu_i) + \frac{\rho}{2}\left\|x_i - z_i\right\|_2^2. \tag{20}$$

Additionally to reporting the consumption of the shared resources, the sub-systems also report their derivatives of the objective and of the active constraints with respect to the local decision variables to the coordination layer in each iteration k. The Hessian and gradient approximations are then calculated using the constraint Jacobian information $\mathscr{C}_i^{(k)}$ ($\mathscr{C}_i^{Active(k)}$) of the (active) constraints from Eq. 6c:

$$\mathscr{C}_i^{Active(k)}[l,:] = \begin{cases} \mathscr{C}_i^{(k)}[l,:], & \text{if } g_i(x_i^{(k)})[l] = 0, \\ 0, & \text{otherwise,} \end{cases} \quad \text{for } l \in \{1,\dots,n_{g_i}\}, \forall i \in \{1,\dots,N\}, \tag{21}$$

where $\mathscr{C}_i^{(k)} = \nabla_{x_i} g_i(x_i)|_{x_i = x_i^{(k)}}$. The modified gradient and Hessian approximation are:

$$\mathscr{g}_i^{(k)} = \nabla_{x_i} f_i(x_i)\big|_{x_i = x_i^{(k)}} + \left(\mathscr{C}_i^{Active(k)} - \mathscr{C}_i^{(k)}\right)^T \mu_i^{(k)}, \tag{22}$$

$$\mathscr{H}_i^{(k)} \approx \nabla_{x_i}^2 \left(f_i(x_i) + \mu_i^T g_i(x_i)\right)\Big|_{x_i = x_i^{(k)},\ \mu_i = \mu_i^{(k)}}. \tag{23}$$

With this information of the sub-systems, instead of doing straight projections onto the feasible set, prices and reference values (z_i) are determined via a quadratic program that approximates the objective functions and active constraints of the sub-systems. The algorithm can be seen as a combination of sequential quadratic programming (SQP) and ADMM. The benefit of using more information from the different sub-systems on the coordinator level is in general a faster convergence. In Houska et al. (2016) it is shown that in theory super-linear to quadratic convergence rates are possible.

Algorithm 4 Augmented Lagrangian based algorithm for distributed non-convex optimization (ALADIN) [24]

Require: $\lambda^{(0)}$, ρ, α_1, α_2, α_3, ν, $\varepsilon_{Feas,Primal}$, $\varepsilon_{Feas,Dual}$

$k \leftarrow 0$

$z_i^{(k)} \leftarrow 0$

repeat

  $k \leftarrow k+1$

  for all $i = 1, \ldots, N$ **do**

    $A_i x_i^{(k)},\ \mu_i^{(k)} \leftarrow A_i\ \arg\min_{x_i,\ \mu_i \geq 0} \mathscr{L}_{\rho,i}\left(x_i, \lambda^{(k-1)}, \mu_i, z_i^{(k-1)}\right)$

    $g_i^{(k)}, \mathscr{H}_i^{(k)}, \mathscr{C}_i^{Active,(k)} \leftarrow$ Evaluate derivatives at the optimum

  end for

  Solve coordinator level QP:

$$\min_{\Delta z_i,\ s} \sum_{i \in \{1,\ldots,N\}} \left(\Delta z_i^T \mathscr{H}_i^{(k)} \Delta z_i + g_i^{(k)T} \Delta z_i\right) + \lambda^{(k-1)T} s + \frac{\nu}{2} \|s\|_2^2 \tag{24a}$$

$$\text{s.t.} \sum_{i \in \{1,\ldots,N\}} A_i \left(x_i^{(k)} + \Delta z_i\right) = b + s, \quad |\lambda_{QP} \tag{24b}$$

$$\mathscr{C}_i^{Active(k)} \Delta z_i = 0. \quad i \in \{1, \ldots, N\} \tag{24c}$$

  $\lambda^{(k)} \leftarrow \lambda^{(k-1)} + \alpha_3 \left(\lambda_{QP} - \lambda^{(k-1)}\right)$

  $z_i^{(k)} \leftarrow z_i^{(k-1)} + \alpha_1 \left(x_i^{(k)} - z_i^{(k-1)}\right) + \alpha_2 \Delta z_i^{(k)}$

until $\Phi_{Primal} \leq \varepsilon_{Feas,Primal} \wedge \Phi_{Dual} \leq \varepsilon_{Feas,Dual}$

return $\lambda^{(k)}$

The update of the Lagrange multipliers λ is done differently compared to the previous two methods without sub-gradient information. Instead, the Lagrange multipliers from the overarching constraints λ_{QP} in the QP are used in the update step. The algorithm for equality constrained problems as proposed in Houska et al. (2016) is given in Algorithm 4. The dual feasibility is defined as follows:

$$\Phi_{Dual}[j] = \rho\left(\sum_{i=1}^{N} \left|x_i^{(k)}[j] - z_i^{(k)}[j]\right|\right). \tag{25}$$

The parameters $\alpha_i \in [0, 1]$ can be used to adjust the behaviour of the algorithm to match frequent changes of the active set. Houska et al. (2016) provide an additional scheme that utilizes the objective values of the sub-systems, based on which these parameters can be adapted in each iteration to guarantee the convergence to a local minimum. The scheme is not considered in this work, because essentially a monolithic optimization is carried out to determine the parameters.

Since trajectory optimization problems with overarching inequality constraints are considered, Eq. 24b is changed to an inequality constraint and the algorithm is modified accordingly. The solution to trajectory optimization problems consists of different arcs, which correspond to active constraints. Since these constraints ultimately act on the inputs, Eq. 24c fixes all Δz_i for the inputs in the QP.

Therefore, the equality constraint Eq. 24c is changed to inequality, such that the variables Δz_i are degrees of freedom. If the reference variables z_i resulting from the QP are infeasible, the Hessian of the sub-systems may become indefinite, which occurs mainly at the beginning of the scheme, when the QP approximations of the active constraints do not reflect the actual solution. Positive definiteness of $\mathscr{H}_i^{(k)}$ is required for ALADIN (Houska et al. 2016) and therefore we propose the following strategy to enforce this condition: The elements on the main diagonal are increased by κI, where I is the identity matrix. Since having high values for κ penalizes large changes in Δz_i, this value is decreased with the number of iterations until $\mathscr{H}_i^{(k)}$ becomes indefinite. Then κ remains fixed at this value, or is increased in subsequent iterations, if the Hessian is still indefinite.

Another challenge that arises from trajectory optimization is that small changes in the Lagrange multipliers of the overarching constraints λ can change the active set of the sub-systems. In order to be able to reach the correct active set, the stepsize of the algorithm possibly has to be infinitesimal. Thus, also the α_i are adapted in each iteration. Eq. 24b is modified to account for smaller values of α_1, such that the new reference variables z_i are always feasible according to the coordinator level QP. In Algorithm 5, the different steps of ALADIN, adjusted to inequality constrained problems, are given.

Algorithm 5 Augmented Lagrangian based algorithm for distributed non-convex optimization (ALADIN) [24] adapted here to inequality constrained problems

Require: $\lambda^{(0)}$, ρ, $\alpha_1^{(0)}$, $\alpha_2^{(0)}$, $\alpha_3^{(0)}$, ω, θ, ν, κ, $\varepsilon_{Feas,Primal}$, $\varepsilon_{Feas,Dual}$

$k \leftarrow 0$

$z_i^{(k)} \leftarrow 0$

repeat

  $k \leftarrow k+1$

  for all $i = 1,\dots,N$ **do**

    $A_i x_i^{(k)},\ \mu_i^{(k)} \leftarrow A_i \ \arg\min_{x_i,\ \mu_i \geq 0} \mathscr{L}_{\rho,i}\left(x_i, \lambda^{(k-1)}, \mu_i, z_i^{(k-1)}\right)$

    $\mathscr{g}_i^{(k)}, \mathscr{H}_i^{(k)}, \mathscr{C}_i^{Active,(k)} \leftarrow$ Evaluate derivatives at the optimum

  end for

  $\kappa^{(k)} \leftarrow \rho\theta^k$ if $\mathscr{H}_i^{(l)}$ positive definite for $l \leq k$ else $-\min\{\mathrm{eig}(\mathscr{H}_i^{(l)})\} + \varepsilon_{Feas,Primal}$

  Solve coordinator level QP:

$$\min_{\Delta z_i,\, s} \sum_{i\in\{1,\dots,N\}} \left(\Delta z_i^T \left(\mathscr{H}_i^{(k)} + \kappa^{(k)} I\right) \Delta z_i + \mathscr{g}_i^{(k)T} \Delta z_i\right) + \lambda^{(k-1)T} s + \frac{\nu}{2} \|s\|_2^2 \tag{26a}$$

$$\text{s.t.} \sum_{i\in\{1,\dots,N\}} A_i \left(x_i^{(k)} + \alpha_1^k \left(z_i^{(k)} - x_i^{(k-1)}\right) + \Delta z_i\right) \leq b + s, \quad |\lambda_{QP} \tag{26b}$$

$$\mathscr{C}_i^{Active(k)} \Delta z_i \leq 0. \quad i \in \{1,\dots,N\} \tag{26c}$$

  $\lambda^{(k)} \leftarrow \lambda^{(k-1)} + \alpha_3^k \left(\lambda_{QP} - \lambda^k\right)$

  $z_i^{(k)} \leftarrow z_i^{(k-1)} + \alpha_1^k \left(x_i^{(k)} - z_i^{(k-1)}\right) + \alpha_2^k \Delta z_i^{(k)}$

until $\Phi_{Primal} \leq \varepsilon_{Feas,Primal} \wedge \Phi_{Dual} \leq \varepsilon_{Feas,Dual}$

return $\lambda^{(k)}$

2.7.4 Other methods

There are a variety of other dual based methods. For instance the ones introduced in Maxeiner and Engell (2020) or Wenzel et al. (2016), where, similar to the sub-gradient method, only Lagrange multipliers and the usage of the shared resources are exchanged. However, these methods are designed for equality constrained shared resource allocation problems.

Other methods, e.g. those presented in Kozma et al. (2014) and Nesterov (2004), use the objective values of the individual sub-systems. In this contribution, we focus on the presented methods, since they do not require the knowledge of the objective values of the individual sub-systems. Hence, they can be applied to solve problems where due to confidentiality the profit of the sub-systems cannot be openly communicated.

3 Problem formulation for multiple trajectories with shared inputs and free final times

Due to the overarching constraint on the sharing of the resources, the terminal states of the trajectories change. With fixed final times, boundary conditions on the terminal state which are infeasible due to the overarching resource-sharing constraints, cannot be satisfied. Thus, additional degrees of freedom that enable the satisfaction of the terminal constraints, i.e., free final times, are needed.

The standard approach to include the final time as an optimization variable in trajectory optimization is time scaling. The time horizon is scaled to the interval [0, 1], discretized, and multiplied with the final time which is a continuous variable that is minimized. The number of discretization intervals stays constant, but the lengths of the discretization intervals change. The downside of this approach, for the considered scenario with sharing of resources, is that the constraints on the shared resources cannot be enforced exactly anymore, because the discretization intervals are not synchronized between the sub-systems.

Another possibility is to adjust the number of intervals. Then, the length of the discretization intervals is fixed and the shared resource constraints can be enforced across all systems. As a result, the additional optimization variable, the number of discrete intervals, is of integer type (Van den Broeck et al. 2011).

In the following, we use the latter approach and consider a single terminal constraint for each sub-system that is feasible without the overarching constraint but not necessarily when it is present. The resulting problem of trajectory optimization with shared resources and free final times for the different trajectories can be written as the following mixed-integer non-linear program (MINLP):

$$\begin{aligned} \min_{\substack{\chi_{i,p},\ u_{i,p},\\ (y_{i,p},\ \hat{y}_{i,p})\ \in\ \{0,1\},\\ \forall(p,\ i)\ \in\\ (\{N_{min},\dots,N_{max}\},\ \{1,\dots,N\})}} & \sum_{i\in\{1,\dots,N\}} \sum_{p\in\mathbb{N}} \left(\hat{y}_{i,p} Y_i(\chi_{i,p}) + y_{i,p}\Theta_i\left(\chi_{i,p}, u_{i,p}, \Delta t, p\right)\right), \end{aligned} \tag{27a}$$

$$\text{s.t.} \quad \sum_{i\in\{1,\dots,N\}} u_{i,p} \le u_{shared,max,p}, \quad \forall p \in \{N_{min},\dots,N_{max}\}, \tag{27b}$$

$$\tilde{F}_i\left(\chi_{i,p}, \chi_{i,p+1}, u_{i,p}, \Delta t, p\right) = 0, \quad \forall p \in \{p' | y_{i,p'} = 1\}, i \in \{1,\dots,N\}, \tag{27c}$$

$$\chi_{i,N_{i,0}} = \chi_{0,i}, \quad \forall i \in \{1,\dots,N\}, \tag{27d}$$

$$P_i\left(\chi_{i,p}, u_{i,p}, \Delta t, p\right) \le 0, \quad \forall p \in \{p' | y_{i,p'} = 1\}, i \in \{1,\dots,N\}, \tag{27e}$$

$$T_i\left(\chi_{i,p}\right) \le y_{i,p} M, \quad \forall p \in \{N_{min},\dots,N_{max}\}, i \in \{1,\dots,N\}, \tag{27f}$$

$$T_i\left(\chi_{i,p}\right) \ge (y_{i,p} - 1)M + 1/M, \quad \forall p \in \{N_{min},\dots,N_{max}\}, i \in \{1,\dots,N\}, \tag{27g}$$

$$p - N_{i,0} \le y_{i,p} M, \quad \forall p \in \{N_{min},\dots,N_{max}\}, i \in \{1,\dots,N\}, \tag{27h}$$

$$p - N_{i,0} \ge (y_{i,p} - 1)M + 1/M, \quad \forall p \in \{N_{min},\dots,N_{max}\}, i \in \{1,\dots,N\}, \tag{27i}$$

$$z_{i,p+1} \ge y_{i,p} - y_{i,p+1}, \quad \forall p \in \{N_{min},\dots,N_{max}\}, i \in \{1,\dots,N\}, \tag{27j}$$

$$z_{i,p+1} \le y_{i,p} + y_{i,p+1}, \quad \forall p \in \{N_{min},\dots,N_{max}\}, i \in \{1,\dots,N\}, \tag{27k}$$

$$z_{i,p+1} \le 1 - y_{i,p+1}, \quad \forall p \in \{N_{min},\dots,N_{max}\}, i \in \{1,\dots,N\}, \tag{27l}$$

$$u_{i,p} = 0, \qquad \forall p \in \{p' | y_{i,p'} = 0\}, i \in \{1,\dots,N\}. \tag{27m}$$

The binary variables $y_{i,p}$ and $\hat{y}_{i,p}$ are used for each sub-system i to indicate in which intervals p the trajectory is active ($y_{i,p}$) and in which interval p the trajectory

satisfies the terminal constraints ($\hat{y}_{i,p}$). The connection between continuous and binary variables is done using the Big M method (Nemhauser and Wolsey 1988).

Different to the problem given in Eqs. 5, N_{max} is not the maximum of $N_{i,f}$ but must be a sufficiently large integer such that the problem is feasible, i.e., that all trajectories can satisfy the terminal constraint when the resources are shared.

The problem given in Eqs. 27 can be solved monolithically using an MINLP solver. Since in this contribution distributed solutions are sought, another way to handle the binary variables is to use the heuristic described in Van den Broeck et al. (2011), where the minimum interval for which the terminal constraint is satisfied is found iteratively. The iterative process of determining the final interval is included into the scheme for the satisfaction of the overarching constraints.

Each sub-system updates its number of intervals via the following equation:

$$N_{i,f} = \begin{cases} N_{i,f} + 1, & \text{if } T_i\left(\chi_{i,N_{i,f}}\right) > 0, \\ N_{i,f} - 1, & \text{if } T_i\left(\chi_{i,N_{i,f}-1}\right) \leq 0, \\ N_{i,f}, & \text{otherwise.} \end{cases} \quad \forall i \in \{1, \ldots, N\}. \tag{28}$$

The adaptation is done if at least one of the terminal constraints cannot be reached with the given final times. As an additional criterion, the number of iterations between two adaptations is increased by 1 each time the final times are adapted.

As the changes in the discrete variables become less frequent, the influence of their adaptation vanishes and the distributed optimization methods converge.

Since the objectives are in general not smooth with respect to discrete changes of the final time, the objective functions of the sub-systems should be scaled with their respective final times, i.e. by dividing the objective by the final time, in order to reduce the effect of the discrete changes.

4 Semi-batch reactor case study

In this paper, we consider a modified version of the isothermal semi-batch reactor with a safety constraint example from Ubrich et al. (1999). This reactor has been widely used as a benchmark in the trajectory optimization literature, e.g., Srinivasan et al. (2003).

A first-order reaction is considered, in which the following reaction occurs:

$$A + B \rightarrow C.$$

Previous to the reaction, the reactor is filled with the amount $V_{0,i}$ $c_{A,0,i}$ of reactant A. The dosage profile of reactant B is a degree of freedom and hence the feed rates $u_i(t)$ are the manipulated variables. The ordinary differential equations that describe the trajectories of the states in each reactor i are:

$$\frac{dc_{A,i}(t)}{dt} = -k\, c_{A,i}(t)\, c_{B,i}(t) - \frac{u_i(t)}{V_i(t)} c_{A,i}(t), \tag{29}$$

$$\frac{dc_{B,i}(t)}{dt} = -k\, c_{A,i}(t)\, c_{B,i}(t) - \frac{u_i(t)}{V_i(t)}\big(c_{B,in,i} - c_{B,i}(t)\big), \tag{30}$$

$$\frac{dV_i(t)}{dt} = u_i(t). \tag{31}$$

The trajectories of each reactor i need to satisfy the following constraints:

- Limitation of the feed rate of the reactant B by $u_{Max,i}$,
- The path constraint that the adiabatic temperature rise in the reactor is limited

$$\Delta T_{ad}(t) = \min\left\{c_{A,i}(t), c_{B,i}(t)\right\} \frac{(-\Delta H_R)}{\rho\, c_p}, \tag{32}$$

  which poses a constraint on the concentration $c_{B,i}(t)$ within the reactor,
- The path constraint on the volume of the reactor, which requires that the reaction volume cannot exceed the maximum available reactor volume $V_{Max,i}$.

As a terminal constraint, the amount of C must be above the desired threshold $n_{C,Des,i}$. This amount can be calculated via $n_{C,i}(t) = c_{A,0,i}V_{0,i} - c_{A,i}(t)V_i(t)$. In Table 1, all case study specific numerical values are given.

In trajectory optimization, different criteria can be selected as economically motivated objectives, e.g.:

- Maximization of the valuable product at the end of the batch time, which yields the maximum material efficiency.
- Minimization of the time necessary to produce a certain amount of valuable product, which yields as many batches as possible per time.
- Maximization of productivity, i.e., the amount of valuable product divided by the batch time.

Here, the maximization of the throughput of product C is chosen as the optimization criterion for each reactor i:

$$Y_i(\chi_{i,N_{i,f}}) = -\frac{n_{C,i,f}}{t_{i,f}} = -\frac{c_{A,i,0}V_{0,i} - c_{A,i,f}V_{i,f}}{(N_{i,f} - N_{i,0})\Delta t}. \tag{33}$$

In Fig. 1 the optimal trajectories of the states and the input of a single semi-batch reactor without overarching constraints are shown for an input discretization interval of 4 h. On the left, the trajectories of the states are shown and the trajectory of the amount of final product is shown. The effective constraints on the quantities are indicated by the thin horizontal lines in the same line style. On the right, the input trajectory is shown. One can see the presence of different arcs, i.e., the presence of different active constraints. At first, the maximum feed rate constraint is active, then the temperature at cooling failure limits the feed rate

Table 1 Case study specific parameters for all reactors i

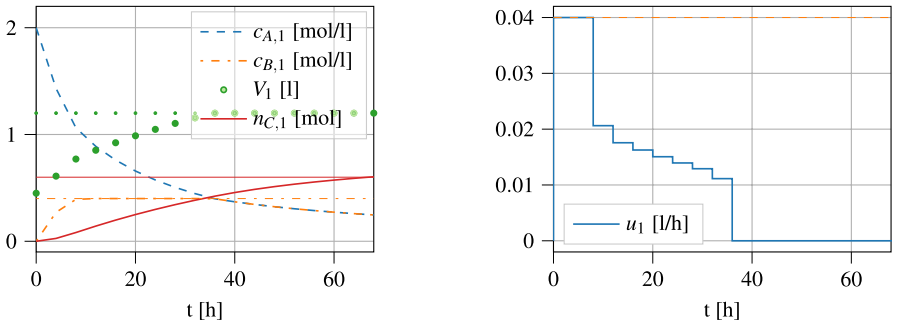
Symbol	Value	Unit
$c_{A,0,i}$	2.0	mol/l
$c_{B,0,i}$	0.0	mol/l
$c_{C,0,i}$	0.0	mol/l
$V_{0,i}$	0.45	l
k	0.0482	l/(mol h)
ΔT_{ad}	20	K
$-\Delta H_R/(\rho \cdot c_p)$	80.0	(l K)/mol
$c_{B,in,i}$	1.2	mol/l
$u_{min,i}$	0.0	l/h
$u_{max,i}$	0.04	l/h
$V_{max,i}$	1.2	l
$n_{C,Des,i}$	0.6	mol
$u_{shared,max}$	0.05	l/h

until the feeding needs to be stopped because the maximum volume of the reaction mixture is reached.

In the case of a single semi-batch reactor, the terminal constraint is satisfied after 17 discrete elements of duration $\Delta t = 4$ h.

Additionally to the individual limitations of the feed flows, we consider a coupling of the reactors via an overarching constraint on the joint feed flow rate of the reactant B:

$$\sum_{i \in \{1,\dots,N\}} u_{i,p} \leq u_{shared,max}, \quad \forall p \in \{N_{min}, \dots, N_{max}\}. \tag{34}$$



(a) States $\chi_1, n_{C,1}$ **(b)** Input u_1

Fig. 1 Optimal trajectories for a single semi-batch reactor without overarching constraints for a time discretization of $\Delta t = 4$ h. On the left, the evolution of the states $c_{A,1}, c_{B,1}$ and V_1 as well as of the amount of final product $n_{C,1}$ is shown. On the right, the corresponding input profile is shown. The thin lines correspond to the constraints on the variables

The maximum combined feed flow rate for all reactors is considered to be constant. Dualizing this overarching constraint Eq. 34 according to Eq. 7 yields the following integral cost term in the objectives of the sub-systems:

$$\Theta_i\left(\chi_{i,p}, u_{i,p}, \Delta t, p\right) = \lambda[p] u_{i,p}, \quad \forall p \in \{N_{min}, \dots, N_{max}\}. \tag{35}$$

4.1 Validation of the solutions

All subsequent distributed solutions are compared to the monolithic solutions of the same problem. In the case of free final times, instead of solving the MINLP, all possible solutions for the final interval $N_{i,f}$ that end no later than 3 intervals from the unconstrained solution are enumerated and used as the benchmark for the evaluation of the different distributed solutions.

For the iterative methods with the augmented Lagrangian term, upon convergence, the solutions are validated with $\rho = 0$ in order to ensure that the constraints are satisfied even without penalty parameters and thus the corresponding Lagrange multipliers satisfy the necessary conditions of optimality.

5 Numerical results

The performance of the different methods is evaluated using the following three criteria:

- required number of iterations,
- evolution of the primal infeasibility over the iterations,
- objective value at convergence.

All scenarios were evaluated using the optimization parameters given in Table 2, which were determined empirically. The optimization problems are solved in Python using IPOPT as NLP solver and the CasADi toolbox for the computation of the derivatives (Andersson et al. 2018; Wächter and Biegler 2006).

In the following, all reactors have the same properties and initial conditions, which is not required in general. Different scenarios are generated by changing the number of reactors, the time discretization Δt, and the starting times of the reactors. Three reactors starting at 0 Δt, 1 Δt, and 2 Δt are indicated by the starting sequence [0, 1, 2].

5.1 Comparison of the methods for fixed final times

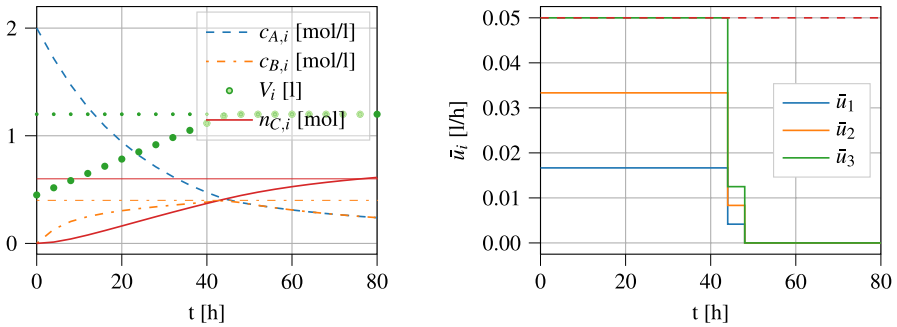
In the following, first the results for fixed final times are discussed. Since changing the fixed final time has only a minor influence on the arc structure of the solution as long as feeding is completed within the considered time horizon, it is not varied.

Table 2 Parameters for the different distributed optimization methods

General	
$\epsilon_{Feas,Primal}$	10^{-5}
$\epsilon_{Feas,Dual}$	10^{-5}
$MaxIter$	5×10^4
$\lambda^{(0)}$	0
Sub-gradient method	
α_0	1
β	0.8
$\gamma_{Decrease}$	0.98
ADMM	
ρ_0	1
z_0	0
$\tau_{Decrease}$	0.98
$\tau_{Increase}$	1.02
δ	2
ALADIN	
ρ	10
z_0	0
ν	10^7
α_1	0.995
α_2	0.995
α_3	0.995
ω	0.9

At first, different scenarios that are generated by varying the starting times of the reactors are considered. Since it is possible to generate scenarios without active overarching constraints, which do not require any coordination, only scenarios where the overarching input constraint is active in at least two intervals are considered.

In Fig. 2 the final distribution of the input between the different sub-systems and the corresponding trajectories of the states are shown for three reactors starting at the same time, i.e. [0, 0, 0]. The input is discretized into piece-wise constant intervals of $\Delta t = 4$ h and the final times for all of the reactors are fixed at 20 Δt. In Fig. 2b, the inputs u_i are stacked on top of each other. Input u_1 is the difference between $\bar{u}_1$ and the baseline at 0, input u_2 is the difference between $\bar{u}_2$ and $\bar{u}_1$, and u_3 is the difference between $\bar{u}_3$ and $\bar{u}_2$. This plot shows how the resources are distributed between the different sub-systems over time and that the shared resource constraint can be satisfied. This is a special case due to the same starting time and the equal distribution of the feedrate (u_i), wherefore all trajectories in response to the prices are the same. The corresponding state profile is displayed in Fig. 2a. It is worth mentioning that the structure of the optimal solutions of the sub-systems (reactors) changes in the distributed optimization. As the maximum feed rate is no

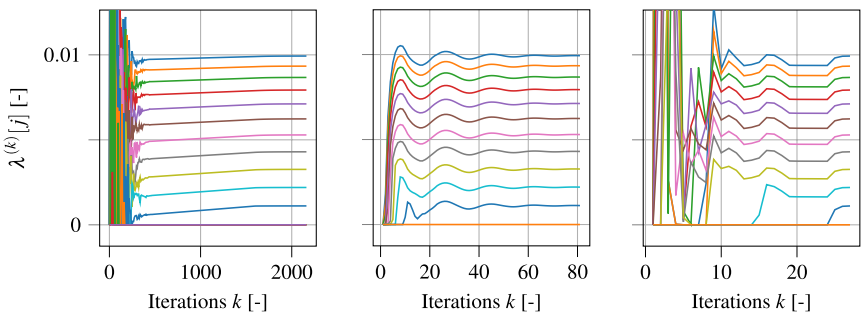


(a) States $\chi_i, n_{C,i}$ **(b)** Inputs u_i stacked on top of each other

Fig. 2 Optimal input trajectories for three reactors starting at [0, 0, 0] (right). Resulting trajectories of the states and the amount of product C for reactors 1, 2, and 3, which are equal due to the same starting time and equal distribution of the input (left)

longer reached, the first arc now is a sensitivity seeking arc, and the constraint on the adiabatic temperature becomes only active at 44 h. So for most of the batch time, the solution of the subsystem problems is not at the constraints, the constraint on the feed is dualized and only enforced by the coordination via the price of the feed.

Figure 3 shows the evolution of the Lagrange multipliers corresponding to the overarching constraints on the shared resources in the maximization of the dual for the three different methods. In Fig. 3a, the spikes ate the beginning of the evolution of the Lagrange multipliers for the sub-gradient method result from the adaptation of the stepsizes to the specific problem. Once the stepsizes are adequate, the prices converge to the values of the monolithic optimization. For ADMM, the prices converge more quickly towards the optimal ones compared to the sub-gradient method, however, the increasing number of active overarching constraints as well as the balancing of primal and dual feasibility result in oscillations towards the optimal



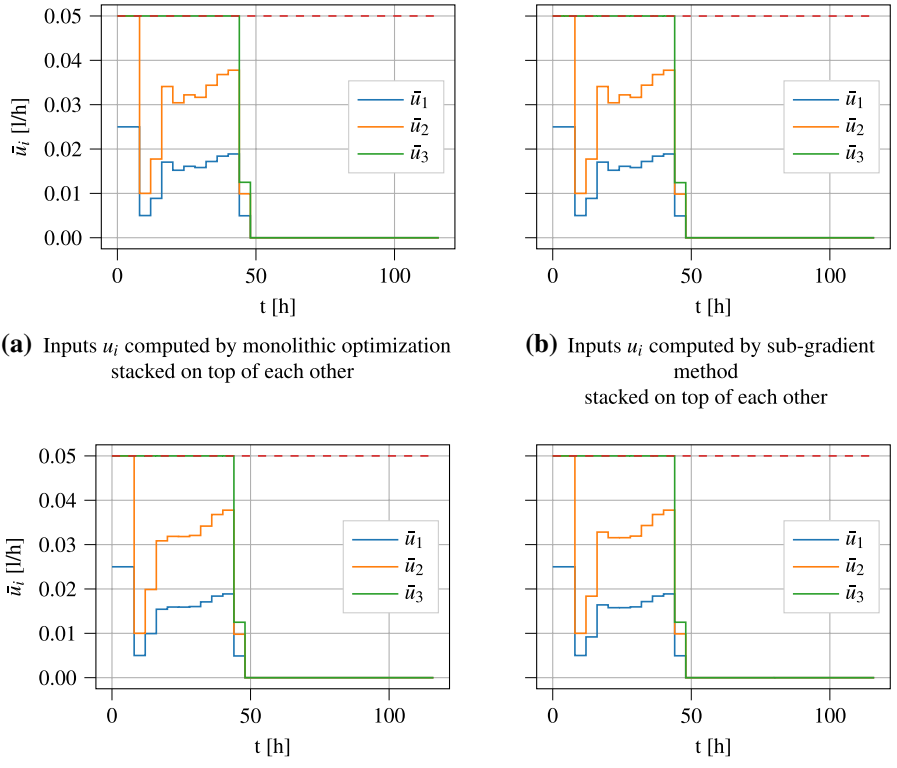
(a) $\lambda^{(k)}$ sub-gradient method **(b)** $\lambda^{(k)}$ ADMM **(c)** $\lambda^{(k)}$ ALADIN

Fig. 3 Evolution of the Lagrange multipliers $\lambda^{(k)}[j], j \in \{1, \dots, m\}$ for three reactors starting at [0, 0, 0] for the different methods to maximize the dual over the iterations k

Lagrange multipliers, as can be seen in Fig. 3b. In Fig. 3c, the prices are adapted according to the ALADIN method, which is based on the derivatives at the currently active set of local inequalities, i.e., the active input and path constraints of all reactors. After a few iterations, the approximated active set is close to the actual one, and the prices converge towards the optimal Lagrange multipliers quickly.

Another interesting scenario that is further examined results for the starting times [0, 0, 2]. The optimized input trajectories are shown for the different methods in Fig. 4. It can be seen that the shared resource constraint, indicated by the dashed line is satisfied for all methods, however, not all methods yield the same trajectories. Nonetheless, the objective values agree up to the 5th significant digit. Thus, the difference in the trajectories can be explained as different local optima that all satisfy (within the specified accuracy) the necessary conditions of optimality.

In Fig. 5, the corresponding evolutions of the primal feasibilities (infinity norm) are shown. As can be seen in Fig. 5a, in this scenario the sub-gradient method first converges to a point at which a further adaptation of the Lagrange multipliers, shown in Fig. 6a, does not have an effect on the primal feasibility. This is due to a set



(a) Inputs u_i computed by monolithic optimization stacked on top of each other

(b) Inputs u_i computed by sub-gradient method stacked on top of each other

(c) Inputs u_i computed by ADMM stacked on top of each other

(d) Inputs u_i computed by ALADIN stacked on top of each other

Fig. 4 Resulting distribution of the resources for three reactors starting at [0, 0, 2]

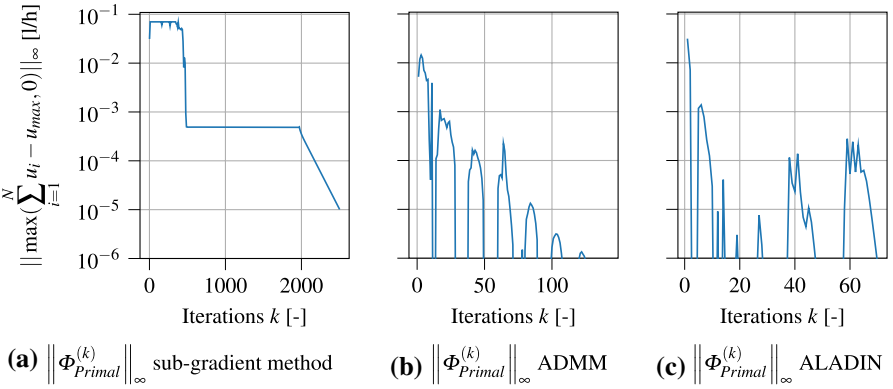


Fig. 5 Evolution of the primal infeasibility over the iterations for the three methods for scenario [0, 0, 2]

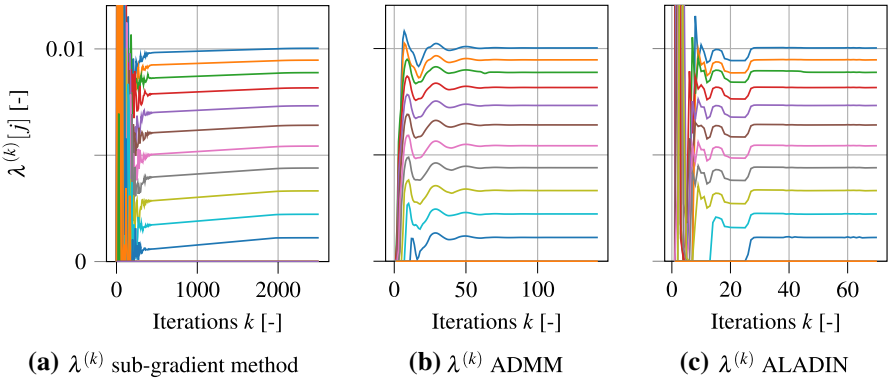


Fig. 6 Evolution of the Lagrange multipliers over the iterations for the three methods for scenario [0, 0, 2]

of local constraints being active. Once this active set changes before iteration 2000, the primal feasibility decreases further and the method converges towards a feasible solution.

For ADMM, the primal feasibility does not steadily decrease but oscillates. Due to the second optimality criterion of dual feasibility, the scheme continues to iterate even when the overarching constraints are satisfied for all intervals. As can be seen in Fig. 6b, starting from iteration 20, the Lagrange multipliers oscillate around their final values before they converge.

Similar to ADMM, ALADIN decreases the infinity norm of the primal feasibility quickly, cf. Fig. 5c. Thereafter, the primal feasibility spikes either when the active set in the QP approximation changes or when new overarching constraints become active. The latter can be seen in Fig. 6c by the new non-zero Lagrange multipliers. The former is only indirectly visible by the significant changes in the Lagrange multipliers. The initial spike in the Lagrange multipliers λ results

from λ_{QP} being calculated based on the wrong active set in the coordinator level QP. Different from the [0, 0, 0] scenario, several iterations are necessary for the Lagrange multipliers, cf. Fig. 5c, to converge to the values that yield the inputs in Fig. 4d.

In Table 3, the results for different scenarios are given. The scenarios are permutations of the starting times between 0 and 8 h. The objective value is scaled by a factor of -1000, such that higher values correspond to better objectives. This table is continued in Table 5 with the remaining permutations of the starting times between 0 and 16 h. If the objective value of a distributed method is slightly better than that of the monolithic solution method, this results from the overarching constraints being satisfied only to the specified tolerance ϵ_{Feas}. The monolithic solution is accurate to the precision of IPOPT, which is set to 10^{-12}. Feeding slightly more into the reactors leads to these small differences in the objective values. Even though it is mostly not reflected in the objective values, the resulting trajectories for the inputs do not always have the same arc structure. In addition to the objective values, also the number of necessary iterations, with the value of the best distributed method highlighted in bold, as well as the numbers of coordinated intervals (# Coord. Ints.), i.e., intervals with active overarching constraints, are shown.

The number of coordinated intervals correlates with the objective value since if fewer intervals need to be coordinated, this means that for more intervals there is no active constraint on the usage of the resources. It is no surprise that the scenarios where the starting times are further apart as well as when the reactors start later, cf. [0, 0, 2] and [0, 2, 2], have a better objective value. The latter results from the fact that $\partial n_{C,i}/\partial t$ decreases once the path constraints on $c_{B,i}$ is active.

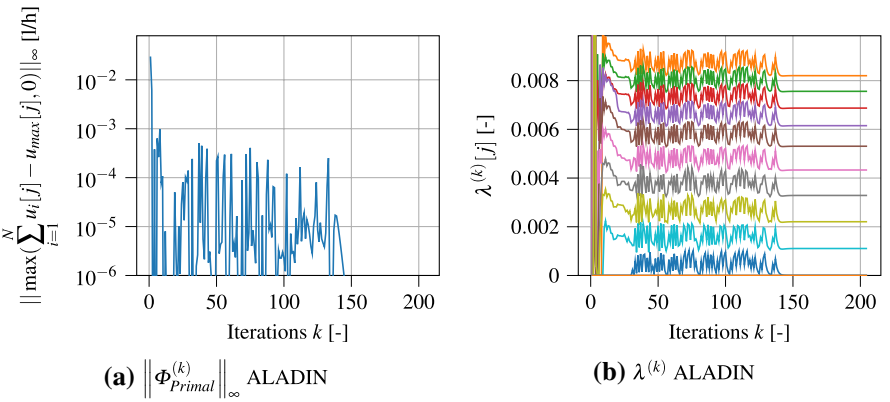
The influence of the granularity of the time discretization on the number of necessary iterations depends on several factors. In general, it can be said that increasing the discretization interval Δt leads on the average to fewer intervals that have to be coordinated and the number of reactor specific constraints that can be active decreases significantly. This can be seen by comparing the results of $\Delta t = 4$ h with the results in Tables 6 and 7 in the Appendix, where the time discretization is changed to $\Delta t = 8$ h and $\Delta t = 16$ h, respectively.

The average number of iterations for the sub-gradient method approximately halves if the time discretization interval is doubled. For all three discretizations, the sub-gradient method has the highest variance. For ADMM the number of iterations is similarly reduced, however, the method converges much more consistently, i.e., the different scenarios do not influence the number of iterations as much as for the other methods. ALADIN exhibits a larger spread for the time discretizations $\Delta t = 4$ h and $\Delta t = 8$ h, however, requires significantly fewer iterations for $\Delta t = 16$ h. An example, where ALADIN requires many iterations is scenario [0, 1, 4], where the evolution of the primal infeasibility and of the Lagrange multipliers are shown in Fig. 7. Between iteration 30 and 140, the stepsize is too large for the algorithm to find the correct QP approximation. Once this set is found, the algorithm converges, however with small steps, such that another 60 iterations are required.

Varying the number of reactors does not make the problem significantly harder. For instance, changing the number of reactors while maintaining their starting position (e.g., [0], [0, 0, 0] and [0, 0, 0, 0, 0, 0]) and adapting the available amount

Table 3 Results for $\Delta t = 4$ h and fixed final times

$N_{i,0}$	Method	Objective	$N_{i,f}$	# Coord. Ints.	Iterations
[0, 0, 0]	Monolithic	22.9866	[20, 20, 20]	11	1
[0, 0, 0]	Sub-gradient	22.9871	[20, 20, 20]	11	2161
[0, 0, 0]	ADMM	22.9855	[20, 20, 20]	11	81
[0, 0, 0]	ALADIN	22.9866	[20, 20, 20]	11	**27**
[0, 0, 1]	Monolithic	23.3897	[20, 20, 20]	11	1
[0, 0, 1]	Sub-gradient	23.3901	[20, 20, 20]	11	1844
[0, 0, 1]	ADMM	23.3897	[20, 20, 20]	11	162
[0, 0, 1]	ALADIN	23.3897	[20, 20, 20]	11	**73**
[0, 0, 2]	Monolithic	23.8303	[20, 20, 20]	11	1
[0, 0, 2]	Sub-gradient	23.8308	[20, 20, 20]	11	2505
[0, 0, 2]	ADMM	23.8303	[20, 20, 20]	11	142
[0, 0, 2]	ALADIN	23.8303	[20, 20, 20]	11	**70**
[0, 1, 1]	Monolithic	23.6869	[20, 20, 20]	10	1
[0, 1, 1]	Sub-gradient	23.6873	[20, 20, 20]	10	994
[0, 1, 1]	ADMM	23.6870	[20, 20, 20]	10	134
[0, 1, 1]	ALADIN	23.6869	[20, 20, 20]	10	**66**
[0, 1, 2]	Monolithic	24.1285	[20, 20, 20]	10	1
[0, 1, 2]	Sub-gradient	24.1289	[20, 20, 20]	10	836
[0, 1, 2]	ADMM	24.1282	[20, 20, 20]	10	111
[0, 1, 2]	ALADIN	24.1285	[20, 20, 20]	10	**62**
[0, 2, 2]	Monolithic	24.4583	[20, 20, 20]	9	1
[0, 2, 2]	Sub-gradient	24.4587	[20, 20, 20]	9	812
[0, 2, 2]	ADMM	24.4582	[20, 20, 20]	9	147
[0, 2, 2]	ALADIN	24.4583	[20, 20, 20]	9	**70**



(a) $\left\|\Phi^{(k)}_{Primal}\right\|_\infty$ ALADIN **(b)** $\lambda^{(k)}$ ALADIN

Fig. 7 Evolution of the primal infeasibility and the Lagrange multipliers over the iterations of ALADIN for scenario [0, 1, 4], where only when the α_i are small enough, the method converges

$u_{shared,max}$ accordingly, i.e., by multiplication with the new number of reactors divided by the old one, does not influence the necessary number of iterations, except for the initial phase, since the solution has the same structure. This can be seen in Fig. 8a–c, where the structure of the imbalances is similar.

The difference in the final numbers of iterations results from the difference in the initial imbalance and the different adaptation of the stepsizes α. The evolution of the prices and of the final structure of the solution are similar.

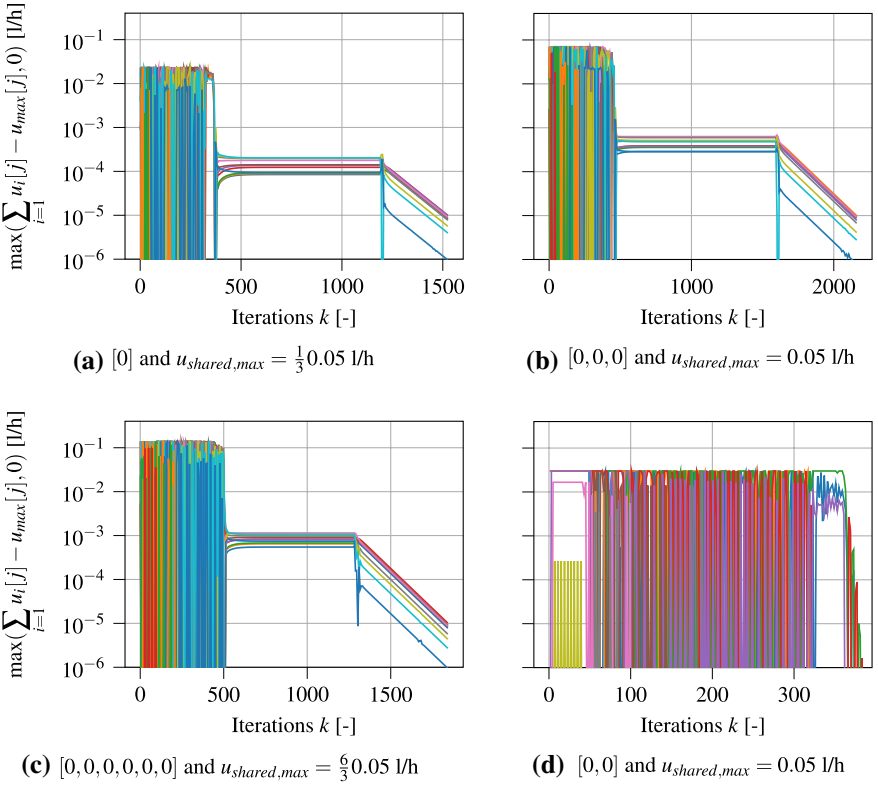
If however $u_{shared,max}$ is kept constant while the number of reactors is changed, the structure of the solution and the trajectory of the multipliers change, cf. Fig. 8d, where the maximum amount of $u_{shared,max} = 0.05$ l/h is distributed between only two reactors.

5.2 Comparison of the methods for variable final times

The three methods are also compared for problems with enforced terminal constraints. The final times are initialized as in the previous case but changed during the optimization.

Additionally to the properties from the previous subsection, the points in time when the final time changes can be evaluated. In Figs. 9 and 10, the evolution of the final times and of the Lagrange multipliers is shown for scenario [0, 0, 2]. It can be seen that the distributed methods converge in the considered cases to the same final times. In the case of the sub-gradient method, as a result of the high fluctuations in the Lagrange multipliers at the beginning, also the final times change significantly until the stepsizes are adjusted accordingly. The large numbers of required iterations are caused by infinitesimal stepsizes resulting from the automatic adaptation of the stepsizes. For ADMM significantly fewer changes can be observed. ALADIN finds the vicinity of the optimal Lagrange multipliers even more quickly, and fewer changes in the final times occur. The number of iterations stays in a similar range as for the case with fixed final times, which can be seen in Table 4 and the resulting distribution of the feed rate between the reactors can be seen in Fig. 11.

In Table 4, as an additional column, the satisfaction of the terminal constraint at convergence is added. The solutions deviate more from the monolithic optimization for the case with free final times. However, for the considered cases and also the ones in the Tables 8, 9, and 10, the heuristic finds feasible solutions. Similarly as with the fixed final times, the smaller Δt is, the more often the distributed solution methods converge to the monolithic solution.



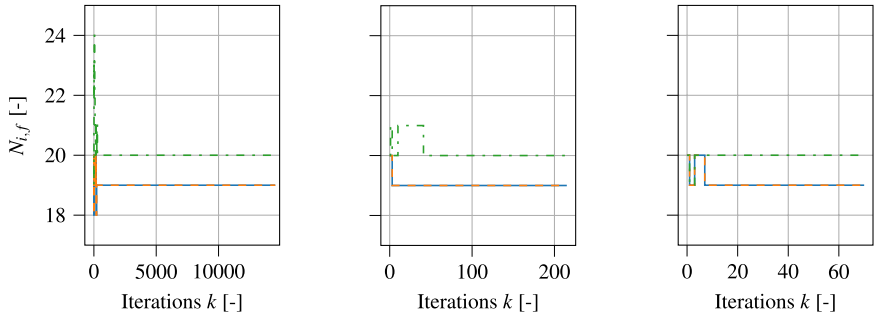
(a) $[0]$ and $u_{shared,max} = \frac{1}{3}0.05$ l/h

(b) $[0,0,0]$ and $u_{shared,max} = 0.05$ l/h

(c) $[0,0,0,0,0,0]$ and $u_{shared,max} = \frac{6}{3}0.05$ l/h

(d) $[0,0]$ and $u_{shared,max} = 0.05$ l/h

Fig. 8 Evolution of the primal feasibilities using the sub-gradient method for different numbers of reactors and available amounts of the shared resource $u_{shared,max}$

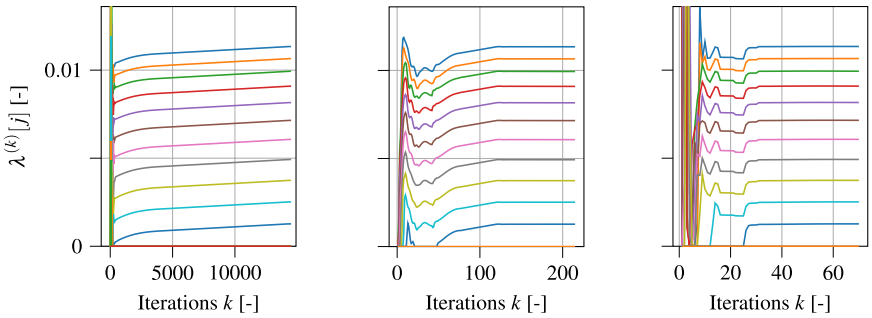


(a) Final times sub-gradient method

(b) Final times ADMM

(c) Final times ALADIN

Fig. 9 Evolution of the final times for scenario [0, 0, 2] with free final times



(a) $\lambda^{(k)}$ sub-gradient method **(b)** $\lambda^{(k)}$ ADMM **(c)** $\lambda^{(k)}$ ALADIN

Fig. 10 Evolution of the Lagrange multipliers for scenario [0, 0, 2] with free final times

6 Discussion

In the following, the results for the distributed optimization methods as well as for the free final times heuristic are discussed and analyzed. By coordinating the shared resource consumption between the individual reactors using Lagrange multipliers, the structure of the optimal solutions for the individual reactors may change. In this case, additional sensitivity seeking arcs arise instead of constraint seeking arcs. So the example shows that both constraint seeking arcs and sensitivity seeking arcs in the sub-problems can be handled.

Since in most cases trajectory optimization is not convex and thus part of a problem class for which few general statements can be made, a qualitative evaluation of the suitability of the methods is done.

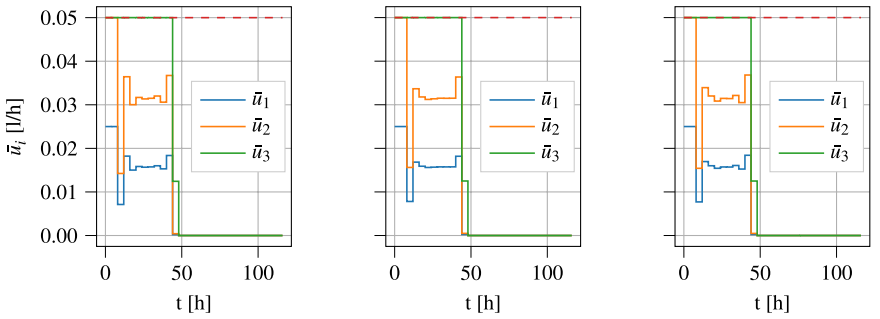
6.1 Comparison of the distributed optimization methods

While for ALADIN an extension exists that guarantees convergence to a local minimum using de facto monolithic optimization steps to determine α_1, α_2, and α_3, for ADMM and for the sub-gradient method no proofs exist that these methods necessarily converge in the non-convex case. Furthermore, it should be noted that all the considered methods based on dual decomposition are infeasible path methods such that only at convergence, the resulting trajectories satisfy the overarching constraints. Nonetheless, in all considered scenarios the distributed optimization methods found feasible solutions with respect to the overarching constraints and converged to at least a local minimum using the parameters given in Table 2.

The sub-gradient method for inequality constrained problems adapts the step-sizes automatically, which has the advantage that no prior information on the

Table 4 Results for $\Delta t = 4$ h and variable final times

$N_{i,0}$	Method	Objective	$N_{i,f}$	Feasible	# Coord. Ints.	Iterations
[0, 0, 0]	Monolithic	22.9866	[20, 20, 20]	True	11	1
[0, 0, 0]	Sub-gradient	22.9867	[20, 20, 20]	True	11	723
[0, 0, 0]	ADMM	22.9857	[20, 20, 20]	True	11	135
[0, 0, 0]	ALADIN	22.9866	[20, 20, 20]	True	11	**27**
[0, 0, 1]	Monolithic	23.9684	[17, 20, 21]	True	9	1
[0, 0, 1]	Sub-gradient	23.8424	[19, 19, 20]	True	11	29987
[0, 0, 1]	ADMM	23.8418	[19, 19, 20]	True	11	108
[0, 0, 1]	ALADIN	23.8418	[19, 19, 20]	True	11	**64**
[0, 0, 2]	Monolithic	24.3662	[17, 20, 21]	True	10	1
[0, 0, 2]	Sub-gradient	24.2821	[19, 19, 20]	True	11	14550
[0, 0, 2]	ADMM	24.2815	[19, 19, 20]	True	11	215
[0, 0, 2]	ALADIN	24.2816	[19, 19, 20]	True	11	**70**
[0, 1, 1]	Monolithic	24.0409	[17, 21, 21]	True	9	1
[0, 1, 1]	Sub-gradient	24.0412	[17, 21, 21]	True	9	740
[0, 1, 1]	ADMM	23.9167	[19, 20, 20]	True	10	65
[0, 1, 1]	ALADIN	23.9180	[19, 20, 20]	True	10	**60**
[0, 1, 2]	Monolithic	24.6857	[18, 19, 21]	True	9	1
[0, 1, 2]	Sub-gradient	24.3585	[19, 20, 20]	True	10	6869
[0, 1, 2]	ADMM	24.3559	[18, 20, 21]	True	9	130
[0, 1, 2]	ALADIN	24.0952	[19, 20, 21]	True	10	**70**
[0, 2, 2]	Monolithic	24.7730	[17, 20, 22]	True	8	1
[0, 2, 2]	Sub-gradient	24.4374	[18, 21, 21]	True	8	820
[0, 2, 2]	ADMM	24.4377	[18, 21, 21]	True	8	**129**
[0, 2, 2]	ALADIN	24.4377	[18, 21, 21]	True	8	206



(a) Inputs u_i sub-gradient method **(b)** Inputs u_i ADMM **(c)** Inputs u_i ALADIN

Fig. 11 Resulting input profiles for the reactors for scenario [0, 0, 2] with free final times

different sub-systems is required. This comes at the cost that for each scenario and for each optimization run a significant number of iterations is required to determine the stepsizes, which are conservative enough such that the active set of the inequality constraints on the shared resources stays mostly the same. Once these stepsizes are found, the method converges slowly and the final number of iterations can vary significantly, especially if local constraints are active, which can prevent improvements of the solution for many iterations as seen for instance in Fig. 5a. While one might conclude from Tables 3 and 4 that the sub-gradient method requires significantly more iterations for the free final times, the continuation in Tables 5 and 8 shows that this is not true. High numbers of necessary iterations are caused by the small stepsizes resulting from the scheme in Eq. 15.

The benefit of the sub-gradient method is its simplicity. The augmentation of the objective function can be interpreted economically as the cost for use of the shared resource and on the coordination layer, the update mechanism matches supply and demand via the prices.

While ADMM does not need more information from the different sub-systems than the sub-gradient method, it introduces artificial penalization terms to regularize the deviations from feasible solutions. This comes with the advantage of a significantly improved speed of convergence in the considered cases. Whether this is acceptable depends on the situation: if distribution is mostly used as a tool to distribute the computational load or to robustify the optimization, it will probably not matter. If the goal is to coordinate the sub-systems while they only optimize their local cost function, it may not be acceptable.

ALADIN uses much more information from the different sub-systems, including state variables as well as derivatives of objective and active constraints, to create QP approximations. As a consequence, ALADIN converges in most cases significantly faster than the other methods, which is also described in the literature by Engelmann and Faulwasser (2019) and Jiang et al. (2017). However, this works only when the QP approximations are accurate. In distributed trajectory optimization there are two factors that can make the approximation difficult: highly non-linear constraints and changing active sets. The former one is the result of the non-linear model equations. The second results from the fact that small changes in the Lagrange multipliers can completely change the active set or the arc structure of the solutions. As long as the active set is not correct, λ_{QP} will not be optimal and ALADIN will not converge. Thus, an adaptive scheme for the stepsizes was presented in Algorithm 5 that, by decreasing the stepsize with the number of iterations, prevents oscillation between different active sets.

In the original ALADIN paper (Houska et al. 2016), the authors recommend that a sufficiently large penalty parameter ρ has to be chosen for the method to converge with a super-linear rate. We found that for the considered problems,

choosing ρ too large resulted in indefinite Hessian matrices $\mathscr{H}_i$. The difficulty to choose ρ and κ results from the following trade-off: If ρ is chosen large, then the Lagrange multipliers μ_i of the local constraints of the different sub-systems can become large, if the z_i variables are not feasible, which in turn can lead to negative definite Hessian matrices $\mathscr{H}_i$. Since $\mathscr{H}_i$ must be positive definite for the coordinator level QP to yield meaningful updates of λ and z_i, the parameter κ, which increases the eigenvalues, must be selected sufficiently large. If however κ is large, then the coordinator level QP will yield very small Δz_i, which again yields small steps towards the optimum. Thus an adaptive scheme was used to prevent indefinite Hessian matrices while eventually allowing larger changes in the reference variables z_i.

These adaptations to ALADIN were made to ensure convergence for all considered scenarios, which is, of course, a trade-off since without these adaptations many scenarios converge much faster.

In general, in some real settings, sharing the gradients of the local objectives may not be acceptable, as this may allow the coordinator to decipher the local cost structure.

6.2 Evaluation of the heuristic for the satisfaction of the terminal constraints

In all considered scenarios with free final times, the terminal constraints were satisfied for the distributed solutions. This can in general not be guaranteed and including this check along with primal and dual feasibility as a convergence criterion can prevent convergence. We thus recommend for the application of distributed optimization with free final times to check the feasibility of all sub-problems at convergence of the distributed optimization method. If this is not satisfied upon convergence, a fallback to a search space with worse objectives can be implemented. For the considered case study, this could, for example, be to increase the final times for all sub-systems and re-optimize without adapting the final times, which would eventually guarantee the satisfaction of the terminal constraint. Other fallbacks could be to allocate $1/N$ of the shared resources to each reactor or to partly disaggregate the profiles.

With respect to the necessary number of required iterations to converge, it can be said that the proposed strategy to adapt the final times can be integrated into the iterative methods without a significant influence on the overall number of iterations.

7 Conclusions

In this contribution, different methods for distributed trajectory optimization, in which the objective values of the sub-systems are not shared, were investigated. As an example, the trajectories of semi-batch reactors that are connected via overarching constraints on the feed rates were optimized. We evaluated and compared different methods based on the optimization of the trajectory of a benchmark semi-batch

reactor and showed that for the considered case, convergence to local minima was achieved. Furthermore, a heuristic was proposed to include the final times of the different trajectories as degrees of freedom. Since the considered problem is not convex, a quantitative analysis of the results was done and possible obstacles for the application of the distributed optimization methods to other trajectory optimization problems were pointed out. In the distributed optimization, the structure of the arcs of the optimal solution may be different from the structure of the solutions for the sub-system problems as constraints for the sub-problems are now dualized.

The three investigated methods, the sub-gradient method, ADMM, and ALADIN, provide different trade-offs between sharing of information and rate of convergence. As expected, in general it can be said that the more information is exchanged between the coordination level and the sub-system level, the faster the methods converge. Since ADMM showed a consistent rate of convergence and it requires no additional information from the sub-systems beyond the resource consumptions, it is recommended as the first choice for distributed trajectory optimization problems if confidentiality is of importance.

The results can also be applied in various other domains where resources have to be allocated or shared between different dynamic systems, e.g., in the coordination of plug-in electric vehicles, the coordination of autonomous robots, distributed control, etc.

Future work will focus on improving the convergence of ALADIN for problems with overarching inequality constraints by better exploiting the available information on the active sets from the sub-problems. There are other techniques than SQP, e.g. interior point or active set methods, which might be adapted to the application with ALADIN to enable faster convergence to the correct active set. Furthermore, the characteristics of the trajectory optimization problems which can be solved with the proposed methods, or more specifically, what structure of arcs and what type of terminal constraints can be coordinated, should be investigated.

Acknowledgements Open Access funding provided by Projekt DEAL.

Appendix A: Results for different Δt and fixed final times

See Tables 5, 6 and 7.

Table 5 Results for $\Delta t = 4$ h and fixed final times

$N_{i,0}$	Method	Objective	$N_{i,f}$	# Coord. Ints.	Iterations
[0, 0, 3]	Monolithic	24.3068	[20, 20, 20]	11	1
[0, 0, 3]	Sub-gradient	24.3072	[20, 20, 20]	11	24174
[0, 0, 3]	ADMM	24.3073	[20, 20, 20]	11	**97**
[0, 0, 3]	ALADIN	24.3068	[20, 20, 20]	11	161
[0, 0, 4]	Monolithic	24.7824	[20, 20, 20]	10	1
[0, 0, 4]	Sub-gradient	24.7821	[20, 20, 20]	10	1797
[0, 0, 4]	ADMM	24.7784	[20, 20, 20]	10	**82**
[0, 0, 4]	ALADIN	24.7823	[20, 20, 20]	10	194
[0, 1, 3]	Monolithic	24.6082	[20, 20, 20]	10	1
[0, 1, 3]	Sub-gradient	24.6086	[20, 20, 20]	10	1022
[0, 1, 3]	ADMM	24.6081	[20, 20, 20]	10	**125**
[0, 1, 3]	ALADIN	24.6082	[20, 20, 20]	10	177
[0, 1, 4]	Monolithic	25.1012	[20, 20, 20]	9	1
[0, 1, 4]	Sub-gradient	25.1008	[20, 20, 20]	9	1040
[0, 1, 4]	ADMM	25.1010	[20, 20, 20]	9	**105**
[0, 1, 4]	ALADIN	25.1011	[20, 20, 20]	9	205
[0, 2, 3]	Monolithic	24.9415	[20, 20, 20]	9	1
[0, 2, 3]	Sub-gradient	24.9417	[20, 20, 20]	9	655
[0, 2, 3]	ADMM	24.9414	[20, 20, 20]	9	**143**
[0, 2, 3]	ALADIN	24.9414	[20, 20, 20]	9	188
[0, 2, 4]	Monolithic	25.4444	[20, 20, 20]	9	1
[0, 2, 4]	Sub-gradient	25.4447	[20, 20, 20]	9	1029
[0, 2, 4]	ADMM	25.4443	[20, 20, 20]	9	118
[0, 2, 4]	ALADIN	25.4444	[20, 20, 20]	9	**87**
[0, 3, 3]	Monolithic	25.1019	[20, 20, 20]	8	1
[0, 3, 3]	Sub-gradient	25.1023	[20, 20, 20]	8	594
[0, 3, 3]	ADMM	25.1020	[20, 20, 20]	8	137
[0, 3, 3]	ALADIN	25.1019	[20, 20, 20]	8	**98**
[0, 3, 4]	Monolithic	25.6230	[20, 20, 20]	8	1
[0, 3, 4]	Sub-gradient	25.6228	[20, 20, 20]	8	429
[0, 3, 4]	ADMM	25.6232	[20, 20, 20]	8	**124**
[0, 3, 4]	ALADIN	25.6230	[20, 20, 20]	8	155
[0, 4, 4]	Monolithic	25.7596	[20, 20, 20]	8	1
[0, 4, 4]	Sub-gradient	25.7600	[20, 20, 20]	8	849
[0, 4, 4]	ADMM	25.7575	[20, 20, 20]	8	**90**
[0, 4, 4]	ALADIN	25.7596	[20, 20, 20]	8	56

Table 6 Results for $\Delta t = 8$ h and fixed final times

$N_{i,0}$	Method	Objective	$N_{i,f}$	# Coord. Ints.	Iterations
[0, 0, 0]	Monolithic	22.9654	[10, 10, 10]	5	1
[0, 0, 0]	Sub-gradient	22.9659	[10, 10, 10]	5	388
[0, 0, 0]	ADMM	22.9648	[10, 10, 10]	5	54
[0, 0, 0]	ALADIN	22.9654	[10, 10, 10]	5	**14**
[0, 0, 1]	Monolithic	23.8082	[10, 10, 10]	5	1
[0, 0, 1]	Sub-gradient	23.8086	[10, 10, 10]	5	407
[0, 0, 1]	ADMM	23.8083	[10, 10, 10]	5	119
[0, 0, 1]	ALADIN	23.8082	[10, 10, 10]	5	**59**
[0, 0, 2]	Monolithic	24.7570	[10, 10, 10]	5	1
[0, 0, 2]	Sub-gradient	24.7566	[10, 10, 10]	5	424
[0, 0, 2]	ADMM	24.7541	[10, 10, 10]	5	54
[0, 0, 2]	ALADIN	24.7570	[10, 10, 10]	5	**17**
[0, 1, 1]	Monolithic	24.4477	[10, 10, 10]	4	1
[0, 1, 1]	Sub-gradient	24.4476	[10, 10, 10]	4	283
[0, 1, 1]	ADMM	24.4476	[10, 10, 10]	4	96
[0, 1, 1]	ALADIN	24.4477	[10, 10, 10]	4	**49**
[0, 1, 2]	Monolithic	25.4168	[10, 10, 10]	4	1
[0, 1, 2]	Sub-gradient	25.4170	[10, 10, 10]	4	222
[0, 1, 2]	ADMM	25.4167	[10, 10, 10]	4	91
[0, 1, 2]	ALADIN	25.4168	[10, 10, 10]	4	**45**
[0, 2, 2]	Monolithic	25.7437	[10, 10, 10]	4	1
[0, 2, 2]	Sub-gradient	25.7441	[10, 10, 10]	4	457
[0, 2, 2]	ADMM	25.7417	[10, 10, 10]	4	65
[0, 2, 2]	ALADIN	25.7437	[10, 10, 10]	4	**28**

Table 7 Results for $\Delta t = 16$ h and fixed final times

$N_{i,0}$	Method	Objective	$N_{i,f}$	# Coord. Ints.	Iterations
[0, 0, 0]	Monolithic	22.9238	[5, 5, 5]	2	1
[0, 0, 0]	Sub-gradient	22.9237	[5, 5, 5]	2	210
[0, 0, 0]	ADMM	22.9230	[5, 5, 5]	2	33
[0, 0, 0]	ALADIN	22.9238	[5, 5, 5]	2	**13**
[0, 0, 1]	Monolithic	24.6118	[5, 5, 5]	2	1
[0, 0, 1]	Sub-gradient	24.6118	[5, 5, 5]	2	237
[0, 0, 1]	ADMM	24.6099	[5, 5, 5]	2	38
[0, 0, 1]	ALADIN	24.6118	[5, 5, 5]	2	**13**
[0, 1, 1]	Monolithic	25.5165	[5, 5, 5]	2	1
[0, 1, 1]	Sub-gradient	25.5164	[5, 5, 5]	2	75
[0, 1, 1]	ADMM	25.5168	[5, 5, 5]	2	75
[0, 1, 1]	ALADIN	25.5165	[5, 5, 5]	2	**14**

Appendix B: Results for different Δt and variable final times

See Tables 8, 9 and 10.

Table 8 Results for $\Delta t = 4$ h and variable final times

$N_{i,0}$	Method	Objective	$N_{i,f}$	Feasible	# Coord. Ints.	Iterations
[0, 0, 3]	Monolithic	24.8066	[17, 20, 21]	True	10	1
[0, 0, 3]	Sub-gradient	24.4615	[19, 19, 21]	True	10	1084
[0, 0, 3]	ADMM	24.4619	[19, 19, 21]	True	10	**118**
[0, 0, 3]	ALADIN	24.4619	[19, 19, 21]	True	10	139
[0, 0, 4]	Monolithic	25.2852	[17, 20, 21]	True	10	1
[0, 0, 4]	Sub-gradient	24.9375	[19, 19, 21]	True	10	1410
[0, 0, 4]	ADMM	24.9382	[19, 19, 21]	True	10	**107**
[0, 0, 4]	ALADIN	24.9379	[19, 19, 21]	True	10	111
[0, 1, 3]	Monolithic	24.8795	[17, 21, 21]	True	9	1
[0, 1, 3]	Sub-gradient	24.7980	[18, 20, 21]	True	9	855
[0, 1, 3]	ADMM	24.7976	[18, 20, 21]	True	9	167
[0, 1, 3]	ALADIN	24.5378	[19, 20, 21]	True	10	**69**
[0, 1, 4]	Monolithic	25.3548	[17, 21, 21]	True	9	1
[0, 1, 4]	Sub-gradient	25.2783	[19, 19, 21]	True	9	4253
[0, 1, 4]	ADMM	25.2275	[18, 19, 22]	True	9	149
[0, 1, 4]	ALADIN	25.2278	[18, 19, 22]	True	9	**69**
[0, 2, 3]	Monolithic	25.2458	[17, 20, 22]	True	8	1
[0, 2, 3]	Sub-gradient	25.1343	[18, 20, 21]	True	8	944
[0, 2, 3]	ADMM	25.1346	[18, 20, 21]	True	8	149
[0, 2, 3]	ALADIN	24.8794	[18, 21, 21]	True	8	**69**
[0, 2, 4]	Monolithic	25.6565	[17, 21, 21]	True	8	1
[0, 2, 4]	Sub-gradient	25.3178	[18, 20, 22]	True	8	512
[0, 2, 4]	ADMM	25.3174	[18, 20, 22]	True	8	164
[0, 2, 4]	ALADIN	25.3175	[18, 20, 22]	True	8	**70**
[0, 3, 3]	Monolithic	24.8544	[17, 21, 23]	True	7	1
[0, 3, 3]	Sub-gradient	24.5231	[18, 22, 22]	True	8	3696
[0, 3, 3]	ADMM	24.5235	[18, 22, 22]	True	8	**128**
[0, 3, 3]	ALADIN	24.5234	[18, 22, 22]	True	8	279
[0, 3, 4]	Monolithic	25.3294	[17, 21, 23]	True	7	1
[0, 3, 4]	Sub-gradient	25.2256	[18, 21, 22]	True	8	4792
[0, 3, 4]	ADMM	25.2254	[18, 21, 22]	True	8	**164**
[0, 3, 4]	ALADIN	25.2253	[18, 21, 22]	True	8	216
[0, 4, 4]	Monolithic	25.1476	[17, 22, 23]	True	7	1
[0, 4, 4]	Sub-gradient	24.8901	[17, 23, 23]	True	7	3429
[0, 4, 4]	ADMM	25.0983	[18, 22, 22]	True	7	**74**
[0, 4, 4]	ALADIN	24.8904	[17, 23, 23]	True	7	126

Table 9 Results for $\Delta t = 8$ h and variable final times

$N_{i,0}$	Method	Objective	$N_{i,f}$	Feasible	# Coord. Ints.	Iterations
[0, 0, 0]	Monolithic	23.4769	[9, 10, 10]	True	5	1
[0, 0, 0]	Sub-gradient	22.9659	[10, 10, 10]	True	5	383
[0, 0, 0]	ADMM	22.9642	[10, 10, 10]	True	5	42
[0, 0, 0]	ALADIN	22.9654	[10, 10, 10]	True	5	**14**
[0, 0, 1]	Monolithic	24.3123	[9, 10, 10]	True	5	1
[0, 0, 1]	Sub-gradient	23.8086	[10, 10, 10]	True	5	392
[0, 0, 1]	ADMM	23.8083	[10, 10, 10]	True	5	120
[0, 0, 1]	ALADIN	23.8082	[10, 10, 10]	True	5	**60**
[0, 0, 2]	Monolithic	25.1245	[9, 9, 11]	True	5	1
[0, 0, 2]	Sub-gradient	25.1250	[9, 9, 11]	True	5	480
[0, 0, 2]	ADMM	25.1244	[9, 9, 11]	True	5	77
[0, 0, 2]	ALADIN	24.1294	[10, 10, 11]	True	5	**37**
[0, 1, 1]	Monolithic	24.4652	[9, 10, 11]	True	4	1
[0, 1, 1]	Sub-gradient	23.9615	[9, 11, 11]	True	4	299
[0, 1, 1]	ADMM	23.9608	[9, 11, 11]	True	4	49
[0, 1, 1]	ALADIN	23.9611	[9, 11, 11]	True	4	**32**
[0, 1, 2]	Monolithic	25.3043	[9, 10, 11]	True	4	1
[0, 1, 2]	Sub-gradient	25.3046	[9, 10, 11]	True	4	364
[0, 1, 2]	ADMM	25.3044	[9, 10, 11]	True	4	94
[0, 1, 2]	ALADIN	25.3043	[9, 10, 11]	True	4	**53**
[0, 2, 2]	Monolithic	25.0878	[9, 11, 11]	True	3	1
[0, 2, 2]	Sub-gradient	24.5917	[10, 11, 11]	True	4	728
[0, 2, 2]	ADMM	24.5917	[10, 11, 11]	True	4	104
[0, 2, 2]	ALADIN	24.0845	[9, 12, 12]	True	3	**74**

Table 10 Results for $\Delta t = 16$ h and variable final times

$N_{i,0}$	Method	Objective	$N_{i,f}$	Feasible	# Coord. Ints.	Iterations
[0, 0, 0]	Monolithic	22.9238	[5, 5, 5]	True	2	1
[0, 0, 0]	Sub-gradient	22.9238	[5, 5, 5]	True	2	209
[0, 0, 0]	ADMM	22.9230	[5, 5, 5]	True	2	33
[0, 0, 0]	ALADIN	22.9238	[5, 5, 5]	True	2	**13**
[0, 0, 1]	Monolithic	23.4988	[5, 5, 6]	True	2	1
[0, 0, 1]	Sub-gradient	23.4988	[5, 5, 6]	True	2	224
[0, 0, 1]	ADMM	23.4971	[5, 5, 6]	True	2	38
[0, 0, 1]	ALADIN	23.4988	[5, 5, 6]	True	2	**20**
[0, 1, 1]	Monolithic	23.4040	[5, 6, 6]	True	2	1
[0, 1, 1]	Sub-gradient	23.4042	[5, 6, 6]	True	2	181
[0, 1, 1]	ADMM	23.4037	[5, 6, 6]	True	2	55
[0, 1, 1]	ALADIN	23.4040	[5, 6, 6]	True	2	**18**

References

Andersson JAE, Gillis J, Horn G, Rawlings JB, Diehl M (2018) CasADi: a software framework for nonlinear optimization and optimal control. Math Program Comput 11:1–36

Bellman R (1957) Dynamic programming, 1st edn. Princeton University Press, Princeton

Bertsekas DP (1995) Dynamic programming and optimal control. Athena Scientific, Belmont

Bertsekas DP (1999) Nonlinear programming. Athena Scientific, Belmont

Bertsekas DP, Tsitsiklis JN (1989) Parallel and distributed computation: numerical methods. Prentice-Hall Inc, Upper Saddle River

Betts JT (1998) Survey of numerical methods for trajectory optimization. J Guid Control Dyn 21(2):193–207

Biegler LT (2007) An overview of simultaneous strategies for dynamic optimization. Chem Eng Process Process Intensif 46(11):1043–1053

Bock HG, Plitt KJ (1985) Multiple shooting algorithm for direct solution of optimal control problems. IFAC Proc Ser 17(2):1603–1608

Boyd S, Parikh N, Chu E, Peleato B, Eckstein J (2010) Distributed optimization and statistical learning via the alternating direction method of multipliers. Found Trends® Mach Learn 3(1):1–122

Boyd S, Vandenberghe L (2004) Convex optimization, vol 25. Cambridge University Press, Cambridge

Camponogara E, Jia D, Krogh B, Talukdar S (2002) Distributed model predictive control. IEEE Control Syst Mag 22(1):44–52

Cheng R, Forbes J, Yip W (2007) Price-driven coordination method for solving plant-wide MPC problems. J Process Control 17(5):429–438

Christofides PD, Scattolini R, Muñoz de la Peña D, Liu J (2013) Distributed model predictive control: a tutorial review and future research directions. Comput Chem Eng 51:21–41

Conejo AJ, Castillo E, Garcia-Bertrand R, Minguez R (2006) Decomposition techniques in mathematical programming: engineering and science applications. Springer, Berlin

Cortés J, Martínez S, Karataş T, Bullo F (2004) Coverage control for mobile sensing networks. IEEE Trans Robot Autom 20(2):243–255

Engell S, Paulen R, Sonntag C, Thompson H, Reniers M, Klessova S, Copigneaux B (2016) Proposal of a European research and innovation agenda on cyber-physical systems of systems. Process Dynamics and Operations Group, Dortmund

Engelmann A, Faulwasser T (2019) Feasibility vs. optimality in distributed AC OPF: a case study considering ADMM and ALADIN. In: Bertsch V, Ardone A, Suriyah M, Fichtner W, Leibfried T, Heuveline V (eds) Advances in energy system optimization. Birkhäuser, Basel, pp 3–12

Esposito WR, Floudas CA (2000) Deterministic global optimization in nonlinear optimal control problems. J Glob Optim 17(1):97–126

Farokhi F, Shames I, Johansson KH (2014) Distributed MPC via dual decomposition and alternative direction method of multipliers. Intell Syst Control Autom Sci Eng 69:115–131

Galceran E, Carreras M (2013) A survey on coverage path planning for robotics. Robot Auton Syst 61(12):1258–1276

Gatsis N, Giannakis GB (2013) Decomposition algorithms for market clearing with large-scale demand response. IEEE Trans Smart Grid 4(4):1976–1987

Hasan M, Hossain E, Kim DI (2014) Resource allocation under channel uncertainties for relay-aided device-to-device communication underlaying LTE-A cellular networks. IEEE Trans Wirel Commun 13(4):2322–2338

Hong M, Luo Zq (2017) On the linear convergence of the alternating direction method of multipliers. Math Program 162(1–2):165–199

Houska B, Frasch J, Diehl M (2016) An augmented lagrangian based algorithm for distributed nonconvex optimization. SIAM J Optim 26(2):1101–1127

Jiang Y, Nimmegeers P, Telen D, Van Impe J, Houska B (2017) A Distributed Optimization Algorithm for Stochastic Optimal Control. IFAC-PapersOnLine 50(1):11263–11268

Jose RA, Ungar LH (2000) Pricing interprocess streams using slack auctions. AIChE J 46(3):575–587

Koutsopoulos I, Iosifidis G (2010) A framework for distributed bandwidth allocation in peer-to-peer networks. Perform Eval 67(4):285–298

Kozma A, Conte C, Diehl M, Methods O, Kozma A, Conte C, Diehl M (2014) Benchmarking large-scale distributed convex quadratic programming algorithms. Optimization methods and software 30(1):191–214

Maestre JM, Negenborn RR (eds) (2014) Distributed model predictive control made easy, intelligent systems, control and automation: science and engineering, vol 69. Springer, Dordrecht

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

Mesarovic MD, Macko D, Takahara Y (1970) Theory of hierarchical, multilevel, systems. Academic Press, New York

Negenborn R, Maestre J (2014) Distributed model predictive control: an overview and roadmap of future research opportunities. IEEE Control Syst 34(4):87–97

Nemhauser G, Wolsey L (1988) Integer and combinatorial optimization. Wiley, Hoboken

Nesterov Y (2004) Introductory lectures on convex optimization. In: Pardalos PM, Hearn DW (eds) Applied optimization, vol 87. Springer, Boston

Nie Y, Biegler LT, Villa CM, Wassick JM (2015) Discrete time formulation for the integration of scheduling and dynamic optimization. Ind Eng Chem Res 54(16):4303–4315

Palomar DP, Chiang M (2006) A tutorial on decomposition methods for network utility maximization. IEEE J Sel Areas Commun 24(8):1439–1451

Palomar DP, Chiang M (2007) Alternative distributed algorithms for network utility maximization: framework and applications. IEEE Trans Autom Control 52(12):2254–2269

Papamichail I, Adjiman CS (2002) A rigorous global optimization algorithm for problems with ordinary differential equations. J Glob Optim 24(1):1–33

Pontryagin L (2018) Mathematical theory of optimal processes. Routledge, New York

Rius-Sorolla G, Maheut J, Estellés-Miguel S, Garcia-Sabater JP (2020) Coordination mechanisms with mathematical programming models for decentralized decision-making: a literature review. Central Eur J Oper Res 28(1):61–104

Rodrigues D, Bonvin D (2019) Dynamic optimization of reaction systems via exact parsimonious input parameterization. Ind Eng Chem Res 58(26):11199–11212

Safdarian A, Fotuhi-Firuzabad M, Lehtonen M (2014) A distributed algorithm for managing residential demand response in smart grids. IEEE Trans Ind Inform 10(4):2385–2393

Sargent R (2000) Optimal control. J Comput Appl Math 124(1–2):361–371

Scattolini R (2009) Architectures for distributed and hierarchical model predictive control—a review. J Process Control 19(5):723–731

Shor NZ (2012) Minimization methods for non-differentiable functions, vol 3. Springer, Berlin

Srinivasan B, Palanki S, Bonvin D (2003) Dynamic optimization of batch processes I. Characterization of the nominal solution. Comput Chem Eng 27(1):1–26

von Stryk O, Bulirsch R (1992) Direct and indirect methods for trajectory optimization. Ann Oper Res 37(1):357–373

Tang W, Allman A, Pourkargar DB, Daoutidis P (2018) Optimal decomposition for distributed optimization in nonlinear model predictive control through community detection. Comput Chem Eng 111:43–54

Ubrich O, Srinivasan B, Stoessel F, Bonvin D (1999) Optimization of a semi-batch reaction system under safety constraints. In: 1999 European control conference (ECC). IEEE, pp 850–855

Van den Broeck L, Diehl M, Swevers J (2011) A model predictive control approach for time optimal point-to-point motion control. Mechatronics 21(7):1203–1212

Van Parys R, Pipeleers G (2017) Distributed MPC for multi-vehicle systems moving in formation. Robot Auton Syst 97:144–152

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

Wang S, Shahidehpour S, Kirschen D, Mokhtari S, Irisarri G (1995) Short-term generation scheduling with transmission and environmental constraints using an augmented Lagrangian relaxation. IEEE Trans Power Syst 10(3):1294–1301

Wang SL, Liao LZ (2001) Decomposition method with a variable parameter for a class of monotone variational inequality problems. J Optim Theory Appl 109(2):415–429

Wenzel S, Paulen R, Beisheim B, Krämer S, Engell S (2017) Market-based coordination of shared resources in cyber-physical production sites. Chemie Ingenieur Technik 89(5):636–644

Wenzel S, Paulen R, Stojanovski G, Krämer S, Beisheim B, Engell S (2016) Optimal resource allocation in industrial complexes by distributed optimization and dynamic pricing. at-Automatisierungstechnik 64(6):428–442

Zhang Y, Gatsis N, Giannakis GB (2013) Robust energy management for microgrids with high-penetration renewables. IEEE Trans Sustain Energy 4(4):944–953

Affiliations

Lukas Samuel Maxeiner[1] · Sebastian Engell[1]

✉ Lukas Samuel Maxeiner
lukas.maxeiner@tu-dortmund.de

[1] Department of Biochemical and Chemical Engineering, Process Dynamics and Operations Group, TU Dortmund University, Emil-Figge-Str. 70, 44227 Dortmund, Germany