GLOBAL AND ADAPTIVE SCALING IN A SEPARABLE AUGMENTED LAGRANGIAN ALGORITHM

ARNAUD LENOIR AND PHILIPPE MAHEY*

Abstract. In this paper, 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.

Key words. Augmented Lagrangian, Decomposition.

AMS subject classifications. 65K05, 90C48, 90C30

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 method of Douglas and Rachford [LM79] 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

*Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes, (L.I.M.O.S), I.S.I.M.A - Université Blaise-Pascal, Clermont-Ferrand, France. e-mail:Philippe.Mahey@isima.fr

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 4 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. Algorithm SALA for convex separable problems. We are interested in solving the following convex program defined on the product space $\mathbb{R}^n = \mathbb{R}^{n_1} \times \ldots \times \mathbb{R}^{n_p}$ (we will use the notation $\langle \cdot, \cdot \rangle$ to denote the dot product in the corresponding finite dimension space).

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

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:

$$\begin{aligned} g_i : \mathbb{R}^{n_i} &\to \mathbb{R}^m \\ x_i &\mapsto g_i(x_i) = G_i x_i - b_i \end{aligned}$$

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 $\mathbb{R}^{mp}$:

$$\mathcal{A} = \left\{ y = \begin{pmatrix} y_1 \\ \vdots \\ y_p \end{pmatrix} \in (\mathbb{R}^m)^p / \sum_{i=1}^{p} y_i = 0 \right\} \tag{1.1}$$

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:

$$\begin{cases} \text{Minimize} & \sum_{i=1}^{p} f_i(x_i) \\ \forall i = 1, \ldots, p & y = g(x) \\ & y \in \mathcal{A} \end{cases} \tag{P2}$$

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

$$L_\lambda(x, y; u) = f(x) - \langle u, g(x) - y \rangle + \frac{\lambda}{2} \|g(x) - y\|^2 \tag{1.2}$$

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

$$L_\lambda(x, y; u) = \sum_{i=1}^{p} L_{\lambda,i}(x_i, y_i; u_i) \tag{1.3}$$

with:

$$L_{\lambda,i}(x_i, y_i; u_i) = f_i(x_i) - \langle u_i, g_i(x_i) - y_i \rangle + \frac{\lambda}{2} \|g_i(x_i) - y_i\|^2 \tag{1.4}$$

We suppose that L_λ has a saddle point $(\bar{x}, \bar{y}; \bar{u})$. 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 minimization of $(x, y) \mapsto L_\lambda(x, y, u^k)$ over $\mathbb{R}^n \times \mathcal{A}$ 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:

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

which can be solved in parallel. Suppose now that u^k is in $\mathcal{A}^\perp$ for all k (this fact 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 finding the closest vector y^{k+1} in $\mathcal{A}$ to $g(x^{k+1})$ *i-e* the projection :

$$y^{k+1} = \Pi_{\mathcal{A}} g(x^{k+1}) \tag{1.5}$$

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

$$u^{k+1} = u^k + \lambda \Pi_{\mathcal{A}^\perp} g(x^{k+1}) \tag{1.6}$$

where the step length λ is the parameter we used in definition (1.2). The projection of $g(x^{k+1})$ on $\mathcal{A}$ is equivalent to compute the amount of the coupling constraint violation:

$$r^{k+1} = \sum_{i=1}^{p} g_i(x_i^{k+1}) \tag{1.7}$$

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

$$y_i^{k+1} = g_i(x_i^{k+1}) - \frac{1}{p} r^{k+1} \tag{1.8}$$

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 $y^0 = u^0 = 0$ for instance), then the updating formulas (1.6), (1.7), (1.8) 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:

$$\mathcal{A}^\perp = \left\{ u = \begin{pmatrix} u_1 \\ \vdots \\ u_p \end{pmatrix} \in (\mathbb{R}^m)^p / u_1 = \ldots = u_p \in \mathbb{R}^m \right\} \tag{1.9}$$

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 (1.9) so the update step becomes:

$$v^{k+1} = v^k - \frac{\lambda}{p} r^{k+1} \tag{1.10}$$

Algorithm 1 SALA : a Separable Augmented Lagrangian Algorithm

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

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. Global scaling of SALA. 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, driving 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.

A multi-dimensional scaling version of SALA is presented in [DGM03]. It consists in setting $\lambda = 1$ and to multiply the coupling constraints by an invertible scaling

matrix $\overline{M}$. We thus deal with the equivalent problem:

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

The use of a matrix $\overline{M}$ provides more freedom for primal-dual balancing. However, this scaling must be the same in every subproblem.

We present now an extension of this technique which allows the matrix to be different from one subproblem to an other. This extension will be called 'Global Scaling'.

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:

$$M = \begin{pmatrix} M_1 & & \\ & \ddots & \\ & & M_p \end{pmatrix} \qquad (2.1)$$

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

$$\mathcal{A}_M = \left\{ z = \begin{pmatrix} z_1 \\ \vdots \\ z_p \end{pmatrix} \in (\mathbb{R}^m)^p / \sum_{i=1}^{p} M_i^{-1} z_i = 0 \right\} \qquad (2.2)$$

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

$$\left\{\begin{array}{ll} \text{Minimize} & \displaystyle\sum_{i=1}^{p} f_i(x_i) \\ \forall i = 1,\ldots,p & z_i = M_i g_i(x_i) \\ & z \in \mathcal{A}_M \end{array}\right. \qquad (P4)$$

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

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

which also decomposes into a sum:

$$L_M(x, z; w) = \sum_{i=1}^{p} L_{i,M_i}(x_i, z_i; w_i) \qquad (2.3)$$

with:

$$L_{i,M_i}(x_i, z_i; w_i) = f_i(x_i) - \langle w_i, M_i g_i(x_i) - z_i \rangle + \frac{1}{2} \|M_i g_i(x_i) - z_i\|^2 \qquad (2.4)$$

Applying the same Gauss-Seidel technique as in SALA, we obtain algorithm 2. The minimization in x still consists in solving independent subproblems:

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

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

$$z^{k+1} = \Pi_{\mathcal{A}_M} Mg(x^{k+1}) \tag{2.5}$$

and the multiplier update will be:

$$w^{k+1} = w^k + \Pi_{\mathcal{A}_M^\perp} Mg(x^{k+1}) \tag{2.6}$$

To project onto $\mathcal{A}_M$ we use the fact that $\mathcal{A}_M = ker(N)$ 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:

$$\Pi_{\mathcal{A}_M} = I - N^\top (NN^\top)^{-1} N \tag{2.7}$$

$$\Pi_{\mathcal{A}_M^\perp} = N^\top (NN^\top)^{-1} N \tag{2.8}$$

Algorithm 2 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 orthogonal subspace where the optimal multipliers live is now

$$\mathcal{A}_M^\perp = \left\{ w = \begin{pmatrix} w_1 \\ \vdots \\ w_p \end{pmatrix} \in (\mathbb{R}^m)^p / M_1^\top w_1 = \ldots = M_p^\top w_p \right\} \tag{2.9}$$

Compared to SALA, 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. Following the dual way, 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 denote $\|y_i\|^2_{\Lambda_i} = \langle y_i, \Lambda_i y_i \rangle$. We then obtain algorithm 3. Note that since M_i is invertible, Λ_i must now be chosen symmetric positive definite.

Algorithm 3 SALAGS : SALA with Global Scaling (reformulation)

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

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

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

The convergence properties of algorithm 3 are going to be examined in the following section through the study of a more general algorithm. We will then be able to prove convergence of SALAGS to a solution of $(P2)$ at the end of the next section with the help of monotone operator theory.

3. A more general setting : the proximal decomposition method. We will see in section 3.2 that problem $(P2)$ can be embedded in a larger 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 will remind algorithm (PDM) in section 3.3 and present its scaled version in section 3.4 before establishing a bound on the asymptotic convergence rate in the differentiable case in section 3.6.

3.1. Finding an orthogonal primal-dual couple in the graph of a maximal monotone operator. A multivalued operator $T : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ is monotone if:

$$\forall y, y' \in \mathbb{R}^n, \forall u \in T(y), \forall u' \in T(y') \quad \langle y - y', u - u' \rangle \geq 0$$

and it is maximal if its graph is not properly contained in the graph of another monotone operator. Let $\mathcal{A}$ be a subspace of $\mathbb{R}^n$. We consider the problem of finding a couple (y, u) such that

$$y \in \mathcal{A} \tag{3.1a}$$
$$u \in \mathcal{A}^\perp \tag{3.1b}$$
$$u \in T(y) \tag{3.1c}$$

These conditions arise for instance as optimality conditions for convex minimization problems of the form $\min_{y\in\mathcal{A}} f(y)$ where $T = \partial f$ is the subdifferential of the closed proper convex function f.

We associate with T the following couple of resolvent operators:

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

Both P and Q are monotone and firmly non-expansive *i-e* for all $s, s' \in \mathbb{R}^n$ it holds:

$$\langle Ps - Ps', s - s'\rangle \geq \|Ps - Ps'\|^2$$
$$\langle Qs - Qs', s - s'\rangle \geq \|Qs - Qs'\|^2$$

P and Q are single-valued. Moreover, the application:

$$\mathbb{R}^n \to \mathbb{R}^n \times \mathbb{R}^n$$
$$s \mapsto (P(s), Q(s))$$

is one-to-one from $\mathbb{R}^n$ to $gr(T)$ (the graph of 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 constant 1) (see [Bre73] for more results on maximal monotone operators).

3.2. Writing problem $(P2)$ as a monotone inclusion problem. To transform problem $(P2)$ under the form of inclusion problem (3.1), we can define the convex implicit function F which gives the optimal cost for a given resource allocation y:

$$F(y) = \inf_x f(x) \text{ s.t } g(x) = y \tag{3.2}$$

The original problem $(P1)$ is now equivalent to minimize F over the coupling subspace $\mathcal{A}$ defined by (1.1) *i-e*:

$$\text{Minimize} \quad F(y) \tag{3.3a}$$
$$\text{s.t} \quad y \in \mathcal{A} \tag{3.3b}$$

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

$$u \in \partial F(y) \tag{3.4a}$$
$$(y, u) \in \mathcal{A} \times \mathcal{A}^\perp \tag{3.4b}$$

Now, if F is closed and proper, then (3.4) fits with formulation (3.1).

3.3. Classical algorithm. The proximal decomposition method aims at solving (3.1) by applying algorithm 4:

Algorithm 4 PDM : Proximal Decomposition Method

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

1: $s^k := y^k + u^k$
2: Compute $(\tilde{y}^{k+1}, \tilde{u}^{k+1}) := (P(s^k), Q(s^k))$
3: Compute $(y^{k+1}, u^{k+1}) := (\Pi_{\mathcal{A}}\tilde{y}^{k+1}, \Pi_{\mathcal{A}^\perp}\tilde{u}^{k+1})$.
4: $k := k + 1$
5: Go to 1

Let denote J the operator:

$$J = \Pi_{\mathcal{A}} P + \Pi_{\mathcal{A}^\perp} Q$$

which maps s^k to s^{k+1}. J satisfies the following property:

PROPOSITION 3.1 ([MOD95]). *A couple (y, u) solves (3.1) iff there is $s \in \mathbb{R}^n$ such that:*

$$s = Js \tag{3.5a}$$
$$y = Ps \tag{3.5b}$$
$$u = Qs \tag{3.5c}$$

Moreover, J is firmly non-expansive of full domain ([MOD95]). Consequently the sequence (s^k, y^k, u^k) converges to a solution of (3.5) for any starting value s^0.

If we apply this algorithm to $\lambda^{-\frac{1}{2}} T \lambda^{-\frac{1}{2}}$ (which is clearly maximal monotone) and $\mathcal{A}$ as defined in subsection 3.2, we then obtain algorithm SALA (algorithm 1).

REMARK 3.1. *We also could have defined:*

$$\tilde{F}(x, y) = \begin{cases} f(x), & \text{if } g(x) = y \\ +\infty, & \text{else.} \end{cases} \tag{3.6}$$

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 could thus obtain an additional proximal term in variable x in step 3 of algorithm 1 like in [Eck94].

3.4. Proximal decomposition with global scaling. We now present a scaled version of PDM from which we will derive SALAGS. We prove that when T is differentiable, we are always (at less theoretically) able to obtain super-linear convergence.

We use an idea already met in [Eck94] which consists in operating a linear change of variable. Let come back to our problem which consists in finding a couple (y, u) such that:

$$u \in T(y) \tag{3.7a}$$
$$(y, u) \in \mathcal{A} \times \mathcal{A}^\perp \tag{3.7b}$$

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 (3.7) is equivalent to find $M^\top w \in T(M^{-1}z)$ with $(z, w) \in \mathcal{A}_M \times \mathcal{A}_M^\perp$ or equivalently:

$$w \in T_M(z) \tag{3.8a}$$
$$(z, w) \in \mathcal{A}_M \times \mathcal{A}_M^\perp \tag{3.8b}$$

with

$$T_M = M^{-\top} T(M^{-1}.) \tag{3.9}$$

But T_M is itself maximal monotone, hence we get a scaled version of original problem (3.7) with exactly the same structure. We can therefore apply PDM to solve it.

Let denote $P_M = (I + T_M)^{-1}$ and $Q_M = (I + T_M^{-1})^{-1}$ as well as:

$$J_M = \Pi_{\mathcal{A}_M} P_M + \Pi_{\mathcal{A}_M^\perp} Q_M \tag{3.10}$$

which are the equivalent of P, Q and J.

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: $s^k := z^k + w^k$
3: Compute $(\tilde{z}^{k+1}, \tilde{w}^{k+1}) := (P_M(s^k), Q_M(s^k))$
4: Compute $(z^{k+1}, w^{k+1}) := (\Pi_{\mathcal{A}_M} \tilde{z}^{k+1}, \Pi_{\mathcal{A}_M^\perp} \tilde{w}^{k+1})$.
5: $(y^{k+1}, u^{k+1}) := (M^{-1} z^k, M^\top w^k)$.
6: $k := k + 1$
7: Go to 1

Applying proposition 3.1, we get the:

PROPOSITION 3.2. *A couple* (y, u) *solves* (3.1) *iff there is* $s \in \mathbb{R}^n$ *such that:*

$$s = J_M s \tag{3.11a}$$
$$y = M^{-1} P_M s \tag{3.11b}$$
$$u = M^\top Q_M s \tag{3.11c}$$

and as J_M is firmly non-expansive of full domain, the sequence (y^k, u^k) generated by algorithm 5 thus converges to a solution of (3.11).

In term of original variables, step 3 is equivalent to find $(\tilde{y}^{k+1}, \tilde{u}^{k+1})$ such that:

$$\tilde{u}^{k+1} \in T(\tilde{y}^{k+1}) \tag{3.12a}$$
$$M\tilde{y}^{k+1} + M^{-\top} \tilde{u}^{k+1} = My^k + M^{-\top} u^k \tag{3.12b}$$

Combining (3.12a) and (3.12b) gives $\tilde{y}^{k+1}$ as the unique solution of:

$$0 \in T(\tilde{y}^{k+1}) + M^\top M(\tilde{y}^{k+1} - y^k) - u^k$$

Steps 4 and 5 then writes:

$$\begin{pmatrix} y^{k+1} \\ u^{k+1} \end{pmatrix} = \begin{pmatrix} M^{-1} P_{\mathcal{A}_M} M & \\ & M^\top P_{\mathcal{A}_M^\perp} M^{-\top} \end{pmatrix} \begin{pmatrix} \tilde{y}^{k+1} \\ \tilde{u}^{k+1} \end{pmatrix} \tag{3.13}$$

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

3.5. Global convergence of SALAGS. The following proposition states the convergence of SALAGS.

PROPOSITION 3.3. *If* F *is proper l.s.c, then the sequence* $\left\{(y^k, u^k)\right\}_k$ *generated by algorithm 3 converges to some* $(\overline{y}, \overline{u})$ *satisfying* (3.8).

Proof. As noticed earlier, algorithm 3 is GSPDM applied to $T = \partial F$ with definition (1.1) 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 (3.8) and consequently the convergence of $\left\{(y^k, u^k)\right\}_k$ generated by GSPDM to solve (3.7). □

3.6. Asymptotic behavior in the differentiable case. Let algorithm 6 converge to (s^*, z^*, w^*). We suppose that T_M is differentiable at z^* *i-e* $T_M(z^*)$ is a singleton (namely $\{w^*\}$) and there is a linear continuous transformation $\nabla T_M(z^*)$ such that:

$$T_M(z) \subset T_M(z^*) + \nabla T_M(z^*)(z - z^*) + o(\|z - z^*\|)B$$

where B is the unit ball of $\mathbb{R}^n$. T_M is differentiable at z^* if and only if T is differentiable at $y^* = M^{-1}z^*$ and:

$$\nabla T_M(\overline{z}) = M^{-\top}\nabla T(\overline{y})M^{-1} \tag{3.14}$$

If $T = \partial F$, this corresponds to twice differentiability and finite-valuedness of F in a neighborhood of y^*.

Differentiability of T_M at z^* implies differentiability of P_M and Q_M at $s^* = z^* + w^*$ and we have the following relations:

$$\nabla P_M(s^*) = (I + \nabla T_M(z^*))^{-1} \tag{3.15a}$$
$$\nabla Q_M(s^*) = I - (I + \nabla T_M(z^*))^{-1} \tag{3.15b}$$
$$\nabla (P_M - Q_M)(s^*) = 2(I + \nabla T_M(z^*))^{-1} - I \tag{3.15c}$$

Using the fact that $2P_M - I = I - 2Q_M = P_M - Q_M$ we rewrite:

$$\begin{aligned} J_M = \Pi_{\mathcal{A}_M} P_M + \Pi_{\mathcal{A}_M^\perp} Q_M &= \frac{1}{2}I + \frac{1}{2}(\Pi_{\mathcal{A}_M} - \Pi_{\mathcal{A}_M^\perp})(P_M - Q_M) \\ &= \frac{1}{2}I + \frac{1}{2}R_{\mathcal{A}_M}(P_M - Q_M) \end{aligned}$$

where $R_{\mathcal{A}_M} = \Pi_{\mathcal{A}_M} - \Pi_{\mathcal{A}_M^\perp}$ is the reflection across the space $\mathcal{A}_M$. So:

$$s^{k+1} = \frac{1}{2}s^k + \frac{1}{2}R_{\mathcal{A}_M}(P_M - Q_M)s^k \tag{3.16}$$

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

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

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

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

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

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

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

$$\limsup_{k\to\infty} \frac{\left\|e^{k+1}\right\|}{\|e^k\|} \leq \frac{1}{2}\left\|(I + R_{\mathcal{A}_M}\nabla(P_M - Q_M)(s^*))\right\| \tag{3.17}$$

By formula (3.15c), we notice that if $\nabla T_M(z^*) = I$ (if $T = \partial F$, it means that $F(M^{-1}\cdot)$ locally behaves like $\frac{1}{2}\|.\|^2$ around z^*) then the convergence rate is lower than $\frac{1}{2}$. Consequently, by formula (3.14), the knowledge a priori 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 $\frac{1}{2}$. A more detailed study of this fact can be found in [LM07] in the quadratic case.

The value $\frac{1}{2}$ actually corresponds to the relaxation factor used in recursion (3.16). 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:

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

Theses observations naturally led us to imagine an auto-adaptive scaling method in which the parameter would ideally tend to $M = (\nabla T(y^*))^{\frac{1}{2}}$.

We will proceed in two steps. We first prove in the next section that convergence is preserved when the scaling parameter is allowed to vary at each iteration and converges quickly enough. Then we propose some heuristics to modify M which are based on observations made here, in the differentiable case, but also applies when T is not differentiable.

4. 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 4.1. *Setting $\Lambda_k = M_k^\top M_k$, the sequence of scaling matrices $\{\Lambda_k\}_k$ satisfies:*

i) Λ_k *is positive definite for all* k
ii) $\Lambda_k \to \Lambda_\infty$ *positive definite*
iii) $\sum_{k=0}^{+\infty} \|\Lambda_{k+1} - \Lambda_k\| < +\infty$

4.1. 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 (3.1) 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:

$$\left\|\bar{s}^k - \sigma^k\right\|^2 \leq \left\|s^k - \sigma^k\right\|^2 - \left\|\bar{s}^k - s^k\right\|^2 \tag{4.1}$$

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

PROPOSITION 4.1. *Under assumption 4.1, there is a sequence $\{\mu_k\}_k$ such that for all k:*

$$(1-\mu_k)\left\|s^{k+1} - \sigma^{k+1}\right\|^2 \leq \left\|\bar{s}^k - \sigma^k\right\|^2 \tag{4.2}$$

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 4.1. *If assumption 4.1 holds, then the sequence $\left\{s^k\right\}_k$ generated by algorithm 6 is bounded.*

Proof. Combining (4.1) and (4.2), we obtain for all k:

$$(1-\mu_k)\left\|s^{k+1}-\sigma^{k+1}\right\|^2 \leq \left\|s^k-\sigma^k\right\|^2 \tag{4.3}$$

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 (4.3) 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 4.2. *If assumption 4.1 holds, the sequence $\left\{s^k\right\}_k$ generated by algorithm 6 converges to a fixed point of J_M.*

Proof. Lemma 4.1 provides the existence of cluster points for the sequence $\left\{s^k\right\}_k$. Combining (4.1) and (4.2) 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 (4.3) 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 $s^k \to s^\infty$. □

4.2. 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 section 3.6 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 $\nabla T(\tilde{y}^k)$.

$$\Lambda_{k+1} = (1-\alpha_k)\Lambda_k + \alpha_k D_k \tag{4.4}$$

In order that the sequence $\left\{\Lambda^k\right\}_k$ satisfies assumption 4.1, we will impose the series of term α_k and D_k to satisfy hypothesis of the following lemma:

LEMMA 4.2 (Convergence of the auto-adaptive scaling sequences). *Let $\{\Lambda_k\}_k$ a sequence of matrices satisfying recursion (4.4). 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 (4.4) 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 4.1. *We could also have defined the recursion:*

$$\Lambda_{k+1} = \Lambda_k^{(1-\alpha_k)} D_k^{\alpha_k} \tag{4.5}$$

in place of (4.4)*. 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.*

4.2.1. 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:

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

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:

$$D_k = (\Pi_{[\underline{\gamma}, \bar{\gamma}]} \gamma^k) I \tag{4.6}$$

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

4.2.2. 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:

$$\gamma_i^{k+1} = \frac{\left\|\tilde{u}_i^{k+1} - \tilde{u}_i^k\right\|}{\left\|\tilde{y}_i^{k+1} - \tilde{y}_i^k\right\|} \tag{4.7}$$

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 4.2. *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 (4.6) *and* (4.7) *are automatically bounded and we can relinquish projection onto $[\underline{\gamma}, \bar{\gamma}]$.*

4.2.3. 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:

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

5. Numerical results.

5.1. Experimentations. We have generated quadratic problems with objective functions $f_i(x_i) = \frac{1}{2}\langle x_i, Q_i x_i\rangle + \langle c_i, x_i\rangle$ for different values of p and m and compared performances of the method with and without updating strategies. We took $p \in [2, 5, 10, 20]$ and $m \in [5, 10, 20]$ and $n_i = m$. Coefficients of b and c were drawn log-uniformly in $[-100, -0.01] \cup [0.01, 100]$. Q_i were constructed by generating matrices P_i and vectors p_i, coefficients of which were drawn log-uniformly respectively in $[-10, -0.1] \cup [0.1, 10]$ and $[0.1, 1]$, and by setting $Q_i = P_i^\top P_i + diag(p_i)$. Coefficients of G_i were drawn uniformly between -10 and 10.

We stopped the algorithm when $\left\|\tilde{y}^{k+1} - y^k\right\|^2 + \left\|\tilde{u}^{k+1} - u^k\right\|^2$ was lower than $p.10^{-5}$ or when the iteration number exceeded 5000.

We used $\Lambda^{k=0} = \lambda^0 I$ as a starting value, with λ^0 ranging in $[10^{-3}, 100]$. The updating strategy implemented is the one described in remark 4.1. We chose $\alpha_k = (k+1)^{-\frac{10}{9}}$ as averaging sequence.

		Updating rule			
p	m	none	**Single**	**Subproblem**	**Component**
2	5	39	41	48	37
	10	78	74	74	89
	20	93	81	81	86
5	5	61	81	53	59
	10	77	83	80	88
	20	104	101	99	110
10	5	85	102	104	127
	10	126	141	135	152
	20	126	161	149	178
20	5	150	178	143	160
	10	187	256	191	200
	20	181	268	236	244

TABLE 5.1
Best number of iterations obtained for $\lambda^0 \in [10^{-3}, 100]$ for different updating strategies.

		Updating rules			
p	m	none	**Single**	**Subproblem**	**Component**
2	5	≥ 415	17	9	21
	10	≥ 2076	56	62	93
	20	≥ 1776	60	56	58
5	5	≥ 2038	38	39	54
	10	≥ 2129	37	31	58
	20	≥ 2158	59	51	55
10	5	≥ 2104	72	39	54
	10	≥ 2112	79	55	112
	20	≥ 2129	180	123	354
20	5	≥ 2108	119	67	430
	10	≥ 2081	251	133	320
	20	≥ 2092	220	131	321

TABLE 5.2
Standard deviation of the number of iterations w.r.t λ^0 for different updating strategies.

5.2. Results. For each updating strategy, we launched the algorithm with several starting values λ^0 ranging in $[10^{-3}, 100]$ and we reported in table 5.1 the minimum number of iterations obtained. We can see that the best case is comparable whatever we use or not an updating strategy. However, the need to choose a good initial value disappear. Table 5.2 shows the standard deviation of the number of iterations with respect to the initial value λ^0. The symbol $'\geq'$ in the first column means that the value is an underestimate because several launch ran over 5000 iterations. Algorithms implemented with an updating rule are much less sensitive to the starting value than the classical method. By way of example, we plotted for each updating rule the number of iterations with respect to $\lambda^{k=0}$ in the case $p = 20$ and $m = 10$ on figure 5.1.

We can mention that updating strategy (4.4) has an asymmetric behavior with respect to the starting parameter. Namely, scaling parameters are much better corrected when the starting value is too low than when it is too high. It seems to be due

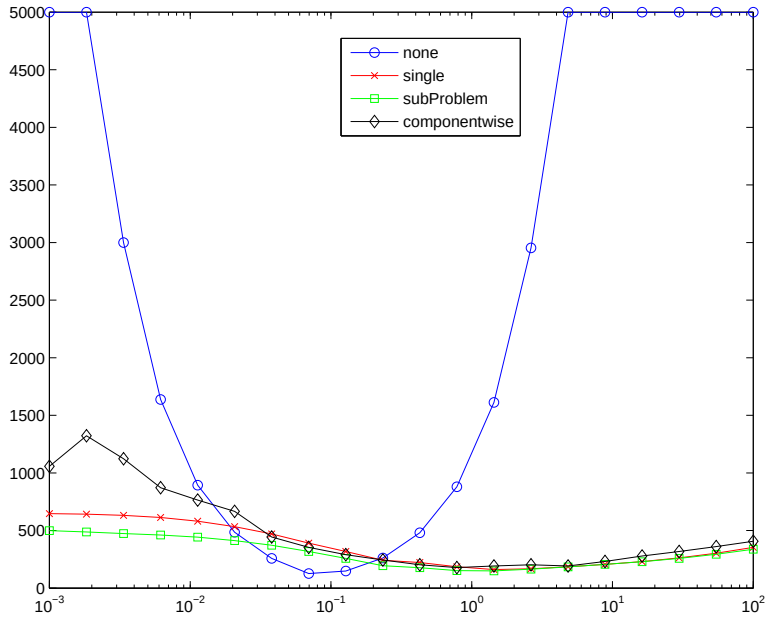

FIG. 5.1. *Number of iterations of SALA for different scaling updating strategies.*

to the difficulty to decrease the order of magnitude with expression (4.4).

6. Conclusion. The main drawback of algorithm SALA lies in the difficult choice of the scaling parameter which is crucial to obtain a good numerical behaviour. We introduced here a generalization of the scaling method already initiated in [DGM03] which allows for several independent parameters in each subproblems. We also proposed auto-adaptive techniques to tune them, underlying idea of which comes from the differentiable case. Supporting numerical results show that while the performance is not really improved in "the best case", the need to choose a good initial parameter values vanishes at a negligible additional computational cost.

In the future, we plan to investigate the impact of auto-adaptive techniques when applied to not such ideal problems, as for instance linear programs, both from a theoretical and numerical point of view.

REFERENCES

[Bre73] H. Brezis, *Opérateurs maximaux montones et semi-groupes de contractions dans les espaces de hilbert*, North Holland, 1973.

[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.

[Eck89] Jonathan Eckstein, *Splitting methods for monotone operators with applications to par-*

allel 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.

[LM07] Arnaud Lenoir and Philippe Mahey, *Accelerating the convergence of a separable augmented lagrangian algorithm*, Research Report LIMOS, Université Blaise Pascal (2007), no. RR-07-14.

[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.

[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).