An augmented Lagrangian algorithm for constrained nonlinear least-squares

Pierre Borie*, Fabian Bastin*

Abstract

We present an algorithm for solving nonlinear least-squares problems subject to a mix of nonlinear and linear constraints. The nonlinear constraints are handled by reformulating the objective as the augmented Lagrangian function while linear constraints are handled directly. Each iteration consists into approximalety solving a linearly constrained problem by the means of a gradient projection technique. Our approach also involves a structured approximation of the augmented Lagrangian Hessian. We show global convergence of the method and provide numerical experiments.

Contents

*University of Montreal, Department of Computer Science and Operations Research, Montreal, QC, Canada

1 Introduction

In this paper, we consider the following nonlinear least-squares problem

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \frac{1}{2}\|r(x)\|^2 \\ \text{s.t.} \quad & c(x) = 0 \\ & x \in \Omega \end{aligned} \tag{1.1}$$

where the residuals function $r\colon \mathbb{R}^n \to \mathbb{R}^{n_r}$ and the constraint function $c\colon \mathbb{R}^n \to \mathbb{R}^{n_c}$ are assumed to be twice continuously differentiable and $\|\cdot\|$ is the ℓ_2 norm. The set Ω consists of linear constraints and is defined as

$$\Omega = \{x \in \mathbb{R}^n \mid Ax = b,\ l \leq x \leq u\}, \tag{1.2}$$

$A \in \mathbb{R}^{m\times n}$, with $m \leq n$, matrix A, $b \in \mathbb{R}^m$, with $m < n$ and $l, u \in \mathbb{R}^n$. Without loss of generality, some components of the latter two vectors can be set to $\pm\infty$ for unbounded parameters.

The Lagrangian for problem (1.1), with respect to the nonlinear constraints, is defined by

$$\mathcal{L}(x, \lambda) = f(x) + \langle \lambda, c(x)\rangle, \tag{1.3}$$

where $\lambda \in \mathbb{R}^{n_c}$ is the vector of Lagrange multipliers and $\langle\cdot,\cdot\rangle$ is the standard inner product.

An algorithm for solving problem (1.1) aims to find a first-order critical point (x^*, λ^*) such that

$$\begin{aligned} & x^* \in \Omega \text{ and } c(x^*) = 0 \\ & P_\Omega\left[x^* - \nabla_x \mathcal{L}(x^*, \lambda^*)\right] = x^*, \end{aligned} \tag{1.4}$$

where $P_\Omega[\cdot]$ is the projector operator onto Ω, defined, for any vector x, as

$$P_\Omega[x] = \operatorname*{argmin}_{v\in\Omega} \|x - v\|. \tag{1.5}$$

We end this section by introducing some notations. Jacobian matrices of r and c evaluated at x are respectively noted $J(x)$ and $C(x)$. Components of a vector $x \in \mathbb{R}^n$ are noted x_i for $i = 1, \ldots, n$. When describing iterative processes to solve problem (1.1), we index vectors and matrices by the iteration number. For instance, at iteration k, (x_k, λ_k) is the current primal-dual iterate. For functions evaluated at x_k, we use the shorthand notation $r_k = r(x_k)$, $c_k = c(x_k)$, $J_k = J(x_k)$ etc. To not interfere with previous notation for components, we will add parentheses to make the distinction clear. For instance, $(x_k)_i$ denotes the i-th component of vector x_k.

The rest of the paper is organised as follows. In section 2, we present our algorithmic framework and discuss implementation details. We study global convergence in section 4. Numerical experiments are shown in section 5 and we end up by a conclusion and perspectives.

2 About the algorithm

We formally state the first assumptions about problem (1.1).

Assumption 1. *Residuals r and constraints c functions c are twice continuously differentiable on $\mathbb{R}^n$.*

Assumption 2. *The equality constraint matrix A is full line rank.*

Assumption 3. *The feasible region for the linear constraints Ω is non-empty.*

Our method is based on the Augmented Lagrangian (AL) function defined as

$$\Phi(x, \lambda, \mu) := \frac{1}{2}\|r(x)\|^2 + \langle \lambda, c(x)\rangle + \frac{\mu}{2}\|c(x)\|^2, \tag{2.1}$$

where $\mu > 0$ is a penalty parameter.

We maintain the linear constraints in the formulation of the problem and only penalize the violation of the nonlinear constraints. One has the following expression for the gradient:

$$\nabla_x \Phi(x, \lambda, \mu) = J(x)^T r(x) + C(x)^T \bar{\lambda}(x, \lambda, \mu), \tag{2.2}$$

with $\bar{\lambda}(x, \lambda, \mu) := \lambda + \mu c(x)$ generally referred are the first-order estimates of the Lagrange multipliers.

The Hessian is given by

$$\nabla^2_{xx} \Phi(x, \lambda, \mu) = J(x)^T J(x) + \mu C(x)^T C(x) + S(x) \tag{2.3}$$

with the second-order terms explicitly given by

$$S(x) = \sum_{i=1}^{n_r} r_i(x) \nabla^2 r_i(x) + \sum_{i=1}^{n_c} \nabla^2 c_i(x) \bar{\lambda}_i(x, \lambda, \mu) \tag{2.4}$$

For fixed λ and μ, reformulating problem (1.1) with function (2.1) gives the linearly constrained problem

$$\begin{aligned} \min_x \quad & \Phi(x, \lambda, \mu) \\ \text{s.t.} \quad & x \in \Omega. \end{aligned} \tag{2.5}$$

Moving the nonlinear constraints into the objective simplifies the problem because only linear constraints are left, which enables one to use iterative methods for linearly constrained optimization while still improving feasibility of the nonlinear constraints. A local minimum of (2.5) can be characterized the condition

$$x^* \in \operatorname*{argmin}_{x \in \Omega} \Phi(x, \lambda, \mu) \iff P_\Omega \left[x^* - \nabla_x \Phi(x^*, \lambda, \mu)\right] = x^*. \tag{2.6}$$

Another pro of AL methods is that they come naturally with the update formula for the multipliers:

$$\lambda_{k+1} = \bar{\lambda}(x_k, \lambda_k, \mu_k). \tag{2.7}$$

We follow the standard scheme of AL methods, which consist in approximately solving a sequence of linearly constrained problems of the form (2.5) until convergence towards a first-order critical point. The Lagrange multipliers and penalty paramaters are adjusted depending on the progress in the feasibility of the nonlinear constraints. Our method is outlined in algorithm 1. For references on general AL methods, we refer the reader to the implementation of the software for nonlinear programming named LANCELOT [12] with contents based on [8, 10, 13, 14] and summarized in [16, Chapter 14]. See also [23, 36, 37] for early theoretical considerations of the method of multipliers and [2, 3, 17, 20] for more recent implementations of AL algorithms.

Algorithm 1 Augmented Lagrangian algorithm

Require: Initial starting point $x_0^s \in \Omega$, Lagrange multipliers estimate λ_0, penalty parameter μ_0

Penalty parameter increase factor $\tau > 1$, tolerance update constants $\omega, \eta, \kappa_\omega, \kappa_\eta, \beta_\omega, \beta_\eta$
Set initial tolerances $\omega_0 \leftarrow \omega \mu_0^{-\kappa_\omega}$ and $\eta_0 \leftarrow \eta \mu_0^{-\kappa_\eta}$
for k=0,1,2... **do**
    Starting from x_k^s, approximately solve $\min_{x \in \Omega} \Phi(x, \lambda_k, \mu_k)$ to find $x_k \in \Omega$ such that

$$\|P_\Omega \left[x_k - \nabla_x \Phi_k\right] - x_k\| \leq \omega_k \tag{2.8}$$

    if $\|c(x_k)\| \leq \eta_k$ **then**
        if $\|P_\Omega \left[x_k - \nabla_x \Phi_k\right] - x_k\| \leq \omega_*$ and $\|c(x_k)\| \leq \eta_*$ **then**
            stop and **return** approximate solution $(x_k), \lambda_k$
        end if
        Update the Lagrange multipliers $\lambda_{k+1} \leftarrow \bar{\lambda}(x_k, \lambda_k, \mu_k)$
        Set the next starting point $x_{k+1}^s \leftarrow x_k$
        Leave the penalty parameter unchanged $\mu_{k+1} \leftarrow \mu_k$
        Decrease the tolerances $\omega_{k+1} \leftarrow \omega_k \mu_{k+1}^{-\beta_\omega}$ and $\eta_{k+1} \leftarrow \eta_k \mu_{k+1}^{-\beta_\eta}$
    else
        Let the iterate unchanged $\left(x_{k+1}^s, \lambda_{K+1}\right) \leftarrow (x_k^s, \lambda_k)$
        Increase the penalty parameter $\mu_{k+1} \leftarrow \tau \mu_k$
        Update tolerances $\omega_{k+1} \leftarrow \omega \mu_{k+1}^{-\kappa_\omega}$ and $\eta_{k+1} \leftarrow \eta \mu_{k+1}^{-\kappa_\eta}$
    end if
end for

One first notices that algorithm 1 has a outer-inner iteration structure, in the sense that each iteration requires to approximately solve an optimization problem (line 4), which also involves an iterative process. The techniques used for the latter are detailed in section 3.

Tolerances in algorithm 1 are updated in such a way that both sequences ω_k and η_k tend to 0 when $k \to \infty$. The update scheme that we outline follows the rules described in [16, Chapter 14]. The parameters values used in our implementation are given in section 3.3.

We describe the convergence in terms of the norm of the projected gradient but this quantity is not easily computable in practice. Indeed, contrary to the case where there is only bounds or linear equality constraints, there is no direct formula for the projection on a polyhedral set of the form Ω. For our implementation, we will use the norm of the "reduced" gradient, which we define now.

For $x \in \Omega$, $I(x)$ denotes the set of all bound constraints satisfied as equalities (active) at x, i.e.:

$$i \in I(x) \implies x_i \in \{l_i, u_i\}. \tag{2.9}$$

We also introduce the notion of tangent space at a point x, , written $T(x)$ and defined as the set of directions d such that the same constraints are satisfied with equality at both x and $x + d$. In our case, this corresponds to the linear equality constraints and the active bounds. Formally:

$$T(x) := \{d \in \mathbb{R}^n \mid Ad = 0,\ d_i = 0 \text{ for all } i \in I(x)\}. \tag{2.10}$$

At a given point $(x, \lambda)$and parameter μ, the reduced gradient is defined as

$$P_{T(x)} \left[\nabla_x \Phi(x, \lambda, \mu)\right]. \tag{2.11}$$

Provided that the multipliers associated to the active bounds are of appropriate sign, the quantity $\|P_{T(x)} \left[\nabla_x \Phi(x, \lambda, \mu)\right]\|$ is an appropriate criticality measure used in optimization algorithms for linearly constrained optimization such as MINOS [33] and LSNNO [40][1]. Projections on $T(x)$ can be computed in an efficient manner. Indeed, assuming that $I(x) = \{i_1, \ldots, i_p\}$ with $p < n - m$ and denoting by $Z \in \mathbb{R}^{p \times n}$ the matrix whose row k is the row i_k of the $n \times n$ identity matrix, $T(x)$ is nothing than the null space of the block matrix

$$\tilde{A} = \begin{pmatrix} A \\ Z \end{pmatrix}.$$

Let $\tilde{N}$ be a matrix whose columns form an orthonormal basis of the null space of $\tilde{A}$. By construction and our full rank assumption 2, $\tilde{A}$ is also full rank. Then, for the projection of a vector v onto the null space of $\tilde{A}$ is given by $\tilde{P}v$ where

$$\tilde{P} = \tilde{N} \left(\tilde{N}^T \tilde{N}\right)^{-1} \tilde{N}^T,$$

There is no need to explicitly form the matrix $\tilde{N}$ because we can make use of the equivalent expression of the projection matrix $\tilde{P}$:

$$\tilde{P} = I - \tilde{A}^T \left(\tilde{A}\tilde{A}^T\right)^{-1} \tilde{A}. \tag{2.12}$$

[1] From Pierre: TODO: Trouver des exemples plus récents

Using (2.12), one can compute the projection of a vector v, i.e. $\tilde{P}v$, by first solving, for an auxiliary variable y, the linear system

$$\left(\tilde{A}\tilde{A}^T\right) y = \tilde{A}v^+, \tag{2.13}$$

and retrieve the projection by

$$v - \tilde{A}^T y. \tag{2.14}$$

This approach, referred as the *normal equations* approach [22], requires to solve the system (2.13). This is done by using the Cholesky decomposition of $\tilde{A}\tilde{A}^T$. The latter is well defined because being well defined the matrix $\tilde{A}\tilde{A}^T$ is symmetric by construction and positive definite by assumption 2. System (2.13) is reduced to two successive lower triangular systems. The factorization $\tilde{A}\tilde{A}^T = \tilde{L}\tilde{L}^T$, with $\tilde{L}$ lower triangular, does not need to be computed from scratch because the structure of $\tilde{A}$ implies a block structured Cholesky factor that involves the Cholesky decomposition of AA^T. The latter is constant throughout the algorithm, so the factor $\tilde{L}$ can be computed in a relatively efficient manner, even though it potentially has to be updated at every new minor iteration. We expect it to be the case when these computations take place in the first outer iterations of algorithm **??**, but that as we progress toward a first-order critical point, the set of active bounds tends to stabilize and remain the same during the inner iterations. Details on this computation can be found in the appendix A.

This discussion justifies why our implementation uses condition

$$\|P_{T_k}\left[\nabla_x \Phi_k\right]\| \leq \omega_k \tag{2.15}$$

in algorithm 1 instead of (2.8).

3 About the inner minimization process

In this section, we describe the inner minimization process used to produced the outer iterates of algorithm 1. Since the current multipliers estimates λ and the penalty parameter μ are fixed, the objective function for the inner minimization only depends on x so we use the shorthand notation $\varphi : x \mapsto \Phi(x, \lambda, \mu)$. Given a point $x_0 \in \Omega$, multipliers estimates λ, a penalty parameter μ, we aim to find an approximate solution of the problem

$$\begin{aligned} \min_x \quad & \varphi(x) \\ \text{s.t.} \quad & x \in \Omega, \end{aligned} \tag{3.1}$$

and such that

$$\|P_{T(x)}\left[\nabla\varphi(x)\right]\| \leq \omega, \tag{3.2}$$

for a tolerance $\omega > 0$. This task also involves an iterative process, for which the iterates will be indexed with subscript k but are distinct from the outer iterates of algorithm **??**. At each iteration k, we compute a step s_k and recursively update the iterate by $x_{k+1} = x_k + s_k$. The step is obtained after minimizing a model of the objective function φ around x_k. We consider the quadratic model:

$$m_k(x) = \varphi_k + \langle g_k, x - x_k\rangle + \frac{1}{2}\left\langle x - x_k, H_k(x - x_k)\right\rangle, \tag{3.3}$$

where $\varphi_k := \varphi(x_k)$, $g_k := \nabla\varphi_k$ and H_k is a symmetric approximation of the true Hessian $\nabla^2_{xx}\varphi_k$. Using the latter would impact negatively the performance of the algorithm because computing the second order terms (2.4) requires too much time and storage in practice. We discuss the aspects of the methods relative to the choice of approximation in subsection 3.2. Model (**??**) is appropriate to describe the algorithm in terms of the iterate. When discussing about subproblems, we prefer to emphasis on the step and would then use the following model of the primal reduction $\varphi(x_k + s) - \varphi_k$, given by

$$q_k(s) = \frac{1}{2}\langle s, H_k s\rangle + \langle g_k, s\rangle . \tag{3.4}$$

One has $q_k(x - x_k) = m_k(x) - m_k(x_k)$.

We seek to compute the step s_k as an approximate solution of the program

$$\begin{aligned} \min_s \quad & q_k(s) && \text{(3.5a)}\\ \text{s.t.} \quad & x_k + s \in \Omega && \text{(3.5b)}\\ & \|s\|_\infty \leq \Delta_k, && \text{(3.5c)} \end{aligned}$$

Vector s denotes the unknown of the subproblem whose solution s_k is the step and is used to compute the new iterate $x_{k+1} = x_k + s_k$.

The constraints $x_k + s \in \Omega$ are satisfied as long as the steps satisfy:

- $As = 0$ (provided that $Ax_0 = b$)
- $x_k - l \leq s \leq u - x_k$

In our method, we also incorporate a trust-region strategy to control and assert the quality of a step. Therefore, we add to each subproblem the constraint $\|s\|_\infty \leq \Delta_k$ with a radius $\Delta_k > 0$. This constraint reflects the domain on which we believe that the model well approximates the true function. Since the ℓ_∞ trust region constraint is equivalent to imposing $x_i \in [-\Delta_k, \Delta_k]$ for all i, we can rewrite this subproblem

$$\begin{aligned} \min_s \quad & q_k(s) && \text{(3.6a)}\\ \text{s.t.} \quad & As = 0 && \text{(3.6b)}\\ & l_k \leq s \leq u_k, && \text{(3.6c)} \end{aligned}$$

with $(l_k)_i = \max(-\Delta_k, l_i - (x_k)_i)$ and $(u_k)_i = \min(\Delta_k, u_i - (x_k)_i)$ for all $i = 1, \ldots, n$.

The success of an iteration and update the trust region is evaluated by the ratio

$$\rho_k = \frac{\varphi(x_k + s_k) - \varphi_k}{q_k(s_k)}. \tag{3.7}$$

We follow the standard step acceptance criteria [16], which require constants $\eta_1, \eta_2, \gamma_1, \gamma_2$ such that

$$0 < \eta_1 \leq \eta_2 < 1 \text{ and } 0 < \gamma_1 \leq \gamma_2 < 1. \tag{3.8}$$

If $\rho_k > \eta_1$, the step is accepted and the trust region is expanded. Otherwise, the step is rejected and the region reduced. A typical scheme to update the radius Δ_k would be to set

$$\Delta_{k+1} \in \begin{cases} [\Delta_k, \infty) & \text{if } \rho_k \geq \eta_2 \\ [\gamma_2\Delta_k, \Delta_k] & \text{if } \rho_k \in [\eta_1, \eta_2) \\ [\gamma_1\Delta_k, \gamma_2\Delta_k] & \text{if } \rho_k < \eta_1 \end{cases} \tag{3.9}$$

The procedure employed to solve subproblem (3.6) is described in algorithm 2 and the latter is analyzed in the rest of this section.

Algorithm 2 Trust region inner iteration algorithm

Require: initial point $x_0 \in \Omega$, radius $\Delta_0 > 0$, constants $\eta_1, \eta_2, \gamma_1, \gamma_2, \gamma_3$ satisfying conditions (3.8).

set $k \leftarrow 0$
repeat
  compute the model m_k
  compute a step s_k that sufficiently reduces the model
  compute the ratio ρ_k (3.7)
  if $\rho_k > \eta_1$ **then** set $x_{k+1} \leftarrow x_k + s_k$
  else set $x_{k+1} \leftarrow x_k$
  end if
  set Δ_{k+1} according to (3.9)
  increment $k \leftarrow k + 1$
until $\|P_{T_k}[g_k]\| \leq \omega$

3.1 Computation of the step

We consider the current feasible iterate x_k and its associate approximate quadratic model m_k (3.3). The step is computed in two phases. We first compute a Cauchy step s_k^C that ensures a decrease of the objective function sufficient to establish global convergence of the inner minimization algorithm. Next, we further minimize the objective function by exploring the subspace defined by the constraints active at the Cauchy point $x_k^C := x_k + s_k^C$. This approach is of common use in gradient projection techniques [34, Chapter 16] and there are different strategies to compute a Cauchy point, such as projected searches [28, 32]. Because of the structure of our constraints, we do not cannot directly project on the whole set Ω in practice but we can exploit prior knowledge on the active bounds. That is the reason why we compute our Cauchy step by finding the first local minimizer of the model along the projected gradient path

$$s_k(t) = P_\Omega[x_k - tg_k] - x_k \text{ for } t \geq 0. \tag{3.10}$$

Our procedure is an adaptation to the polyhedral case of algorithm SBMIN [9] used for the inner minimization phase of LANCELOT solver [12].

Assume we have found a Cauchy step s_k^C. In order to have an efficient algorithm, we want the total step to achieve a better reduction than the Cauchy step. To do so, we build the next iterate x_{k+1} after a finite sequence of M minor iterates $x_{k,1}, \ldots, x_{k,M+1}$. The sequence starts at the Cauchy point and ends at the next iterate, i.e. $x_k + s_k^C = x_{k,1}$ and $x_{k+1} = x_{k,M+1}$. This type of approach has shown to be effective for general optimization with bound constraints [28] and has also been incorporated into other AL algorithms for the inner loop minimization [3, 12].

Each minor iterate is defined after the previous one and is decomposed into

$$x_{k,j+1} = x_{k,j} + w_{k,j},$$

where $w_{k,j}$ is a descent direction for the quadratic model q_k. For each minor iterate, we require

$$x_{k,j} \in \Omega, \quad \|x_{k,j} - x_k\|_\infty \leq \Delta_k, \quad I(x_k^C) \subseteq I(x_{k,j}). \tag{3.11}$$

The first two conditions are merely that each minor iterate is feasible and the associated step lies within the trust region, while the last condition means that we can only add active bounds during this process. Note that is this context, a bound can become active with respect to the trust region and not only the original bounds on the variables. We also require the sufficient decrease between two successive minor iterates

$$m_k(x_{k,j+1}) \leq m_k(x_{k,j}), \qquad j = 1, \ldots, M. \tag{3.12}$$

After each minor iteration, the corresponding step is $s_{k,j} := x_{k,j} - x_k$.

The search direction $w_{k,j}$ is an approximate minimizer of the subproblem

$$\min_w \quad m_k(x_{k,j} + w) \tag{3.13a}$$

$$\text{s.t.} \quad Aw = 0 \tag{3.13b}$$

$$w_i = 0, \qquad i \in I(x_{k,j}). \tag{3.13c}$$

At a minor iteration, the free variables, indexed by $\mathcal{F}(x_{k,j})$, are implicitly subject to the bounds

$$\left(l_{(k,j)}\right)_i \leq w_i \leq \left(u_{(k,j)}\right)_i, \qquad i \in \mathcal{F}(x_{k,j}) \tag{3.14}$$

where $l^{(k,j)} := l_k - s_{k,j}$ and $u^{(k,j)} := u_k - s_{k,j}$.

We now describe how the minor subproblem (3.13) is solved. The idea is to apply the projected conjugate gradient [22] method with the previous minor iterate $x_{k,j}$ as a starting point. Three termination cases can occur. First, we can generate a direction such that a component in the current active set $\mathcal{I}(x_{k,j})$ violates one on the bounds (3.14). When this happens, we scale the direction so that the associate component lies at the bound and stop the CG iterations. The second case occurs when we generate a direction of negative curvature and is handled similarly as the firs one, i.e. we modify the direction so that it reaches the limit of the feasible set. The third case is the normal termination one and occurs when we find a local minimizer, with respect to a given tolerance. In all cases, we return the obtained search direction $w_{k,j}$, perform the projected line search to compute the point $x_{k,j+1}$ such that eqs. (3.11) and (3.12) are verified. The handling of the termination cases differ in the continuation of the procedure. The last two cases (negative curvature and local minimality), cause the stopping of our minor iterates mechanism. On the contrary, if the CG iterations stopped because a bound was hit, we go back to solving (3.13) using projected conjugate gradient method with $x_{k,j+1}$ as a starting point and $I(x_{k,j+1}) \supset I(x_{k,j})$ as a new set of fixed components.

Each iteration of the conjugate gradient method applied to (3.13) requires computing projections onto the null space of the constraints (3.13c)–(3.13c). We denote the associated projection operator, in this case a $n \times n$ matrix, by $\tilde{P}$. The specification of the projected conjugate gradient method applied to solving (3.13) is outlined in algorithm 3.

The projections occurring at lines 1 and 17, and the associated operators, are computed with the normal equations approach described in previous section.

An interesting feature would be to complement each search direction by a steplength computed by a projected search [28, 32]. This allows to add more than one constraint at each minor iteration to the active set. The downside is that is would also require to compute the projection of the gradient direction (3.10) on the feasible set Ω at every new trial value of the steplength. When Ω only contains bound constraints, the projection is trivial and cheap to compute. The linear equality case is more costly but can still be handled efficiently by a *normal equations* or

Algorithm 3 The projected conjugate gradient method applied to (3.13)

Require: Positive constant $\kappa_{cg} = 0.1$

Set $w \leftarrow 0,\ r \leftarrow H_k(x_{k,j} - x_k) + g_k,\ v \leftarrow \tilde{P}r,\ p \leftarrow -v$
Set tolerance $\varepsilon_{cg} \leftarrow \kappa_{cg}\|v\|$
repeat
  if $\langle p, H_k p\rangle \leq 0$ **then**
    Set $w^+ \leftarrow w + \gamma p$ where γ is the smallest factor such that $w_i^+ \in \{l_i^{(k,j)}, u_i^{(k,j)}\}$
    for $i \in \mathcal{F}(x_{k,j})$
    STOP
  end if
  Set $\alpha \leftarrow \langle r, v\rangle \,/\, \langle p, H_k p\rangle$
  Set $w^+ \leftarrow w + \alpha p$
  if w^+ violates a bound **then**
    $w^+ \leftarrow w + \gamma p$ where γ is the smallest factor such that $w_i^+ \in \{l_i^{(k,j)}, u_i^{(k,j)}\}$
    for $i \in \mathcal{F}(x_{k,j})$
    STOP
  end if
  Set $r^+ \leftarrow r + \alpha H_k p$
  Set $v^+ \leftarrow \tilde{P}r^+$
  if $\sqrt{\langle r^+, v^+\rangle} < \varepsilon_{cg}$ **then STOP**
  end if
  Set $\beta \leftarrow \langle r^+, v^+\rangle \,/\, \langle r, v\rangle$
  Set $p \leftarrow -v^+ + \beta p$
  Set $w \leftarrow w^+,\ r \leftarrow r^+,\ v \leftarrow v^+$
until $2(n - m - |I(x_{k,j})|)$ iterations have been done
return w^+

augmented system approach [22]. When Ω is of the form (1.2), the projection is less trivial and requires to solve the associated minimum-distance quadratic program with a dedicated solver for quadratic programming, which we currently prefer to avoid.

We stop the minor-iterations procedure when the reduced gradient associated to the current active set is small enough. More formally, assume we performed the $j+1$ minor minimizations and let $T_{k,j}$ be the tangent space at $x_{x,j}$ with respect to the active bounds in $I(x_{k,j})$. We stop the minor iteration loop whenever the following inequality is satisfied:

$$\left\| P_{T_{k,j}} \left[\nabla m_k(x_{k,j+1}) \right] \right\| \leq \kappa_{mlt} \left\| P_{T_{k,j}} \left[\nabla m_k(x_k) \right] \right\|, \tag{3.15}$$

with $\kappa_{mlt} \in (0,1)$[2]. This inequality estimates if there is relative progress that can be made in the tangent subspace spanned by the free variables.

We have described our algorithm with respect to formulation (1.1) with equality constraints but we also accepts problems with nonlinear inequality constraints of the form $g(x) \geq 0$. We transform the latter into equality constraints by adding non-negative slack variables, which gives the new constraints

$$g(x) - \nu = 0, \quad \nu \geq 0. \tag{3.16}$$

. The lower and upper bounds associated to these slack variables are thus 0 and ∞ respectively. Of course, a similar treatment can be applied to transform potential linear inequality constraints into equalities. The AL function is now

$$\Phi(x, \nu, \lambda, \mu) = \varphi(x, \nu) = \frac{1}{2}\|r(x)\|^2 + \left\langle \lambda, \begin{pmatrix} c(x) \\ g(x) - \nu \end{pmatrix} \right\rangle + \frac{\mu}{2} \left\| \begin{matrix} c(x) \\ g(x) - \nu \end{matrix} \right\|^2$$

When slack variables are present, the step can be complemented by an additional *magical* step that is guaranteed to further reduce the AL function. This procedure, described in [15] and also used in [3], consists into a special update of the slack variables that exploits the structure of the AL. Let (x_k, ν_k) be the current iterate and $((\bar{s}_k)_x, (\bar{s}_k)_\nu)$ the step computed at an iteration of algorithm 2. We write the associated trial point

$$\begin{pmatrix} \bar{x}_k \\ \bar{\nu}_k \end{pmatrix} = \begin{pmatrix} x_k \\ u_k \end{pmatrix} + \begin{pmatrix} (\bar{s}_k)_x \\ (\bar{s}_k)_\nu \end{pmatrix}.$$

Since $\nu \mapsto \Phi(x, \nu, \lambda, \mu)$ is quadratic and convex, we can further minimize Φ by solving

$$\begin{aligned} \min_{\nu} \quad & \varphi(\bar{x}_k, \nu) \\ \text{s.t.} \quad & \nu \geq 0. \end{aligned} \tag{3.17}$$

The solution, noted $\hat{\nu}_k$, is explicit with components given by

$$(\hat{\nu}_k)_i = \max\left(0, \frac{\lambda_i}{\mu} + g_i(\bar{x}_k)\right). \tag{3.18}$$

The trial step for the current iteration is thus

$$s_k = \begin{pmatrix} (s_k)_x \\ (s_k)_\nu \end{pmatrix} = \begin{pmatrix} (\bar{s}_k)_x \\ \hat{\nu}_k - \bar{\nu}_k. \end{pmatrix} \tag{3.19}$$

[2]Minor Loop Tolerance

Using the latter step is tantamount to directly setting the slack variables values to (3.18), as we do not change the step relative to the x variables. This also implies that it does not require additional functions evaluations, since one only needs $r(\bar{x}_k)$ and $g(\bar{x}_k)$ to compute the actual reduction $\varphi\left(x_k + (s_k)_x, \nu_k + (s_k)_\nu\right)$. However, the predicted reduction needs to be changed to reflect the effect of the complementary step. Since we did not use the model to compute the second step, it is not relevant to evaluate the model reduction at s_k. A suitable choice is to define the new predicted reduction as the sum of the predicted reduction obtained by the first step with the actual reduction obtained by the second step, i.e.

$$q_k(s_k) + \varphi\left(x_k + (s_k)_x, \nu_k + (s_k)_\nu\right) - \varphi\left(x_k + (\bar{s}_k)_x, \nu_k + (\bar{s}_k)_\nu\right). \tag{3.20}$$

During our numerical experiments, we have observed significant robustness improvements of the algorithm on problems with inequality constraints.

3.2 About the Hessian approximation

We now discuss how the Hessian of the AL is iteratively updated when we define the quadratic model of iteration k in algorithm 2. To set up the context and notations of this paragraph, we wish to approximate the true Hessian

$$\nabla^2_{xx}\varphi(x_k) = J_k^T J_k + \mu C_k^T C_k + S_k,$$

by a symmetric matrix H_k. We remind that the need for an approximation mainly comes form the fact that evaluating the second order terms in S_k requires too much time and storage to be done in practice, especially for problems with a large number of variables and residuals.

The first approximation we can think of, and the simplest one, is obtained after merely linearizing the residuals and constraints in expression (2.1), which gives

$$H_k = J_k^T J_k + \mu C_k^T C_k. \tag{3.21}$$

We will call it the Gauss-Newton (GN) approximation, in reference to its counterpart in the unconstrained case. On the one hand, this approximation is very convenient and cheap as it involves already available first-order derivatives, since they are required to evaluate the gradient. Also, when the Jacobians are full rank, the resulting matrix is positive-definite which guarantees the convexity of subproblems of the form (3.1). On the other hand, it is known to be less efficient on problems with non zero residuals at the solution. This downside is amplified when we are to approximate the Hessian of the Lagrangian, or the AL in our case. Indeed, setting S_k to the zero matrix not only neglects the contribution of the residuals to the curvature of the model, but is also neglects the contribution of the Lagrange multipliers, for which there are no reasons to equal zero.

We thus need to take into account the full Hessian to form an approximation. Looking at the literature on this subject, one can observe that there is a variety of approaches focusing on the unconstrained case, as it is a major challenge in improving the efficiency of algorithms for problems with non zero residuals at the solution. Nevertheless, relevant parallels can be drawn with the AL situation, or the constrained case in general, since these studies explore techniques to deal with second order terms. Because the first-order terms are readily available and provide curvature information, only the second-order components need to be approximated. Therefore, we look for an approximation of the form

$$H_k = B_k^{GN} + B_k. \tag{3.22}$$

where B_k^{GN} is the right hand side of (3.21) and B_k is a symmetric approximation of S_k. The simplest choice, apart from setting $S_k = 0$, is to approximate S_k by a scalar multiple of the identity matrix $\sigma_k I$ where σ_k is an iteration dependent regularization parameter. This approach, known as the Levenberg-Marquardt [26, 31] (LM) method, is more robust than the GN method and performs well in practice, although on paper, it can be slowly convergent on large residuals problems. This technique have also been used in algorithms for nonlinear least-squares with equality constraints. Indeed, the authors in [6] use it to approximate the Hessian of the Lagrangian, whereas in [35], the authors add primal and dual regularization terms to the objective function in order to derive a regularized KKT system of equations.

The third class of methods, and the one we will use, consists to start with an initial approximation S_0 and update it iteratively after by a formula derived from a secant equation. Standard quasi-Newton methods employ DFP, BFGS or SR1 formulas [34, Chapter 6]. Since this exploits the structure of the Hessian, we will talk about structured quasi-Newton (SQN) methods. If B_k is the current approximation of S_k, in order to compute the approximation B_{k+1} of S_{k+1}, we require that a secant equation of the form

$$B_{k+1}s_k = y_k, \tag{3.23}$$

is satisfied. In (3.23), s_k is the iteration step and the right hand side y_k is defined after available quantities relative to the current iteration. Then, as for standard quasi-Newton methods, one can impose B_{k+1} to be close to B_k for a chosen matrix norm, which yields the DFP and BFGS formulas, or require a rank-one equation to be satisfied, for the SR1 formula. All these approaches, with different secant equations, have been studied and specialized to least-squares problems, with additional features to deal with sizing strategies [4, 7, 18, 24, 25, 29]. We also mention the SQN methods proposed in [41, 42], and the references therein, where the matrix S_k is approximated in factorized form by imposing a secant equation on a lower triangular matrix L_k such that $S_k \approx L_k^T L_k$.

Structured approximations of the AL Hessian have been studied for general objective functions [38] but our approach is closer to the one employed in [27]. In the latter, the authors base their approximation on a secant equation derived from a heuristic initially introduced in the unconstrained case [4] that they adapted to approximate the Hessian of the Lagrangian and use it in a SQP algorithm for constrained nonlinear least-squares. This strategy is also at the heart of the SQN method from the popular package for unconstrained nonlinear least-squares NL2SOL [18, 25]. The idea is the following. If we were to approximate each matrix term of the sum (2.4), we would have

$$B_{k+1} = \sum_{i=1}^{n_r} r_i(x_{k+1})B_{k+1}^{r_i} + \sum_{i=1}^{n_c} \bar{\lambda}_i(x_{k+1}, \lambda, \mu)B_{k+1}^{c_i}, \tag{3.24}$$

with $B_{k+1}^{r_i} \approx \nabla^2 r_i(x_{k+1})$ and $B_{k+1}^{c_i} \approx \nabla^2 c_i(x_{k+1})$. To get an accurate approximation, given a step s_k, it is reasonable to require

$$B_{k+1}^{r_i}s_k = \nabla r_i(x_{k+1}) - \nabla r_i(x_k) \qquad \text{and} \qquad B_{k+1}^{c_i}s_k = \nabla c_i(x_{k+1}) - \nabla c_i(x_k), \tag{3.25}$$

i.e. that each Hessian maps the change in the variables to the change in the gradients. Noticing that, for each i, $\nabla r_i(x_{k+1}) - \nabla r_i(x_k)$, resp. $\nabla c_i(c_{k+1}) - \nabla c_i(x_k)$, is the i-th row of $(J_{k+1} - J_k)^T$, resp. $(C_{k+1} - C_k)^T$, summing over the residuals and constraints indices all the terms of (3.25) gives, by (3.24), the structured secant equation

$$B_{k+1}s_k = (J_{k+1} - J_k)^T r_{k+1} + (C_{k+1} - C_k)^T \bar{\lambda}_{k+1}. \tag{3.26}$$

To follow the conventional notations of quasi-Newton methods, we will note $y_k := g_{k+1} - g_k$ and denote the right hand side of (3.23) by y_k^A to insist on its link with the AL formulation.

From (3.26), we can derive the SR1 update formula

$$B_{k+1} = B_k + \frac{(y_k^A - B_k s_k)(y_k^A - B_k s_k)^T}{(y_k^A s_k - B_k s_k)^T s_k}, \tag{3.27}$$

if $(y_k^A - S_k s_k)^T s_k \neq 0$. When the denominator in (3.27) is zero, we simply set $B_{k+1} = B_k$. To avoid numerical errors, we apply the update when

$$\left|\left\langle y_k^A - S_k s_k, s_k \right\rangle\right| \geq \kappa_{sds} \|s_k\| \|y_k^A - S_k s_k\|, \tag{3.28}$$

for $\kappa_{sds} \in (0,1)$[3]. Inequality (3.28) is one of the most commons safeguards used in SR1 methods.

The choice of the SR1 method is motivated by its robustness in the least-squares context, as the exhaustive benchmarks in [29] show, and its tendency to better approximate the true Hessian with each iteration [11]. The approximation could thus be indefinite, which could be considered as a weakness compared to BFGS of DFP updates, that are guaranteed to be positive definite as long as it is the case for the initial approximation B_0. However, as it is highlighted in algorithm 3, the use of the conjugate gradients method in a trust region framework handles, and even exploits, the potential non-convexity of the subproblems. Moreover, this update does not require a curvature condition $\langle s_k, y_k \rangle > 0$ to be satisfied. In other works on the constrained case, SQN methods are used to approximate the projected Hessian of the Lagrangian [39], or of a merit function [30], in the null space of the active constraints. The BFGS update is then preferred because this matrix is, under standard assumptions, positive semidefinite, according to the second-order necessary conditions.

We end our description of the Hessian approximation by discussing about hybrid updates, a feature that we do not implement but that is commonly used in SQN methods for nonlinear least-squares algorithms, either in the unconstrained [1, 18, 19] or constrained [27, 39] case. The idea is to test at every iteration if the problem has small or large residuals at the solution. In the latter case, a structured update is used to increase the accuracy of the approximated Hessian whereas in the former, the GN approximation is employed. Note that the update, such as (3.27), is still computed at every iteration to continue accumulate curvature information. We have tested a strategy inspired from [19] adapted to the constrained case, that computes the relative reduction

$$\zeta_k = \frac{\varphi(x_k) - \varphi(x_{k+1})}{\varphi(x_k)}, \tag{3.29}$$

and updates the approximated Hessian based on the rule

$$H_{k+1} = \begin{cases} B_{k+1}^{GN} & \text{if } \zeta_k \leq \kappa_{hyb} \\ B_{k+1}^{GN} + B_{k+1} & \text{otherwise,} \end{cases} \tag{3.30}$$

where κ_{hyb} is a parameter in $(0,1)$. Update (3.30) is used with $\kappa_{hyb} = 0.1$ in [39] where the AL plays the role of a merit function. We observed that the algorithm does less outer and inner iterations without using an hybrid switching rule. We suspect that a rule base on the ratio (3.29) is not suited for the AL Hessian, as there are second-order terms that can not legitimately be neglected even for small residuals problems and close to a feasible point.

[3] Small Denominator Safeguard

3.3 Summary and implementation details

We end our description of algorithm by summarizing the relations between outer, inner and minor iterates and the default values for the different constants exposed in this section.

- The outer iterates x_K are the main iterates of the algorithm and are approximate minimizers of the AL
- Each outer iterate is formed after a sequence of inner iterates $(x_k)_k$, linked by $x_{k+1} = x_k + s_k$ where s_k is an approximate solution of the QP (3.6)
- Each inner iterate is formed after a sequence of M minor iterates $x_{k,j}$ linked by $x_{k,j+1} = x_{k,j} + w_{k,j}$ where $w_{k,j}$ is a descent direction obtained by applying the projected conjugate gradient algorithm 3

Our intention with the work presented in this paper was to conceive an algorithm able to handle directly linear constraints of the form (1.2) but we also accept problems where linear constraints are only bounds and then

$$\Omega = \{x \in \mathbb{R}^n \mid l \leq x \leq u\}.$$

The framework we have presented remains valid for this formulation. The only practical difference is in the computation of projections. In the bound constrained case, the projection of a vector x on the set $\mathcal{B}$ has components

$$\left(P_\Omega[x]\right)_i = \begin{cases} l_i & \text{if } x_i < l_i \\ x_i & \text{if } x_i \in [l_i, u_i] \\ u_i & \text{if } x_i > u_i \end{cases} \tag{3.31}$$

One could argue that, since computing (3.31) is cheaper than solving the normal equations (2.13), we could make use of projected searches [32] as it is done in the solver TRON [28]. Indeed, this method have interesting convergence properties and could improve the efficiency of the Cauchy step computation or the steplength. We preferred, however, not to include it, because with a feasible region of the form (1.2), projected searches require to compute projections very often, which could negatively impact the performance. Nevertheless, those are likely to be investigated for future updates of the bounds constrained version. The other practical difference with the polyhedral case is that we can use directly the norm of the projected gradient

$$\|x_K - P_\Omega[x_K - \Phi_K]\|,$$

as a criticality measure.

The default values for tolerances and constants of algorithm 1 are

$$\mu_0 = 10,\ \kappa_\omega = \beta_\omega = \eta = \omega = 1,\ \kappa_\eta = 0.1,\ \beta_\eta = 0.9,\ \tau = 100,\ \omega^* = \eta^* = 10^{-7}.$$

The constants relative to the inner minimization process of algorithm 2 are set to

$$\kappa_{cg} = \kappa_{mlt} = 0.1.$$

We now discuss aspects relative to the trust region handling. The initial radius value for algorithm 2 is set to

$$\Delta_0 = 0.1\|g_0\|_\infty. \tag{3.32}$$

The mechanism to update the trust region given in (3.9) still leaves important flexibility. We choose to follow a standard strategy similar to the one exposed in [16, Chapter 17]. We also added a refinement to handle the case where the ratio ρ_k is negative, which can happen when there is very poor agreement between the function and the model. In such cases, we reduce more severely the radius by taking

$$\Delta_{k+1} = \gamma_1 \Delta_k.$$

The complete update rule for the trust region radius is then

$$\Delta_{k+1} = \begin{cases} \max\left(\alpha_2 \|s_k\|_\infty, \Delta_k\right) & \text{if } \rho_k \geq \eta_2 \\ \Delta_k & \text{if } \rho_k \in [\eta_1, \eta_2) \\ \alpha_1 \Delta_k & \text{if } \rho_k \in [0, \eta_1) \\ \min\left(\alpha_1 \|s_k\|_\infty, \gamma_1 \Delta_k\right) & \text{if } \rho_k < 0, \end{cases} \tag{3.33}$$

with constant values

$$\alpha_1 = 0.25,\ \alpha_2 = 2.5,\ \eta_1 = 0.25,\ \eta_2 = 0.75,\ \gamma_1 = 0.0625.$$

We have also implemented two safeguards to prevent the inner minimization algorithm from stalling. This can occur when the trust region radius is so small that no relative progress can be made, or when two consecutive iterates are indistinguishable from each, up to a relative tolerance. For the former, we stop algorithm 2 when

$$\Delta_k \leq \epsilon_{rad} \|x\|_\infty, \tag{3.34}$$

for some $\epsilon_{rad} > 0$. For the second condition, the algorithm is stopped when

$$|(s_k)_i| < \epsilon_{step} \left|(x_k)_i\right|, \tag{3.35}$$

for all $i = 1, \ldots, n$ such that $(x_k)_i \neq 0$. In our implementation, we have set

$$\epsilon_{rad} = \epsilon_{step} = \sqrt{\epsilon_{dbl}},$$

where ϵ_{dbl} is the relative double machine precision. During our numerical experiments, we have tested several values, but these ones best reflected the numerical intuition that a radius or step were "too" small in norm.

4 Convergence analysis

In this section, we study the convergence of algorithm TRAULLS[4], towards a first-order critical point of problem (1.1). We start by showing global convergence of algorithm 2, implying that the inner minimization phase of algorithm 1 is always well defined. We then establish global convergence of the main algorithm.

[4]Trust Region AUgmented nonLinear Least-squares Solver

4.1 Global convergence of the inner minimization

Because of the similarities between our algorithms, the proof given in this paper follows the structure of the one outlined in [8]. Most of the work consists into formulating and adapting intermediate results to the polyhedral case. Since we are interested into establishing theoretical convergence towards first-order critical points, we will use the norm of the projected gradient as a criticality measure.

We first make the standard assumption

Assumption 4. *The set $\mathcal{X} = \{x \mid \varphi(x) \leq \varphi(x_0)\} \cap \Omega$ is non empty and compact.*

By assumption 1, it is implicit that the function φ is twice continuously differentiable on $\mathcal{X}$ so we will not state this in the propositions given in this section.

We remind that the quadratic model is of the form

$$q_k(s) = \frac{1}{2} \langle s, H_k s \rangle + \langle g_k, s \rangle ,$$

where $g_k = \nabla\varphi(x_k)$ and H_k is an approximation of the Hessian based on the structured SR1 formula from 3.2. We formulate two assumptions relative to those approximations. The norm used is the induced norm on matrices, i.e. $\|M\| := \sup_{\|x\|=1} \|Mx\|$ for a given matrix M.

Assumption 5. *Defining, for every iteration index k, the scalars b_k by*

$$b_k = 1 + \max_{0 \leq i \leq k} \|H_i\| ,$$

we require that the series $\sum_k \frac{1}{b_k}$ diverges to ∞.

The second assumption and states that the norm of the approximating Hessians should not increase too fast compared with the speed of convergence of the function values.

Assumption 6.

$$\lim_{k \to \infty} b_k(\varphi_{k+1} - \varphi_k) = 0.$$

The projected gradient path is defined as

$$s_k(t) = P_\Omega \left[x_k - t g_k\right] - x_k \text{ for } t \geq 0.$$

The reduction of the model along the projected gradient path may thus be defined as the piecewise quadratic function

$$\psi(t) = q_k(s_k(t)),$$

and we denote by t_k^C the first local minimum of ψ subject to the trust region constraint

$$\|s_k(t)\|_\infty \leq \Delta_k. \tag{4.1}$$

The associated Cauchy step is

$$s_k^C = s_k(t_k^C). \tag{4.2}$$

Because of the choice of the ℓ_∞ norm, computing $s_k(t)$ within the trust region corresponds to project the direction $x_k - t g_k$ onto

$$\{d \in \mathbb{R}^n \mid Ad = 0,\ l_k \leq d \leq u_k\} , \tag{4.3}$$

with $(l_k)_i = \max(-\Delta_k, l_i - (x_k)_i)$ and $(u_k)_i = \min(\Delta_k, u_i - (x_k)_i)$ for all $i = 1, \ldots, n$.

We assume that the total step s_k produces a fraction of the reduction achieved by the Cauchy point:

$$q_k(s_k) \leq \kappa_{fcd}\ q_k(s_k^C), \tag{4.4}$$

for $\kappa_{fcd} \in (0, 1]$[5].

We detail the behavior of the polygonal line $s_k(t)$. Let $I(t)$ be the set of bounds l_k or u_k satisfied with equality at $s_k(t)$. Notice that this definition slightly differs from the active set $I(x)$ defined earlier, as the new formulation know takes into account the trust region. At first, if no bounds are active at x_k, projecting on the set (4.3) for t close to 0 is equivalent to projecting onto the null space of A. As t increases, the direction might hit several bounds. Since the set of active bounds can only be increased, we have that

$$I(t) \subseteq I(t') \quad \text{for all } 0 < t \leq t'. \tag{4.5}$$

Let

$$0 = t_0 < t_1 < \ldots < t_p,$$

be the successive values of t at which the projected step $s_k(t)$ hits a bound, also called breakpoints. Hitting a bound not only affects the corresponding components, that are now fixed, but it also affects the other components, as the steepest direction must now be projected on a subspace of the form

$$\{d \in \mathbb{R}^n \mid Ad = 0,\ d_i = 0 \text{ for some } i\}$$

This motivates to adapt the notion of tangent space at a point on the projected gradient path:

$$T(t) := \{d \in \mathbb{R}^n \mid Ad = 0,\ d_i = 0 \text{ for all } i \in I(t)\}, \tag{4.6}$$

for $t \geq 0$. This enables us to give a recursive expression of $s_k(t)$ on each interval $[t_i, t_{i+1})$, with $0 \leq i \leq p$, as:

$$s_k(t) = (t - t_i) P_{T(t_i)}[-g_k] + s_k(t_i). \tag{4.7}$$

We also define the reduced gradient on the projected gradient path:

$$z_k(t) := P_{T(t)}[g_k]. \tag{4.8}$$

Note that the reduced gradient, and hence the path $s_k(t)$ are well defined because of the full rank assumption 2 of the matrix A. As for the differentiability of φ, we will not remind it in the results of this section.

We now state a first lemma on how the tangent spaces and the reduced gradients at two different positions compare to each other.

Lemma 1. *For all $0 < t < t'$, we have that*

$$T(t) \supseteq T(t'), \tag{4.9}$$

and

$$\|z_k(t)\| \geq \|z_k(t')\|. \tag{4.10}$$

[5]Fraction Cauchy Decrease

Proof. The first statement follows from the inclusion (4.5). Inequality (4.10) then results from the fact that projecting the same vector on a smaller linear subspace reduces the norm of its projection. □

When referring to the tangent space and the reducted gradient at a given breakpoint t_i, we will make use of the shorthand notation $T_i := T(x_k + s_k(t_i))$ and $z_i := z_k(t_i)$. Notice that because the active set is fixed on each interval $[t_i, t_{i+1})$, so is the tangent space and hence, the reduced gradient is constant. Also, we can deduce from lemma 1 that the tangent spaces associated to each breakpoint form a finite sequence of nested linear subspaces

$$T_0 \supseteq T_1 \supseteq \ldots \supseteq T_p.$$

We can now expand equation (4.7):

$$s_k(t) = -(t - t_i)z_i - \sum_{j=0}^{i-1}(t_{j+1} - t_j)z_j, \tag{4.11}$$

which shows than on each interval (t_i, t_{i+1}), $s_k(t)$ is differentiable w.r.t. t and that

$$s_k'(t) = -z_k(t) = -z_i, \tag{4.12}$$

for $t \in (t_i, t_{i+1})$.

Our proof follows the structure of proof of global convergence given in [8] because of the similarity between our algorithm and the one described in [9], used in the inner minimization phase of LANCELOT [12] for the bound constrained case. Most of the work consists into giving an adapted formulation for the intermediate results involving the structure of the linear constraints. We start by establishing an inequality on the decrease of the objective function after taking step s_k. This will involve the norm of the projected gradient, i.e.:

$$h_k := \|s_k(1)\|. \tag{4.13}$$

Lemma 2. *If assumption 4 holds and that $h_k > 0$, then*

$$\|z_k(t_k^{(1)})\| \geq \frac{h_k}{2},$$

where κ_{ubg} is the constant defined by

$$\kappa_{ubg} := \max\left(1, \max_{x \in \mathcal{X}} \|\nabla\varphi(x)\|\right),$$

and $t_k^{(1)} = \frac{h_k}{2\kappa_{ubg}}$.

Proof. First note that the constant κ_{ubg} is well defined because φ is continuously differentiable on $\mathcal{X}$, the latter being compact by assumption 4.

By non-expansivity of the projection mapping onto the convex set Ω, one has, for any $t \geq 0$:

$$\begin{aligned} \|s_k(t)\| &= \|P_\Omega[x_k - tg_k] - x_k\| \\ &= \|P_\Omega[x_k - tg_k] - P_\Omega[x_k]\| \\ &\leq \|x_k - tg_k - x_k\| \\ &\leq t\|g_k\|. \end{aligned} \tag{4.14}$$

Choosing $t = t_k^{(1)}$ and because $\|g_k\| \leq \kappa_{ubg}$ by definition of κ_{ubg}, we get

$$\|s_k(t_k^{(1)})\| \leq \frac{h_k}{2}. \tag{4.15}$$

Now, defined $t_k^{(2)}$ as the smallest $t \geq 0$ such that

$$\|s_k(t_k^{(2)})\| = h_k. \tag{4.16}$$

By (4.15), we have

$$0 < t_k^{(1)} < t_k^{(2)} \leq 1. \tag{4.17}$$

On each interval (t_i, t_{i+1}), the left, resp. right, derivative of $s_k(t)$ at t_i, resp. t_{i+1}, are well defined and equal z_i. We can thus rewrite (4.11) as

$$s_k(t) = \int_{t_i}^{t} s_k'(t)dt + \sum_{j=0}^{i-1} \int_{t_j}^{t_{j+1}} s_k'(t)dt, \tag{4.18}$$

which we simplify by

$$s_k(t) = -\int_0^t z_k(t)dt, \tag{4.19}$$

using (4.12) and keeping in mind that it is a sum of integrals defined on each segment $[t_i, t_{i+1}]$. Now let i_1, resp. i_2 be the breakpoint index such that $t_k^{(1)} \in [t_{i_1}, t_{i_1+1})$, resp. $t_k^{(2)} \in [t_{i_2}, t_{i_2+1})$. Then by (4.19):

$$s_k(t_k^{(2)}) - s_k(t_k^{(1)}) = -\int_{t_k^{(1)}}^{t_k^{(2)}} z_k(t)dt, \tag{4.20}$$

which leads to

$$\begin{aligned} \|s_k(t_k^{(2)}) - s_k(t_k^{(1)})\| &\leq \int_{t_k^{(1)}}^{t_k^{(2)}} \|z_k(t)\|dt \\ &\leq \int_{t_k^{(1)}}^{t_k^{(2)}} \|z_k(t_k^{(1)})\|dt \\ &\leq (t_k^{(2)} - t_k^{(1)})\|z_k(t_k^{(1)})\|, \end{aligned} \tag{4.21}$$

where we have used (4.10) to bound the norm of the reduced gradient on $[t_k^{(1)}, t_k^{(2)}]$. Combined with (4.17) and (4.16), we get

$$\begin{aligned} \|z_k(t_k^{(1)})\| &\geq (t_k^{(2)} - t_k^{(1)})\|z_k(t_k^{(1)})\| \\ &\geq \|s_k(t_k^{(2)}) - s_k(t_k^{(1)})\| \\ &\geq \|s_k(t_k^{(2)})\| - \|s_k(t_k^{(1)})\| \\ &\geq \frac{h_k}{2}, \end{aligned} \tag{4.22}$$

which is the desired inequality. □

The next lemma gives an upper bound on the quadratic model $\psi(t)$ in an interval of interest.

Lemma 3. *If assumption 4 holds and that for some $t_k^{(3)} > 0$, one has*

$$\alpha_k = \|z_k(t_k^{(3)})\| > 0,$$

then, if T is the set of points in $[0, t_k^{(3)}]$ at which the piecewise quadratic ψ is differentiable,

$$\psi'(t) \leq -\alpha_k^2 + t_k^{(3)} \kappa_{ubg}^2 \|H_k\|, \tag{4.23}$$

for all $t \in T$.

Furthermore,

$$\psi'(t) \leq -\frac{1}{2}\alpha_k^2, \tag{4.24}$$

for all $t \in T \cap [0, t_k^{(4)}]$ and

$$\psi(t) \leq -\frac{\alpha_k^2}{2} t, \tag{4.25}$$

for all $t \in [0, t_k^{(4)}]$ where

$$t_k^{(4)} = \min\left(t_k^{(3)}, \frac{\alpha_k^2}{2\kappa_{ubg}^2(\|H_k\| + 1)} \right). \tag{4.26}$$

Proof. For $t \in T$, let i be the breakpoint index such that $t \in (t_i, t_{i+1})$. On the latter interval, ψ is differentiable and we have, by expression (4.11):

$$\begin{aligned} \psi'(t) &= \langle g_k, s_k'(t) \rangle + \langle s_k(t), H_k s_k'(t) \rangle \\ &= -\langle g_k, z_i \rangle - \langle s_k(t), H_k z_i \rangle . \end{aligned} \tag{4.27}$$

Because z_i is, by definition, the orthogonal projection of the vector g_k on a linear subspace:

$$\begin{aligned} \langle g_k, z_i \rangle &= \langle z_i, z_i \rangle \\ &\geq \|z_k(t_k^{(3)})\|^2 = \alpha_k^2, \end{aligned} \tag{4.28}$$

where the last inequality follows from the monotonicity of the reduced gradient norm on $[0, \infty)$.

We now look at the quadratic terms. First, by Cauchy-Schwarz inequality:

$$|\langle s_k(t), H_k z_i \rangle| \leq \|s_k(t)\| \|H_k z_i\|. \tag{4.29}$$

Then, applying the triangle inequality to expression (4.11), we obtain

$$\begin{aligned} \|s_k(t)\| &\leq (t - t_i)\|z_i\| + \sum_{j=0}^{i-1} (t_{j+1} - t_j)\|z_j\| \\ &\leq (t - t_i)\|g_k\| + \sum_{j=0}^{i-1} (t_{j+1} - t_j)\|g_k\| \\ &\leq t\|g_k\| \leq t_k^{(3)} \kappa_{ubg}, \end{aligned} \tag{4.30}$$

where we have used the facts that for all breakpoints indices j, $\|z_i\| \leq \|g_k\| \leq \kappa_{ubg}$ and that the scalar t is taken in T. For the remaining terms:

$$\|H_k z_i\| \leq \|H_k\| \|z_i\| \leq \kappa_{ubg} \|H_k\|. \tag{4.31}$$

Combining the latter inequality with (4.30), we get the following bound for the quadratic terms:

$$|\langle s_k(t), H_k z_i \rangle| \leq t_k^{(3)} \kappa_{ubg}^2 \|H_k\|. \tag{4.32}$$

Hence, using (4.28) and (4.32):

$$\psi'(t) \leq -\alpha_k^2 + t_k^{(3)} \kappa_{ubg}^2 \|H_k\|, \tag{4.33}$$

for all $t \in T$, which proves (4.23). Now, considering $t_k^{(4)}$ as defined by (4.26), since $t_k^{(4)} \leq t_k^{(3)}$, we get that $\|z_k(t_k^{(4)})\| \geq \alpha_k > 0$. The above reasoning based on $t_k^{(4)}$ thus yields

$$\psi'(t) \leq -\alpha_k^2 + t_k^{(4)} \kappa_{ubg}^2 \|H_k\| \leq -\frac{\alpha_k^2}{2}, \tag{4.34}$$

for $t \in T$, $t \leq t_k^{(4)}$. It follows that, for all $t \in [0, t_k^{(4)}]$:

$$\psi(t) \leq -\frac{\alpha_k^2}{2} t, \tag{4.35}$$

which completes the proof. □

The next lemma bounds the decrease guaranteed by the step.

Lemma 4. *If assumptions 4, 5 hold and that $h_k > 0$, then*

$$q_k(s_k) \leq -\kappa_{mdc} h_k^2 \min\left(\frac{h_k^2}{b_k}, \Delta_k\right), \tag{4.36}$$

where

$$\kappa_{mdc} = \frac{\kappa_{fcd}}{64 \kappa_{ubg}^2}. \tag{4.37}$$

Furthermore, if iteration k is successful, then

$$\varphi(x_k) - \varphi(x_k + s_k) \geq \kappa_{sdc} h_k^2 \min\left(\frac{h_k^2}{b_k}, \Delta_k\right), \tag{4.38}$$

with $\kappa_{sdc} = \eta_1 \kappa_{mdc}$.

Proof. We first observe that if we use $t_1^{(k)}$ as $t_k^{(3)}$ in the proof of lemma 3 and apply lemma 2, we get

$$q_k(t) \leq -\frac{h_k^2}{0} t, \tag{4.39}$$

for $t \in [0, t_k^{(5)}]$ with

$$t_k^{(5)} := \min\left(t_k^{(1)}, \frac{h_k^2}{8\kappa_{ubg}^2(\|H_k\| + 1)}\right) = \frac{h_k^2}{8\kappa_{ubg}^2(\|H_k\| + 1)}.$$

First, assume that $\|s_k(t_k^{(5)})\|_\infty \leq \Delta_k$. Then [6], by (4.4):

$$q_k(s_k) \leq -\kappa_{fcd} \frac{h_k^2}{8} t_k^{(5)}. \tag{4.40}$$

[6] TODO: might have to justify this "then"

Now, assume that $\|s_k(t_k^{(5)})\|_\infty > \Delta_k$. The latter implies that $\|s_k(t_k^C)\|_\infty = \Delta_k$ and from

$$\left\| s_k\left(\frac{\Delta_k}{\kappa_{ubg}}\right)\right\|_\infty \leq \left\| s_k\left(\frac{\Delta_k}{\kappa_{ubg}}\right)\right\| \leq \frac{\Delta_k}{\kappa_{ubg}}\|g_k\| \leq \Delta_k, \tag{4.41}$$

we can deduce that

$$t_k^C \geq \frac{\Delta_k}{\kappa_{ubg}}. \tag{4.42}$$

Therefore, using (4.39) with $t = t_k^C$ and (4.4) implies

$$q_k(s_k) \leq -\kappa_{fcd}\frac{h_k^2}{8\kappa_{ubg}}\Delta_k \leq -\kappa_{fcd}\frac{h_k^2}{64c_2^2}\Delta_k, \tag{4.43}$$

because $\kappa_{ubg} \geq 1$. The inequality (4.36) results from gathering (4.40), (4.43) and the definition of scalar b_k.

Finally, if the step is successful, we get inequality (4.38) by using (4.36) and the step acceptance condition $\rho_k > \eta_1$ from algorithm 2. □

Once we have established the guaranteed decrease inequality, the rest of the proof does not involve the polyhedral structure of the constraints and we can fall back into the standard convergence theory of trust region methods. We state the main convergence theorem, whose proof and intermediate developments can be found in [8].

Theorem 5 (Theorem 11 from [8])**.** *If assumptions 4, 5, 6 hold, then*

$$\lim_{k\to\infty} h_k = 0.$$

4.2 Global convergence of the algorithm

The content of this section follows the developments exposed in [14]. In this paper, the authors prove global convergence of AL methods for problems where the non-penalized constraints are linear inequalities, which fits to our context. Indeed, it is trivial that linear constraints of problem (1.1) can be written $\bar{A}x \geq \bar{b}$ with

$$\bar{A} = \begin{pmatrix} A \\ -A \\ I \\ -I \end{pmatrix} \text{ and } \bar{b} = \begin{pmatrix} b \\ -b \\ l \\ -u \end{pmatrix}.$$

Let $(x_k)_k$ be an infinite sequence of outer iterates generated by algorithm **??**. We make the following assumption.

Assumption 7. *The iterates lie within a closed bounded domain of* $\mathbb{R}^n$.

The last assumption mixes a constraint qualification of the nonlinear constraints and a condition for the simultaneous feasibility of both nonlinear and linear constraints. Before formulating it, we introduce some notations used throughout the rest of this section.

Let $\mathcal{K} \subseteq \mathbb{N}$ such that the sub-sequence $(x_k)_{k\in\mathcal{K}}$ converges to a limit point x^*. We define N_* as a matrix whose columns form a basis of $T(x^*)$.

Assumption 8. *The rank of matrix $C(x^*)N_*$ is no smaller than n_c at any limit point x^* of the sequence $(x_k)_k$.*

Assumption 8 ensures that the dimension of the null space of the active linear constraints is large enough to achieve feasibility of the nonlinear constraints and that their gradients are linearly independent within that space. With this assumption, the least-squares Lagrange multipliers estimates

$$\lambda^{LS}(x) = [C(x)N_*]^\dagger N_*^T \nabla f(x), \tag{4.44}$$

are well defined at $x = x^*$ with $[C(x)N_*]^\dagger$ denoting the pseudo-inverse [5] of $C(x)N_*$.

In the following theorem, written in the spirit of [14, Lemma 4.4 , Theorem 4.6] we state our global convergence result.

Theorem 6. *Assume that assumptions 1 and 3 hold. Let $(x_k)_{k\in\mathcal{K}}$ be an infinite sub-sequence of iterates produced by algorithm 1 converging to a limit point x^* for which assumptions 7 and 8 hold and let $\lambda^* = \lambda^{LS}(x^*)$ be its associate least-squares multipliers. Then $c(x^*) = 0$ and x^* is a first-order critical point of problem* (1.1) *with corresponding multipliers λ^*. Moreover, the sequence $\big(\bar{\lambda}(x_k, \lambda_k, \mu_k)\big)_{k\in\mathcal{K}}$ converges to λ^*.*

5 Numerical experiments

TODO

Appendix

A Block Cholesky factorization of the augmented matrix

In this appendix, we show how to construct the Cholesky decomposition of the matrix $\tilde{A}\tilde{A}^T$, where $\tilde{A}$ has been introduced at (**??**). The content of this appendix is adapted from the description given in [21] on how to compute a Cholesky decomposition for a general block matrix.

We remind that, considering $\{i_1, \dots, i_p\} \subseteq \{1, \dots, n\}$, with $p < n - m$, we have

$$\tilde{A} = \begin{bmatrix} A \\ Z \end{bmatrix} \in \mathbb{R}^{(m+p)\times n},$$

where $Z \in \mathbb{R}^{p\times n}$ the matrix whose row k is the row i_k of the $n \times n$ identity matrix. We assume that we already know a $m \times m$ lower triangular matrix L such that $AA^T = LL^T$. Following the reasoning in [21, section 4.2.9], we will show how, at the cost of extra computations, we can recover the Cholesky factor $\tilde{L}$ using the block structure of $\tilde{A}\tilde{A}^T$ and the factor L. The definition of $\tilde{A}$ naturally leads to the block pattern

$$\tilde{A}\tilde{A}^T = \begin{bmatrix} AA^T & AZ^T \\ ZA^T & ZZ^T. \end{bmatrix}$$

Simple computations first show that

$$ZZ^T = I_p, \quad AZ^T = [A_{i_1}, \dots, A_{i_p}].$$

The latter, written $A_{|\mathcal{A}}$, is the restriction of A to the columns corresponding to the indices in $\mathcal{A}$. Let $\tilde{L}$ be $(m + p) \times (m + p)$ triangular such that $\tilde{A}\tilde{A}^T = \tilde{L}\tilde{L}^T$. We apply the same block structure to $\tilde{L}$, i.e.

$$\tilde{L} = \begin{bmatrix} \tilde{L}_{11} & 0 \\ \tilde{L}_{21} & \tilde{L}_{22}, \end{bmatrix},$$

with

- $\tilde{L}_{11}$ $m \times m$ lower triangular
- $\tilde{L}_{21}$ $p \times m$
- $\tilde{L}_{22}$ $p \times p$ lower triangular

Verifying $\tilde{A}\tilde{A}^T = \tilde{L}\tilde{L}^T$ implies the matrix equality

$$\begin{bmatrix} AA^T & A_{|\mathcal{A}} \\ A_{|\mathcal{A}}^T & I_p. \end{bmatrix} = \begin{bmatrix} \tilde{L}_{11}\tilde{L}_{11}^T & \tilde{L}_{11}\tilde{L}_{21}^T \\ \tilde{L}_{21}\tilde{L}_{11}^T & \tilde{L}_{21}\tilde{L}_{21}^T + \tilde{L}_{22}\tilde{L}_{22}^T \end{bmatrix}. \tag{A.1}$$

By comparison of the blocks in (A.1), it follows that

$$\begin{aligned} AA^T &= \tilde{L}_{11}\tilde{L}_{11}^T \\ A_{|\mathcal{A}} &= \tilde{L}_{11}\tilde{L}_{21}^T \\ I_p &= \tilde{L}_{21}\tilde{L}_{21}^T + \tilde{L}_{22}\tilde{L}_{22}^T. \end{aligned}$$

Therefore, we can form $\tilde{L}$ by first setting $\tilde{L}_{11} = L$, then solve m lower triangular systems to compute $\tilde{L}_{21}$ and finally compute the Cholesky factor of $I_p - \tilde{L}_{21}\tilde{L}_{21}^T$ to get $\tilde{L}_{22}$.

B Cauchy point computation

We now describe a procedure to compute the Cauchy point as the first local minimizer of the quadratic model along the projected gradient path adapted from [8] to the case where linear equalities are present. For the seek of clarity, we omit the iteration index during the rest of this subsection. The piece-wise arc $s(t) = P_\Omega [x - g] - x$ satisfies the constraints

- $As(t) = 0$
- $s^{(l)} \leq s(t) \leq s^{(u)}$,

with $s_i^{(l)} = \max(-\Delta, l_i - x_i)$ and $s_i^{(u)} = \min(\Delta, u_i - x_i)$.

The following breakpoints:

$$0 = t_0 < t_1 < \ldots < t_p, \tag{B.1}$$

correspond to the successive scalars at which at least one component of $s(t)$ satisfies a bound with equality. Note that since A has m rows and is full rank, there are $n-m$ degrees of freedom remaining and thus at most $n - m$ breakpoints before the projected gradient path is constant. We recall the recursive expression

$$s(t) = -(t - t_i)z_i + s(t_i),$$

for $t \in [t_i, t_{i+1})$ and with the reduced gradient z_i given by (4.8).

To find the first local minimum of the scalar function $\psi(t) = q(s(t))$, we successively study each interval $[t_i, t_{i+1})$ to assert whether or not it contains a local minimizer. Assume we have not found a local minimizer on $[t_0, t_i)$ and thus look at the interval $[t_i, t_{i+1})$. The model along this arc can be written

$$\psi(t) = \frac{\psi_i''}{2}(\Delta t)^2 + \psi_i' \Delta t \tag{B.2}$$

with

- $\psi_i'' = \langle z_i, H z_i \rangle$
- $\psi_i' = -\langle s(t_i), H z_i \rangle - \langle g, z_i \rangle$
- $\Delta t = t - t_i$

Different cases can occur depending on the values of the slope ψ_i' and the curvature ψ_i''. Firstly, if

$$\begin{aligned} &\psi_i' > 0 \text{ or} \\ &\psi_i' = 0 \text{ and } \psi_i'' > 0, \end{aligned}$$

then t_i is the required minimizer. Next, if

$$\psi_i' < 0 \text{ and } \psi_i'' > 0,$$

the quadratic (B.2) has a strict minimizer at

$$t_i - \frac{\psi_i'}{\psi_i''}.$$

If the latter belongs to the interval of interest, i.e.

$$t_i - \psi_i'/\psi_i'' < t_{i+1},$$

then it is the required minimizer. In other cases, the minimizer is at or beyond t_{i+1}. To prepare for the study of the next interval, we first need to find the next breakpoint, given by the smallest scalar $t_{i+1} > t_i$ such that, at $s(t_{i+1})$, one of the free component hits one of its bounds. By introducing

$$\delta_i^- = \max_{s(t_i)_j<0} \left\{ s_j^{(l)}/s(t_i)_j \right\}, \qquad \delta_i^+ = \min_{s(t_i)_j>0} \left\{ s_j^{(u)}/s(t_i)_j \right\},$$

the next breakpoint is given by

$$t_{i+1} = t_i + \delta_i, \tag{B.3}$$

with $\delta_i = \min\left(\delta_i^-, \delta_i^+\right)$. We then add the corresponding variable index to the list of fixed components, compute the next reduced gradient z_{i+1} and finally update the slope and curvature of the model along the next interval. A last termination case can occur. Since A is of rank m, at most $n - m$ bounds can become active so if we never find a local minimum, the procedure still ends whenever it reaches the last breakpoint t_p. Indeed, past this breakpoint, the model along the projected gradient path is constant so we can return the last accumulated step $s(t_p)$ as the Cauchy step. Note that, in this case, it is also the total step, because no more variables can me modified. The presence of the trust region constraint ensures that all bounds become active. The procedure for the Cauchy step computation is outlined in algorithm 4.

Algorithm 4 Cauchy step computation

Require: Hessian H, gradient g bounds on the direction $s^{(l)},\ s^{(u)}$

Identify $I(0)$ and compute the direction z_0
Set **found** $\leftarrow$ **false**
Set $t_0 \leftarrow 0$, $s^{(0)} \leftarrow 0$ and initialize counter $i \leftarrow 0$
repeat
    $\psi_i' \leftarrow -\langle g, z_i \rangle\ ,\ \phi_i'' \leftarrow \langle z_i, Hz_i \rangle$
    Find the next breakpoint t_{i+1} by (B.3)
    if $\psi_i' > 0$ or $\psi_i' = 0$ and $\psi_i'' > 0$ **then**
        Set $s^C \leftarrow s^{(i)}$
        found $\leftarrow$ **true**
    else if $\psi_i' < 0$ and $\psi_i'' > 0$ and $t_i - \psi_i'/\psi_i'' < t_{i+1}$ **then**
        Set $\Delta t \leftarrow -\phi_i'/\phi_i''$
        Set$s^C \leftarrow s^{(i)} - \Delta t z_i$, **found** $\leftarrow$ **true**
    else
        Set $\Delta t \leftarrow t_{i+1} - t_i$
        Compute the next direction z_i
        Set $s^{(i+1)} \leftarrow s^{(i)} - \Delta t z_i$
    end if
    Increment $i \leftarrow i + 1$
    if $|I(t_i)| = n - m$ **then**
        Set $s^C \leftarrow s^{(i)}$, **found** $\leftarrow$ **true**
    end if
until found
return s^C

References

[1] M. Al-Baali and R. Fletcher. Variational methods for non-linear least-squares. *The Journal of the Operational Research Society*, 36(5):405–421, 1985. doi: 10.1057/jors.1985.68.

[2] 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(4):1286–1309, 2008. doi: 10.1137/060654797.

[3] S. Arreckx, A. Lambe, J.R.R.A. Martins, and D. Orban. A matrix-free augmented Lagrangian algorithm with application to large-scale structural design optimization. *Optimization and Engineering*, 17:359–384, 2016. doi: 10.1007/s11081-015-9287-9.

[4] M.C. Bartholomew-Biggs. The estimation of the hessian matrix in nonlinear least-squares problems with non-zero residuals. *Mathematical Programming*, 12:67–80, 1977. doi: 10.1007/BF01593770.

[5] A. Ben-Israel and T.N.E. Greville. *Generalized Inverses*. Springer New York, NY, USA, second edition, 2003. doi: 10.1007/b97366.

[6] E.H. Bergou, Y. Diouane, V. Kungurtsev, and C.W. Royer. A nonmonotone matrix-free algorithm for nonlinear equality-constrained least-squares problems. *SIAM Journal on Scientific Computing*, 43(5):S743–S766, 2021. doi: 10.1137/20M1349138.

[7] J.T. Betts. Solving the nonlinar least square problem: Application of a general method. *Journal of Optimization Theory and Applications*, 18:469–483, 1976. doi: 10.1007/BF00932656.

[8] A.R. Conn, N.I.M. Gould, and Ph.L. Toint. Global convergence of a class of trust region method algorithms for optimization with simple bounds. *SIAM Journal on Numerical Analysis*, 25(2):433–460, 1988. doi: 10.1137/0725029.

[9] A.R. Conn, N.I.M Gould, and Ph.L. Toint. Testing a class of methods for solving minimization problems with simple bounds on the variables. *Mathematics of Computation*, 50 (182):399–430, 1988. doi: 10.1090/S0025-5718-1988-0929544-3.

[10] A.R. Conn, N.I.M. Gould, and Ph.L. Toint. A globally convergent Augmented Lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Numerical Analysis*, 28(2):545–572, 1991. doi: 10.1137/0728030.

[11] A.R. Conn, N.I.M. Gould, and Ph.L. Toint. Convergence of quasi-Newton matrices generated by the symmetric rank one update. *Mathematical Programming*, 50:177–195, 1991. doi: 10.1007/BF01594934.

[12] A.R. Conn, N.I.M Gould, and Ph.L. Toint. *LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization (Release A)*. Springer Series in Computational Mathematics. Springer Berlin, Heidelberg, 1992. doi: 10.1007/978-3-662-12211-2.

[13] A.R. Conn, N. Gould A. Sartenaer, and Ph.L. Toint. Global convergence of a class of trust region algorithms for optimization using inexact projections on convex constraints. *SIAM Journal on Optimization*, 3(1):164–221, 1993. doi: 10.1137/0803009.

[14] A.R. Conn, N. Gould A. Sartenaer, and Ph.L. Toint. Convergence properties of an Augmented Lagrangian algorithm for optimization with a combination of general equality and linear constraints. *SIAM Journal on Optimization*, 6(3):674–703, 1996. doi: 10.1137/S1052623493251463.

[15] A.R. Conn, L.N. Vicente, and C. Visweswariah. Two-step algorithms for nonlinear optimization with structured applications. *SIAM Journal on Optimization*, 9(4):924–947, 1999. doi: 10.1137/S1052623498334396.

[16] A.R. Conn, N.I.M Gould, and Ph.L. Toint. *Trust Region Methods*. SIAM: Society of Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000. doi: 10.1137/1.9780898719857.

[17] F.E. Curtis, H. Jiang, and D.P. Robinson. An adaptative Augmented Lagrangian method for large-scale constrained optimization. *Mathematical Programming, Series A*, 152:201–245, 2015. doi: 10.1007/s10107-014-0784-y.

[18] J.E. Dennis Jr, D.M. Gay, and R.E. Walsh. An adaptive nonlinear least-squares algorithm. *ACM Transactions on Mathematical Software*, 7(3):348–368, 1981. doi: 10.1145/355958.355965.

[19] R. Fletcher and C. Xu. Hybrid methods for nonlinear least squares. *IMA Journal of Numerical Analysis*, 7:373–389, 1987. doi: 10.1093/imanum/7.3.371.

[20] P.E. Gill and D.P. Robinson. A primal-dual Augmented Lagrangian. *Computationnal Optimization and Applications*, 51:1–25, 2012. doi: 10.1007/s10589-010-9339-1.

[21] G.H. Golub and C.F. Van Loan. *Matrix Computations*. JHU Press, Maryland, MD, USA, 4th edition, 2013. doi: 10.56021/9781421407944.

[22] N.I.M. Gould, M.E. Hribar, and J. Nocedal. On the solution of equality constrained quadratic programming problems arising in optimization. *SIAM Journal on Scientific Computing*, 23(4):1376–1395, 2001. doi: 10.1137/S1064827598345667.

[23] M.R. Hestenes. Multipler and gradient methods. *Journal of Optimization Theory and Applications*, 4:303–320, 1969. doi: 10.1007/BF00927673.

[24] J. Huschens. On the use of product structure in secant methods for nonlinear least squares problems. *SIAM Journal on Optimization*, 4(1):108–129, 1994. doi: 10.1137/0804005.

[25] J.E.Dennis Jr, H.J. Martinez, and R.A. Tapia. Convergence theory for the structured BFGS method with an application to nonlinear least squares. *Journal of Optimization Theory and Applications*, 61(2):161–178, 1999. doi: 10.1007/BF00962795.

[26] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2:164–168, 1944.

[27] Z. Li, M. Osborne, and T. Prvan. Adaptative algorithm for constrained least-squares problems. *Journal of Optimization Theory and Applications*, 114:423–441, 2002. doi: 10.1023/A:1016043919978.

[28] C.-J. Lin and J.J Moré. Newton's method for large bound-constrained optimization problems. *SIAM Journal on Optimization*, 9(4):1100–1127, 1999. doi: 10.1137/S1052623498345075.

[29] L. Lukšan, C. Matonoha, and J. Vlček. Hybrid methods for nonlinear least squares problems. Technical Report 1246, Institute of Computer Science, Academy of Sciences of the Czech Republic, 2019.

[30] N. Mahdavi-Amiri and R.H. Bartels. Constrained nonlinear least squares: An exact penalty approach with projected structured quasi-Newton update. *ACM Transcations on Mathematical Software*, 15(3):220–242, 1989. doi: 10.1145/66888.66891.

[31] D.W. Marquardt. An algorithm for least squares estimation of non-linear parameters. *SIAM Journal*, 11:431–441, 1963.

[32] J.J. Moré and G. Toraldo. On the solution of large quadratic programming problems with bound constraints. *SIAM Journal on Optimization*, 1(1):93–113, 1991. doi: 10.1137/0801008.

[33] B.A. Murtagh and M.A. Saunders. Large-scale linearly constrained optimization. *Mathematical Programming*, 14:41–72, 1978. doi: 10.1007/BF01588950.

[34] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer series in Operation Reasearch and Financial Engineering. Springer, New York, NY, USA, 2nd edition, 2006. doi: 10.1007/978-0-387-40065-5.

[35] D. Orban and A.S. Siqueira. A regularization method for constrained nonlinear least squares. *Computational Optimization and Applications*, 76(3):961–989, 2020. doi: 10.1007/s10589-020-00201-2.

[36] M.J.D. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*, pages 283–298. Academic Press, London, 1969.

[37] R.T. Rockafellar. The multiplier method of Hestenes and Powell applied to convex programming. *Journal of Optimization Theory and Applications*, 12(6):555–562, 1973. doi: 10.1007/BF00934777.

[38] R. Tapia. On secant updates for use in general constrained optimization. *Mathematics of Compuations*, 51(183):181–202, 1988. doi: 10.2307/2008585.

[39] I.B. Tjoa and L.T. Biegler. Simultaneous solution and optimization strategies for parameter estimation of differential-algebraic equation systems. *Industrial & Engineering Chemistry Research*, 30(2):376–385, 1991. doi: 10.1021/ie00050a015.

[40] Ph.L. Toint and D. Tuyttens. LSNNO, a FORTRAN subroutine for solving large-scale nonlinear network optimization problems. *ACM Transactions on Mathematical Software*, 18(3):308–328, 1992. doi: 10.1145/131766.131771.

[41] H. Yabe and T. Takahashi. Factorized quasi-Newton methods for nonlinear least squares problems. *Mathematical Programming*, 51:75–100, 1991.

[42] W. Zhou and X. Chen. Global convergence of a new hybrid Gauss-Newton structured BFGS method for nonlinear least squares problems. *SIAM Journal on Optimization*, 20 (5):2422–2441, 2010. doi: 10.1137/090748470.