How the augmented Lagrangian algorithm can deal with an infeasible convex quadratic optimization problem
—
Motivation, analysis, implementation

J.Ch. Gilbert (INRIA Paris-Rocquencourt)
Joint work with
Alice Chiche (EDF ⇝ Artelys)
Émilie Joannopoulos (INRIA Paris-Rocquencourt ⇝ Sherbrooke Univ.)

April 23, 2015


Since you ask me to mention a gratifying paper, let me pick “A method for nonlinear constraints in minimization problems”, because it is regarded as one of the sources of the “augmented Lagrangian method”, which is now of fundamental importance in mathematical programming. I have been very fortunate to have played a part in discoveries of this kind.

M.J.D. Powell [19; 2003]
A brief overview of numerical nonlinear optimization

The problem to solve

A standard generic nonlinear optimization problem consists in

\[
\begin{align*}
(P_{EI}) \quad \left\{ \begin{array}{l}
\inf_x f(x) \\
c_E(x) = 0 \\
c_I(x) \leq 0,
\end{array} \right. 
\end{align*}
\]

where \( f : \mathbb{R}^n \to \mathbb{R} \), \( c_E : \mathbb{R}^n \to \mathbb{R}^{m_E} \), and \( c_I : \mathbb{R}^n \to \mathbb{R}^{m_I} \) are smooth (possibly non convex) functions.

Sometimes we will consider simplified a version (to avoid being cumbersome), namely

\[
(P_I) \quad \left\{ \begin{array}{l}
\inf_x f(x) \\
c_I(x) \leq 0.
\end{array} \right. 
\]
A primal algorithm gives priority to the visible or primal variables $x$.

Main ideas
- penalize the constraints with penalty parameter $r \to$ (some limit),
- apply an unconstrained algorithm to solve the penalized problem.

Example 1: exterior penalization (quadratic penalization)

$$(P_I) \quad \begin{cases} \inf_x f(x) \\ c_I(x) \leq 0 \end{cases} \quad \sim \quad (P_{I,r}) \quad \inf_x \left( f(x) + \frac{r}{2} \| c_I(x)^+ \|_2^2 \right).$$

Pros and cons
⊕ Easy to implement.
⊖ Sequence of problems to solve.
⊕ Ill-conditioning.
A brief overview of numerical nonlinear optimization

Primal algorithms

Example 2: interior penalization (interior point methods)

\[
(P_I) \quad \left\{ \begin{array}{l}
\inf_x f(x) \\
c_i(x) \leq 0
\end{array} \right\} \sim (P_{I,r}) \quad \inf_x \left( f(x) - r \sum_{i \in I} \log |c_i(x)| \right).
\]

Pros and cons

⊕ Easy to implement.
⊖ Sequence of problems to solve.
⊖ Ill-conditioning.
⊕ Each problem \((P_{I,r})\) can be solved inexactly (a single Newton step, in linear optimization).

\[
f(x) = 1 - x - \frac{1}{5}x^3
\]

Chiche, Gilbert, Joannopoulos

9 / 101

A brief overview of numerical nonlinear optimization

Dual algorithms

A dual algorithm gives priority to the hidden or dual variables \(\lambda\).

○ The hidden variables are revealed by the optimality conditions (= local description of optimality).

○ If \(x_*\) is a local solution to \((P_{EI})\) (+ smoothness and qualification assumptions), there exist multipliers or dual variables \(\lambda_* \in \mathbb{R}^m\) such that

\[
(KKT) \quad \left\{ \begin{array}{l}
\nabla_x \ell(x_*, \lambda_*) = 0 \\
c_E(x_*) = 0 \\
0 \leq (\lambda_*)_i \perp c_i(x_*) \leq 0.
\end{array} \right\}
\]

where

• KKT = Karush-Kuhn-Tucker,
• Lagrangian function \(\ell(x, \lambda) = f(x) + \lambda^T c(x) = f(x) + \sum_i \lambda_i c_i(x)\).
A brief overview of numerical nonlinear optimization
Dual algorithms

How to generate dual iterates?

- For some coupling function \( \varphi : X \times \Lambda \to \mathbb{R} \), write \((P_{EI})\) as an \(\inf\sup\):

\[
(P_{EI}) \quad \inf_{x \in X} \sup_{\lambda \in \Lambda} \varphi(x, \lambda).
\]

- The dual problem then reads

\[
(D_{EI}) \quad \sup_{\lambda \in \Lambda} \inf_{x \in X} \varphi(x, \lambda) = - \inf_{\lambda \in \Lambda} \left( \sup_{x \in X} - \varphi(x, \lambda) \right).
\]

- Generate the dual iterates by minimizing on \(\Lambda\) the dual function

\[
\lambda \in \Lambda \mapsto \delta(\lambda) := \sup_{x \in X} - \varphi(x, \lambda) \in \mathbb{R}.
\]

How to chose the coupling function \(\varphi\)?

- The problem \((P_{EI})\) must be identical to

\[
\inf_{x \in \mathbb{R}^n} \sup_{\lambda \in \Lambda} \varphi(x, \lambda).
\]

- In some sense, \((D_{EI})\) must be “equivalent” to \((P_{EI})\).

Ensured if a PD solution \((x_*, \lambda_*)\) to \((P_{EI})\) is a saddle-point of \(\varphi\):

\[
\forall x \in \mathbb{R}^n, \, \forall \lambda \in \Lambda : \quad \varphi(x_*, \lambda) \leq \varphi(x_*, \lambda_*) \leq \varphi(x, \lambda_*).
\]
A brief overview of numerical nonlinear optimization

Dual algorithms

Lagrangian relaxation

- The problem \((P_{EI})\) can be written

\[
\inf_{x \in \mathbb{R}^n} \sup_{\lambda \in \Lambda} f(x) + \lambda^T c_E(x) + \lambda^T c_I(x),
\]

where \(\Lambda := \{ \lambda \in \mathbb{R}^m : \lambda_I \geq 0 \} \).

- Hence the dual problem \((D_{EI})\) consists in minimizing the dual function

\[
\lambda \in \mathbb{R}^m \mapsto \delta(\lambda) := \left( \sup_{x \in \mathbb{R}^n} -\ell(x, \lambda) \right) + I_\Lambda(\lambda) \in \mathbb{R},
\]

which is nonsmooth, convex, and closed (i.e., l.s.c.).

- Saddle-point at a KKT point \((x_*, \lambda_*)\) if \((P_{EI})\) is convex.

- Typical (and difficult) algorithm: bundle method [17].

Augmented Lagrangian relaxation (multiplier method)

- For any \(r > 0\), problem \((P_I)\) can also be written \((c_I(x) + y = 0, y \geq 0)\)

\[
\inf_{(x, y) \in \mathbb{R}^n \times \mathbb{R}^m} \sup_{\lambda \in \mathbb{R}^m} f(x) + \lambda^T (c_I(x) + y) + \frac{r}{2} \|c_I(x) + y\|^2, \quad \ell_r(x, y, \lambda)
\]

where \(\ell_r\) is called the augmented Lagrangien.

- Hence the dual problem \((D_{EI})\) consists in minimizing the dual function

\[
\lambda \in \mathbb{R}^m \mapsto \delta_r(\lambda) := \sup_{(x, y) \in \mathbb{R}^n \times \mathbb{R}^m} -\ell_r(x, y, \lambda), \quad \text{solution } (x_+, y_+)
\]

which is smooth \((C^{1,1})\), convex, and closed.

- Local saddle-point at a KKT+SOC2 point \((x_*, \lambda_*)\) if \(r\) is large enough.

- Easy algorithm: \(\lambda_+ := \lambda + r [c_I(x_+) + y_+]\) [16, 18, 21, 4, 1, 23, 24].
A brief overview of numerical nonlinear optimization

Dual algorithms

Outline of the augmented Lagrangian (AL) algorithm

One iteration: from \((\lambda_k, r_k) \in \mathbb{R}^m \times \mathbb{R}^+\) to \((\lambda_{k+1}, r_{k+1})\).

- Compute (if possible, exit otherwise)
  \[
  (x_{k+1}, y_{k+1}) \in \arg\min_{(x, y) \in \mathbb{R}^n \times \mathbb{R}^m_+} \ell_{r_k}(x, y, \lambda_k).
  \]

- Update the multipliers by \(\lambda_{k+1} = \lambda_k + r_k[c_I(x_{k+1}) + y_{k+1}]\).
- Stop if \([c_I(x_{k+1}) + y_{k+1}] \simeq 0\).
- Update \(r_k \rightarrow r_{k+1} \ldots\)

Pros and cons

⊕ Do not require convexity (but easier if \((PEI)\) is convex).
⊕ Convergence well understood if \((PEI)\) is convex.
⊕ A sequence of nonlinear optimization problems to solve in (1).
⊕ (1) sometimes difficult (\(y \geq 0\), destroy decomposition, ill-conditioning).
⊕ Update of \(r_k\) is tricky.

Another point of view on the augmented Lagrangian

- The original idea [16, 18] was to penalize \(\ell(\cdot, \lambda_*)\) instead of \(f\) because this yields
  - exactness (solving a single penalty problem),
  - better conditioning (\(r\) large but not infinite).

- Since \(\lambda_*\) is not known, an iterative process must generate \(\lambda_k \rightarrow \lambda_*\) (by minimizing the dual function).
An important property of the AL algorithm, when \((P_{EI})\) is convex

AL algorithm = proximal algorithm on the dual function \(\delta\).

The proximal algorithm on the dual function \(\delta\) computes \(\lambda_{k+1}\) from \(\lambda_k\) by

\[
\lambda_{k+1} = \arg \min_{\lambda \in \mathbb{R}^m} \left( \delta(\lambda) + \frac{1}{2r_k} \|\lambda - \lambda_k\|^2 \right).
\]

Optimality: \(\exists s_{k+1} \in \partial \delta(\lambda_{k+1})\) such that

\[
0 = s_{k+1} + \frac{1}{r_k} (\lambda_{k+1} - \lambda_k) \quad \text{or} \quad \lambda_{k+1} = \lambda_k - r_k s_{k+1}, \quad \text{for some } s_{k+1} \in \partial \delta(\lambda_{k+1}).
\]

Hence it is an implicit subgradient method (implicit Euler).

One writes

\[
\lambda_{k+1} = \prox_{\delta, r_k}(\lambda_k)
\]
Proposition (Rockafellar [22; 1973])

If \( \delta \in \mathcal{C}^{\text{conv}}(\mathbb{R}^m) \) and \( r_k > 0 \), then

\[
- \inf_{(x,y) \in \mathbb{R}^n \times \mathbb{R}^m} \ell_r(x, y, \lambda_k) = \inf_{\lambda \in \mathbb{R}^m} \left( \delta(\lambda) + \frac{1}{2r_k} \|\lambda - \lambda_k\|^2 \right).
\]

Any solution \((x_{k+1}, y_{k+1})\) to the problem in the LHS and the unique solution \(\lambda_{k+1}\) to the problem in the RHS are linked by

\[
\begin{align*}
\lambda_{k+1} &= \lambda_k + r_k [c_l(x_{k+1}) + y_{k+1}] \\
- [c_l(x_{k+1}) + y_{k+1}] &\in \partial \delta(\lambda_{k+1}).
\end{align*}
\]

Hence the multiplier computed by the AL algorithm is \( \lambda_{k+1} = \text{prox}_{\delta, r_k}(\lambda_k) \).
A brief overview of numerical nonlinear optimization

Primal-dual algorithms

A primal-dual algorithm generates a PD sequence \( \{(x_k, \lambda_k)\} \)

Consider the generic problem

\[
(P_{EI}) \quad \begin{cases} 
\inf_x f(x) \\
c_E(x) = 0 \\
c_I(x) \leq 0, 
\end{cases}
\]

The classical primal-dual algorithm works on the first order optimality conditions directly

\[
(KKT) \quad \begin{cases} 
\nabla_x \ell(x^*, \lambda^*) = 0 \\
c_E(x^*) = 0 \\
0 \leq (\lambda^*)_I \perp c_I(x^*) \leq 0. 
\end{cases}
\]

“Linearization” gives the displacement \((d, \mu)\) of \((x, \lambda)\):

\[
(KKT') \quad \begin{cases} 
\nabla_x \ell(x_k, \lambda_k) + \nabla^2_{xx} \ell(x_k, \lambda_k) d + c'(x_k)^T \mu = 0 \\
c_E(x_k) + c'_E(x_k) d = 0 \\
0 \leq (\lambda_k + \mu)_I \perp (c_I(x_k) + c'_I(x_k) d) \leq 0. 
\end{cases}
\]

The system \((KKT')\) is formed of the first order optimality conditions of the following osculating quadratic problem in \(d\):

\[
(OQP) \quad \begin{cases} 
\inf_d \nabla f(x_k)^T d + \frac{1}{2} d^T \nabla^2_{xx} \ell(x_k, \lambda_k) d \\
c_E(x_k) + c'_E(x_k) d = 0 \\
c_I(x_k) + c'_I(x_k) d \leq 0, 
\end{cases}
\]

whose multipliers are \(\lambda_k^{QP} := \lambda_k + \mu\).

One iteration of the local SQP/SQO algorithm: from \((x_k, \lambda_k)\) to \((x_{k+1}, \lambda_{k+1})\)

- If possible, solve \((OQP)\), to get \(d_k\) and \(\lambda_k^{QP}\).
- Update \(x_{k+1} := x_k + d_k\) and \(\lambda_{k+1} := \lambda_k^{QP}\). 

Chiche, Gilbert, Joannopoulos
In the sequel:

**Analyse/implement an AL algorithm to the solve efficiently the OQP of the SQP algorithm.**

---

**Convex quadratic optimization**

The QP to solve

The problem to solve

\[
(P) \quad \begin{cases} 
\inf_{x \in \mathbb{R}^n} q(x) \\
\mathbf{l} \leq Ax \leq \mathbf{u},
\end{cases}
\]

(2)

where \( q \) is a **convex quadratic** function defined at \( x \in \mathbb{R}^n \) by

\[
q(x) = \mathbf{g}^T x + \frac{1}{2} x^T \mathbf{H} x
\]

and

- \( g \in \mathbb{R}^n \)
- \( \mathbf{H} \succeq 0 \) (NP-hard otherwise, \( P \) encompasses linear optimization),
- \( A \) is \( m \times n \),
- \( \mathbf{l}, \mathbf{u} \in \mathbb{R}^m \) satisfy \( \mathbf{l} < \mathbf{u} \).

Also equality constraints in all solvers.
Convex quadratic optimization
Can one still make progress in convex quadratic optimization?

The problem is polynomial and can be solved by
- active-set methods → probably non-polynomial,
- interior-point methods → polynomial,
- nonsmooth methods → polynomial on subclasses,
- other methods (including the augmented Lagrangian method).

Has this discipline been fully explored in the XXth century?

---

**Observation 1.** Odd behavior of Quadprog (Matlab). If the data is

\[ g = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \quad H = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 4 & 2 \\ 0 & 2 & 1 \end{pmatrix}, \quad x \geq \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix}, \]

Quadprog-active-set answers

Exiting: the solution is unbounded and at infinity;
Function value: 3.2000e+33

Very odd, since the problem has a unique solution, which is

\[ x = \begin{pmatrix} -1 \\ -1 \\ 2 \end{pmatrix} \quad \text{and} \quad \text{val}(P) = -1.5. \]

It is a benign flaw, since if \( H \sim H + \epsilon I \), Quadprog finds a near solution.
Convex quadratic optimization
Can one still make progress in convex quadratic optimization?

Quadprog-reflective-trust-region (default algorithm) answers

Optimization terminated: relative function value changing by
less than OPTIONS.TolFun.
Function value: -1.5

Correct answer!

Conclusion: the good algorithm may depend on the problem.

Observation 2. On the solvable convex QPs of the CUTEst collection:
• first group: 138 problems, solvers in Fortran or C++,
• second group: 58 problems ($n \leq 500$), solver in Matlab.

<table>
<thead>
<tr>
<th>Solvers</th>
<th>% failure</th>
<th>% too slow</th>
<th>% infeasibility</th>
<th>% other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qpa (AS)</td>
<td>30 %</td>
<td>15 %</td>
<td>15 %</td>
<td>–</td>
</tr>
<tr>
<td>Qpb (IP)</td>
<td>20 %</td>
<td>5 %</td>
<td>2 %</td>
<td>13 %</td>
</tr>
<tr>
<td>Ooqp (IP)</td>
<td>54 %</td>
<td>1 %</td>
<td>12 %</td>
<td>41 %</td>
</tr>
<tr>
<td>Quadprog (AS)</td>
<td>33 %</td>
<td>12 %</td>
<td>19 %</td>
<td>2 %</td>
</tr>
</tbody>
</table>

• “too slow”: requires more than 600 seconds,
• “infeasibility”: wrong diagnosis of infeasibility,
• “other”: “too small stepsize”, “too small direction”, “ill-conditioning”, and “unknown”.
Convex quadratic optimization
Can one still make progress in convex quadratic optimization?

The problem does not come from some very difficult QPs. For example, on the CUTEst problem QSCTAP1 ($n = 480$, $n_b = 480$ lower bounds, $m_l = 180$ lower bounds, $m_E = 120$):

- Qpa claims that the problem is unbounded,
- Qpb claims that the problem has a solution,
- Ooqp claims that the problem is infeasible,
- Quadprog stops on a too large number of iterations ($\geq 10^4$).

⇒ Still progress to do.

Observation 3 (more important).

Most (all?) solvers do not give appropriate information when the QP is special, they just return a flag.

- **Special** means $\text{val}(P) \notin \mathbb{R}$ below:
  - $\text{val}(P) \in \mathbb{R} \iff$ the problem has a solution (Frank-Wolfe [10; 1956]),
  - $\text{val}(P) = -\infty \iff$ the problem is feasible and unbounded,
  - $\text{val}(P) = +\infty \iff$ the problem is infeasible.

- **Appropriate** means useful when the QP solver is used in the SQP algorithm for solving a nonlinear optimization problem.
Towards the AL algorithm

- The problem is transformed by using an auxiliary variable \( y \):
  \[
  (P) \quad \inf_{x \in \mathbb{R}^n} q(x) \quad \text{s.t.} \quad l \leq Ax \leq u \\
  \Leftrightarrow \quad (P') \quad \inf_{(x, y) \in \mathbb{R}^n \times \mathbb{R}^m} q(x) \quad \text{s.t.} \quad Ax = y \quad \text{and} \quad l \leq y \leq u.
  \]

- Equality constraints penalized by the augmented Lagrangian
  \[
  \ell_r(x, y, \lambda) := q(x) + \lambda^T (Ax - y) + \frac{r}{2} \|Ax - y\|^2.
  \]

- At each iteration the AL algorithm solves
  \[
  \inf_{(x, y) \in \mathbb{R}^n \times [l, u]} \ell_r(x, y, \lambda). \tag{3}
  \]

- The AL algorithm makes sense if it is easier to solve (3) than (P).

The AL algorithm for a solvable convex QP

One iteration, from \((\lambda_k, r_k) \in \mathbb{R}^m \times \mathbb{R}^+\) to \((\lambda_{k+1}, r_{k+1})\):

- Compute (if possible, exit otherwise)
  \[
  (x_{k+1}, y_{k+1}) \in \arg \min_{(x, y) \in \mathbb{R}^n \times [l, u]} \ell_{r_k}(x, y, \lambda_k).
  \]

- Update the multipliers
  \[
  \lambda_{k+1} = \lambda_k - r_k s_{k+1}, \quad \text{where} \ s_{k+1} := y_{k+1} - Ax_{k+1}.
  \]

- Stop if
  \[
  s_{k+1} \approx 0.
  \]

- Update \( r_k \rho_{k+1} > 0: \rho_k := \|s_{k+1}\|/\|s_k\| \) and
  \[
  r_{k+1} := \max \left(1, \frac{\rho_k}{\rho_{\text{des}}} \right) r_k.
  \]
The AL algorithm
The AL algorithm for a solvable convex QP

Interpretation of the AL algorithm
One iteration, from \((\lambda_k, r_k) \in \mathbb{R}^m \times \mathbb{R}^+\) to \((\lambda_{k+1}, r_{k+1})\):

- Compute (if possible, exit otherwise)
  \[ (x_{k+1}, y_{k+1}) \in \mathop{\arg\min}_{(x,y) \in \mathbb{R}^n \times [l,u]} \ell_{r_k}(x, y, \lambda_k). \]

- Update the multipliers
  \[ \lambda_{k+1} = \lambda_k - r_k s_{k+1}, \quad \text{where} \quad s_{k+1} := y_{k+1} - Ax_{k+1}. \]

- Stop if
  \[ s_{k+1} \approx 0. \]

- Update \( r_k \leftarrow r_{k+1} > 0 \):
  \[ \rho_k := \|s_{k+1}\|/\|s_k\| \quad \text{and} \quad r_{k+1} := \max\left(1, \frac{\rho_k}{\rho_{\text{des}}}\right) r_k. \]

The dual function \( \delta : \mathbb{R}^m \to \mathbb{R} \), defined at \( \lambda \in \mathbb{R}^m \) by

\[
\delta(\lambda) := -\inf_{(x,y) \in \mathbb{R}^n \times [l,u]} \left( q(x) + \lambda^T (Ax - y) \right).
\]

- \( \delta \) is convex, closed, and \( \delta > -\infty \).
- \( \text{dom} \delta \neq \emptyset \iff \delta \neq +\infty \iff \delta \in \text{Conv}(\mathbb{R}^m). \)
- Piecewise quadratic (quadratic on each orthant).

If \((P) \equiv (P')\) has a solution:

\[ 0 \in \partial \delta(\bar{\lambda}) \iff \bar{\lambda} \text{ is a dual solution to } (P'). \]

The AL algorithm looks for a

\[ \bar{\lambda} \in \mathop{\arg\min} \delta. \]
The AL algorithm
The AL algorithm for a solvable convex QP

AL iterates minimizing the dual function for a solvable QP

- $\delta$ is piecewise quadratic
  $$\delta(\lambda) = \frac{1}{2}\lambda^T S\lambda + (v + y\lambda)^T \lambda + C_{st}$$
- $S_D := \arg \min \delta$
- $\partial\delta(\lambda_{k+1})$ contains
  $$\frac{\lambda_k - \lambda_{k+1}}{r_k} = y_{k+1} - Ax_{k+1}$$
- small $r_k$'s in the figure

Motivation of the update rule of the penalty parameters

One iteration, from $(\lambda_k, r_k) \in \mathbb{R}^m \times \mathbb{R}^{++}$ to $(\lambda_{k+1}, r_{k+1})$:

- Compute (if possible, exit otherwise)
  $$\arg \min (x, y) \in \mathbb{R}^n \times \mathbb{R}^{++} \ell_r(x, y, \lambda_k).$$
- Update the multipliers
  $$\lambda_{k+1} = \lambda_k - r_k s_{k+1}, \quad \text{where } s_{k+1} := y_{k+1} - Ax_{k+1}.$$  
- Stop if
  $$s_{k+1} \approx 0.$$
- Update $r_k \ni r_{k+1} > 0$: $\rho_k := \|s_{k+1}\|/\|s_k\|$ and
  $$r_{k+1} := \max \left(1, \frac{\rho_k}{\rho_{des}}\right) r_k.$$
The AL algorithm
The AL algorithm for a solvable convex QP

- The update rule of $r_k$ is based on the following global linear convergence result [8; 2005].
  - If $(P)$ has a solution, then the dual solution set $S_D \neq \emptyset$ and
    \[
    \forall \beta > 0, \exists L > 0, \text{ dist}_{S_D}(\lambda_0) \leq \beta \quad \text{implies that} \quad \forall k \geq 1, \|s_{k+1}\| \leq \min \left(1, \frac{L}{r_k}\right) \|s_k\|,
    \]
    where $s_k := y_k - Ax_k$.
  - (4) comes from a quasi-global error bound on the dual solution set $S_D$:
    for any bounded set $B \subset \mathbb{R}^m$, there is an $L > 0$, such that
    \[
    \forall \lambda \in S_D + B : \quad \text{dist}_{S_D}(\lambda) \leq L \left( \inf_{s \in \partial \delta(\lambda)} \|s\| \right).
    \]
  - The Lipschitz constant $L$ is difficult to deduce from the data . . .

Let $m = 1$ and $l < 0 < u$. Consider the problem

\[
\begin{array}{l}
\inf 0 \\
l \leq 0x \leq u,
\end{array}
\]

- The dual function reads
  \[
  \delta(\lambda) = \begin{cases} \\
  l\lambda & \text{if } \lambda \leq 0 \\
  u\lambda & \text{if } \lambda > 0.
  \end{cases}
  \]

- Hence $S_D = \{0\}$ and the quasi-global error bound reads
  \[
  \forall B > 0, \exists L > 0, |\lambda| \leq B \quad \Rightarrow \quad |\lambda| \leq \begin{cases} \\
  -Ll & \text{if } \lambda < 0 \\
  0 & \text{if } \lambda = 0 \\
  Lu & \text{if } \lambda > 0.
  \end{cases}
  \]

- Therefore, for $B$ fixed, $L \uparrow \infty$ when $l \searrow 0$ or $u \nearrow 0$ (fix $\lambda$ in the error bound).
The AL algorithm

The AL algorithm for a solvable convex QP

The rule of the nonlinear solver Algencan [2; 2014]:

\[ r_0 = P_{[10^{-8}, 10^{+8}]} \left( 10 \frac{\max(1, |q(x_0)|)}{\max(1, \|Ax_0 - y_0\|^2)} \right). \]

- Motivation: balancing the objective and constraint parts of the \( \ell_2 \) penalty function.
- In the previous example, the rule yields (whatever is \( l \) and \( u \)):
  \[ r_0 = 10. \]
- It does not catch the following fact:
  
  for some problems, the appropriate \( r \) depends on the distance from the optimal constraint value \( A\bar{x} \) to \([l, u]^c\).

In Oqla/Qpalm, \( L \) is guessed and \( r_k \) is set by the observation of

\[ \rho_k := \frac{\|s_{k+1}\|}{\|s_k\|}, \]

thanks to the global linear convergence:

\[ \forall \beta > 0, \exists L > 0, \text{dist}_{S_D}(\lambda_0) \leq \beta \quad \text{implies that} \quad \forall k \geq 1, \|s_{k+1}\| \leq \frac{L}{r_k} \|s_k\|. \]

- Lower bound of \( L \):
  \[ L_{\text{inf}, k} := \max_{1 \leq i \leq k} \rho_i r_i. \]

- Setting of \( r_{k+1} \):
  \[ r_{k+1} = \frac{L_{\text{inf}, k}}{\rho_{\text{des}}}. \]

- With \( \rho_{\text{des}} = 1/10 \), convergence occurs in 10..15 AL iterations.
Effect of the update rule of $r_k$ for infeasible QPs

If the QP is infeasible:

- $\|s_k\| \searrow \sigma > 0$ and
  $$\rho_k := \frac{\|s_{k+1}\|}{\|s_k\|} \to 1,$$

- the rule (increases $r_k$ whenever $\rho_k > \rho_{des}$ [$\rho_{des} < 1$]) $\implies r_k \nearrow \infty$,

- the AL subproblems become ill-conditioned,

- could stop when $r_k \geq \bar{r}$, but
  - difficult to find a universal threshold $\bar{r}$,
  - no information on the problem on return.

Can one have a global linear convergence when the QP is infeasible?

The smallest feasible shift

- It is always possible to find a shift $s \in \mathbb{R}^m$ such that
  $$l \leq Ax + s \leq u \text{ is feasible for } x \in \mathbb{R}^n.$$

- These feasible shifts are exactly those in $S := [l, u] + \mathcal{R}(A)$:

  - The smallest feasible shift $\bar{s} := \arg \min \{\|s\| : s \in S\}$.
  - $\bar{s} = 0 \iff \text{(P) is feasible.}$

$\text{Chiche, Gilbert, Joannopoulos}$
The closest feasible problem

The shifted QPs (feasible iff \( s \in S \), may be unbounded)

\[
(P_s) \quad \begin{cases} 
\inf_x q(x) \\
I \leq Ax + s \leq u
\end{cases} \quad \text{and} \quad (P'_s) \quad \begin{cases} 
\inf_x q(x) \\
Ax + s = y \\
I \leq y \leq u.
\end{cases}
\] (6)

The closest feasible problems (feasible, may be unbounded)

\[
(P_{\bar{s}}) \quad \begin{cases} 
\inf_x q(x) \\
I \leq Ax + \bar{s} \leq u.
\end{cases} \quad \text{and} \quad (P'_{\bar{s}}) \quad \begin{cases} 
\inf_x q(x) \\
Ax + \bar{s} = y \\
I \leq y \leq u.
\end{cases}
\] (7)

Claims clarified below ([26, 5])

- The AL algorithm actually “solves” the closest feasible problem \((P_{\bar{s}})\).
- The speed of convergence is globally linear.

The AL algorithm

Detection of unboundedness \((\text{val}(P) = -\infty)\)

When is the AL algorithm well defined?

Proposition ([5])

For the convex QP (2), the following properties are equivalent:

(i) \( \text{dom} \delta \neq \emptyset \quad (\iff \delta \neq +\infty \iff \delta \in \text{Conv}(\mathbb{R}^m)) \),
(ii) for some/any \( s \in S \), the shifted QP (6) is solvable,
(iii) for some/any \( r > 0 \) and \( \lambda \in \mathbb{R}^m \), the AL subproblem (3) is solvable,
(iv) there is no \( d \in \mathbb{R}^n \) such that \( g^T d < 0 \), \( Hd = 0 \), and \( Ad \in [l, u]^{\infty} \).

- \( C^\infty \) denotes the asymptotic/recession cone of a convex set \( C \).
- A direction like \( d \) in (iv) is called here an unboundedness direction.
- The failure of these conditions can be detected on the first AL subproblem (3), by finding a direction \( d \) such that

\[
g^T d < 0, \quad Hd = 0, \quad \text{and} \quad Ad \in [l, u]^{\infty}.
\]

- Fundamental assumption: (i)-(iv) holds from now on.
The AL algorithm
Convergence for an infeasible QP (val($P$) = $+\infty$)

Feasibility and dual function

- No duality gap:

  \[ \text{the QP is feasible} \iff \delta \text{ is bounded below}. \]

  - [$\implies$] (contrapositive) true for any convex problem by weak duality.
  - [$\impliedby$] (contrapositive) $\delta \not\equiv +\infty$ and $\delta \to -\infty$ along $\vec{s} \neq 0$ ($S$ is closed).

- Consequence for a convex QP:

  the QP is infeasible \implies \delta \text{ is unbounded below}

  \implies \{\lambda_k\} \text{ blows up}

  (by the proximal interpretation).

- One can say more.

The AL algorithm
Convergence for an infeasible QP (val($P$) = $+\infty$)

Level curves of the dual function $\delta$ (infeasible QP, $H \succ 0$)
The AL algorithm
Convergence for an infeasible QP (val($P$) = $+\infty$)

Level curves of the dual function $\delta$ (infeasible QP, $H = 0$)

A surprising identity [5; 2015]

When dom $\delta \neq \emptyset$,

$$S = \mathcal{R}(\partial \delta).$$

- Surprising since
  - $S$ only depends on the constraints of the QP,
  - $\delta$ also depends on the objective of the QP.

- We already know that $S \cap \mathcal{R}(\partial \delta) \neq \emptyset$:

  $$S = [l, u] + \mathcal{R}(A) \ni s_{k+1} := y_{k+1} - Ax_{k+1} \in \partial \delta(\lambda_{k+1}) \subset \mathcal{R}(\partial \delta).$$
The AL algorithm
Convergence for an infeasible QP ($\text{val}(P) = +\infty$)

When $\text{dom} \delta \neq \emptyset$,

$$S = \mathcal{R}(\partial \delta).$$

Proof

- The value function $v(s) := \inf \{q(x) : l \leq Ax + s \leq u, \ x \in \mathbb{R}^n\}$ verifies
  $$\text{dom} \ v = S \quad \text{and} \quad \delta = v^*.$$

- No duality gap: $\text{val}(P'_s) = \text{val}(D'_s)$, which can be written
  $$v = \delta^*.$$

Proof (continued)

- $[S \subset \mathcal{R}(\partial \delta)]$ (Frank-Wolfe and constraint qualification)
  $$s \in S \implies (P'_s) \text{ has a primal-dual solution } ((x_s, y_s), \lambda_s)$$
  $$\implies (x_s, y_s) \in \arg \min \{\ell(x, y, \lambda_s) + s^T \lambda_s : (x, y) \in \mathbb{R}^n \times [l, u]\}$$
  $$\implies (x_s, y_s) \in \arg \min \{\ell(x, y, \lambda_s) : (x, y) \in \mathbb{R}^n \times [l, u]\}$$
  $$\implies s = y_s - Ax_s \in \partial \delta(\lambda_s) \subset \mathcal{R}(\partial \delta).$$

- $[S \supset \mathcal{R}(\partial \delta)]$ ($\delta \neq +\infty$, no duality gap)
  $$s \in \mathcal{R}(\partial \delta) \implies s \in \partial \delta(\lambda) \text{ for some } \lambda$$
  $$\implies \lambda \in \partial \delta^*(s) = \partial v(s)$$
  $$\implies s \in \text{dom} \ v = S.$$
Is the identity \( S = \mathcal{R}(\partial \delta) \) true for an arbitrary convex problem?

For an arbitrary convex function \( \delta \in \text{Conv}(\mathbb{R}^m) \), there holds
\[
\text{ri}(\text{dom } \delta^* ) \subset \mathcal{R}(\partial \delta) \subset \text{dom } \delta^*,
\]
Taking the closure yields
\[
\text{cl dom } \delta^* = \text{cl } \mathcal{R}(\partial \delta).
\]

The identity \( S = \mathcal{R}(\partial \delta) \) holds for a convex QP (with \( \delta \neq +\infty \)) since
\[
\delta^* = v \text{ (no duality gap) (not always true)} \implies \text{cl dom } v = \text{cl } \mathcal{R}(\partial \delta),
\]
\[
\text{dom } v = S \text{ (always true)} \implies \text{cl } S = \text{cl } \mathcal{R}(\partial \delta),
\]
\[
S \text{ is closed (not always true)} \implies S = \mathcal{R}(\partial \delta),
\]
\[
\mathcal{R}(\partial \delta) \text{ is closed (not always true)} \implies S = \mathcal{R}(\partial \delta).
\]
The AL algorithm
Convergence for an infeasible QP ($\text{val}(P) = +\infty$)

**Why $s_k \to \bar{s}$ implies that the AL algorithm solves the CFQP?**

Since

\[
(x, y) \in \arg \min \ell_r(x', y', \lambda) \quad \forall (x', y') \in \mathbb{R}^n \times [l, u]
\]

and $Ax + \bar{s} = y$

imply that $(x, y)$ is a solution to the CFQP.

---

The AL algorithm
Convergence for an infeasible QP ($\text{val}(P) = +\infty$)

**Global linear convergence $s_k \to \bar{s}$ [5]**

$(P_{\bar{s}})$ with solution $\Rightarrow$ the dual solution set of $(P_{\bar{s}})$, namely

\[
\tilde{\mathcal{S}}_D := \{\lambda \in \mathbb{R}^m : \bar{s} \in \partial \delta(\lambda)\}
\]

is nonempty and

\[
\forall \beta > 0, \quad \exists L > 0, \quad \text{dist}_{\tilde{\mathcal{S}}_D}(\lambda_0) \leq \beta \quad \text{implies that} \quad \forall k \geq 1, \quad \|s_{k+1} - \bar{s}\| \leq \frac{L}{r_k} \|s_k - \bar{s}\|.
\]

**Comments:**

- Similar to the solvable case, but with $s_k \nRightarrow s_k - \bar{s}$,
- $\bar{s}$ is not known $\Rightarrow$ more difficult to design an update rule for $r_k$:
  - instead of $s_k - \bar{s}$, observe $s'_k := s_k - s_{k-1} \to 0$ globally linearly.
The AL algorithm

Convergence for an infeasible QP (val($\mathcal{P}$) = $+\infty$)

Proof

Let $\tilde{\lambda} \in \tilde{\mathcal{S}}_D$, $\tilde{\lambda}_k := \lambda_k - r_k \bar{s}$, and subtract $\tilde{\lambda} + r_k \bar{s}$ from the iteration $\lambda_{k+1} = \lambda_k - r_k s_{k+1}$:

$$\lambda_{k+1} = \lambda_k - r_k \left[ \begin{array}{c} s_{k+1} - \bar{s} \\ 0 \end{array} \right] \in \partial \delta(\lambda_{k+1}).$$

- Monotonicity of $\partial \tilde{\delta}(\cdot) = \partial \delta(\cdot) - \bar{s}$:
  $$\forall \tilde{\lambda} \in \tilde{\mathcal{S}}_D : \| s_{k+1} - \bar{s} \| \leq \frac{1}{r_k} \| \tilde{\lambda}_k - \tilde{\lambda} \|.$$  

- $\tilde{\lambda} \in \tilde{\mathcal{S}}_D$ is arbitrary and $-\bar{s} \in \tilde{\mathcal{S}}_D^\infty$:
  $$\| s_{k+1} - \bar{s} \| \leq \frac{1}{r_k} \text{dist}_{\tilde{\mathcal{S}}_D}(\tilde{\lambda}_k) \leq \frac{1}{r_k} \text{dist}_{\tilde{\mathcal{S}}_D}(\lambda_k).$$  

(9)

- Quasi-global error bound (5) on $\tilde{\mathcal{S}}_D$:
  $$\text{dist}_{\tilde{\mathcal{S}}_D}(\lambda_k) \leq L \| s_k - \bar{s} \|.$$  

(10)

(9) and (10) imply (8).

The AL algorithm

The revised AL algorithm

Set $\lambda_0 \in \mathbb{R}^m$, $r_0 > 0$, $\rho'_{\text{des}} \in ]0, 1[$, and repeat for $k = 0, 1, 2, \ldots$

- Compute (if possible, exit with a direction of unboundedness otherwise)
  $$\left( x_{k+1}, y_{k+1} \right) \in \arg \min_{(x, y) \in \mathbb{R}^n \times [l, u]} \ell_{r_k}(x, y, \lambda_k).$$

- Update the multipliers
  $$\lambda_{k+1} = \lambda_k - r_k s_{k+1}, \quad \text{where } s_{k+1} := y_{k+1} - Ax_{k+1}.$$  

- Stop if
  $$A^T(Ax_{k+1} - y_{k+1}) \approx 0 \quad \text{and} \quad P_{[l, u]}(Ax_{k+1}) \approx y_{k+1}.$$  

- Update $r_k \cap r_{k+1} > 0$: $s'_k := s_k - s_{k-1}$, $\rho'_k := \| s'_{k+1} \| / \| s'_k \|$, and
  $$r_{k+1} := \max \left( 1, \frac{\rho'_k}{\rho'_{\text{des}}} \right) r_k.$$
The AL algorithm
Interactions with the SQP algorithm (in progress)

The SQP algorithm

The (LS-qN) SQP algorithm solves the nonlinear optimization problem
\[
\begin{align*}
\inf_x f(x) \\
l \leq c(x) \leq u,
\end{align*}
\]  
(11)
as follows.

- It computes at the current iterate \( x \) the search direction \( d \) by solving the osculating quadratic problem (OQP)

\[
d \in \arg \min_{l' \leq Ad \leq u'} \left( g^T d + \frac{1}{2} d^T H d \right),
\]  
(12)

with \( g \) := \( \nabla f(x) \), \( H \) is a positive definite approximation of the Hessian of the Lagrangian of (11), \( A := c'(x) \), \( l' := l - c(x) \), and \( u' := u - c(x) \).

- Then it computes a \textit{stepsize} \( \alpha > 0 \) along \( d \) in order to decrease a \textit{merit function} and takes as new iterate

\[
x_+ := x + \alpha d.
\]

The AL algorithm
Interactions with the SQP algorithm (in progress)

A classical \textit{merit function} is

\[
x \in \mathbb{R}^n \mapsto \Theta_\sigma(x) = f(x) + \sigma \text{ dist}_{[l, u]}(c(x))
= f(x) + \sigma \|c(x)^\#\|,
\]

where \( \sigma > 0 \) and

\[
[l, u] 
\]

\[
\begin{align*}
\nu^# &= P_{[l, u]} \nu - \nu.
\end{align*}
\]
Using an unboundedness direction

If the closest feasible OQP is infeasible, the AL algorithm can return an unboundedness direction \( d \), i.e., satisfying

\[
g^T d < 0, \quad Hd = 0, \quad \text{and} \quad Ad \in [l', u']\infty.
\]

**Proposition**

Let \( d \) be an unboundedness direction of the closest feasible OQP (14) at \( x \). Then

\[
(\|c(\cdot)^\#\|)'(x; d) \leq 0 \quad \text{and} \quad \Theta'_\sigma(x; d) < 0. \tag{13}
\]

Again, a direction of unboundedness \( d \) of the closest feasible OQP allows the SQP algorithm to make a LS along it.

---

The AL algorithm

Interactions with the SQP algorithm (in progress)

Using a solution to the closest feasible QP

If the OQP is infeasible, the AL algorithm solves instead the closest feasible OQP

\[
d \in \arg\min_{l' \leq Ad + \bar{s} \leq u'} \left( g^T d + \frac{1}{2} d^T Hd \right). \tag{14}
\]

**Proposition (link to make with [3; 1989])**

If \( x \) is not a stationary point of the feasible problem

\[
\left\{ \begin{array}{l}
\inf_y f(y) \\
\quad l \leq c(y) + c(x)^\# \leq u,
\end{array} \right.
\]

if \( \sigma \) is large enough, if \( d \) solves (14), and if \( H \succ 0 \), then

\[
\Theta'_\sigma(x; d) \leq -d^T Hd - \bar{\sigma} \left( \|c(x)^\#\| - \|\bar{s}(x)\| \right) < 0.
\]

Hence a solution \( d \) to the closest feasible osculating QP allows the SQP algorithm to make a LS along it.
Numerical results
The codes Oqla and Qpalm and the selected test-problems

Oqla and Qpalm

Implementation of the revised AL algorithm in two solvers [12], soon freely available at https://who.rocq.inria.fr/Jean-Charles.Gilbert:

- **Oqla**
  - in C++,
  - fast execution, but slow implementation,
  - OO → easy to take into account new data structures, like Ooqp [11] (dense, sparse, ℓ-BFGS, . . .),
  - AL subproblems solved by an active-set (AS) method,
  - more than 1 year of work for one engineer!

- **Qpalm**
  - in Matlab,
  - AL subproblems solved by an AS method,
  - fast implementation, easy to try new ideas, but slow execution.

**Main objective of these tests**: is it worth continuing working on the development of Oqla/Qpalm?

Selected Cutest problems

Comparison made on the Cutest collection of test-problems [15].

- 138 convex quadratic problems (all solvable, but 4?),
- 58 problems among them, with $n \leq 500$ (for comparison in Matlab).

![Histogram of problem dimensions]
Reading performance profiles [9]

Performance profiles drawn with Libopt [13].

Chiche, Gilbert, Joannopoulos

Comparison of Oqla and Qpalm on iteration counters

- Close to each other (see x-axis \([10^{0.05} \approx 1.12]\) and y-axis [even scores]).
- Difference in failures due to the slowness of Qpalm in Matlab (or still not clear).
**Numerical results**

**Performance profiles**

**Comparison of Oqla and Qpalm on CPU time**

![Performance Comparison Graph]

- **Oqla** (in C++) is 10..2000 times faster than **Qpalm** (in Matlab).

Two more codes, which use active-set methods:

- **Quadprog**
  - the standard QP solver of the Matlab optimization toolbox [25],
  - Options ‘Algorithm’ → ’active-set’ and ’LargeScale’ → ‘off’ \(\implies\) active-set method.

- **Qpa**
  - free code,
  - from the **Galahad** library [14],
  - in Fortran,
  - uses preprocessing and preconditioning?
Numerical results
Comparison with active-set methods

Comparison of Qpalm and Quadprog on CPU time

Qpalm is often twice faster than Quadprog (but not always faster).
Qpalm is more robust than Quadprog (81% success to 67%).
Progress is still possible with Qpalm.

Chiche, Gilbert, Joannopoulos

Numerical results
Comparison with active-set methods

Comparison of Oqla and Qpa on CPU time

Qpa is more often faster than Oqla, but not significantly.
Oqla and Qpa have the same robustness (73% and 71% success respectively).
Progress is still possible with Oqla.

Chiche, Gilbert, Joannopoulos
Two more codes, which use interior-point methods:

- **Ooqp**
  - free code,
  - written by Gertz and Wright in 2003 [11],
  - to show the interest of an OO implementation.

- **Qpb**
  - free code,
  - from the Galahad library [14],
  - in Fortran,
  - uses preprocessing and preconditioning?

Comparison of Oqla, Ooqp, and Qpb on CPU time

- IP methods are clearly faster than our AL+AS method (in particular with Ooqp).
- Poor robustness of Ooqp $\implies$ careful implementation yields much improvement?
- Oqla is located between Qpb and Ooqp in terms of robustness.
Behaviors in an SQP framework

- Recall that one iteration of the SQP algorithm computes a PD solution \((d^{QP}, \lambda^{QP})\) of the OQP

\[
\min_{l' \leq Ax \leq u', \lambda' \geq 0} \left( g^T d + \frac{1}{2} d^T H d \right)
\]

and then updates (locally) the PD variables \((x, \lambda)\) by

\[
x_+ := x + d^{QP} \quad \text{and} \quad \lambda_+ := \lambda^{QP}.
\]

- Close to the solution to the nonlinear problem, \(x_+ \simeq x\) and \(\lambda_+ \simeq \lambda\), therefore a good guess of the PD solution to the QP is available:

\((0, \lambda)\).

- Hence, it makes sense to see how the QP solvers behave when the starting point is close to the solution to the QP.

### Numerical results
Comparison with interior-point methods

Oqla vs. Qpb, starting from a primal-dual solution, on CPU time

- Motivation: see whether Oqla can take advantage of a good starting point,
- 64 problems, for which an accurate primal-dual solution has been found,
- Qpb has no warm restart.
Numerical results
Comparison with interior-point methods

Oqla vs. Qpb, starting from a perturbed \((10^{-8})\) primal-dual solution

$$\text{Relative performance 'cpu' (log\(_{10}\) scale)}$$

![Graph showing the comparison between OQLA and QPB starting from a perturbed solution at \(10^{-8}\).]
Numerical results
Comparison with interior-point methods

Oqla vs. Qpb, starting from a perturbed \((10^{-6})\) primal-dual solution

![Graph showing relative performance](image)

Oqla vs. Qpb, starting from a perturbed \((10^{-5})\) primal-dual solution

![Graph showing relative performance](image)
Numerical results
Comparison with interior-point methods

Oqla vs. Qpb, starting from a perturbed \((10^{-4})\) primal-dual solution

![Graph showing relative performance 'cpu' (log scale) for OQLA and QPB starting from a perturbed primal-dual solution.]
Numerical results
Comparison with interior-point methods

Oqla vs. Qpb, starting from a perturbed \((10^{-2})\) primal-dual solution

![Graph showing relative performance 'cpu' (log scale) for OQLA and QPB starting from perturbed solutions.](chart)

Chiche, Gilbert, Joannopoulos
93 / 101

Numerical results
Comparison with interior-point methods

Oqla vs. Qpb, starting from a perturbed \((10^{-1})\) primal-dual solution

![Graph showing relative performance 'cpu' (log scale) for OQLA and QPB starting from perturbed solutions.](chart)

Chiche, Gilbert, Joannopoulos
94 / 101
Numerical results
Comparison with interior-point methods

Oqla vs. Qpb, starting from a perturbed (10^0) primal-dual solution

Chiche, Gilbert, Joannopoulos
95 / 101

Numerical results
Comparison with interior-point methods

Oqla vs. Qpb, starting from a perturbed (10^1) primal-dual solution

Chiche, Gilbert, Joannopoulos
96 / 101
Numerical results

Comparison with interior-point methods

**Oqla vs. Qpb, starting from a perturbed \((10^2)\) primal-dual solution**

![Graph showing relative performance 'cpu' (log scale) for Oqla and Qpb.]

**Conclusion**: for perturbations less than 100 \%, the AL+AS solver Oqla is “more often better” than the IP solver Qpb.

Discussion and future work

**Discussion**

- **Oqla/Qpalm** give interesting answers on infeasible or unbounded QPs.
- **Oqla** and **Qpalm** are not ridiculous, with respect to well established active-set solvers (**Qpa**), and sometimes clearly better (**Quadprog**).
- The present version of **Oqla/Qpalm** is not as efficient as the IP solver **Qpb**, but much more robust than **Ooqp**.
- **Oqla/Qpalm** can take advantage of an estimate of the solution (not the case of the other tested IP solvers) \(\Rightarrow\) nice for SQP.

Still many possible improvements:

- using preprocessing,
- inexact minimization of the AL subproblems (3), while keeping the global linear convergence,
- trying other solvers of the AL subproblems (3), like IP or Newton-min,
- . . . .
Future work

Can one preserve the global linear convergence of the AL algorithm when the AL subproblems (3) are solved inexactly?

Try to use one (a few) interior point step(s) to solve the AL subproblems (3), in order to obtain polynomiality.

Improve nonsmooth methods and use them to solve the AL subproblems (3), in order to gain in efficiency.

Extend the result of Dean and Glowinski [7] to convex inequality constrained QP: for strictly convex QP with the single equality constraint $Ax = b$, the Lagrangian relaxation

$$x_k = \arg \min_{x \in \mathbb{R}^n} q(x) + \lambda_k^T (Ax - b)$$

$$\lambda_{k+1} = \lambda_k + \alpha_k (Ax_k - b),$$

where $\alpha_k$ is chosen as a compact of $]0, 2/\mu[,$ generates iterates that converge globally linearly to the unique solution to the closest feasible problem

$$\left\{ \begin{array}{l}
\inf_x q(x) \\
A^T (Ax - b) = 0.
\end{array} \right.$$
Discussion and future work

The end

Main references


Global linear convergence of an augmented Lagrangian algorithm for solving convex quadratic optimization problems.
Journal of Convex Analysis, 12, 45–69.
[preprint] [editor].

Benchmarking optimization software with performance profiles.
Mathematical Programming, 91, 201–213.
[doi].

M. Frank, P. Wolfe (1956).
An algorithm for quadratic programming.
Naval Research Logistics Quarterly, 3, 95–110.
[doi].

Object-oriented software for quadratic programming.
[doi].

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).

LIBOPT – An environment for testing solvers on heterogeneous collections of problems.
Submitted to ACM Transactions on Mathematical Software.

GALAHAD: a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization.
Technical report, Rutherford Appleton Laboratory, Oxfordshire OX11 0QX.
[internet].
The multiplier method of Hestenes and Powell applied to convex programming.  
[doi].

Augmented Lagrange multiplier functions and duality in nonconvex programming.  

Quadprog (2014).  
Quadratic programming.  
[internet].

A projection method for least-squares solutions to overdetermined systems of linear inequalities.  
*Linear Algebra and its Applications*, 86, 211–236.  
[doi].