Accelerating convergence of a Separable Augmented Lagrangian Algorithm

Arnaud Lenoir, Philippe Mahey

Accelerating convergence of a Separable Augmented Lagrangian Algorithm

Arnaud Lenoir Philippe Mahey

Research Report LIMOS/RR-07-14

22nd October 2007

Abstract

We analyze the numerical behaviour of a separable Augmented Lagrangian algorithm with multiple scaling parameters, different parameters associated with each dualized coupling constraint as well as with each subproblem. We show that an optimal superlinear rate of convergence can be theoretically attained in the twice differentiable case and propose an adaptive scaling strategy with the same ideal convergence properties. Numerical tests performed on quadratic programs confirm that Adaptive Global Scaling subsumes former scaling strategies with one or many parameters.

Keywords: Augmented Lagrangian, Decomposition

Résumé

Nous analysons ici le comportement numérique d'un algorithme de Lagrangien augmenté séparable avec plusieurs paramètres de mise à l'échelle : un par contrainte couplante dualisée et par sous-problème. Nous montrons qu'un taux de convergence superlinéaire peut être obtenu en théorie dans le cas deux fois différentiable et nous proposons une stratégie de mise à jour auto-adaptive des paramètres, qui conserve les propriétes de convergence. Des test numériques effectués sur des programmes quadratiques confirment l'amélioration apportée par rapport à d'autres stratégies de mise à l'echelle avec un ou plusieurs paramètres.

Mots clés : Lagrangien augmenté, Décomposition

Introduction

This work is motivated by the need to improve the performance of SALA, a separable Augmented Lagrangian algorithm proposed by Hamdi et al [HMD97] for solving large-scale decomposable optimization problems.

(SALA) is an extension of the Proximal Decomposition method (see [MOD95]) and belongs to the family of splitting algorithms like the double-backward method of Douglas and Rachford and the Alternate Direction Method of Multipiers (see [Fuk92, Eck89]). It is shown in [MOD95] that the Proximal Decomposition leads to a separable regularization of the dual function which induces in the primal space some block-coordinate Augmented Lagrangian functions. A parameter can naturally be associated with the quadratic terms of the Augmented Lagrangian (denoted hereafter by the *scaling parameter* $\lambda > 0$). The algorithm was shown to be very sensitive to the value of that parameter, turning difficult in practical situations to obtain the best convergence rate.

Furthermore, it was shown in [MDBH00] that, while λ penalizes the primal coupling constraints and greater values will accelerate the primal sequence, λ^{-1} penalizes the dual sequence, so that a compromise value is expected to be optimal. This is the reason why the parameter is no more a penalty parameter like in the classical Augmented Lagrangian method, but a scaling parameter.

A first bound on the rate of convergence of the Douglas-Rachford's algorithm for finding a zero of the sum of two maximal monotone operators was given early by Lions and Mercier [LM79]. This bound was derived too in the scaled version of the Proximal Decomposition method by Mahey et al [MOD95] and then improved when the optimal situation was known to correspond to a compromise between accelerating the primal or the dual sequences [MDBH00]. Complementary results on the theoretical convergence rate of that family of algorithms have been also reported by Luque [Luq84]. In his PhD thesis [Eck89], Eckstein has reported many numerical experimentations on the Alternate Direction Method of Multipliers, exhibiting the characteristic spiralling effect on the primal-dual iterates. Multidimensional scaling was analyzed by Dussault et al [DGM03] but the first experiments in the quadratic case have induced the belief that the convergence rate could not be better than 1/2.

We are interested here in the numerical behaviour of SALA with a new scaling strategy, quoted as the 'Global Scaling', with respect to any coupling constraints as well as to any subproblem. Moreover, practical implementation issues like the iterative adjustment of the scaling parameters are also discussed in order to build a general decomposition algorithm with global scaling.

After introducing SALA and its scaled version with multidimensional scaling, we propose in section 2 a global scaling strategy with different parameters in each subproblems. That strategy, sometimes called 'block-scaling', appeared in former works by Eckstein [Eck94, EF90] or Kontogiorgis and Meyer [KM95], but it has not been formally analyzed from a computational point of view. In order to analyze its convergence properties, we use the formal setting of maximal monotone operators leading to a new version of the Proximal Decomposition method with Global Scaling in section 3. The asymptotic behaviour of the

differentiable case shows that superlinear convergence could be attained at least in theory, thus improving the bound of 1/2 observed in the case of a single parameter. The optimal rate is derived from the gradient of the monotone operator, i.e. from the Hessian matrix when dealing with optimization problems.

An adaptive scaling is introduced in section 7 to accelerate the convergence in the general case. Numerical results described in the last section compare the different updating rules on quadratic minimization problems with linear coupling constraints.

1 Convex equality-constrained separable problems

We are interested in solving the following convex program defined on the product space $X = X_1 \times \ldots \times X_p$, $(X_i = \mathbb{R}^{n_i})$.

$$\left\{ \begin{array}{ll} \text{Minimize} & \displaystyle\sum_{i=1}^{p} f_i(x_i) \\ \forall i, x_i \in X_i & \displaystyle\sum_{i=1}^{p} g_i(x_i) = 0 \end{array} \right. \qquad (P1)$$

where f_i are extended real valued convex functions supposed to be proper and lower-semi-continuous (l.s.c) and g_i are affine:

$$g_i : x_i \mapsto g_i(x_i) = G_i x_i - b_i \qquad (1)$$

Algorithm SALA tackles solving this problem via a decomposition -coordination scheme which involves subproblems with local augmented lagrangian functions to be minimized. Before introducing the method, we first reformulate $(P1)$ with the help of the following subspace of $\mathbb{R}^{mp}$:

$$\mathcal{A} = \left\{ y = \begin{pmatrix} y_1 \\ \vdots \\ y_p \end{pmatrix} \in (\mathbb{R}^m)^p / \sum_{i=1}^{p} y_i = 0 \right\} \qquad (2)$$

referenced to as the *coupling subspace*. If we allocate resource vectors to each term of the coupling constraint $y_i = g_i(x_i)$, or in a shortened way:

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_p \end{pmatrix} = \begin{pmatrix} g_1(x_1) \\ \vdots \\ g_p(x_p) \end{pmatrix} = g(x)$$

we get an embedded formulation of $(P1)$ with a distributed coupling:

$$\left\{ \begin{array}{ll} \text{Minimize} & \displaystyle\sum_{i=1}^{p} f_i(x_i) \\ \forall i = 1, \ldots, p & y = g(x) \\ & y \in \mathcal{A} \end{array} \right. \qquad (P2)$$

2 SALA : A Separable Augmented Lagrangian Algorithm

The augmented Lagrangian function (of parameter $\lambda > 0$) obtained by associating multipliers u_i to allocation constraints $y_i = g_i(x_i)$ is:

$$L_\lambda(x, y; u) = f(x) - \langle u, g(x) - y \rangle + \frac{\lambda}{2} \|g(x) - y\|^2 \tag{3}$$

where $f(x)$ denotes $\sum_{i=1}^{p} f_i(x_i)$. It decomposes into the sum:

$$L_\lambda(x, y; u) = \sum_{i=1}^{p} L_{\lambda,i}(x_i, y_i; u_i) \tag{4}$$

with:

$$L_{\lambda,i}(x_i, y_i; u_i) = f_i(x_i) - \langle u_i, g_i(x_i) - y_i \rangle + \frac{\lambda}{2} \|g_i(x_i) - y_i\|^2 \tag{5}$$

Hypotheses on f and g ensures the existence of a saddle point of L_λ. (référence) Consequently $(P2)$ can be solved through the maxi-minimization of L_λ. The classical method of multipliers finds such a saddle-point by alternating:

- the exact minimization of $(x, y) \mapsto L_\lambda(x, y, u^k)$ over $X \times \mathcal{A}$ giving (x^{k+1}, y^{k+1})
- the multipliers update : $u^{k+1} = u^k - \lambda(g(x^{k+1}) - y^{k+1})$

This method is known not to take profit from the decomposable structure of the problem. So as to exploit the separability of $(P2)$, algorithm SALA (algorithm 1) minimizes successively over x and y in a Gauss-Seidel fashion. The minimization in x then decomposes into the p independent subproblems:

$$\min_{x_i \in X_i} L_{\lambda,i}(x_i, y_i^k; u_i^k) \tag{SP_i}$$

which can be solved in parallel. Suppose now that u^k is in $\mathcal{A}^\perp$ for all k (this fact that will be verified later), the dot product $\langle u^k, y \rangle$ is hence null for any $y \in \mathcal{A}$ and consequently the minimization with respect to y reduces to find the closest vector y^{k+1} in $\mathcal{A}$ to $g(x^{k+1})$ *i-e* the projection :

$$y^{k+1} = \Pi_{\mathcal{A}} g(x^{k+1}) \tag{6}$$

with $\Pi_{\mathcal{A}}$, $\Pi_{\mathcal{A}^\perp}$ denoting the projections on $\mathcal{A}$ and on $\mathcal{A}^\perp$. Since the optimality conditions for $(P2)$ states $u \in \mathcal{A}^\perp$, the final stage consists in a projected subgradient step:

$$u^{k+1} = u^k + \lambda \Pi_{\mathcal{A}^\perp} g(x^{k+1}) \tag{7}$$

where the step length λ is the parameter we used in definition (3). The projection of $g(x^{k+1})$ on $\mathcal{A}$ is equivalent to compute the amount of the coupling constraint violation:

$$r^{k+1} = \sum_{i=1}^{p} g_i(x_i^{k+1}) \tag{8}$$

and then to equitably distribute it among the subproblems. The components of the projected vector are then:

$$y_i^{k+1} = g_i(x_i^{k+1}) - \frac{1}{p} r^{k+1} \tag{9}$$

Observe that if we choose $y^{k=0}$ and $u^{k=0}$ in their mutually orthogonal feasibility subspaces $\mathcal{A}$ and $\mathcal{A}^\perp$ (we can take $y^{k=0} = u^{k=0} = 0$ for instance), then the updating formula (7), (8), (9) will provide sequences $\left\{y^k\right\}_k$ and $\left\{u^k\right\}_k$ staying respectively in $\mathcal{A}$ and $\mathcal{A}^\perp$. The latter subspace has the explicit formulation:

$$\mathcal{A}^\perp = \left\{ u = \begin{pmatrix} u_1 \\ \vdots \\ u_p \end{pmatrix} \in (\mathbb{R}^m)^p / u_1 = \ldots = u_p \in \mathbb{R}^m \right\} \tag{10}$$

We can interpret the fact that we compel u^k to stay in $\mathcal{A}^\perp$ as the wish to obtain the same utility of the shared resource in each of the subproblems. At every iteration k, the knowledge of u^k reduces to the knowledge of its common component value $v^k = u_1^k = \cdots = u_p^k$ in (10) so the update step becomes:

$$v^{k+1} = v^k - \frac{\lambda}{p} r^{k+1} \tag{11}$$

Algorithm 1 SALA : a Separable Augmented Lagrangian Algorithm

Require: $\lambda > 0, \epsilon > 0, y^{k=0} \in \mathcal{A}, v^{k=0} \in \mathbb{R}^m$

repeat
  for all $i = 1, \ldots, p$ **do**
    $x_i^{k+1} :\in \arg\min_{x_i} f_i(x_i) + \frac{\lambda}{2} \left\| g_i(x_i) - y_i^k \right\|^2 - \left\langle v^k, g_i(x_i) \right\rangle$
  end for
  $r^{k+1} \leftarrow \sum_{i=1}^p g_i(x_i^{k+1})$
  for all $i = 1, \ldots, p$ **do**
    $y_i^{k+1} \leftarrow g_i(x_i^{k+1}) - \frac{1}{p} r^{k+1}$
  end for
  $v^{k+1} \leftarrow v^k - \frac{\lambda}{p} r^{k+1}$
  $k \leftarrow k + 1$
until $\left\| g(x^{k+1}) - y^k \right\| < \epsilon$

2.1 Choosing λ

SALA can be derived from the Douglas-Rachford splitting scheme for the sum of two monotone operators [LM79]. In many numerical tests performed on algorithms deriving from this method ([Eck89], [EB90], [Kon94], [MDBH00], [MOD95], [RW91], [SR03]), the performance was observed to heavily depend on the choice of λ. This parameter actually behaves like a scaling parameter between primal and dual sequences, governing the behavior of the algorithm. A

too small (respectively too huge) value of λ will make the sequence too conservative with respect to primal (respectively to dual) information from one iteration to other. In both cases, the resulting algorithm will turn out to be slow. Thus, parameter λ must be tuned so as to balance the progress of primal and dual sequences.

2.2 SALA with multidimensionnal scaling

A multi-dimensionnal scaling version of SALA is presented in [DGM03]. It consists in setting $\lambda = 1$ and to multiply the coupling constraint by an invertible scaling matrix $\overline{M}$. We thus deal with the equivalent problem:

$$\left\{ \begin{array}{ll} \text{Minimize} & \sum_{i=1}^{p} f_i(x_i) \\ \forall i = 1, \ldots, p & z_i = \overline{M} g_i(x_i) \\ & z \in \mathcal{A} \end{array} \right. \qquad (P3)$$

This modifies steps $3, 7, 9$ of algorithm 1, giving algorithm 2.

Algorithm 2 SALAMS: SALA with Multidimensional Scaling

Require: $\overline{M}, \epsilon > 0, y^{k=0} \in \mathcal{A}, v^{k=0} \in \mathbb{R}^m$
1: **repeat**
2: **for all** $i = 1, \ldots, p$ **do**
3: $x_i^{k+1} :\in \arg\min_{x_i} f_i(x_i) + \frac{1}{2} \left\| \overline{M} g_i(x_i) - z_i^k \right\|^2 - \left\langle v^k, \overline{M} g_i(x_i) \right\rangle$
4: **end for**
5: $r^{k+1} \leftarrow \sum_{i=1}^{p} g_i(x_i^{k+1})$
6: **for all** $i = 1, \ldots, p$ **do**
7: $z_i^{k+1} \leftarrow \overline{M} \left(g_i(x_i^{k+1}) - \frac{1}{p} r^{k+1} \right)$
8: **end for**
9: $v^{k+1} \leftarrow v^k - \frac{1}{p} r^{k+1}$
10: $k \leftarrow k + 1$
11: **until** $\forall i = 1, \ldots, n \quad \left\| \overline{M} g_i(x_i^{k+1}) - z_i^k \right\| < \epsilon$

The use of a matrix $\overline{M}$ provides more freedom for primal-dual balancing. However, this scaling must be the same in every subproblems. In the next section, we present an extension of this scaling technique to allow the parameter to be different in each subproblem.

3 Global scaling of SALA

3.1 Scaled allocations

Instead of allocating $y_i = \overline{M} g_i(x_i)$ with a common matrix $\overline{M}$ for all subproblems in $(P3)$, we can introduce matrices M_i different from each other for every i. We

thus use scaled allocations $z_i = M_i g_i(x_i)$ or $z = Mg(x)$ with M denoting the invertible bloc diagonal scaling matrix:

$$M = \begin{pmatrix} M_1 & & \\ & \ddots & \\ & & M_p \end{pmatrix} \tag{12}$$

To override the effect of this scaling, the concatenated scaled allocation vector z has now to live in a subspace depending on M:

$$\mathcal{A}_M = \left\{ z = \begin{pmatrix} z_1 \\ \vdots \\ z_p \end{pmatrix} \in (\mathbb{R}^m)^p / \sum_{i=1}^{p} M_i^{-1} z_i = 0 \right\} \tag{13}$$

and we get a new equivalent formulation of $(P1)$:

$$\begin{cases} \text{Minimize} & \sum_{i=1}^{p} f_i(x_i) \\ \forall i = 1, \ldots, p & z_i = M_i g_i(x_i) \\ & z \in \mathcal{A}_M \end{cases} \tag{P4}$$

Following the same approach as in SALAMS, we can write the augmented lagrangian function (of parameter $\lambda = 1$) obtained by associating a multiplier w to the scaled allocation constraint $z = Mg(x)$:

$$L_M(x, z; w) = f(x) - \langle w, Mg(x) - z \rangle + \frac{1}{2} \|Mg(x) - z\|^2 \tag{14}$$

which also decomposes into a sum:

$$L_M(x, z; w) = \sum_{i=1}^{p} L_{i,M_i}(x_i, z_i; w_i) \tag{15}$$

with:

$$L_{i,M_i}(x_i, z_i; w_i) = f_i(x_i) - \langle w_i, M_i g_i(x_i) - z_i \rangle + \frac{1}{2} \|M_i g_i(x_i) - z_i\|^2 \tag{16}$$

3.2 Algorithm

Applying the same Gauss-Seidel technique as in SALA and SALAMS, we obtain algorithm 3. The minimisation in x still consists in solving independant subproblems:

$$\min_{x_i} L_{i,M_i}(x_i, z_i^k; w_i^k) \tag{SP_i'}$$

The minimization with respect to z will reduce to the projection :

$$z^{k+1} = \Pi_{\mathcal{A}_M} Mg(x^{k+1}) \tag{17}$$

and the multiplier update will be:

$$w^{k+1} = w^k + \Pi_{\mathcal{A}_M^\perp} M g(x^{k+1}) \tag{18}$$

To project onto $\mathcal{A}_M$ we use the fact that $\mathcal{A}_M = ker(N)$ and $\mathcal{A}_M^\perp = Im(N^\top)$ with:

$$N = \left(M_1^{-1} | \ldots | M_p^{-1}\right)$$

Consequently, the projectors onto $\mathcal{A}_M$ and $\mathcal{A}_M^\perp$ respectively are:

$$\Pi_{\mathcal{A}_M} = I - N^\top (NN^\top)^{-1} N \tag{19}$$

$$\Pi_{\mathcal{A}_M^\perp} = N^\top (NN^\top)^{-1} N \tag{20}$$

The inner matrix $(N^\top N)^{-1}$ is given by $(\sum_{i=1}^p M_i^{-1} M_i^{-\top})^{-1}$ so we can compute the projection v of a vector $\tilde{v}$ by the following procedure:

$$r = \sum_{i=1}^p M_i^{-1} \tilde{v}_i \tag{21a}$$

$$\forall i \quad v_i = M_i^{-T} (\sum_{i'=1}^p M_{i'}^{-1} M_{i'}^{-\top})^{-1} r \tag{21b}$$

Finally, if v is the projection of $\tilde{v}$ onto $\mathcal{A}_M^\perp$ then $(v - \tilde{v})$ is the projection onto $\mathcal{A}_M$.

Algorithm 3 SALAGS : SALA with Global Scaling

Require: M_i invertible $i = 1, \ldots, p; \epsilon > 0; z^0 \in \mathcal{A}_M; w^0 \in \mathcal{A}_M^\perp$

repeat
  for all $i \in \{1, \ldots, p\}$ **do**
    $x_i^{k+1} :\in \arg\min_{x_i} f_i(x_i) - \langle w_i, M_i g_i(x_i) - z_i \rangle + \frac{1}{2} \|M_i g_i(x_i) - z_i\|^2$
  end for
  $r^{k+1} \leftarrow \sum_i g_i(x_i^{k+1})$
  for all $i \in \{1, \ldots, p\}$ **do**
    $s_i^{k+1} \leftarrow M_i^{-\top} (\sum_{i'=1}^p M_{i'}^{-1} M_{i'}^{-\top})^{-1} r^{k+1}$
    $z_i^{k+1} \leftarrow M_i g_i(x_i^{k+1}) - s_i^{k+1}$
    $w_i^{k+1} \leftarrow w_i^k + s_i^{k+1}$
  end for
  $k \leftarrow k + 1$
until $\|Mg(x) - z\| < \epsilon$

The main difference between SALAGS and SALAMS (or SALA) is that the orthogonal subspace where the optimal multipliers live is

$$\mathcal{A}_M^\perp = \left\{ w = \begin{pmatrix} w_1 \\ \vdots \\ w_p \end{pmatrix} \in (\mathbb{R}^m)^p / M_1^\top w_1 = \ldots = M_p^\top w_p \right\} \tag{22}$$

and consequently, each component of vector w will be different from each other. However, we have not to store all of the w_i but only $v = M_1^\top w_1 = \ldots = M_p^\top w_p$. We thus operate the change of variable $u = M^\top w \in \mathcal{A}^\perp$ and just work with the common component v as before. In the same spirit, we can also set $y = M^{-1}z$ so as to work with the real allocations i-e vectors corresponding directly to $g_i(x_i)$ instead of $M_i g_i(x_i)$. This notably simplifies the algorithmic procedure. Letting $\Lambda_i = M_i^\top M_i$, we then obtain algorithm 4. Note that since M_i is invertible, Λ_i must now be chosen symetric positive definite.

Algorithm 4 SALAGS : SALA with Global Scaling (reformulation)

Require: $\Lambda_i; i = 1, \ldots, p; \epsilon > 0; y^0 \in \mathcal{A}; v^0 \in \mathbb{R}^m$

repeat
  for all $i \in \{1, \ldots, p\}$ **do**
    $x_i^{k+1} :\in \arg\min_{x_i} f_i(x_i) + \left\|g_i(x_i) - y_i^k\right\|_{\Lambda_i}^2 - \left\langle v^k, g_i(x_i)\right\rangle$
  end for
  $r^{k+1} \leftarrow \sum_i g_i(x_i^{k+1})$
  for all $i \in \{1, \ldots, p\}$ **do**
    $y_i^{k+1} \leftarrow g_i(x_i^{k+1}) - \Lambda_i^{-1}(\sum_{i'=1}^p \Lambda_{i'}^{-1}) r^{k+1}$
  end for
  $v^{k+1} \leftarrow v^k - (\sum_{i'=1}^p \Lambda_{i'}^{-1}) r^{k+1}$
  $k \leftarrow k + 1$
until $\left\|g_i(x_i^{k+1}) - y_i^k\right\|_{\Lambda_i}^2 < \epsilon$

Obviously, SALAGS generalizes SALA and SALAMS in the sense they respectively are obtained by the choices $\Lambda = \lambda I$ and:

$$\Lambda = \begin{pmatrix} \overline{M}^\top \overline{M} & & \\ & \ddots & \\ & & \overline{M}^\top \overline{M} \end{pmatrix}$$

4 Study of a quadratic case

We analyze in this section the very special case of the quadratic programming version of $(P1)$ when G_i are square and invertible and f_i are quadratic functions $x_i \mapsto \frac{1}{2} {x_i}^\top Q_i x_i + {q_i}^\top x_i$ where Q_i are $n_i \times n_i$ symmetric positive-definite matrices, $q_i \in \mathbb{R}^{n_i}$:

$$\text{Minimize} \qquad \sum_{i=1}^p \frac{1}{2} {x_i}^\top Q_i x_i + {q_i}^\top x_i \tag{23a}$$

$$\sum_{i=1}^p G_i x_i = \sum_{i=1}^p b_i \tag{23b}$$

In this case, the convergence is linear and we theoretically always are able to obtain the convergence rate of $\frac{1}{2}$.

We denote by Q and q the bloc-diagonal matrix and the vector:

$$Q = \begin{pmatrix} Q_1 & & \\ & \ddots & \\ & & Q_p \end{pmatrix}, q = \begin{pmatrix} q_1 \\ \vdots \\ q_p \end{pmatrix}$$

so that the objective function becomes:

$$x \mapsto \frac{1}{2} x^\top Q x + q^\top x$$

4.1 Direct analysis of the iteration matrix

For this problem, one iteration of SALAGS reduces to applying an affine mapping $\mathfrak{I}_M$ the spectral radius of which determines the asymptotic rate of convergence of the algorithm. We derive an upper bound for this rate by decomposing $\mathfrak{I}_M$ into a product of three matrices.

Subproblems in step 3 of algorithm 3 take the form:

$$\begin{aligned} x^{k+1} :\in \arg\min_x \quad & \frac{1}{2} \langle x, Qx \rangle + \langle q, x \rangle \\ & - \left\langle w^k, M(Gx - b) - z^k \right\rangle + \frac{1}{2} \left\| M(Gx - b) - z^k \right\|^2 \end{aligned}$$

the solution of which is:

$$x^{k+1} = (Q + G^\top M^\top M G)^{-1} \left[G^\top M^\top (w^k + z^k + Mb) - q \right]$$

The corresponding optimal resource consumption is:

$$\tilde{z}^{k+1} = M(Gx^{k+1} - b)$$

We can rewrite it with respect to w^k and z^k:

$$\tilde{z}^{k+1} = R_M(w^k + z^k) + K \tag{24}$$

with $R_M = MG(Q + G^\top M^\top M G)^{-1} G^\top M^\top$ and K is a constant depending on the problem data. To simplify notations, we set:

$$T = G^{-\top} Q G^{-1}$$

we then have:

$$R_M = (I + M^{-\top} T M^{-1})^{-1}$$

We now introduce the cost sensitivity:

$$\tilde{w}^{k+1} = w^k + z^k - \tilde{z}^{k+1} \tag{25}$$

Indeed, vector $\tilde{w}^{k+1}$ is such that:

$$G^\top M^\top \tilde{w}^{k+1} = Qx^{k+1} + q \tag{26a}$$

$$= \nabla(\frac{1}{2}\langle x, Qx\rangle + \langle q, x\rangle)(x^{k+1}) \tag{26b}$$

We remark that:

$$z^{k+1} = \Pi_{\mathcal{A}_M} \tilde{z}^{k+1} \tag{27a}$$

$$w^{k+1} = \Pi_{\mathcal{A}_M^\perp} \tilde{w}^{k+1}. \tag{27b}$$

Moreover, from (24) and (25), we have:

$$\tilde{w}^{k+1} = (I - R_M)(w^k + z^k) - K \tag{28}$$

We now are able to combine (24),(27),(28) to formulate the recursive formula giving $(\tilde{z}^{k+1}, \tilde{w}^{k+1})$ from $(\tilde{z}^k, \tilde{w}^k)$:

$$\begin{pmatrix} \tilde{z}^{k+1} \\ \tilde{w}^{k+1} \end{pmatrix} = \mathfrak{I}_M \begin{pmatrix} \tilde{z}^k \\ \tilde{w}^k \end{pmatrix} + \begin{pmatrix} K_w \\ K_z \end{pmatrix}$$

where $\mathfrak{I}_M$ is given by:

$$\mathfrak{I}_M = \underbrace{\begin{pmatrix} R_M & 0 \\ 0 & (I - R_M) \end{pmatrix}}_{S_M} \underbrace{\begin{pmatrix} I & I \\ I & I \end{pmatrix}}_{\Sigma} \underbrace{\begin{pmatrix} \Pi_{\mathcal{A}_M} & 0 \\ 0 & \Pi_{\mathcal{A}_M^\perp} \end{pmatrix}}_{\Pi} \tag{29}$$

and K_w and K_z are constant vectors depending on the problem data. The matrix R_M is bloc diagonal:

$$R_M = \begin{pmatrix} R_{M,1} & & \\ & \ddots & \\ & & R_{M,p} \end{pmatrix} = \begin{pmatrix} (I + M_1^{-\top} T_1 M_1^{-1})^{-1} & & \\ & \ddots & \\ & & (I + M_p^{-\top} T_p M_p^{-1})^{-1} \end{pmatrix}$$

Each of the three matrices in the product (29) correspond to one stage in the algorithm:

- Π operates the projection on $\mathcal{A}_M \times \mathcal{A}_M^\perp$
- Σ computes the subproblems input by adding primal and dual variables
- S_M is the linear part of the application which solves subproblems

The asymptotic convergence rate of the algorithm is equal to the spectral radius $\rho(\mathfrak{I}_M)$ of $\mathfrak{I}_M$. So, if we want to accelerate the convergence, we might choose M as a minimizer of this spectral radius. We can derive from relation (29) the upper bound:

$$\rho(\mathfrak{I}_M) \leq \rho(S_M) \underbrace{\rho(\Sigma.\Pi)}_{=1} \leq \rho(S_M) \tag{30}$$

that we should minimize with respect to M.

Let remind that S_M is the bloc-diagonal matrix:

$$S_M = \begin{pmatrix} (R_{M,1}) & & & & & & \\ & \ddots & & & & & \\ & & (R_{M,p}) & & & & \\ & & & (I - R_{M,1}) & & & \\ & & & & \ddots & & \\ & & & & & (I - R_{M,p}) \end{pmatrix}$$

with

$$R_{M,i} = (I + M_i^{-\top} G_i Q_i^{-1} G_i^{\top} M_i^{-1})^{-1} \quad \forall i = 1, \ldots, p$$

The trace of S_M, is equal to 1 whatever M. Moreover, if μ is an eigenvalue of S_M and (s_z, s_w) an associated eigenvector, then $(1 - \mu)$ is also an eigenvalue associated to (s_w, s_z). The minimization of the spectral radius will then consist in a compromise between minimizing eigenvalues of R_M and the ones of $(I - R_M)$.

The better situation will occur when $sp(R_M) = \{0.5\}$ (in this case, we will also have $sp(I - R_M) = \{0.5\}$) furnishing the optimal value $\rho(S_M) = \max(\rho(R_M), \rho(I - R_M)) = 0.5$. As soon as one eigenvalue becomes lower than 0.5, another one will necessary become greater than 0.5, increasing the spectral radius of S_M.

This "optimal" case only happens when $sp\left(M^{-\top} T M^{-1}\right) = \{1\}$. Under our study hypotheses T is symetric positive definite, so it admits a square root $T^{\frac{1}{2}}$. If we choose $M = T^{\frac{1}{2}}$ *i-e* $\Lambda = T$, we obtain $sp\left(M^{-\top} T M^{-1}\right) = sp(I) = \{1\}$

5 A more general setting : PDM

Problem $(P2)$ can be embedded in a more general class of monotone inclusion problems. Algorithm SALA then appears as a by-product of a more general algorithm ([MOD95]) namely the proximal decomposition method (PDM). We first present the class of problems in question (31) and the way to transform $(P2)$ in (31). Then we remind algorithm PDM and we give a bound on the asymptotic convergence rate in the differentiable case.

5.1 Finding an orthogonal primal-dual couple in the graph of a maximal monotone operator

Let $\mathcal{H}$ be an Hilbert space, of norm and inner product denoted by $\|.\|$ and $\langle ., . \rangle$. A multivalued operator $T : \mathcal{H} \rightrightarrows \mathcal{H}$ is monotone if:

$$\forall y, y' \in \mathcal{H}, \forall u \in T(y), \forall u' \in T(y') \quad \langle y - y', u - u' \rangle \geq 0$$

Let $\mathcal{A}$ be a closed subspace of $\mathcal{H}$. We consider the problem of finding a couple (y, u) such that

$$y \in \mathcal{A} \tag{31a}$$
$$u \in \mathcal{A}^{\perp} \tag{31b}$$
$$u \in T(y) \tag{31c}$$

These conditions arise as optimality condition of optimization and variationnal inequalities problems when the primal variable is constrained to live in $\mathcal{A}$.

We associate to T the following couple of operators:

$$P = (I + T)^{-1}$$
$$Q = (I + T^{-1})^{-1}$$

Both P and Q are monotone and firmly non-expansive, hence single-valued. Moreover, the application:

$$\mathcal{H} \to \mathcal{H} \times \mathcal{H}$$
$$s \mapsto (P(s), Q(s))$$

is one-to-one from $\mathcal{H}$ to $gr(T)$. These operators also have the property that $P + Q = I$ and $N = 2P - I = I - 2Q = P - Q$ is non-expansive (*i-e* Lipschitz with modulus 1).

5.2 Writing problem $(P2)$ as a monotone inclusion problem

Let now transform problem $(P2)$ under the form of inclusion problem (31), we can define the convex implicit function F which gives the optimal cost for a given resource allocation y:

$$F(y) = \min_{x} f(x) \text{ s.t } g(x) = y \tag{32}$$

The original problem $(P1)$ is now equivalent to minimize F over the coupling subspace $\mathcal{A}$ defined by (2) *i-e* :

$$\text{Minimize} \quad F(y) \tag{33a}$$
$$\text{s.t} \quad y \in \mathcal{A} \tag{33b}$$

still equivalent to find (y, u) such that:

$$(y, u) \in gr(\partial F) \tag{34a}$$
$$(y, u) \in \mathcal{A} \times \mathcal{A}^{\perp} \tag{34b}$$

where $gr(\partial F)$ refers to the graph of the maximal monotone operator ∂F. Now, if F is proper l.s.c, (it is true for instance when $dom(f)$ is bounded or when f is coercive ([Roc70], theorem 9.2)) then ∂F is maximal monotone ([Roc70], corolary 31.5.2).

5.3 Algorithm

The proximal decomposition method aims at solving (31) by applying the following scheme:

1. Choose arbitrarly $s^0 \in \mathcal{H}$
2. Compute $(\tilde{y}^{k+1}, \tilde{u}^{k+1}) = (P(s^k), Q(s^k))$
3. Compute $(y^{k+1}, u^{k+1}) = (\Pi_{\mathcal{A}}(\tilde{y}^k), \Pi_{\mathcal{A}^\perp}(\tilde{u}^k))$
4. Set $s^{k+1} = y^{k+1} + u^{k+1}$ and go to 2

The operator $\Pi_{\mathcal{A}} P + \Pi_{\mathcal{A}^\perp} Q$ which maps s^k to s^{k+1} is firmly non-expansive of full domain ([MOD95]). Consequently s^k weakly converges to a solution s^* of (31) for any starting value s^0.

If we apply this algorithm to T and $\mathcal{A}$ as defined in subsection 5.2, we then obtain algorithm SALA.

Remark 5.1. *We also could have defined:*

$$\tilde{F}(x,y) = \begin{cases} f(x), & \text{if } g(x) = y \\ +\infty, & \text{else.} \end{cases} \tag{35}$$

The problem then results in minimizing $\tilde{F}$ over $\mathbb{R}^n \times \mathcal{A}$ the orthogonal of which is $\{0_{\mathbb{R}^n}\} \times \mathcal{A}^\perp$. We obtain an additional proximal term in variable x like in [Eck94].

5.4 Convergence rate in the differentiable case

We suppose that T is differentiable at $\overline{y} \in \mathcal{H}$ *i-e* $T(\overline{y})$ is a singleton and there is a linear continuous transformation $\nabla T(\overline{y})$ such that:

$$T(y) \subset T(\overline{y}) + \nabla T(\overline{y})(y - \overline{y}) + o(\|y - \overline{y}\|)B$$

Let the algorithm converge to $s^* = y^* + u^*$. If T is differentiable at y^*. Then P and Q are also differentiable and:

$$\begin{aligned} \nabla P(s^*) &= (I + \nabla T(y^*))^{-1} \\ \nabla Q(s^*) &= I - (I + \nabla T(y^*))^{-1} \\ \nabla (P - Q)(s^*) &= 2(I + \nabla T(y^*))^{-1} - I \end{aligned}$$

Using the fact that $2P - I = I - 2Q = P - Q$ we rewrite:

$$\begin{aligned} \Pi_{\mathcal{A}} P + \Pi_{\mathcal{A}^\perp} Q &= \frac{1}{2} I + \frac{1}{2} (\Pi_{\mathcal{A}} - \Pi_{\mathcal{A}^\perp})(P - Q) \\ &= \frac{1}{2} I + \frac{1}{2} R_{\mathcal{A}} (P - Q) \end{aligned}$$

where $R_{\mathcal{A}} = \Pi_{\mathcal{A}} - \Pi_{\mathcal{A}^\perp}$ is the reflection through space $\mathcal{A}$. so:

$$s^{k+1} = \frac{1}{2}s^k + \frac{1}{2}R_{\mathcal{A}}(P-Q)s^k \tag{36}$$

Now, as s^* is a fixed point of this operator, we can write:

$$\begin{aligned} s^{k+1} - s^* &= \frac{1}{2}(s^k - s^*) + \frac{1}{2}R_{\mathcal{A}}(P-Q)s^k - \frac{1}{2}R_{\mathcal{A}}(P-Q)s^* \\ &= \frac{1}{2}(s^k - s^*) + \frac{1}{2}R_{\mathcal{A}}\left((P-Q)s^k - (P-Q)s^*\right) \end{aligned}$$

Using the fact that $(P-Q)$ is differentiable at s^*, we have:

$$(P-Q)s^k - (P-Q)s^* \in \nabla(P-Q)(s^*)(s^k - s^*) + o(\left\|s^k - s^*\right\|)B$$

Denoting $e^k = s^k - s^*$, we obtain:

$$\begin{aligned} e^{k+1} &\in \frac{1}{2}e^k + \frac{1}{2}R_{\mathcal{A}}\nabla(P-Q)(s^*)e^k + o(\left\|e^k\right\|)B \\ \left\|e^{k+1}\right\| &\leq \frac{1}{2}\left\|(I + R_{\mathcal{A}}\nabla(P-Q)(s^*))\right\|\left\|e^k\right\| + o(\left\|e^k\right\|) \end{aligned}$$

Finally when $s^k \to s^*$,

$$\lim_{k\to\infty}\sup\frac{\left\|e^{k+1}\right\|}{\left\|e^k\right\|} \leq \frac{1}{2}\left\|(I + R_{\mathcal{A}}\nabla(P-Q)(s^*))\right\| \tag{37}$$

By formula $\nabla(P-Q)(s^*) = 2(I+\nabla T(y^*))^{-1} - I$, we notice that if $\nabla T(y^*) = I$ (if $T = \partial f$, it means that f locally behaves like $\frac{1}{2}\left\|.\right\|^2$ around y^*) then convergence rate is lower than $\frac{1}{2}$. The value $\frac{1}{2}$ actually corresponds to the relaxation factor used in recursion (36). Super-linear convergence can therefore be achieved in this case if we set this value to 0 (which consists in applying Peaceman-Rachford scheme instead of the Douglas-Rachford one in [LM79]) or if we make it tend to 0:

$$\begin{aligned} s^{k+1} &= \alpha^k s^k + (1-\alpha^k)R_{\mathcal{A}}(P-Q)s^k \\ \alpha^k &\to 0 \end{aligned}$$

6 Scaling of the proximal decomposition method

We now show that SALAGS can also be generalized in the setting of monotone inclusion and that we are always (at less theoretically) able to obtain a linear convergence rate of $\frac{1}{2}$.

6.1 Principle

Let come back to our problem which consists in finding a couple (y, u) such that:

$$u \in T(y) \tag{38a}$$
$$(y, u) \in \mathcal{A} \times \mathcal{A}^{\perp} \tag{38b}$$

Let M be an invertible matrix. We operate the change of variable:

$$z \leftarrow My$$
$$w \leftarrow M^{-\top} u$$

We denote $\mathcal{A}_M = M\mathcal{A}$ and $\mathcal{A}_M^{\perp} = M^{-\top}\mathcal{A}^{\perp}$. So the problem is equivalent to find $M^{\top} w \in T(M^{-1} z)$ with $(z, w) \in \mathcal{A}_M \times \mathcal{A}_M^{\perp}$. But operator:

$$U = M^{-\top} T(M^{-1}.) \tag{39}$$

is maximal monotone so we get a scaled version of original problem (38) which keeps an identical structure *i-e* find (z, w) such that:

$$w \in U(z) \tag{40a}$$
$$(z, w) \in \mathcal{A}_M \times \mathcal{A}_M^{\perp} \tag{40b}$$

6.2 Algorithm

Applying PDM to this formulation gives algorithm 5:

Algorithm 5 GSPDM : Globally Scaled PDM

Require: $(y^0, u^0) \in \mathcal{A} \times \mathcal{A}^{\perp}$, M invertible.

1: $(z^k, w^k) = (My^k, M^{-\top} u^k)$.

2: find $(\tilde{z}^{k+1}, \tilde{w}^{k+1})$ such that:

$$\tilde{w}^{k+1} \in U(\tilde{z}^{k+1})$$
$$\tilde{z}^{k+1} + \tilde{w}^{k+1} = z^k + w^k$$

3: compute $(z^{k+1}, w^{k+1}) = \Pi_{\mathcal{A}_M \times \mathcal{A}_M^{\perp}}(\tilde{z}^{k+1}, \tilde{w}^{k+1})$.

4: $(y^{k+1}, u^{k+1}) = (M^{-1} z^k, M^{\top} w^k)$.

In term of original variables, step 2 is equivalent to find $(\tilde{y}^{k+1}, \tilde{u}^{k+1})$ such that:

$$\tilde{u}^{k+1} \in T(\tilde{y}^{k+1}) \tag{41a}$$
$$M\tilde{y}^{k+1} + M^{-\top}\tilde{u}^{k+1} = My^k + M^{-\top} u^k \tag{41b}$$

Combining (41a) and (41b) gives $\tilde{y}^{k+1}$ as the unique solution of:

$$\begin{aligned} My^k + M^{-\top}u^k &\in (M + M^{-\top}T)(\tilde{y}^{k+1}) \\ 0 &\in T(\tilde{y}^{k+1}) + M^\top M(\tilde{y}^{k+1} - y^k) - u^k \end{aligned}$$

Remark that steps 3 and 4 can be written:

$$\begin{pmatrix} y^{k+1} \\ u^{k+1} \end{pmatrix} = \begin{pmatrix} M^{-1}P_{\mathcal{A}_M}M & \\ & M^\top P_{\mathcal{A}_M^\perp}M^{-\top} \end{pmatrix} \begin{pmatrix} \tilde{y}^{k+1} \\ \tilde{u}^{k+1} \end{pmatrix} \tag{42}$$

which is different from projecting directly onto $\mathcal{A} \times \mathcal{A}^\perp$.

6.3 Global convergence of SALAGS

Proposition 6.1. *If F is proper l.s.c, then the sequence $\left\{(y^k, u^k)\right\}_k$ generated by algorithm 4 converges to some $(\overline{y}, \overline{u})$ satisfying* (40).

Proof. As noticed earlier, algorithm 4 is GSPDM applied to $T = \partial F$ with definition (2) of $\mathcal{A}$. If F is proper l.s.c, then so is $F(M\cdot)$ and $U = \partial(F(M\cdot))$ is therefore maximal monotone. From ([MOD95], $p459$) we obtain the convergence of $\left\{(z^k, w^k)\right\}_k$ generated by PDM to solve (40) and consequently the convergence of $\left\{(y^k, u^k)\right\}_k$ generated by GSPDM to solve (38). □

6.4 Convergence rate

If T is differentiable at $\overline{y}$, then $U = M^{-\top}TM^{-1}$ is differentiable at $\overline{z} = M\overline{y}$ and:

$$\nabla U(\overline{z}) = M^{-\top}\nabla T(\overline{y})M^{-1}$$

so the convergence rate for the sequence $\left\{(z^k, w^k)\right\}_k$ is given by (37):

$$\lim_{k\to\infty} \sup \frac{\left\|(z^{k+1}, w^{k+1}) - (z^*, w^*)\right\|}{\|(z^k, w^k) - (z^*, w^*)\|} \leq \frac{1}{2}\left\|\left(I + R_{\mathcal{A}}(2(I + M^{-\top}\nabla T(y^*)M^{-1})^{-1} - I)\right)\right\|$$

Consequently, if y^* is unique, then the knowledge of $\nabla T(y^*)$ and the choice $M = (\nabla T(y^*))^{\frac{1}{2}}$ will make the algorithm applied to the scaled problem to converge with a linear rate lower than $\frac{1}{2}$.

7 Variable scaling matrix

We study in this section the method when the matrix M is allowed to change at each iteration according to a sequence $\{M_k\}_k$ converging to an invertible limit M *i-e* algorithm 6:

Algorithm 6 Variable metric GSPDM

Require: $(y^0, u^0) \in \mathcal{A} \times \mathcal{A}^\perp$, $\{M_k\}_k \to M$

1: $(z^k, w^k) := (M_k y^k, M_k^{-\top} u^k)$
2: $s^k := z^k + w^k$
3: Compute $(\tilde{z}^{k+1}, \tilde{w}^{k+1}) := (P_{M_k}(s^k), Q_{M_k}(s^k))$
4: Compute $(\bar{z}^{k+1}, \bar{w}^{k+1}) := (\Pi_{\mathcal{A}_{M_k}} \tilde{z}^{k+1}, \Pi_{\mathcal{A}^\perp_{M_k}} \tilde{w}^{k+1})$.
5: $(y^{k+1}, u^{k+1}) := (M_k^{-1} z^k, M_k^\top w^k)$.
6: $k := k + 1$
7: Go to 1

Variable scaling parameter have already been introduced in the litterature in Douglas-Rachford splitting based methods. For instance in [KM95], a variable scaling matrix is employed but the difference between two successive matrices must be positive semi-definite from some rank onwards. In [HYW00], [MDBH00] or [HLW03] an assumption on the speed of convergence of the scaling parameter is needed. However, only the case of a one-dimensional scaling parameter is treated.

We propose here to use a sequence of matrix scaling parameter. We will prove the convergence under hypothesis similar to the one in [HLW03], namely:

Assumption 7.1. *Setting $\Lambda_k = M_k^\top M_k$, the sequence of scaling matrices $\{\Lambda_k\}_k$ converges and:*

$$\sum_{k=0}^{+\infty} \|\Lambda_{k+1} - \Lambda_k\| < +\infty$$

7.1 Convergence analysis of global scaling with a variable metric

We focus on sequences:

$$s^k = z^k + w^k$$
$$\bar{s}^k = \bar{z}^k + \bar{w}^k$$

generated by algorithm 6. Let (y^*, u^*) be any solution of (31) and $\{\sigma_k\}_k$ the sequence defined by:

$$\sigma^k = M_k y^* + M_k^{-\top} u^*$$

We have for all k:

$$\bar{s}^k = J_{M_k} s^k$$
$$\sigma^k = J_{M_k} \sigma^k$$

so firm-nonexpansiveness of J_{M_k} gives:

$$\left\|\bar{s}^k - \sigma^k\right\|^2 \leq \left\|s^k - \sigma^k\right\|^2 - \left\|\bar{s}^k - s^k\right\|^2 \tag{43}$$

The following lemma gives an asymptotic relation between s^{k+1} and $\bar{s}^k$ as M_k converges.

Proposition 7.1. *Under assumption 7.1, there is a sequence $\{\mu_k\}_k$ such that for all k:*

$$(1-\mu_k)\left\|s^{k+1}-\sigma^{k+1}\right\|^2 \leq \left\|\bar{s}^k-\sigma^k\right\|^2 \tag{44}$$

and:

$$\sum_{k=0}^{+\infty}\mu_k = S < +\infty$$

Proof. We have in the one hand:

$$\begin{aligned}\left\|s^{k+1}-\sigma^{k+1}\right\|^2 &= \left\|M_{k+1}(\bar{y}^k-y^*)+M_{k+1}^{-\top}(\bar{u}^k-u^*)\right\|^2\\ &= \left\langle \bar{y}^k-y^*, \Lambda_{k+1}(\bar{y}^k-y^*)\right\rangle + \left\langle \bar{u}^k-u^*, \Lambda_{k+1}^{-1}(\bar{u}^k-u^*)\right\rangle\end{aligned}$$

because $\bar{y}^k-y^*\in\mathcal{A}$ and $\bar{u}^k-u^*\in\mathcal{A}^\perp$ and in the other hand:

$$\begin{aligned}\left\|\bar{s}^k-\sigma^k\right\|^2 &= \left\|M_k(\bar{y}^k-y^*)+M_k^{-\top}(\bar{u}^k-u^*)\right\|^2\\ &= \left\langle \bar{y}^k-y^*, \Lambda_k(\bar{y}^k-y^*)\right\rangle + \left\langle \bar{u}^k-u^*, \Lambda_k^{-1}(\bar{u}^k-u^*)\right\rangle\end{aligned}$$

Substracting the second term from the first one gives:

$$\begin{aligned}&\left\|s^{k+1}-\sigma^{k+1}\right\|^2 - \left\|\bar{s}^k-\sigma^k\right\|^2\\ &= \left\langle \bar{y}^k-y^*, (\Lambda_{k+1}-\Lambda_k)(\bar{y}^k-y^*)\right\rangle + \left\langle \bar{u}^k-u^*, (\Lambda_{k+1}^{-1}-\Lambda_k^{-1})(\bar{u}^k-u^*)\right\rangle\\ &= \left\langle \bar{y}^k-y^*, \Lambda_{k+1}\Lambda_{k+1}^{-1}(\Lambda_{k+1}-\Lambda_k)(\bar{y}^k-y^*)\right\rangle\\ &\qquad + \left\langle \bar{u}^k-u^*, \Lambda_{k+1}^{-1}\Lambda_{k+1}(\Lambda_{k+1}^{-1}-\Lambda_k^{-1})(\bar{u}^k-u^*)\right\rangle\\ &\leq \left\|\Lambda_{k+1}^{-1}(\Lambda_{k+1}-\Lambda_k)\right\| \left\langle \bar{y}^k-y^*, \Lambda_{k+1}(\bar{y}^k-y^*)\right\rangle\\ &\qquad + \left\|\Lambda_{k+1}(\Lambda_{k+1}^{-1}-\Lambda_k^{-1})\right\| \left\langle \bar{u}^k-u^*, \Lambda_{k+1}(\bar{u}^k-u^*)\right\rangle\\ &\leq \mu_k \left\|s^{k+1}-\sigma^{k+1}\right\|^2\end{aligned}$$

where $\mu_k = \max(\left\|\Lambda_{k+1}^{-1}(\Lambda_{k+1}-\Lambda_k)\right\|, \left\|\Lambda_{k+1}(\Lambda_{k+1}^{-1}-\Lambda_k^{-1})\right\|)$. Therefore:

$$\begin{aligned}\mu_k &\leq \left\|\Lambda_{k+1}^{-1}(\Lambda_{k+1}-\Lambda_k)\right\|\\ &\leq \left\|\Lambda_{k+1}^{-1}\right\| \left\|\Lambda_{k+1}-\Lambda_k\right\|\end{aligned}$$

and:

$$\begin{aligned}\mu_k &\leq \left\|\Lambda_{k+1}(\Lambda_{k+1}^{-1}-\Lambda_k^{-1})\right\|\\ &\leq \left\|(\Lambda_k-\Lambda_{k+1})\Lambda_k^{-1}\right\|\\ &\leq \left\|\Lambda_k^{-1}\right\| \left\|\Lambda_{k+1}-\Lambda_k\right\|\end{aligned}$$

as, $\left\{\left\|\Lambda_k^{-1}\right\|\right\}_k$ converges, it is consequently bounded and as $\mu_k \geq 0$, it follows that $\sum_{k=0}^{\infty}\mu_k$ converges. □

Lemma 7.1. *If assumption 7.1 holds, then the sequence* $\left\{s^k\right\}_k$ *generated by algorithm 6 is bounded.*

Proof. Combining (43) and (44), we obtain for all k:

$$(1-\mu_k)\left\|s^{k+1}-\sigma^{k+1}\right\|^2 \leq \left\|s^k-\sigma^k\right\|^2 \tag{45}$$

There is a rank K and a constant $\alpha > 0$ such that the partial product $\prod_{k=K}^{l}(1-\mu_k) \geq \alpha$. Using recursively (45) from K until l we obtain:

$$\left(\prod_{k=K}^{l}(1-\mu_k)\right)\left\|s^l-\sigma^l\right\|^2 \leq \left\|s^K-\sigma^K\right\|^2$$
$$\left\|s^l-\sigma^l\right\|^2 \leq \alpha^{-1}\left\|s^K-\sigma^K\right\|^2$$

hence, the sequence $\left\{\left\|s^k-\sigma^k\right\|^2\right\}_k$ is bounded. Moreover, since M_k converges, so does $\left\{\sigma^k\right\}_k$ which entails boundedness of $\left\{s^k\right\}_k$. □

Proposition 7.2. *If assumption 7.1 holds, the sequence* $\left\{s^k\right\}_k$ *generated by algorithm 6 converges to a fixed point of* J_M*.*

Proof. Lemma 7.1 provides the existence of cluster points for the sequence $\left\{s^k\right\}_k$. Combining (43) and (44) gives for all k:

$$(1-\mu_k)\left\|s^{k+1}-\sigma^{k+1}\right\|^2 \leq \left\|s^k-\sigma^k\right\|^2 - \left\|\bar{s}^k-s^k\right\|^2$$

Summing up this relation until K, we get:

$$\sum_{k=0}^{K}\left\|\bar{s}^k-s^k\right\|^2 \leq \left\|s^0-z^0\right\|^2 - \left\|s^K-\sigma^K\right\|^2 + \sum_{k=0}^{K}\mu_k\left\|s^{k+1}-\sigma^{k+1}\right\|^2$$

every term of the right hand side is bounded, hence the left hand side series is bounded and $\left\|\bar{s}^k-s^k\right\|^2 \to 0$. So every cluster point $(\bar{s}^\infty, s^\infty)$ of $(\bar{s}^k, s^k)$ satisfies $\bar{s}^\infty = s^\infty$. Moreover $(\bar{s}^k, s^k) \in gr(J_k)$ for all k, and J_{M_k} graphically converges to J_{M_∞} so $(\bar{s}^\infty, s^\infty) \in gr(J_{M_\infty})$ and cluster points of $\left\{s^k\right\}_k$ are fixed point of J_{M_∞}.

Let (y^∞, u^∞) the solution associated to a cluster point s^∞. The sequence $\sigma^k = M_k y^\infty + M_k^{-\top} u^\infty$ converges to s^∞. Let $\epsilon > 0$ and K a rank such that:

i) $\left\{\sigma^k\right\}_{k\geq K} \subset B(s^\infty, \epsilon)$

ii) $s_K \in B(s^\infty, \epsilon)$

iii) $\prod_{k=K}^{l}(1-\mu_k) \geq \alpha > 0$ for all $l \geq K$

We have by i) and ii):

$$\left\|s^K - \sigma^K\right\| \leq 2\epsilon$$

using recursively (45) from K to $l \geq K$, we get:

$$\prod_{k=K}^{l} (1-\mu_k) \left\|s^l - \sigma^l\right\|^2 \leq \left\|s^K - \sigma^K\right\|^2 \leq 4\epsilon^2$$

by iii), so:

$$\left\|s^l - \sigma^l\right\| \leq \frac{2\epsilon}{\sqrt{\prod_{k=K}^{L}(1-\mu_k)}} \leq \frac{2\epsilon}{\sqrt{\alpha}}$$

and:

$$\begin{aligned}\left\|s^l - s^\infty\right\| &\leq \left\|s^l - \sigma^l\right\| + \left\|\sigma^l - s^\infty\right\| \\ &\leq \left\|s^l - \sigma^l\right\|^2 + \epsilon \\ &\leq (1 + \frac{2}{\sqrt{\alpha}})\epsilon\end{aligned}$$

That is true for all $\epsilon > 0$ so $s^k \to s^\infty$. □

7.2 Adaptive scaling strategy

We propose in this section a family of strategies to accelerate the convergence of SALA which is based on observations made in the differentiable case in a previous section but applies in the general case.

Indeed, if the involved operator T were differentiable, we would be able to gain knowledge about $\nabla T(y^*)$ as the algorithm progresses by using intermediate guesses $\tilde{y}^k$ and $\tilde{u}^k$ which satisfy :

$$\tilde{u}^k = T(\tilde{y}^k)$$

The differences between two iterates thus satisfies:

$$\tilde{u}^{k+1} - \tilde{u}^k = \nabla T(\tilde{y}^k)(\tilde{y}^{k+1} - \tilde{y}^k) + o(\left\|\tilde{y}^{k+1} - \tilde{y}^k\right\|)$$

The idea is to use a matrix update so as to obtain a sequence of matrices approaching $\nabla T(y^*)$ as $y^k \to y^*$.

The procedures we propose in what follows consist in updating the scaling matrix at each iteration by taking a convex combination (of coefficient $0 < \alpha_k \leq 1$) of the current matrix Λ_k with an other matrix D_k we hope to be close to $\nabla T(\tilde{y}^k)$.

$$\Lambda_{k+1} = (1-\alpha_k)\Lambda_k + \alpha_k D_k \tag{46}$$

In order that the sequence $\left\{\Lambda^k\right\}_k$ satisfies assumption 7.1, we will impose the series of term α_k and D_k to satisfy hypothesis of the following lemma:

Lemma 7.2 (Convergence of adaptive scaling sequences)**.** *Let* $\{\Lambda_k\}_k$ *a sequence of matrices satisfying:*

$$\Lambda_{k+1} = (1-\alpha_k)\Lambda_k + \alpha_k D_k \tag{47}$$

Let denote $0 < d_1^k \leq \ldots \leq d_m^k$ *the* m *eigenvalues of* D_k*. If there exists* $0 < \underline{d} \leq \bar{d} < +\infty$ *such that:*

$$\underline{d} \leq d_1^k \leq \ldots \leq d_m^k \leq \bar{d}$$

and if the series of term $\alpha_k \geq 0$ *converges:*

$$\sum_{k=0}^{+\infty} \alpha_k = S < +\infty$$

then $\{\Lambda_k\}_k$ *is positive definite and converges to a positive definite limit, moreover:*

$$\sum_{k=0}^{+\infty} \|\Lambda_{k+1} - \Lambda_k\| = S' < +\infty.$$

Proof. Without loss of generality, we can choose $[\underline{d}, \bar{d}]$ containing the eigenvalues of Λ_0. Then, the compact convex set $\mathcal{B}$ of symmetric matrices eigeinvalues of which are in $[\underline{d}, \bar{d}]$ contains the sequence $\{D_k\}_k$ along with Λ_0. As the update formula (47) consists in making convex combinations in $\mathcal{B}$, the whole sequence $\{\Lambda_k\}_k$ is in $\mathcal{B}$ and every Λ_k is positive definite as well as the possible limit. Moreover, there exists a constant $C > 0$ such that for every k:

$$\begin{aligned}
\Lambda_{k+1} - \Lambda_k &= -\alpha_k \Lambda_k + \alpha_k D_k \\
\|\Lambda_{k+1} - \Lambda_k\| &\leq \alpha_k \left(\|\Lambda_k\| + \|D_k\|\right) \\
&\leq C\alpha_k
\end{aligned}$$

so, we get $\sum_{k=0}^{+\infty} \|\Lambda_{k+1} - \Lambda_k\| = S' \in \mathbb{R}$ and as a consequence, the convergence of $\{\Lambda_k\}_k$. □

Remark 7.1. *We could also have defined the recursion:*

$$\Lambda_{k+1} = \Lambda_k^{(1-\alpha_k)} D_k^{\alpha_k} \tag{48}$$

in place of (47)*. Using the same arguments, we obtain that the sequence* $\{\ln(\Lambda_k)\}_k$ *is in the set of matrices eigenvalues of which are in* $[\ln(\underline{d}), \ln(\bar{d})]$ *obtaining the positive definiteness of* $\{\Lambda_k\}_k$ *and of its limit. Then, using the fact that the logarithm is Lipschitz on every compact set, we also obtain the absolute convergence.*

7.2.1 Single parameter update

If we restrict ourselves to a single real parameter *i-e* $\Lambda_k = \lambda^k I$, one way to use second order information is to take profit directly of the ratio:

$$\gamma^k = \frac{\left\|\tilde{u}^{k+1} - \tilde{u}^k\right\|}{\left\|\tilde{y}^{k+1} - \tilde{y}^k\right\|}$$

In the general case, we have no idea about the boundedness of $\{\gamma_k\}_k$, so we can consider an interval $[\underline{\gamma}, \bar{\gamma}]$ and use:

$$D_k = (\Pi_{[\underline{\gamma},\bar{\gamma}]}\gamma^k)I \tag{49}$$

where $\Pi_{[\underline{\gamma},\bar{\gamma}]}$ represents the projection onto $[\underline{\gamma}, \bar{\gamma}]$.

7.2.2 Subproblem parameter update

We can apply the same updating policy but separately in each of the subproblems *i-e* for $i = 1, \ldots, p$, we set:

$$\gamma_i^{k+1} = \frac{\left\|\tilde{u}_i^{k+1} - \tilde{u}_i^k\right\|}{\left\|\tilde{y}_i^{k+1} - \tilde{y}_i^k\right\|} \tag{50}$$

and D_k is the diagonal matrix:

$$D_k = \begin{pmatrix} (\Pi_{[\underline{\gamma},\bar{\gamma}]}\gamma_1^k)I_m & & \\ & \ddots & \\ & & (\Pi_{[\underline{\gamma},\bar{\gamma}]}\gamma_p^k)I_m \end{pmatrix}$$

Remark 7.2. *If T_i is strongly monotone of modulus a_i and lipschitz with constant L_i then:*

$$a_i \leq \frac{\left\|\tilde{u}_i^{k+1} - \tilde{u}_i^k\right\|}{\left\|\tilde{y}_i^{k+1} - \tilde{y}_i^k\right\|} \leq L_i$$

and

$$\min_i a_i \leq \frac{\left\|\tilde{u}^{k+1} - \tilde{u}^k\right\|}{\left\|\tilde{y}^{k+1} - \tilde{y}^k\right\|} \leq \max_i L_i$$

therefore, the sequences γ^k and γ_i^k defined in (49) *and* (50) *are automatically bounded and we can relinquish projection onto $[\underline{\gamma}, \bar{\gamma}]$.*

7.2.3 Component update

An other updating policy involving as many parameters as the dimension of the problem consists in computing for all $i = 1, \ldots, p, j = 1, \ldots, m$ the ratio:

$$\gamma_{i,j}^{k+1} = \frac{\left\|(\tilde{u}_i^{k+1})_j - (\tilde{u}_i^k)_j\right\|}{\left\|(\tilde{y}_i^{k+1})_j - (\tilde{y}_i^k)_j\right\|}$$

Matrix D_k will then be the diagonal matrix:

$$D_k = \begin{pmatrix} \Pi_{[\underline{\gamma},\bar{\gamma}]}\gamma_{1,1}^k & & & & & & \\ & \ddots & & & & & \\ & & \Pi_{[\underline{\gamma},\bar{\gamma}]}\gamma_{1,m}^k & & & & \\ & & & \ddots & & & \\ & & & & \Pi_{[\underline{\gamma},\bar{\gamma}]}\gamma_{p,1}^k & & \\ & & & & & \ddots & \\ & & & & & & \Pi_{[\underline{\gamma},\bar{\gamma}]}\gamma_{p,m}^k \end{pmatrix}$$

8 Some examples

We now present some numerical results obtained on small instances of problem (23). We show the relevance of scaling techniques on some basic examples.

8.1 First example : on the utility of multidimensionnal scaling

Let consider the following example

$$Q_1 = Q_2 = \begin{pmatrix} 1 & 0 \\ 0 & 100 \end{pmatrix}$$
$$c_1 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, c_2 = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$$
$$G_1 = G_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$
$$b_1 = b_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

We have:

$$T_1 = T_2 = \begin{pmatrix} 1 & 0 \\ 0 & 0,01 \end{pmatrix}$$

which involves:

$$(R_\lambda)_1 = (R_\lambda)_2 = \begin{pmatrix} \frac{1}{1+\lambda} & 0 \\ 0 & \frac{1}{1+0,01\lambda} \end{pmatrix}$$

We have $sp\,(S_\lambda) = \left\{ \frac{1}{1+\lambda}, \frac{1}{1+0.01\lambda} \right\}$ so the spectral radius of the iteration operator cannot go beyond $\frac{10}{11}$. This optimal upper bound is attained for the value $\lambda^\sharp = 10$ (see figure 1).

If we now allow Λ to be a diagonal positive definite matrix:

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

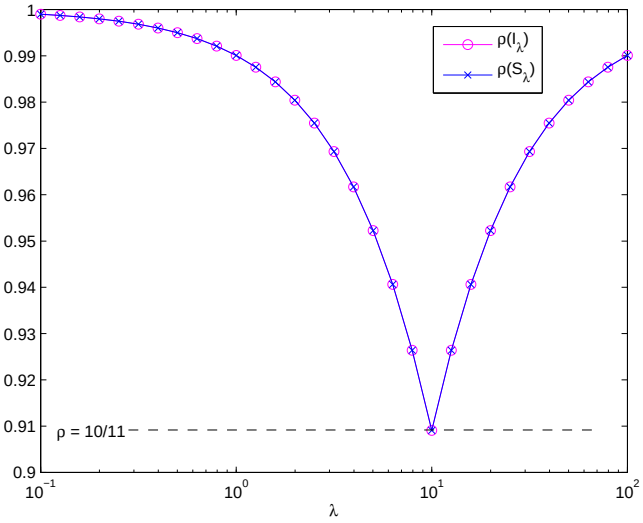

Figure 1: Spectral radius of S_λ with respect to λ in example 1

We now get:

$$sp\left(S_\Lambda\right) = \left\{\frac{1}{1+\lambda_1}, \frac{1}{1+0.01\lambda_2}\right\}$$

While with a single parameter, the best convergence rate is $\frac{10}{11}$ (see figure 1). Here, we can attain the value $\frac{1}{2}$ by choosing $\lambda_1 = 1$ et $\lambda_2 = 100$.

8.2 Second example: on the utility of subproblems scaling

Let consider the following example:

$$Q_1 = \begin{pmatrix} 100 & 50 \\ 50 & 100 \end{pmatrix}, Q_2 = \begin{pmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & 1 \end{pmatrix}$$
$$c_1 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, c_2 = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$$
$$G_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, G_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$
$$b_1 = b_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Figure 2 shows the number of iterations with different scaling values (λ_1, λ_2) and the corresponding single parameter case obtained by enforcing $\lambda_1 = \lambda_2$.

We launched SALA with and without the implementation of an updating rule. Figure 3 compares the number of iterations needed to converge in both

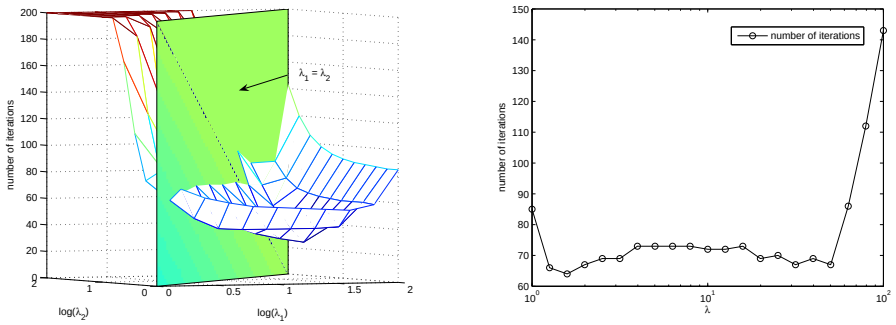

Figure 2: Number of iterations with and without problem scaling on example 2.

cases, for several starting values of $\lambda^{k=0}$. Figure 4 shows the behavior of the corresponding sequences $(\lambda)_k$. The rule implemented here consists in taking:

$$\begin{cases} \alpha^k \equiv 1 & \text{if } mod(k,3) = 0; \\ \alpha^k \equiv 0 & \text{else.} \end{cases}$$

which clearly do not satisfy hypothesis of lemma 7.2.

8.3 A third example

Let consider this third example:

$$Q_1 = \begin{pmatrix} 8 & 4 \\ 4 & 6 \end{pmatrix}, Q_2 = \begin{pmatrix} 30 & 10 \\ 10 & 20 \end{pmatrix}, Q_3 = \begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix}$$
$$c_1 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, c_2 = \begin{pmatrix} 3 \\ 4 \end{pmatrix}, c_3 = \begin{pmatrix} 5 \\ 6 \end{pmatrix}$$
$$G_1 = G_2 = G_3 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$
$$b_1 = b_2 = b_3 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

We lanched SALA with and without auto-adaptive strategies with $\alpha_k = k^{-\frac{10}{9}}$. Number of iterations for several starting values are reported on figures 5 and 6. The first plot (figure 5) is obtained with additive updating rule while the second one (figure 6) is obtained with logarithmic updating rule. Algorithm was stopped when $\left\|\tilde{u}^{k+1} - u^k\right\|^2 + \left\|\tilde{y}^{k+1} - y^k\right\|^2 < 3.10^{-5}$. We observ that additive updating rules are less efficient when initial scaling parameters are too much grater than ideal values.

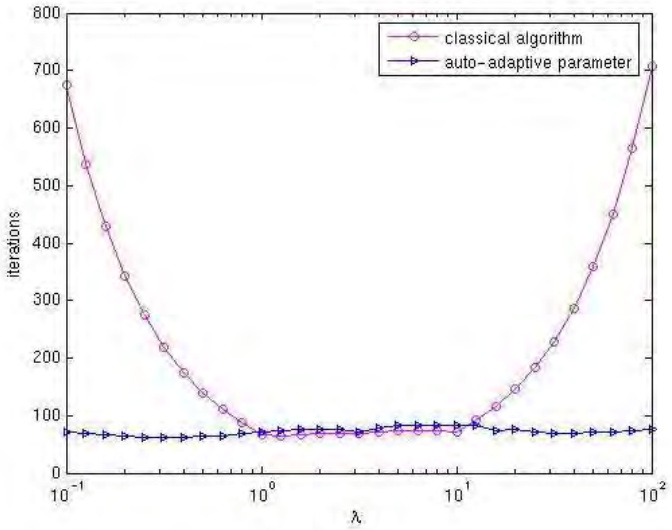

Figure 3: Number of iterations of SALA with and without updating λ

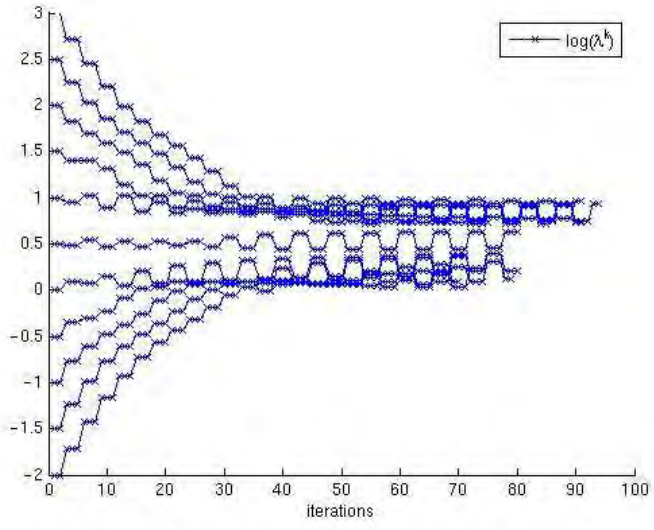

Figure 4: Trajectories of $(\lambda_k)_k$ for several starting values

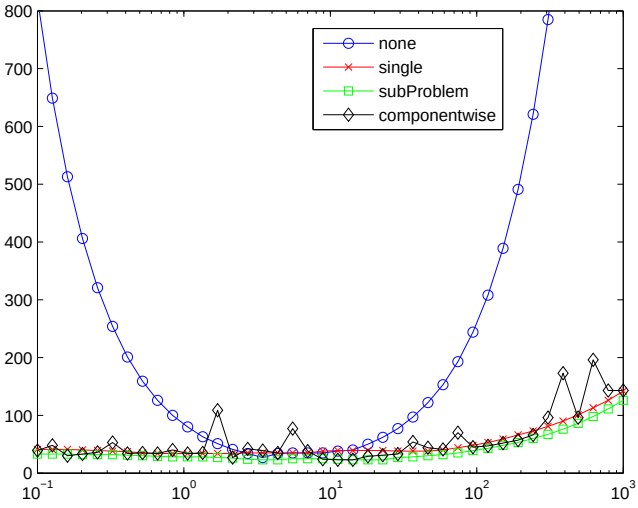

Figure 5: Additive updating rules in example 3

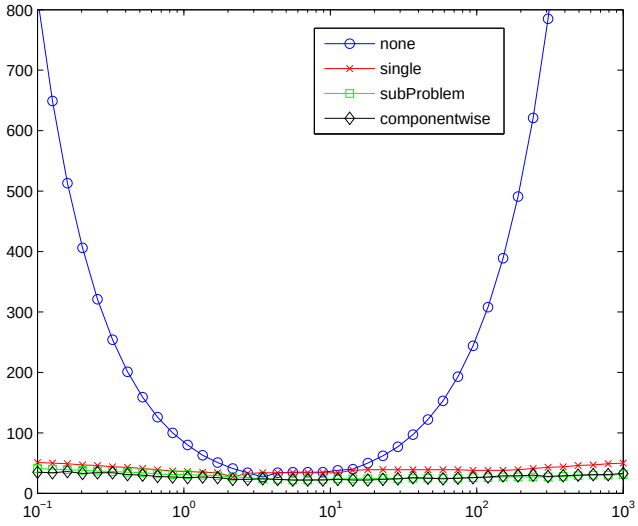

Figure 6: Logarithmic updating rules in example 3

9 Numerical results

9.1 Experimentations

We generated quadratic problems for different values of p and m and we compared performances of the method with and without updating heuristics. We take $p \in [2, 5, 10, 20, 40]$ and $m \in [5, 10, 20, 50, 100]$ and $n_i = m$. Coefficients of b and c are drawn log-uniformly in $[-100, -0.01] \cup [0.01, 100]$. Q is contructed by generating a matrix P and a vector p of coefficients drawn log-uniformly respectively in $[-10, -0.1] \cup [0.1, 10]$ and $[0.1, 1]$, and by setting $Q = P^\top P + diag(p)$. Updates are performed 1 iteration out of 3.

We stop the algorithm when $\left\|\tilde{y}^{k+1} - y^k\right\|^2 + \left\|\tilde{u}^{k+1} - u^k\right\|^2$ is lower than $p.10^{-3}$ or when the iteration number exceeds 3000.

9.2 Results

We used $\Lambda^{k=0} = \lambda^0 I$ as a starting value for λ^0 ranging in $[10^{-4}, 100]$. Table 1 gives the best number of iterations obtained for these values. We observe that BFGS update gives the faster convegence. Other updating rules do not improve significantly convergence with respect to the classical algorithm. We can also note that the coupling update is not competitive when the size of the problem increases.

Unfortunately, BFGS update has the same drawback as the original method : it heavily depends on the original scaling parameter. Table 2 gives the standard deviation of the number of iterations with respect to the initial value λ^0. To illustrate this fact, we plotted for each updating rule the number of iterations with respect to the starting value $\lambda^{k=0}$ in the case $p = 20$ and $m = 10$, on figure 7.

Moreover, updating separately in each subproblem gives a better convergence (for an equivalent computational cost) when the number of subproblems is large. The reason is that the subproblem-update approaches the matrix $\nabla T(y^*)$ with block second-order information which are averaged in the single-parameter update.

References

[DGM03] Jean-Pierre Dussault, Oumar Mandione Guèye, and Philippe Mahey, *Separable augmented lagrangian algorithm with multidimensional scaling for monotropic programming*, Journal of Optimization Theory and Application **127** (2003), 1–20.

[EB90] Jonathan Eckstein and Dimitri P. Bertsekas, *An alternating direction method for linear programming*, Tech. report, Laboratory for Information and Decision Sciences, MIT, April 1990.

		Mises à jour				
p	m	without	**Single**	**Subproblem**	**Coupling**	**BFGS**
2	5	55	43	45	39	35
	10	112	116	116	78	42
	20	56	63	66	60	39
	50	69	62	62	63	47
	100	74	79	79	77	55
5	5	45	51	51	44	24
	10	62	64	71	57	32
	20	61	63	61	82	44
	50	83	88	83	80	50
	100	81	103	102	101	57
10	5	129	75	74	53	23
	10	81	70	66	73	33
	20	89	91	87	119	41
	50	96	141	84	132	53
	100	103	190	111	161	63
20	5	52	67	51	53	22
	10	67	72	68	76	28
	20	82	86	87	117	36
	50	107	149	107	1487	49
	100	117	300	122	3000	64
40	5	56	50	39	52	22
	10	55	69	59	65	24
	20	69	74	71	82	31
	50	102	230	111	2673	42
	100	119	299	135	3000	54

Table 1: Best iteration number.

		Mises à jour				
p	m	without	**Single**	**Subproblem**	**Coupling**	**BFGS**
2	5	720.21	6.58	6.74	11.20	74.21
	10	790.36	31.76	41.99	19.00	207.00
	20	723.18	7.90	8.58	11.54	200.37
	50	800.90	5.14	5.19	9.48	222.67
	100	915.96	5.51	5.37	8.97	335.24
5	5	833.13	7.15	9.21	8.75	665.61
	10	870.71	11.37	8.91	8.91	437.60
	20	743.84	10.26	10.13	.	591.45
	50	851.43	7.93	7.83	10.54	548.15
	100	749.94	7.00	6.72	10.35	433.48
10	5	885.91	7.97	8.41	6.16	504.62
	10	662.29	38.72	21.81	8.53	456.35
	20	831.13	37.54	18.11	21.54	450.76
	50	801.91	24.51	23.47	353.45	577.97
	100	841.89	45.24	21.79	.	647.38
20	5	729.93	5.93	6.99	6.01	659.09
	10	692.33	11.10	6.66	67.06	628.24
	20	839.47	31.05	8.70	347.06	644.39
	50	821.78	66.33	8.87	.	561.73
	100	835.67	223.10	27.31	.	475.39
40	5	861.50	7.31	7.05	7.97	460.00
	10	807.06	9.66	8.47	164.98	549.61
	20	705.92	15.02	9.67	752.24	574.56
	50	885.72	41.58	16.02	.	580.81
	100	668.82	189.17	16.75	.	591.20

Table 2: Standard deviation of the number of iteration w.r.t $\log \lambda^0$.

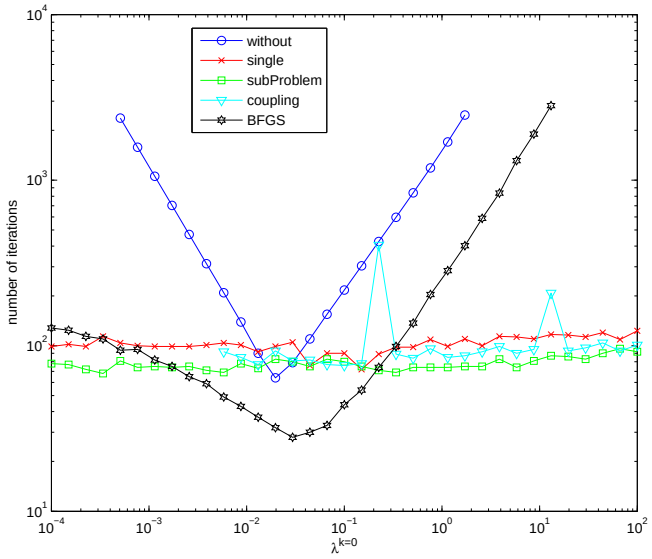

Figure 7: Number of iterations w.r.t $\lambda^{k=0}$ for different updating rules. Case $p = 20$, $m = 10$.

[Eck89] Jonathan Eckstein, *Splitting methods for monotone operators with applications to parallel optimization*, Ph.D. thesis, Massachusetts institute of technology, cambridge, June 1989.

[Eck94] ———, *Some saddle-function splitting methods for convex programming*, Optimization Methods and Software **4** (1994), 75–83.

[EF90] Jonathan Eckstein and Michael C. Ferris, *Operator splitting methods for monotone affine variational inequalities, with a parallel application to optimal control*, Tech. report, Laboratory for Information and Decision Sciences, MIT, April 1990.

[Fuk92] M. Fukushima, *Application of the alternating directions method of multipliers to separable convex programming problems*, Computational Optimization and Applications **1(1)** (1992), 83–111.

[HLW03] B.S He, L-Z Liao, and S.L Wang, *Self-adaptive operator splitting methods for monotone variational inequalities*, Numerische Mathematik **94** (2003), 715–737.

[HMD97] A. Hamdi, P. Mahey, and J.P Dussault, *A new decomposition method in nonconvex programming via a separable augmented lagrangian*, Lecture Notes in Economics and Mathematical Systems **452** (1997), 90–104.

[HYW00] B.S He, H. Yang, and S.L Wang, *Alternating directions method with self-adaptive penalty parameters for monotone variational inequalities*, Journal of Optimization Theory and applications **106** (2000), 349–368.

[KM95] Spyridon Kontogiorgis and Robert R. Meyer, *A variable-penalty alternating directions method for convex optimization*, Tech. Report MP-TR-1995-18, University of Wisconsin Computer sciences department, 1995.

[Kon94] Spyridon. A. Kontogiorgis, *Alternating directions methods for the parallel solution of large-scale block-structured optimization problems*, Ph.D. thesis, University of Wisconsin - Madison, 1994.

[LM79] P.L. Lions and B. Mercier, *Splitting algorithms for the sum of two nonlinear operators*, SIAM Journal on Numerical Analysis **16** (1979), 964–979.

[Luq84] Fernando Ravier Luque, *Asymptotic convergence analysis of the proximal point algorithm*, SIAM J.Control and optimization **22** (1984), 277–293.

[MDBH00] Philippe Mahey, Jean-Pierre Dussault, Abdelhamid Benchakroun, and Abdelouahed Hamdi, *Adaptive scaling and convergence rates of a separable augmented lagrangian algorithm*, Lecture Notes in Economics and Mathematical Systems **481** (2000), 278–287.

[MOD95] Philippe Mahey, Said Oualibouch, and Pham Din Tao, *Proximal decomposition on the graph of a maximal monotone operator*, SIAM J. Optimization **5** (1995), 454–466.

[Roc70] Rockafellar, *Convex analyis*, Princeton University Press, 1970.

[RW91] R. Tyrrel Rockafellar and Roger. J-B Wets, *Scenarios and policy aggregation in optimization under uncertainty*, Mathematics of Operations Research **16** (1991), 119–147.

[SR03] David. H Salinger and R. Tyrrel Rockafellar, *Dynamic splitting:an algorithm for deterministic and stochastic multiperiod optimization*, Working Paper (2003).