Distributed Optimization and Control with ALADIN

Boris Houska and Yuning Jiang

1 Introduction

This chapter aims to give a concise overview of distributed optimization and control algorithms based on the Augmented Lagrangian based Alternating Direction Inexact Newton (ALADIN) method. Here, our goal is to provide a tutorial-style introduction to this relatively new distributed optimization algorithm. In contrast to other existing algorithms, which are often tailored for convex optimization problems, ALADIN is particularly suited for solving non-convex optimization problems. Moreover, another principal advantage of ALADIN is that it can achieve a super-linear or even quadratic convergence rate if suitable Hessian approximations are used.

Brief literature overview

Existing algorithms for distributed convex optimization include dual decomposition [2, 12] as well as the Alternating Direction Method of Multipliers (ADMM) [15, 16]. The main idea of dual decomposition is that for minimization problems with separable strictly convex objective function and affine coupling constraint, the dual function can be evaluated by solving decoupled optimization problems. One way to solve the dual maximization problem is by using a gradient or an accelerated gradient method as suggested in a variety of articles [26, 32, 33]. Other methods for solving this maximization problem are based on semi-smooth Newton methods [13, 14, 25], which have, however, the disadvantage that additional smoothing

B. Houska and Y. Jiang
School of Information Science and Technology, ShanghaiTech University, China.
e-mail: [borish,jiangyn]@shanghaitech.edu.cn

heuristics and line-search routines are needed, as the dual function of most practically relevant convex optimization problem is usually not twice differentiable. An example for a software based on a combination of dual decomposition and a semi-smooth Newton algorithm is the code `qpDunes` [14].

An alternative to dual decomposition algorithms is the Alternating Direction Method of Multipliers (ADMM), which has originally been introduced in [15, 16]. In contrast to dual decomposition, which constructs a dual function by minimizing the Lagrangian over the primal variables, ADMM is based on the construction of augmented Lagrangian functions. During the last decade ADMM received great attention from many researchers and can by now be considered as one of the most promising methods for distributed optimization [6, 8, 9]. In particular, [4] contains a self-contained convergence proof for a rather general class of convex optimization problems. The local convergence behavior of ADMM is linear for most problems of practical relevance, as, for example, discussed in [20] under mild technical assumptions.

Besides dual decomposition, ADMM, and their variants, there exist a variety of other large-scale optimization methods some of which admit the parallelization or even distribution of most of their operations. For example, although sequential quadratic programming methods [3, 30, 31, 37] have not originally been developed for solving distributed optimization problems, they can exploit the partially separable structure of the objective function by using either block-sparse or low-rank Hessian approximations [36]. In particular, limited memory BFGS (L-BFGS) methods are highly competitive candidates for large-scale optimization [27]. As an alternative class of large-scale optimization methods augmented Lagrangian methods [1, 19, 35, 29] have been analyzed and implemented in the software collection `GALAHAD` [7, 17]. A more exhaustive review of such augmented Lagrangian based decomposition methods for convex and non-convex optimization algorithms can be found in [18].

On the road between ADMM and SQP

One of the main differences of ADMM compared to general Newton-type methods or, more specifically, in the context of optimization, Sequential Quadratic Programming (SQP) methods is that they are not invariant under scaling. In practice this means that it is advisable to apply a pre-conditioner in order to pre-scale the optimization variables before applying an ADMM method, as the method may converge very slowly otherwise. A similar statement holds for dual decomposition, if the dual maximization problem is solved with a gradient or an accelerated gradient method. Of course, the question whether it is desirable to exploit second order information in a distributed or large-scale optimization method must be discussed critically.

On the one hand, one would like to avoid linear algebra overhead and decomposition of matrices in large-scale optimization, while, on the other hand, (approximate) second order information, as, for example, exploited by many SQP methods, might improve the convergence rate as well as the robustness of an optimization method with respect to the scaling of the optimization variables. Notice that the above reviewed L-BFGS methods are a good example for a class of optimization algorithms, which attempt to approximate second order terms while the algorithm is running, and which are competitive for large-scale optimization when compared to purely gradient based methods. Thus, if sparse or low-rank matrix approximation techniques, as for example used in SQP methods, could be featured systematically in a distributed optimization framework, this might lead to highly competitive distributed optimization algorithms that are robust with respect to scaling.

Why ALADIN?

The goal of the current chapter is to provide a light introduction to the recently proposed ALADIN method, which can solve convex and non-convex optimization problems in a distributed way [22]. As such this chapter does not present any new technical contribution relative to the original ALADIN article [22] or its variants that can be found in [10, 24, 28], but it summarizes recent developments from a unifying perspective. Here, the main reasons, why we focus on ALADIN, can be summarized as follows:

1. ALADIN is a distributed algorithm that is locally equivalent to a Newton type method, which means that locally superlinear or quadratic convergence rates can be achieved under certain regularity assumptions by using suitable Hessian matrix approximations. If the objective or constraint functions are non-differentiable, these regularity assumptions typically fail to hold. In this case, ALADIN can still be shown to converge, but only weaker convergence rate estimates are possible. It can also be shown that ALADIN contains SQP methods in the limit case if no nonlinear constraints are present and if the augmented Lagrangian parameters tend to infinity as shown in [22].
2. ALADIN can be used to solve non-convex optimization problems to local optimality. Although there exist similar results for ADMM methods [21] under certain assumptions on the augmented Lagrangian parameter, ALADIN has the advantage that its local convergence properties are unaffected by how this parameter is adjusted. Moreover, in [10] a detailed numerical comparison of ADMM and ALADIN is presented for large-scale power network optimization problems, where it is indeed confirmed that both ALADIN and ADMM converge in principle, but ALADIN needs much fewer iterations to achieve the same accuracy.

2 Problem formulation

This section is about structured optimization problems of the form

$$\min_x \sum_{i=1}^{N} f_i(x_i) \quad \text{s.t.} \quad \sum_{i=1}^{N} A_i x_i = b \;\mid\; \lambda \;. \tag{1}$$

Here, the functions $f_i : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ are, in the most general case, semi lower-continuous functions, while the matrices $A_i \in \mathbb{R}^{m\times n}$ and the vector $b \in \mathbb{R}^m$ are given. Throughout this paper dual variables are written immediately after the constraint; that is, in the above problem formulation, $\lambda \in \mathbb{R}^m$ denotes the multiplier that is associated with the coupled equality constraint. In practice, optimization problems of the above form often have the following characteristics:

1. the number $N \in \mathbb{N}$ is potentially large,
2. the functions f_i are potentially non-convex, and
3. the matrices A_i are typically sparse.

In order to solve (1) numerically, it is desirable to distribute the required computations. Here, one often assumes that the i-th processor or agent stores the function f_i and the matrix A_i. For example, in the special case that we have no equality constraints, or if these equality constraints happen to be redundant, (1) can be solved by solving the smaller decoupled optimization problems

$$\min_{x_i} f(x_i) \tag{2}$$

for all $i \in \{1,\ldots,N\}$ separately. Of course, in practice, the equality constraints are usually not redundant, but the solution of (2) might still be a good initial approximation of the solution of (1). Intuitively, one might expect that this is the case, if the dual variable λ is relatively small compared to variations of the objective functions f_i.

2.1 Consensus constraints

As much as distributed optimization problems arise in many applications, they are almost never in standard form (1). In this case, one needs to rewrite the optimization problem first in order to bring it into standard form. In order to understand this, it is helpful to have a look at the sensor network that is shown in Figure 1. In this example, we have a network with 7 sensors. The position of each sensor is denoted by $\chi_i \in \mathbb{R}^2$. If each sensor only takes a measurement η_i of its position χ_i and solves

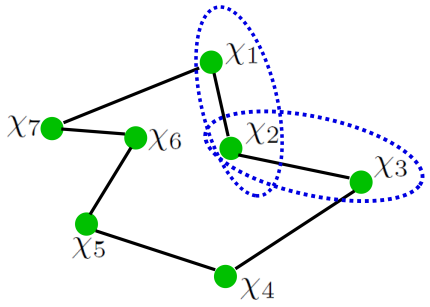

Fig. 1 Example for a 2-dimensional sensor network. The location of the i-th sensor is denoted by $\chi_i \in \mathbb{R}^2$. The distance between the first and the second sensor is a nonlinear function of χ_1 and χ_2, the distance between the second and third sensor depends on χ_2 and χ_3, and so on. Thus, one needs to introduce auxiliary variables in order to express these distances by using local variables only.

$$\forall i \in \{1,\dots,7\}, \qquad \min_{\chi_i} \|\chi_i - \eta_i\|_2^2 \ ,$$

all sensors can estimate their position independently; that is, without communicating with other sensors. However, as soon as the sensors additionally measure the distance to their neighbors, we would like to solve a least-squares localization problem of the form

$$\min_{\chi} \sum_{i=1}^{7} \left\{ \|\chi_i - \eta_i\|_2^2 + \left(\|\chi_i - \chi_{i+1}\|_2 - \bar{\eta}_i\right)^2 \right\} \qquad \text{with } \ \chi_8 = \chi_1 \ . \tag{3}$$

Here, $\bar{\eta}_i$ denotes the measurement for the distance between the i-th and the $(i+1)$-th sensor. Notice that this optimization is not yet in the form (1), because the objective function in (3) comprises terms of the form $\left(\|\chi_i - \chi_{i+1}\|_2 - \bar{\eta}_i\right)^2$ that are, unfortunately, coupled in χ_i and χ_{i+1}. However, such couplings can be resolved by introducing the auxiliary variables

$$x_i = (\chi_i, \zeta_i)^\mathsf{T} \qquad \text{with} \qquad \zeta_i = \chi_{i+1} \ .$$

This has the advantage that we can introduce the separable non-convex objective functions

$$f_i(x_i) = \frac{1}{2}\|\chi_i - \eta_i\|_2^2 + \frac{1}{2}\|\zeta_i - \eta_{i+1}\|_2^2 + \frac{1}{2}\left(\|\chi_i - \zeta_i\|_2 - \bar{\eta}_i\right)^2 \ .$$

Moreover, the affine coupling, $\zeta_i = \chi_{i+1}$, can be written as

$$\sum_{i=1}^{7} A_i x_i = 0$$

by defining the matrices $A_i \in \{-1,0,1\}^{7\times 4}$ appropriately. Constraints of this form are called *consensus constraints*, because they enforce the local copies of all vari-

ables of the different agents to coincide. Thus, a general strategy for formulating distributed optimization problems can be summarized as follows:

1. introduce local copies for neighbor variables until all terms in the objective function are decoupled, and
2. introduce affine consensus constraints that enforce all local copies of the same variable to coincide.

Notice that the same strategy can also be used to decouple inequality constraints as further elaborated in the section below.

2.2 Inequality constraints and hidden variables

Although we regard (1) as the standard form of distributed optimization problems, it is important to keep in mind that the functions f_i often have a very particular structure. This is because one is often interested in solving inequality constrained nonlinear optimization problems of the form

$$\min_{x,z} \quad \sum_{i=1}^{N} F_i(x_i,z_i) \quad \text{s.t.} \quad \begin{cases} \sum_{i=1}^{N} A_i x_i = b & \mid \lambda \\ h_i(x_i,z_i) \leq 0 & \mid \kappa_i \, . \end{cases} \tag{4}$$

Here, the functions h_i can be used to model bounds or other constraints on the local variables. Moreover, the above problem formulation distinguishes between the variables z_i that are entirely local and the variables x_i that are coupled via an affine constraint. In order to write (4) in the form of the standard problem (1), one needs to set

$$f_i(x_i) \; = \; \min_{z_i} \quad F_i(x_i,z_i) \quad \text{s.t.} \quad h_i(x_i,z_i) \leq 0 \, . \tag{5}$$

This means that an evaluation of the functions f_i requires one to solve a parametric optimization problem. In particular, this notation is only possible, if one additionally defines $f_i(x_i) = \infty$ for all x_i at which (5) is infeasible. Notice that this notation hides both the inequality constraints and the variables z_i. This is why we call the variables z_i *hidden variables*. Here, an obvious advantage of introducing the functions f_i is that this simplifies notation and makes some of our theoretical derivations below easier to read. However, one has to keep in mind that this construction implies that

1. the functions f_i are typically non-differentiable, and
2. the functions f_i potentially take the value $+\infty$ (if the hidden optimization problem is infeasible).

Thus, in the following, although we will usually work with the simplified notation in (1), we will sometimes come back to formulation (4) whenever the particular structure of this hidden optimization problem needs to be exploited.

3 Augmented Lagrangian based Alternating Direction Inexact Newton (ALADIN) method

The most basic variant of ALADIN proceeds by repeating two main steps. In the first step, one solves decoupled optimization problems of the form

$$\min_{y_i} \; f_i(y_i) + \lambda^\mathsf{T} A_i y_i + \frac{1}{2} \|y_i - x_i\|^2_{\Sigma_i} \; , \tag{6}$$

where $\lambda \in \mathbb{R}^m$ is the current iterate for the multiplier of the consensus constraint, x_i the current iterate for the primal variable, and $\Sigma_i \succ 0$ a positive definite scaling matrix. Notice that the optimization problem is called "decoupled", because the objective function depends only on the i-th objective function f_i and on the matrix A_i that belongs to the i-th agent, too. This means in particular that the optimization problems (6) can for all $i \in \{1,\ldots,N\}$ be solved in parallel without the need to communicate information between the agents.

In order to understand the terms in the objective, it is helpful to first consider the case that the functions f_i are twice continuously differentiable. In this case, the optimality condition of the decoupled NLPs (6) can be written in the form

$$\nabla f_i(y_i) = \Sigma_i(x_i - y_i) - A_i^\mathsf{T} \lambda \; .$$

This optimality condition must be compared to the first order stationarity condition

$$\nabla f_i(x_i^*) = -A_i^\mathsf{T} \lambda^*$$

that is necessarily satisfied at KKT points (x^*, λ^*) of the original optimization problem (1) that we wish to compute. Here, it becomes clear that if our current iterate (x, λ) is close to (x^*, λ^*), then the solution y_i of (6) can be expected to be close to x_i^*.

Remark 1 *The objective function of* (6) *is in the distributed optimization literature known under the name Augmented Lagrangian. The properties of such augmented Lagrangians have been analyzed by many authors [2, 18]. They are frequently used for developing proximal methods [6] and they may also be considered as one of the basic ingredients for developing ADMM methods [5, 8].*

In the context of ALADIN, the computation of decoupled NLP solutions is alternated with the second main step, namely, the solution of coupled quadratic programming problems of the form

$$\min_{\Delta y} \quad \sum_{i=1}^{N} \left\{ \frac{1}{2} \Delta y_i^\mathsf{T} H_i \Delta y_i + g_i^\mathsf{T} \Delta y_i \right\} \quad \text{s.t.} \quad \sum_{i=1}^{N} A_i(y_i + \Delta y_i) = b \;\mid\; \lambda^+ \,. \qquad (7)$$

Here, the gradient $g_i = \nabla f_i(y_i)$ and the Hessian matrix approximation $H_i \approx \nabla^2 f_i(y_i)$ are both evaluated at the solutions of the decoupled NLP problems. In contrast to ADMM methods, the introduction of this coupled QP is inspired from the field of Sequential Quadratic Programming (SQP). In the most basic variant of ALADIN, the NLP (4) and the QP (7) is all we need to setup a simple method for distributed optimization. This basic variant is summarized in the form of Algorithm 1. Before we discuss the convergence properties of this method in more detail, we have a brief look at possible termination conditions as well as certain advanced variants, which helps us to get an overview of why and how this basic variant of ALADIN can be refined.

Algorithm 1: Basic ALADIN

Input: Initial guesses $x_i \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}^m$, scaling matrices $\Sigma_i \in \mathbb{S}^n_{++}$ and a termination tolerance $\varepsilon > 0$.

Repeat:

1. Solve for all $i \in \{1,\ldots,N\}$ the decoupled NLPs
$$\min_{y_i} \; f_i(y_i) + \lambda^\mathsf{T} A_i y_i + \frac{1}{2} \|y_i - x_i\|^2_{\Sigma_i} \,.$$
2. Set $g_i = \nabla f_i(y_i)$ and $H_i \approx \nabla^2 f_i(y_i)$.
3. Solve the coupled equality constrained QP
$$\min_{\Delta y} \; \sum_{i=1}^{N} \left\{ \frac{1}{2} \Delta y_i^\mathsf{T} H_i \Delta y_i + g_i^\mathsf{T} \Delta y_i \right\} \quad \text{s.t.} \quad \sum_{i=1}^{N} A_i(y_i + \Delta y_i) = b \;\mid\; \lambda^+ \,.$$
4. Set $x \leftarrow x^+ = y + \Delta y$ and $\lambda \leftarrow \lambda^+$ and continue with Step 1.

Remark 2 *If the functions f_i are quadratic forms and if the matrices H_i are set to the exact Hessian matrices, Algorithm 1 converges trivially in one step. This is in contrast to standard ADMM methods, which usually do not converge in one step, even if all variables are scaled optimally. For example, if we consider the quadratic optimization problem*

$$\min_x \; (qx-1)^2 \quad \text{s.t.} \quad x = 0 \,,$$

with $q \neq 0$, the standard way of applying parallel ADMM would be to write this problem in consensus form

$$\min_{x,y} \; I_0(x) + (qy-1)^2 \quad \text{s.t.} \quad x = y \,, \qquad \textit{with} \qquad I_0(x) = \begin{cases} 0 & \textit{if } x = 0 \\ \infty & \textit{otherwise} \end{cases} \,.$$

The augmented Lagrangian of this optimization problem can be written in the form

$$L_\rho(x,y,\lambda) = I_0(x) + (qy-1)^2 + \lambda(y-x) + \frac{\rho}{2}(y-x)^2$$

such that the associated ADMM iteration has the form

$$y^+ = \underset{y}{\operatorname{argmin}}\, L_\rho(x,y,\lambda) = \frac{1}{2q^2+\rho}[2q - \lambda + \rho x] \tag{8}$$

$$\lambda^+ = \lambda + \rho(y-x) \tag{9}$$

$$x^+ = \underset{x}{\operatorname{argmin}}\, L_\rho(x,y^+,\lambda^+) = 0\,, \tag{10}$$

which simplifies to

$$[\lambda^+ - 2q] = \frac{2q^2}{2q^2+\rho}[\lambda - 2q]\,.$$

It is clear that this ADMM iteration converges for any $\rho > 0$ as $\frac{2q^2}{2q^2+\rho} \in (0,1)$. Because we have to choose $\rho < \infty$, the contraction factor is never equal to 0. Moreover, rescaling the optimization variable y is equivalent to using a different $q \neq 0$. Thus, no matter how we scale the variables or choose ρ, ADMM never converges in one step (assuming that we do not initialize at the optimal solution). Thus, ALADIN has, at least for this test example, a clear advantage over ADMM, since it can achieve one-step convergence by using exact Hessian matrices.

3.1 Termination conditions

Notice that Algorithm 1 has two main primal iterates: the iterates x, which are obtained as the solutions of the coupled QPs, and the iterates y, whose block-components are obtained as the solutions of the decoupled NLPs. Now, one can terminate Algorithm 1 if the condition

$$\|y_i - x_i\| \leq \varepsilon \tag{11}$$

is satisfied for all $i \in \{1,\ldots,N\}$ for a user-specified numerical tolerance $\varepsilon > 0$. Notice that this termination condition measures the distance between the iterates x and y. Because the iterates y satisfy

$$\nabla f_i(y_i) = \Sigma_i(x_i - y_i) - A_i^\mathsf{T}\lambda \qquad \overset{(11)}{\Longrightarrow} \qquad \|\nabla f_i(y_i) + A_i^\mathsf{T}\lambda\| \leq \mathbf{O}(\varepsilon)\,,$$

this termination condition is sufficient to ensure that the violation of the first order stationary KKT condition is small if ε is small. Moreover, because the iterates x satisfy the coupled feasibility condition, we have

$$\sum_i^N A_i x_i = b \qquad \overset{(11)}{\Longrightarrow} \qquad \left\| \sum_{i=1}^N A_i y_i - b \right\| \leq \mathbf{O}(\varepsilon) \ ,$$

where the latter conclusion follows from (11) and the triangle inequality for norms. Thus, in summary, it is sufficient to check that (11) holds upon termination, if our goal is to ensure that (y_i, λ) satisfies the first order KKT optimality conditions up to a term of order $\mathbf{O}(\varepsilon)$.

Remark 3 *In practice, one has different options for choosing the norm $\|\cdot\|$ in* (11)*. Because the scaling matrices Σ_i are assumed to be positive definite, one can use the norm*

$$\|y_i - x_i\| = \|y_i - x_i\|_{\Sigma_i} = \sqrt{(y_i - x_i)^\mathsf{T} \Sigma_i (y_i - x_i)} \ .$$

Moreover, if a positive definite Hessian approximation H_i is available, one can directly set $\Sigma_i = H_i$. In Section 4.2 we will show that this particular scaling leads to a consistent convergence proof.

3.2 Derivative-free variants

The basic ALADIN Algorithm 1 has (at least in a very similar version) for the first time been proposed in [22]. However, by now, there have appeared several variants of this algorithm. One of the most important generalizations (see, for example, [28]) is the derivative-free variant that is briefly discussed in this section. Here, the main motivation for generalizing Algorithm 1 is that in this algorithm one needs to evaluate the first order derivatives $g_i = \nabla f_i(y_i)$ of the functions f_i. However, as we have discussed in Section 2.2, these functions are, in practice, not always differentiable. But, as long as the functions f_i are Lipschitz continuous, we still have

$$\Sigma_i(x_i - y_i) - A_i^\mathsf{T} \lambda \in \partial f_i(y_i)$$

at the solution y_i of the decoupled NLPs (4). Here, $\partial f_i(y_i)$ denotes the Clarke subdifferential of f_i at y_i, which coincides with the standard subdifferential of f_i for the case that this function is convex. Clearly, the above inclusion motivates to set

$$g_i = \Sigma_i(x_i - y_i) - A_i^\mathsf{T} \lambda \ ,$$

such that $g_i \in \partial f_i(y_i)$. This expression has the advantage that we can compute g_i without needing to evaluate a derivative of f_i—in fact, such derivatives do not even have to exist at y_i. The corresponding derivative-free variant of ALADIN is summarized in the form of Algorithm 2. As we shall see in the following section, one can still ensure convergence of Algorithm 2 under surprisingly mild conditions; that is, without assuming that the functions f_i are differentiable. Notice that for this variant

Algorithm 2: Derivative-free ALADIN

Input: Initial guesses $x_i \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}^m$ and a termination tolerance $\varepsilon > 0$.

Repeat:

1. Solve for all $i \in \{1,\dots,N\}$ the decoupled NLPs
$$\min_{y_i} \; f_i(y_i) + \lambda^\mathsf{T} A_i y_i + \frac{1}{2}\|y_i - x_i\|^2_{\Sigma_i} \; .$$
2. Choose matrices $H_i \in \mathbb{S}^n_{++}$ and set $g_i = H_i(x_i - y_i) - A_i^\mathsf{T}\lambda$.
3. Solve the coupled equality constrained QP
$$\min_{\Delta y} \sum_{i=1}^N \left\{ \frac{1}{2}\Delta y_i^\mathsf{T} H_i \Delta y_i + g_i^\mathsf{T} \Delta y_i \right\} \quad \text{s.t.} \quad \sum_{i=1}^N A_i(y_i + \Delta y_i) = b \;\mid\; \lambda^+ \; .$$
4. Set $x \leftarrow x^+ = y + \Delta y$ and $\lambda \leftarrow \lambda^+$ and continue with Step 1.

of ALADIN the matrices H_i and Σ_i should both be properly chosen positive definite matrices in order to scale the method.

3.3 Inequality constraint variants

In this section we recall that in many applications, the evaluation of the functions f_i requires itself to solve optimization problems of the form

$$f_i(x_i) \;=\; \min_{z_i} \; F_i(x_i, z_i) \quad \text{s.t.} \quad h_i(x_i, z_i) \leq 0 \; . \tag{12}$$

Here, we also recall that the variables z_i are called hidden variables, as discussed in Section 2.2. Of course, if the functions f_i have this form, one should not introduce a bilevel structure for solving the decoupled NLPs, but solve the joint augmented Lagrangian minimization problem instead,

$$\begin{aligned} \min_{y_i, z_i} \quad & F_i(y_i, z_i) + \lambda^\mathsf{T} A_i y_i + \frac{1}{2}\|y_i - x_i\|^2_{\Sigma_i} \\ \text{s.t.} \quad & h_i(y_i, z_i) \leq 0 \;\mid\; \kappa_i \; . \end{aligned} \tag{13}$$

Notice that this optimization problem corresponds to the expanded form of the decoupled NLPs in Algorithm 2. Here, one should recall that Algorithm 2 is rather general and can, at least in principle, be applied to any non-differentiable functions f_i. Nevertheless, if the functions F_i and h_i are twice Lipschitz-continuously differentiable, one might nevertheless be interested in evaluating the second derivatives

of these functions in order to achieve a better scaling of Algorithm 2. Unfortunately, there is in general no systematic way of pre-computing optimal choices for the positive definite matrices H_i. However, one possible heuristic is to introduce a Gauss-Newton proximal objective function of the form

$$\tilde{F}_i(\xi_i,\zeta_i) \;=\; F_i(\xi_i,\zeta_i) + \sum_{j\in\mathbb{A}_i} \frac{\mu_{i,j}}{2} \left\| \begin{pmatrix} \nabla_y h_j(y_i,z_i) \\ \nabla_z h_j(y_i,z_i) \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} \xi_i - y_i \\ \zeta_i - z_i \end{pmatrix} \right\|_2^2$$

with tuning parameters $\mu_{i,j} > 0$. Here, y_i and z_i denote the solutions of the current (expanded) decoupled NLPs and $\mathbb{A}_i = \{ j \mid \kappa_{i,j} > 0 \}$ the strictly active set of the i-th decoupled optimization problem. The second derivatives of $\tilde{F}_i$ can now be evaluated at y_i and z_i, which suggest to set the Hessian matrix H_i to the Schur complement

$$H_i \;=\; \nabla^2_{\xi,\xi}\tilde{F}_i(y_i,z_i) - \nabla^2_{\xi,\zeta}\tilde{F}_i(y_i,z_i) \left[\nabla^2_{\zeta,\zeta}\tilde{F}_i(y_i,z_i)\right]^{-1} \nabla^2_{\zeta,\xi}\tilde{F}_i(y_i,z_i)\,. \qquad (14)$$

We will see in the sections below that this choice for H_i can be justified in the sense that one can establish locally quadratic convergence for this variant of Algorithm 2 under suitable regularity assumptions as long as the inverse tuning parameters,

$$\frac{1}{\mu_{i,j}} = \mathbf{O}\left(\|x_i - y_i\|\right)\,,$$

tend to 0 as the algorithm converges. However, at the current status of research, we only have working heuristics but no systematic way for choosing these tuning parameters. More systematic inequality handling routines for ALADIN remain among the most important open problems for future research.

3.4 Implementation details

Notice that almost all the steps of Algorithms 1 and 2 are completely decoupled. Consequently, all these steps can be implemented in parallel by the agents of the system without communication. Here, the only exception is the step in which we have to solve the coupled QP

$$\min_{\Delta y} \quad \sum_{i=1}^N \left\{ \frac{1}{2}\Delta y_i^{\mathsf{T}} H_i \Delta y_i + g_i^{\mathsf{T}} \Delta y_i \right\} \quad \text{s.t.} \quad \sum_{i=1}^N A_i(y_i + \Delta y_i) = b \;\mid\; \lambda^+\,, \qquad (15)$$

which requires the agents to communicate with each other. Let us have a closer look at how we can solve this coupled QP. For this aim, we introduce the terms

$$R = \sum_{i=1}^{N} A_i[y_i - H_i^{-1} g_i] - b \qquad \text{and} \qquad M = \sum_{i=1}^{N} A_i H_i^{-1} A_i^{\mathsf{T}} , \tag{16}$$

which can be computed by evaluating the above running sums by communicating the vectors $A_i[y_i - H_i^{-1} g_i]$ and the projected inverse Hessian matrices $A_i H_i^{-1} A_i$, which is possible as long as we choose positive definite Hessian approximations. In the following, we call M the dual Hessian matrix, which is known to be invertible, if the matrices A_i satisfy the Linear Independence Constraint Qualification (LICQ), that is, if the matrix $[A_1, A_2, \ldots, A_N]$ has full-rank. Next, the dual solution of the QP is given by

$$\lambda^+ = M^{-1} R \, . \tag{17}$$

Notice that the evaluation of the matrix vector product can often be further simplified if the matrix M (and its inverse) have a particular structure. Here, we assume that the LICQ condition holds such that the matrix M is indeed invertible. The primal solution can then be recovered as

$$\Delta y_i = -H_i^{-1} \left(g_i + A_i^{\mathsf{T}} \lambda^+ \right) \, . \tag{18}$$

The evaluation of this term can be done in decoupled mode.

Notice that if the matrices H_i are kept constant during the iterations, the dual Hessian matrix M and its pseudo-inverse (or a suitable decomposition of M) can be computed in advance before starting the algorithm. However, one of the main motivations for developing Algorithms 1 and 2 is their similarity to sequential quadratic programming algorithms, which motivates to update the matrices H_i during the iterations as discussed next. In this case, the decomposition of M needs to be updated, too.

4 Convergence analysis

The goal of this section is to concisely summarize the main ideas for establishing convergence of Algorithms 1 and 2. Here, one should keep in mind that the stronger the assumptions on the functions f_i are, the more can be said about the convergence properties of ALADIN. The following sections focus on the following two prototype situations:

1. If the functions f_i are non-convex, global convergence statements can in general not be made. Therefore, Section 4.1 focuses on the local convergence properties of ALADIN under various regularity assumptions, but without assuming that the functions f_i are convex.

2. Another case of practical interest is the case, where the functions f_i are (strictly) convex. In this case, ALADIN converges under rather mild assumptions to global minimizers of (1) without needing regularity conditions on the minimizer or differentiability of the functions f_i.

At this point, it should be mentioned that the sections below are not intended to be general. Our summary of convergence conditions should rather be understood as an introduction, which is intended to help the reader to understand the main conceptual ideas for proving convergence of ALADIN. Of course, in practice, these ideas might have to be re-combined or generalized depending on the particular properties of the functions f_i. For a complete (but also much more technical) discussion of the convergence properties of ALADIN, we refer to [22].

4.1 Local convergence results

One of the main advantages of Algorithm 1 compared to other distributed optimization algorithms is its favorable local convergence behavior. In order to analyze this local convergence behavior, we assume for a moment that the objective functions f_i are twice Lipschitz-continuously differentiable in a neighborhood of a local minimizer x^* of (1). This has the advantage that the following auxiliary result can be derived, which has, in a similar version, originally been established in [22].

Lemma 1 *Let the functions f_i be twice continuously differentiable and let Σ_i be such that the Hessian matrices,*

$$\nabla^2 f_i(x_i) + \Sigma_i \succ \sigma I \, ,$$

of the decoupled NLPs are all positive definite, $\sigma > 0$, for all x in a local neighborhood of a minimizer x^* of (1)*. Then there exist constants $\chi_1, \chi_2 < \infty$ such that the solution y of the decoupled NLPs satisfies*

$$\|y - x^*\| \leq \chi_1 \, \|x - x^*\| + \chi_2 \, \|\lambda - \lambda^*\| \, , \tag{19}$$

whenever $\|x - x^*\|$ and $\|\lambda - \lambda^*\|$ are sufficiently small.

Proof. By writing out the first order necessary optimality conditions for both the decoupled NLPs as well as the original coupled optimization problem (1), we find the equations

$$0 = \nabla f_i(y_i) + A_i^\mathsf{T} \lambda + \Sigma_i (y_i - x_i) \tag{20}$$

$$\text{and} \qquad 0 = \nabla f_i(x_i^*) + A_i^\mathsf{T} \lambda^* \, . \tag{21}$$

Subtracting the second from the first equation and resorting terms yields

$$\nabla f_i(x_i^*) - \nabla f_i(y_i) + \Sigma_i(x_i - x_i^*) = A_i^\mathsf{T}(\lambda - \lambda^*) + \Sigma_i(y_i - x_i^*) \,. \tag{22}$$

Next, we use the fact that the Hessian matrix $\nabla^2 f_i(x_i) + \Sigma_i$ is positive definite in a neighborhood of x_i^*, which means that the inequality

$$\|\nabla f_i(x_i^*) - \nabla f_i(y_i) + \Sigma_i(x_i^* - y_i)\| \geq \sigma \|x_i^* - y_i\|$$

holds in this neighborhood. Thus, if we take the norm on both sides of (22) and substitute the above inequality, we find the inequality

$$\|y - x^*\| \leq \frac{\|A\|}{\sigma}\|\lambda - \lambda^*\| + \frac{\|\Sigma\|}{\sigma}\|x - x^*\| \,.$$

This inequality implies that the statement of the lemma holds with $\chi_2 = \frac{\|A\|}{\sigma}$ and $\chi_1 = \frac{\|\Sigma\|}{\sigma}$. ◇

Theorem 1 *Let the functions f_i be twice Lipschitz-continuously differentiable and let x^* be a (local) minimizer of* (1) *at which the conditions from Lemma 1 are satisfied. If the LICQ condition for NLP* (1) *is satisfied and if we set $H_i = \nabla^2 f_i(x_i)$ in Step 1 of Algorithm 1, then this algorithm converges with locally quadratic convergence rate; that is, there exists a constant $\omega < \infty$ such that we have*

$$\|x^+ - x^*\| + \|\lambda^+ - \lambda^*\| \leq \omega \left(\|x - x^*\| + \|\lambda - \lambda^*\|\right)^2 \,,$$

if $\|x - x^*\|$ and $\|\lambda - \lambda^*\|$ are sufficiently small.

Proof. Notice that the statement of this proposition has been established in a very similar version (and under more general conditions) in [22]. Therefore, we only briefly recall the two main steps of the proof. In the first step, we analyze the decoupled NLPs in Step 2 of Algorithm 1. Lemma 1 states that there exist constants $\chi_1, \chi_2 < \infty$ such that the solution of the decoupled NLPs satisfy

$$\|y - x^*\| \leq \chi_1 \|x - x^*\| + \chi_2 \|\lambda - \lambda^*\| \tag{23}$$

for all x, λ in the neighborhood of (x^*, λ^*). Thus, it remains to analyze the coupled QP, which can be written in the form

$$\begin{aligned} \min_{\Delta y} \quad & \sum_{i=1}^N \left\{ \frac{1}{2} \Delta y_i^\mathsf{T} \nabla^2 f_i(x_i) \Delta y_i + \nabla f_i(y_i)^\mathsf{T} \Delta y_i \right\} \\ \text{s.t.} \quad & \sum_{i=1}^N A_i(y_i + \Delta y_i) = b \;\mid\; \lambda^+ \,, \end{aligned}$$

since we may substitute $g_i = \nabla f_i(y_i)$ and $H_i = \nabla^2 f_i(x_i)$. Next, since the functions $\nabla^2 f_i$ are Lipschitz continuous, we can apply a result from standard SQP theory [27] to show that there exists a constant $\chi_3 < \infty$ such that

$$\left\|x^+ - x^*\right\| \leq \chi_3 \left\|y - x^*\right\|^2 \qquad \text{and} \qquad \left\|\lambda^+ - \lambda^*\right\| \leq \chi_3 \left\|y - x^*\right\|^2 . \tag{24}$$

The statement of the theorem follows now by combining (23) and (24). ◇

Remark 4 *Recall that for the inequality constrained case, which has been discussed in Section 3.3, the functions f_i are in general not twice differentiable and, as such, the above results are not directly applicable. However, under the additional assumption that (x^*, z^*) is a regular KKT point of* (4)*; that is, such that the LICQ conditions, the second order sufficient optimality condition, and the strict complementarity conditions are satisfied at this point, then f_i is still twice continuously differentiable in a neighborhood of x^* (see [27] for details). Thus, under such a regularity assumption the statement of Lemma 1 can be rescued. The statement of Theorem 1 can be generalized for this case, too: if the Hessian matrix H_i is chosen as in* (14) *and if $\frac{1}{\mu_{i,j}} = \mathbf{O}(\|x_i - y_i\|)$, then we can still establish locally quadratic convergence of Algorithm 2. This follows simply from the fact that the particular construction of H_i in* (14) *ensures that solving the coupled QP is locally equivalent to an (inexact) SQP step. Notice that a more formal proof of this result for the case with inequalities can be found in [22].*

4.2 Global convergence results

This section discusses conditions under which the iterates y of Algorithm 2 converge globally to the set of minimizers of (1). Notice that such global convergence conditions can be established for the special case that the matrices $\Sigma_i = H_i$ are constant and under the additional assumption that the functions f_i are closed, proper, and convex. Although these assumptions are restrictive, this result is nevertheless relevant, since it implies that ALADIN converges for strictly convex optimization problems from any starting point and without needing a line-search or other globalization routines. Moreover, the condition that we must set $\Sigma_i = H_i$ and that these matrices are constant can later be relaxed again, but they make our analysis easier, because under these assumptions the following statement holds.

Lemma 2 *If we set $\Sigma_i = H_i \succ 0$, then the solutions $x^+ = y + \Delta y$ and λ^+ of the coupled QP in Algorithms 1 and 2 satisfy*

$$M\lambda^+ = M\lambda + 2r \tag{25}$$

$$\text{and} \qquad x_i^+ = 2y_i - x_i - H_i^{-1} A_i^{\mathsf{T}} (\lambda^+ - \lambda) , \tag{26}$$

with $r = \sum_{i=1}^N A_i y_i - b$.

Proof. We start with Equation (17), which can be written in the form

$$M\lambda^+ = R = \sum_{i=1}^{N} A_i[y_i - H_i^{-1} g_i] - b \,. \tag{27}$$

Next, because we have $\Sigma_i = H_i$, we substitute the expression for the gradient

$$g_i = \Sigma_i(x_i - y_i) - A_i^\mathsf{T}\lambda = H_i(x_i - y_i) - A_i^\mathsf{T}\lambda \;,$$

which yields

$$M\lambda^+ = R = \sum_{i=1}^{N} A_i[y_i - (x_i - y_i) + H_i^{-1} A_i^\mathsf{T}\lambda] - b \tag{28}$$

$$= M\lambda + 2\sum_{i=1}^{N} A_i y_i - 2b = M\lambda + 2r \,. \tag{29}$$

This corresponds to the first equation of this lemma. The second equation follows after substituting the above explicit expressions for g_i and λ^+ in (18). $\diamond$

The main technical idea for establishing global convergence results for ALADIN is to introduce the function

$$\mathscr{L}(x,\lambda) = \|\lambda - \lambda^*\|_M^2 + \sum_{i=1}^{N} \|x_i - x_i^*\|_{H_i}^2 \tag{30}$$

recalling that $M = \sum_{i=1}^{N} A_i H_i^{-1} A_i^\mathsf{T}$ denotes the dual Hessian matrix of the coupled QP. Here, x^* denotes a primal and λ^* a dual solution of (1).

Definition 1 *In the following, we use the symbol $\mathscr{K}$ to denote the set of continuous and monotonously increasing functions $\alpha : \mathbb{R}_+ \to \mathbb{R}$ that satisfy $\alpha(x) > 0$ for all $x > 0$ and $\alpha(0) = 0$.*

An important descent property of $\mathscr{L}$ along the iterates of Algorithms 1 and 2 is established next.

Theorem 2 *Let the functions f_i be closed, proper, and strictly convex, let Problem* (1) *be feasible and such that strong duality holds, and let the matrices $H_i = \Sigma_i$ be symmetric and positive definite. If $x^* = y^*$ denotes the primal and λ^* a (not necessarily unique) dual solution of* (1)*, then the iterates of Algorithm 2 satisfy*

$$\mathscr{L}(x^+,\lambda^+) \leq \mathscr{L}(x,\lambda) - \alpha\left(\|y - y^*\|\right) \tag{31}$$

for a function $\alpha \in \mathscr{K}$.

Proof. In Lemma 2 we have shown that the optimality conditions for the coupled QPs in Algorithms 1 and 2 can be written in the form

$$M\lambda^+ = M\lambda + 2r \tag{32}$$

$$\text{and} \qquad x_i^+ = 2y_i - x_i - H_i^{-1} A_i^\mathsf{T}(\lambda^+ - \lambda) \,, \tag{33}$$

where $r = \sum_{i=1}^{N} A_i y_i - b$ denotes the constraint residuum. Next, the optimality condition for the decoupled NLPs can be written in the form

$$\begin{aligned} 0 \;&\in\; \partial f_i(y_i) + A_i^\mathsf{T} \lambda + H_i(y_i - x_i) \\ &\overset{(33)}{=} \partial f_i(y_i) + A_i^\mathsf{T} \lambda^+ + H_i(x_i^+ - y_i) \end{aligned}$$

recalling that $\partial f_i(y_i)$ denotes the subdifferential of f_i at y_i. Since the functions f_i are assumed to be convex, y_i is a minimizer of the auxiliary function

$$\tilde{f}_i(\xi) = f_i(\xi) + \left(A_i^\mathsf{T} \lambda^+ + H_i(x_i^+ - y_i) \right)^\mathsf{T} \xi \; .$$

Moreover, since we additionally assume that f_i is strictly convex, the function $\tilde{f}$ inherits this property and we find

$$\tilde{f}_i(y_i) \leq \tilde{f}_i(y_i^*) - \tilde{\alpha}\left(\| y_i - y_i^* \| \right)$$

for a $\mathscr{K}$-function $\tilde{\alpha} \in \mathscr{K}$. Summing the above inequalities for all i, substituting the definition of $\tilde{f}_i$, and rearranging terms yields

$$\begin{aligned} \sum_{i=1}^{N} \left\{ f_i(y_i) - f_i(y_i^*) \right\} &\leq \sum_{i=1}^{N} \left(A_i^\mathsf{T} \lambda^+ + H_i(x_i^+ - y_i) \right)^\mathsf{T} (y_i^* - y_i) - \tilde{\alpha}\left(\| y_i - y_i^* \| \right) \\ &= -r^\mathsf{T} \lambda^+ + \sum_{i=1}^{N} (x_i^+ - y_i)^\mathsf{T} H_i (y_i^* - y_i) - \tilde{\alpha}\left(\| y_i - y_i^* \| \right) \; . \end{aligned} \tag{34}$$

Here, the second equality holds due to the primal feasibility of y^*; that is, $b = A y^*$. In order to be able to proceed, we need a lower bound on the left-hand expression of the above inequality. Fortunately, we can use Lagrangian duality to construct such a lower bound. Here, the main idea is to introduce the auxiliary function

$$G(\xi) = \sum_{i=1}^{N} f_i(\xi_i) + \left(\sum_{i=1}^{N} A_i y_i - b \right) \lambda^* \; .$$

Since y^* is a minimizer of the (strictly convex) function G, we find that

$$G(y^*) \leq G(y) - \alpha'\left(\| y - y^* \| \right) \tag{35}$$

for a function $\alpha' \in \mathscr{K}$. This inequality can be written in the equivalent form

$$-r^\mathsf{T} \lambda^* + \alpha'\left(\| y - y^* \| \right) \leq \sum_{i=1}^{N} \left\{ f_i(y_i) - f_i(y_i^*) \right\} \; . \tag{36}$$

By combining (34) and (36), and sorting terms it follows that

$$
\begin{aligned}
-\alpha(\|y-y^*\|) \;&\geq\; r^{\mathsf{T}}(\lambda^+-\lambda^*)+\sum_{i=1}^{N}(y_i-y_i^*)^{\mathsf{T}}H_i(x_i^+-y_i) \\
&\overset{(32)}{=} \frac{1}{2}(\lambda^+-\lambda)^{\mathsf{T}}M(\lambda^+-\lambda^*)+\sum_{i=1}^{N}(y_i-x_i^*)^{\mathsf{T}}H_i(x_i^+-y_i)\,.
\end{aligned}
$$

with $\alpha=\tilde{\alpha}+\alpha'$. The sum on the right hand of the above equation can be written in the form

$$
\begin{aligned}
&\textstyle\sum_{i=1}^{N}(y_i-x_i^*)^{\mathsf{T}}H_i(x_i^+-y_i) \\
&\overset{(26)}{=} \tfrac{1}{4}\textstyle\sum_{i=1}^{N}\left(x_i^++x_i-2x_i^*+H_i^{-1}A_i^{\mathsf{T}}(\lambda^+-\lambda)\right)^{\mathsf{T}}H_i\left(x_i^+-x_i-H_i^{-1}A_i^{\mathsf{T}}(\lambda^+-\lambda)\right) \\
&= -\tfrac{1}{4}(\lambda^+-\lambda)^{\mathsf{T}}M(\lambda^+-\lambda)+\tfrac{1}{4}\textstyle\sum_{i=1}^{N}\left\|x_i^+-x_i^*\right\|_{H_i}^2-\tfrac{1}{4}\sum_{i=1}^{N}\|x_i-x_i^*\|_{H_i}^2
\end{aligned}
$$

By substituting this expression back into the former inequality, it turns out that

$$
\begin{aligned}
-\alpha(\|y-y^*\|) &\geq \frac{1}{2}(\lambda^+-\lambda)^{\mathsf{T}}M(\lambda^+-\lambda^*)-\frac{1}{4}(\lambda^+-\lambda)^{\mathsf{T}}M(\lambda^+-\lambda) \\
&\quad+\frac{1}{4}\sum_{i=1}^{N}\left\|x_i^+-x_i^*\right\|_{H_i}^2-\frac{1}{4}\sum_{i=1}^{N}\|x_i-x_i^*\|_{H_i}^2 \\
&= \frac{1}{4}\mathscr{L}(x^+,\lambda^+)-\frac{1}{4}\mathscr{L}(x,\lambda)\,.
\end{aligned}
$$

This inequality is equivalent to (31), which corresponds to the statement of the theorem. ◇

Notice that Theorem 2 can be used to establish global convergence of ALADIN from any starting point if the matrices H_i are kept constant during the iterations. This result is summarized in the corollary below.

Corollary 1 *Let the functions f_i be closed, proper, and strictly convex and let problem* (1) *be feasible and such that strong duality holds. If the matrices $H_i=\Sigma_i\succ 0$ are kept constant during the iterations, then the primal iterates y of Algorithm 2 converge (globally) to the unique primal solution $x^*=y^*$ of* (1)*, $y\to y^*$.*

Proof. If the matrices H_i are constant, it follows from (31) that the function $\mathscr{L}$ is strictly monotonously decreasing whenever $y\neq y^*$. As $\mathscr{L}$ is bounded from below by 0, the value of $\mathscr{L}(x,\lambda)$ must converge as the algorithm progresses, but this is only possible if y converges to y^*. ◇

The statement of Corollary 1 is rather general in the sense that it establishes global convergence of the primal ALADIN iterates for potentially non-differentiable but strictly convex functions f_i and without assuming that any constraint qualification holds (although the dual iterates might not converge in such a general scenario). Nevertheless, so far, we have not yet addressed the question what happens if the

functions f_i are only convex but not strictly convex. Before we will answer this question, it should be noted first that if the functions f_i are only known to be convex, the set $\mathbb{Y}^*$ of minimizers of (1) is in general not a singleton. Thus, the best we can hope for in such a general case is that the ALADIN iterates y converge to the set $\mathbb{Y}^*$ rather than to a specific minimizer. The following theorem shows that such a convergence statement is indeed possible.

Theorem 3 *Let the functions f_i be closed, proper, and convex and let us assume that strong duality holds for* (1)*. Let $\mathbb{Y}^*$ denote the set of minimizers of* (1)*. If the matrices $H_i \succ 0$ are kept constant during all iterations, then the iterates y of Algorithm 1 converge (globally) to $\mathbb{Y}^*$,*

$$\min_{z \in \mathbb{Y}^*} \|y - z\| \to 0 \, .$$

Proof. Let Λ^* denote the set of dual solutions of (1). As Λ^* is a closed convex set, we can pick a $\lambda^* \in \mathbf{relint}(\Lambda^*)$, where $\mathbf{relint}(\Lambda^*)$ denotes the relative interior[1] of Λ^*. Now, we define the function $\mathscr{L}$ as above but by using any $x^* = y^* \in \mathbb{Y}^*$ and the above particular choice of λ^* in the relative interior of Λ^*. Now, the main idea of the proof of this theorem is to have a closer look at the auxiliary function

$$G(\xi) = \sum_{i=1}^{N} f_i(\xi_i) + \left(\sum_{i=1}^{N} A_i y_i - b \right) \lambda^* \, ,$$

which has already been used in the proof of Theorem 2. Clearly, since we assume that strong duality holds, y^* is a minimizer of this function, but we may have $G(y) = G(y^*)$ even if $y \neq y^*$. However, fortunately, we know that $G(y) = G(y^*)$ if and only if $y \in \mathbb{Y}^*$, since we have strong duality and we have chosen λ^* in the relative interior of Λ^*. Consequently, since closed, proper, and convex functions are lower semi-continuous [5], there must exist a continuous and strictly monotonously increasing function $\alpha : \mathbb{R} \to \mathbb{R}$ with $\alpha(0) = 0$ such that

$$G(y^*) \leq G(y) - \frac{1}{4} \alpha \left(\min_{z \in \mathbb{Y}^*} \|y - z\| \right) \, .$$

By following the same argumentation as in the proof of Theorem 2, we find that this implies

$$\mathscr{L}(x^+, \lambda^+) \leq \mathscr{L}(x, \lambda) - \alpha \left(\min_{z \in \mathbb{Y}^*} \|y - z\| \right) \, . \tag{37}$$

The proof of the corollary follows now by using an analogous argumentation as in Corollary 1. ◇

Notice that the convergence statements of Corollary 1 and Theorem 3 only prove that the iterates y of Algorithm 1 converge to the set of minimizers of $\mathbb{Y}^*$ of Prob-

[1] If Λ^* is a singleton, we have $\mathbf{relint}(\Lambda^*) = \Lambda^*$

lem 1, but no statement is made about the convergence of the iterates x and λ. However, if additional regularity assumptions are introduced, the iterates x and λ also converge, as expected.

Lemma 3 *If the conditions of Theorem* (3) *are satisfied and if the functions f_i are continuously differentiable, then we have both*

$$x \to y \quad \textit{and, as a consequence}, \quad \min_{z \in \mathbb{Y}^*} \|x - z\| \to 0 \, .$$

Moreover, if the LICQ condition for (1) *holds, the dual iterates also converge, $\lambda \to \lambda^*$.*

Proof. As we assume that the functions f_i are continuously differentiable, we have

$$\left\| \nabla f_i(y_i) + A_i^\mathsf{T} \lambda^* \right\|^2_{H_i^{-1}} \to 0 \, ,$$

since ∇f_i is continuous and $y \to y^*$. By writing the left-hand expression in the form

$$\begin{aligned} \sum_{i=1}^N \left\| \nabla f_i(y_i) + A_i^\mathsf{T} \lambda^* \right\|^2_{H^{-1}} &= \sum_{i=1}^N \left\| H_i(x_i - y_i) - A_i^\mathsf{T} (\lambda - \lambda^*) \right\|^2_{H_i^{-1}} \\ &= (\lambda - \lambda^*)^\mathsf{T} M (\lambda - \lambda^*) \\ &\quad + \sum_{i=1}^N \|x_i - y_i\|^2_{H_i} + 2 r^\mathsf{T} (\lambda - \lambda^*) \, , \end{aligned}$$

now, $y \to y^*$ implies $r \to 0$ and, consequently, we find

$$(\lambda - \lambda^*)^\mathsf{T} M (\lambda - \lambda^*) + \sum_{i=1}^N \|x_i - y_i\|^2_{H_i} \to 0 \, ,$$

which implies $\|x - y\| \to 0$. If LICQ holds, the dual Hessian matrix M is invertible and we also have $\lambda \to \lambda^*$. ◇

Remark 5 *The assumption that the functions f_i are continuously differentiable is rather restrictive for practical applications. However, this regularity condition can be relaxed by generalizing the proof of Lemma 3. For example, if f_i can be written in the form*

$$f_i(x_i) = \tilde{f}_i(x_i) + \begin{cases} 0 & \textit{if } \tilde{h}_i(x_i) \leq 0 \\ \infty & \textit{otherwise} \end{cases}$$

with closed, proper, convex, and differentiable functions $\tilde{f}_i : \mathbb{R}^n \to \mathbb{R}$ as well as $\tilde{h}_i : \mathbb{R}^n \to \mathbb{R}^{n_h}$ and if (x^*, λ^*) is a regular KKT point of (1) *as defined in [27], we can still show that $x \to y$ and $\lambda \to \lambda^*$. The proof of this generalization of Lemma 3 is technical but straightforward and therefore not further elaborated in the current chapter.* ◇

5 Numerical implementation and examples

In this section we present one explicitly worked out tutorial example, which illustrates the convergence properties of ALADIN, as well as an application of distributed optimization that arises in the context of model predictive control.

5.1 Tutorial example

In order to understand the local and global convergence properties of Algorithm 2, let us consider the tutorial optimization problem

$$\min_x \ \frac{1}{2}q_1x_1^2+\frac{1}{2}q_2(x_2-1)^2 \quad \text{s.t.} \quad x_1-x_2=0 \ \mid \ \lambda \tag{38}$$

with $q_1,q_2\geq 0$, $q_1+q_2>0$. The explicit solution of this problem is given by

$$z^*=x_1^*=x_2^*=\frac{q_2}{q_2+q_1} \quad \text{and} \quad \lambda^*=-\frac{q_2q_1}{q_2+q_1}\,.$$

The initialization of the primal variable is set to $x_1=x_2=z$. Thus, if we choose $H_1=H_2=H>0$ the subproblems from Step 1 of Algorithm 2 can be written in the form

$$\min_{y_1}\frac{1}{2}q_1y_1^2+\lambda y_1+\frac{H}{2}(y_1-z)^2 \quad \text{and} \quad \min_{y_2}\frac{1}{2}q_2(y_2-1)^2-\lambda y_2+\frac{H}{2}(y_2-z)^2\,.$$

The explicit solution of these subproblems is given by

$$y_1=\frac{Hz-\lambda}{q_1+H} \quad \text{and} \quad y_2=\frac{Hz+\lambda+q_2}{q_2+H}\,.$$

Next, we work out the solution of the QP in Step 3 (z^+,λ^+), with $z^+=x_1^+=x_2^+$, which yields

$$\begin{pmatrix}\lambda^+-\lambda^*\\ z^+-z^*\end{pmatrix}=C\begin{pmatrix}\lambda-\lambda^*\\ z-z^*\end{pmatrix}$$

with

$$C=\frac{1}{(q_1+H)(q_2+H)}\begin{pmatrix}q_1q_2-H^2 & H^2(q_2-q_1)\\ q_1-q_2 & H^2-q_1q_2\end{pmatrix}\,.$$

In this form it becomes clear that the convergence rate of the ALADIN iterates depends on the magnitude of the eigenvalues of the matrix C, which are given by

$$\mathrm{eig}(C) = \pm\sqrt{\frac{q_1 - H}{q_1 + H}\frac{q_2 - H}{q_2 + H}}\,.$$

Notice that for any $H > 0$ these eigenvalues are contained in the open unit disc. This implies that ALADIN converges independent of how we choose H. However, the above eigenvalues depend on the term $\frac{q_1-H}{q_1+H} \in (-1,1)$, which can be interpreted as the relative Hessian approximation error of the first objective function. Similarly, $\frac{q_2-H}{q_2+H} \in (-1,1)$ is the relative error associated with the second objective function. Thus, the closer H approximates q_1 or q_2 the better the convergence rate will get. This convergence behavior is also visualized in Figure 2.

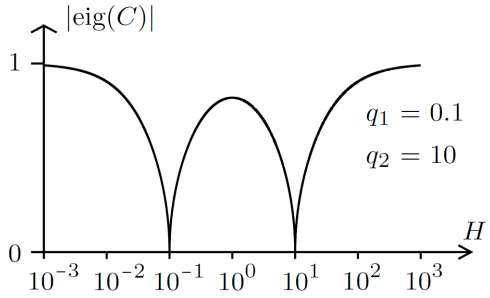

Fig. 2 The absolute value of the maximum eigenvalue, $|\mathrm{eig}(C)|$, of the contraction matrix C versus the scaling $H \in [10^{-3}, 10^{3}]$ for $q_1 = 0.1$ and $q_2 = 10$. Notice that we have $|\mathrm{eig}(C)| < 1$ for all $H > 0$, which implies that ALADIN converges for all choices of H. However, the method contracts faster if $H \approx q_1$ or if $H \approx q_2$, as the eigenvalues of C are close to 0 in these cases.

5.2 Model Predictive Control

This section applies ALADIN in the context of model predictive control (MPC). We consider an optimal control problem in discrete-time form,

$$\min_{\xi,\nu} \sum_{i=0}^{N-1} \ell(\xi_i, \nu_i) + \mathscr{T}(\xi_N) \quad \text{s.t.} \quad \begin{cases} \forall i \in \{0,...,N-1\}, \\ \xi_{i+1} = \mathscr{A}\xi_i + \mathscr{B}\nu_i \\ \xi_i \in \mathbb{X}\,,\ \nu_i \in \mathbb{U}\,,\ \xi_N \in \mathbb{X}_N\,. \end{cases} \tag{39}$$

The main idea of MPC is to solve (39) iteratively at every sampling time based on the current initial state or measurement ξ_0. Here ξ and ν denote state trajectory and control inputs, respectively. The stage cost is denoted by ℓ and $\mathscr{T}$ denotes the terminal cost. The matrices $\mathscr{A}$ and $\mathscr{B}$ are assumed to be given. Moreover, $\mathbb{X}$ and $\mathbb{U}$ denote state and control constraint sets, and $\mathbb{X}_N$ denotes a terminal set.

Standard form

The above optimization problem can be written in the form of the distributed optimization problem (1). For this aim, we introduce the optimization variables

$$x_0 = v_0 \, , \; x_i = [\xi_i^\mathsf{T}, v_i^\mathsf{T}]^\mathsf{T} \, , \text{ and } x_N = \xi_N \, .$$

The associated objective functions are given by

$$f_i(x_i) = \begin{cases} \ell(\xi_i, v_i) \text{ if } (\xi_i, v_i) \in \mathbb{X} \times \mathbb{U} \\ \infty \qquad \text{otherwise} \end{cases} \quad \text{and} \quad f_N(x_N) = \begin{cases} \mathscr{T}(\xi_N) \text{ if } \xi_N \in \mathbb{X}_N \\ \infty \qquad \text{otherwise} \end{cases} .$$

Last but not least, we introduce the matrices

$$A_0 = \begin{pmatrix} \mathscr{B} \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} , A_1 = \begin{pmatrix} -I & 0 \\ \mathscr{A} & \mathscr{B} \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{pmatrix} , A_2 = \begin{pmatrix} 0 & 0 \\ -I & 0 \\ \mathscr{A} & \mathscr{B} \\ \vdots & \vdots \\ 0 & 0 \end{pmatrix} , \ldots , A_N = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ -I \end{pmatrix}$$

in order to represent the dynamics in the form of an affine coupling constraint. Notice that this optimization problem has $6N$ primal and $4(N+1)$ dual variables.

Parameters

In order to setup a benchmark case study with 4 states and 2 controls, we use the system matrices

$$\mathscr{A} = \begin{pmatrix} 0.999 & -3.008 & -0.113 & -1.608 \\ 0.000 & 0.986 & 0.048 & 0.000 \\ 0.000 & 2.083 & 1.010 & 0.000 \\ 0.000 & 0.053 & 0.050 & 1.000 \end{pmatrix} , \mathscr{B} = \begin{pmatrix} -0.080 & -0.635 \\ -0.029 & -0.014 \\ -0.868 & -0.092 \\ -0.022 & -0.002 \end{pmatrix} ,$$

the quadratic stage cost

$$\ell(\xi, v) = \xi^\top \begin{pmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 100 \end{pmatrix} \xi + v^\top \begin{pmatrix} 10 & 0 \\ 0 & 10 \end{pmatrix} v \, ,$$

the locally exact terminal cost

$$\mathscr{T}(\xi) = \xi^\top \begin{pmatrix} 1.42 & -26.08 & -0.96 & 10.33 \\ -26.08 & 1462.89 & 53.93 & -776.41 \\ -0.96 & 53.93 & 10.25 & 36.37 \\ 10.33 & -776.41 & 36.37 & 1291.95 \end{pmatrix} \xi \,,$$

and the constraint sets

$$\mathbb{X} = \left\{ \xi \in \mathbb{R}^4 \,\middle|\, \begin{bmatrix} -0.5 \\ -10 \end{bmatrix} \preceq \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \xi \preceq \begin{bmatrix} 0.5 \\ 10 \end{bmatrix} \right\},$$
$$\mathbb{U} = \left\{ \nu \in \mathbb{R}^2 \,\middle|\, \begin{bmatrix} -25 \\ -25 \end{bmatrix} \preceq \nu \preceq \begin{bmatrix} 25 \\ 25 \end{bmatrix} \right\}.$$

The terminal constraint is set to $\mathbb{X}_N = \mathbb{X}$ and we use the initial state

$$\xi_0 = (10, 0, 10, 10)\,.$$

Numerical results

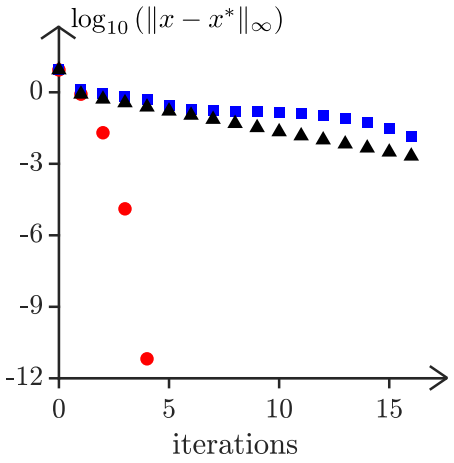

Fig. 3 Logarithm of the distance of the current iterates to the optimal solution for ALADIN with exact Hessian (red circles), ALADIN with fixed Hessians (black triangles), and ADMM (blue rectangles).

In this section, we solve the above MPC problem with ADMM [4] as well as the presented ALADIN method. Figure 3 shows the convergence of two variants of ALADIN versus traditional ADMM for the first MPC iteration. Here, the first variant of ALADIN uses a constant Hessian matrix while the other one updates H_i during the iterations. Notice that both ADMM as well as ALADIN with constant Hessian have a linear convergence rate. This is in contrast to the other variant of ALADIN,

where we use exact Hessians in order to achieve a locally quadratic convergence rate, as discussed in Theorem 1.

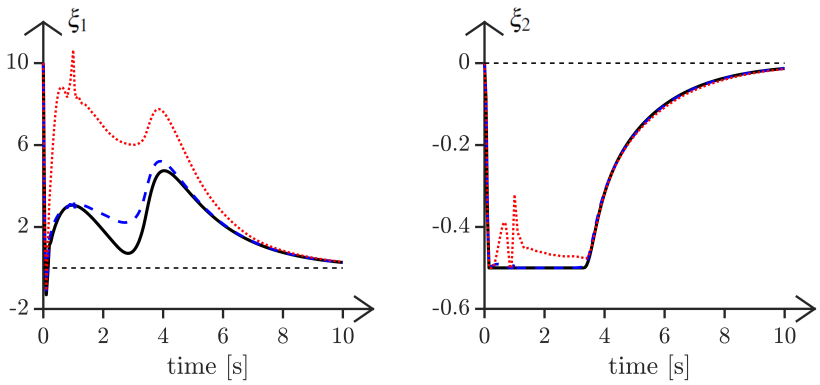

Fig. 4 First two components of the closed-loop state trajectory for 3 ALADIN iterations per sampling time (red dotted line) and 10 ALADIN iterations per sampling time (blue dashed line). The black solid line depicts the optimal closed-loop trajectory.

Figure 4 shows the closed-loop state trajectory that has been obtained by running the proposed algorithm with a fixed number of iterations per sampling time. Here, we simulate 200 time steps with sampling time 0.05 using 3 or 10 ALADIN iterations per sampling time, respectively. The corresponding optimal closed-loop trajectory is shown as a reference in the form of the black solid line.

6 Conclusions

This chapter has presented a light introduction to a relatively new distributed optimization algorithm, named Augmented Lagrangian based Alternating Direction Inexact Newton (ALADIN) method. This algorithm can be used to solve both convex as well as non-convex optimization problems, while exploiting separable structures. Here, one major advantage to traditional ADMM methods is that one can update the Hessian matrix during the iterations such that a locally quadratic convergence rate can be observed. Moreover, we have discussed how global convergence guarantees can be derived for ALADIN under the assumption that the objective functions are strictly convex.

In a second part of this chapter we have additionally illustrated the numerical performance of ALADIN compared to ADMM, as well as its use in real-time model predictive control. During the past years, there have appeared a couple of articles on applications of ALADIN for convex and non-convex distributed optimiza-

tion [10, 11, 22, 23, 24, 28, 34], which all report promising numerical performance for a large variety of applications. Nevertheless, as we have also discussed in this chapter, there are still a number of pressing open problems in distributed optimization. For example, it is not clear at the current status of research whether one can still ensure global convergence of ALADIN for convex problems, if the Hessian matrices are updated during the iterations. Similarly, it is not at clear at all how to update these Hessian matrices in the presence of active set changes in the distributed solvers. Last but not least, although one option for designing a globalization routine for ALADIN with application to non-convex problems has been presented in [22], much more research will be needed to make these globalization routines robust enough to perform well on larger benchmark case studies for non-convex optimization. Thus, as much as this chapter has provided an introduction to ALADIN, it has also been intended to serve as an invitation to the optimization and control community to join this promising direction of distributed optimization method development.

References

1. R. Andreani, E.G. Birgin, J.M. Martinez, and M.L. Schuverdt. On augmented Lagrangian methods with general lower-level constraints. *SIAM Journal on Optimization*, 18:1286–1309, 2007.
2. D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, second ed., 1999.
3. P.T. Boggs and J.W. Tolle. Sequential quadratic programming. *Acta numerica*, 4:1–51, 1995.
4. S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3:1–122, 2011.
5. S. Boyd and L. Vandenberghe. *Convex Optimization*. University Press, Cambridge, 2004.
6. G. Chen and M. Teboulle. A proximal-based decomposition method for convex minimization problems. *Mathematical Programming*, 64:81–101, 1994.
7. A. R. Conn, GIM Gould, and P. L. Toint. *LANCELOT: a Fortran package for large-scale nonlinear optimization (Release A)*, volume 17. Springer Science & Business Media, 2013.
8. J. Eckstein and D.P. Bertsekas. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55:293–318, 1992.
9. J. Eckstein and M.C. Ferris. Operator-splitting methods for monotone affine variational inequalities, with a parallel application to optimal control. *INFORMS Journal on Computing*, 10:218–235, 1998.
10. A. Engelmann, Y. Jiang, T. Mühlpfordt, B. Houska, and T. Faulwasser. Towards distributed OPF using ALADIN. *IEEE Transactions on Power Systems*, 34(1):584–594, 2019.
11. A. Engelmann, T. Mühlpfordt, Y. Jiang, B. Houska, and T. Faulwasser. Distributed stochastic AC optimal power flow based on polynomial chaos expansion. In *Proceedings of the American Control Conference*, pages 6188–6193, 2018.
12. H. Everett. Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 11(399–417), 1963.
13. H.J. Ferreau, A. Kozma, and M. Diehl. A parallel active-set strategy to solve sparse parametric quadratic programs arising in MPC. In *Proceedings of the 4th IFAC Nonlinear Model Predictive Control Conference*, 2012.
14. J.V. Frasch, S. Sager, and M. Diehl. A parallel quadratic programming method for dynamic optimization problems. *Mathematical Programming Computation*, 7(3):289–329, 2015.

15. D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximations. *Computers and Mathematics with Applications*, 2:17–40, 1976.
16. R. Glowinski and A. Marrocco. Sur l'approximation, par elements finis d'ordre un, et la resolution, par penalisation-dualite, d'une classe de problems de dirichlet non lineares. *Revue Francaise d'Automatique, Informatique, et Recherche Operationelle*, 9:41–76, 1975.
17. N.I.M. Gould, D. Orban, and P. Toint. GALAHAD, a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization. *ACM Transactions on Mathematical Software*, 29(4):353–372, 2004.
18. A. Hamdi and S.K. Mishra. Decomposition methods based on augmented Lagrangian: a survey. In *Topics in Nonconvex Optimization. Mishra, S.K., Chapter 11*, pages 175–204, 2011.
19. M.R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4:302–320, 1969.
20. M. Hong and Z.Q. Luo. On the linear convergence of the alternating direction method of multipliers. *Mathematical Programming*, 162(1):165–199, 2017.
21. M. Hong, Z.Q. Luo, and M. Razaviyayn. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. *SIAM Journal on Optimization*, 26(1):337–364, 2016.
22. B. Houska, J. Frasch, and M. Diehl. An augmented Lagrangian based algorithm for distributed non-convex optimization. *SIAM Journal on Optimization*, 26(2):1101–1127, 2016.
23. Y. Jiang, J. Oravec, B. Houska, and M. Kvasnica. Parallel explicit model predictive control. *arXiv preprint:1903.06790*, 2019.
24. D. Kouzoupis, R. Quirynen, B. Houska, and M. Diehl. A block based ALADIN scheme for highly parallelizable direct optimal control. In *Proceedings of the American Control Conference*, pages 1124–1129, 2016.
25. A. Kozma, J. Frasch, and M. Diehl. A distributed method for convex quadratic programming problems arising in optimal control of distributed systems. In *Proceedings of the 52nd IEEE Conference on Decision and Control*, pages 1526–1531, 2013.
26. I. Necoara, D. Doan, and J.A.K. Suykens. Application of the proximal center decomposition method to distributed model predictive control. In *Proceedings of the 47th IEEE Conference on Decision and Control*, pages 2900–2905, 2008.
27. J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering Springer, 2 edition, 2006.
28. J. Oravec, Y. Jiang, B. Houska, and M. Kvasnica. Parallel explicit MPC for hardware with limited memory. In *Proceedings of the 20th IFAC World Congress*, pages 3356–3361, 2017.
29. M.J.D. Powell. *A method for nonlinear constraints in minimization problems*. Optimization, (R. Fletcher, ed.), Academic Press, 1969.
30. M.J.D. Powell. *A fast algorithm for nonlinearly constrained optimization calculations*. Numerical Analysis Dundee, Springer Verlag, Berlin, 1977.
31. M.J.D. Powell. *The convergence of variable metric methods for nonlinearly constrained optimization calculations*. Nonlinear Programming 3, Academic Press, New York and London, 1978.
32. A. Rantzer. Dynamic dual decomposition for distributed control. In *Proceedings of the 2009 American Control Conference*, pages 884–888, 2009.
33. S. Richter, M. Morari, and C.N. Jones. Towards computational complexity certification for constrained MPC based on Lagrange relaxation and the fast gradient method. In *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, 2011.
34. J. Shi, Y. Zheng, Y. Jiang, M. Zanon, R. Hult, and B. Houska. Distributed control algorithm for vehicle coordination at traffic intersections. In *Proceedings of the 17th European Control Conference*, pages 1166–1171, 2011.
35. RA Tapia. Quasi-newton methods for equality constrained optimization: Equivalence of existing methods and a new implementation. In *Nonlinear Programming 3*, pages 125–164. Elsevier, 1978.

36. P. Toint. On sparse and symmetric matrix updating subject to a linear equation. *Mathemathics of Computation*, 31(140):954–961, 1977.
37. R.B. Wilson. *A simplicial algorithm for concave programming*. PhD thesis, Graduate School of Business Administration, Harvard University, 1963.