Ongoing work on MCWAL

Pierre Borie*

Abstract

This informal document deals with the ongoing work and thinking on a algorithm for constrained nonlinear least squares. The current algorithm name is MCWAL for Moindres Carrés With Augmented Lagrangian. Some sections about more general aspects about nonlinear programming might also appear here and there. The aim of this document is to reflect the thinking and understanding of different kinds of notions.

1 Introduction

We consider least squares problems subject to both nonlinear and linear constraints of the form

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \frac{1}{2}\|r(x)\|^2 \\ \text{s.t.} \quad & h(x) = 0 \\ & \langle c_i, x\rangle = b_i, \quad i = 1,\dots,m \\ & l \le x \le u, \end{aligned} \tag{1.1}$$

where $r\colon \mathbb{R}^n \to \mathbb{R}^d$ and $h\colon \mathbb{R}^n \to \mathbb{R}^t$ are assumed to be nonlinear, potentially non convex, continuously differentiable functions, $\langle\cdot,\cdot\rangle$ is the canonical inner product and $\|\cdot\|$ its induced euclidean norm, c_i are m independent vectors of $\mathbb{R}^n$, $(m \le n)$, $b = (b_1,\dots,b_m)^T \in \mathbb{R}^m$ and ℓ and u are vectors in $\mathbb{R}^n$. Without loss of generality, some components of the latter two vectors can be set to $\pm\infty$ for unbounded parameters. In the context of least squares problems, components r_i of the function r are often denoted as the residuals.

We will also refer to the linear constraints using the following set notation

$$\mathcal{X} = \{x \in \mathbb{R}^n \mid Cx = b,\ l \le x \le u\}, \tag{1.2}$$

where C is the matrix whose columns are the vectors c_i. By linear independence of those vectors, C is a full rank matrix. The set $\mathcal{X}$ is thus convex.

1.1 Notations

The Jacobian matrix of constraints function h is noted A.

When considering iterative methods for solving problem (1.1), k will, if not mentioned otherwise, refer to the iteration number. In order to simplify notations, quantities relative to a given iteration will be noted with the iteration number as an index, such as x_k for the iterate, r_k for $r(x_k)$,J_k for $J(x_k)$ etc.

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

Symbol := shall be used to state the definition of a numerical object (function, vector etc.)

For a set E, a sequence $(x_k)_k$ converging to $x^* \in E$ when $k \to +\infty$, we will write $x_k \to_E x^*$ if, above a certain index, all x_k are in E.

1.2 Optimality conditions for nonlinear programming

In this section, we describe optimality conditions for nonlinear programs more general than (1.1) from different views.

1.2.1 An algebraic view

We consider the general mathematical program[1]

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h(x) = 0 \\ & l \leq x \leq u, \end{aligned} \tag{1.3}$$

with the same assumptions on of differentiability of functions f and h. We introduce the Lagrangian associated to problem (1.3):

$$\ell(x, \lambda) = f(x) + \langle \lambda, h(x) \rangle \,, \tag{1.4}$$

where λ is the vector of Lagrange multipliers. Algorithms discussed aim to find local minimum of problem (1.3), i.e. feasible and of minimal value in s neighborhood. Under suitable assumptions, one can establish necessary and even sufficient conditions for local optimality. One of the most important assumptions belongs to the familty of constraints qualifications. The following is the one we will employ.

Definition 1. ***(LICQ)*** *The LICQ holds at x^* if the gradients of the equality constraints evaluated at x^* are linearly independent. In other terms, the matrix $A(x)$ is full rank.*

Using this and standard differentiability assumptions, one can state first-order necessary optimality conditions, also known as KKT conditions.

Definition 2. ***(KKT conditions)*** *A point x^* satisfies the KKT conditions if x^* is feasible and there exists multipliers λ^* such that*

$$\nabla_x \ell(x^*, \lambda^*) = 0.$$

A point satisfying those conditions if also said to be a KKT point or a first order critical point for problem (1.3). Depending on the context, we would refer to a KKT point either by just wrting x^* or the couple formed after x^* and its associated Lagrange multiplier λ^*The necessary conditions follow.

Theorem 3. ***(First Order Necessary Conditions)[7, Theorem 12.1]*** *Let x^* be a local solution of* (1.3) *at which the LICQ holds. Then x^* is a KKT point.*

Sufficient optimality conditions can be established using second order information.

[1] Shall I write the KKT conditions w.r.t this formulation and then consider multipliers for the bounds?

Definition 4. ***(Second Order Conditions)*** *A point (x^*, λ^*) satisfies the second order conditions if it is a KKT point for* (1.3) *at which the LICQ holds and if the matrix $\nabla^2_{xx}\ell(x^*, \lambda^*)$ is positive definite on the null space of the constraints Jacobian, i.e.:*

$$\forall w \text{ verifying } \langle A(x^*), w\rangle = 0,\ \left\langle w, \nabla^2_{xx}\ell(x^*, \lambda^*)w\right\rangle > 0.$$

Theorem 5. ***(Second Order Sufficient Conditions) [7, Theorem 12.5]*** *If x^* satisfies the second order conditions, then x^* is a local minimum of* (1.3).

1.2.2 A geometric view

We now consider the general program

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & x \in \mathcal{C}, \end{aligned} \tag{1.5}$$

where the feasible set $\mathcal{C}$ is a (of course non empty), subset of $\mathbb{R}^n$. The idea is to describe optimality conditions using inclusions of certain vectors to sets and cones. General results can be stated without any further assumptions on $\mathcal{C}$. Yet, we will give important results about projections that require to assume $\mathcal{C}$ is convex. As we shall see in section 2, our method will imply solving quadratic programs subject to linear, hence convex, constraints. Most of the content exposed is from [7, Chapter 12].

We start by defining two important sets, that are actually cones. We remind that a set K is a cone if for all $x \in K$ and $\alpha > 0$, $\alpha x \in K$. A cone is pointed if it contains 0.

Definition 6. ***Tangent Vector*** *Let $x \in \mathcal{C}$. A vector $v \in \mathbb{R}^n$ is said to be a tangent vector to $\mathcal{C}$ at x if there are sequences $(x_k)_k \to_{\mathcal{C}} x$ and $(t_k)_k \searrow 0^+$ such that*

$$\lim_{k\to\infty} \frac{x_k - x}{t_k} = v.$$

Definition 7. *The* ***tangent cone*** *is the set of all tangents vectors to $\mathcal{C}$ at x and is denoted by $T_{\mathcal{C}}(x)$.*

For a better understanding, the limit in the definition can be written as the relation

$$x_k - x = t_k v + o(1).$$

Intuitively, tangent vectors are all the directions going from x that stay in $\mathcal{C}$, up to a scaling by a very small positive factor. In the context of constrained optimization, they characterize directions in which one can step away from x while remaining feasible.

Remark

Rigorously, it is a $o(t_k)$ but since $t_k \to 0$, a $o(t_k)$ is also a $o(1)$. Our intent is to highlight the visualization of a tangent vector as a scaled $x_k - x$ for $(x_k)_k \to x$.

Definition 8. *The* ***normal cone*** *to $\mathcal{C}$ at x is the set defined by*

$$N_{\mathcal{C}}(x) := \{w \in \mathbb{R}^n \mid \forall v \in T_{\mathcal{C}}(x), \langle v, w\rangle \leq 0\}.$$

The normal cone is merely the orthogonal complement of the tangent cone and is involved in the following first order optimality condition.

Theorem 9. *If x^* is a local minimum of f, then*

$$-\nabla f(x^*) \in N_{\mathcal{C}}(x^*).$$

This theorem reflects the intuitive fact that a local minimum is a point from which the objective function cannot be reduced while remaining feasible. While this result is simple is elegant, its use is limited by the difficulty to express both tangent and normal cones without any other assumptions. For instance, with the LICQ, the tangent cone can be expressed as an intersection of hyperplanes depending on the constraints gradients and combining theorem 9 with a Farkas-like lemma gives the KKT conditions. Similar consequences occur when the feasible set is convex, which we will assume from now on.

We first remind the definition a convex set.

Definition 10. *A set $\mathcal{C} \subseteq \mathbb{R}^n$ is **convex** if $\forall (x, y) \in \mathcal{C}^2$ and $\alpha \in [0, 1]$, one has $\alpha x + (1-\alpha)y \in \mathcal{C}$. In other words, the line segment between two points remains in $\mathcal{C}$, for all points of $\mathcal{C}$.*

In this setting, the normal cone can be described more simply.

Proposition 11. *Let $\mathcal{C}$ a non empty convex set. Then for all $x \in \mathcal{C}$:*

$$N_{\mathcal{C}}(x) = \{v \mid \forall y \in \mathcal{C}, \langle v, x - y\rangle \geq 0\}.$$

For $x \notin \mathcal{C}$, $N_{\mathcal{C}}(x) = \emptyset$.

Then, if x^* is a local minima of (1.5), the inclusion in theorem 9 becomes

$$\forall x \in \mathcal{C}, \langle \nabla f(x^*), x - x^*\rangle \geq 0. \tag{1.6}$$

Remarks on (1.6)

1. This condition if sufficient when the function f is convex
2. It also reflects the very visual fact that in $\mathbb{R}^2$, the line tangent to a local minima is below the curve

So we now have a more simple way to characterize a minimum but it is still not suited to be used in algorithms. Obviously, one cannot check the sign of an infinite number of inner product but more fundamentally, algorithm must be able to manipulate points that are likely to be outside of the feasible set. Using projections is the key.

Definition 12. *Let $\mathcal{C} \subseteq \mathbb{R}^n$. The **projection operator** is the set-value mapping $P_{\mathcal{C}} \colon \mathbb{R}^n \rightrightarrows \mathcal{C}$ defined by*

$$P_C(\bar{x}) = \operatorname*{argmin}_{y \in \mathcal{C}} \frac{1}{2}\|x - y\|^2.$$

For this definition, we introduce the notion of set-value mapping [8] to remain general but in the convex case, it is single valued and well defined if the concerned set is closed.

Theorem 13. *Let $\mathcal{C}$ be a close convex set and $x \in \mathbb{R}^n$. Then*

$$\bar{x} = P_{\mathcal{C}}(x) \iff \forall y \in \mathcal{C}, \langle \bar{x} - x, y - \bar{x} \geq 0.\rangle$$

Since the norm is convex, this theorem is simply derived from the first order necessary optimality conditions. Also the inequality kind of looks like the inequality in proposition (11), which motivates the following theorem.

Theorem 14. *Let $\mathcal{C}$ be a close convex set, $x \in \mathbb{R}^n$ and $z \in \mathcal{C}$. Then*

$$z = P_{\mathcal{C}}(x) \iff (x - z) \in N_{\mathcal{C}}(z).$$

From this result, one deduces the following useful corollary.

Corollary 15. *Let $\mathcal{C}$ be a close convex set, $x \in \mathcal{C}$ and $z \in \mathbb{R}^n$. If $z \in N_{\mathcal{C}}(x)$, then $\forall t \geq 0$, $P_{\mathcal{C}}(x + tz) = x$.*

Applying the above corollary to opposite of the gradient of a local minimum, thus verifying theorem 9, gives the next theorem.

Theorem 16. *Consider the problem* (1.5) *where the non empty feasible set $\mathcal{C}$ is closed and convex. Then x^* is a local minimum if and only if for all $t \geq 0$, $P_{\mathcal{C}}\left(x^* - t\nabla f(x^*)\right) = x^*$.*

There is our desired result that we shall use as a termination criteria, up to a small tolerance. In practice, we will work with the Cauchy arc defined as

$$p(t, x) := P_{\mathcal{C}}(x - t\nabla f(x)), \tag{1.7}$$

that is the path alongside the steepest descent direction. The Cauchy point is the point minimizing the objective function on the Cauchy arc, i.e.:

$$x^C := x - t^C \nabla f(x) \text{ where } t^C = \operatorname*{argmin}_{t \geq 0} f\left(p(t, x)\right). \tag{1.8}$$

If an algorithm uses the Cauchy point as the new iterate, then the algorithm would converge. Moreover, if each iterate reduces the objective function as much as a fraction of the Cauchy point, the algorithm would also converge. This pretty much sets the foundation, and intuition, for the theory of convergence for algorithms producing inexact solutions of the sub problems, especially for trust regions methods [2]. In a less formal way, it translates into a theoretical notion the practical aspect that computers are by nature inexact, because of floating-point arithmetic, and thus we are more interested into proving that "not exact but good enough" solutions will do the job. An algorithm that does not converge if solutions are not exact might be nice on paper but will not be very useful in practice.

1.3 Generalities on least squares

Rewriting the objective function of problem (1.1) as $f\colon x \mapsto \frac{1}{2}\|r(x)\|^2$, one has:

$$\nabla f(x) = J(x)^T r(x) \tag{1.9a}$$

$$\nabla^2 f(x) = J(x)^T J(x) + S(x), \tag{1.9b}$$

where $J(x) = \left[\dfrac{\partial r_i}{\partial x_j}\right]_{(i,j)}$ is the Jacobian matrix of the residuals and the second component of the Hessian $S(x) = \sum_{i=1}^{d} r_i(x)\nabla^2 r_i(x)$. The latter is expensive in both computational time

and storage, since it requires d computations of $n \times n$ matrices. Hence, this component of the Hessian is the one to be approximated.

The Jacobian matrix of constraints function h is noted A.

When considering iterative methods for solving problem (1.1), k will, if not mentioned otherwise, refer to the iteration number. In order to simplify notations, quantities relative to a given iteration will be noted with the iteration number as an index, such as x_k for the iterate, r_k for $r(x_k)$,J_k for $J(x_k)$ etc.

Symbol := shall be used to state the definition of a numerical object (function, vector etc.)

We now address is a quick review of the three most popular classes of approximations. For a comprehensive review of these methods, we refer the reader to the chapter 10 of [3].

1.3.1 Gauss-Newton method

Originally used by Gauss the prince himself, **Gauss–Newton** (GN) approximation sets $S(x)$ to the zero matrix. It is the cheapest to compute, since the Jacobian is necessary to evaluate the gradient and works well in practice for zero residuals problems [3]. When solving problem (1.1) using an iterative method, this approximation amounts to linearizing the residuals function within the norm. Indeed, approximating $\nabla f^2(x)$ by $J(x)^T J(x)$ in a quadratic model $\mathcal{Q}$ of f around x gives

$$\mathcal{Q}^{GN}(p) = \frac{1}{2} p^T \nabla f^2(x) p + \nabla f(x)^T p = \frac{1}{2} \left\| J(x)p + r(x) \right\|^2, \tag{1.10}$$

which corresponds to injecting the linearization $r(x+p) \approx J(x)p + r(x)$ in the squared norm.

1.3.2 Levenberg-Marquardt

Next, we describe the **Levengerg–Marquardt** (LM) method, respectively named after the first author to publish it [5] and the author of its best-known rediscovery [6]. As noted by Marquardt in his paper, the two authors started from a different line of reasoning but came to the same conclusion. In this method, matrix $S(x)$ is set to a multiple of the identity matrix σI where σ is a positive scalar. The latter is called regularization parameter, because setting $\nabla f^2(x)$ to $J(x)^T J(x)$ leads to the quadratic model

$$\mathcal{Q}^{LM}(p) = \frac{1}{2} \left\| J(x)p + r(x) \right\|^2 + \frac{\sigma}{2} \|p\|^2. \tag{1.11}$$

In other words, one can see very schematically that

$$\text{LM model} = \text{GN model} + \text{regularization term}.$$

In practice, the LM approximation works well on zero and small residuals problems and tends to be more robust than the GN approximation. It is still cheap to compute but only requires updating the regularization parameter throughout the iterative process. This method is often referred as the early stage of the trust region methods [2]. For the link with regularization methods, consider the trust region problem

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & \|x\| \leq \Delta. \end{aligned}$$

By applying KKT conditions and assuming that there is a minimizer x^* lying on the trust region, i.e. $\|x^*\| = \Delta$, then (under strict complementarity), there is a strictly positive scalar (the multiplier) λ such that

$$\nabla f(x^*) + \frac{2\lambda}{\Delta} x^* = 0.$$

As a consequence, x^* is a critical point oh the unconstrained regularized problem

$$\min_x \quad f(x) + \frac{\delta}{2}\|x\|^2,$$

where $\delta = \frac{2\lambda}{\Delta}$.

Some references for LM methods: [1]

1.3.3 Quasi-Newton

Finally, one can compute an approximation of $\nabla^2 f(x)$ in a similar pattern as in **quasi-Newton** methods [7, Chapter 6] but targeted on the second order components. It has a higher computational cost than the previous two but is more accurate on large residuals problems. In [4], the authors exploit this approach in a adaptative scheme, where an estimation of the curvature is used to decide whether or not the quasi-Newton approximation is worth to use compared to the Gauss-Newton one.

It is important to bear in mind that choosing between a "cheap" approximation and a quasi-Newton type one implies making compromises. Depending on the initialization, the quasi-Newton approximation will take some iterations to be good and the accuracy will not be there when most needed, i.e. at the starting point potentially far from the solution, and will match the Gauss-Newton close to the solution on small residuals problems. In other words, the quasi-Newton is not accurate enough when most needed and very accurate when a way cheaper alternative does the same job.

References for quasi-Newton specialized to least-squares: [4, 9].

2 Augmented Lagrangian reformulation

In this section, we introduce the framework of Augmented Lagrangian-based algorithms and describe an application in the least-squares setting of problem (1.1).

2.1 Generalities

In order to remain general, we temporarily consider the mathematical program (1.3) and shall comeback to our beloved least-squares shortly after. We introduce the Augmented Lagrangian (AL) function associated to program (1.3):

$$\Phi_A(x, \lambda, \mu) := f(x) + \langle \lambda, h(x) \rangle + \frac{\mu}{2}\|h(x)\|^2, \tag{2.1}$$

where $\lambda \in \mathbb{R}^m$ is the vector of Lagrange multipliers and $\mu > 0$ is the penalty parameter.

Function (2.1) is nothing than the Lagrangian (1.4) with a quadratic penalty term, hence the adjective *Augmented*.Assume we fixed λ and μ, the problem of interest is now the bound constrained program

$$\begin{aligned} \min_x \quad & \Phi_A(x, \lambda, \mu) \\ \text{s.t.} \quad & l \leq x \leq u. \end{aligned} \tag{2.2}$$

AL-based algorithms fall into the class of penalty methods that generally enable one to use iterative methods for unconstrained optimization while steel achieving feasibility. In our case, moving the nonlinear constraints into the objective simplifies the set of constraints since only bounds constraints are left.

One of the pros of AL methods is that they come naturally with an update formula for the multipliers. Let (x_k, λ_k) be the current primal-dual iterate, then one can choose λ_{k+1} as

$$\lambda_{k+1} = \lambda_k + \mu_k h(x_k). \tag{2.3}$$

This relation is merely derived from $\nabla_x \Phi_A = 0$ by identifying the right hand side of equation (2.3) as multipliers satisfying the KKT condition $\nabla_x \ell = 0$.

The procedure of an AL method is relatively and relies on solving successively problem (2.2) with respect to the primal variable until a first order critical point of the original problem (1.3) is found. At every iteration, the penalty parameter is increased and the multipliers are updated by formula (2.3). This general pattern is outlined in algorithm 1 and the main steps of the outer iteration are given in algorithm 2.

Algorithm 1 Basic AL algorithm for solving (1.3)

Require: Starting point (x_0^s, λ_0), parameter μ_0 parameter μ_0 and tolerances $\omega_*, \omega_0, \eta_*, \eta_0$.
  repeat
    Compute an approximate solution x_k of (2.2) starting from x_k^s with tolerance ω_k.
    if $\|h(x_k)\| < \eta_k$ **then**
      Update iterate and increase penalty parameter.
    else
      Restart minimization of (2.2) with a higher penalty parameter.
    end if
    Update tolerances
  until $\|h(x_k)\| < \eta_*$ **and** $\|\nabla_x \ell(x_k, \lambda_k)\| < \omega_*$
  Return current approximate solution.

TODO How to compute the approximate minimizer? Projected conjugate gradient!

2.2 Application to structured least-squares

We now consider program (1.1), for which the AL function is given by

$$\Phi_A(x, \lambda, \mu) := \frac{1}{2}\|r(x)\|^2 + \langle \lambda, h(x) \rangle + \frac{\mu}{2}\|h(x)\|^2, \tag{2.4}$$

Contrary to formulation (2.1), we keep the linear equality constraints as is and penalize the violation of the nonlinear constraints. Although the computation of the projection onto the set $\mathcal{X}$ shall differ, the framework remains the same, One has the following expression of the gradient:

$$\nabla_x \Phi_A(x, \lambda, \mu) = J(x)^T r(x) + A^T \pi(x, \lambda, \mu), \tag{2.5}$$

Algorithm 2 Outer iteration of basic AL algorithm with trust region

Step 1: Inner Iteration Starting from x_0^s, approximately solve $\min_x \Phi_A(x, y_k, \mu_k)$ by computing x_k such that $\|x_k - P(x_k - \nabla_k \Phi_A)\| \leq \omega_k$ by a trust region process.

If $\|h(x_k)\| \leq \eta_k$, execute **Step 2**.

Otherwise, excecute **Step 3**.

Step 2: Iterate Update Update y_{k+1} by formula (2.3) and set $x_{k+1}^s \leftarrow x_k$.

Choose new tolreances ω_{k+1}, η_{k+1} and new penalty parameter μ_{k+1}.

Increment k and go back to **Step 1**.

Step 3: Adjustment of the Penalty Parameter Choose μ_{k+1} significantly greater than μ_k.

Leave the iterate unchanged: $\left(x_{k+1}^s, \lambda_{k+1}\right) \leftarrow (x_k^s, \lambda_k)$.

Go back to **Step 1**.

---

with $\pi(x, \lambda, \mu) := \lambda + \mu h(x)$ is the first-order estimates of the Lagrange multipliers.

The Hessian is given by

$$\nabla_{xx}^2 \Phi_A(x, \lambda, \mu) = J(x)^T J(x) + \mu A(x)^T A(x) + S(x) + \sum_{i=1}^{d} \nabla^2 h_i(x) \pi(x, \lambda, \mu). \tag{2.6}$$

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

$$\begin{aligned} \min_x \quad & \Phi_A(x, \lambda, \mu) \\ \text{s.t.} \quad & x \in \mathcal{X} \end{aligned} \tag{2.7}$$

As for any other AL based algorithm, the idea behind MCWAL is to solve by an iterative method problem (2.7) until a first order critical point of problem (1.1) is found. At every iteration, a the new iterate will be computed after (approximately) solving a trust region subproblem formed after a quadratic model of the AL around the current iterate.

2.3 Subproblem

Given a primal-dual iterate (x_k, λ_k) and a penalty parameter μ_k, we consider a quadratic model of the AL around x_k:

$$\mathcal{Q}_k(p) = \frac{1}{2} \langle p, H_k p \rangle + \langle g_k, p \rangle , \tag{2.8}$$

where $H_k := \nabla_{xx}^2 \Phi_A(x_k, \lambda_k, \mu_k)$ or an approximation of it and $g_k := \nabla_x \Phi_A(x_k, \lambda_k, \mu_k)$.

Vector p denotes the unknown of the subproblem whose (approximate) solution p_k shall be used to compute the new iterate $x_{k+1} = x_k + p_k$.

For the linear constraints, we want $x_k + p \in \mathcal{X}$ which will be provided if:

- $Cp = 0$ (provided that $Cx_0 = b$)

- $x_k - l \leq p \leq u - x_k$

The above conditions shall be written $p \in \mathcal{X}_k$ where $\mathcal{X}_k := \{p \mid Cp = 0,\ x_k - l \leq p \leq u - x_k\}$.

As part of our method, we will also add a trust region constraint of the form $\|p\|_k \leq \Delta_k$ for a radius Δ_k and a norm $\|\cdot\|_k$. Index k in the latter means that the norm might depend on the iteration. A priori, we would use the euclidean norm.

The subproblem of an outer iteration is then given by

$$\begin{aligned} \min_{p \in \mathcal{X}_k} \quad & \mathcal{Q}_k(p) \\ \text{s.t.} \quad & \|p\|_k \leq \Delta_k. \end{aligned} \tag{2.9}$$

A first sketch of the MCWAL procedure is drawn in algorithm 3.

Algorithm 3 Sketch of MCWAL

```
Require: x_0 ∈ 𝒳, λ_0, μ_0, τ_0 and constants η_s
  while not optimal² do
      Evaluate H_k and g_k
      Compute a solution p_k of subproblem (2.9)
      Compute ratio ρ_k³
      if ρ_k ≥ η_s then                                   ▷ Good step
          x_{k+1} ← x_k + p_k
          Choose Δ_{k+1} > Δ_k
          if ‖h(x_k)‖ ≤ τ_k then
              y_{k+1} ← π(x_k, y_k, μ_k)
              Choose μ_{k+1} > μ_k and τ_{k+1} < τ_k
          else
              y_{k+1} ← y_k
              Choose μ_{k+1} < μ_k
          end if
      else                                                ▷ Bad step
          x_{k+1} ← x_k
          y_{k+1} ← y_k
          μ_{k+1} ← μ_k
          Choose Δ_{k+1} < Δ_k
      end if
  end while
```

References

[1] S. Bellavia, S. Gratton, and E. Riccietti. A Levenberg-Marquardt method for large nonlinear least-squares problems with dynamic accuracy in functions and gradients. *Numerische Mathematik*, 140:791–825, 2018. doi: 10.1007/s00211-018-0977-z.

[2] 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.

[3] J.E. Dennis Jr and R.B. Schnabel. *Numerical Methods for Unconstrained optimization and Nonlinear Equations*. Classics in Applied Mathematics. SIAM, Philadelphia, PA, USA, 1996. doi: 10.1137/1.9781611971200.

[4] 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.

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

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

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

[8] R.T. Rockafellar and R.J.B. Wets. *Variational Analysis*. Springer Berlin, Heidelberg, 1st edition, 1998. doi: 10.1007/978-3-642-02431-3.

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