Inside Oqla and Qpalm

Jean Charles Gilbert, Émilie Joannopoulos

Inside Oqla and Qpalm

Jean Charles Gilbert[1] and Émilie Joannopoulos[1]

January 28, 2015

This report describes the technical details of the implementation of the augmented Lagrangian algorithm used in Oqla and Qpalm, which are pieces of software designed to solve a convex quadratic optimization problem. The goal of the report is to make easier the reading and the understanding of the C++/Matlab functions defining the solvers. The Oqla and Qpalm user's guides can be found elsewhere.

[1]INRIA-Paris-Rocquencourt, BP 105, F-78153 Le Chesnay Cedex (France); e-mails: `Jean-Charles.Gilbert@inria.fr` and `emilie.joannopoulos@inria.fr`.

1 The problem to solve

The pieces of software Oqla and Qpalm aim at solving a *convex quadratic optimization problem*, which is assumed to have the following structure

$$\text{(P)} \quad \begin{cases} \inf_{x\in\mathbb{R}^n} \ g^\mathsf{T}x + \frac{1}{2}\,x^\mathsf{T}Hx \\ l_B \leqslant x \leqslant u_B \\ l_I \leqslant A_I x \leqslant u_I \\ A_E x = b_E. \end{cases} \tag{1.1}$$

In this problem,

- $g \in \mathbb{R}^n$ is the gradient of the objective at the origin;
- $H \in \mathbb{R}^{n\times n}$ is the Hessian matrix of the objective, which is assumed to be symmetric and positive semi-definite;
- l_B and $u_B \in \overline{\mathbb{R}}^n$ specify lower and upper bounds on x; they must satisfy $l_B < u_B$ (i.e., $l_i < u_i$ for all indices $i \in B \equiv [1\,{:}\,n] \equiv \{1,\dots,n\}$);
- $A_I \in \mathbb{R}^{m_I\times n}$ is the inequality constraint Jacobian; I is supposed to be an index set whose cardinality is $m_I := |I|$; the rows of A_I are denoted by A_i for $i \in I$;
- $l_I \in \overline{\mathbb{R}}^{m_I}$ specify lower and upper bounds on $A_I x$; they must satisfy $l_I < u_I$ (i.e., $l_i < u_i$ for all indices $i \in I$);
- $A_E \in \mathbb{R}^{m_E\times n}$ is the equality constraint Jacobian; E is supposed to be an index set whose cardinality is $m_E := |E|$; the rows of A_E are also denoted by A_i for $i \in E$;
- $b_E \in \mathbb{R}^{m_E}$.

One of the goals of this paper is to adapt to the more general framework given above the results established in [5, 4, 15], where it is assumed that the feasible set is defined by using only linear inequality constraints.

We will denote the cost function by $q : \mathbb{R}^n \to \mathbb{R}$, hence with the value at $x \in \mathbb{R}^n$ given by

$$q(x) = g^\mathsf{T}x + \frac{1}{2}x^\mathsf{T}Hx.$$

Below, the inequality constraints will be written more compactly as follows $l \leqslant (x, A_I x) \leqslant u$, in which

$$l = \begin{pmatrix} l_B \\ l_I \end{pmatrix} \qquad \text{and} \qquad u = \begin{pmatrix} u_B \\ u_I \end{pmatrix}.$$

Since H may vanish, (1.1) also models linear optimization.

It is convenient to denote by $B := [1\,{:}\,n]$ the index set of the bounds on x and by $A_B = I_n$ the identity matrix of order n. Then the inequalities $l \leqslant (x, A_I x) \leqslant u$ read in a similar manner $l_B \leqslant A_B x \leqslant u_B$ and $l_I \leqslant A_I x \leqslant u_I$. We also take the notation

$$m = n + m_I + m_E \qquad \text{and} \qquad A = \begin{pmatrix} A_B \\ A_I \\ A_E \end{pmatrix} \in \mathbb{R}^{m\times n}.$$

An inequality constraint with the index $i \in B \cup I$ is said to be *active* at x if $A_i x = l_i$ or u_i. The *active set* at x is defined by

$$\mathcal{A}(x) := \{i \in B \cup I : A_i x = l_i \text{ or } u_i\}.$$

The *Lagrangian* of problem (1.1) is written

$$\ell(x,\lambda) = g^{\mathsf{T}}x + \frac{1}{2}\,x^{\mathsf{T}}Hx + (\lambda^l_{B\cup I})^{\mathsf{T}}(l - A_{B\cup I}x) + (\lambda^u_{B\cup I})^{\mathsf{T}}(A_{B\cup I}x - u) + \lambda_E^{\mathsf{T}}(A_Ex - b_E),$$

in which the vector of multipliers is

$$\lambda := (\lambda^l_{B\cup I}, \lambda^u_{B\cup I}, \lambda_E).$$

The solver Oqla is written in C++, while Qpalm is written in Matlab.

2 The solution method

2.1 Algorithmic scheme

The method used by Oqla and Qpalm to solve problem (1.1) works on a transformation of the problem, using an auxiliary variable $y \in \mathbb{R}^{m_I}$, namely

$$\text{(P')} \quad \begin{cases} \inf\ g^{\mathsf{T}}x + \frac{1}{2}\,x^{\mathsf{T}}Hx \\ l \leqslant (x,y) \leqslant u \\ A_Ix = y \\ A_Ex = b_E. \end{cases} \tag{2.1}$$

Note that the auxiliary variable y is only associated with the complex linear inequalities $l_I \leqslant A_Ix \leqslant u_I$, not with the bound constraints $l_B \leqslant x \leqslant u_B$. The goal is to have simple bound constraints on the variables and additional linear equality constraints.

The *augmented Lagrangian*, which plays a major part in the solution method, is the one associated with the equality constraint of (P'). It is a function defined on $\mathbb{R}^n \times \mathbb{R}^{m_I+m_E}$, which is defined at (x,λ) for an *augmentation parameter* $r > 0$ by

$$\begin{aligned}\ell_r(x,y,\lambda) = g^{\mathsf{T}}x + \frac{1}{2}\,x^{\mathsf{T}}Hx + \lambda_I^{\mathsf{T}}(A_Ix - y) + \frac{r}{2}\|A_Ix - y\|^2 \\ + \lambda_E^{\mathsf{T}}(A_Ex - b_E) + \frac{r}{2}\|A_Ex - b_E\|^2,\end{aligned} \tag{2.2}$$

where $\|\cdot\|$ denotes the Euclidean norm. The AL algorithm implemented in Oqla and Qpalm is built on the following scheme.

Algorithm 2.1 (AL algorithm) Let be given initial multiplier $\lambda \in \mathbb{R}^{m_I+m_E}$.

while *convergence* **do**

Let (x^+,y^+) be a solution, if any, of the AL subproblem

$$\inf_{(x,y)\in[l,u]} \ell_r(x,y,\lambda). \tag{2.3}$$

Updates the multiplier by the formula

$$\lambda^+ = \lambda + r\begin{pmatrix} A_Ix^+ - y^+ \\ A_Ex^+ - b_E \end{pmatrix}. \tag{2.4}$$

end

Problem (2.3) may have several solutions (x^+, y^+) but all of them gives the same value to the constraints $(A_I x^+ - y, A_E x^+ - b_E)$, so that the new multiplier in (2.4) is independent of the chosen solution (x^+, y^+).

Therefore, the algorithm implemented in Oqla and Qpalm decomposes the original problem (1.1) in a sequence of less difficult subproblems (2.3), while (2.4) has a straightforward implementation. These AL subproblems are less difficult since they only have bound constraints on the variables. This feature of the feasible set allows for the use of projections. The algorithm implemented in Oqla and Qpalm to solve (2.3) combines gradient projection (GP) steps and an active set method in a manner similar to the one proposed by Moré and Toraldo [23, 24; 1989-1991]. A distinctive aspect of the implementation comes, however, from the special role played by the variable y in the subproblem. The Hessian of the AL with respect to y is indeed a multiple of the identity matrix.

The fundamental assumption, which guarantees that the AL algorithm works correctly is that

$$\text{problem (2.3) has a solution} \tag{2.5}$$

or, equivalently that it is bounded (the optimal value is not $-\infty$), for some (or any) $\lambda \in \mathbb{R}^{m_I+m_E}$ and $r > 0$ (see proposition 2.5 in [4]). If (2.5) fails, the *closest feasible problem* to problem (1.1) is unbounded; in particular, problem (1.1) is unbounded if it is feasible.

2.2 Some theory

In this section, we adapt part of the theory developed in [4; 2014] to the present context. The goal of that paper is to analyze the behavior of the AL algorithm when the QP is infeasible. With the equality constraints and bound constraints on the primal variable x, the QP (1.1) is a little more complex than the one considered in [4], which has only linear inequality constraints. This makes the analysis a little more difficult, but the logic works the same. We summarize below its aspects that are useful to understand the design of the solvers Oqla and Qpalm.

Let us start by identifying the conditions under which problem (1.1) is unbounded.

Proposition 2.2 (unboundedness) *Suppose that the convex quadratic optimization problem (1.1) is feasible. Then the problem is unbounded if and only if there is a direction $d \in \mathbb{R}^n$ such that*

$$g^T d < 0, \quad Hd = 0, \quad d \in [l_B, u_B]^\infty, \quad A_I d \in [l_I, u_I]^\infty, \quad \text{and} \quad A_E d = 0. \tag{2.6}$$

Proof. [⇐] It is clear that the conditions (2.6) imply the unboundedness of the feasible problem (1.1) since, given an arbitrary feasible point x_0, the points $x_\alpha = x_0 + \alpha d$ with $\alpha \geqslant 0$ are feasible and $q(x_\alpha) = q(x_0) + \alpha(g^T d) \to -\infty$ when $\alpha \to \infty$.

[⇒] If (1.1) is unbounded, this is also the case of the problem

$$\begin{cases} \inf_{x\in\mathbb{R}^n} \ g^\mathsf{T}x + \frac{1}{2}x^\mathsf{T}Hx \\ l_B \leqslant x \leqslant u_B \\ l_I \leqslant A_I x \leqslant u_I \\ -\infty \leqslant A_E x \leqslant b_E \\ -\infty \leqslant -A_E x \leqslant -b_E, \end{cases}$$

which has the same cost function and the same feasible set as (1.1). Applying lemma 2.2 in [4] to this latter problem yields (2.6). □

Below, we say that $d \in \mathbb{R}^n$ is a *direction of unboundedness* if it satisfies the conditions in (2.6). By proposition 2.2, the QP is unbounded if and only if it has a direction of unboundedness. The solvers Oqla and Qpalm are designed to detect such a direction when the closest feasible QP is unbounded.

Remark 2.3 If we apply proposition 2.2 to the feasible problem (2.3) with $r > 0$, we see that this problem is unbounded (or, equivalently, it has no solution) if and only if there is a direction $(d_x, d_y) \in \mathbb{R}^n \times \mathbb{R}^{m_I}$ such that

$$\Big[g + Hx + A_I^\mathsf{T}[\lambda_I + r(A_I x - y)] + A_E^\mathsf{T}[\lambda_E + r(A_E x - b_E)]\Big]^\mathsf{T} d_x - \Big[\lambda + r(A_I x - y)\Big]^\mathsf{T} d_y < 0,$$
$$\begin{pmatrix} H + rA_{I\cup E}^\mathsf{T} A_{I\cup E} & -rA_I^\mathsf{T} \\ -rA_I & rI \end{pmatrix} \begin{pmatrix} d_x \\ d_y \end{pmatrix} = 0, \qquad \text{and} \qquad (d_x, d_y) \in [l, u]^\infty.$$

These conditions are equivalent to (2.6) and $(d_x, d_y) = (d, A_I d)$. □

A key notion is the one of feasible shift. We say that $s = (s_I, s_E) \in \mathbb{R}^{m_I} \times \mathbb{R}^{m_E}$ is a *feasible shift* for the constraints of problem (P), if there is an $x \in \mathbb{R}^n$ such that

$$l \leqslant (x, A_I x + s_I) \leqslant u \qquad \text{and} \qquad A_E x + s_E = b_E. \tag{2.7}$$

Notice that there is no need to introduce shifts for the bound constraints $l_B \leqslant x \leqslant u_B$, since these are always feasible (by the assumption $l_B < u_B$). On the other hand, these bound constraints must intervene in (2.7) since the searched x must be in $[l_B, u_B]$. We denote by $\mathcal{S} \subset \mathbb{R}^{m_I+m_E}$, the set of feasible shifts. It is clear that

$$\mathcal{S} = \begin{pmatrix} [l_I, u_I] \\ b_E \end{pmatrix} - \begin{pmatrix} A_I \\ A_E \end{pmatrix} [l_B, u_B].$$

We denote by $\tilde{s} = (\tilde{s}_I, \tilde{s}_E)$ the smallest feasible shift, which is the projection of zero on $\mathcal{S}$ or the solution to the optimization problem in

$$\tilde{s} := \operatorname*{arg\,min}_{s\in\mathcal{S}} \|s\|. \tag{2.8}$$

This problem also reads

$$\begin{cases} \min_{(x,y,s)} \ \frac{1}{2}\|s\|^2 \\ (x, y) \in [l, u] \\ A_I x + s_I = y \\ A_E x + s_E = b_E. \end{cases}$$

Now, one can eliminate $s \in \mathbb{R}^{m_I+m_E}$, so that the problem is only expressed in terms of the variables $(x, y) \in \mathbb{R}^n \times \mathbb{R}^{m_I}$:

$$\min_{(x,y)\in[l,u]} \tfrac{1}{2}\|A_I x - y\|^2 + \tfrac{1}{2}\|A_E x - b_E\|^2. \tag{2.9}$$

The elimination of s is useful to characterize the solution to (2.8) in terms of the primal variables (x, y) generated by Oqla and Qpalm. This is the goal of proposition 2.4 below, which gives optimality conditions for problem (2.8), without using the unknown vector $\tilde{s}$, by considering the equivalent problem (2.9). For compactness, we use in its statement (ii) the orthogonal projector $P_{-\,\mathrm{T}_{\bar{x}}[l_B,u_B]}$ on $-\,\mathrm{T}_{\bar{x}}[l_B, u_B]$, the opposite of the tangent cone to $[l_B, u_B]$ at $\bar{x}$. Since

$$d \in \mathrm{T}_{\bar{x}}[l_B, u_B] \quad \Longleftrightarrow \quad \forall\, i \in [1\!:\!n] : \quad d_i \begin{cases} \geqslant 0 & \text{if} \quad l_i = \bar{x}_i \\ \leqslant 0 & \text{if} \quad \bar{x}_i = u_i, \end{cases}$$

there holds

$$\forall\, i \in [1\!:\!n] : \quad \big(P_{-\,\mathrm{T}_{\bar{x}}[l_B,u_B]} v\big)_i = \begin{cases} \min(0, v_i) & \text{if} \quad l_i = \bar{x}_i \\ v_i & \text{if} \quad l_i < \bar{x}_i < u_i \\ \max(0, v_i) & \text{if} \quad \bar{x}_i = u_i. \end{cases}$$

Therefore

$$P_{-\,\mathrm{T}_{\bar{x}}[l_B,u_B]} v = 0 \quad \Longleftrightarrow \quad v_i \begin{cases} \geqslant 0 & \text{if} \quad l_i = \bar{x}_i \\ = 0 & \text{if} \quad l_i < \bar{x}_i < u_i \\ \leqslant 0 & \text{if} \quad \bar{x}_i = u_i. \end{cases}$$

Proposition 2.4 (caracterizations of the smallest feasible shift) *The following properties of $(\bar{x}, \bar{y}) \in [l, u]$ are equivalent:*
(i) $(\bar{x}, \bar{y})$ is a solution to problem (2.9),
(ii) $P_{-\,\mathrm{T}_{\bar{x}}[l_B,u_B]} \big(A_I^\mathsf{T}(A_I\bar{x} - \bar{y}) + A_E^\mathsf{T}(A_E\bar{x} - b_E)\big) = 0$ and $P_{[l_I,u_I]}(A_I\bar{x}) = \bar{y}$,
(iii) $A_I\bar{x} + \tilde{s}_I = \bar{y}$ and $A_E\bar{x} + \tilde{s}_E = b_E$.

Proof. $[(i) \Leftrightarrow (ii)]$ Since problem (2.9) consists in minimizing a convex function on a box, the pair $(\bar{x}, \bar{y}) \in [l, u]$ solves that problem if and only if the projected gradient vanishes at that point, which reads

$$P_{-\,\mathrm{T}_{(\bar{x},\bar{y})}[l,u]} \begin{pmatrix} A_I^\mathsf{T}(A_I\bar{x} - \bar{y}) + A_E^\mathsf{T}(A_E\bar{x} - b_E) \\ -(A_I\bar{x} - \bar{y}) \end{pmatrix} = 0.$$

This can be decomposed in

$$P_{-\,\mathrm{T}_{\bar{x}}[l_B,u_B]} \Big(A_I^\mathsf{T}(A_I\bar{x} - \bar{y}) + A_E^\mathsf{T}(A_E\bar{x} - b_E)\Big) = 0 \qquad \text{and} \qquad P_{-\,\mathrm{T}_{\bar{y}}[l_I,u_I]}(\bar{y} - A_I\bar{x}) = 0.$$

The first identity is the first part of (ii). The second identity is equivalent to

$$\begin{cases} \bar{y}_i \geqslant A_i\bar{x} & \text{if} \quad l_i = \bar{y}_i \\ \bar{y}_i = A_i\bar{x} & \text{if} \quad l_i < \bar{y}_i < u_i \\ \bar{y}_i \leqslant A_i\bar{x} & \text{if} \quad \bar{y}_i = u_i \end{cases} \tag{2.10}$$

or equivalently $P_{[l_I,u_I]}(A_I\bar{x}) = \bar{y}$, which is the second part of (ii).

[$(i) \Leftrightarrow (iii)$] This is a direct consequence of the fact that (2.8) and (2.9) are two expressions of the same problem. □

The *closest feasible problem* (CFQP) is the one in which the constraints are shifted with $\tilde{s}$:

$$(\mathrm{P}_{\tilde{s}}) \quad \begin{cases} \inf_x \ g^\mathsf{T} x + \frac{1}{2} x^\mathsf{T} H x \\ l \leqslant (x, A_I x + \tilde{s}_I) \leqslant u \\ A_E x + \tilde{s}_E = b_E. \end{cases} \tag{2.11}$$

Since $\tilde{s} \in \mathcal{S}$, this problem is feasible (at least in exact arithmetics). Using the auxiliary variable y, we obtain

$$(\mathrm{P}'_{\tilde{s}}) \quad \begin{cases} \inf_{(x,y)} \ g^\mathsf{T} x + \frac{1}{2} x^\mathsf{T} H x \\ l \leqslant (x, y) \leqslant u \\ A_I x + \tilde{s}_I = y \\ A_E x + \tilde{s}_E = b_E. \end{cases} \tag{2.12}$$

The next proposition gives optimality conditions of the closest feasible problem (2.11) without using the unknown smallest feasible shift $\tilde{s}$, but in terms of concepts and quantities that are known to the QP solvers Oqla and Qpalm.

Proposition 2.5 (stopping criterion) *Let $r \geqslant 0$ and let ℓ_r be the augmented Lagrangian (2.2). Then $(\bar{x}, \bar{y})$ is a solution to the closest feasible problem (2.12) if and only if there is some $\bar{\lambda} = (\bar{\lambda}_I, \bar{\lambda}_E) \in \mathbb{R}^{m_I + m_E}$ such that*

$$(\bar{x}, \bar{y}) \in \operatorname*{arg\,min}_{(x,y) \in [l,u]} \ell_r(x, y, \bar{\lambda}), \tag{2.13a}$$

$$P_{-\,\mathrm{T}_{\bar{x}}[l_B, u_B]}\Big(A_I^\mathsf{T}(A_I\bar{x} - \bar{y}) + A_E^\mathsf{T}(A_E\bar{x} - b_E)\Big) = 0, \tag{2.13b}$$

$$P_{[l_I, u_I]}(A_I\bar{x}) = \bar{y}. \tag{2.13c}$$

Proof. The most straightforward way of proving the equivalence is to write the long lists of optimality conditions of the problems (2.12) and (2.13a), which by convexity and constraint qualification characterize the solutions of these problems, to add (2.13b) and (2.13c) to the list of optimality conditions (2.13a), and to show that one can deduce any resulting list from the other. This is what we do below.

The optimality conditions of (2.12) read: $(\bar{x}, \bar{y}) \in [l, u]$ solves that problem if and only if there exist multipliers $(\lambda^l, \lambda^u, \lambda) \in \mathbb{R}^{n+m_I} \times \mathbb{R}^{n+m_I} \times \mathbb{R}^{m_I+m_E}$ such that

$$\begin{gathered} g + H\bar{x} + \lambda_B^u - \lambda_B^l + A_I^\mathsf{T}\lambda_I + A_E^\mathsf{T}\lambda_E = 0, \\ \lambda_I = \lambda_I^u - \lambda_I^l, \\ 0 \leqslant \lambda_B^l \perp (\bar{x} - l_B) \geqslant 0, \\ 0 \leqslant \lambda_B^u \perp (u_B - \bar{x}) \geqslant 0, \\ 0 \leqslant \lambda_I^l \perp (\bar{y} - l_I) \geqslant 0, \\ 0 \leqslant \lambda_I^u \perp (u_I - \bar{y}) \geqslant 0, \\ A_I\bar{x} + \tilde{s}_I = \bar{y}, \qquad (2.14a) \\ A_E\bar{x} + \tilde{s}_E = b_E. \qquad (2.14b) \end{gathered}$$

The optimality conditions of (2.13a) read: $(\bar{x}, \bar{y}) \in [l, u]$ solves that problem if and only if there exist multipliers $(\lambda^l, \lambda^u) \in \mathbb{R}^{n+m_I} \times \mathbb{R}^{n+m_I}$ such that

$$
\begin{aligned}
g + H\bar{x} + \lambda_B^u - \lambda_B^l + A_I^T \left(\bar{\lambda}_I + r(A_I\bar{x} - \bar{y})\right) + A_E^T \left(\bar{\lambda}_E + r(A_E\bar{x} - b_E)\right) = 0,\\
\bar{\lambda}_I + r(A_I\bar{x} - \bar{y}) = \lambda_I^u - \lambda_I^l,\\
0 \leqslant \lambda_B^l \perp (\bar{x} - l_B) \geqslant 0,\\
0 \leqslant \lambda_B^u \perp (u_B - \bar{x}) \geqslant 0,\\
0 \leqslant \lambda_I^l \perp (\bar{y} - l_I) \geqslant 0,\\
0 \leqslant \lambda_I^u \perp (u_I - \bar{y}) \geqslant 0.
\end{aligned}
$$

The proof is concluded by observing that (2.14a) and (2.14b) are equivalent to (2.13b) and (2.13c), thanks to the equivalence $(ii) \Leftrightarrow (iii)$ of proposition 2.4, and by making the following correspondence between λ and $\bar{\lambda}$:

$$
\lambda_I \equiv \bar{\lambda}_I + r(A_I\bar{x} - \bar{y}) \qquad \text{and} \qquad \lambda_E \equiv \bar{\lambda}_E + r(A_E\bar{x} - b_E). \qquad \square
$$

3 Algorithmic details

3.1 The AL algorithm

3.1.1 Notation

We introduce

$$
s_k := \begin{pmatrix} y_k - A_I x_k \\ b_E - A_E x_k \end{pmatrix}. \tag{3.1}
$$

3.1.2 Update of the augmentation parameters

Oqla and Qpalm offer 3 ways of tuning the augmentation parameters r_k during the run. This is specified by the option `options.rctl` ("r control"). This option is specified by a string that can take one of the three values:

- `'fixed'`, in which case r_k is maintained fixed during the iterations,
- `'constraint'`, in which case r_k is updated from the end of iteration $k = 2$ in order to try to have linear convergence of the constraint value to zero at the rate specified by `options.dcr`; this option should be only used when the QP is known to be feasible and is recommended in that case;
- `'constraint_change'`, in which case r_k is updated
  - at the end of iteration $k = 2$ in order to try to have linear convergence of the constraint value to zero at the rate specified by `options.dcr` and
  - from the end of iteration $k = 3$ in order to try to have linear convergence of the *shifted* constraint value to zero at the rate specified by the same `options.dcr`;

  this last value is recommended and is, therefore, the default value of `options.rctl`.

More details on these options are given below.

Constant parameter

When `options.rctl` is set to `'fixed'`, the augmentation parameter r_k is maintained constant to the value `options.rlag` (its default value is 1) during all the run. This option is not recommended, if efficiency is desired. It may be useful, however, to check the behavior of the solver and is therefore provided for the algorithmicians.

Parameters adapted to the constraint decrease

If `options.rctl` is set to `'constraint'`, the augmentation parameters r_k are updated by Oqla and Qpalm at each iteration in order to try to get a linear rate of convergence to zero, the one specified by the option `options.dcr`, for the constraints. The initial value r_1, the one that is used at the first two AL iterations, is given in `options.rlag` (its default value is 1).

This parameter update technique is based on a study made in [5], which assumes that the QP has a solution. It is shown in that contribution that at each iteration

$$\rho_k \leqslant \frac{L}{r_k}, \tag{3.2}$$

where L denotes the unknown radial Lipschitz module at zero of $\partial\delta^{-1}$, the reciprocal of the dual function subgradient multifunction, and

$$\rho_k := \frac{\|s_{k+1}\|}{\|s_k\|}$$

is the ratio of two successive constraint norms. Since this ration can only been computed at the end of the second iteration, the value of the augmentation parameter remains the same during the first two iterations: $r_2 = r_1$. The estimate (3.2) is only valid when problem (1.1) has a solution.

Then, the update rule of r_k aims at having ρ_k less than the desired convergence rate

$$\rho_{\text{des}} \in]0, 1[,$$

given in `options.dcr`. The update rule is the following

$$r_{k+1} = \max\left(1, \frac{\rho_k}{\rho_{\text{des}}}\right) r_k,$$

meaning that r_k is not modified if $\rho_k \leqslant \rho_{\text{des}}$ and is multiplied by ρ_k/ρ_{des} otherwise. If the problem is infeasible, this rule generates an unbounded sequence $\{r_k\}$.

Parameters adapted to the shifted constraint decrease

If `options.rctl` is set to `'shifted_constraint'`, the augmentation parameters r_k are updated by Oqla and Qpalm at each iteration in order to get a linear rate of convergence to zero, the one specified by the option `options.dcr`, for the *shifted* of constraints. The initial value r_1, the one that is used at the first two AL iterations, is given in `options.rlag` (its default value is 1).

This parameter update technique is based on a study made in [4], which does not assume that the QP is feasible. It is shown that at each iteration

$$\|s_{k+1} - \tilde{s}\| \leqslant \frac{L}{r_k} \|s_k - \tilde{s}\|, \tag{3.3}$$

where s_k is defined by (3.1), $\tilde{s}$ is the smallest feasible shift defined by (2.8), and L denotes an unknown constant. Since $\tilde{s}$ is not known during the run, the quotient $\|s_{k+1}-\tilde{s}\|/\|s_k-\tilde{s}\|$ cannot be observed in order to decide whether an increase of r_k is necessary to get the desired convergence rate. A bypass can be realized by observing, instead, the behavior of

$$s'_k := s_{k+1} - s_k$$

which converges to zero, at a linear speed similar to that of $s_k - \tilde{s}$. Hence, one defines

$$\rho'_k := \frac{\|s'_k\|}{\|s'_{k-1}\|}$$

and the solvers adapt r_k to try to get

$$\rho'_k \leqslant \rho'_{\text{des}} := \frac{\rho_{\text{des}}}{1 + 2\rho_{\text{des}}}.$$

This is because when ρ'_k is permanently less than ρ'_{des}, then

$$\rho_k := \frac{\|s_{k+1} - \tilde{s}\|}{\|s_k - \tilde{s}\|}$$

is permanently less than ρ_{des} (see point 2 in [4; proposition 4.1]).

Then, the update rule of r_k is the following

$$r_{k+1} = \max\left(1, \frac{\rho'_k}{\rho'_{\text{des}}}\right) r_k,$$

meaning that r_k is not modified if $\rho'_k \leqslant \rho'_{\text{des}}$ and is multiplied by $\rho'_k/\rho'_{\text{des}}$ otherwise. This update rule maintains the sequence $\{r_k\}$ bounded even when the QP is infeasible.

Avoiding the parameter blowing-up

Having a too large value of the augmentation parameter r_k may have unfortunate consequences. A large r may, indeed, induce a severe ill-conditioning of the AL subproblems (2.3), at a point that the linear solvers intervening in their solution may have difficulties to solve their linear systems with a sufficient precision. In turn, the value of the constraints used in (2.4) may be far from the appropriate subgradient of the dual function, which may prevent the convergence of the updated multipliers by (2.4). In that case, the update rule of r_k discussed above would increase the augmentation parameter, which would deteriorate even more the situation. For this reason, Oqla and Qpalm use a heuristics to see whether

- the linear systems to solve are ill-conditioned and
- this ill-conditioning comes from a too large value of r_k.

Measuring the ill-conditioning is an expensive operation. To avoid it, Oqla and Qpalm use instead the accuracy measure `info.accu` returned by the linear solver (see sections 3.2.1), hence attributing a low accuracy (indicated by a small value of `info.accu`) to ill-conditioning. If `info.accu` is too low (i.e., less than `options.accu`), the next augmentation parameter r_{k+1} is set either to r_{k-1} (the augmentation parameter used before the current iteration) if this one exists, or to $r_k/10$ otherwise. This decrease of the augmentation parameter is done at most 3 times, since if `info.accu` is still too low after these 3 decreases, it is probably not due to a too large value of r_k but to an inherent ill-conditioning of the original problem.

If the user is not satisfied with this heuristic decrease of the augmentation parameter, it is always possible to fix r_k to a given value by setting `option.rctl` to `'fixed'` and `option.rlag` to the desired value.

3.1.3 Stopping criteria

Unconstrained problems

If the problem is unconstrained, Oqla and Qpalm stop on a small gradient of the objective; they stop at a point x satisfying

$$\|g + Hx\| \leqslant \varepsilon_g,$$

where $\varepsilon_g > 0$ is given by the user in `options.gtol`.

Bound constrained problems

If the constraints of the problem reduce to bounds on the variables, Oqla and Qpalm stop on a small projected gradient.

General feasible problems

When $m_I + m_E > 0$ and `options.rctl` is set to `'constraint'`, Oqla and Qpalm consider that the user of the solver knows that the inequality and/or equality constrained QP is feasible. In this case, a simplified stopping criterion is implemented. Therefore, if the problem is known to be feasible, it is better to declare this fact by setting `options.rctl` to `'constraint'`.

The pair $(\bar{x}, \bar{y})$ is a solution to a feasible problem (2.11) if and only if $(\bar{x}, \bar{y})$ is a minimizer on $[l, u]$ of the AL (for some multiplier $\bar{\lambda} \in \mathbb{R}^{m_I+m_E}$ and any augmentation parameter $r \geqslant 0$) and is feasible for problem (2.11). Since at iteration k of the AL algorithm, (x_{k+1}, y_{k+1}) is a minimizer of $(x, y) \in [l, u] \mapsto \ell_{r_k}(x, y, \lambda_k)$, optimality of the QP is reached if

$$A_I x_{k+1} - y_{k+1} = 0 \qquad \text{and} \qquad A_E x_{k+1} - b_E = 0. \tag{3.4}$$

Therefore, for a problem that is declared feasible by the user, Oqla and Qpalm only use the value of the constraints to decide when to stop. More precisely, the solvers decide to stop when

$$\gamma_{k+1} := \|s_{k+1}\| \leqslant \varepsilon_{\text{feas}}, \tag{3.5}$$

where s_{k+1} is defined by (3.1) and $\varepsilon_{\text{feas}} > 0$ is a feasibility threshold given by the user in `options.feas`.

In exact arithmetic, the criterion (3.5) can always be realized when the QP is feasible, whatever is $\varepsilon_{\text{feas}} > 0$. In floating point arithmetic, however, this is no longer the case, in particular when the positive threshold $\varepsilon_{\text{feas}}$ is very small. Oqla and Qpalm use the following mechanism to detect the prevalence of rounding errors and to stop verifying (3.5).

At the end of the first two iterations, say for $k = 1$ and $k = 2$, Oqla and Qpalm check whether (3.5) holds. At the end of the second iteration (for $k = 2$), the solvers can compute

$$\rho_k := \frac{\gamma_{k+1}}{\gamma_k}. \tag{3.6}$$

From the third iteration (for $k = 3$), Oqla and Qpalm can compare ρ_k to its previous values $\rho_2, \ldots, \rho_{k-1}$. It is on the basis of that comparison that the solvers decide to give up trying to realize (3.5). This abandonment is motivating by the following observation. A property of the proximal algorithm ensures that ρ_k is always < 1, so that $\log_{10} \rho_k < 0$ always holds. On the other hand, the solvers try to have ρ_k sufficiently small, actually less than $\rho_{\text{des}} \in]0, 1[$, by increasing the augmentation parameter r_k if necessary (see section 3.1.2), so that $\log_{10} \rho_k < 0$ should stay away from zero. It this does not occur, either rounding errors prevail or the problem is infeasible. For these reasons, Oqla and Qpalm decide at iteration $k_1 \geqslant 3$ to give up checking (3.5) and to stop if

$$\log_{10} \rho_{k_1} \geqslant \varepsilon_\rho \left(\min_{2 \leqslant i < k_1} \log_{10} \rho_i \right), \tag{3.7}$$

where $\varepsilon_\rho \in [0, 1[$ is set by `options.dcrf` (its default value is 1/100). This test is inspired by a stopping criterion proposed by Moré and Toraldo [24]. By taking $\varepsilon_\rho = 0$, the test (3.7) is made ineffective unless rounding errors make $\rho_{k_1} \geqslant 1$.

General infeasible problems

When `options.rctl` is *not* set to `'constraint'`, Oqla and Qpalm consider that the user of the solver does not know whether the inequality and/or equality constrained QP is feasible. In that case, the solvers adapt its stopping criteria during the iterations, according to its appreciation of the problem feasibility. This section explains how the solvers proceed.

Like for feasible problems, at the end of the first two iterations, say for $k = 1$ and $k = 2$, Oqla and Qpalm check whether (3.5) holds. Again, like for feasible problems, from the third iteration (for $k = 3$), Oqla and Qpalm compare ρ_k given by (3.6) to its previous values ρ_2, $\ldots$, ρ_{k-1}, and the solvers continue checking (3.5) as long as (3.7) does not hold. If (3.7) holds, for some $\varepsilon_\rho \in [0, 1[$ set by `options.dcrf` (its default value is 1/100), the solvers no longer stop iterating but switch in a new state in which they try to realize optimality of the closest feasible problem.

Proposition 2.5 shows that $(\bar{x}, \bar{y})$ is a solution to the closest feasible problem (2.12) if and only if $(\bar{x}, \bar{y})$ is a minimizer on $[l, u]$ of the AL (for some multiplier $\bar{\lambda} \in \mathbb{R}^{m_I + m_E}$ and any augmentation parameter $r \geqslant 0$, see (2.13a)) that is feasible for problem (2.12) (see (2.13b)-(2.13c) and proposition 2.4). Since at iteration k of the AL algorithm, (x_{k+1}, y_{k+1}) is a minimizer of $(x, y) \in [l, u] \mapsto \ell_{r_k}(x, y, \lambda_k)$, optimality of the closest feasible problem is reached if

$$P_{-\mathrm{T}_{x_{k+1}}[l_B, u_B]}\Big(A_I^\mathsf{T}(A_I x_{k+1} - y_{k+1}) + A_E^\mathsf{T}(A_E x_{k+1} - b_E)\Big) = 0, \tag{3.8a}$$

$$P_{[l_I, u_I]}(A_I x_{k+1}) = y_{k+1}. \tag{3.8b}$$

The stopping criterion associated with (3.8) is

$$\gamma_{k+1}^s := \left\| \begin{pmatrix} P_{-\mathrm{T}_{x_{k+1}}[l_B,u_B]}\Big(A_I^\mathsf{T}(A_I x_{k+1} - y_{k+1}) + A_E^\mathsf{T}(A_E x_{k+1} - b_E)\Big) \\ P_{[l_I,u_I]}(A_I x_{k+1}) - y_{k+1} \end{pmatrix} \right\| \leqslant \varepsilon_{\text{feas}}^s, \quad (3.9)$$

where $\varepsilon_{\text{feas}}^s > 0$ is a feasibility threshold given by the user in `options.feass`.

In exact arithmetic, the criterion (3.9) can always be realized, whatever is $\varepsilon_{\text{feas}}^s > 0$. In floating point arithmetic, however, this is no longer the case, in particular when the positive threshold $\varepsilon_{\text{feas}}^s$ is very small. Oqla and Qpalm use the same mechanism as the one described above to decide that rounding errors prevail and that the AL iterations must be interrupted. This will be the case if for some $k \geqslant k_1$:

$$\log_{10} \rho_k^s \geqslant \varepsilon_\rho \left(\min_{2 \leqslant i < k} \log_{10} \rho_i^s \right). \quad (3.10)$$

The same value of ε_ρ is used in both (3.7) and (3.10).

3.2 Linear solvers

Oqla and Qpalm solve the linear systems encountered during the solution process by preconditioned conjugate gradient iterations. Note indeed that the matrices intervening in the linear systems are positive semi-definite, thanks to the convexity of the original problem (1.1). In case, a matrix of a linear system is only positive semi-definite (this is the case if the original problem (1.1) is linear), both method may suffer from instabilities.

3.2.1 Conjugate gradient solver

Measure of the solution precision

When used in the framework of an augmented Lagrangian algorithm, a conjugate (CG) solver may suffer from ill-conditioning, not only because of the ill-conditioning present in the original QP (1.1) to solve, but also because of the one introduced by the algorithmic approach. Indeed, the AL subproblems, from which the linear systems are derived, can be much more ill-conditioned than the original QP, when the augmentation parameter r is large. Ill-conditioning prevents the CG algorithm from solving efficiently the linear systems (and therefore the AL subproblems), so that the theoretical speed-up of the convergence of the AL algorithm obtained by the increase of r is destroyed by the inaccuracy of the numerical solution of the AL subproblem.

The CG solver in `qpalm_tcg` measures its ability to compute a solution of a given linear system by comparing the iterated residual to the exact residual at the end of the iterative process. To make this more precise, suppose that the linear system consists in finding x such that $Ax = b$. The *exact residual* at the final iteration K is

$$r_K = Ax_K - b.$$

In floating point arithmetic, the *iterated residual* at the final iteration K is an approximation $\hat{r}_K$ of r_K, obtained thanks to the recurrence

$$\hat{r}_{k+1} = \hat{r}_k + \alpha_k p_k, \qquad \text{for } k = 1, \ldots, K-1,$$

where $\alpha_k > 0$ is the stepsize, $p_k = Ad_k$, and d_k is the displacement in x. In exact arithmetic, $\hat{r}_K = r_K$. An example of behavior of $\|r_k\|_2/\|r_1\|_2$ and $\|\hat{r}_k\|_2/\|r_1\|$ in logarithmic scale as

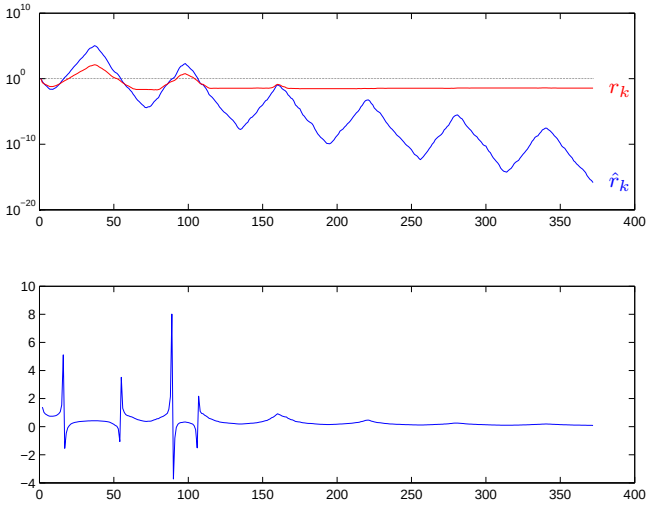

Figure 3.1: Above: example of behavior of the iterated ($\hat{r}_k$, in blue) and exact (r_k, in red) residuals as functions of the CG iteration counter k. Below: the accuracy of the computation measured by `info.accu` in (3.11) along the CG iterations.

functions of the iteration index k is shown in figure 3.1. As in the figure, it is often the case that the iterated residual carries on decreasing to zero until the stopping criterion is satisfied, while the exact residual remains stationary in norm rather rapidly. This is usually due to the ill-conditioning of the linear system. This ill-conditioning may come from a too large value of the augmentation parameter of from the Hessian H. It is measured in Oqla and Qpalm by the *accuracy number*

$$\texttt{info.accu} := \frac{\log_{10} \|r_K\|_2/\|r_1\|_2}{\log_{10} \|\hat{r}_K\|_2/\|r_1\|_2}. \tag{3.11}$$

In case the same linear system is solved on several affine subspaces, the minimum of the ratios in the right hand side, computed in each of these runs, is returned. It can be considered that the higher is `info.accu`, the more precise is the computation. Of course, to keep the computational cost reasonable, the exact residual is only computed when the CG solver stops iterating.

3.2.2 Detection of nonconvexity

The nonconvexity of the QP is usually detected during the CG iterations or when a preconditioner is built.

Note however, that the QP may not be convex and that this fact may not be seen by Oqla and Qpalm. In this case, the solvers find a local minimum of the problem. This phenomenon can be observed when the solvers deal with problem `values` of the CUTEst collection. Two reasons contribute to this phenomenon.

- The first reason is that one may have $H \not\succeq 0$ and $H + rA^\mathsf{T}A \succeq 0$ (the rows of A is formed of some rows of A_I and the rows of A_E), so that the nonconvexity cannot be viewed when solving the AL subproblem (2.3).
- The second reason is that one may have $H + rA^\mathsf{T}A \not\succeq 0$, but some bounds are activated

and $H + rA^T A$ is positive semi-definite on the corresponding subspace, so that, again, the nonconvexity cannot be viewed when solving the AL subproblem (2.3).

3.3 Minimization of the augmented Lagrangian subproblems

At each iteration, a multiplier $\lambda \in \mathbb{R}^{m_I+m_E}$ and a penalty parameter $r \geqslant 0$ are chosen and the AL subproblem (2.3) is solved. We recall that subproblem below

$$\inf_{(x,y)\in[l,u]} \ell_r(x, y, \lambda). \tag{3.12}$$

The augmented Lagrangian ℓ_r is defined in (2.2).

Since the Hessian of ℓ_r with respect to y is the identity matrix, the minimizer of $y \mapsto \ell_r(x, y, \lambda)$ on $[l_I, u_I]$ can be viewed as a projection for the Euclidean scalar product, allowing an analytical expression of its unique minimizer $\breve{y}(x)$ and of the function

$$\varphi_\lambda : x \in \mathbb{R}^n \mapsto \varphi_\lambda(x) := \ell_r(x, \breve{y}(x), \lambda) = \inf_{y\in[l_I,u_I]} \ell_r(x, y, \lambda) \in \overline{\mathbb{R}}, \tag{3.13}$$

where ℓ_r is defined by (2.2). This function φ_λ is differentiable. The QP subproblem (3.12) is equivalent to minimizing the piecewise quadratic function φ_λ on $[l_B, u_B]$:

$$\inf_{x\in[l_B,u_B]} \varphi_\lambda(x). \tag{3.14}$$

Despite φ_λ is not quadratic, we prefer (3.14) to (3.12), because (3.14) has less variables.

3.3.1 The function φ_λ

As a function of y, the augmented Lagrangian (2.2) can be written

$$\ell_r(x, y, \lambda) = \frac{r}{2} \left\| A_I x + \frac{1}{r}\lambda_I - y \right\|^2 + \text{terms independent of } y.$$

Therefore the minimizer of ℓ_r with respect to $y \in [l_I, u_I]$ is the projection on $[l_I, u_I]$ of the unconstrained minimizer $A_I x + \frac{1}{r}\lambda_I$ of $\ell_r(x, \cdot, \lambda)$:

$$\breve{y}(x) := P_{[l_I,u_I]} \left(A_I x + \frac{1}{r}\lambda_I \right). \tag{3.15}$$

Note that if $(\bar{x}, \bar{\lambda})$ is a primal-dual solution to the QP (1.1), $\breve{y}(\bar{x}) = A_I \bar{x}$. Indeed
- if $A_i\bar{x} \in]l_i, u_i[$, there hold $\bar{\lambda}_i = 0$ and $P_{[l_i,u_i]}(A_i\bar{x} + \bar{\lambda}_i/r) = A_i\bar{x}$,
- if $A_i\bar{x} = l_i$, there hold $\bar{\lambda}_i \leqslant 0$ and $P_{[l_i,u_i]}(A_i\bar{x} + \bar{\lambda}_i/r) = l_i = A_i\bar{x}$,
- if $A_i\bar{x} = u_i$, there hold $\bar{\lambda}_i \geqslant 0$ and $P_{[l_i,u_i]}(A_i\bar{x} + \bar{\lambda}_i/r) = u_i = A_i\bar{x}$.

Proposition 3.1 (the convex differentiable function φ_λ) *The function φ_λ defined by (3.13) is convex and differentiable on $\mathbb{R}^n$. Its gradient at $x \in \mathbb{R}^n$ is given by*

$$\begin{aligned} \nabla\varphi_\lambda(x) &= \nabla_x \ell_r(x, \breve{y}, \lambda) \\ &= g + Hx + A_{I\cup E}^T \lambda_{I\cup E} + rA_I^T(A_I x - \breve{y}(x)) + rA_E^T(A_E x - b_E), \end{aligned} \tag{3.16}$$

where $\check{y}(x)$ is the minimizer of $\ell_r(x,\cdot,\lambda)$ given by (3.15).

PROOF. Observe first that φ_λ is convex since it is the marginal function of the convex function

$$(x,y) \mapsto \ell_r(x,y,\lambda) + \mathcal{I}_{\mathbb{R}^n \times [l_I, u_I]}(x,y),$$

where $\mathcal{I}$ is used to denote an indicator function (in plain words, $\varphi_\lambda(x)$ is obtained by minimizing this convex function in y). This property also implies that its subdifferential at x is given by (see [20])

$$\partial\varphi_\lambda(x) = \{s : (s,0) \in \partial_{(x,y)}\ell_r(x,\check{y}(x),\lambda)\}.$$

Now, since ℓ_r is differentiable, $\partial_{(x,y)}\ell_r(x,\check{y}(x),\lambda)$ is a singleton. Therefore, $\partial\varphi_\lambda(x)$ is also a singleton. This implies that φ_λ is actually a convex differentiable function with a gradient given by

$$\nabla\varphi_\lambda(x) = \nabla_x\ell_r(x,\check{y}(x),\lambda),$$

which yields (3.16). □

Proposition 3.2 (the piecewise quadratic function φ_λ) *The function φ_λ defined by (3.13) is piecewise quadratic. For any partition (I^l, I^0, I^u) of I, φ_λ is quadratic on the possibly empty convex polyhedron*

$$P_{I^l,I^0,I^u} = \left\{ x \in \mathbb{R}^n : A_{I^l}x + \frac{\lambda_{I^l}}{r} \leqslant l_{I^l},\ l_{I^0} \leqslant A_{I^0}x + \frac{\lambda_{I^0}}{r} \leqslant u_{I^0},\ u_{I^u} \leqslant A_{I^u}x + \frac{\lambda_{I^u}}{r} \right\}.$$

On this polyhedron, φ_λ is quadratic and takes at $x \in P_{I^l,I^0,I^u}$ the value

$$\begin{aligned}\varphi_\lambda(x) = g^\mathsf{T}x + \frac{1}{2}x^\mathsf{T}Hx &+ \lambda_{I^l}^\mathsf{T}(A_{I^l}x - l_{I^l}) + \frac{r}{2}\|A_{I^l}x - l_{I^l}\|^2 - \frac{1}{2r}\|\lambda_{I^0}\|^2 \\ &+ \lambda_{I^u}^\mathsf{T}(A_{I^u}x - u_{I^u}) + \frac{r}{2}\|A_{I^u}x - u_{I^u}\|^2 \\ &+ \lambda_E^\mathsf{T}(A_Ex - b_E) + \frac{r}{2}\|A_Ex - b_E\|^2.\end{aligned} \tag{3.17}$$

PROOF. For x in the convex polyhedron P_{I^l,I^0,I^u}, $\check{y}(\cdot)$ given by (3.15) is linear and therefore $\varphi_\lambda(x)$ is quadratic with a value $\varphi_\lambda(x) = \ell_r(x,\check{y}(x),\lambda)$ given by (3.17). □

3.3.2 Detecting optimality of the AL subproblem

Since the AL subproblem consists in minimizing the convex function φ_λ on $[l_B, u_B]$, optimality is reached when the projected gradient of that function vanishes. This one reads at $x \in [l_B, u_B]$:

$$\nabla^{\mathrm{P}}\varphi_\lambda(x) := P_{-\,\mathrm{T}_{\bar{x}}[l_B,u_B]}\nabla\varphi_\lambda(x) = \begin{cases} \min(0,[\nabla_x\varphi_\lambda(x)]_i) & \text{if } l_i = x_i \\ [\nabla_x\varphi_\lambda(x)]_i & \text{if } l_i < x_i < u_i \\ \max(0,[\nabla_x\varphi_\lambda(x)]_i) & \text{if } x_i = u_i, \end{cases}$$

where $\nabla_x \varphi_\lambda(x)$ is given by (3.16). The AL subproblem solver will stop when

$$\|\nabla^{\mathrm{P}} \varphi_\lambda(x)\| \leqslant \texttt{options.gtol},$$

where `options.gtol` is a user given positive tolerance.

3.3.3 Gradient projection phase

The gradient projection phase consists in forcing the decrease of φ_λ along the projected gradient path

$$p : \alpha \mapsto p(\alpha) := P_{[l_B, u_B]}\left(x - \alpha \nabla \varphi_\lambda(x)\right). \tag{3.18}$$

Figure 3.2 shows on the left hand side a typical form of the function $\alpha \mapsto \psi_\lambda(\alpha) :=$

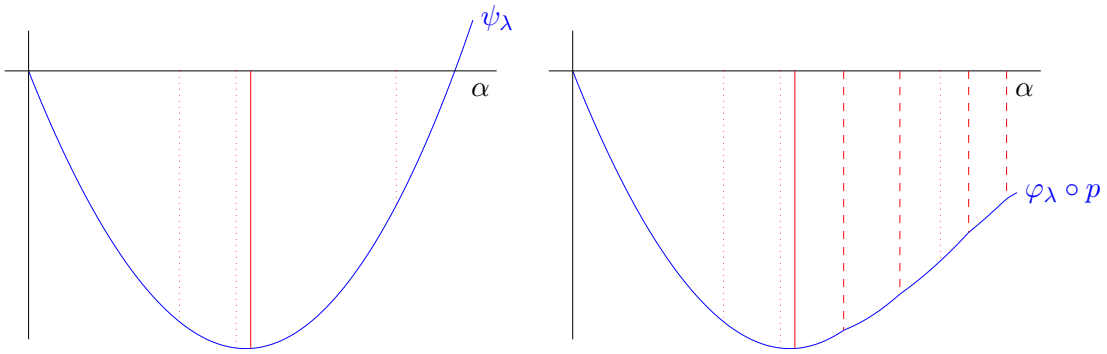

Figure 3.2: Functions ψ_λ and $\psi_\lambda \circ p$

$\varphi_\lambda(x - \alpha \nabla \varphi_\lambda(x))$. The dotted vertical lines correspond to a change in the sets I^l and/or I^u, while the plain vertical line locates the estimated minimizer $\check{\alpha}$. The change in the curvature of ψ_λ at the dotted lines is hardly visible, in particular because of the differentiability of the curve. On the other hand, the fact that $\check{\alpha}$ is not the exact minimizer of ψ_λ reflects the change in curvature of each piece of the curve ($\check{\alpha}$ is computed using the curvature of the first piece). On the right hand side of Figure 3.2, one finds the plot of the curve

$$\alpha \mapsto (\varphi_\lambda \circ p)(\alpha),$$

where p is defined by (3.18). The dashed vertical lines correspond to a change in the activity of the variable x. Here the nonconvexity of this curve appears rather clearly, when α crosses these lines.

The sufficient decrease φ_λ along p is obtained either by the Armijo rule or by the Goldstein rule (see below).

The initial curvature

The intial curvature of φ_λ at x along d can be computed from formula (3.17). This curvature is used to determine an initial stepsize for the Armijo or Goldstein linesearch rules described below. Let the index sets I^l and I^u be computed by

$$\begin{aligned} i \in I^l &\iff \Big(A_i x + \lambda_i / r < l_i\Big) \text{ or } \Big(A_i x + \lambda_i / r = l_i \text{ and } A_i d < 0\Big), \\ i \in I^u &\iff \Big(A_i x + \lambda_i / r > u_i\Big) \text{ or } \Big(A_i x + \lambda_i / r = u_i \text{ and } A_i d > 0\Big) \end{aligned}$$

and let $I^0 := I \setminus (I^l \cup I^u)$. Then, for all sufficiently small $\alpha \geqslant 0$, $\varphi_\lambda(x + \alpha d)$ is given by formula (3.17) with x replaced by $x + \alpha d$. The initial curvature of $\alpha \mapsto \varphi_\lambda(x + \alpha d)$ is therefore

$$d^\mathsf{T} \left(H + r A_{I^l \cup I^u \cup E}^\mathsf{T} A_{I^l \cup I^u \cup E} \right) d.$$

Armijo rule

In the Armijo rule, the actual stepsize $\alpha > 0$ must satisfy

$$\varphi_\lambda(p(\alpha)) \leqslant \varphi_\lambda(x) + \omega_1 \nabla \varphi_\lambda(x)^\mathsf{T} (p(\alpha) - x), \tag{3.19}$$

where the constant $\omega_1 = 0.01$ in Oqla and Qpalm. The technique used to find a stepsize α satisfying this inequality is called *backtracking*. It is described in Algorithm 3.3.

Algorithm 3.3 (Armijo's rule) *Let be given an interpolation safeguard $\tau_i \in]0, \frac{1}{2}[$ and an initial trial stepsize $\alpha > 0$.*

while *true* **do**
  if (3.19) *holds* **then**
    break
  end
  choose a new trial stepsize α in the interval $[\tau_i \alpha, (1-\tau_i)\alpha]$
end

The inconvenient of the Armijo rule is that the selected stepsize is always less than the first trial stepsize. If this one is very small, there is no means to increase it. For this reason Goldstein's rule is sometimes preferred.

Goldstein rule

In the Goldstein rule, the actual stepsize $\alpha > 0$ must satisfy

$$\varphi_\lambda(p(\alpha)) \leqslant \varphi_\lambda(x) + \gamma_1 \nabla \varphi_\lambda(x)^\mathsf{T} (p(\alpha) - x) \tag{3.20}$$

$$\varphi_\lambda(p(\alpha)) \geqslant \varphi_\lambda(x) + \gamma_2 \nabla \varphi_\lambda(x)^\mathsf{T} (p(\alpha) - x), \tag{3.21}$$

where the constants are set to $\gamma_1 = 0.01$ and $\gamma_2 = 0.99$ in Oqla and Qpalm. We use Lemaréchal's technique for determining an α satisfying these inequalities (see [2]). The implemented algorithm, described below, is taken from the exercise 6.3 in [14]:

Algorithm 3.4 (Goldstein's rule) *Let be given the constants $\underline{\alpha} := 0$, $\overline{\alpha} := +\infty$, $\tau_i \in]0, \frac{1}{2}[$, and $\tau_e > 1$. Let be given an initial trial stepsize $\alpha > 0$.*

while *true* **do**
  if (3.20) *does not hold* **then**
    set $\overline{\alpha} = \alpha$ *and choose a new trial stepsize* α *in the interval*

$$[(1-\tau_i)\underline{\alpha} + \tau_i\overline{\alpha}, \tau_i\underline{\alpha} + (1-\tau_i)\overline{\alpha}]. \tag{3.22}$$

  else
    if (3.21) *holds* **then**
      break
    else
      $\underline{\alpha} = \alpha$
      if $\overline{\alpha} = +\infty$ **then**
        Choose a new trial stepsize $\alpha \in [\tau_e\underline{\alpha}, \infty[$.
      else
        Choose a new trial stepsize α *in the interval* (3.22).
      end
    end
  end
end

This linesearch is however not recommended, since it is less likely to be successful in the presence of rounding error, in particular in the last iterations of the algorithm. The two plots in figure 3.3 try to highlight this. The figure on the left represents the function

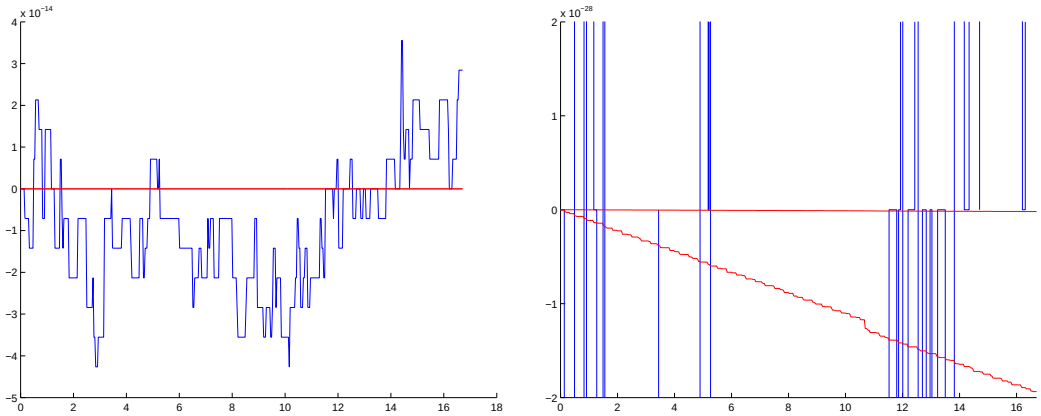

Figure 3.3: The function $\alpha \mapsto \varphi_\lambda(p(\alpha))$ (in blue) and the right hand sides of the Goldstein conditions (3.20) and (3.21) (in red), in the presence of rounding errors. The plot on the right is a zoom of the left hand side plot (the y-axis changes)

$\alpha \mapsto \varphi_\lambda(p(\alpha))$ in blue and the right hand sides of the Goldstein conditions (3.20) and (3.21) in red, in the very last iteration, which happened to be unsuccessful. The irregular behavior of the function $\varphi_\lambda \circ p$ is due to rounding errors. The figure on the right represents the same functions but with a different y-axis. One observes that very few stepsizes, if any, realize the Goldstein inequalities (3.20) and (3.21), while the Armijo inequality (3.19) is satisfied much more often.

The Moré-Toraldo criterion for exiting the GP phase

We present below two counter-examples showing that when the problem is unbounded, the Moré-Toraldo algorithm [24; 1991] may not find a direction of unboundedness, independently of the fact that the CG iterations minimize completely the function on the affine hull of the activated faces; the problem comes indeed from the CG phases of the algorithm. For papers related to the Moré-Toraldo algorithm, see [8, 9, 12, 11, 3; 1997-2014].

Counter-example 3.5 Consider the following unbounded convex QP in $x = (x_1, x_2) \in \mathbb{R}^2$:

$$\begin{cases} \inf_x \ -x_1 + \frac{1}{2}x_2^2 \\ x_2 \geqslant -1, \end{cases}$$

which consists in minimizing the convex quadratic function q subject to a bound on x_2. Then the Moré-Toraldo algorithm, starting from $x_1 = (1, -1)$, may generate the following infinite sequence of iterates x_k, for $k \geqslant 1$, defined by

$$x_k = \begin{cases} (2k-1, 1) & \text{if } k \text{ is even} \\ (2k-1, -1) & \text{if } k \text{ is odd.} \end{cases} \tag{3.23}$$

Therefore $q(x_k) = 3/2 - 2k \to -\infty$, but the algorithm does not determine a direction of unboundedness.

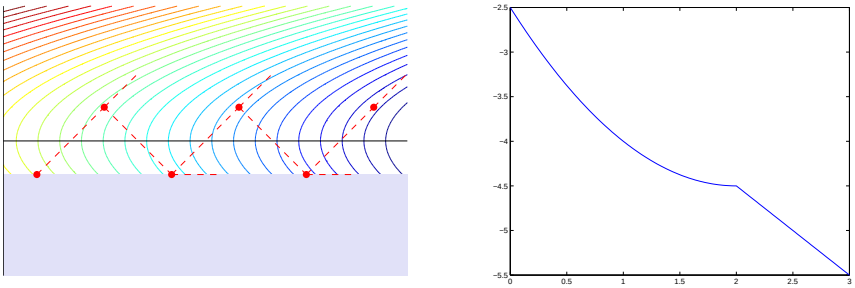

Figure 3.4: Illustration of counter-example 3.5: the iterates (left) and q along the search path from x_2 (right).

Proof. Suppose that the Moré-Toraldo algorithm starts from $x_1 = (1, -1)$ by a GP phase and that the piecewise linesearch in the GP phases try first the stepsize $\alpha_k = 2$. Then we show that all the subsequent iterations are of GP type and that (3.23) holds. We denote by P_X the orthogonal projector on the feasible set $X := \{x \in \mathbb{R}^2 : x_2 \geqslant -1\}$ and consider in sequence the cases when the current iterate x_k has an odd index k and next an even index k.

Consider first that the index k of the current iterate $x_k = (2k-1, -1)$ is odd. At this iterate the working set W_k can be set to $\{1\}$ because the bound is active. Since the gradient of q at x_k reads $g_k = (-1, -1)$, the GP step at x_k minimizes approximately by linesearch the function $\alpha \in \mathbb{R}_+ \mapsto q(P_X(x_k - \alpha g_k)) = q(P_X(2k - 1 + \alpha, -1 + \alpha)) = q(2k - 1 + \alpha, \alpha - 1) = 1 - \alpha - 2k + (\alpha - 1)^2/2$. Actually, the trial stepsize $\alpha_k = 2$ is the minimizer of this function, which makes this stepsize acceptable by any reasonable linesearch rule. Then the next iterate is indeed $x_{k+1} = (2k + 1, 1)$. Since the working set $W_{k+1} = \varnothing$ has changed and since the cost function has decreased by the significant amount 2, the algorithm may decide to pursue with a GP step at the next iteration.

Consider now an iterate $x_k = (2k-1, 1)$ with k even (see figure 3.4), at which the working set is necessarily $W_k = \varnothing$. Since the gradient of q at x_k reads $g_k = (-1, 1)$, the GP step at x_k minimizes approximately by linesearch the function $\alpha \in \mathbb{R}_+ \mapsto q(P_X(x_k - \alpha g_k)) = q(P_X(2k - 1 + \alpha, 1 - \alpha))$. This function is convex quadratic on $[0, 2]$ and affine on $[2, \infty[$ (see the right plot in figure 3.4). The trial stepsize $\alpha_k = 2$ minimizes of the quadratic part of the function, which makes this stepsize acceptable by any reasonable linesearch rule. Then the next iterate is indeed $x_{k+1} = (2k+1, -1)$. Since the working set W_{k+1} can be set to $\{1\}$ because the bound is active at x_{k+1}, it is modified by the iteration; on the other hand, the cost function has decreased by the significant amount 2; therefore the algorithm may decide to pursue with a GP step at the next iteration. □

Counter-example 3.5 is simple, but may not be convincing since at even iterate, pursuing the search along the piecewise linear path beyond the next odd iterate would detect a direction of unboundedness. For this reason, we provide another counter-example, which is a sophistication of the previous one, in which the detection of a direction of unboundedness along the piecewise linear path of the GP phase is certainly more difficult. The idea is to add a coordinate x_3 such that the second part of the search path is not along the direction of unboundedness e^1.

Counter-example 3.6 Consider the following unbounded convex QP in $x = (x_1, x_2, x_3) \in \mathbb{R}^3$:

$$\begin{cases} \inf_x \ -2^{1/2} x_1 + \frac{1}{2} x_2^2 + \frac{1}{2} x_3^2 \\ x_2 \geqslant -1, \end{cases}$$

which consists in minimizing the convex quadratic function q subject to a bound on x_2. Then the Moré-Toraldo algorithm may generate the following infinite sequence of iterates x_k, for $k \geqslant 1$, defined by

$$x_k = \begin{cases} (2^{3/2}(k-1), 1, 1) & \text{if } k \text{ is even} \\ (2^{3/2}(k-1), -1, -1) & \text{if } k \text{ is odd.} \end{cases} \tag{3.24}$$

Therefore $q(x_k) = 5 - 4k \to -\infty$ when $k \to \infty$, but the algorithm does not determine a direction of unboundedness.

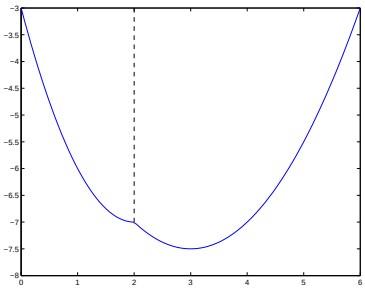

Figure 3.5: Illustration of counter-example 3.6: q along the search path from x_2.

Proof. Suppose that the Moré-Toraldo algorithm starts from $x_1 = (0, -1, -1)$ by a GP phase and that the piecewise linesearch in the GP phases tries first the stepsize $\alpha_k = 2$. Then we show that all the subsequent iterations are of GP type and that (3.24) holds. We denote by P_X the orthogonal projector on the feasible set $X := \{x \in \mathbb{R}^3 : x_2 \geqslant -1\}$ and

consider in sequence the cases when the current iterate x_k has an odd index k and next an even index k.

Consider first that the index k of the current iterate $x_k = (2^{3/2}(k-1), -1, -1)$ is odd. At this iterate the working set W_k can be set to $\{1\}$ because the bound is active. Since the gradient of q at x_k reads $g_k = (-2^{1/2}, -1, -1)$, the GP step at x_k minimizes approximately by linesearch the function $\alpha \in \mathbb{R}_+ \mapsto q(P_X(x_k - \alpha g_k)) = q(P_X(2^{3/2}(k-1) + 2^{1/2}\alpha, \alpha - 1, \alpha - 1)) = q(2^{3/2}(k-1) + 2^{1/2}\alpha, \alpha - 1, \alpha - 1) = -4(k-1) - 2\alpha + (\alpha - 1)^2$. Actually, the trial stepsize $\alpha_k = 2$ is the minimizer of this convex quadratic function, which makes this stepsize acceptable by any reasonable linesearch rule. Then the next iterate is indeed $x_{k+1} = (2^{3/2}k, 1, 1)$. Since the working set $W_{k+1} = \varnothing$ has changed and since the cost function has decreased by the significant amount 4, the algorithm may decide to pursue with a GP step at the next iteration.

Consider now an iterate $x_k = (2^{3/2}(k-1), 1, 1)$ with k even, at which the working set is necessarily $W_k = \varnothing$. Since the gradient of q at x_k reads $g_k = (-2^{1/2}, 1, 1)$, the GP step at x_k minimizes approximately by linesearch the function $\alpha \in \mathbb{R}_+ \mapsto q(P_X(x_k - \alpha g_k)) = q(P_X(2^{3/2}(k-1) + 2^{1/2}\alpha, 1 - \alpha, 1 - \alpha))$. This function is convex quadratic on $[0, 2]$ and on $[2, \infty[$ (see figure 3.5). The trial stepsize $\alpha_k = 2$ minimizes of the first quadratic piece of the function (hence can be computed by quadratic interpolation using the value, slope and curvature at $\alpha = 0$), which makes this stepsize acceptable by any reasonable linesearch rule. Then the next iterate is indeed $x_{k+1} = (2^{3/2}k, -1, -1)$. Since the working set W_{k+1} can be set to $\{1\}$ because the bound is active at x_{k+1}, it is modified by the iteration; on the other hand, the cost function has decreased by the significant amount 4; therefore the algorithm may decide to pursue with a GP step at the next iteration. □

A remedy to these counter-examples would be to stop the GP phase if it brings a too important decrease of the cost (for example a fraction or a multiple of the decrease brought by the last minimization phase). The logic of this rule is that the GP phase is not aimed at yielding an important decrease of the cost but at identifying the right active constraints; hence if the cost decreases significantly during the GP phase, it is likely that the problem is unbounded; it is then better to let the minimization phase to determine a direction of unboundedness. In particulier, the algorithm would do a single GP search during the first GP phase since the decrease obtained during the previous CG phase is zero.

3.3.4 Minimization phase

The goal of the minimization phase is to implement an *active set method* to minimize the augmented Lagrangian for (x, y) in the box $[l, u]$, problem (3.12). At each stage of this minimization process, some of the variables $(x, y)_i$ are fixed to one of the bounds l_i or u_i. The collection of these indices is called the *working set* and is denoted by

$$W \subset B \cup I \qquad \text{and} \qquad W := W^l \cup W^u,$$

where W^l (resp. W^u) is the set of indices i such that $(y, u)_i$ is fixed to l_i (resp. to u_i). We note

$$\begin{array}{lll} W_B^l := W^l \cap B, & W_B^u := W^u \cap B, & W_B := W \cap B = W_B^l \cup W_B^u, \\ W_I^l := W^l \cap I, & W_I^u := W^u \cap I, & W_I := W \cap I = W_I^l \cup W_I^u. \end{array} \tag{3.25}$$

The collection of the indices that are not in W is denoted by

$$V := (B \cup I) \setminus W.$$

We also introduce

$$V_B := V \cap B \qquad \text{and} \qquad V_I := V \cap I. \tag{3.26}$$

The working set W can be determined in many ways. What is important for the convergence of the algorithm that solves the AL subproblem is that the minimization phase decreases the augmented Lagrangian; then the convergence can be ensured by the gradient projection iterations that occur between the minimization phases []. The determination of the working set W depends on the strategy implemented. Oqla and Qpalm implement two strategies: the *hit-and-fix* strategy and the *Moré-Toraldo* strategy.

The hit-and-fix strategy

In the hit-and-fix strategy, the iterates (x, y) are maintained in the feasible set $[l, u]$. If, during the minimization phase, a bound is hit by some variables, these are fixed to the corresponding bound. The method, then proceeds with more fixed variables until a minimum is reached in the associated faces of the feasible set.

The algorithm implements a CG method in x only, trying to minimize at best the augmented Lagrangian $\ell_r(\cdot, \cdot, \lambda)$. The choice to do CG iterations only in x is motivated by the desire to reduce at most the number of iterations. On the other hand, the minimization in y is done by minimizing analytically $\ell_r(x, (\cdot, y_{W_I}), \lambda)$ on $[l_{V_I}, u_{V_I}]$. Let us give the details.

Let $(x^0, y^0) \in [l, u]$ be the iterate on entry into the minimization phase. Then the working sets W_B^0 and W_I^0 are set as follows

$$\begin{aligned} W_B^{l,0} &:= \{i \in B : x_i^0 = l_i\}, \qquad W_B^{u,0} := \{i \in B : x_i^0 = u_i\}, \qquad W_B^0 := W_B^{l,0} \cup W_B^{u,0} \\ W_I^{l,0} &:= \{i \in I : y_i^0 = l_i\}, \qquad W_I^{u,0} := \{i \in I : y_i^0 = u_i\}, \qquad W_I^0 := W_I^{l,0} \cup W_I^{u,0}. \end{aligned}$$

After this setting, it results that

$$(x^0, y^0)_V \in]l_V, u_V[,$$

so that the free variables can change a little without encountering the bounds of $[l_V, u_V]$.

Let us now specify how the problem

$$\inf_{(x,y)_V \in [l_V, u_V]} \ell_r(x, y, \lambda). \tag{3.27}$$

is approximately solved by CG iterations in x only, until a bound in x or y is hit. If there were no bound constraints in (3.27), this convex problem would be identical to finding $(x, y)_V = (x_{V_B}, y_{V_I})$ satisfying

$$\begin{aligned} &\Big(H_{V_B V_B} + r A_{(I\cup E)V_B}^{\mathsf{T}} A_{(I\cup E)V_B}\Big) x_{V_B} - r A_{V_I V_B}^{\mathsf{T}} y_{V_I} && (3.28)\\ &\quad = \quad - \Big(H_{V_B W_B} + r A_{(I\cup E)V_B}^{\mathsf{T}} A_{(I\cup E)W_B}\Big) x_{W_B} + r A_{W_I V_B}^{\mathsf{T}} y_{W_I} && (3.29)\\ &\qquad\quad - g_{V_B} - A_{(I\cup E)V_B}^{\mathsf{T}} \lambda_{I\cup E} + r A_{EV_B}^{\mathsf{T}} b_E && (3.30)\\ &A_{V_I V_B} x_{V_B} - y_{V_I} && (3.31)\\ &\quad = \quad - A_{V_I W_B} x_{W_B} - \lambda_{V_I}/r. && (3.32) \end{aligned}$$

The last equation also reads

$$y_{V_I} = A_{V_I} x + \frac{1}{r} \lambda_{V_I}. \tag{3.33}$$

Eliminating y_{V_I} in the first equation from its value given by (3.33) leads to

$$\boxed{\begin{aligned}&\Big(H_{V_BV_B} + rA^{\mathsf{T}}_{(W_I\cup E)V_B}A_{(W_I\cup E)V_B}\Big)x_{V_B}\\ &\qquad = A^{\mathsf{T}}_{(W_I\cup E)V_B}z_{W_I\cup E} - g_{V_B} - \Big(H_{V_BW_B} + rA^{\mathsf{T}}_{(W_I\cup E)V_B}A_{(W_I\cup E)W_B}\Big)x_{W_B},\end{aligned}} \tag{3.34}$$

where

$$z := r\begin{pmatrix} y \\ b_E \end{pmatrix} - \lambda_{I\cup E}. \tag{3.35}$$

The CG iterations solve (3.34) in x_{V_B}, starting from $x^0_{V_B}$, until a bound is hit in x or y (with y_{V_I} updated from x by (3.33) and y_{W_I} fixed to $y^0_{W_I}$).

A straightforward computation shows that, when $y = \check{y}(x)$, the system (3.34) is equivalent to

$$[\nabla\varphi_\lambda(x)]_{V_B} = 0 \qquad \text{or} \qquad [\nabla_x\ell_r(x,\check{y}(x),\lambda)]_{V_B} = 0, \tag{3.36}$$

where $\nabla\varphi_\lambda(x)$ is given by (3.16).

Let us now give an interpretation of the procedure. Proceeding like in the definition (3.15) of $\check{y}(x)$, one can write

$$\ell_r(x,y',\lambda) = \frac{r}{2}\left\|A_{V_I}x + \frac{1}{r}\lambda_{V_I} - y'_{V_I}\right\|^2 + \text{terms independent of } y'_{V_I}.$$

Therefore

$$\ell_r(x,y,\lambda) \leqslant \inf_{\substack{y'_{V_I}\in[l_{V_I},u_{V_I}]\\ y'_{W_I}=y^0_{W_I}}} \ell_r(x,y',\lambda) \qquad \text{and} \qquad y_{V_I} = (\check{y}(x))_{V_I},$$

as long as y_{V_I} has not hit one of the bounds l_{V_I} or u_{V_I}.

Note that the generated y may differ from $\check{y}(x)$ since it may occur that $y_{W_I} \neq (\check{y}(x))_{W_I}$. Therefore setting $y = \check{y}(x)$, like on entry of the GP phase, may inactivate some inequality bounds.

Proposition 3.7 below assumes that the CG phases end up by minimizing q completely on a certain face. This is the meaning of the word "exact".

Proposition 3.7 *The exact hit-and-fix strategy finds either a direction of unboundedness of the QP or a solution to the augmented Lagrangian subproblems.*

Proof. We split the proof in two cases.

Suppose first that the closest feasible QP is bounded. Then, it is known that the AL subproblems are also bounded [4; proposition 2.5]. Then each CG phase finds a minimum of the AL on a particular face. If that particular minimum is not a minimizer of the AL, the GP phase that follows desactivate the face and this one is never again visited by the algorithm (since the AL decreases strictly at each GP phase). Since there are a finite number of faces, the hit-and-fix algorithm ends up by finding the minimum of the AL on $[l,u]$.

Suppose now that the closest feasible QP is unbounded. Then, by [4; proposition 2.5], the first AL subproblem is also unbounded. We claim that, in this case, the hit-and-fix algorithm ends up by finding an unboundedness direction of the AL on $[l,u]$, which

a direction $(d_x, d_y) = (d, A_I d)$ where d is a direction of unboundedness of the QP (see remark 2.3), so that the proof will be concluded. The faces of $[l, u]$ can be partitionend in two sets: a set $\mathcal{F}_b$ of faces on which the AL is bounded and a set $\mathcal{F}_u$ of faces on wich the AL is unbounded. Since, once the CG algorithm has found a minimum on a face in $\mathcal{F}_b$, this face is no longer visited by the iterates, and since the number of faces is finite, one CG phase ends up by exploring a face $F \in \mathcal{F}_u$. Since the CG iterations minimize the AL on affine subspaces included in aff F of strictly increasing dimension and since the AL is unbounded on F, it eventually constructs a direction d of unboundedness of the AL on F. □

The Moré-Toraldo strategy

The Moré-Toraldo strategy for the minimization phase in Oqla and Qpalm, called that way because it is largely inspired from the one proposed in [24], can be described in vague terms as follows. The strategy also aims at solving problem (3.27), but the CG iterations are not stopped as soon as an iterate crosses the boundaty of $[l_V, u_V]$, like in the hit-and-fix strategy. Rather, the CG iterations pursue the minimization of the AL on the affine hull of the activated face and are interrupted when the following criterion is satisfied:

$$T_1 \quad \text{and} \quad (T_2 \quad \text{or} \quad T_3), \tag{3.37}$$

where the tests T_i can be expressed in vague terms as follows

- $T_1 =$ "ℓ_r no longer decreases sufficiently" (this includes the case when a minimizer has been obtained),
- $T_2 =$ "the current iterate is outside $[l_V, u_V]$",
- $T_3 =$ "the current iterate is inside $[l_V, u_V]$ but the activated face does not look like to be the optimal one".

We make these expressions concrete below. When T_1 and T_2 are satisfied, the current iterate is outside $[l_V, u_V]$ and a projected search ensures that the final iterate of the minimization phase is in $[l, u]$ and has forced the decrease of the AL. When T_1 and T_3 are satisfied, these conclusions are satisfied by the CG iterations without the need of a projection search. We now decribe with more precision the various ingredients defining the Moré-Toraldo strategy.

By convexity, solving problem (3.27) is equivalent to solving its optimality conditions (3.29)-(3.32), from which y_{V_I} can be eliminated using the second equation or (3.33). Therefore, the CG iterations are generated in order to solve the linear system (3.34) in x_{V_B}, like in the hit-and-fix strategy, except that in the Moré-Toraldo strategy the CG iterations do not stop when an iterate is generated outside the box $[l_{V_B}, u_{V_B}]$; the test (3.37) is used instead. Solving the linear system (3.34) in x_{V_B} is equivalent to minimizing the AL (2.2) completely in y_{V_I}, which yields the value

$$y_{V_I} = A_{V_I} x + \lambda_{V_I}/r, \tag{3.38}$$

and then to minimizing in x_{V_B} the resulting function

$$x \mapsto \tilde{\varphi}_\lambda(x) = \ell_r\Big(x, (A_{V_I} x + \lambda_{V_I}/r, y_{W_I}), \lambda\Big). \tag{3.39}$$

Let the CG iterates be denoted by x_j, $j \in \mathbb{N}$ (we have dropped the index showing the dependence of the sequence on the phase).

There are several possibilities to implement the test T_1, all invariant. It can rest on the smallness of the residual of the linear system (3.34) or on the smallness of the decrease in $\tilde{\varphi}_\lambda$ realized at the current iteration with respect to those obtained in the previous iterations. Let us be more precise. The first possibility reads

$$T_1' \iff \|r_j\| \leqslant \varepsilon_r \|r_0\|,$$

where r_j is the residual of (3.34) at the current iteration j and $\varepsilon_r \in]0,1[$. This test is standard but it suffers from the erratic behavior of the norm of the residual $\|r_j\|$, as a function of the iteration index j (a benign default according to us), and the lack of universal threshold ε_r (more annoying). The second possibility, used in [24], reads

$$T_1'' \iff \begin{cases} \tilde{\varphi}_\lambda(x_0) - \tilde{\varphi}_\lambda(x_1) \leqslant 10\,\varepsilon_m\,|\tilde{\varphi}_\lambda(x_0)| & \text{if } j = 1 \\ \tilde{\varphi}_\lambda(x_{j-1}) - \tilde{\varphi}_\lambda(x_j) \leqslant \varepsilon_\ell \max_{1 \leqslant j' < j} \big(\tilde{\varphi}_\lambda(x_{j'-1}) - \tilde{\varphi}_\lambda(x_{j'})\big) & \text{if } j > 1, \end{cases}$$

where ε_m is the machine epsilon and $\varepsilon_\ell \in]0,1[$. Hence, it consists in testing whether the decrease in $\tilde{\varphi}_\lambda$ obtained during the last CG iteration is not too small with respect to the best one obtained in the previous iterations. The parameter ε_ℓ can be chosen rather independently of the problem to solve. The defect of this technique comes from the well known fact that the decrease of $\tilde{\varphi}_\lambda$ can be arbitrary [16], so that the interruption of the CG iterations may occur prematurely.

The test T_2 is clear:

$$T_2 \iff (x_j, y_j) \notin [l, u].$$

The test T_3 makes use of the gradient of $\tilde{\varphi}_\lambda$, which reads

$$\begin{aligned} \nabla\tilde{\varphi}_\lambda(x) &= \nabla_x \ell_r(x, A_I x + \lambda_I/r, \lambda) + \begin{pmatrix} A_I \\ 0_{W_I} \end{pmatrix}^{\mathsf{T}} \nabla_y \ell_r(x, A_I x + \lambda_I/r, \lambda) \\ &= g + Hx + A_I^{\mathsf{T}}\big[\lambda_I + r(A_I x - y)\big] + A_E^{\mathsf{T}}\big[\lambda_E + r(A_E x - b_E)\big] \\ &\quad - \begin{pmatrix} A_I \\ 0_{W_I} \end{pmatrix}^{\mathsf{T}} \Big(\lambda_I + r(A_I x - y)\Big) \\ &= g + Hx + A_{W_I}^{\mathsf{T}}\big[\lambda_{W_I} + r(A_{W_I} x - y_{W_I})\big] + A_E^{\mathsf{T}}\big[\lambda_E + r(A_E x - b_E)\big]. \end{aligned} \tag{3.40}$$

Note that the linear system (3.34) is equivalent to $[\nabla\tilde{\varphi}_\lambda(x)]_{V_B} = 0$. The projected gradient $\nabla^{\mathrm{P}}\tilde{\varphi}_\lambda(x)$ has its ith component given by

$$[\nabla^{\mathrm{P}}\tilde{\varphi}_\lambda(x)]_i = \begin{cases} \min(0, [\nabla\tilde{\varphi}_\lambda(x)]_i) & \text{if } l_i = x_i \\ [\nabla\tilde{\varphi}_\lambda(x)]_i & \text{if } l_i < x_i < u_i \\ \max(0, [\nabla\tilde{\varphi}_\lambda(x)]_i) & \text{if } \quad x_i = u_i. \end{cases}$$

The test T_3 is inspired by Rosen's stopping criterion [26, 13]:

$$T_3 \iff \begin{cases} (x_j, y_j) \in [l, u] \\ \Big\|[\nabla^{\mathrm{P}}\tilde{\varphi}_\lambda(x)]_{W_B}\Big\| > \gamma_R \Big\|[\nabla\tilde{\varphi}_\lambda(x)]_{V_B}\Big\|, \end{cases}$$

where $\gamma_R \geqslant 0$ is called the Rosen constant. In other words, the test T_3 is active if the iterate (x_j, y_j) is feasible and if the norm of the W_B-components of $\nabla\tilde{\varphi}_\lambda(x)$ with the wrong sign is

too large with respect to the norm of the V_B-components of $\nabla\tilde{\varphi}_\lambda(x)$, that the current CG iterations try to vanish. The test T_3 is more permissive (i.e., more tolerent or less often active) than the Moré-Toraldo test [24], which is recovered from T_3 by taking $\gamma_R = 0$. Now, the Moré-Toraldo strategy is slightly more expensive, since the CG iterations must also update the W_B-components of $\nabla\tilde{\varphi}_\lambda(x_j)$ in order to be able to check T_3; this computation is not necessary in the hit-and-fix strategy.

When both T_1 and T_2 are satisfied at the CG iterate $z_j := (x_j, y_j)$, this one is outside the box $[l, u]$ and a projected search is triggered to ensure that the final iterate $z := (x, y)$ of the minimization phase is in $[l, u]$ and has sufficiently decreased the AL with respect to its value $\ell_r(x_0, y_0, \lambda)$ at the initial iterate $z_0 := (x_0, y_0)$ of the minimization phase. More concretely, the projection search determines $z = p(\alpha)$ along the piecewise linear path

$$\alpha \in [0, 1] \mapsto p(\alpha) := P_{[l,u]}(z_0 + \alpha(z_j - z_0))$$

such that

$$\ell_r(x, y, \lambda) \leqslant \ell_r(x_0, y_0, \lambda) + \omega_1 \nabla_{(x,y)}\ell_r(x_0, y_0, \lambda)^T(p(\alpha) - x_0). \tag{3.41}$$

Since $(z_j)_W = (z_0)_W \in [l_W, u_W]$, it follows that $p(\alpha)_W = (z_0)_W$ and only the V-components of $\nabla_{(x,y)}\ell_r(x_0, y_0, \lambda)$ must be computed to verify inequality (3.41). These can be obtained from the already computed gradient $\nabla\tilde{\varphi}_\lambda(x_0)$, since by (3.38) and (3.40):

$$\begin{aligned}
\nabla_{x_{V_B}}\ell_r(x, y, \lambda) &= \Big(g + Hx + A_I^T\big[\lambda_I + r(A_Ix - y)\big] + A_E^T\big[\lambda_E + r(A_Ex - b_E)\big]\Big)_{V_B} \\
&= \Big(g + Hx + A_{W_I}^T\big[\lambda_{W_I} + r(A_{W_I}x - y_{W_I})\big] + A_E^T\big[\lambda_E + r(A_Ex - b_E)\big]\Big)_{V_B} \\
&= \nabla_{x_{V_B}}\tilde{\varphi}_\lambda(x),
\end{aligned}$$

and by (3.38) again:

$$\nabla_{y_{V_I}}\ell_r(x, y, \lambda) = -\big[\lambda_I + r(A_Ix - y)\big]_{V_I} = 0.$$

As a result, the Armijo test (3.41) becomes

$$\ell_r(x, y, \lambda) \leqslant \ell_r(x_0, y_0, \lambda) + \omega_1 \nabla_{x_{V_B}}\tilde{\varphi}_\lambda(x_0)^T(p(\alpha) - x_0)_{V_B}. \tag{3.42}$$

This test is satisfied for $\alpha > 0$ sufficienty small since, first, for small $\alpha > 0$, $z_0 + \alpha(z_j - z_0) \in [l, u]$, so that $p(\alpha) - x_0 = \alpha(z_j - z_0)$, and, second, $(z_j - z_0)_{V_B} = (x_j - x_0)_V$ is a descent direction of $\tilde{\varphi}_\lambda$ at x_0 (use the convexity of $\tilde{\varphi}_\lambda$ and $\tilde{\varphi}_\lambda(x_j) < \tilde{\varphi}_\lambda(x_0)$ due to the CG iterations).

3.3.5 Preconditioning of the CG iterations

We address in this section the question of the preconditioning of the linear system (3.34). Let us denote the matrix of that system by

$$\begin{aligned}
M \equiv M(V_B, W_I) &= H_{V_BV_B} + rA_{(W_I\cup E)V_B}^T A_{(W_I\cup E)V_B}, \\
&= H_{V_BV_B} + rA_{EV_B}^T A_{EV_B} + r\sum_{i\in W_I} A_{iV_B}^T A_{iV_B}.
\end{aligned} \tag{3.43}$$

Two difficulties arise in realizing a preconditioner of $M(V_B, W_I)$, due to the fact that V_B and/or W_I change at each new CG phase: the order of the matrix is decreased when an x-bound is hit and a row of A is added when an y-bound is hit.

Diagonal preconditioner

One takes the inverse of $\mathrm{Diag}(M)$, which is easy to form and to update.

- When an x-bound is hit, the preconditioner has simply one fewer diagonal element.
- When an y-bound is hit, say with index $i \in I$, one adds rA_{ij}^2 to all the elements $j \in V_B$ of the diagonal preconditioner.

Singularity occurs when some diagonal elements of case, one can determine a candidate for a direction of unboundedness. This topics is explored in section 3.4.3.

A diagonal preconditioner is rather easy to construct, but is not always very efficient since it does not consider an important part of the matrix M.

Cholesky preconditioner

When $M \succcurlyeq 0$, there is a lower triangular matrix L and a permutation matrix Q such that

$$QMQ^\mathsf{T} = LL^\mathsf{T}. \tag{3.44}$$

See for example Higham [18; 2002, théorème 10.9]. The matrix L can be written

$$L = \begin{pmatrix} L_{11} & 0 \\ L_{21} & 0 \end{pmatrix},$$

where L_{11} is lower triangular. The rank of L is that of L_{11}, which is therefore the number of columns of L_{11}:

$$\mathrm{rank}(M) = \mathrm{rank}(L_{11}).$$

When M is positive definite, one can take $Q = I$ in (3.44) and the factor L is non-singular. Let us see how to proceed. The Cholesky factorization of M without pivoting presented by Higham [18; 2002, section 10.3] is the following. If the factorization (3.44) exists with $Q = I$, one can write

$$M = LL^\mathsf{T} = \sum_{j=1}^{n} L_{\bullet j} L_{\bullet j}^\mathsf{T},$$

where L is a lower triangular matrix, with a positive diagonal, and $L_{\bullet j}$ denotes the jth column of L. Then, for $k = 1, \ldots, n+1$, let us introduce

$$M^{(k)} := M - \sum_{j=1}^{k-1} L_{\bullet j} L_{\bullet j}^\mathsf{T} = \sum_{j=k}^{n} L_{\bullet j} L_{\bullet j}^\mathsf{T} = \begin{pmatrix} 0_{k-1,k-1} & 0_{k-1,n-k+1} \\ 0_{n-k+1,k-1} & B^{(k)} \end{pmatrix}, \tag{3.45}$$

where $B^{(k)}$ is of order $n-k+1$ and is positive definite. Hence $M^{(1)} = M$ and $M^{(n+1)} = 0$. The columns $L_{\bullet j}$, $j = 1, \ldots, n$, can be built recurcively: step j consists in zeroing the column and row j of the resulting matrix in the right hand side of (3.45). For instance, $L_{\bullet 1}$ is made by

$$M = \begin{pmatrix} \sqrt{M_{11}} \\ M_{2:n,1}/\sqrt{M_{11}} \end{pmatrix} \begin{pmatrix} \sqrt{M_{11}} \\ M_{2:n,1}/\sqrt{M_{11}} \end{pmatrix}^\mathsf{T} + \begin{pmatrix} 0_{1,1} & 0_{1,2:n} \\ 0_{2:n,1} & M_{2:n,2:n} - M_{2:n,1} M_{2:n,1}^\mathsf{T}/M_{11} \end{pmatrix},$$

so that one must take

$$\begin{aligned} L_{11} &= \sqrt{M_{11}}, \\ L_{2:n,1} &= M_{2:n,1}/L_{11}, \\ B^{(1)} &= M_{2:n,2:n} - L_{2:n,1} L_{2:n,1}^\mathsf{T}. \end{aligned}$$

These considerations yield the following algorithm (in pseudo-Matlab notation), in which $B^{(k)}$ is stored in $M_{k:n,k:n}$.

Algorithm 3.8 (Cholesky) On input: a matrix `M` of order `n`. On return: its Cholesky factor `L`.

```
for k = 1:n
  if M(k,k) <= 0; stop; end                                    (*)
  L(k,k) = sqrt(M(k,k));                                       (**)
  L(k+1:n,k) = M(k+1:n,k)/L(k,k);
  M(k+1:n,k+1:n) = M(k+1:n,k+1:n) - L(k+1:n,k)*L(k+1:n,k)';
end
```

At `(*)`, the algorithm detects that the matrix M is not positive definite and stops. In Lapack [1; 1999], the subroutines Blas-2 and Blas-3 compute the Cholesky factorization of a positive definite matrix without pivoting .

The linear system (3.34) can then be preconditioned by $P = M^{-1} = L^{-\mathsf{T}}L^{-1}$. When applied to a vector, such a preconditioning can then be realized by solving two triangular linear systems. This is somehow a perfect preconditioner, since the linear system (3.34) is then solved by a single CG iteration in exact arithmetics; in floating point arithmetics more than one iteration could be necessary; these iterations can then be viewed as a way of realizing iterative refinement.

We consider now the case when the factorization needs updating since an x or y bound has been hit. If the kth bound in x is hit, formula (3.43) shows that one has to compute the Cholesky factorization of the matrix $\tilde{M} \equiv M(V_B \setminus \{k\}, W_I)$, which is obtained from M by deleting its kth row and column. The matrices M and $\tilde{M}$ are related by

$$M = \begin{pmatrix} \tilde{M}_{11} & \vdots & \tilde{M}_{21}^{\mathsf{T}} \\ \cdots & \cdot & \cdots \\ \tilde{M}_{21} & \vdots & \tilde{M}_{22} \end{pmatrix},$$

where the dots correspond to kth row and column. Let us write $\tilde{M} = \tilde{L}\tilde{L}^{\mathsf{T}}$ the Cholesky factorization of $\tilde{M}$ and let us partition L and $\tilde{L}$ as follows

$$L = \begin{pmatrix} L_{11} & 0 & 0 \\ \cdots & \cdot & 0 \\ L_{21} & v & L_{22} \end{pmatrix} \qquad \text{and} \qquad \tilde{L} = \begin{pmatrix} \tilde{L}_{11} & 0 \\ \tilde{L}_{21} & \tilde{L}_{22} \end{pmatrix}.$$

A straighforward computation yields

$$\begin{aligned} \tilde{M}_{11} &= L_{11}L_{11}^{\mathsf{T}} = \tilde{L}_{11}\tilde{L}_{11}^{\mathsf{T}}, \\ \tilde{M}_{21} &= L_{21}L_{11}^{\mathsf{T}} = \tilde{L}_{21}\tilde{L}_{11}^{\mathsf{T}}, \\ \tilde{M}_{22} &= L_{21}L_{21}^{\mathsf{T}} + vv^{\mathsf{T}} + L_{22}L_{22}^{\mathsf{T}} = \tilde{L}_{21}\tilde{L}_{21}^{\mathsf{T}} + \tilde{L}_{22}\tilde{L}_{22}^{\mathsf{T}}. \end{aligned}$$

The uniqueness of the Cholesky factorization and the nonsingularity of L_{11} show that there must hold:

$$\tilde{L}_{11} = L_{11}, \quad \tilde{L}_{21} = L_{21}, \quad \text{and} \quad \tilde{L}_{22}\tilde{L}_{22}^{\mathsf{T}} = L_{22}L_{22}^{\mathsf{T}} + vv^{\mathsf{T}}.$$

Therefore, $\tilde{L}_{22}\tilde{L}_{22}^{\mathsf{T}}$ is the Cholesky factorization of a rank one correction of $L_{22}L_{22}^{\mathsf{T}}$. It can be obtained by solving a single linear system with the matrix L_{22}.

If the kth bound in y is hit, formula (3.43) shows that one has to compute the Cholesky factorization of the matrix $\tilde{M} \equiv M(V_B, W_I \cup \{k\})$, which is obtained from M by adding a rank one matrix:

$$\tilde{M} = M + r A_{kV_B}^{\mathsf{T}} A_{kV_B}.$$

When the Cholesky factorization of M is known and M is nonsingular, the update of the Cholesky factor of M to get the one of $\tilde{M}$ is a standard operation. In Matlab, it can be done with the function `cholupdate`.

When M is positive semidefinite, but singular, algorithm 3.8 usually fails since M_{kk} in `(**)` can vanish (or even be negative because of rounding errors) and a division by zero occurs in the next statement. Furthermore, the pivoting matrix Q is necessary to get (3.44). A pivoting operation consists in permuting the rows/columns k and $l_k \in [k:n]$ of $M^{(k)}$, where

$$l_k := \min\left(\arg\max\left\{M_{ii}^{(k)} : i \in [k:n]\right\}\right).$$

If Q^{kl_k} denotes the (symmetric) matrix permuting the rows/columns k and l_k, there holds

$$\begin{aligned} Q^{kl_k} M^{(k)} (Q^{kl_k})^{\mathsf{T}} &:= Q^{kl_k} M (Q^{kl_k})^{\mathsf{T}} - \sum_{j=1}^{k-1} (Q^{kl_k} L_{\bullet j})(Q^{kl_k} L_{\bullet j})^{\mathsf{T}} \\ &= \sum_{j=k}^{n} (Q^{kl_k} L_{\bullet j})(Q^{kl_k} L_{\bullet j})^{\mathsf{T}} \\ &= Q^{kl_k} \begin{pmatrix} 0_{k-1,k-1} & 0_{k-1,n-k+1} \\ 0_{n-k+1,k-1} & B^{(k)} \end{pmatrix} (Q^{kl_k})^{\mathsf{T}}. \end{aligned}$$

This calculus shows that each permutation acts on the rows k and l_k of $L_{\bullet j}$ and on the columns/rows k and l_k of $B^{(k)}$. The goal of this operation is to get $(Q^{kl_k} M^{(k)} (Q^{kl_k})^{\mathsf{T}})_{kk} > 0$. If this is not possible (up to a given precision), it means either that $B^{(k)} = 0$ (hence M is of rank $k-1$) or that M is not positive semidefinite.

It is now necessary to specify a stopping criterion, allowing the algorithm to estimate the rank of M and to detect its possible indefiniteness. In the code `Xchdc` of Linpack [7; 1979], the test at stage k consists in detecting the nonpositivity of the pivot:

$$B_{l_k l_k}^{(k)} \leqslant 0.$$

Hammarling, Higham, and Lucas [17; 2007] (see also [21, 22; 2004]) suggest to take

$$B_{l_k l_k}^{(k)} \leqslant n\varepsilon\, B_{l_1 l_1}^{(1)},$$

where $\varepsilon_m > 0$ is the machine epsilon. Nothing is said for the stopping test at stage 1, by examining $M = B^{(1)}$.

This yields the following algorithm.

Algorithm 3.9 (Cholesky with pivoting) On input: a matrix `M` of order `n`. On

return: the permutation matrix `Q` and the Cholesky factor `L` of `Q*M*Q'`.

```
for k = 1:n
  determine the index l of the pivot in [k:n];
  stop if M(l,l) is too small;
  pivot the rows k and l in L(:,1:k-1);
  pivot the rows and columns k and l in M(k:n,k:n);
  L(k,k) = sqrt(M(k,k));
  L(k+1:n,k) = M(k+1:n,k)/L(k,k);
  M(k+1:n,k+1:n) = M(k+1:n,k+1:n) - L(k+1:n,k)*L(k+1:n,k)';
end
```

When M is singlar, the factor L is rank deficient. The factorization is then used to compute a candidate for a direction of unboundedness, a topics that is explored in section 3.4.3.

Cholesky preconditioner update

We now consider the case when the singular Cholesky factorization needs updating since an x or y bound has been hit.

If the kth bound in x is hit, formula (3.43) shows that one has to compute the (possibly singular) Cholesky factorization of the matrix $\tilde{M} \equiv M(V_B \setminus \{k\}, W_I)$, which is obtained from $M = Q^T L L^T Q$ by deleting its kth row/column. Let us denote by l the single index determined by the condition $Q_{lk} = 1$. Then we have to remove row l and column k of Q, and to remove row/column l of LL^T. Indeed, if we denote by E_k the identity matrix from which row k has been removed, $E_k M$ removes the row k of M and ME_k^T removes the column k of M. Therefore, the desired matrix is $E_k Q^T L L^T Q E_k^T$. Since the column l of $E_k Q^T$ vanishes (by definition of l), there holds $E_k Q^T = E_k Q^T E_l^T E_l$ and the desired matrix is $E_k Q^T L L^T Q E_k^T = (E_k Q^T E_l^T)(E_l L L^T E_l^T)(E_l Q E_k^T)$. The product $E_l Q E_k^T$ means that one has to remove the row l and column k of Q, while the product $E_l L L^T E_l^T$ means that one has to remove the row/column l of LL^T.

Removing the row l and the column k of Q makes no difficulty.

Let us look at the Cholesky factorization $\tilde{L}\tilde{L}^T$ of $E_l L L^T E_l^T$ (remind that L may be rank deficient). We denote by r the rank of M and examine two complementary cases.

- *Case when $l \leqslant r$.* Then the matrices L and $\tilde{L}$ can be written

$$L = \begin{pmatrix} L_{11} & 0 & 0 \\ \cdots & \times & 0 \\ L_{21} & v & L_{22} \end{pmatrix} \qquad \text{and} \qquad \tilde{L} = \begin{pmatrix} \tilde{L}_{11} & 0 \\ \tilde{L}_{21} & \tilde{L}_{22} \end{pmatrix},$$

  where the dots in L lie on row l, L_{11} is the NW principal submatrix of L of order $l-1$ (hence lower triangular and nonsingular), $\times$ is a nonzero scalar, v is a vector, and L_{22} is a lower triangular matrix, which may be rank deficient (L_{22} is absent or zero when $l = r$). An identification of the corresponding blocks of $E_l L L^T E_l^T = \tilde{L}\tilde{L}^T$ yields

$$\begin{aligned} L_{11} L_{11}^T &= \tilde{L}_{11} \tilde{L}_{11}^T, \\ L_{21} L_{11}^T &= \tilde{L}_{21} \tilde{L}_{11}^T, \\ L_{21} L_{21}^T + vv^T + L_{22} L_{22}^T &= \tilde{L}_{21} \tilde{L}_{21}^T + \tilde{L}_{22} \tilde{L}_{22}^T. \end{aligned}$$

Since L_{11} is nonsingular, the uniqueness of the nonsingular Cholesky factorization implies that $\tilde{L}_{11} = L_{11}$. Next it follows that

$$\tilde{L}_{11} = L_{11}, \quad \tilde{L}_{21} = L_{21}, \quad \text{and} \quad \tilde{L}_{22}\tilde{L}_{22}^T = L_{22}L_{22}^T + vv^T. \tag{3.46}$$

Hence $\tilde{L}_{22}$ is the (possibly singular) Cholesky factor of $L_{22}L_{22}^T + vv^T$.

- *Case when $l = r$.* If L_{22} is absent (i.e., $l = r = |V_B|$), the last condition in (3.46) is also absent. Otherwise, $L_{22} = 0$ and

$$\tilde{L}_{22}\tilde{L}_{22}^T = vv^T.$$

  - If $v = 0$, then $\tilde{L}_{22} = 0$ and $\tilde{L}$ is the rank $r-1$ matrix

$$\tilde{L} = E_l L E_l^T.$$

  - If $v \neq 0$, $\tilde{L}_{22}$ is the rank one Cholesky factor of vv^T. This one is obtained by some pivoting matrix $\tilde{Q}_{22}$ putting the largest element of v ahead: $\tilde{L}_{22} = \tilde{Q}_{22}v$. Introducing the permutation matrix $\tilde{Q} := \mathrm{Diag}(I_{l-1}, \tilde{Q}_{22})$, we obtain for $\tilde{L}$ the rank r matrix

$$\tilde{L} = \tilde{Q}E_l L E_{l+1}^T.$$

- *Case when $l < r$.* Then the last condition in (3.46) can be written

$$\tilde{L}_{22}\tilde{L}_{22}^T = \begin{pmatrix} v & L_{22} \end{pmatrix} \begin{pmatrix} v & L_{22} \end{pmatrix}^T.$$

  We use Givens rotations to restore triangularity of the factors [19; 2008]. These form an orthogonal matrix G such that

$$\begin{pmatrix} v & L_{22} \end{pmatrix} G^T = \tilde{L}_{22}.$$

- *Case when $l > r$.* Then the matrices L and $\tilde{L}$ can be written

$$L = \begin{pmatrix} L_{11} & 0 & 0 \\ \cdots & \cdot & 0 \\ L_{21} & 0 & 0 \end{pmatrix} \qquad \text{and} \qquad \tilde{L} = \begin{pmatrix} \tilde{L}_{11} & 0 \\ \tilde{L}_{21} & \tilde{L}_{22} \end{pmatrix},$$

  where the dots in L lie on row l and L_{11} is the NW principal submatrix of L of order $l-1$ (hence lower triangular, but possibly singular). An identification of the corresponding blocks of LL^T and $\tilde{L}\tilde{L}^T$ (the matrix LL^T minus its row/column l) yield

$$\begin{aligned} L_{11}L_{11}^T &= \tilde{L}_{11}\tilde{L}_{11}^T, \\ L_{21}L_{11}^T &= \tilde{L}_{21}\tilde{L}_{11}^T, \\ L_{21}L_{21}^T &= \tilde{L}_{21}\tilde{L}_{21}^T + \tilde{L}_{22}\tilde{L}_{22}^T. \end{aligned}$$

  These identities are satisfied by taking

$$\tilde{L}_{11} = L_{11}, \quad \tilde{L}_{21} = L_{21}, \quad \text{and} \quad \tilde{L}_{22} = 0.$$

  Hence, in this case, the rank of the updated Cholesky factor $\tilde{L}$ is still r (it cannot be larger than r, as a principal submatrix of a rank r matrix, and $\operatorname{rank} L_{11} = r$).

If the kth bound in y is hit, formula (3.43) shows that one has to compute the Cholesky factorization of the matrix $\tilde{M} \equiv M(V_B, W_I \cup \{k\})$, which is obtained from M by adding a rank one matrix:

$$\tilde{M} = M + r A_{kV_B}^T A_{kV_B}.$$

Now, $M = Q^T L L^T Q$, so that

$$\tilde{M} = Q^T \left(LL^T + r \left(QA_{kV_B}^T \right) \left(QA_{kV_B}^T \right)^T \right) Q.$$

Therefore, it is required to compute the Cholesky factorization of the rank 1 correction of LL^T by the matrix vv^T, where $v = r^{1/2} Q A_{kV_B}^T$. This factor update may need a permutation matrix Q_k, so that

$$LL^T + r \left(QA_{kV_B}^T \right) \left(QA_{kV_B}^T \right)^T = Q_k^T \tilde{L} \tilde{L}^T Q_k.$$

Denoting $\tilde{Q} = Q_k Q$, we obtain the new factorisation

$$\tilde{M} = \tilde{Q}^T \tilde{L} \tilde{L}^T \tilde{Q}.$$

Special features in Oqla

Special features in Qpalm

In Qpalm, when Cholesky preconditioning is required, a standard Cholesky factorization of M is first tried, using the Matlab function `chol`. If this one fails (usually because M is rank deficient), a second attempt is made using the function `qpalm_cholp`. The latter is a self-made Matlab (hence slower) version of the algorithm presented in [18]. It uses pivoting and can deal with rank deficiency; in particular, it returns a pivoting matrix Q such that $QMQ^T = LL^T$, where L may also be rank deficient.

In Qpalm `0.3`, a new Cholesky factorization is done after each GP phase.

This strategy is very efficient in terms of CG iterations, but is time consuming.

Other preconditioners

References to look at: Nocedal et Wright [25; 2006], Domorádová and Dostál [6; 2007], la section 5.10 chez Dostál [10; 2009], Dostál, Domorádová and Sadowská [11; 2011]. Check also what has been done in QPA.

3.3.6 Rosen's criterion

The AL subproblem (2.3) is solved by minimizing completely its objective on a sequence of faces of the feasible set $[l, u]$. When the currently explored face is likely not to be the one that is active at the solution, it looks reasonable to leave it as soon as this observation is done. The goal of the Rosen criterion is precisely to determine whether the current face is likely to be the optmal one. It does so by comparing

3.3.7 The Moré-Toraldo strategy

In [24; 1991], Moré and Toraldo propose a method for solving a large scale optimization problem, consisting in minimizing a strictly convex function subject to bounds on the variables. We have adapted this method for the minimization of the augmented Lagrangian subproblems, which differs from the problem considered by Moré and Toraldo on two aspects: the quadratic function is only convex (it may have a direction of unboundedness) and the minimization is in a pair of variables (x, y), with a straightforward minimization in y that is done analytically.

The algorithm of Moré and Toraldo can be viewed in several manners. We would like to present it has a succession of three phases repeated up to convergence: the conjugate gradient (CG) minimization phase, a direction projection (DP) phase, and a gradient projection (GP) phase. Moré and Toraldo consider the CG and DP phases as a single phase, but we prefer slitting it since it is also used elsewhere in the code. More precisely, modifications of these three phases will allow us to consider several algorithms that will be compared, with respect to their efficiency.

The CG phase

At the current point $z = (x, y)$, one determines a working set $W(z) \subset B \cup I$ that is part of the *active set*

$$A(z) = \{i \in B \cup I : z_i \in \{l_i, u_i\}\}.$$

Moré and Toraldo choose $W(z) = A(z)$, while we prefer choosing $W(z) = B(z)$ where

$$\begin{aligned} B(z) &= \{i \in A(z) : [\nabla_z \ell_r(z, \lambda)]_i \geqslant 0 \text{ si } x_i = l_i \text{ et } [\nabla_z \ell_r(z, \lambda)]_i \leqslant 0 \text{ si } x_i = u_i\} \\ &= \{i \in A(z) : [\nabla_z^{\mathrm{P}} \ell_r(z, \lambda)]_i \neq 0\}, \end{aligned}$$

is the *binding set*. This choice is motivated by the fact that the constraints with index in $A(z) \setminus B(z)$ are inactivated by the GP method.

The DP phase

The GP phase

3.4 Unboundedness of the closest feasible problem

It is known [4] that the closest feasible QP and the AL subproblem (whatever is the choice of λ and $r \geqslant 0$) are unbounded simultaneously (hence bounded simultaneously). This possible unboundedness property can be easily detected on the current AL subproblem, which is what is done by Oqla and Qpalm. This detection can occur in three occasions: in a GP phase, in a CG phase, and when a preconditioner is built. We give in this section the results on which this detection is grounded.

3.4.1 Detection in a GP phase

It is known [14; 2013, exercise 14.2] that

$$P_{[l_B, u_B]}(x - \alpha \nabla \varphi_\lambda(x)) = x - \alpha \nabla^{\mathrm{P}} \varphi_\lambda(x) \qquad \Longleftrightarrow \qquad x - \alpha \nabla^{\mathrm{P}} \varphi_\lambda(x) \in [l_B, u_B]$$

and that, when the feasible set is polyhedral, like here, these properties hold for all small nonnegative α. Let

$$d := -\nabla^{\rm p}\varphi_\lambda(x)$$

and assume that $x + \mathbb{R}_+ d \subset [l_B, u_B]$. Then the question arises to know whether φ_λ is unbounded along d, meaning that $\varphi_\lambda(x+\alpha d) \to -\infty$ when $\alpha \to \infty$?

The next proposition gives a new look at the detection of an unboundedness direction d, using the unboundedness of φ_λ along d. We denote by X^∞ the *asymptotic cone* of a nonempty closed convex set X.

Proposition 3.10 (unboundedness) *Let $x \in [l_B, u_B]$ and $d \in \mathbb{R}^n$.*

1) *If d satisfies $Hd = 0$, $A_I d \in [l_I, u_I]^\infty$, and $A_E d = 0$, then $\nabla\varphi_\lambda(x)^{\mathsf T} d \leqslant g^{\mathsf T} d$.*
2) *The following two conditions on d are equivalent*
   (a) *$g^{\mathsf T} d < 0$, $Hd = 0$, $A_I d \in [l_I, u_I]^\infty$, and $A_E d = 0$,*
   (b) *$\varphi_\lambda(x+\alpha d) \to -\infty$ when $\alpha \to \infty$.*
3) *If d satisfies $d \in [l_B, u_B]^\infty$ and one of the equivalent conditions in point 2, then d is a direction of unboundedness.*

Proof. 1) Using (3.16), $Hd = 0$ and $A_E d = 0$, one gets

$$\nabla\varphi_\lambda(x)^{\mathsf T} d = g^{\mathsf T} d + [\lambda_I + r(A_I x - \breve{y})]^{\mathsf T} A_I d. \tag{3.47}$$

Now, since $\breve{y} \equiv \breve{y}(x)$ is the projection of $A_I x + \lambda_I/r$ on $[l_I, u_I]$, it follows that

$$[\breve{y} - (A_I x + \lambda_I/r)]^{\mathsf T}(y - \breve{y}) \geqslant 0, \qquad \forall\, y \in [l_I, u_I].$$

By the assumption $A_I d \in [l_I, u_I]^\infty$, so that $y := \breve{y} + A_I d \in [l_I, u_I]$. Using this y in the previous inequality yields $[\lambda_I + r(A_I x - \breve{y})]^{\mathsf T} A_I d \leqslant 0$. The desired inequality now follows from (3.47).

2) $[(a) \Rightarrow (b)]$ For $\alpha \geqslant 0$ and some $y \in [l_I, u_I]$, define $x_\alpha := x + \alpha d$ and $y_\alpha := y + \alpha A_I d$. Observe that $A_E x_\alpha - b_E = A_E x - b_E$ (since $A_E d = 0$) and that $A_I x_\alpha - y_\alpha = A_I x - y$. Therefore, using $Hd = 0$ and $g^{\mathsf T} d < 0$, one gets from formula (2.2):

$$\ell_r(x_\alpha, y_\alpha, \lambda) = \ell_r(x, y, \lambda) + \alpha g^{\mathsf T} d \to -\infty, \quad \text{when } \alpha \to \infty. \tag{3.48}$$

Now $y_\alpha \in [l_I, u_I]$ (since $y \in [l_I, u_I]$ and $A_I d \in [l_I, u_I]^\infty$), so that by the very definition of φ_λ in (3.13), there holds $\varphi_\lambda(x_\alpha) \leqslant \ell_r(x_\alpha, y_\alpha, \lambda)$. Now (3.48) shows that $\varphi_\lambda(x_\alpha)$ tends to $-\infty$ when $\alpha \to \infty$.

$[(b) \Rightarrow (a)]$ Let P_{I^l, I^0, I^u} be a polyhedron that contains $x + \alpha d$ for all large α, say for $\alpha \geqslant \bar\alpha$ for some $\bar\alpha \geqslant 0$ (the concept is well defined since the number of polyhedra P_{I^l, I^0, I^u} is finite and these are convex). Then, for $\alpha \geqslant \bar\alpha$, the function $\xi : \alpha \in \mathbb{R}_+ \mapsto \xi(\alpha) := \varphi_\lambda(x + \alpha d)$ is quadratic and there holds

$$\forall\, \alpha > \bar\alpha : \quad \xi(\alpha) = \xi(\bar\alpha) + (\alpha - \bar\alpha)\xi'(\bar\alpha) + \frac{(\alpha - \bar\alpha)^2}{2}\,\xi'', \tag{3.49}$$

where the second derivative $\xi'' = \xi''(\alpha)$ is independent of $\alpha > \bar\alpha$.

- The very definition of P_{I^l,I^0,I^u} implies that $A_{I^0}d = [l_{I^0}, u_{I^0}]^\infty$, since otherwise,

$$l_{I^0} \leqslant A_{I^0}(x + \alpha d) + \frac{\lambda_{I^0}}{r} \leqslant u_{I^0}$$

would not hold for α sufficiently large, contradicting the fact that $x + \alpha d \in P_{I^l,I^0,I^u}$ for large α.

- The expression (3.17) of φ_λ on P_{I^l,I^0,I^u} shows that

$$\xi'' = d^\mathsf{T}\left(H + rA_{I^l\cup I^u\cup E}^\mathsf{T} A_{I^l\cup I^u\cup E}\right)d. \tag{3.50}$$

By (3.49) and (b), there must hold $\xi'' = 0$, implying that $Hd = 0$ and $A_{I^l\cup I^u\cup E}d = 0$.

- Using again the expression (3.17) of φ_λ on P_{I^l,I^0,I^u} and (3.50) show that

$$\xi'(\bar\alpha) = g^\mathsf{T}d.$$

By (3.49) and (b), there must hold $g^\mathsf{T}d < 0$.

3) This is essentially proposition 2.2. □

Counter-example 3.11 Equality does not necessarily holds in point 1 of proposition 3.10. Consider indeed the problem

$$\begin{cases} \inf_{x\in\mathbb{R}} x \\ 0 \leqslant x, \end{cases}$$

in which the bound constraint is considered as an inequality constraint with $A_I = 1$, $l_I = 0$, and $u_I = +\infty$. Now let $\lambda_I = 1$, $r = 3$, $x = -1$ (since $B = \varnothing$, the given x need not satisfy $x \geqslant 0$) and $d = 1$. Then

$$\begin{aligned} g^\mathsf{T}d &= 1 \\ \breve y(x) &= P_{[0,+\infty[}(-1 + 1/2) = 0. \\ \nabla\varphi_\lambda(x)^\mathsf{T}d &= g^\mathsf{T}d + (\lambda_I + r[A_Ix - \breve y(x)])^\mathsf{T}A_Id = 1 + (1 + 3[-1 - 0])1 = -1. \end{aligned}$$

Hence d may be a (strict) descent direction of φ_λ at x, verifying $Hd = 0$, $A_Id \in [l_I, u_I]^\infty$, and $A_Ed = 0$, but not a direction of unboundedness. This shows the claim.

Actually, it is not difficult to compute

$$\varphi_\lambda(x) = \begin{cases} \frac{3}{2}x^2 + 2x & \text{if } x \leqslant -1/3 \\ x - 1/6 & \text{otherwise.} \end{cases}$$

Therefore $\nabla^{\mathrm{p}}\varphi_\lambda(x) = \nabla\varphi_\lambda(-1) = -1$, so that $d = -\nabla^{\mathrm{p}}\varphi_\lambda(x)$ and nothing is to be gained by taking for d the negative gradient projection direction.

Now if u_I is finite, the given d is no longer in $[l_I, u_I]^\infty$ and the counter-example no longer works. Actually, the situation is then particularly simplified as shown by proposition 3.12 below. That proposition is useless because making the assumption that l_I and u_I are finite is not realistic. □

Proposition 3.12 (unboundedness in the GP) *Assume that l_I and u_I are finite. Let $x \in [l_B, u_B]$ and suppose that the direction $d := -\nabla^{\mathrm{P}}\varphi_\lambda(x)$ is nonzero. Then the following properties are equivalent*

(i) $Hd = 0$ *and* $A_{I\cup E}d = 0$,
(ii) $\varphi_\lambda(x + \alpha d) \to -\infty$ *when* $\alpha \to \infty$.

Therefore, if condition (i) holds and if $d \in [l_B, u_B]^\infty \setminus \{0\}$, the AL subproblem is unbounded.

Proof. $[(i) \Rightarrow (ii)]$ Let (I^l, I^0, I^u) be a partition of I such that x is in the polyhedron P_{I^l,I^0,I^u} defined in proposition 3.2. Since $A_I d = 0$, the direction $d \in P^\infty_{I^l,I^0,I^u}$. Therefore, the function $\xi : \alpha \in \mathbb{R}_+ \mapsto \xi(\alpha) := \varphi_\lambda(x + \alpha d)$ is quadratic and there holds

$$\forall\, \alpha \geqslant 0 : \quad \xi(\alpha) = \xi(0) + \alpha\xi'(0) + \frac{\alpha^2}{2}\,\xi'', \tag{3.51}$$

where the second derivative $\xi'' = \xi''(\alpha)$ is independent of $\alpha > 0$. Now,

$$\begin{aligned}
\xi'(0) &= \nabla\varphi_\lambda(x)^{\mathsf{T}} d \qquad \text{[differentiability of } \varphi_\lambda\text{]} \\
&= -\nabla\varphi_\lambda(x)^{\mathsf{T}} P_{-\,\mathrm{T}_x[l_B,u_B]} \nabla\varphi_\lambda(x) \qquad \text{[definition of } d\text{]} \\
&= -\|P_{-\,\mathrm{T}_x[l_B,u_B]} \nabla\varphi_\lambda(x)\|^2 \qquad \text{[property of the projection]} \\
&= -\|d\|^2 \qquad \text{[definition of } d\text{]} \\
&< 0 \qquad [d \neq 0].
\end{aligned}$$

On the other hand, the expression (3.17) of φ_λ on P_{I^l,I^0,I^u} and (i) show that

$$\xi'' = d^{\mathsf{T}} \left(H + r A^{\mathsf{T}}_{I^l\cup I^u\cup E} A_{I^l\cup I^u\cup E} \right) d = 0.$$

The conclusion (ii) now follows from (3.51).

$[(ii) \Rightarrow (i)]$ Let P_{I^l,I^0,I^u} be a polyhedron that contains $x + \alpha d$ for all large α, say for $\alpha \geqslant \bar\alpha$ for some $\bar\alpha \geqslant 0$ (the concept is well defined since the number of polyhedra P_{I^l,I^0,I^u} is finite and these are convex). Then, for $\alpha \geqslant \bar\alpha$, the function $\xi : \alpha \in \mathbb{R}_+ \mapsto \xi(\alpha) := \varphi_\lambda(x + \alpha d)$ is quadratic and there holds

$$\forall\, \alpha > \bar\alpha : \quad \xi(\alpha) = \xi(\bar\alpha) + (\alpha - \bar\alpha)\xi'(\bar\alpha) + \frac{(\alpha - \bar\alpha)^2}{2}\,\xi'', \tag{3.52}$$

where the second derivative $\xi'' = \xi''(\alpha)$ is independent of α. Let us now prove (i).

- The very definition of P_{I^l,I^0,I^u} implies that $A_{I^0} d = 0$, since otherwise,

  $$l_{I^0} \leqslant A_{I^0}(x + \alpha d) + \frac{\lambda_{I^0}}{r} \leqslant u_{I^0}$$

  would not hold for α sufficiently large, contradicting the fact that $x + \alpha d \in P_{I^l,I^0,I^u}$ for large α.

- Like above, the expression (3.17) of φ_λ on P_{I^l,I^0,I^u} shows that

  $$\xi'' = d^{\mathsf{T}} \left(H + r A^{\mathsf{T}}_{I^l\cup I^u\cup E} A_{I^l\cup I^u\cup E} \right) d.$$

  By (3.52) and (ii), there must hold $\xi'' = 0$, implying that $Hd = 0$ and $A_{I^l\cup I^u\cup E} d = 0$.

The last claim of the proposition follows from the following observations. If $d \in [l_B, u_B]^\infty$, $x + \alpha d \in [l_B, u_B]$ for all $\alpha \geqslant 0$. On the other hand, by the implication $(i) \Rightarrow (ii)$, $\varphi_\lambda(x + \alpha d) \to -\infty$ when $\alpha \to \infty$. Hence φ_λ is unbounded on $[l_B, u_B]$. □

3.4.2 Detection in a CG phase

We claim that a direction of unboundedness of the QP can easily be recognized when solving (3.34). It is actually a direction $d \in \mathbb{R}^n$ such that

$$\begin{cases} d_{W_B} = 0, \\ \left(H_{V_B V_B} + r A^{\mathsf{T}}_{(W_I \cup E) V_B} A_{(W_I \cup E) V_B}\right) d_{V_B} = 0, \\ d_{V_B} \in [l_{V_B}, u_{V_B}]^\infty, \\ A_{V_I V_B} d_{V_B} \in [l_{V_I}, u_{V_I}]^\infty. \end{cases} \tag{3.53}$$

Indeed, these conditions readily imply that $d^{\mathsf{T}}(H + r A^{\mathsf{T}}_{W_I \cup E} A_{W_I \cup E}) d = 0$, $d \in [l_B, u_B]^\infty$, and $A_{V_I} d \in [l_{V_I}, u_{V_I}]^\infty$. Positive semidefiniteness then implies $Hd = 0$, $A_{W_I} d = 0$, and $A_E d = 0$. To show that the conditions in (2.6) hold, we still have to prove that $g^{\mathsf{T}} d < 0$. By the CG properties, the scalar product of the residual of the linear system (3.34) at the current point and a nonzero computed direction is negative, which yields indeed $g^{\mathsf{T}}_{V_B} d_{V_B} < 0$ or $g^{\mathsf{T}} d < 0$.

3.4.3 Detection when preconditioning

The question arises as to when a possible direction of unboundedness must be detected in case the CG is preconditioned. There are three possibilities: when the preconditioner is built, when it is used, or when it is updated. The first (built) and last (updated) cases are problematic. Indeed, something must be done with the possible unboundedness direction that is detected there. If it is observed that along the direction a bound is hit, the preconditioner must be updated again, so that one enters a loop disconnected from the cycle of CG phases. For this reason, we have chosen to signal a possible direction of unboundedness when the preconditioner is used.

Let us denote by $M \equiv M(V_B, W_I)$ the matrix defined by (3.43).

Diagonal preconditioner

The diagonal preconditioner is the inverse of the diagonal of the matrix M when this one is nonsingular (see page 28). If M has a zero diagonal element, say $M_{\nu\nu}$, the preconditioner is not well defined. We show in this section that, in this case, a good candidate for a direction of unboundedness d can be built.

The precise result is the following:

$$\left.\begin{aligned} M_{\nu\nu} &= 0 \\ b_\nu &\neq 0 \\ \operatorname{sgn}(b_\nu) e^\nu_{V_B} &\in [l_{V_B}, u_{V_B}]^\infty \\ \operatorname{sgn}(b_\nu) A_{V_I} e^\nu &\in [l_{V_I}, u_{V_I}]^\infty \end{aligned}\right\} \quad \Longrightarrow \quad \operatorname{sgn}(b_\nu) e^\nu \text{ is a direction of unboundedness,} \tag{3.54}$$

where r is the right hand side of the linear system (3.34) at the current point, $\operatorname{sgn}(b_\nu)$ denotes the sign of b_ν, and e^ν denotes the νth basic vector of $\mathbb{R}^n$ (e^ν_i is 1 if $i = \nu$ and is

zero if $i \neq \nu$). This is a clear consequence of (3.53), since when $M_{\nu\nu} = 0$, the column (and the row) with index ν vanishes (a consequence of the positive semidefiniteness of M).

Let us examine the situations that can occur when a diagonal preconditioning is used. Note that the detection of a vanishing element $M_{\nu\nu}$ is realized at the first CG iteration, when preconditioning the first residual, so that the conjugacy mechanism of the CG algorithm has still not come into play. There are four possible disjoint situations.

- If $\mathrm{Diag}(M) > 0$, the preconditioning by $\mathrm{Diag}(M)$ can be realized during all the CG iterations used to solve (3.34).
- If an index ν is found such that the conditions on the left hand side of (3.54) are realized, the closest feasible QP is declared unbounded and the solver interrupts its job.
- If for all index ν in $N := \{\nu' : M_{\nu'\nu'} = 0\}$, there holds $b_\nu = 0$, the preconditioning is realized on the components not in N. Hence the CG algorithm generates iterates in the affine space $x_0 + \mathrm{vect}\{e^i : i \notin N\}$.
- If an index ν is found such that $M_{\nu\nu} = 0$ and $b_\nu \neq 0$, but that either $\mathrm{sgn}(b_\nu)e^\nu_{V_B} \notin [l_{V_B}, u_{V_B}]^\infty$ or $\mathrm{sgn}(b_\nu)A_{V_I}e^\nu \notin [l_{V_I}, u_{V_I}]^\infty$, then a step is taken up the closest bound in x or $A_I x$ and a new CG phase is undertaken next. In both cases, there will be no cycling:
  - if an x-bound is hit first, the index ν leaves V_B, so that $M_{\nu\nu} = 0$ is no longer in the diagonal preconditioner in the next CG phase,
  - if a y-bound is hit first, say bound $i \in V_I$, the index i enters W_I and the term $rA^\mathsf{T}_{iV_B}A_{iV_B}$ is added to M to become M'; hence in the next CG phase, $M'_{\nu\nu} = rA^2_{i\nu}$, which is nonzero, since when $y_{V_I} + \mathbb{R}_+A_{V_I}e^\nu$ crosses the ith bound, it must be that $A_ie^\nu \equiv A_{i\nu}$ is nonzero.

Cholesky preconditioner

The Cholesky preconditioner is the inverse of the matrix M when this one is nonsingular and is applied to a vector by solving two triangular linear systems (see point 2 on page 28). If M is rank deficient, say of rank $r < |V_B|$, the factorization procedure returns a pivoting matrix Q and a rank r lower triangular matrix L such that $QMQ^\mathsf{T} = LL^\mathsf{T}$, so that one searches a solution $x \equiv x_{V_B}$ to the system

$$LL^\mathsf{T}Qx = Qb,$$

where b is the right hand side of (3.34). Let us write L as follows

$$L = \begin{pmatrix} L_{11} & 0 \\ L_{21} & 0 \end{pmatrix},$$

where L_{11} is an order r lower triangular nonsingular matrix. We show in this section that, in this case, a good candidate for a direction of unboundedness d can be built when $r < |V_B|$.

To be comprehensive, let us examine all the situations that can occur.

- If L has full rank (i.e., $r = |V_B|$), the Cholesky preconditioning will work without having to take special precautions except the possible pivoting described by Q: a vector z is preconditioned by returning z_p solution to $Mz_p = z$ or $QMQ^\mathsf{T}Qz_p = Qz$ or $LL^\mathsf{T}Qz_p = Qz$ or
$$z_p = Q^\mathsf{T}L^{-\mathsf{T}}L^{-1}Qz.$$

The deconditioning of a preconditioned vector z is the vector

$$z_d = Q^T L L^T Q z_p = M z.$$

- If L is rank deficient (i.e., $r < |V_B|$) but $b \in \mathcal{R}(M)$ or, equivalently, $Qb \in \mathcal{R}(L)$, the linear system has a solution. Checking that $Qb \in \mathcal{R}(L)$ can be done by verifying that

$$L_{21} v = (Qb)_2, \tag{3.55}$$

where v solves

$$L_{11} v = (Qb)_1. \tag{3.56}$$

If this holds (up to some precision), whatever is $w \in \mathbb{R}^{|V_B|-r}$, the following $x \equiv x_{V_B}$ is solution to the system (3.34):

$$x = Q^T \begin{pmatrix} L_{11}^{-T}(v - L_{21}^T w) \\ w \end{pmatrix} = Q^T \begin{pmatrix} L_{11}^{-T} v \\ 0 \end{pmatrix} + Q^T Z w,$$

where

$$Z = \begin{pmatrix} -L_{11}^{-T} L_{21}^T \\ I \end{pmatrix}$$

is an injective matrix such that $\mathcal{R}(Z) = \mathcal{N}(LL^T)$ or $\mathcal{R}(Q^T Z) = \mathcal{N}(M)$. In order to minimize the number of operations, we take the particular solution obtained by setting $w = 0$.

- If L is rank deficient (i.e., $r < |V_B|$) and $b \notin \mathcal{R}(M)$, then (3.55) does not hold with v defined by (3.56) and one can define a candidate for a direction d of unboundedness of the QP. One takes $d_{W_B} = 0$, while d_{V_B} must necessarily satisfy

$$d_{V_B} = Q^T Z w \in \mathcal{R}(Q^T Z) = \mathcal{N}(M) \qquad \text{and} \qquad b^T d_{V_B} > 0.$$

This implies that one has to find w such that

$$0 < b^T d_{V_B} = (Qb) Z w = -(Qb)_1^T L_{11}^{-T} L_{21}^T w + (Qb)_2^T w = ((Qb)_2 - L_{21} v)^T w.$$

A natural candidate is $w = (Qb)_2 - L_{21} v \neq 0$. Finally, the candidate direction for unboundedness is defined by

$$d_{V_B} = Q^T \begin{pmatrix} L_{11}^{-T} L_{21}^T \\ -I \end{pmatrix} (L_{21} v - (Qb)_2) \qquad \text{and} \qquad d_{W_B} = 0.$$

Once v is known, this direction can be evaluated by solving a single triangular linear system. To ensure that this direction d is indeed a direction of unboundedness of the QP, one must still check that

$$d_{V_B} \in [l_{V_B}, u_{V_B}]^\infty \qquad \text{and} \qquad A_{V_I V_B} d_{V_B} \in [l_{V_I}, u_{V_I}]^\infty.$$

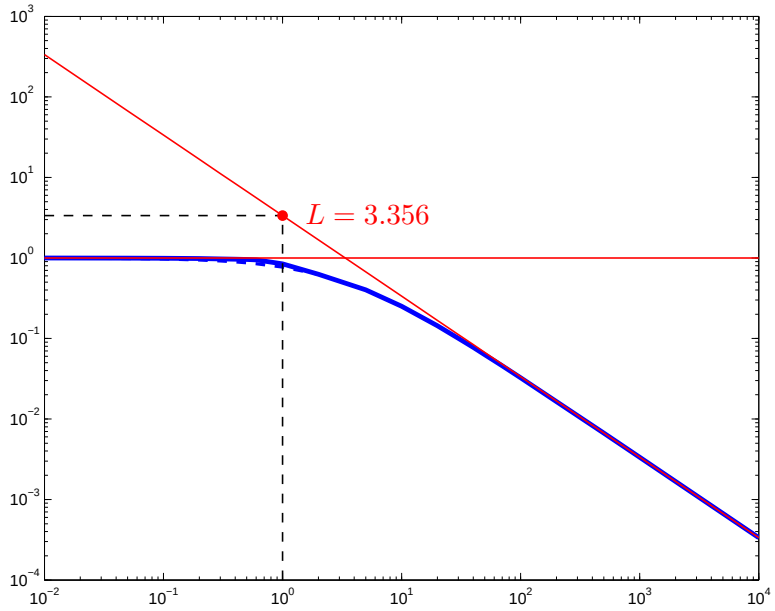

Figure 3.6: Global (plain blue curve) and asymptotic (dashed blue curve) linear rates of convergence for the problem `randqp10i` in log-log scale.

3.5 Rates of convergence

We present in this section numerical results that demonstrate the global linear convergence to zero of the constraints (in case of a feasible problem) or the shifted constraints (in case of an infeasible problem).

One first considers a problem with inequality and equality constraints, as well as bounds on the variables. The problem data has been generated randomly, in such a way that the problem has a solution. The global linear convergence of the constraint value to zero is highlighted in figure 3.6. The plain blue curve represents in log-log scale the function $r \mapsto \max_k \rho_k$, where ρ_k is the quotent in (3.6) and the maximum is taken over all the iterations k of a run. One point of the curve is obtained by maintaining the augmentation parameter r fixed during the run. It is known [5; 2005] that this curve is below the map $r \mapsto \max(1, L/r)$, where L is an unknown Lipschitz constant. In log-log scale, this upper bound is the piecewise linear map $\log r \mapsto \max(0, \log L - \log r)$, which is represented by the two red lines in the figure. Actually, the unknown constant L has been estimated by making the line with slope -1 tangent to the blue curve, which results in $L \simeq 3.356$ in the present case. We observe that for this test-problem the piecewise linear upper bound of the rate of convergence is rather tight and differs from the blue curve only slightly around its kink. This experiment is consistent with the one presented in [5; 2005]. We have also represented by a dashed blue curve the *asymptotic* linear rate of convergence (i.e., $r \mapsto \lim_{k\to\infty} \rho_k$), which is of course below the global linear rate of convergence. The curve is hardly visible and is different from the global linear rate only when r is "small" (for $r \leqslant 1$ in the present example).

The smooth behavior of the convergence rate curve in figure 3.6 is not necessariy always observed and depends on the problem and the starting point. The curve is indeed the result of an exploration of a small part of the dual function, the one along the paths followed by the iterates for various value of the augmentation parameter r, all starting from the same

first iterate.

3.6 Initializations

3.6.1 Primal-dual variables

The setting of the primal variable x_0 and dual variable λ_0 made by the user of the codes is always respected by Oqla and Qpalm. Now, if λ_0 is not set on entry, Oqla and Qpalm set it to zero. The setting of x_0, if not done on entry by the user, is a little more complex.

- If λ_0 is also not set by the user, then x_0 is set to zero, since there is no information available at all for an initialization.
- If λ_0 is set by the user, then for $i \in B$:
  - $(x_0)_i$ is set to l_i if $(\lambda_0)_i < 0$ and l_i is finite,
  - $(x_0)_i$ is set to u_i if $(\lambda_0)_i > 0$ and u_i is finite,

  then the resulting x_0 is projected on the box $[l_B, u_B]$.

References

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J.D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen (1999). *LAPACK Users' Guide* (third edition). SIAM, Philadelphia. 29

[2] J.F. Bonnans, J.Ch. Gilbert, C. Lemaréchal, C. Sagastizábal (2006). *Numerical Optimization – Theoretical and Practical Aspects* (second edition). Universitext. Springer Verlag, Berlin. [authors] [editor] [google books]. 18

[3] W. Cheng, Q. Liu, D. Li (2014). An accurate active set conjugate gradient algorithm with project search for bound constrained optimization. *Optimization Letters*, 8(2), 763–776. [doi]. 20

[4] A. Chiche, J.Ch. Gilbert (2015). How the augmented Lagrangian algorithm can deal with an infeasible convex quadratic optimization problem. *Journal of Convex Analysis* (to appear). [pdf]. 2, 4, 5, 10, 24, 34

[5] F. Delbos, J.Ch. Gilbert (2005). Global linear convergence of an augmented Lagrangian algorithm for solving convex quadratic optimization problems. *Journal of Convex Analysis*, 12, 45–69. [preprint] [editor]. 2, 9, 41

[6] M. Domorádová, Z. Dostál (2007). Projector preconditioning for patially bound-constrained quadratic optimization. *Numerical Linear Algebra with Applications*, 14, 791–806. [doi]. 33

[7] J.J. Dongarra, J.R. Bunch, C.B. Moler, G.W. Stewart (1979). *LINPACK Users' Guide*. SIAM, Philadelphia. 30

[8] Z. Dostál (1997). Box constrained quadratic programming with proportioning and projections. *SIAM Journal on Optimization*, 7(3), 871–887. [doi]. 20

[9] Z. Dostál (2003). A proportioning based algorithm with rate of convergence for bound constrained quadratic programming. *Numerical Algorithms*, 34(2-4), 293–302. [doi]. 20

[10] Z. Dostál (2009). *Optimal Quadratic Programming Algorithms*, volume 23 of *Springer Optimization and Its Applications*. Springer. 33

[11] Z. Dostál, M. Domorádová, M. Sadowská (2011). Superrelaxation and the rate of convergence in minimizing quadratic functions subject to bound constraints. *Computational Optimization and Applications*, 48(1), 23–44. [doi]. 20, 33

[12] Z. Dostál, A. Friedlander, S.A. Santos (2003). Augmented Lagrangians with adaptive precision control for quadratic programming with simple bounds and equality constraints. *SIAM Journal on Optimization*, 13, 1120–1140. [doi]. 20

[13] D.-Z. Du, X.-S. Zhang (1989). Global convergence of Rosen's gradient projection method. *Mathematical Programming*, 44, 357–366. [doi]. 26

[14] J.Ch. Gilbert (2013). *Éléments d'Optimisation Différentiable – Théorie et Algorithmes*. Syllabus de cours à l'ENSTA, Paris. [internet]. 18, 34

[15] J.Ch. Gilbert, É. Joannopoulos (2015). OQLA/QPALM - Convex quadratic optimization solvers using the augmented Lagrangian approach, with an appropriate behavior on infeasible or unbounded problems. Research report, INRIA, BP 105, 78153 Le Chesnay, France. (to appear). 2

[16] A. Greenbaum, V. Pták, Z. Strakoš (1996). Any nonincreasing convergence curve is possible for GMRES. *SIAM Journal on Matrix Analysis and Applications*, 17, 465–469. [doi]. 26

[17] S. Hammarling, N.J. Higham, C. Lucas (2007). LAPACK-style codes for pivoted Cholesky and *QR* updating. In B. Kågström (editor), *Applied Parallel Computing - State of the Art in Scientific Computing*, Lecture Notes in Computer Science 4699, pages 137–146. Springer, Berlin, Heidelberg. [doi]. 30

[18] N.J. Higham (2002). *Accuracy and Stability of Numerical Algorithms* (second edition). SIAM Publication, Philadelphia. 28, 33

[19] N.J. Higham (2008). Cholesky factorization. MIMS EPrint 2008.116, University of Manchester. 32

[20] J.-B. Hiriart-Urruty, C. Lemaréchal (1993). *Convex Analysis and Minimization Algorithms*. Grundlehren der mathematischen Wissenschaften 305-306. Springer-Verlag. 16

[21] C. Lucas (2004). LAPACK-style codes for level 2 and 3 pivoted Cholesky factorizations. Technical Report Numerical Analysis Report No. 442, University of Manchester. 30

[22] C. Lucas (2004). *Algorithms for Cholesky and QR Factorizations, and the Semidefinite Generalized Eigenvalue Problem*. PhD Thesis, University of Manchester. 30

[23] J.J. Moré, G. Toraldo (1989). Algorithms for bound constrained quadratic programming problems. *Numerische Mathematik*, 55, 377–400. [doi]. 4

[24] J.J. Moré, G. Toraldo (1991). On the solution of large quadratic programming problems with bound constraints. *SIAM Journal on Optimization*, 1, 93–113. [doi]. 4, 12, 20, 25, 26, 27, 34

[25] J. Nocedal, S.J. Wright (2006). *Numerical Optimization* (second edition). Springer Series in Operations Research. Springer, New York. 33

[26] J.B. Rosen (1960). The gradient projection method for nonlinear programming; part I: linear constraints. *Journal of the Society for Industrial and Applied Mathematics*, 8, 181–217. [doi]. 26

Index