Ongoing work on MCWAL

Pierre Borie

January 17, 2025

Abstract

This informal document reflects the ongoing work and thinking on a algorithm for constrained nonlinear least squares. The current algorithm (rapper) name is MCWAL for Moindres Carrés With Augmented Lagrangian.

1 Introduction

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

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

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 $\mathbb{R}^n$, $(m \leq n)$, $b = (b_1,\dots,b_m)^T \in \mathbb{R}^m$ 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

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

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 and reminders of nonlinear programming

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.)

In this section 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)[6, 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.

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) [6, Theorem 12.5]*** *If x^* satisfies the second order conditions, then x^* is a local minimum of (1.3).*

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

1.2 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:

$$\nabla f(x) = J(x)^T r(x) \tag{1.5a}$$

$$\nabla^2 f(x) = J(x)^T J(x) + S(x), \tag{1.5b}$$

where $J(x) = \left[\frac{\partial r_i}{\partial x_j}\right]_{(i,j)}$ 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 reviews, we refer the reader to the chapter 10 of [2].

The **Gauss–Newton** 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 [2]. When solving problem (1.1) using an iterative method, this approximation amounts to linearizing the residuals function within the norm.

Next, the **Levengerg–Marquardt** (LM) method [4, 5] sets $S(x)$ to a multiple of the identity matrix σI where σ is a regularization parameter. It is still cheap to compute but only requires updating the regularization parameter throughout the iterative process. This method can be seen as the early stage of the trust region methods [1]. It works well in practice on zero and small residuals problems.

Finally, one can compute an approximation of $\nabla^2 f(x)$ in a similar pattern as in **quasi-Newton** methods [6, 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 [3], 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.

2 Augmented Lagrangian reformulation

In this section, we introduce the framework of Augmented Lagrangian-based algorithms and describe an application to our least-squares case.

2.1 Generalities

In order to remain general, we temporarly consider the mathematical program (1.3) and shall comeback to the 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*.

2.2 Application to structured least-squares

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

$$\Phi_A(x, \lambda, \mu) := \frac{1}{2} \|r(x)\|^2 + \langle \lambda, h(x) \rangle + \frac{\mu}{2} \|h(x)\|^2, \tag{2.2}$$

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

$$\nabla_x \Phi_A(x, \lambda, \mu) = J(x)^T r(x) + A^T \pi(x, \lambda, \mu), \tag{2.3}$$

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

The Hessian is given by

$$\nabla^2_{xx} \Phi_A(x, \lambda, \mu) = J(x)^T J(x) + \mu A(x)^T A(x) + S(x) + \sum_{i=1}^{d} \nabla^2 h_i(x) \pi(x, \lambda, \mu). \tag{2.4}$$

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

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

As for any other AL based algorithm, the idea behind MCWAL is to solve by an iterative method problem (2.5) 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:

$$\mathcal{Q}_k(p) = \frac{1}{2} \langle p, H_k p \rangle + \langle g_k, p \rangle , \tag{2.6}$$

where $H_k := \nabla^2_{xx} \Phi_A(x_k, \lambda_k, \mu_k)$ 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

$$\begin{aligned} \min_{p \in \mathcal{X}_k} \quad & \mathcal{Q}_k(p) \\ \text{s.t.} \quad & \|p\|_k \leq \Delta_k. \end{aligned} \tag{2.7}$$

A first sketch of the MCWAL procedure is drawn in algorithm 1.

Algorithm 1 Sketch of MCWAL

Require: $x_0 \in \mathcal{X}$, λ_0, μ_0, τ_0 and constants η_s
  while not optimal[2] **do**
    Evaluate H_k and g_k
    Compute a solution p_k of subproblem (2.7)
    Compute ratio ${\rho_k}^3$
    if $\rho_k \geq \eta_s$ **then** ▷ Good step
      $x_{k+1} \leftarrow x_k + p_k$
      Choose $\Delta_{k+1} > \Delta_k$
      if $\|h(x_k)\| \leq \tau_k$ **then**
        $y_{k+1} \leftarrow \pi(x_k, y_k, \mu_k)$
        Choose $\mu_{k+1} > \mu_k$ and $\tau_{k+1} < \tau_k$
      else
        $y_{k+1} \leftarrow y_k$
        Choose $\mu_{k+1} < \mu_k$
      end if
    else ▷ Bad step
      $x_{k+1} \leftarrow x_k$
      $y_{k+1} \leftarrow y_k$
      $\mu_{k+1} \leftarrow \mu_k$
      Choose $\Delta_{k+1} < \Delta_k$
    end if
  end while

References

[1] 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.

[2] 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.

[3] 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.

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

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

[6] 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.