Mathematical Programming manuscript No.
(will be inserted by the editor)

Convex Optimization with ALADIN

Boris Houska · Dimitris Kouzoupis · Yuning Jiang · Moritz Diehl

the date of receipt and acceptance should be inserted later

Abstract This paper presents novel convergence results for the Augmented Lagrangian based Alternating Direction Inexact Newton method (ALADIN) in the context of distributed convex optimization. It is shown that ALADIN converges for a large class of convex optimization problems from any starting point to minimizers without needing line-search or other globalization routines. Under additional regularity assumptions, ALADIN can achieve superlinear or even quadratic local convergence rates. The theoretical and practical advantages of ALADIN compared to other distributed convex optimization algorithms such as dual decomposition and the Alternating Direction Method of Multipliers (ADMM) are discussed and illustrated by numerical case studies.

1 Introduction

Modern algorithms and software for distributed convex optimization are often based on dual decomposition [12] or on the Alternating Direction Method of Multipliers (ADMM) [17,18]. There exist excellent review articles about both types of methods [2, 6]. Therefore, the following review focusses only on a small number of selected articles that are relevant in the context of the current paper.

Dual decomposition algorithms exploit the fact that the Lagrangian dual function of minimization problems with a separable, strictly convex objective function and affine coupling constraints can be evaluated by solving decoupled optimization problems. One way to solve the associated dual ascent problem is by using a gradient or an accelerated gradient method as suggested in a variety of articles [33, 40, 41]. Other methods for solving the dual ascent problem are based on semi-smooth Newton methods [13, 15, 30], which have the disadvantage that additional smoothing heuristics and line-search routines are needed, as the dual function of most practically relevant convex optimization problem is 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` [16].

A powerful alternative to dual decomposition algorithms is the alternating direction method of multipliers (ADMM), which has originally been introduced in [17,18]. 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. Nowadays, ADMM is accepted as one of the most promising methods for distributed optimization and it has been analyzed by many authors [7,10, 11,19]. In particular [6] contains a self-contained convergence proof of ADMM 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 [26] 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 (SQP) [4,36,37,46] 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 [45]. In particular, limited memory BFGS (L-BFGS) methods are highly competitive candidates for large-scale optimization [34]. As an alternative class of large-scale optimization methods, augmented Lagrangian methods [1,25,35,44] have been analyzed and implemented in the software collection `GALAHAD` [8,20]. A more exhaustive review of such augmented Lagrangian based decomposition methods for convex and non-convex optimization algorithms can be found in [24].

One of the main differences between ADMM and general Newton-type methods, specifically SQP methods in the context of optimization, is that ADMM is not invariant under scaling [6]. 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 iterates may converge slowly otherwise. A similar statement holds for the standard variant of dual decomposition, if the dual ascent problem is solved with a gradient or an accelerated gradient method [16,41]. In general, 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 method can be considered as a practically relevant example for an optimization algorithm, which attempts to approximate second order terms while the algorithm is running, and which is competitive for large-scale optimization when compared to purely gradient based methods. Another example for a practical large-scale optimization method based on forward-backward splitting has been proposed in [43], where a variable-metric proximal gradient method is analyzed leading to promising numerical performance. If sparse or low-rank matrix approximation techniques, as for example used in SQP or more general Quasi-Newton 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.

The goal of this paper is to seek for the simplest possible scaling invariant decomposition method that converges fast and reliably to minimizers of general convex optimization problems. Here, the practical motivation is that we would like to implement a generic method that does not require, at least by default, users to provide meta-data about their problem formulation such as Lipschitz constants or variable scales. Moreover, a principal goal of this paper is to develop a distributed optimization algorithm that aims at minimal communication by avoiding expensive global sub-routines such as line-search. The developments in this paper are based on a variant of the recently proposed Augmented Lagrangian based Alternating Direction Inexact Newton method (ALADIN), which has originally been developed for solving distributed non-convex optimization problems [28].

1.1 Contribution

Section 2 proposes a variant of the distributed optimization algorithm ALADIN [28, 29]. By default, this ALADIN variant[1] can be used to find minimizers of any convex, closed, and proper (potentially non-differentiable) convex optimization problem. A corresponding proof of convergence under mild regularity assumptions can be found in Section 3. However, as mentioned above, a principal motivation of using ALADIN rather than dual decomposition or ADMM is that it can exploit derivative information (whenever available) and, therefore, Section 4 discusses implementation details and local convergence rate estimates with a particular focus on how to approximate Hessian matrices in the context of ALADIN. Here, the main motivation is to develop a generic and globally convergent distributed optimization framework that can benefit from mature matrix approximation methods such as low-rank matrix approximation by directional algorithmic differentiation for second order derivatives in forward or adjoint mode [9,21,27,39], Gauss-Newton Hessian approximation methods [3,34], or BFGS and L-BFGS methods [31,34]. Examples and numerical comparison of ALADIN versus ADMM can be found in Section 5.

1.2 Notation

We use the symbols $\mathbb{S}^n_+$ and $\mathbb{S}^n_{++}$ to denote the set of symmetric, positive semi-definite and symmetric, positive definite matrices in $\mathbb{R}^{n\times n}$. The pseudo inverse of a matrix $M \in \mathbb{S}^n_+$ is denoted by $M^\dagger$. For a given matrix $\Sigma \in \mathbb{S}^n_+$ the notation

$$\|x\|_\Sigma = \sqrt{x^\mathsf{T}\Sigma x}$$

is used, although this notation should be used with caution, since $\|\cdot\|_\Sigma$ is only a norm if $\Sigma \succ 0$. Moreover, we call a function $c : \mathbb{R}^n \to \mathbb{R}\cup\{\infty\}$ strongly convex with matrix parameter $\Sigma \in \mathbb{S}^n_+$, if the inequality

$$c(tx + (1-t)y) \leq tc(x) + (1-t)c(y) - \frac{1}{2}t(1-t)\|x-y\|^2_\Sigma$$

[1] The ALADIN algorithm itself has been proposed in a very similar form in earlier work by some of the authors [28,29] and, consequently, the introduction of this algorithm is not an original contribution of this paper, but all convergence results in Section 3, the developments in Section 4, and the numerical results in Section 5 are new.

is satisfied for all $x, y \in \mathbb{R}^n$ and all $t \in [0,1]$. Notice that this definition contains the standard definition of convexity as a special case for $\Sigma = 0$. Similarly, c is called strictly convex, if there exists a continuous and strictly monotonously increasing function $\alpha : \mathbb{R} \to \mathbb{R}$ with $\alpha(0) = 0$ such that

$$c(tx + (1-t)y) \leq tc(x) + (1-t)c(y) - \frac{1}{2}t(1-t)\alpha\left(\|x-y\|\right)$$

is satisfied for all $x, y \in \mathbb{R}^n$ and all $t \in [0,1]$. Notice that all convex functions in this paper are assumed to be closed and proper [5]. Finally, for a convex function c the notation

$$\partial c(x) = \left\{ z \in \mathbb{R}^n \mid \forall y \in \mathbb{R}^n,\ c(x) + z^\mathsf{T}(y-x) \leq c(y) \right\}$$

is used to denote the subdifferential of c at x. Recall that x^* is a minimizer of a proper convex function c if and only if $0 \in \partial c(x^*)$, see [42].

2 Convex ALADIN

This section concerns structured convex 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 $f_i : \mathbb{R}^{n_i} \to \mathbb{R} \cup \{\infty\}$ are assumed to be proper, closed, and convex and the matrices $A_i \in \mathbb{R}^{m \times n_i}$ 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. Thus, the Lagrangian of (1) is given by

$$L(x, \lambda) \ = \ \sum_{i=1}^N f_i(x_i) + \lambda^\mathsf{T} \left(\sum_{i=1}^N A_i x_i - b \right) \ .$$

Algorithm 1 outlines a derivative-free variant of ALADIN [28] for solving (1). In the case where the functions f_i are differentiable, the optimality condition for the decoupled NLPs in Step 1 can be written in the form

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

Consequently, if the objective functions f_i are differentiable, the QP gradients, given by $g_i = \nabla f_i(y_i)$, are equal to the gradients of f_i. However, for non-differentiable f_i, we only know that $g_i \in \partial f_i(y_i)$, where $\partial f_i(y_i)$ denotes the subdifferential of f_i at the point y_i. Thus, if the termination criterion in Step 4 of Algorithm 1 is satisfied, we have

$$-A_i^\mathsf{T}\lambda \in \partial f_i(y_i) + \mathbf{O}\left(\epsilon\right) \ .$$

Moreover, since we have $\sum_{i=1}^N A_i x_i = b$, we also have

$$\sum_{i=1}^N A_i y_i - b = \mathbf{O}\left(\epsilon\right)$$

upon termination, i.e., y satisfies the stationarity and primal feasibility condition of (1) up to a small error of order $\mathbf{O}\left(\epsilon\right)$.

Algorithm 1: Derivative-free ALADIN

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

Initialization:

- Project the initial guess (if necessary) such that $\sum_{i=1}^N A_i x_i = b$.
- Choose initial scaling matrices $H_i \in \mathbb{S}^{n_i}_{++}$.

Repeat:

1. Solve for all $i \in \{1, \ldots, N\}$ the decoupled NLPs
$$\min_{y_i} \; f_i(y_i) + \lambda^\mathsf{T} A_i y_i + \frac{1}{2} \|y_i - x_i\|^2_{H_i} \; .$$
In all following steps, y_i denotes an optimal solution of the above NLP.
2. Set $g_i = H_i(x_i - y_i) - A_i^\mathsf{T} \lambda$.
3. Optionally choose new scaling matrices $H_i \in \mathbb{S}^{n_i}_{++}$ (update).
4. If $\|x_i - y_i\| \leq \epsilon$ for all $i \in \{1, \ldots, N\}$, terminate.
5. Solve the coupled equality constrained QP
$$\min_{\Delta y} \sum_{i=1}^N \left\{ \frac{1}{2} \Delta y_i^\mathsf{T} H_i \Delta y_i + g_i^\mathsf{T} \Delta y_i \right\} \quad \text{s.t.} \quad \sum_{i=1}^N A_i(y_i + \Delta y_i) = b \;\; | \;\; \lambda^+ \; .$$
In all following steps, Δy denotes a minimizer of this QP and λ^+ denotes an associated dual solution.
6. Set $x \leftarrow x^+ = y + \Delta y$ and $\lambda \leftarrow \lambda^+$ and continue with Step 1.

Remark 1 *Notice that the initial projection step in Algorithm 1 is introduced for simplicity of presentation only. If we skip this projection step, the coupled QP in Step 5 ensures that the updated iterates $x^+ = y + \Delta y$ satisfy*

$$\sum_{i=1}^N A_i x_i^+ = b \; .$$

Thus, after the first ALADIN iteration the iterates x_i are anyhow feasible, i.e., if the initial projection step is skipped, all the analysis results in the remainder of this paper are valid from the second ALADIN iteration on. ◇

2.1 Implementation details

Steps 1, 2, 3, and 6 of Algorithm 1 are completely decoupled. However, Step 4 and Step 5 require communication between the agents. As the inequalities in Step 4 can be verified in a distributed way, the agents only need to communicate single bits in order to check on whether they agree to terminate. Thus, let us have a closer look on the more expensive Step 5, where a coupled QP has to be solved. This coupled QP can be solved by first computing the sums

$$r \; = \; \sum_{i=1}^N A_i y_i - b \qquad \text{and} \qquad M \; = \; \sum_{i=1}^N A_i H_i^{-1} A_i^\mathsf{T} \; . \tag{2}$$

Here, M is called the dual Hessian matrix, which is known to be invertible, if the matrices A_i satisfy the linear independence constraint qualification (LICQ), i.e., if the matrix $[A_1, A_2, \dots, A_N]$ has full-rank. Next, the dual solution of the QP, given by[2]

$$\lambda^+ = M^\dagger \left[r - \sum_{i=1}^{N} H_i^{-1} g_i \right] , \tag{3}$$

needs to be communicated to all agents. Finally, the primal solution of the coupled QP can be found without additional communication, as

$$x_i^+ \; = \; y_i + \Delta y_i \; = \; y_i - H_i^{-1} \left(g_i + A_i^\mathsf{T} \lambda^+ \right) . \tag{4}$$

Notice that the above equations can be derived by writing out the optimality conditions of the coupled QP from Step 5. Of course, 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, e.g., in the initialization routine. However, one of the main motivations for developing Algorithm 1 is its 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.[3]

Remark 2 *If the matrices H_i are not updated in Step 3, then we can substitute the gradient equation*

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

in (3) *and* (4)*, which leads to the equations*

$$\lambda^+ = \lambda + M^\dagger \left[2r - r_x \right] \quad \textit{with} \quad r_x = \sum_{i=1}^{N} A_i x_i - b \tag{5}$$

and

$$x_i^+ \; = \; y_i + \Delta y_i \; = \; 2y_i - x_i - H_i^{-1} A_i^\mathsf{T} (\lambda^+ - \lambda) . \tag{6}$$

Moreover, (5) *can be simplified further, since we have $r_x = 0$ during all ALADIN iterations under the assumption that the initial projection step is implemented, i.e., we have*

$$\lambda^+ = \lambda + 2M^\dagger r . \tag{7}$$

If the initial projection step is skipped, as discussed in Remark 1, we implement (5) *in the first ALADIN iteration, but then use* (7) *from the second ALADIN iteration on.* $\diamond$

[2] If LICQ does not hold, we have $M\lambda^+ = r - \sum_{i=1}^{N} H_i^{-1} g_i$. Under the additional assumption that the linear coupling constraint is feasible, this implies that (7) also holds if LICQ is violated, but in this case M is not invertible and the pseudo-inverse of M has to be used.

[3] If low-rank updates such as BFGS are applied to the matrices H_i, the Sherman-Morrison Woodbury formula can be used to derive associated formulas for updating $M^\dagger$ directly [22].

2.2 Relation to sequential quadratic programming

In order to motivate why Algorithm 1 has favorable local convergence properties, we assume for a moment that the objective functions f_i are twice differentiable and that the second derivatives, denoted by $\nabla^2 f_i$, are Lipschitz continuous functions. Of course, this is a very restrictive assumption on the functions f_i, which we will drop again later on in the paper. The only purpose of this assumption is that it allows us to illustrate why ALADIN inherits the favorable local convergence properties of sequential quadratic programming (SQP) methods. More precisely, an "Exact Hessian ALADIN" algorithm is obtained by setting $H_i = \nabla^2 f_i(y_i)$ in Step 3 of the proposed algorithm such that the coupled QP in Step 5 of Algorithm 1 is very similar to the QPs that would be solved when applying an exact Hessian SQP method for solving the original coupled NLP.

Proposition 1 *Let the functions f_i be strongly convex for parameters $\Sigma_i \succ 0$ and twice Lipschitz-continuously differentiable and let the LICQ condition for NLP* (1) *be satisfied such that its minimizer x^* is unique. If we set $H_i = \nabla^2 f_i(y_i)$ in Step 3 of Algorithm 1, then this algorithm converges with locally quadratic convergence rate, i.e., 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 [28]. Therefore, we only briefly recall the two main steps of the proof. In the first step, we analyze the decoupled NLPs in Step 1 of Algorithm 1. The solutions y_i of these NLPs are unique and satisfy the second order sufficient optimality condition, since we assume that the functions f_i are strongly convex and twice continuously differentiable. Moreover, as the LICQ condition is satisfied, the dual solution λ^* of the coupled optimization problem is also unique. Next, a standard result from parametric nonlinear programming can be used (e.g., Proposition 4.2.3 in [2]), to show that there exist constants $\chi_1, \chi_2 < \infty$ such that the solution of the decoupled NLPs satisfy

$$\left\|y - x^*\right\| \leq \chi_1 \left\|x - x^*\right\|_2 + \chi_2 \left\|\lambda - \lambda^*\right\|_2 \tag{8}$$

for all x, λ in the neighborhood of (x^*, λ^*) (see Lemma 3 in [28] for a proof). In the second step, we analyze the coupled QP, which can be written in the form

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

since we may substitute $g_i = \nabla f_i(y_i)$ and $H_i = \nabla^2 f_i(y_i)$. Next, since the functions $\nabla^2 f_i$ are Lipschitz continuous, we can apply a result from standard SQP theory [34] to show that there exists a constant $\chi_3 < \infty$ such that

$$\left\|x^+ - x^*\right\| \leq \chi_3 \left\|y - x^*\right\|^2 \qquad \text{and} \qquad \left\|\lambda^+ - \lambda^*\right\| \leq \chi_3 \left\|y - x^*\right\|^2 . \tag{9}$$

The statement of the theorem follows now by combining (8) and (9). ◇

2.3 Tutorial example

In order to understand the local and global convergence properties of Algorithm 1, let us consider the tutorial optimization problem

$$\min_x \ \frac{1}{2}q_1 x_1^2 + \frac{1}{2}q_2(x_2-1)^2 \quad \text{s.t.} \quad x_1 - x_2 = 0 \ \mid \ \lambda \tag{10}$$

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

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

The initial projection step of Algorithm 1 ensures that $x_1 = x_2 = z$. Thus, if we choose $H_1 = H_2 = H > 0$ the subproblems from Step 1 of Algorithm 1 can be written in the form

$$\min_{y_1} \frac{1}{2}q_1 y_1^2 + \lambda y_1 + \frac{H}{2}(y_1 - z)^2 \quad \text{and} \quad \min_{y_2} \frac{1}{2}q_2(y_2-1)^2 - \lambda y_2 + \frac{H}{2}(y_2 - z)^2 \ .$$

The explicit solution of these subproblems is given in this case by

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

Next, working out the solution of the QP in Step 5 (z^+, λ^+), with $z^+ = x_1^+ = x_2^+$, yields

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

with contraction matrix

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

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

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

Notice that for any $H > 0$ these eigenvalues are contained in the open unit disc, i.e., ALADIN converges for this example 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 1. Notice that ALADIN converges for this example after at most two iterations (and therefore terminates at latest during the third iteration) if $H = q_1$ or if $H = q_2$, as C is a nilpotent matrix in this case. The goal of the following sections is to establish similar convergence properties for general convex optimization problems.

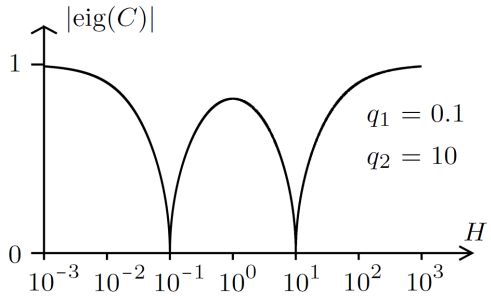

Fig. 1 The absolute value of the maximum eigenvalue, $|\mathrm{eig}(C)|$, of the contraction matrix C versus the scaling $H \in [10^{-3}, 10^{3}]$ for $q_1 = 0.1$ and $q_2 = 10$. Notice that we have $|\mathrm{eig}(C)| < 1$ for all $H > 0$, i.e., ALADIN converges for all choices of H. However, the method 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.

3 Convergence

This section presents the main contribution of this paper, namely, a proof of the fact that the iterates y of Algorithm 1 converge (under mild technical assumptions) to the set of minimizers of the optimization problem (1), if the functions f_i are closed, proper, and convex. This result is relevant, since it implies that ALADIN converges for a large class of convex optimization problems from any starting point and without needing a line-search or other globalization routines. The main technical idea for establishing this result is to introduce the function

$$\mathcal{L}(x, \lambda) = \left\|\lambda - \lambda^*\right\|_M^2 + \sum_{i=1}^{N} \left\|x_i - x_i^*\right\|_{H_i}^2 \tag{11}$$

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). An important monotonicity property of this function $\mathcal{L}$ along the iterates of Algorithm 1 is established next. The following theorem uses the shorthand notation

$$\left\|y - y^*\right\|_\Sigma^2 = \sum_{i=1}^{N} \left\|y_i - y_i^*\right\|_{\Sigma_i}^2$$

with $y^* = x^*$. Recall that $\|\cdot\|_\Sigma$ is not necessarily a norm if one of the Σ_is is only positive semi-definite.

Theorem 1 *Let the functions f_i be closed, proper, and strongly convex with parameter $\Sigma_i \in \mathbb{S}_+^{n_i}$ and let problem (1) be feasible such that strong duality holds, and let the matrices H_i be positive definite and not updated in the current iteration. If $x^* = y^*$ denotes the primal and λ^* a (not necessarily unique) dual solution of (1), then the iterates of Algorithm 1 satisfy the inequality*

$$\mathcal{L}(x^+, \lambda^+) \leq \mathcal{L}(x, \lambda) - 4\left\|y - y^*\right\|_\Sigma^2 . \tag{12}$$

Proof. The optimality conditions for the coupled QP in Step 5 of Algorithm 1 can be written in the form

$$M\lambda^+ = M\lambda + 2r \tag{13}$$

$$\text{and} \qquad x_i^+ = 2y_i - x_i - H_i^{-1}A_i^\mathsf{T}(\lambda^+ - \lambda)\ , \tag{14}$$

where $r = \sum_{i=1}^N A_i y_i - b$ denotes the constraint residuum, as discussed in Remark 2. Next, the optimality condition for the decoupled NLPs can be written in the form

$$\begin{aligned} 0 \ &\in \ \partial f_i(y_i) + A_i^\mathsf{T}\lambda + H_i(y_i - x_i) \\ &\overset{(14)}{=} \partial f_i(y_i) + A_i^\mathsf{T}\lambda^+ + H_i(x_i^+ - y_i) \end{aligned}$$

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

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

Now, since f_i is strongly convex with parameter $\Sigma \succeq 0$, the function F inherits this property and we find

$$F_i(y_i) \leq F_i(y_i^*) - \frac{1}{2}\left\|y_i - y_i^*\right\|_{\Sigma_i}^2\ .$$

Adding the above inequalities for all i, substituting the definition of F_i, and rearranging terms yields

$$\begin{aligned} \sum_{i=1}^N \left\{f_i(y_i) - f_i(y_i^*)\right\} &\leq \sum_{i=1}^N \left(A_i^\mathsf{T}\lambda^+ + H_i(x_i^+ - y_i)\right)^\mathsf{T}(y_i^* - y_i) - \frac{\|y - y^*\|_\Sigma^2}{2} \\ &= -r^\mathsf{T}\lambda^+ + \sum_{i=1}^N (x_i^+ - y_i)^\mathsf{T} H_i(y_i^* - y_i) - \frac{\|y - y^*\|_\Sigma^2}{2}\ . \end{aligned} \tag{15}$$

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

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

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

$$G(y^*) \leq G(y) - \frac{\|y - y^*\|_\Sigma^2}{2}\ . \tag{16}$$

This inequality can be written in the equivalent form

$$-r^\mathsf{T}\lambda^* + \frac{1}{2}\left\|y - y^*\right\|_\Sigma \leq \sum_{i=1}^N \left\{f_i(y_i) - f_i(y_i^*)\right\}\ . \tag{17}$$

By combining (15) and (17), and sorting terms it follows that

$$
\begin{aligned}
-\left\|y-y^*\right\|_\Sigma^2 \;&\geq\; r^\mathsf{T}(\lambda^+-\lambda^*)+\sum_{i=1}^N (y_i-y_i^*)^\mathsf{T} H_i(x_i^+-y_i) \\
&\overset{(13)}{=}\; \frac{1}{2}(\lambda^+-\lambda)^\mathsf{T} M(\lambda^+-\lambda^*)+\sum_{i=1}^N (y_i-x_i^*)^\mathsf{T} H_i(x_i^+-y_i)\;.
\end{aligned}
$$

The sum term on the right hand of the above equation can be written in the form

$$
\begin{aligned}
&\textstyle\sum_{i=1}^N (y_i-x_i^*)^\mathsf{T} H_i(x_i^+-y_i) \\
&\overset{(14)}{=} \textstyle\frac{1}{4}\sum_{i=1}^N \left(x_i^+ + x_i - 2x_i^* + H_i^{-1}A_i^\mathsf{T}(\lambda^+-\lambda)\right)^\mathsf{T} H_i \left(x_i^+ - x_i - H_i^{-1}A_i^\mathsf{T}(\lambda^+-\lambda)\right) \\
&= \textstyle -\frac{1}{4}(\lambda^+-\lambda)^\mathsf{T} M(\lambda^+-\lambda) + \frac{1}{4}\sum_{i=1}^N \left\|x_i^+-x_i^*\right\|_{H_i}^2 - \frac{1}{4}\sum_{i=1}^N \left\|x_i-x_i^*\right\|_{H_i}^2 \;.
\end{aligned}
$$

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

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

This inequality is equivalent to (12), which corresponds to the statement of the theorem. ◇

Remark 3 *In the case where the functions f_i are strictly convex (but not necessarily strongly convex for a parameter $\Sigma_i \succ 0$), the iterates of ALADIN satisfy the inequality*

$$
\mathcal{L}(x^+,\lambda^+) \leq \mathcal{L}(x,\lambda) - \alpha\left(\left\|y-y^*\right\|\right)\;, \tag{18}
$$

where $\alpha : \mathbb{R} \to \mathbb{R}$ is a continuous and strictly monotonously increasing function that satisfies $\alpha(0) = 0$. This result can be established by replacing all terms of the form "$\frac{1}{2}\|y-y^*\|_\Sigma^2$" in (15) *and* (17) *in the proof of Theorem 1 by "$\frac{1}{8}\alpha\left(\|y-y^*\|\right)$" and setting $\Sigma_i = 0$. This is always possible if the functions f_i are strictly convex. The rest of the proof of is then analogous to the arguments in the proof of Theorem 1 and* (18) *follows.*

Remark 4 *By comparing the proof technique from the above theorem with the convergence proof for ADMM that has been proposed in [6], there are a number of similarities in the sense that the convergence proof also uses inequalities that are similar to* (15) *and* (17)*. However, the resulting descent properties (or non-expansiveness) properties are different. Of course, ALADIN has been inspired by the developments in the field of ADMM and Algorithm 1 has been constructed in such a way that some of the convergence proof ideas from ADMM carry over to ALADIN. However, ALADIN and ADMM are in this form neither equivalent nor special cases of each other and should therefore, despite many similarities, be regarded as different algorithms. The advantage of ALADIN over ADMM will become clearer in the remainder of this paper.*

Notice that Theorem 1 (or alternatively the inequality in Remark 3) 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 intermediate 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 such that strong duality holds. If the matrices $H_i \succ 0$ are kept constant during the iterations, then the primal iterates y of Algorithm 1 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 (18) that the function $\mathcal{L}$ is strictly monotonously decreasing whenever $y \neq y^*$. As $\mathcal{L}$ is a non-negative function, i.e., bounded below by 0, the value of $\mathcal{L}(x, \lambda)$ must converge as the algorithms progresses, but this is only possible if y converges to y^*. ◇

The statement of Corollary 1 is already quite 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. Nevertheless, there are two important questions that have not been addressed so far:

1. What happens if the functions f_i are only convex but not necessarily strictly convex?
2. What happens if we update the matrices H_i during the iterations?

In order to address the first question, it should be noted first that if the functions f_i are only known to be convex, the set of minimizers of (1) is in general not a singleton. Let Λ^* denote the set of dual solutions of (1). In the following, we denote with $\mathbb{Y}^*$ the set of stationary points of (1), which is defined to be the set of points y^*, which satisfy the stationarity condition

$$\forall \lambda^* \in \Lambda^*, \quad 0 \in \sum_{i=1}^{N} \left\{ \partial f_i(y_i^*) + A_i^\mathsf{T} \lambda^* \right\} .$$

Notice that for many practical convex optimization problems, $\mathbb{Y}^*$ coincides with the set of minimizer of (1), although it is possible to construct counter-examples [5].

Theorem 2 *Let the functions f_i be closed, proper, and convex and let problem* (1) *be feasible such that strong duality holds. Let $\mathbb{Y}^*$ be the set of stationary points of* (1) *as defined above. 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 us pick any $\lambda^* \in \operatorname{relint}(\Lambda^*)$, i.e., such that λ^* is in the relative interior of Λ^*. Next, we define the function $\mathcal{L}$ as above but by choosing any minimizer $x^* = y^*$ of (1). The main idea of the proof of this theorem is to have a closer look at the auxiliary function

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

which has already been used in the proof of Theorem 1. 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, due to the definition of $\mathbb{Y}^*$ and since $\lambda^* \in \operatorname{relint}(\Lambda^*)$, we know that $G(y) = G(y^*)$ if and only if $y \in \mathbb{Y}^*$. 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 1, we find that this implies

$$\mathcal{L}(x^+, \lambda^+) \leq \mathcal{L}(x, \lambda) - \alpha\left(\min_{z \in \mathbb{Y}^*} \|y - z\|\right) . \tag{19}$$

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 2 only prove that the iterates y of Algorithm 1 converge to a minimizer or at least to the set of stationary points of Problem 1 (if the termination check is skipped), but no statement is made about the convergence of the iterates x and λ. Thus, we have not formally proven yet that Algorithm 1 terminates for any given $\epsilon > 0$ after a finite number of iterations. Unfortunately, additional regularity assumptions are needed in order to be able to establish convergence of x and λ. The following Lemma shows that x does converge if the functions f_i are continuously differentiable. Of course, this is a strong regularity assumption, which might not hold in practice. However, the purpose of Lemma 1 is to explain the main idea how one can establish convergence of x and λ without needing much analysis overhead. Later, in Section 4, we will explain how to enforce convergence of x and λ under much weaker (but more technical) assumptions, which avoid the assumption that the functions f_i are differentiable.

Lemma 1 *If the conditions of Theorem 2 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

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

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

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

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

4 Matrix updates

This section addresses the question how to update the matrices H_i. The main motivation for these updates is that they might improve the local convergence rate of ALADIN as discussed in Proposition 1. However, there are also two disadvantages of such matrix updates, which should be kept in mind. Firstly, as discussed in Section 2.1, updating the matrices H_i has the disadvantage that the dual Hessian matrix, given by,

$$M = \sum_{i=1}^{N} A_i H_i^{-1} A_i^\mathsf{T} \,,$$

has to be re-computed after every update. This requires additional communication between the agents, which may be expensive. Secondly, if we do not keep the matrices H_i constant, the function $\mathcal{L}$, which has been defined in the previous section, is re-defined after every matrix update, i.e., the global convergence proof from Corollary 1 is, at least in this form, not applicable anymore. The goal of the following sections is to develop tailored matrix update strategies, which remedy these problems.

4.1 Merit functions

The main idea for establishing convergence of ALADIN with matrix updates, is to control the frequency of the updates. Here, the progress of the algorithm is measured by using merit functions. Merit functions, such as L1-penalties, have originally been developed in the context of globalization in nonlinear programming and SQP [23,34]. In the following, we denote by

$$\Phi(y) = \sum_{i=1}^{N} f_i(y_i) + \bar{\lambda} \left\| \sum_{i=1}^{N} A_i y_i - b \right\|_1$$

the standard L1-merit function. If $\bar{\lambda} < \infty$ is a sufficiently large constant, then the minimizers of the function Φ coincide with the minimizers of (1) under mild regularity assumptions [23]. In the context of SQP, the function Φ is often used as a starting point to develop line-search routines [34]. However, as mentioned in the introduction, the goal of this paper is to avoid line-search and, thus, our motivation for introducing the function Φ is of a different nature, namely, to control the frequency of matrix updates, as elaborated below. Notice that a single evaluation of Φ at the current iterate y comes at basically no extra communication cost, as we have to compute the primal constraint residuum

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

anyhow when solving the coupled QP in Step 5.

Remark 5 *Unfortunately, the rigorous construction of a constant $\bar{\lambda} < \infty$ requires meta-data from the user, in the sense that Φ is only an exact penalty function, if $\bar{\lambda}$ is an (a-priori) upper bound on the ∞-norm of the dual solutions of* (1)*. However, state-of-the-art SQP solvers [34] implement effective heuristics for choosing $\bar{\lambda}$ online in order to avoid requiring meta-data from the user and such heuristics also can be used in the context of the current paper.* ◇

4.2 Matrix update policies

A simple but effective matrix update policy that ensures global convergence of Algorithm 1 is outlined next.

1. When we run the first ALADIN iteration, we store the merit function value
$$\gamma = \Phi(y) \ .$$
During this first step we might or might not update H_i.
2. Next, from the second iteration on, we run ALADIN without updating the matrices H_i (i.e., Step 3 of Algorithm 1 is skipped) until the condition
$$\Phi(y) \leq \gamma - \hat{\epsilon} \tag{20}$$
is satisfied for a small constant $\hat{\epsilon}$ in the order of the machine precision (or NLP solver accuracy). If the conditions from Corollary 1 are satisfied and if Φ is an exact penalty function, this must happen after a finite number of iterations, as ALADIN converges for fixed matrices H_i. Thus, once the above condition is met, we may reset $\gamma = \Phi(y)$, update all H_is but then continue the iterations again without updates until the above condition is satisfied for the second time, and so on.

The above strategy has the advantage that it ensures that the convergence statement from Corollary 1 also applies to ALADIN with matrix updates. This follows simply from the fact that Φ is an exact penalty function that descents after every finite number of iterations by a small constant. Of course, (20) should be interpreted as the weakest possible descent condition that can be checked with finite precision arithmetic and which works well in practice. One could also replace (20) by—at least from a theoretical perspective—more appealing descent conditions that involve the directional derivative of Φ, for example, analogous to standard Armijo line search conditions [34]. However, such conditions are more expensive to check and therefore not further investigated in the current paper. At this point, we highlight once more that the above strategy only controls the frequency of matrix updates but it never rejects steps or slows down the progress of the ALADIN iterations. This is in contrast to line-search routines, which are typically implemented by back-tracking or other heuristics which may lead to step rejections and additional computations.

Remark 6 *If the iterates y of the Algorithm 1 contract with locally superlinear or quadratic rate, i.e., $\|y^+ - y^*\| \leq c\|y^+ - y^*\|$ with $c \to 0$, the function Φ descents in every step in the neighborhood of a regular minimizer [34], i.e., inside such a neighborhood* (20) *is always satisfied and does not prevent matrix updates. For more details*

of this statement we refer to the literature on the so-called Maratos effect [32, 14, 38], which does however not occur in our convex setting. In particular, the local convergence statement from Proposition 1 is not affected by the above outlined matrix update controller. ◇

4.3 ALADIN with explicit inequalities

The proposed ALADIN algorithm can also be used to solve optimization problems of the form

$$\min_x \sum_{i=1}^N \tilde{f}_i(x_i) \quad \text{s.t.} \quad \begin{cases} \sum_{i=1}^N A_i x_i = b \mid \lambda \\ \tilde{h}_i(x_i) \leq 0 \quad \mid \kappa_i \end{cases} \tag{21}$$

for convex functions $\tilde{f}_i$ and $\tilde{h}_i$. In order to do this, we only need to define the functions

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

and run Algorithm 1 without further modifications. Of course, all global convergence statements for the iterates y, in particular Corollary 1 and Theorem 2, are fully applicable to this case and ALADIN can be applied as usual, since Algorithm 1 does not require the computation of derivatives at any point. Nevertheless, in this context, there are two issues that need to be discussed:

1. If the functions $\tilde{f}_i$ and $\tilde{h}_i$ are twice Lipschitz-continuously differentiable, one might wish to update the matrices H_i in such a way that quadratic convergence in the local neighborhood of regular minimizers can be established. Unfortunately, the functions f_i are, however, not differentiable in general and, consequently, the statement of Proposition 1 is—at least in this form—not applicable.
2. Even if the functions $\tilde{f}_i$ and $\tilde{h}_i$ are continuously differentiable, the function f_i are not differentiable in general and, consequently, the statement of Lemma 1 cannot be used directly to establish finite termination of Algorithm 1 with an ϵ-optimal solution.

In order to address the first problem under the assumptions that the functions $\tilde{f}_i$ and $\tilde{h}_i$ are twice Lipschitz-continuously differentiable, we run ALADIN with scaling matrices of the form

$$H_i = \nabla^2 \left\{ \tilde{f}_i(y_i) + \kappa_i^\mathsf{T} \tilde{h}_i(y_i) \right\} + \mu \left(C_i^{\text{act}}(y_i) \right)^\mathsf{T} C_i^{\text{act}}(y_i) \,,$$

where y_i is the solution of the i-th decoupled NLP. Moreover, κ_i denotes the multiplier that is associated with the inequalities, $\mu > 0$ is a tuning parameter, and

$$C_i^{\text{act}}(y_i) = \nabla \tilde{h}_i^{\text{act}}(y_i)$$

is the constraint Jacobian of the constraints that are strongly active at y_i. Of course, this heuristic for choosing H_i does in general not lead to a locally quadratically convergent algorithm, but it can nevertheless be shown that, if μ is sufficiently large and if the minimizer x^* is a regular KKT point, ALADIN is locally equivalent to an inexact

Newton method that converges superlinearly if μ tends to $+\infty$. The corresponding argumentation is analogous to Proposition 1, as we can rely on standard results from SQP theory [34]. Notice that SQP theory also resolves the second problem: since, SQP methods converge in a neighborhood of regular KKT points, we must have $\Delta y \to 0$, i.e., $\|x - y\| \to 0$, which implies finite termination of Algorithm 1 for any given $\epsilon > 0$ under the assumption that the minimizer is such a regular KKT point.

5 Numerical Case Studies

This section presents numerical results, which confirm the theoretical convergence properties of ALADIN for convex optimization problems and also illustrate the practical advantages of ALADIN compared to ADMM.

5.1 Lasso

This section compares the performance of ALADIN versus ADMM for lasso problems of the form

$$\min_x \ \frac{1}{2} \|Ax - b\|_2^2 + \kappa \|x\|_1 \ . \tag{22}$$

Here, we choose the matrices $A \in \mathbb{R}^{10\times 100}$ and the vector $b \in \mathbb{R}^{10}$ randomly in order to generate a large number of test problems. The constant $\kappa > 0$ is a scalar regularization parameter. As suggested in [6] we write (22) in the form

$$\min_x \ f_1(x_1) + f_2(x_2) \quad \text{s.t.} \quad x_1 - x_2 = 0 \ \mid \ \lambda \, , \tag{23}$$

where $f_1(x_1) = \frac{1}{2} \|Ax_1 - b\|_2^2$ and $f_2(x_2) := \kappa \|x_2\|_1$ are the contributions from the different norms. As shown in [6] the standard ADMM algorithm for (23) can be written in the form

$$\text{ADMM:} \quad \left\{ \begin{array}{l} x_1^+ = \left(A^\top A + \rho I\right)^{-1} \left(A^\top b + \rho \left(x_2 - \lambda\right)\right) \\ x_2^+ = S_{\kappa/\rho}\left(x_1^+ + \lambda\right) \\ \lambda^+ = \lambda + x_1^+ - x_2^+ \end{array} \right\} \, . \tag{24}$$

Here, the shorthand notation

$$S_{\kappa/\rho}(v) := \max(0, v - \kappa/\rho \mathbf{1}) - \max(0, -v - \kappa/\rho \mathbf{1}) \, , \tag{25}$$

is used, where $\mathbf{1} \in \mathbb{R}^n$ denotes a vector of ones and $\max(\cdot, \cdot)$ the element-wise maximum of two vectors. Similarly, if we set $H_1 = A^\top A$ and $H_2 = \rho I$, the ALADIN iterates can

be written in the form[4]

$$\text{ALADIN:} \quad \left\{ \begin{array}{l} y_1 = x_1 - \frac{1}{2}\left(A^\top A\right)^\dagger \left(\lambda + A^\top (Ax_1 - b)\right) \\ y_2 = S_{\kappa/\rho}\left(x_2 + \rho^{-1}\lambda\right) \\ \lambda^+ = \lambda + 2\rho A^\top A \left(A^\top A + \rho I\right)^{-1} (y_1 - y_2) \\ x_1^+ = x_2^+ = 2y_2 - x_2 + \rho^{-1}\left(\lambda^+ - \lambda\right) \end{array} \right\}. \tag{26}$$

For simplicity of presentation and in order to make ALADIN and ADMM more comparable, the matrices H_1 and H_2 are kept constant during this numerical experiment, although further improvements of ALADIN can be obtained by adjusting these matrices online. Notice that the cost of performing one ALADIN iteration (Eq. (26)) is slightly larger than the cost of one ADMM iteration (Eq. (24)), but if all matrix decompositions and related operations are cached, the run-time difference per iteration is small.

Figure 2 compares the performance of ADMM and ALADIN for a randomly generated $A \in \mathbb{R}^{10\times 100}$ and $b \in \mathbb{R}^{10}$ using a Gaussian distribution with zero mean and a standard deviation of $\frac{1}{10}$ for all coefficients. We may set $\kappa = 1$ without loss of generality, as all scales are determined by A and b. Moreover, all primal and dual variables have been initialized with 0 for both ALADIN and ADMM recalling that the optimal solution depends on the randomly generated coefficients A and b. The black solid line in Figure 2 shows the distance of the ADMM iterates from the optimal solution, while the red dashed line shows the corresponding distance,

$$\left\| \begin{pmatrix} x - x^* \\ \lambda - \lambda^* \end{pmatrix} \right\|_\infty$$

for ALADIN as a function of the number of iterations. In this example, ALADIN converges approximately twice as fast as ADMM. This trend is also confirmed when running a large number of randomly generated lasso problems (more than 1000) finding that ADMM takes 310 ± 27 iterations[5] to achieve convergence while ALADIN converges after 147 ± 13 iterations. Here, the termination tolerance was set to $\epsilon = 10^{-8}$ and $\rho = 1$. Among the 1000 randomly generated examples, there was none where ADMM was converging faster than ALADIN. Notice that if we generate ill-conditioned examples on purpose or if ALADIN is run with additional matrix updates, ALADIN converges typically much faster than ADMM (more than a factor 2). However, such examples are not shown here, as the authors attempted to keep the comparison between ALADIN and ADMM as fair as possible for this case study. The results in Figure 2 are solely based on (24) and (26) without tuning anything or making other adjustments.

[4] The matrix $H_1 = A^\top A$ is in general not invertible and therefore the update for y_1 in (26) uses the pseudo-inverse of H_1 and assumes additionally that λ is initialized with 0. It can be checked easily that this minor modification of Algorithm 1 for non-invertible H_1 has no further implications on the convergence properties of the algorithm for this particular example. Alternatively, one could also add a Levenberg-Marquardt regularization [34] to H_1, which leads to similar numerical results.

[5] In order to be more precise, this notation means that ADMM took on average ≈ 310 iterations and the standard deviation was ≈ 27 iterations.

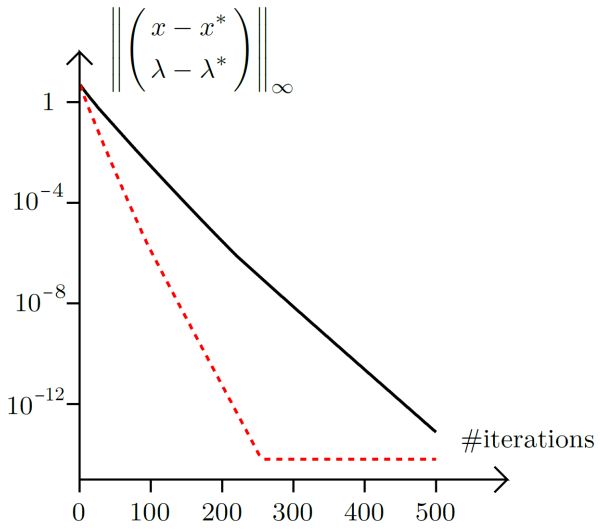

Fig. 2 Distance of the ADMM iterates (black solid line) and of the ALADIN iterates (red dashed line) to the optimal solution for a randomly generated lasso problem. Notice that for accuracies less than 10^{-14} numerical side effects are dominating, which depend on the accuracy settings for the linear algebra routines, which have, however, not been further analyzed in this paper.

5.2 Quadratically constrained quadratic programming

In order to illustrate that Hessian matrix updates can help significantly to improve the convergence properties of ALADIN, the following convex quadratically constrained quadratic programming (QCQP) problem is introduced:

$$\min_z \ \frac{1}{2}\|z-\bar{z}\|_2^2 \quad \text{s.t.} \quad \begin{cases} (z-c_1)^\top (z-c_1) \leq r_1^2 \ \mid \ \mu_1 \\ (z-c_2)^\top (z-c_2) \leq r_2^2 \ \mid \ \mu_2 \end{cases} . \tag{27}$$

Here, the given points $c_1 = (-1,0)^\mathsf{T}$ and $c_2 = (2,0)^\mathsf{T}$ can be interpreted as the centers of two discs with given radius, in this example, $r_1 = r_2 = 2$. The minimizer of the above optimization problem corresponds the point closest to $\bar{z} = (1,1)^\mathsf{T}$, which is contained in the intersection area of the two discs. Next, in order to apply ALADIN, we write (27) in the form

$$\min_{x_1,x_2} \ f_1(x_1) + f_2(x_2) \quad \text{s.t.} \quad A_1 x_1 + A_2 x_2 = b \ \mid \ \lambda$$

with $A_1 = I$, $A_2 = -I$, $b = 0$ and

$$f_1(x_1) = \begin{cases} \frac{1}{4}\|x_1-\bar{z}\|_2^2 & \text{if } (x_1-c_1)^\top (x_1-c_1) \leq r_1^2 \\ \infty & \text{otherwise} \end{cases}$$

$$f_2(x_2) = \begin{cases} \frac{1}{4}\|x_2-\bar{z}\|_2^2 & \text{if } (x_2-c_2)^\top (x_2-c_2) \leq r_2^2 \\ \infty & \text{otherwise} \end{cases} . \tag{28}$$

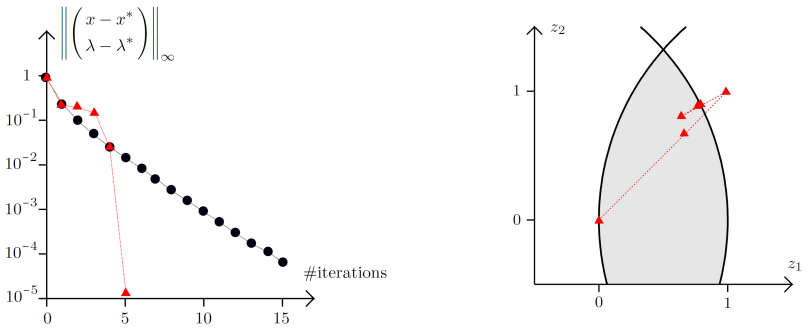

Fig. 3 LEFT: distance of the ALADIN iterates to the optimal solution versus the number of iterations for constant scaling matrices (black dots) and exact Hessian updates (red triangles). RIGHT: visualization of the ALADIN iterates (red triangles) on the (z_1, z_2)-plane. The gray shaded area corresponds to feasible set of the optimization problem.

The left part of Figure 3 shows the distance of the ALADIN iterates from the optimal solution as a function of the number of iterations for both fixed Hessian matrix (black dots) and for exact Hessians (red triangles) based on the update strategy that has been introduced in Sections 4.2 and 4.3. Notice that the exact Hessian information leads to a quadratically convergent algorithm and in this case leads to faster convergence compared to the ALADIN variant with constant Hessian matrix. The right part of Figure 3 visualizes the ALADIN iterates (with exact Hessian updates) on the (z_1, z_2)-plane. Here, the feasible set (intersection of the two discs) is shaded in gray.

6 Conclusions

This paper has introduced a derivative free variant of ALADIN, summarized in Algorithm 1, which is suitable for solving distributed optimization problems with separable convex objective functions and coupled affine constraints. The main theoretical contributions have been summarized in Corollary 1 and Theorem 2, which state that ALADIN convergences for a large class of convex optimization problems from any starting point without introducing line-search or other expensive search routines. Moreover, one of the main advantages of ALADIN compared to ADMM is its similarity to SQP methods. In practice, this means that ALADIN can be used without pre-conditioners as advanced matrix update strategies can be applied for scaling the iterations online. We have also discussed how to use ideas that have originally been developed in the context of SQP methods in order to obtain ALADIN variants with locally superlinear or even quadratic convergence rates.

Acknowledgements The work of Boris Houska and Yuning Jiang was supported by National Natural Science Foundation China (NSFC), Nr. 61473185, as well as ShanghaiTech University, Grant-Nr. F-0203-14-012. The work of Moritz Diehl and Dimitris Kouzoupis was supported by the EU via ERC-HIGHWIND (259 166), ITN-TEMPO (607 957) and ITN-AWESCO (642 682), by DFG via the Research Unit FOR 2401 and by the Ministerium für Wissenschaft, Forschung und Kunst Baden-Wuerttemberg (Az: 22-7533.-30-20/9/3).

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. A. Björck Numerical methods for least squares problems. SIAM, Philadelphia, 1996.
4. P.T. Boggs, J.W. Tolle. Sequential quadratic programming. Acta Numerica, 4, pp:1–51, 1996.
5. S. Boyd and L. Vandenberghe. *Convex Optimization*. University Press, Cambridge, 2004.
6. S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine Learning, 3:1–122, 2011.
7. G. Chen and M. Teboulle. A proximal-based decomposition method for convex minimization problems. Mathematical Programming, 64:81–101, 1994.
8. A.R. Conn, N.I.M. Gould, and P.L. Toint. LANCELOT: a FORTRAN package for large-scale nonlinear optimization (Release A), no. 17 in Springer Series in Computational Mathematics, Springer-Verlag, New York, 1992.
9. M. Diehl, A. Walther, H. G. Bock, and E. Kostina. An adjoint-based SQP algorithm with quasi-Newton Jacobian updates for inequality constrained optimization. Optimization Methods and Software, 25:531–552, 2010.
10. 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.
11. 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.
12. H. Everett. Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. Operations Research, 11(3):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. Proceedings of the 4th IFAC Nonlinear Model Predictive Control Conference, 2012.
14. R. Fletcher. Second order corrections for non-differentiable optimization. D. Griffiths, ed., Springer Verlag, in Numerical Analysis, pages 85–114, 1982.
15. J.V. Frasch, S. Sager, M. Diehl. A Parallel Quadratic Programming Method for Dynamic Optimization Problems. Mathematical Programming Computation, DOI: 10.1007/s12532-015-0081-7, 2015.
16. J.V. Frasch. qpDUNES Website. `http://mathopt.de/qpDUNES`.
17. D. Gabay, 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.
18. R. Glowinski, 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.
19. T. Goldstein, B. O´Donoghue, S. Setzer. Fast Alternating Direction Optimization Methods. Tech. Report, Department of Mathematics, University of California, Los Angeles, USA, 2012.
20. N.I.M. Gould, D. Orban, and P. Toint. GALAHAD, a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization. ACM Trans. Math. Software 29(4):353-372, 2004.
21. A. Griewank. *Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation*. Number 19 in Frontiers in Appl. Math. SIAM, Philadelphia, 2000.
22. W.W. Hager. Updating the inverse of a matrix. SIAM Review. 31(2):221–239, 1989.
23. S. P. Han. A Globally Convergent Method for Nonlinear Programming. *JOTA*, 22:297–310, 1977.
24. A. Hamdi, S.K. Mishra. Decomposition Methods based on Augmented Lagrangian: a Survey. In Topics in Nonconvex Optimization. Mishra, S.K., Chapter 11, 175–204, 2011.
25. M.R. Hestenes. Multiplier and gradient methods. Journal of Optimization Theory and Applications, 4:302–320, 1969.
26. M. Hong, Z.Q. Luo. On the linear convergence of the alternating direction method of multipliers. Mathematical Programming, pages 1–35, 2016. (online first)

27. B. Houska and M. Diehl. A quadratically convergent inexact SQP method for optimal control of differential algebraic equations. Optimal Control Applications & Methods, John Wiley & Sons, 34, pp:396–414, 2012.
28. B. Houska, J. Frasch, M. Diehl. An Augmented Lagrangian Based Algorithm for Distributed Non-Convex Optimization. SIAM Journal on Optimization, Volume 26(2), pp. 1101–1127, 2016.
29. D. Kouzoupis, R. Quirynen, B. Houska, M. Diehl. A block based ALADIN scheme for highly parallelizable direct optimal control. In Proceedings of the 2016 American Control Conference, Boston, USA, pp. 1124–1129, 2016.
30. A. Kozma, J. Frasch, 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, pp:1526–1531, 2013.
31. D.C. Liu, J. Nocedal. On the Limited Memory Method for Large Scale Optimization. Mathematical Programming, 45(3):503–528, 1989.
32. N. Maratos. *Exact penaly function algorithms for finite-dimensional and control optimization problems*. PhD thesis, Imperial College, London, 1978.
33. I. Necoara, D. Doan, 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, pp:2900-2905, 2008.
34. J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2 edition, 2006.
35. M.J.D. Powell. A method for nonlinear constraints in minimization problems. In Optimization, (R. Fletcher, ed.), Academic Press, 1969.
36. M.J.D. Powell. A fast algorithm for nonlinearly constrained optimization calculations. In Numerical Analysis Dundee 1977, G.A. Watson, ed., pp:144–157, Springer Verlag, Berlin, 1977.
37. M.J.D. Powell. The convergence of variable metric methods for nonlinearly constrained optimization calculations. In Nonlinear Programming 3, Academic Press, pp:27–63, New York and London, 1978.
38. M.J.D. Powell. Convergence properties of algorithms for nonlinear optimization. *SIAM Review*, 28:487–500, 1984.
39. R. Quirynen, B. Houska, M. Vallerio, D. Telen, F. Logist, J. Van Impe, M. Diehl. Symmetric algorithmic differentiation based exact Hessian SQP method and software for economic MPC. In proceedings of the IEEE Conference on Decision and Control (CDC), pp. 2752–2757, 2014.
40. A. Rantzer. Dynamic dual decomposition for distributed control. In Proceedings of the 2009 American Control Conference, pp:884–888, 2009.
41. S. Richter, M. Morari, 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.
42. R.T. Rockafellar. Convex Analysis. Princeton University Press, 1970.
43. L. Stella, A. Themelis, P. Patrinos. Forward-backward quasi-Newton methods for nonsmooth optimization problems arXiv:1604.08096, 2016.
44. R.A. Tapia. Quasi-Newton methods for equality constrained optimization: Equivalence of existing methods and a new implementation. In Nonlinear Programming 3, O. Mangasarian, R. Meyer, S. Robinson, eds, Academic Press, pp:125–164, New York, NY, 1978.
45. P. Toint. On sparse and symmetric matrix updating subject to a linear equation. *Mathemathics of Computation*, 31(140):954–961, 1977.
46. R.B. Wilson. A simplicial algorithm for concave programming. Ph.D. thesis, Graduate School of Business Administration, Harvard University, 1963.