An Alternating Trust-Region Newton Algorithm for Distributed Bound-Constrained Nonlinear Programs
Application to the Optimal AC Power Flow

Jean-Hubert Hours and Colin N. Jones

Received: date / Accepted: date

Abstract A novel trust-region Newton method for solving bound-constrained nonlinear programs (NLP) is presented. The proposed technique is amenable to a distributed implementation, as its salient ingredient is an alternating proximal gradient sweep in place of the Cauchy point computation. It is proven that the algorithm yields a sequence that globally converges to a critical point. As a result of some changes to the standard trust-region method, namely a proximal regularisation of the trust-region subproblem, it is shown that the local convergence rate is Q-linear with an arbitrarily small ratio. Thus, convergence is locally almost Q-superlinear, under standard regularity assumptions. The proposed method is successfully applied to compute local solutions to the Optimal Power Flow (OPF) problem on a 9-bus transmission network and on 47-bus and 56-bus distribution networks.

Keywords Non-convex optimisation, Distributed optimisation, Trust-region methods, Coordinate gradient descent, Linear convergence rate

1 Introduction

Minimising a separable smooth non-convex function subject to partially separable coupling equality constraints and separable constraints,

\[
\begin{align*}
\text{minimise} & \sum_{i=1}^{N} f_i(x_i) \\
\text{s.t.} & \quad C(x_1, \ldots, x_N) = 0 \\
& \quad g_i(x_i) = 0 \\
& \quad x_i \in X_i \\
& \quad i \in \{1, \ldots, N\} ,
\end{align*}
\]

Jean-Hubert Hours · Colin N. Jones
Automatic Control Laboratory, Ecole Polytechnique Fédérale de Lausanne
E-mail: jean-hubert.hours@epfl.ch, colin.jones@epfl.ch
appears in many engineering problems such as Distributed Nonlinear Model Predictive Control (DN-MPC) [23], power systems [20] and wireless networking [7]. For such problems involving a large number of agents, which result in large-scale non-convex NLPs, it may be desirable to perform computations in a distributed manner, meaning that all operations are not carried out on one single node, but on multiple nodes spread over a network and that information is exchanged during the optimisation process. Such a strategy may prove useful to reduce the computational burden in the case of extremely large-scale problems. Moreover, autonomy of the agents may be hampered by a purely centralised algorithm. Case in points are cooperative tracking using DNMPC [18] or the Optimal Power Flow problem (OPF) over a distribution network [16], into which generating entities may be plugged or unplugged. Moreover, it has been shown in a number of studies that distributing and parallelising computations can lead to significant speed-up in solving large-scale non-convex NLPs [33]. Splitting operations can be done on distributed memory parallel environments such as clusters [33], or on parallel computing architectures such as Graphical Processing Units (GPU) [14].

Our objective is to develop nonlinear programming methods in which most of the computations can be distributed or parallelised. Some of the key features of a distributed optimisation strategy are the following:

(i) **Shared memory.** Vectors and matrices involved in the optimisation process are stored on different nodes. This requirement rules out direct linear algebra methods, which require the assembly of matrices on a central unit.

(ii) **Concurrency.** A high level of parallelism is obtained at every iteration.

(iii) **Cheap exchange.** Global communications of agents with a central node are cheap (scalars). More costly communications (vectors) remain local between neighbouring agents. In general, the amount of communication should be kept as low as possible. It is already clear that globalisation strategies based on line-search do not fit with the distributed framework [14], as these entail evaluating a ‘central’ merit function multiple times per iteration, thus significantly increasing communications.

(iv) **Inexactness.** Convergence is ‘robust’ to inexact solutions of the subproblems, since it may be necessary to truncate the number of sub-iterations due to communication costs.

(v) **Fast convergence.** The sequence of iterates converges at a fast (at least linear) local rate. Slow convergence generally results in a prohibitively high number of communications.

As we are interested in applications such as DNMPC, which require solving distributed parametric NLPs with a low latency [18], a desirable feature of our algorithm should also be

(vi) **Warm-start and activity detection.** The algorithm detects the optimal active-set quickly and enables warm-starting.

Whereas a fair number of well-established algorithms exist for solving distributed convex NLPs [2], there is, as yet, no consensus around a set of practical methods applicable to distributed non-convex programs. Some work [33] exists on the parallelisation of linear algebra operations involved in solving non-convex NLPs with IPOPT [30], but the approach is limited to very specific problem structures and the globalisation phase of IPOPT (filter line-search) is not suitable for fully distributed implementations (requirements (iii), (iv) and (vi) are not met). Among existing strategies capable of addressing a broader class of distributed non-convex programs, one can make a clear distinction between Sequential Convex Programming (SCP) approaches and augmented Lagrangian techniques.

An SCP method consists in iteratively solving distributed convex NLPs, which are local approximations of the original non-convex NLP. To date, some of the most efficient algorithms for solving distributed convex NLPs combine Lagrangian decomposition with smoothing techniques [23,
On the contrary, an augmented Lagrangian method aims at decomposing a non-convex auxiliary problem inside an augmented Lagrangian loop [8, 17, 19]. While convergence guarantees can be derived in both frameworks, computational drawbacks also exist on both sides. For instance, it is not clear how to preserve the convergence properties of SCP schemes when every sub-problem is solved to a low level of accuracy. Hence, (iv) is not satisfied immediately. Nevertheless, for some recent work in this direction, one may refer to [27]. The convergence rate of the algorithm of [27] is at best linear, thus not fulfilling (v). On the contrary, the inexactness issue can be easily handled inside an augmented Lagrangian algorithm, as global and fast local convergence is guaranteed even though the sub-problems are not solved to a high level of accuracy [15, 9]. However, in practice, poor initial estimates of the dual variables can drive the process to infeasible points. Moreover, it is still not clear how the primal non-convex problems should be decomposed and solved efficiently in a distributed context. The quadratic penalty term of an augmented Lagrangian does not allow for the same level of parallelism as a (convex) dual decomposition. Thus, requirement (ii) is not completely satisfied. To address this issue, we have recently proposed applying Proximal Alternating Linearised Minimisations (PALM) [3] to solve the auxiliary augmented Lagrangian problems [18, 19]. The resulting algorithm inherits the slow convergence properties of proximal gradient methods and is difficult to precondition in practice. In this paper, a novel mechanism for handling the augmented Lagrangian sub-problems in a more efficient manner is proposed and analysed. The central idea is to use PALM to compute a Cauchy point in a trust-region Newton method [11].

When looking at practical trust-region methods for solving bound-constrained problems [32], one may notice that the safeguarded Conjugate Gradient (sCG) is well-suited to distributed implementations, as the main computational tasks are sparse matrix-vector and vector-vector multiplications, which do not require the assembly of a matrix on a central node. Moreover, the central communications involved in an sCG algorithm are cheap. Thus, sCG satisfies (i), (ii) and (iii). The implementation of CG on distributed architectures has been extensively explored [29, 13, 14]. Furthermore, a trust-region update requires only one ‘central’ objective evaluation per iteration. From a computational perspective, it is thus comparable to a dual update, which requires evaluating the constraints functional and is ubiquitous in distributed optimisation algorithms. However, computing the Cauchy point in a trust-region loop is generally done by means of a projected line-search [32] or sequential search [10]. Whereas it is broadly admitted that the Cauchy point computation is cheap, this operation requires a significant amount of global communications in distributed memory parallel environments, and is thus hardly amenable to such applications [14]. This hampers the implementability of trust-region methods with good convergence guarantees on distributed computing platforms, whereas many parts of the algorithm are attractive for such implementations. The aim of this paper is to bridge the gap by proposing a novel way of computing the Cauchy point that is more tailored to the distributed framework. Coordinate gradient descent methods such as PALM, are known to be parallelisable for some partial separability structures [2]. Moreover, in practice, the number of backtracking iterations necessary to select a block step-size, can be easily bounded, making the approach suitable for ‘Same Instruction Multiple Data’ architectures. Therefore, we propose using one iteration of PALM to compute a Cauchy point. As shown next, PALM turns out to be efficient at identifying the optimal active-set. It can then be accelerated by means of an inexact Newton method. As our algorithm differs from the usual trust-region Newton method, we provide a detailed convergence analysis. Finally, one should mention a recent paper [31], in which a trust-region method is combined with alternating minimisations, namely the Alternating Directions Method of Multipliers (ADMM) [2], but in a very different way from the strategy described next. The contributions of the paper are the following:
We propose a novel way of computing a Cauchy point in a trust-region framework, which is suitable for distributed implementations.

We adapt the standard trust-region algorithm to the proposed Cauchy point computation. Global convergence along with an almost Q-superlinear local rate is proven under standard assumptions.

The proposed trust-region method is used as a primal solver in an augmented Lagrangian dual loop, resulting in an algorithm that meets requirements (i)-(vi), and is applied to solve OPF programs in a distributed fashion.

In Section 2, some basic notion in variational analysis is recalled. In Section 3, the trust-region algorithm with distributed Cauchy point computation is presented. Its convergence properties are analysed in Section 4. Global convergence to a critical point is guaranteed, as well as almost Q-superlinear local convergence. Finally, the proposed algorithm is tested on OPF problems over transmission and distribution networks in Section 5.

2 Background

Given a closed convex set \( \Omega \), the projection operator onto \( \Omega \) is denoted by \( P_\Omega \) and the indicator function of \( \Omega \) is defined by

\[
\iota_\Omega(x) = \begin{cases} 0 & \text{if } x \in \Omega \\ +\infty & \text{if } x \notin \Omega \end{cases} .
\]

We define the normal cone to \( \Omega \) at \( x \in \Omega \) as

\[
N_\Omega(x) := \{ v \in \mathbb{R}^d \mid \forall y \in \Omega, v^\top (y - x) \leq 0 \} .
\]

The tangent cone to \( \Omega \) at \( x \) is defined as the closure of feasible directions at \( x \) [25]. Both \( N_\Omega(x) \) and \( T_\Omega(x) \) are closed convex cones. As \( \Omega \) is convex, for all \( x \in \Omega \), \( N_\Omega(x) \) and \( T_\Omega(x) \) are polar to each other [25].

**Theorem 1** (Moreau’s decomposition [22]). Let \( K \) be a closed convex cone in \( \mathbb{R}^d \) and \( K^o \) its polar cone. For all \( x, y, z \in \mathbb{R}^d \), the following two statements are equivalent:

1. \( z = x + y \) with \( x \in K \), \( y \in K^o \) and \( x^\top y = 0 \).
2. \( x = P_K(z) \) and \( y = P_{K^o}(z) \).

The box-shaped set \( \{ x \in \mathbb{R}^d \mid \forall i \in \{1, \ldots, d\}, l_i \leq x_i \leq u_i \} \) is denoted by \( B(l, u) \). For \( x \in \mathbb{R}^d \) and \( r > 0 \), the open ball of radius \( r \) centered around \( x \) is denoted by \( B(x, r) \). Given \( x \in \Omega \), the set of active constraints at \( x \) is denoted by \( A_\Omega(x) \).

**Lemma 1** ([11]). The function \( \phi \) defined by

\[
\phi(t) = \| P_\Omega(x + tv) - x \|_2 , \quad t > 0 ,
\]

is non-decreasing for all \( x \in \mathbb{R}^d \) and \( v \in \mathbb{R}^d \).

Given a set \( S \subseteq \mathbb{R}^d \), its relative interior is defined as the interior of \( S \) within its affine hull, and is denoted by \( \text{ri}(S) \).

A critical point \( x^* \) of the function \( f + \iota_\Omega \) with \( f \) differentiable, is said to be non-degenerate if

\[
-\nabla f(x^*) \in \text{ri}(N_\Omega(x^*)) .
\]
Given a Lipschitz-continuous function \( f \), its Lipschitz constant is denoted by \( \lambda(f) \). Given a differentiable function \( f \) of several variables \( x_1, \ldots, x_N \), its gradient with respect to variable \( x_k \) is denoted by \( \nabla_k f \). Given a matrix \( M \in \mathbb{R}^{m \times n} \), its \((i, j)\) element is denoted by \( M(i, j) \).

Recall that in a finitedimensional setting, the norms \( \| \cdot \|_2 \) and \( \| \cdot \|_\infty \) are equivalent, more precisely
\[
\| x \|_\infty \leq \| x \|_2 \leq \sqrt{d} \| x \|_\infty ,
\]
for all \( x \in \mathbb{R}^d \).

A sequence \( \{x^l\} \) converges to \( x^* \) at a Q-linear rate \( \rho \in (0, 1) \) if, for \( l \) large enough,
\[
\frac{\|x^{l+1} - x^*\|_2}{\|x^l - x^*\|_2} \leq \rho .
\]
The convergence rate is said to be Q-superlinear if the above ratio tends to zero as \( l \) goes to infinity.

### 3 A trust-region algorithm with distributed activity identification

#### 3.1 Algorithm formulation

The problem we consider is that of minimising a partially separable objective function subject to separable convex constraints.

\[
\begin{align*}
\text{minimise} & \quad L(\hat{x}_1, \ldots, \hat{x}_N) \\
\text{s.t.} & \quad \hat{x}_i \in \hat{\Omega}_i, \quad \forall i \in \{1, \ldots, N\} ,
\end{align*}
\]

where \( \hat{x} := (\hat{x}_1^\top, \ldots, \hat{x}_N^\top)^\top \in \mathbb{R}^n \), with \( n = \sum_{i=1}^N n_i \), and \( \hat{\Omega} := \hat{\Omega}_1 \times \ldots \times \hat{\Omega}_N \), where the sets \( \hat{\Omega}_i \subset \mathbb{R}^{n_i} \) are closed and convex. The following assumption is standard in distributed computations [2].

**Assumption 1 (Colouring scheme).** The sub-variables \( \hat{x}_1, \ldots, \hat{x}_N \) can be re-ordered and grouped together in such a way that a Gauss-Seidel minimisation sweep on the function \( L \) can be performed in \( K \ll N \) parallel steps. In the sequel, the re-ordered variable is denoted by \( x = (x_1^\top, \ldots, x_K^\top)^\top \). The set \( \hat{\Omega} \) is transformed accordingly into \( \Omega = \Omega_1 \times \ldots \Omega_K \). It is worth noting that each set \( \Omega_k \) with \( k \in \{1, \ldots, K\} \) can then be decomposed further into sets \( \Omega_i \) with \( i \in \{1, \ldots, N\} \).

As a consequence of Assumption 1, NLP (1) is equivalent to

\[
\begin{align*}
\text{minimise} & \quad L(x_1, \ldots, x_K) \\
\text{s.t.} & \quad x_k \in \Omega_k, \quad \forall k \in \{1, \ldots, K\} .
\end{align*}
\]

**Remark 1.** Such a partially separable structure in the objective (Assumption 1) is encountered very often in practice, for instance when relaxing equality constraints that stand for network coupling constraints via an augmented Lagrangian penalty. In NLPs resulting from the direct transcription of optimal control problems, the objective is generally separable and the constraints are stage-wise with a coupling between the variables at a given time instant with the variables of the next time instant. In this particular case, \( K = 2 \). In Section 5, we illustrate this property by means of examples arising from various formulations of the Optimal Power Flow (OPF) problem. The number of colours \( K \) represents the level of parallelism that can be achieved in a Gauss-Seidel method for solving (1).
For the sake of exposition, in order to make the distributed setting of our algorithm apparent, we assume that every sub-variable \( \tilde{x}_i \), with \( i \in \{1, \ldots, N\} \), is associated with a computing node. Two nodes are called neighbours if they are coupled in the objective \( L \). The iterative method described next aims at computing every iterate in a distributed fashion, which requires communications between neighbouring nodes and leads to a significant level of concurrency.

**Assumption 2.** The set \( \{ x \in \Omega \mid L(x) \leq L(x^0) \} \) is compact with a non-empty interior. The objective function \( L \) is bounded below on this set.

The algorithm formulation can be done for any convex set \( \Omega \), but some features are more suitable for the bound-constrained case. Bound constraints are ubiquitous in nonlinear programming, as slack variables can be introduced to turn general inequality constraints into equality constraints subject to bounds on the variables.

**Assumption 3 (Bound constraints).** The sets \( \{ \Omega_i \}_{i=1}^N \) are bound constraints, that is for all \( i \in \{1, \ldots, N\} \), there exists \( w_i, \pi_i \in \mathbb{R}^n \) such that \( \Omega_i = B(w_i, \pi_i) \), and \( \Omega_i \) is non-empty.

**Assumption 4.** The objective function \( L \) is twice continuously differentiable in an open set containing \( \Omega \). The gradient \( \nabla L \) is Lipschitz continuous on \( \Omega \).

It is well-known [11] that for problem (1), \( x^* \) being a critical point is equivalent to

\[
P_{\Omega} (x^* - \nabla L(x^*)) = x^*. \tag{2}
\]

Algorithm 1 below is designed to compute a critical point \( x^* \) of the function \( L + \iota_{\Omega} \). Algorithm 1 is built within the standard trust-region framework. At every iteration, a model \( m \) of the objective function \( L \) is constructed around the current iterate \( x \) as follows

\[
m(x') := L(x) + \nabla L(x) ^T (x' - x) + \frac{1}{2} (x' - x) ^ T \nabla^2 L(x) (x' - x), \tag{3}
\]

where \( x' \in \mathbb{R}^n \).

**Assumption 5.** There exists \( \hat{H} > 0 \) such that

\[
\| \nabla^2 L(x) \|_2 \leq \hat{H},
\]

for all \( x \in \Omega \).

**Remark 2.** It is worth noting that the partial separability structure of the objective function \( L \) is transferred to the sparsity pattern of the hessian \( \nabla^2 L \). Hence, a Gauss-Seidel sweep on the model function \( m \) can also be carried out in \( K \) parallel steps.

In the remainder, the gradient \( \nabla L(x) \) is often denoted by \( g(x) \) and the hessian \( \nabla^2 L(x) \) by \( H(x) \). The model function \( m \) is an approximation of the objective function \( L \) around the current iterate \( x \). The quality of the approximation is controlled by the trust-region, defined as the box \( B(x - \Delta, x + \Delta) \), where \( \Delta \) is the trust-region radius. At every iteration, Algorithm 1 performs an activity detection step by computing points \( z_1, \ldots, z_K \) (Lines 4 to 8). This is the main novelty of Algorithm 1, compared to existing trust region techniques, and allows for different step-sizes \( s_1, \ldots, s_K \), which is relevant in a distributed framework, as the current active-set can be split among nodes and does not need to be stored centrally. In the trust-region literature, the point \( z \) is
An Alternating Trust-Region Newton Algorithm for Distributed Bound-Constrained NLPs

Algorithm 1 Trust-region method with distributed activity detection

1: Parameters: Initial trust-region radius $\Delta$, trust-region update parameters $c_1$, $c_2$ and $c_3$ such that $0 < c_1 < c_2 < 1 < c_3$, test ratios $r_1$ and $r_2$ such that $0 < r_1 < r_2 < 1$, coefficients $\gamma_1 \in (0, 1)$ and $\gamma_2 > 0$, termination tolerance $\epsilon$, regularisation parameters $\eta_1, \ldots, \eta_K$.

2: Input: Initial guess $x$, projection operators $\{P_{D_i}\}_{i=1}^N$, objective function $L$, gradient $\nabla L$ and hessian $\nabla^2 L$.

3: while $\|P_{D_i}(x - \nabla L(x)) - z\|_2 > \epsilon$ do

4: Active-set identification (PALM):

5: for $k = 1, \ldots, K$ do

6: $z_k \leftarrow P_{D_k} \left( x_k - s_k \nabla_k m \left( \bar{z}_{1,k-1}, x_{k,K} \right) \right)$.

7: where $s_k$ is computed according to requirements (4), (5) and (6).

8: end for

9: Refinement (sCG):

10: Find $y \in H$ such that

11: $m(x) - m(y) \geq \gamma_1 (m(x) - m(z))$

12: $\|y - x\|_{\infty} \leq \gamma_2 \Delta$

13: $A_{D_k}(z_k) \subset A_{D_k}(y_k)$ for all $k \in \{1, \ldots, K\}$

14: Trust-region update:

15: $\bar{H} \leftarrow \frac{1}{r_{y,x}}(D(y) - m(x) - m(y))$

16: if $R < r_1$ then $\triangleright$ Not successful

17: (Do not update $x$)

18: Take $\Delta$ within $[c_1 \Delta, c_2 \Delta]$

19: else if $R \in [r_1, r_2]$ then $\triangleright$ Successful

20: $x \leftarrow y$

21: Take $\Delta$ within $[c_1 \Delta, c_3 \Delta]$

22: Compute $\nabla L(x)$ and $\nabla^2 L(x)$

23: else $\triangleright$ Very successful

24: $x \leftarrow y$

25: Take $\Delta$ within $[\Delta, c_4 \Delta]$

26: Compute $\nabla L(x)$ and $\nabla^2 L(x)$

27: end if

28: end while

often referred to as the Cauchy point. We keep this terminology in the remainder of the paper. It is clear from its formulation that Algorithm 1 allows one to compute Cauchy points independently on every node. Once the Cauchy points $z_1, \ldots, z_K$ have been computed, they are used in the refinement step to compute a new point $y$ that satisfies the requirements shown from Lines 9 to 13. The last step consists in checking if the model decrease $m(y) - m(x)$ is close enough to the variation in the objective $L$ (Lines 16 to 27). In this case, the iterate is updated and the trust-region radius $\Delta$ increased, otherwise the radius is shrunk and the iterate frozen. This operation requires a global exchange of information between nodes.

In Algorithm 1, the block-coordinate step-sizes $s_k$ are chosen so that for all $k \in \{1, \ldots, K\}$, the Cauchy point $z_k$ satisfies

$$\begin{align*}
& m \left( z_{1,k-1}, x_{k,K} \right) - m \left( z_{1,k}, x_{k+1,K} \right) \geq \eta_k \| z_k - x_k \|_2^2 \\
& \| z_k - x_k \|_\infty \leq \nu_1 \Delta \\
& z_k \in \Omega_k
\end{align*}$$

(4)

where $z_{1,k-1}$ stands for $(z_1^\top, \ldots, z_{k-1}^\top)^\top$, along with one of the following two conditions

$$s_k > \frac{1}{r \| E_k \bar{H}(x) \|_2} \quad \text{or} \quad s_k \geq \bar{s}_k$$

(5)
where
\[ \| P_{\Omega_k} (x_k - s_k \nabla_k m (z_{[1,k-1]}^T, x_{[k,K]}^T)) - x_k \|_\infty \geq \nu_2 \Delta . \] (6)

The matrix \( E_k \in \mathbb{R}^{n_k \times n} \) is such that for \( i \in \{1,\ldots,n_k\} \),
\[ E_k(i,n_1 + \ldots + n_{k-1} + i) = 1, \] (7)
and all other entries are zero. In Eq. (4) and Eq. (6), \( \{\eta_k\}_{k=1}^K \), \( \nu_1 \) and \( \nu_2 \) are positive constants such that
\[ \nu_1 > \nu_2 \quad \text{and} \quad \| E_k H(x) \|_2 > \eta_k . \]
for all \( k \in \{1,\ldots,K\} \). In Eq. (5), the constant \( r \) is an integer larger than two. We define
\[ \undereta := \min \{\eta_k\}_{k=1}^K \quad \text{and} \quad \overeta := \max \{\eta_k\}_{k=1}^K . \]
Conditions (4) ensure that the step-sizes \( s_k \) are small enough to enforce a sufficient decrease coordinate-wise, as well as containment within a scaled trust-region. Conditions (5) and (6) guarantee that the step-size \( s_k \) does not become arbitrarily small. The rationale behind condition (5) is that if the trust-region radius is too small, the Cauchy point is taken on the boundary of the trust-region. All conditions (4), (5) and (6) can be tested in parallel in each of the \( K \) groups of variables. In the next two paragraphs of this section, the choice of step-sizes \( s_k \) ensuring the sufficient decrease is clarified, as well as the distributed refinement step. In the next section, the convergence properties of Algorithm 1 are analysed. Numerical examples are presented in the last section of the paper.

3.2 Step-sizes computation in the activity detection phase

At a given iteration of Algorithm 1, the step-sizes \( s_k \) may be computed by backtracking to ensure a form of sufficient decrease and coordinate-wise containment in an extended trust-region as formalised by (4). For the purpose of estimating the step-size \( s_k \), a quadratic model \( q_k \) is defined for each sub-block \( k \in \{1,\ldots,K\} \) as follows
\[ q_k(x'',s;x') := m(x') + \nabla_k m(x')^T (x'' - x_k^T) + \frac{1}{2s} \| x'' - x_k^T \|_2^2 , \]
where \( x' \in \mathbb{R}^n, x'' \in \mathbb{R}^{n_k}, s > 0 \) and \( m \) is the model function around \( x \) and defined by (3). For the remainder, it is important to remember that, given \( k \in \{1,\ldots,K\} \) and \( x' \in \mathbb{R}^n \),
\[ \nabla_k m(x') = \nabla_k L(x) + E_k H(x) (x' - x) , \]
where the \( n_k \times n \) matrix \( E_k \) is defined in (7).

**Remark 3.** It is worth noting that the backtracking search described in Algorithm 2 above can be run in parallel among the variables of group \( k \), as they are decoupled from each other. As a result, there is one step-size per sub-variable \( \tilde{x}_i \) in group \( k \). Yet, for simplicity, we write it as a single step-size \( s_k \). The reasoning of Section 4 can be adapted accordingly.

**Lemma 2.** Let \( k \in \{1,\ldots,K\} \). The backtracking procedure of Algorithm 2 terminates with an iterate \( z_k \) satisfying conditions (4).
Algorithm 2 Coordinate-wise backtracking at group $k$

1: **Parameters:** backtracking parameter $c < 1$.
2: **Input:** variable $x' = (z_1^T, \ldots, z_{k-1}^T, x_k^T, \ldots, x_K^T)^T$, initial guess $s$ for the step-size.
3: $x_k \leftarrow x_k$, $s_k \leftarrow s$.
4: while $m(z_{[1,k-1]}, x_k, x_{[k+1,K]}) + \frac{\eta_k}{2} \|z_k - x_k\|_2^2 > q_k(z_k, s_k; x')$ do
5: $s_k \leftarrow c \cdot s_k$
6: $z_k \leftarrow P_{\Omega_k}(x_k - s_k \nabla_k m(x'))$
7: end while
8: **Output:** $z_k$

**Proof.** From the descent Lemma,

$$m(z_{[1,k-1]}, z_k, x_{[k+1,K]}) \leq m(x') + \nabla_k m(x')^T (z_k - x_k) + \frac{\lambda(\nabla_k m)}{2} \|z_k - x_k\|_2^2,$$

where the variable $x'$ is defined in Algorithm 2 above. As $\lambda(\nabla_k m) = \|E_k H(x)\|_2$, by choosing $s_k < \sqrt{1/(\|E_k H(x)\|_2 + \eta_k)}$, which is satisfied after a large enough number of iterations of the while loop in Algorithm 2, the terminating condition of Line 4 is met. However, from the definition of $z_k$ at Line 6,

$$z_k = \arg\min_{z \in \Omega_k} \nabla_k m(x')^T (z - x_k) + \frac{1}{2s_k} \|z - x_k\|_2^2,$$

which implies

$$\nabla_k m(x')^T (z_k - x_k) + \frac{1}{2s_k} \|z_k - x_k\|_2^2 \leq 0,$$

and then the sufficient decrease (4). Thus, it is clear that both conditions of (4) are simultaneously satisfied after enough iterations in the backtracking loop, since the second inequality in (4) can be ensured by further backtracking.

**Remark 4.** From the proof of Lemma 2, one can notice that the backtracking procedure described in 2 can be avoided if a Lipschitz constant is available for the model gradient $\nabla m$. Such a Lipschitz constant is given by the maximum singular value of the model hessian $H$.

Conditions (5) are satisfied if the number of backtracking iterations remains bounded, which is the case since the norm of the hessian $\nabla^2 L$ is bounded above by Assumption 5.

3.3 Distributed computations in the refinement step

In Algorithm 1, the objective gradient $g(x)$ and hessian $H(x)$ are re-computed after every successful iteration. This task requires exchanges of variables between neighbouring nodes, as the objective is partially separable (Ass. 1). Node $i$ only needs to store the sub-part of the objective function $L$ that combines its variable $\tilde{x}_i$ and the variables associated to its neighbours. However, the refinement step (line 9 to 13 in Algorithm 1), in which one obtains a fraction of the model decrease yielded by PALM, should also be computed in a distributed manner. As detailed next, this phase consists in solving the Newton problem on the subspace of the free variables at the current iteration,
which is yielded by the set of free variables at the Cauchy points \( z_1, \ldots, z_K \). In order to achieve a reasonable level of efficiency in the trust-region procedure, this step is generally performed via the Steihaug-Toint CG, or sCG [26]. The sCG algorithm is a standard CG that is cut if a negative curvature direction is encountered or a bound is hit during the search. Compared to direct methods, iterative methods such as the sCG procedure have advantages in a distributed framework, for they do not require building the hessian on a central node. Furthermore, their convergence speed can be enhanced via block-diagonal preconditioning, which does not hamper distributing operations. In the sequel, we briefly show how a significant level of distributed computations can be obtained in the sCG procedure, mainly due to the sparsity structure of the hessian, which matches the partial separability structure of the objective function, and the matrix-vector products. More details on distributed implementations of the CG algorithm can be found in numerous research papers [29, 13, 14]. The sCG algorithm that is described next is a rearrangement of the standard sCG procedure following the idea of [13]. The two separate inner products that usually appear in the CG are grouped together at the same stage of the Algorithm.

An important feature of the refinement step is the increase of the active set at every iteration. More precisely, for reasons that are exposed next, the set of active constraints at the point \( y \) needs to contain the set of active constraints at the Cauchy point \( z \).

**Assumption 6.** For all \( k \in \{1, \ldots, K\} \), \( A_{\Omega_k}(z_k) \subset A_{\Omega_k}(y_k) \).

Assumption 6 is very easy to fulfil in the bound-constrained case, as it just requires enforcing the constraint \( x_{k,i} = z_{k,i}, \ i \in \{ j \in \{1, \ldots, n_k \} \mid z_{k,j} = x_{k,j} \text{ or } x_{k,j} \} \) for all groups \( k \in \{1, \ldots, K\} \) in the trust-region problem.

For the convergence analysis that follows, the refinement step needs to be modified compared to existing trust-region techniques. Instead of solving the standard refinement problem

\[
\begin{align*}
\text{minimise} & \quad g(x)^\top p + \frac{1}{2} p^\top H(x) p \\
\text{s.t.} & \quad \|p\|_\infty \leq \gamma_2 \Delta \\
& \quad x + p \in \Omega \\
& \quad p_i = 0, \ i \in A_{\Omega}(z),
\end{align*}
\]

in which the variables corresponding to indices of active constraints at the Cauchy point \( z \) are fixed to zero, we solve a regularised version

\[
\begin{align*}
\begin{align*}
\text{minimise} & \quad g(x)^\top (y - x) + \frac{1}{2} (y - x)^\top H(x) (y - x) + \sigma \frac{1}{2} \|y - z\|_2^2 \\
\text{s.t.} & \quad \|y - x\|_\infty \leq \gamma_2 \Delta \\
& \quad y_i = z_i, \ i \in A_{\Omega}(z),
\end{align*}
\end{align*}
\]  

where \( \sigma \in (\sigma, \bar{\sigma}) \) with \( \sigma > 0 \), and \( z \) is the Cauchy point yielded by the procedure described in the previous paragraph 3.2. The regularisation coefficient \( \sigma \) should not be chosen arbitrarily, as this may hamper the fast local convergence properties of the Newton method. This point is made clearer in
paragraph 4.3 of Section 4. The regularised trust-region subproblem (8) can be equivalently written
\[
\begin{align*}
\text{min} & \quad g_\sigma(x)^T p + \frac{1}{2} p^T H_\sigma(x) p \\
\text{s.t.} & \quad x + p \in \Omega \\
& \quad \|p\|_\infty \leq \gamma_2 \Delta \\
& \quad p_i = 0, \ i \in A_\Omega(z),
\end{align*}
\]
with
\[
g_\sigma(x) := g(x) - \sigma(z - x), \quad H_\sigma(x) := H(x) + \sigma^2 I.
\]

We denote by \( Z \) the \( n \)-by-(\( n - |A_\Omega(z)| \)) selection matrix of the free variables, which are the variables that are not fixed to a bound \( x_i \) or \( \pi_i \) at the Cauchy point \( z \). The columns of \( Z \) form an orthonormal basis of the subspace defined by the active constraints at the Cauchy point \( z \).

**Algorithm 3** Distributed safeguarded CG

1: Input: reduced hessian \( \hat{H}_\sigma := Z^T H_\sigma Z \), reduced gradient \( \hat{g} := Z^T g \), initial guess \( \hat{z} := Z^T z \)
2: Parameters: stopping tolerance \( \hat{\epsilon} := \xi \|\hat{g}\|_2 \), with \( \xi \in (0, 1) \)
3: \((\hat{x}, \hat{p}, \hat{v}, \hat{t}, \hat{u}_{\text{prev}}) \leftarrow \text{InitCG} (z, x, Z, H_\sigma, \hat{H}_\sigma, \hat{g})\)
4: loop
5: \( \hat{s} \leftarrow \hat{H}_\sigma \hat{p} \)
6: for \( k = 1 \ldots K \) do
7: \( \hat{r}_k \leftarrow \hat{s}_k \) \( \triangleright \) Local communications
8: \( \hat{u} \leftarrow \sum_{k=1}^{K} \hat{r}_k \hat{r}_k \)
9: \( \hat{s} \leftarrow \sum_{k=1}^{K} \hat{r}_k \hat{s}_k \)
10: \( \hat{\beta} \leftarrow \hat{u} / \hat{u}_{\text{prev}}, \hat{t} \leftarrow \hat{\beta} - \hat{t}_k^2 \)
11: for \( k = 1 \ldots K \) do
12: \( \hat{p}_k \leftarrow \hat{r}_k + \hat{\beta} \hat{p}_k \)
13: \( \hat{v}_k \leftarrow \hat{s}_k + \hat{\beta} \hat{v}_k \)
14: Compute smallest \( \alpha_k \) such that \( \hat{x}_k + \alpha_k \hat{p}_k \) hits a bound \( \hat{\pi}_k \) or the trust-region radius \( \Delta \)
15: end for
16: \( \hat{\alpha} \leftarrow \text{StepSizeCG} (\hat{t}, \hat{u}, \alpha_1, \ldots, \alpha_K) \) \( \triangleright \) Centrally
17: for \( k = 1 \ldots K \) do
18: \( \hat{x}_k \leftarrow \hat{x}_k + \hat{\alpha} \hat{p}_k \)
19: \( \hat{r}_k \leftarrow \hat{r}_k - \hat{\alpha} \hat{v}_k \)
20: end for
21: if \( \sqrt{\hat{u}} < \hat{\epsilon} \) or \( \hat{t} \leq 0 \) then
22: break
23: else
24: \( \hat{u}_{\text{prev}} \leftarrow \hat{u} \)
25: end if
26: end loop
27: Output: \( y \leftarrow z + Z(\hat{x} - \hat{z}) \)

**Remark 5.** It is worth noting that the requirement \( m(x) - m(y) \geq \gamma_1 (m(x) - m(z)) \), with \( \gamma_1 < 1 \), is satisfied after any iteration of Algorithm 3, as the initial guess is the Cauchy point \( z \) and the sCG iterations ensure monotonic decrease of the regularised model (Theorem 2.1 in [26]).
Remark 6. It is worth noting that the sparsity pattern of the reduced hessian $\hat{H}_\sigma$ has the same structure as the sparsity pattern of the hessian $H$, as the selection matrix $Z$ has a block-diagonal structure. Moreover, the partial separability structure of the objective matches the sparsity patterns of both the hessian and the reduced hessian. For notational convenience, Algorithms 3, 4 and 5 are written in terms of variables $x_1, \ldots, x_K$, but they are actually implementable in terms of variables $\hat{x}_1, \ldots, \hat{x}_N$ with the same level of parallelism.

Algorithm 4 StepSizeCG: Step-size selection in sCG Algorithm 3

1: Input: Inner products $\hat{t}$ and $\hat{u}$, step-sizes $\{\alpha_1, \ldots, \alpha_K\}$
2: if $\hat{t} \leq 0$ then
3: $\hat{\alpha} \leftarrow \min \{\alpha_1, \ldots, \alpha_K\}$
4: else
5: $\hat{\alpha} \leftarrow \hat{u}/\hat{t}$
6: end if
7: Output: Step-size $\hat{\alpha}$

Algorithm 5 InitCG: Initialisation of sCG Algorithm 3

1: Input: selection matrix $Z$, full hessian $H$, reduced hessian $\hat{H}_\sigma$, reduced gradient $\hat{g}$, Cauchy point $z$, iterate $x$
2: Parameters: trust-region radius $\Delta$, regularisation coefficient $\sigma$, bounds $\hat{x}$
3: $\hat{r} \leftarrow \hat{g} + Z^T (H z - x) - \sigma z$, $\hat{p} \leftarrow \hat{r}$
4: for $k = 1 \ldots K$ do $\triangleright$ Neighbour-to-neighbour communications
5: Compute smallest step-size $\alpha_k$ such that $\hat{x}_k + \alpha_k \hat{p}_k$ hits a bound $\hat{x}_k$, $\hat{r}_k$ or the trust-region radius $\Delta$
6: end for
7: $\hat{t} \leftarrow \sum_{k=1}^K \hat{r}_k^T \hat{r}_k$ $\triangleright$ Global summation
8: if $\hat{t} \leq 0$ then
9: $\hat{\alpha} \leftarrow \min \{\alpha_1, \ldots, \alpha_K\}$
10: for $k = 1 \ldots K$ do $\triangleright$ In parallel among $N$ nodes
11: $\hat{x}_k \leftarrow \hat{x}_k + \hat{\alpha} \hat{p}_k$
12: end for
13: break in Algorithm 3
14: end if
15: for $k = 1 \ldots K$ do $\triangleright$ In parallel among $N$ nodes
16: $\hat{u} \leftarrow \sum_{k=1}^K \hat{r}_k \hat{r}_k^T$, $\hat{\alpha} \leftarrow \hat{u}/\hat{t}$
17: $\hat{x}_k \leftarrow \hat{x}_k + \hat{\alpha} \hat{p}_k$
18: end for
19: $\hat{r}_k \leftarrow \hat{r}_k - \hat{\alpha} \hat{v}_k$
20: end for
21: Output: $\hat{x}$, $\hat{p}$, $\hat{v}$, $\hat{r}$, $\hat{t}$, $\hat{u}$

In Algorithm 3, the reduced hessian $\hat{H}$ can be evaluated when computing the product at line 5, which requires local exchanges of variables between neighbouring nodes. From a distributed implementation perspective, the costly parts of Algorithm 3 are at line 9 and line 16. These operations consist in summing up the inner products from all nodes and a minimum search over the step-sizes that ensure constraint satisfaction and containment in the trust-region. They need to be performed on a central node that has access to all data from other nodes, or via a consensus algorithm. There-
fore lines 9 and 16 come with a communication cost, although the amount of transmitted data is very small (one number per node).

In the end, one should notice that the information that is required to be known globally by all nodes \( \{1, \ldots, N\} \) is fairly limited at every iteration of Algorithm 1. It only consists of the trust-region radius \( \Delta \) and the step-sizes \( \hat{\alpha} \) and \( \hat{\beta} \) in the refinement step 3. Finally, at every iteration, all nodes need to be informed of the success or failure of the iteration so as to update or freeze their local variables. This is the result of the trust-region test, which needs to be carried out on a central node.

4 Convergence analysis

The analysis of Algorithm 1 that follows is along the lines of the convergence proof of trust region methods in [5], where the Cauchy point is computed via a purely sequential line-search along the projected gradient path. However, for Algorithm 1, the fact that the Cauchy point is yielded by an alternating projected gradient step on the model function requires some modifications in the analysis. Namely, the lower bound on the decrease in the model and the upper bound on criticality at the Cauchy point are expressed in a rather different way.

In this section, for theoretical purposes only, another first-order criticality measure different from (2) is used. We utilise the condition that \( x^* \in \Omega \) is a first-order critical point if the projected gradient at \( x^* \) is zero,

\[
P_{T_\Omega(x^*)}(-\nabla L(x^*)) = 0 \ .
\] (11)

Discussions on this first-order criticality measure can be found in [11]. It follows from Moreau’s decomposition that a point \( x^* \) satisfying (11) automatically satisfies (2). Thus, it is perfectly valid to use (2) for the convergence analysis of Algorithm 1.

4.1 Global convergence to first-order critical points

We start with an estimate of the block-coordinate-wise model decrease provided by the Cauchy points \( z_k, k \in \{1, \ldots, K\} \), of Algorithm 1. For this purpose, we define for all \( k \in \{1, \ldots, K\} \),

\[
m_k(x') := m_\Omega(z_{[1,k-1]}, x', x_{[k+1,K]})
\] (12)

where \( x' \in \mathbb{R}^{n_k} \). This corresponds to the model function evaluated at \( x' \) with the block-coordinates 1 to \( k-1 \) being fixed to the associated Cauchy points \( z_1, \ldots, z_{k-1} \) and the block-coordinates \( k+1 \) to \( K \) having values \( x_{k+1}, \ldots, x_K \). Note that by definition of the function \( m_k \),

\[
m_k(z_k) = m_{k+1}(x_{k+1})
\]

for all \( k \in \{1, \ldots, K - 1\} \).

**Lemma 3.** There exists a constant \( \nu_3 > 0 \) such that, for all \( k \in \{1, \ldots, K\} \),

\[
m_k(x_k) - m_k(z_k) \geq \nu_3 \frac{\|z_k - x_k\|_2}{s_k} \min \left\{ \Delta, \frac{1}{1 + \|H(x_k)\|_2} \frac{\|z_k - x_k\|_2}{s_k} \right\} \ .
\] (13)
Theorem. As the gradient $\nabla m_k$ is Lipschitz continuous with Lipschitz constant

$$\lambda (\nabla m_k) = \|E_k H(x)\|_2,$$

the descent Lemma yields,

$$m_k (x_k) - m_k (z_k) \geq - (\nabla m_k (x_k))^\top (z_k - x_k) - \frac{\|E_k H(x)\|_2}{2} \|z_k - x_k\|_2^2.$$

However, from the definition of $z_k$ in Algorithm 1, the convexity of $\Omega_k$ and $x_k \in \Omega_k$,

$$- (\nabla m_k (x_k))^\top (z_k - x_k) \geq \frac{\|z_k - x_k\|_2^2}{s_k}.$$

Moreover, from the proof of Lemma 2, the step-size $s_k$ satisfies

$$s_k < \frac{1}{\|E_k H(x)\|_2}.$$

Thus,

$$m_k (x_k) - m_k (z_k) \geq \frac{\|z_k - x_k\|_2^2}{2s_k}.$$

We then incorporate condition (5). If $s_k > 1/(r\|E_k H(x)\|_2)$,

$$m_k (x_k) - m_k (z_k) \geq \frac{1}{2r\|H(x)\|_2} \left( \frac{\|z_k - x_k\|_2}{s_k} \right)^2,$$

as $\|E_k H(x)\|_2 \leq \|H(x)\|_2$. If $s_k \geq \bar{s}_k$, by Lemma 1,

$$m_k (x_k) - m_k (z_k) \geq \frac{\nu_3}{2} \frac{\|z_k - x_k\|_2}{s_k}.$$

The sufficient decrease (13) is then obtained with by posing $\nu_3 := \min\{\nu_2, 1/r\}/2$.

From Lemma 3, an estimate of the decrease in the model provided by the Cauchy point $z$ is derived.

**Corollary 1** (Sufficient decrease). The following inequality holds

$$m (x) - m (z) \geq \nu_3 \sum_{k=1}^K \frac{\|z_k - x_k\|_2}{s_k} \min \left\{ \Delta, \frac{1}{1 + \|H(x)\|_2} \frac{\|z_k - x_k\|_2}{s_k} \right\}.$$

Proof. This is a direct consequence of Lemma 3 above, as

$$m (x) - m (z) = \sum_{k=1}^K m_k (x_k) - m_k (z_k).$$

from the definition of $m_k$ in Eq. (12).
In a similar manner to [5], the level of criticality reached by the Cauchy point $z$ is measured by the norm of the projected gradient of the objective, which can be upper bounded by the difference between the current iterate $x$ and the Cauchy point $z$.

**Lemma 4 (Relative error condition).** The following holds

$$\| P_{T_\Omega}(-\nabla L(z)) \|_2 \leq K (\lambda \| \nabla L \|_2 + \| H(x) \|_2) \| z - x \|_2 + \sum_{k=1}^{K} \frac{\| z_k - x_k \|_2}{s_k} .$$  \hspace{1cm} (15)

**Proof.** From the definition of $z_k$ as the projection of $x_k - s_k \nabla m_k(x_k)$ onto the closed convex set $\Omega_k$, there exists $v_k \in N_{\Omega_k}(z_k)$ such that

$$0 = v_k + \nabla m_k(x_k) + \frac{z_k - x_k}{s_k} .$$

Hence,

$$\| v_k + \nabla k L(z) \|_2 \leq \| \nabla k L(z) - \nabla k L(x) \|_2 + \| E_k H(x) \|_2 \| z - x \|_2 + \frac{\| z_k - x_k \|_2}{s_k}$$

$$\leq (\lambda \| \nabla L \|_2) \| z - x \|_2 + \frac{\| z_k - x_k \|_2}{s_k} .$$

However,

$$\| P_{N_{\Omega_k}}(z_k) (-\nabla k L(z)) + \nabla k L(z) \|_2 \leq \| v_k + \nabla k L(z) \|_2 ,$$

and by Moreau’s decomposition theorem,

$$-\nabla k L(z) = P_{N_{\Omega_k}}(z_k) (-\nabla k L(z)) + P_{T_{\Omega_k}}(z_k) (-\nabla k L(z)) .$$

Thus,

$$\| P_{T_{\Omega_k}}(z_k) (-\nabla k L(z)) \|_2 \leq (\lambda \| \nabla L \|_2 + \| H(x) \|_2) \| z - x \|_2 + \frac{\| z_k - x_k \|_2}{s_k} .$$

As the sets $\{\Omega_k\}_{k=1}^{K}$ are closed convex,

$$T_\Omega (z) = T_{\Omega_1} (z_1) \times \ldots \times T_{\Omega_K} (z_K) .$$

Subsequently,

$$\| P_{T_\Omega}(-\nabla L(z)) \|_2 \leq \sum_{k=1}^{K} \| P_{T_{\Omega_k}}(z_k) (-\nabla k L(z)) \|_2$$

and Lemma 4 follows. \hfill \Box
Lemma 5. The sequence of iterates yielded by Algorithm 1 satisfies that for all $k \in \{1, \ldots, K\}$,

$$\liminf_{l \to +\infty} \frac{\|z^l_k - x^l_k\|_2}{s^l_k} = 0,$$

where $l$ is an iteration index for Algorithm 1.

Proof. For the sake of contradiction, assume that there exists $k_0 \in \{1, \ldots, K\}$ and $\epsilon > 0$ such that

$$\frac{\|z^l_{k_0} - x^l_{k_0}\|_2}{s^l_{k_0}} \geq \epsilon$$

for all $l \geq 1$. Following the standard proof mechanism of trust-region methods, the main idea is to show that the series $\sum_{l=1}^{+\infty} \frac{1}{h^l}$ is convergent, where for all $l \geq 1$,

$$h^l := \max \{1 + \|H(x^j)\|_2 \mid j \in \{1, \ldots, l\}\}.$$ 

This is a contradiction, since $h^l$ is upper bounded by $1 + \tilde{H}$ (Assumption 5) for all $l \geq 1$. The key ingredient for convergence of the series is an upper bound on the term $\frac{1}{h^l}$. We can actually show that there exists $\epsilon' > 0$ such that $h^l \Delta^l \geq \epsilon'$ for all $l \geq 1$, which yields a proper upper bound (see the proof of Theorem 4.4 in [5]). In the sequel, we define for all $l \geq 1$, $p^l := y^l - x^l$.

Assume that there exists a subsequence $\{l_j\}$ such that $R^{l_j} \leq r_1$ for all $j \geq 1$, and $h^{l_j} \Delta^{l_j}$ goes to zero. As $h^{l_j} \geq 1$, $\Delta^{l_j} \to 0$, which implies that $\|p^{l_j}\|_2 \to 0$. Hence, there exists a positive sequence $\{\delta^{l_j}\}$ converging to zero, such that for $j$ large enough,

$$\left| L \left( y^{l_j} \right) - L \left( x^{l_j} \right) - \left( g \left( x^{l_j} \right) \right)^\top p^{l_j} - \frac{1}{2} \left( p^{l_j} \right)^\top H \left( x^{l_j} \right) p^{l_j} \right| \leq \delta^{l_j} \|p^{l_j}\|_2 + h^{l_j} \|p^{l_j}\|_2^2.$$ 

However, from Corollary 1, for $j$ large enough, as $h^{l_j} \Delta^{l_j} \to 0$,

$$m^{l_j} \left( x^{l_j} \right) - m^{l_j} \left( y^{l_j} \right) \geq \nu_3 \frac{\|z^l_{k_0} - x^l_{k_0}\|_2}{s^l_{k_0}} \min \left\{ \Delta^{l_j}, 1 + \frac{1}{\|H(x^{l_j})\|_2} \right\} \left\|z^l_{k_0} - x^l_{k_0}\|_2 \right\|_2 \right\}$$

$$\geq \nu_3 \epsilon \min \left\{ \Delta^{l_j}, \frac{\epsilon}{H^{l_j}} \right\}$$

$$\geq \frac{\nu_3 \epsilon}{\nu_2 \sqrt{n}} \|p^{l_j}\|_2.$$ 

This implies that for $j$ large enough

$$|R^{l_j} - 1| \leq \frac{\nu_2 \sqrt{n} \left( \delta^{l_j} \|p^{l_j}\|_2 + h^{l_j} \|p^{l_j}\|_2^2 \right)}{\nu_3 \epsilon \|p^{l_j}\|_2},$$
which implies \( R^b_l \to 1 \), thus contradicting \( R^b_l \leq r_1 < 1 \). By induction, this can be easily extended to the inequality \( h^l \Delta^l \geq \epsilon' \) for all \( l \geq 1 \).

We are now ready to state the main Theorem of this section. It is claimed that all limit points of the sequence \( \{x^l\} \) generated by Algorithm 1 are critical points of (1).

**Theorem 2** (Limit points are critical points). If \( x^* \) is a limit point of \( \{x^l\}_{l \geq 1} \), then there exists a subsequence \( \{l_i\} \) such that

\[
\begin{align*}
\lim_{i \to +\infty} & \left\| P_{\Omega_l}(z_{l_i}) (\nabla L(z_{l_i})) \right\|_2 = 0, \\
\end{align*}
\]

Moreover, \( P_{\Omega_l(x^*)}(\nabla L(x^*)) = 0 \), meaning that \( x^* \) is a critical point of \( L + \nu \Omega_l \).

**Proof.** Let \( \{x^l\} \) be a subsequence of \( \{x^l\} \) such that \( x^l \to x^* \). If for all \( k \in \{1, \ldots, K\} \), \( \|x^l_k - x^*_k\|_2/s^*_k \) converges to zero, we are done. Thus, given \( \epsilon > 0 \), assume that there exists \( k_0 \in \{1, \ldots, K\} \) such that for all \( i \geq 1 \), \( \|x^l_k - x^*_k\|_2/s^*_k \geq \epsilon \). From Lemma 5, given \( \epsilon' \in (0, \epsilon) \), one can build a subsequence \( \{j_i\} \) such that for \( i \) large enough, \( l_i < j_i \) and for all \( l \in \{l_i, \ldots, j_i - 1\} \), \( \|x^l_k - x^*_k\|_2/s^*_k \geq \epsilon' \) along with \( \|x^l_k - x^*_k\|_2/s^*_k < \epsilon' \) for all \( k \in \{1, \ldots, K\} \). Hence, from Corollary 1, for successful iterations \( l \) in \( \{l_i, \ldots, j_i - 1\} \), defined in lines 19 and 23 of Algorithm 1, with \( i \) large enough,

\[
L(x^l) - L(x^{l+1}) \geq r_1 \gamma_1 \nu_3 \frac{\|x^l_k - x^*_k\|_2}{s^*_k} \min \left\{ \Delta^l, \frac{1}{\mu_l} \frac{1}{s^*_k} \right\} \geq r_1 \gamma_1 \nu_3 \epsilon' \min \left\{ \Delta^l, \frac{1}{\mu_l} \epsilon' \right\}.
\]

As the sequence \( \{L(x^l)\} \) converges (non-increasing and bounded below) and \( \{h^l\} \) is upper-bounded, it follows that

\[
L(x^l) - L(x^{l+1}) \geq r_1 \gamma_1 \nu_3 \epsilon' \Delta^l,
\]

for all successful indices \( l \). However, \( \|x^{l+1} - x^l\|_\infty \leq \gamma_2 \Delta^l \). This implies that for all \( l \in \{l_i, \ldots, j_i - 1\} \),

\[
L(x^l) - L(x^{l+1}) \geq \frac{r_1 \gamma_1 \nu_3 \epsilon'}{\gamma_2 \sqrt{n}} \|x^{l+1} - x^l\|_2.
\]

(17)

Summing all inequalities (17) for \( l \in \{l_i, \ldots, j_i - 1\} \), one obtains that

\[
L(x^{l_i}) - L(x^{j_i}) \geq \frac{r_1 \gamma_1 \nu_3 \epsilon'}{\gamma_2 \sqrt{n}} \|x^{l_i} - x^{j_i}\|_2.
\]

Subsequently, \( x^{j_i} \) converges to \( x^* \) along with \( \|x^l_k - x^*_k\|_2/s^*_k \) converging to zero for all \( k \in \{1, \ldots, K\} \). From Lemma 4, this implies that \( \left\| P_{\Omega_l}(z_{l_i}) (\nabla L(z_{l_i})) \right\|_2 \) converges to zero. Finally, one gets that \( x^* \) is a critical point by lower semicontinuity of the mapping \( \|P_{\Omega_l(\cdot)} (\nabla L(\cdot))\|_2 \), which is proven in [6].
Theorem 2 above proves that all limit points of the sequence \( \{ x^l \} \) generated by Algorithm 1 are critical points. It does not actually claim convergence of \( \{ x^l \} \) to a single critical point. However, such a result can be obtained under standard regularity assumptions [24], which ensure that a critical point is an isolated local minimum.

**Assumption 7** (Strong second-order sufficiency condition). The sequence \( \{ x^l \} \) yielded by Algorithm 1 has a non-degenerate limit point \( x^* \) such that for all \( v \in \mathcal{N}_\Omega (x^*) \), where
\[
\mathcal{N}_\Omega (x^*) := \{ v \in \mathbb{R}^n \mid \forall w \in \mathcal{N}_\Omega (x^*), \ w^\top v = 0 \},
\]
one has
\[
v^\top \nabla^2 L (x^*) v \geq \kappa \| v \|_2^2,
\]
where \( \kappa > 0 \).

**Theorem 3** (Convergence to first-order critical points). The sequence \( \{ x^l \} \) generated by Algorithm 1 converges to a non-degenerate critical point \( x^* \) of \( L + \iota_\Omega \).

**Proof.** As Assumptions 5, 6 and 7 are fulfilled, this is an immediate consequence of Corollary 6.7 in [5]. \( \square \)

### 4.2 Active-set identification

In most of the trust-region algorithms for constrained optimisation, the Cauchy point acts as a predictor of the set of active constraints at a critical point. Therefore, it is important that the novel mechanism of Algorithm 1 maintains this desirable property. In this paragraph, we show that Algorithm 1 is equivalent to the standard projected line-search as an asymptotic predictor of the optimal active set.

**Lemma 6.** Given a face \( \mathcal{F} \) of \( \Omega \), there exists faces \( \mathcal{F}_1, \ldots, \mathcal{F}_K \) of \( \Omega_1, \ldots, \Omega_K \) respectively, such that \( \mathcal{F} = \mathcal{F}_1 \times \ldots \times \mathcal{F}_K \).

**Remark 7.** Given a point \( x \in \Omega \), there exists a face \( \mathcal{F} \) of \( \Omega \) such that \( x \in \text{ri} (\mathcal{F}) \). The normal cone to \( \Omega \) at \( x \) is the cone generated by the normal vectors to the active constraints at \( x \). As the set of active constraints is constant on the relative interior of a face, one can write without distinction \( \mathcal{N}_\Omega (x) \) or \( \mathcal{N} (\mathcal{F}) \).

The following Lemma is similar in nature to Lemma 7.1 in [5], yet with an adaptation in order to account for the novel way of computing the Cauchy point. In particular, it is only valid for a sufficiently high iteration count, contrary to Lemma 7.1 of [5], which can be written independently of the iteration count. This is essentially due to the fact that the Cauchy point is computed via alternating gradient steps of the model function, contrary to [5], where the gradient of the objective is used.

**Lemma 7.** Let \( x^* \) be a non-degenerate critical point of (1) that belongs to the relative interior of a face \( \mathcal{F}^* \) of \( \Omega \). Let \( \{ \mathcal{F}_k^* \}_{k=1}^K \) be faces of \( \{ \Omega_k \}_{k=1}^K \) such that \( \mathcal{F}^* = \mathcal{F}_1^* \times \ldots \times \mathcal{F}_K^* \) and thus \( x_k^* \in \text{ri} (\mathcal{F}_k^*) \), for all \( k \in \{1, \ldots, K\} \).

Assume that \( x^l \rightarrow x^* \). For \( l \) large enough, for all \( k \in \{1, \ldots, K\} \) and all \( s_k > 0 \), there exists \( \epsilon_k > 0 \) such that
\[
x_k^l \in \mathcal{B} (x_k^*, \epsilon_k) \cap \text{ri} (\mathcal{F}_k^*) \Rightarrow P_{\Omega_k} \left( x_k^l - t_k \nabla \mathcal{m}^l \left( x_{k-1}^l, x_{k,k}^l \right) \right) \in \text{ri} (\mathcal{F}_k^*) ,
\]
for all \( t_k \in (0, s_k) \).
Proof. Similarly to the proof of Lemma 7.1 in [5], the idea is to show that there exists a neighbourhood of \( x^*_k \) such that if \( x^l_k \) lies in this neighbourhood, then

\[
x^l_k - s_k \nabla k m \left( z^l_{[1,k-1]}, x^l_{[k,K]} \right) \in \text{ri} \left( F^*_k + N ( F^*_k ) \right).
\]

Lemma 7 then follows by using the properties of the projection operator onto a closed convex set and Theorem 2.3 in [5].

For simplicity, we prove the above relation for \( k = 2 \). It can be trivially extended to all \( k \in \{3, \ldots, K\} \). Let \( s_2 > 0 \) and \( l \geq 1 \).

\[
x^l_2 - s_2 \nabla_2 m \left( z^l_{[1,2]}, x^l_{[2,K]} \right) = x^l_2 - s_2 \nabla_2 L \left( x^l \right) - s_2 E_2 H^l E^l_1 \left( z^l_1 - x^l_1 \right),
\]

where the matrix \( E_k \) is defined in (7). As \( x^* \) is non-degenerate,

\[
x^* - s_2 \nabla L \left( x^* \right) \in \text{ri} \left( F^* \right) + \text{ri} \left( N ( F^* ) \right).
\]

However, as the sets \( \{ F^*_k \} \) are convex, one has [25]

\[
\text{ri} \left( F^* \right) = \text{ri} \left( F^*_1 \right) \times \ldots \times \text{ri} \left( F^*_K \right) \text{ and } N ( F^* ) = N ( F^*_1 ) \times \ldots \times N ( F^*_K ).
\]

Hence,

\[
x^l_2 - s_2 \nabla_2 L \left( x^* \right) \in \text{ri} \left( F^*_2 \right) + \text{ri} \left( N ( F^*_2 ) \right) = \text{int} \left( F^*_2 + N ( F^*_2 ) \right),
\]

by Theorem 2.3 in [5]. By continuity of \( \nabla L \), there exists \( \delta_2 > 0 \) such that

\[
\| x^l - x^* \|_2 < \delta_2 \implies x^l_2 - s_2 \nabla_2 L \left( x^l \right) \in \text{int} \left( F^*_2 + N ( F^*_2 ) \right).
\]

However, as shown beforehand (Lemma 5),

\[
\lim_{l \to +\infty} \| z^l_1 - x^l_1 \|_2 = 0.
\]

Moreover, \( E_2 H^l E^l_1 \) is bounded above (Ass. 5), subsequently for \( l \) large enough,

\[
x^l_2 - s_2 \nabla_2 m \left( z^l_1, x^l_2, \ldots, x^l_K \right) \in \text{int} \left( F^*_2 + N ( F^*_2 ) \right) \subseteq \text{ri} \left( F^*_2 + N ( F^*_2 ) \right),
\]

by Theorem 2.3 in [5]. Then, Lemma 7 follows by properly choosing the radii \( \epsilon_k \) so that \( \sum_{k=1}^{K} \epsilon_k^2 = \left( \min \{ \epsilon_k \} \right)^2 \).

We have just shown that, for a large enough iteration count \( l \), if the primal iterate \( x^l \) is sufficiently close to the critical point \( x^* \) and on the same face \( F^* \), then the set of active constraints at the Cauchy point \( z^l \) is the same as the set of active constraints at \( x^* \).

**Theorem 4.** The following holds

\[
\lim_{l \to +\infty} \| P_{T_{\Omega} (x^l)} \left( -\nabla L \left( x^l \right) \right) \|_2 = 0.
\]

Moreover, there exists \( l_0 \) such that for all \( l \geq l_0 \),

\[
A_{\Omega} ( x^l ) = A_{\Omega} ( x^* ) .
\]
Proof. The reasoning of the proof of Theorem 7.2 in [5] can be applied using Lemma 7 and Assumption 6. The first step is to show that the Cauchy point $z$ identifies the optimal active set after a finite number of iterations. This is guaranteed by Theorem 2.2 in [5], since $P_{T_d(z^l)}(-\nabla L(z^l)) \to 0$ by Theorem 2, and the sequence $\{x^l\}$ converges to a non-degenerate critical point by Theorem 3. Lemma 7 is useful to show that if $x^l$ is close enough to $x^*$, then the Cauchy point $z^l$ remains in the relative interior of the same face of $x^l$, and thus the active constraints do not change after some point. 

Theorem 4 shows that the optimal active set is identified after a finite number of iterations, which corresponds to the behaviour of the gradient projection in standard trust-region methods. This fact is crucial for the local convergence analysis of the sequence $\{x^l\}$, as fast local convergence rate can generally be obtained if the active-set does not settle down.

4.3 Local convergence rate

In this paragraph, we show that the local convergence rate of the sequence $\{x^l\}$ generated by Algorithm 1 is Q-superlinear. To establish this property, a key step is to prove that the trust-region radius is ultimately bounded away from zero. It turns out that the regularisation of the trust-region problem (8) plays an important role in this proof. As shown next, after a large enough number of iterations, the trust-region radius does not interfere and an inexact Newton step is always taken in the refinement step, implying fast local convergence depending on the level of accuracy in the computation of the Newton direction.

Lemma 8. There exists an index $l_1 \geq 1$ and $\Delta^* > 0$ such that for all $l \geq l_1$, $\Delta^l \geq \Delta^*$.

Proof. The idea is to show that the ratio $R^l$ converges to one, which implies that all iterations are ultimately successful, and subsequently, by the mechanism of Algorithm 1, the trust-region radius is bounded away from zero asymptotically. For all $l \geq 1$,

$$|R^l - 1| = \left| \frac{L(y^l) - L(x^l) - (y^l)^\top(y^l - x^l) - \frac{1}{2}(y^l - x^l)^\top H^l(y^l - x^l)}{m^l(x^l) - m^l(y^l)} \right|. \quad (20)$$

However,

$$m^l(x^l) - m^l(y^l) = m^l(x^l) - m^l(z^l) + m^l(z^l) - m^l(y^l)$$

$$\geq \frac{\eta}{2} \|z^l - x^l\|_2^2 + \frac{\sigma}{2} \|y^l - z^l\|_2^2$$

$$\geq \frac{\min \{\eta, \sigma\} \left( \|z^l - x^l\|_2^2 + \|y^l - z^l\|_2^2 \right)}{2}$$

$$\geq \frac{\min \{\eta, \sigma\}}{2} \max \left\{ \|z^l - x^l\|_2^2, \|y^l - z^l\|_2^2 \right\},$$

and

$$\|y^l\|_2 \leq \|y^l - z^l\|_2 + \|z^l - x^l\|_2$$

$$\leq 2 \max \left\{ \|y^l - z^l\|_2, \|z^l - x^l\|_2 \right\}.$$
Hence,

\[ m^l (x^l) \geq \min \left\{ \frac{\eta \sigma}{8} \right\} \| p^l \|_2^2. \]

Moreover, using the mean-value theorem, one obtains that the numerator in (20) is smaller than

\[ \frac{1}{2} \psi^l \| p^l \|_2^2, \]

where

\[ \psi^l := \sup_{\tau \in [0, 1]} \| \nabla^2 L (x^t + \tau p^t) - \nabla^2 L (x^t) \|_2. \] (21)

Subsequently, we have

\[ |R^l - 1| \leq \frac{4}{\min \{ \eta, \sigma \}} \psi^l, \]

and the result follows by showing that \( p^l \) converges to zero. Take \( l \geq l_0 \), where \( l_0 \) is as in Theorem 4. Thus, \( p^l \in \mathcal{N}(\mathcal{F}^*)^\perp \). However, from the model decrease, one obtains

\[ \frac{1}{2} (p^l)^\top \nabla^2 L (x^l) p^l \leq -\nabla L (x^l)^\top p^l. \]

From Theorem 3, the sequence \( \{ x^l \} \) converges to \( x^* \), which satisfies the strong second-order sufficiency condition 7. Hence, by continuity of the hessian \( \nabla^2 L \) and the fact that \( A_\Omega (x^t) = A_\Omega (x^*) \), one may claim that there exists \( l_1 \geq l_0 \) such that for all \( l \geq l_1 \), for all \( v \in \mathcal{N}_\Omega (x^l) \perp = \mathcal{N}(\mathcal{F}^*) \perp \),

\[ v^\top \nabla^2 L (x^l) v \geq \kappa \| v \|_2^2. \]

Thus, by Moreau’s decomposition, it follows that

\[ \frac{\kappa}{2} \| p^l \|_2^2 \leq (P_{\mathcal{F}^*_\perp} (x^l)) (\nabla L (x^l)) + P_{\mathcal{N}_\Omega (x^l)} (-\nabla L (x^l)))^\top p^l \leq \| P_{\mathcal{F}^*_\perp} (x^l)) (\nabla L (x^l)) \|_2 \| p^l \|_2, \]

since \( p^l \in \mathcal{N}(\mathcal{F}^*)^\perp \). Finally, \( p^l \) converges to zero, as a consequence of Lemma 4 and the fact that \( \| z^l - x^l \|_2 \) converges to 0.

The refinement step in Algorithm 1 actually consists of a truncated Newton method, in which the Newton direction is generated by an iterative procedure, namely the distributed sCG described in Algorithm 3. The Newton iterations terminate when the residual \( \hat{s} \) is below a tolerance that depends on the norm of the projected gradient at the current iteration. In Algorithm 3, the stopping condition is set so that at every iteration \( l \geq 1 \), there exists \( \xi^l \in (0, 1) \) satisfying

\[ \| Z^l (Z^l)^\top (g_{\sigma^l} (x^l) + H_{\sigma^l} (x^l) p^l) \|_2 \leq \xi^l \| Z^l (Z^l)^\top g (x^l) \|_2. \] (22)

The local convergence rate of the sequence \( \{ x^l \} \) generated by Algorithm 1 is controlled by the sequences \( \{ \xi^l \} \) and \( \{ \sigma^l \} \), as shown in the following Theorem.
Theorem 5 (Local linear convergence). Assume that the direction $p$ yielded by Algorithm 3 satisfies (22) if $\|p\|_\infty \leq \gamma^* \Delta$ and $A_O(x) = A_O(x + p)$, given $\gamma^* \in (0, \gamma_2)$. For a small enough $\sigma$, the sequence $\{x^l\}$ generated by Algorithm 1 converges Q-linearly to $x^*$ if $\xi^* < 1$ is small enough, where

$$\xi^* := \limsup_{l \to +\infty} \xi^l .$$

If $\xi^* = 0$, the Q-linear convergence ratio can be made arbitrarily small by properly tuning $\sigma$, resulting in almost Q-superlinear convergence.

Proof. Throughout the proof, we assume that $l$ is large enough so that the active-set has settled down to $A_O(x^*)$ and that $p^l$ satisfies condition (22). This is ensured by Lemma 8 and Theorem 4, as the sequence $\{p^l\}$ converges to zero. Thus, we can write $Z^l = Z^*$. The orthogonal projection onto $\mathcal{N}(F^*)^\perp$ is represented by the matrix $Z^* (Z^*)^\top$. A first order development yields a positive sequence $\{\delta^l\}$ converging to zero such that

$$\|Z^* (Z^*)^\top \nabla L (x^{l+1})\|_2 \leq \|Z^* (Z^*)^\top (g(x^l) + H(x^l)p^l)\|_2 + \delta^l \|p^l\|_2$$

$$\leq \frac{2\delta^l}{\kappa} \|Z^* (Z^*)^\top g(x^l)\|_2 + \|Z^* (Z^*)^\top (g_{\sigma^l}(x^l) + H_{\sigma^l}(x^l)p^l)\|_2$$

$$+ \sigma \|Z^* (Z^*)^\top \left( \frac{p^l}{2} + z^l - x^l \right)\|_2$$

$$\leq \left( \frac{2\delta^l}{\kappa} + \xi^l \right) \|Z^* (Z^*)^\top g(x^l)\|_2$$

$$+ \sigma \left( \frac{1}{\kappa} + \frac{\|Z^* (Z^*)^\top (z^l - x^l)\|_2}{\|Z^* (Z^*)^\top g(x^l)\|_2} \right) \|Z^* (Z^*)^\top g(x^l)\|_2 .$$

where the second inequality follows from the last inequality in Lemma 8, and the definition of $g_{\sigma}$ and $H_{\sigma}$ in Eq. (10). However, from the computation of the Cauchy point described in paragraph 3.2 and Assumption 5, the term

$$\frac{\|Z^* (Z^*)^\top (z^l - x^l)\|_2}{\|Z^* (Z^*)^\top g(x^l)\|_2}$$

is bounded by a constant $C > 0$. Hence,

$$\frac{\|Z^* (Z^*)^\top \nabla L (x^{l+1})\|_2}{\|Z^* (Z^*)^\top \nabla L (x^l)\|_2} \leq \frac{2\delta^l}{\kappa} + \xi^l + \sigma \left( \frac{1}{\kappa} + C \right) .$$

Moreover, a first-order development provides us with a constant $\chi > 0$ such that

$$\|Z^* (Z^*)^\top \nabla L (x^l)\|_2 \leq \left( \tilde{H} + \chi \right) \|x^l - x^*\|_2 .$$

There also exists a positive sequence $\{\epsilon^l\}$ converging to zero such that

$$\|Z^* (Z^*)^\top \nabla L (x^{l+1})\|_2 \geq \|Z^* (Z^*)^\top \nabla^2 L (x^*) (x^{l+1} - x^*)\|_2 - \epsilon^l \|x^{l+1} - x^*\|_2 .$$
However, since $x^{l+1} - x^*$ lies in $N(x^*)^\perp$, $Z^* (Z^*)^\top (x^{l+1} - x^l) = x^{l+1} - x^l$. Thus, by Assumption (19),

$$
\|Z^* (Z^*)^\top \nabla L (x^{l+1})\|_2 \geq (\kappa - \ell') \|x^{l+1} - x^*\|_2 ,
$$

which implies that, for $l$ large enough, there exists $\tau \in (0, \kappa)$ such that

$$
\|Z^* (Z^*)^\top \nabla L (x^{l+1})\|_2 \geq (\kappa - \tau) \|x^{l+1} - x^*\|_2 .
$$

Finally,

$$
\frac{\|x^{l+1} - x^*\|_2}{\|x^l - x^*\|_2} \leq \frac{\bar{H} + \chi}{\kappa - \tau} \left( \frac{2\delta^l}{\kappa} + \xi^l + \sigma \left( \frac{1}{\kappa} + C \right) \right) ,
$$

which yields the result.

5 Numerical examples

The optimal AC power flow constitutes a challenging class of non-convex problems for benchmarking optimisation algorithms and software. It has been used very recently in the testing of a novel adaptive augmented Lagrangian technique [12]. The power flow equations form a set of nonlinear coupling constraints over a network. Some distributed optimisation strategies have already been explored for computing OPF solutions, either based on convex relaxations [21] or non-convex heuristics [20]. As the convex relaxation may fail in a significant number of cases [4], it is also relevant to explore distributed strategies for solving the OPF in its general non-convex formulation. Naturally, all that we can hope for with this approach is a local minimum of the OPF problem. Algorithm 1 is tested on the augmented Lagrangian sub-problems obtained via a polar coordinates formulation of the OPF equations, as well as rectangular coordinates formulations. Algorithm 1 is run as an inner solver inside a standard augmented Lagrangian loop [1] and in the more sophisticated LANCELOT dual loop [9]. More precisely, if the OPF problem is written in the following form

$$
\begin{align*}
\text{minimise} & \quad f(x) \\
\text{s.t.} & \quad g(x) = 0 \\
& \quad x \in \mathcal{X} ,
\end{align*}
$$

with $\mathcal{X}$ bound constraints, an augmented Lagrangian loop consists in computing an approximate critical point of the auxiliary program

$$
\begin{align*}
\text{minimise} & \quad L_\rho(x, \mu) := f(x) + \left( \mu + \frac{\rho}{2} g(x) \right)^\top g(x) \\
\text{s.t.} & \quad x \in \mathcal{X} ,
\end{align*}
$$

with $\mu$ a dual variable associated to the power flow constraints and $\rho > 0$ a penalty parameter, which are both updated after a sequence of primal iterations in program (24). Using the standard dual update formula, only local convergence of the dual sequence can be proven [1]. On the contrary, in the LANCELOT outer loop, the dual variable $\mu$ and the penalty parameter $\rho$ are updated according to the level of satisfaction of the power flow (equality) constraints, resulting in global convergence of the dual sequence [9]. In order to test Algorithm 3, we use it to compute approximate critical points
of the auxiliary problems (23), which are of the form (1). The rationale behind choosing LANCELOT or a standard augmented Lagrangian method as the outer loop is that LANCELOT interrupts the inner iterations at an early stage, based on a KKT tolerance that is updated at every dual iteration. Hence, it does not allow one to really measure the absolute performance of Algorithm 1, although it is likely more efficient than a standard augmented Lagrangian for computing a solution of the OPF program. Thus, for all cases presented next, we provide the results of the combination of Algorithm 1 with a basic augmented Lagrangian (AL) and LANCELOT. The basic AL loop is utilised to show the performance of Algorithm 1 as a bound-constrained solver, whereas LANCELOT is expected to provide better overall performance. All results are compared to the solution yielded by the nonlinear interior-point solver IPOPT [30] with the sparse linear solver MA27.

5.1 Optimal AC power flow in polar coordinates

We consider the optimal AC power flow problem in polar coordinates

\[
\begin{align*}
\text{minimise} & \quad \sum_{g \in \mathcal{G}} e_0^g + e_1^g p_g^G + c_2 \left(p_g^G\right)^2 \\
\text{s.t.} & \quad \sum_{g \in \mathcal{G}} p_g^G = \sum_{d \in \mathcal{D}_b} P_d^L + \sum_{b' \in \mathcal{B}_b} p_{bb'} + G_b^L v_b^2 \\
& \quad \sum_{g \in \mathcal{G}} q_g^G = \sum_{d \in \mathcal{D}_b} Q_d^L + \sum_{b' \in \mathcal{B}_b} q_{bb'} - B_b^L v_b^2 \\
& \quad p_{bb'}^L = G_{bb'} v_b^2 + (G_{bb'} \cos (\theta_b - \theta_{b'}) + B_{bb'} \sin (\theta_b - \theta_{b'})) v_b v_{b'} \\
& \quad q_{bb'}^L = -B_{bb'} v_b^2 + (G_{bb'} \sin (\theta_b - \theta_{b'}) - B_{bb'} \cos (\theta_b - \theta_{b'})) v_b v_{b'} \\
& \quad (p_{bb'}^L)^2 + (q_{bb'}^L)^2 + s_{bb'} = (S_{bb'}^M) \\
& \quad v_b^L \leq v_b \leq v_b^U \\
& \quad p^L \leq p_g^G \leq p^U \\
& \quad q^L \leq q_g^G \leq q^U \\
& \quad s_{bb'} \geq 0 ,
\end{align*}
\]

which corresponds to the minimisation of the overall generation cost, subject to power balance constraints at every bus \(b\) and power flow constraints on every line \(bb'\) of the network, where \(\mathcal{G}\) denotes the set of generators and \(\mathcal{B}_b\) is the set of generating units connected to bus \(b\). The variables \(p_g^G\) and \(q_g^G\) are the active and reactive power output at generator \(g\). The set of loads connected to bus \(b\) is denoted by \(\mathcal{D}_b\). The parameters \(P_d^L\) and \(Q_d^L\) are the demand active and reactive power at load unit \(d\). The letter \(\mathcal{B}_b\) represents the set of buses connected to bus \(b\). Variables \(p_{bb'}^L\) and \(q_{bb'}^L\) are the active and reactive power flow through line \(bb'\). Variables \(v_b\) and \(\theta_b\) denote the voltage magnitude and voltage angle at bus \(b\). Constants \(v_b^L\), \(v_b^U\) are lower and upper bounds on the voltage magnitude at bus \(b\). Constants \(p^L\), \(p^U\), \(q^L\) and \(q^U\) are lower and upper bounds on the active and reactive power generation. It is worth noting that a slack variable \(s_{bb'}\) has been added at every line \(bb'\) in order to turn the usual inequality constraint on the power flow through line \(bb'\) into an equality constraint. The derivation of the optimal power flow problem in polar form can be found in [34].
As a numerical test example for Algorithm 1, we consider a particular instance of NLP (25) on the 9-bus transmission network shown in Fig. 1. As in (24), the augmented Lagrangian sub-problem is obtained by relaxing the equality constraints associated with buses and lines in (25). The bound constraints, which can be easily dealt with via projection, remain unchanged. One should notice that NLP (25) has partially separable constraints and objective, so that LANCELOT could efficiently deal with it, yet in a purely centralised manner. In some sense, running Algorithm 1 in a LANCELOT dual loop can be seen as a first step towards a distributed implementation of LANCELOT for solving the AC-OPF. It is worth noting that the dual updates only require exchange of information between neighbouring nodes and lines. Moreover, each LANCELOT dual update requires a global communication, as the full nonlinear equality constraint needs to be checked [9]. For the 9-bus example in Fig. 1,

Fig. 1: The 9-bus transmission network from http://www.maths.ed.ac.uk/optenergy/LocalOpt/.

the Cauchy search of Algorithm 1 on the augmented Lagrangian sub-problem (24) can be carried out in five parallel steps. This can be observed by introducing local variables for every bus \( b \in \{1, \ldots, 9\} \),

\[ x_b := (v_b, \theta_b)^\top, \]

and for every line

\[ bb' \in \{1, 4\}, \{4, 5\}, \{4, 9\}, \{8, 9\}, \{2, 8\}, \{7, 8\}, \{6, 7\}, \{3, 6\}, \{5, 6\} \],
with the line variable $y_{bb'}$ being defined as

$$y_{bb'} := (p_{bb'}, q_{bb'}, s_{bb'})^\top.$$ 

The line variables $y_{bb'}$ can be first updated in three parallel steps, which corresponds to

$$\{y(2,8), y(6,7), y(4,9)\}, \{y(7,8), y(3,6), y(4,5)\}, \{y(8,9), y(5,6), y(1,4)\}.$$ 

Then, the subset

$$\{x_1, x_2, x_3, x_5, x_7, x_9\}$$

can be updated, followed by the subset

$$\{x_4, x_6, x_8\}.$$ 

As a result, backtracking iterations can be run in parallel at the nodes potentially associated with each line and bus. If a standard trust-region Newton method would be applied, the projected line-search would have to be computed on the same central node without a bound on the number of iterations. Thus, the activity identification phase of Algorithm 1 allows one to reduce the number of global communications involved in the whole procedure. The results obtained via a basic augmented Lagrangian loop and a LANCELOT outer loop are presented in Tables 1 and 2 below. The data is taken from the archive [http://www.maths.ed.ac.uk/optenergy/LocalOpt/](http://www.maths.ed.ac.uk/optenergy/LocalOpt/). To obtain the results presented in Tables 1 and 2, the regularisation parameter $\sigma$ of Algorithm 3 is set to $1\cdot 10^{-10}$. For Table 1, the maximum number of iterations in the inner loop (Algorithm 1) is fixed to 300 and the stopping tolerance on the level of satisfaction of the KKT conditions to $1\cdot 10^{-5}$. For Table 2 (LANCELOT), the maximum number of inner iterations is set to 100 for the same stopping tolerance on the KKT conditions. In the refinement step of Algorithm 3, a block-diagonal preconditioner is applied. It is worth noting that the distributed implementation of Algorithm 3 is not affected by such a change. To obtain the results of Table 1, the initial penalty parameter $\rho$ is set to 10 and is multiplied by 30 at each outer iteration. In the LANCELOT loop, it is multiplied by 100. The regularisation coefficients $\alpha_i$ in the activity detection phase of Algorithm 2 are set to $1\cdot 10^{-7}$. In the end, an objective value of 2733.55 up to feasibility $1.64\cdot 10^{-8}$ of the Power Flow (PF) constraints is obtained, whereas the interior-point solver IPOPT, provided with the same primal-dual initial guess, yields an objective value of 2733.5 up to feasibility $2.23\cdot 10^{-11}$. The first column of the tables corresponds to the number of dual iterations, the second column to the number of iterations in the main

| Outer iter. count | # inner it. | # cum. stCG | Inner KKT | PF eq. constr. |
|-------------------|-------------|-------------|-----------|----------------|
| 1                 | 79          | 388         | 2.01·10^{-7} | 0.530         |
| 2                 | 2           | 40          | 2.71·10^{-10} | 0.530         |
| 3                 | 300         | 2215        | 2.39·10^{-2}  | 0.292         |
| 4                 | 101         | 2190        | 6.50·10^{-4}  | 6.56·10^{-3}  |
| 5                 | 123         | 2873        | 2.10·10^{-3}  | 5.02·10^{-6}  |
| 6                 | 56          | 1194        | 4.14·10^{-2}  | 1.11·10^{-10} |

Table 1: Results for the 9-bus OPF (Fig. 1) using a standard augmented Lagrangian dual loop with Algorithm 1 as primal solver.
Table 2: Results for the 9-bus OPF (Fig. 1) using a LANCELOT dual loop with Algorithm 1 as primal solver.

| Outer iter. count | # inner it. | # cum. sCG | Inner KKT | PF eq. constr. |
|-------------------|-------------|------------|-----------|----------------|
| 1                 | 37          | 257        | 7.29 \cdot 10^{-2} | 0.530 |
| 2                 | 5           | 25         | 1.01 \cdot 10^{-2} | 0.530 |
| 3                 | 6           | 71         | 3.23 \cdot 10^{-5} | 0.530 |
| 4                 | 100         | 1330       | 8.30 \cdot 10^{-3} | 4.33 \cdot 10^{-2} |
| 5                 | 100         | 1239       | 1.80 \cdot 10^{-3} | 2.53 \cdot 10^{-3} |
| 6                 | 100         | 2269       | 4.33 \cdot 10^{-2} | 2.69 \cdot 10^{-5} |
| 7                 | 64          | 1541       | 3.2 \cdot 10^{-3}  | 1.64 \cdot 10^{-8} |

5.2 Optimal AC power flow on distribution networks

Algorithm 1 is then applied to solve two OPF problems in rectangular coordinates on distribution networks. Both 47-bus and 56-bus networks are taken from [16]. Our results are compared against the nonlinear interior-point solver IPOPT [30], which is not amenable to a fully distributed implementation, and the SOCP relaxation proposed by [16], which may be distributed (as convex) but fails in some cases, as shown next. It is worth noting that any distribution network is a tree, so a minimum colouring scheme consists of two colours, resulting in 2 parallel steps for the activity identification in Algorithm 1.

5.2.1 On the 56-bus OPF

On the 56-bus OPF, an objective value of 233.9 is obtained with feasibility 8.00 \cdot 10^{-7}, whereas the nonlinear solver IPOPT yields an objective value of 233.9 with feasibility 5.19 \cdot 10^{-7} for the same initial primal-dual guess.

In order to increase the efficiency of Algorithm 1, following a standard recipe, we build a block-diagonal pre-conditioner from the hessian of the augmented Lagrangian by extracting block-diagonal elements corresponding to buses and lines. Thus, constructing and applying the pre-conditioner can be done in parallel and does not affect the distributed nature of our Algorithm. In Fig. 2, the satisfaction of the KKT conditions for the bound constrained problem (24) is plotted for a preconditioned refinement step and non-preconditioned one. One can conclude from Fig. 2 that preconditioning the refinement step does not only affect the number of iterations of the sCG Algorithm 3.
Fig. 2: KKT satisfaction vs iteration count in the seventh LANCELOT subproblem formed on the AC-OPF with 56 buses. Without preconditioning of the sCG (dashed), with block-diagonal preconditioning (full).

(Fig. 5), but also the performance of the main loop of Algorithm 1. From a distributed perspective, it is very appealing, for it leads to a strong decrease in the overall number of global communications.

Fig. 3: Active-set history in the first LANCELOT iteration for the 56-bus AC-OPF. PALM is used as activity detector (Algorithm 1).
From Fig. 3, PALM proves very efficient at identifying the optimal active set in a few iterations (more than 20 constraints enter the active set in the first three iterations and 50 constraints are dropped in the following two iterations), which is a proof of concept for the analysis of Section 4. In Fig. 4, the PF constraints are evaluated after a run of Algorithm 1 on program (24). The dual variables and penalty coefficient are updated at each outer iteration. Overall, the coupling of Algorithm 1 with the augmented Lagrangian appears to be successful.

![Figure 4: Norm of PF constraints on the 56-bus network against dual iterations of a LANCELOT dual loop with Algorithm 1 as primal solver.](image)

| Outer iter. count | # inner it. | # cum. sCG | Inner KKT | PF eq. constr. |
|-------------------|-------------|------------|-----------|----------------|
| 1                 | 122         | 1382       | 8.45 \cdot 10^{-9} | 6.68           |
| 2                 | 189         | 4486       | 6.71 \cdot 10^{-9} | 1.49 \cdot 10^{-1} |
| 3                 | 139         | 11865      | 9.87 \cdot 10^{-8} | 8.79 \cdot 10^{-4} |
| 4                 | 49          | 3958       | 6.75 \cdot 10^{-6} | 7.92 \cdot 10^{-6} |
| 5                 | 9           | 936        | 5.45 \cdot 10^{-7} | 4.58 \cdot 10^{-9} |

Table 3: Results for the 56-bus OPF of [16] using a basic augmented Lagrangian dual loop with Algorithm 1 as primal solver.

Tables 3 and 4 are obtained with an initial penalty coefficient $\rho = 10$ and a multiplicative coefficient of 20.
Fig. 5: Cumulative sCG iterations vs iteration count in the third LANCELOT subproblem formed on the AC-OPF with 56 buses. Without preconditioning of the sCG (blue), with block-diagonal preconditioning (black).

Table 4: Results for the 56-bus OPF of [16] using a LANCELOT dual loop with Algorithm 1 as primal solver.

| Outer iter. count | # inner it | # cum. sCG | Inner KKT       | PF eq. constr. |
|-------------------|------------|------------|-----------------|----------------|
| 1                 | 100        | 924        | 9.74 · 10⁻⁸     | 6.42           |
| 2                 | 133        | 3587       | 2.40 · 10⁻³     | 3.60 · 10⁻¹    |
| 3                 | 54         | 4531       | 1.03 · 10⁻⁴     | 4.00 · 10⁻³    |
| 4                 | 100        | 858        | 4.20 · 10⁻⁶     | 1.02 · 10⁻³    |
| 5                 | 42         | 3288       | 4.37 · 10⁻⁶     | 2.32 · 10⁻⁴    |
| 6                 | 13         | 916        | 1.82 · 10⁻⁵     | 4.35 · 10⁻⁵    |
| 7                 | 40         | 6878       | 3.70 · 10⁻⁷     | 8.16 · 10⁻⁶    |
| 8                 | 6          | 420        | 4.64 · 10⁻⁶     | 4.97 · 10⁻⁷    |

5.2.2 On the 47-bus OPF

On the 47-bus OPF, a generating unit was plugged at node 12 (bottom of the tree) and the load at the substation was decreased to 3 pu. On this modified problem, the SOCP relaxation provides a solution, which does not satisfy Ohm’s law. An objective value of 502.3 is obtained with feasibility 2.57 · 10⁻⁷ for both the AL loop (Tab. 5) and the LANCELOT loop (Tab. 6). The SOCP relaxation returns an objective value of 265.75, but physically impossible, as the PF constraints are not satisfied. The nonlinear solver IPOPT yields an objective value of 502.3 with feasibility 5.4 · 10⁻⁸.
6 Conclusion

A trust-region Newton method based on distributed activity detection has been described and analysed. In particular, as a result of a proximal regularisation of the trust-region problem with respect to the Cauchy point yielded by an alternating projected gradient sweep, global and fast local convergence to first-order critical points has been proven under standard regularity assumptions. It has been argued further how the approach can be implemented in distributed platforms. The proposed strategy has been successfully applied to solve various non-convex OPF problems, for which distributed algorithms are currently raising interest.

References

1. Bertsekas, D.P. Constrained optimization and Lagrange multiplier methods. Athena Scientific, 1982.
2. Bertsekas, D.P. and Tsitsiklis, J.N. Parallel and distributed computation: numerical methods. Athena Scientific, 1997.
3. Bolte, J. and Sabach, S. and Teboulle, M. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. Mathematical Programming, 146(1-2):459–494, 2014.
4. Bukhsh, W.A. and Grotchey, A. and McKinno, K.I.M. and Trodden, P.A. Local solutions of the optimal power flow problem. IEEE Transactions on Power Systems, 28(4), November 2013.
5. J.V. Burke, J.J. Moré, and G. Toraldo. Convergence properties of trust region methods for linear and convex constraints. Mathematical Programming, 47:305–336, 1990.
6. Calamai, P.H. and More, J.J. Projected gradient methods for linearly constrained problems. Mathematical Programming, 39:93–116, 1987.
7. M. Chiang, S. Low, A. Calderbank, and J.C. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. Proceedings of the IEEE, 95(1):255–312, January 2007.
8. Cohen, G. Auxiliary problem principle and decomposition of optimization problems. Journal of Optimization Theory and Applications, 32(3):277–305, November 1980.
9. Conn, A. and Gould, N.I.M. and Toint, P.L. A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds. SIAM Journal on Numerical Analysis, 28:545–572, 1991.
10. Conn, A.R. and Gould, N.I.M and Toint, P.L. Global convergence of a class of trust region algorithms for optimization with simple bounds. SIAM Journal on Numerical Analysis, 25(2), 1988.
11. Conn, A.R. and Gould, N.I.M. and Toint, P.L. Trust Region Methods. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.
12. Curtis, F.E. and Gould, N.I.M. and Jiang, H. and Robinson, D.P. Adaptive augmented Lagrangian methods: algorithms and practical numerical experience. Technical Report 14T-006, COR@L Laboratory, Department of ISE, Lehigh University, 2014. In first review for Optimization Methods and Software.
13. D’Azevedo, E. and Eijkhout, V. and Romine, C. LAPACK Working Note 56: Reducing Communication Costs in the Conjugate Gradient Algorithm on Distributed Memory Multiprocessors. Technical report, University of Tennessee, Knoxville, TN, USA, 1993.
14. Fei, Y. and Guodong, R. and Wang, B. and Wang, W. Parallel L-BFGS-B algorithm on GPU. Computers and Graphics, 40:1–9, 2014.
15. Fernández, D. and Solodov, M.V. Local convergence of exact and inexact augmented Lagrangian methods under the second-order sufficient optimality condition. SIAM Journal on Optimization, 22(2):384–407, 2012.
16. Gan, L. and Li, N. and Topcu, U. and Low, S.H. Exact convex relaxation of optimal power flow in radial network. IEEE Transactions on Automatic Control, 2014. Accepted for publication.
17. Hamdi, A. and Mishra, S.K. Decomposition methods based on augmented Lagrangian: a survey. In Topics in nonconvex optimization. Mishra, S.K., 2011.
18. Hours, J.-H. and Jones, C.N. A parametric non-convex decomposition algorithm for real-time and distributed NMPC. Technical Report EPFL-REPORT-201489, École Polytechnique Fédérale de Lausanne, 2014. http://arxiv.org/abs/1408.5120.
19. Hours, J.-H. and Jones, C.N. An augmented Lagrangian coordination-decomposition algorithm for solving distributed non-convex programs. In Proceedings of the 2014 American Control Conference, pages 4312–4317, June 2014.
20. Kim, B.H. and Balduck, R. Coarse-grained distributed optimal power flow. IEEE Transactions on Power Systems, 12(2), May 1997.
21. Lam, A.Y.S. and Zhang, B. and Tse, D.N. Distributed algorithms for optimal power flow. In Proceedings of the 51st Conference on Decision and Control, pages 430–437, December 2012.
22. J.J. Moreau. Décomposition orthogonale d’un espace hilbertien selon deux cônes mutuellement polaires. C.R. Académie des Sciences, 255:238–240, 1962.
23. Necora, I. and Savorgnan, C. and Tran DinH, Q. and Suykens, J. and Diehl, M. Distributed nonlinear optimal control using sequential convex programming and smoothing techniques. In Proceedings of the 48th Conference on Decision and Control, 2009.
24. Nocedal, J. and Wright, S. Numerical Optimization. Springer, New-York, 2006.
25. Rockafellar, R.T. and Wets, R.J.-B. Variational Analysis. Springer, 2009.
26. Steihaug, T. The conjugate gradient method and trust regions in large scale optimization. SIAM Journal on Numerical Analysis, 20:626–637, 1983.
27. Tran Dinh, Q. and Necora, I. and Diehl, M. A dual decomposition algorithm for separable nonconvex optimization using the penalty framework. In Proceedings of the 52nd Conference on Decision and Control, 2013.
28. Tran-Dinh, Q. and Savorgnan, C. and Diehl, M. Combining Lagrangian decomposition and excessive gap smoothing technique for solving large-scale separable convex optimization problems. Computational Optimization and Applications, 55(1):75–111, 2013.
29. Verschoor, M. and Jalba, A.C. Analysis and performance estimation of the Conjugate Gradient method on multiple GPUs. Parallel Computing, 38(10-11):552–575, 2012.
30. Wächter, A. and Biegler, L.T. On the implementation of a primal-dual interior point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming, 106(1):25–57, 2006.
31. Xue, D. and Sun, W. and Qi, L. An alternating structured trust-region algorithm for separable optimization problems with nonconvex constraints. Computational Optimization and Applications, 57:365–386, 2014.
32. Zavala, V.M. and Auitescu, M. Scalable nonlinear programming via exact differentiable penalty functions and trust-region Newton methods. SIAM Journal on Optimization, 24(1):528–558, 2014.
33. Zavala, V.M. and Laird, C.D. and Biegler, L.T. Interior-point decomposition approaches for parallel solution of large-scale nonlinear parameter estimation problems. Chemical Engineering Science, 63:4834–4845, 2008.
34. Zhu, J. Optimization of Power System Operation. IEEE Press, Piscataway, NJ, USA, 2009.