COMPUTATIONAL EXPERIENCE WITH NUMERICAL METHODS FOR NONNEGATIVE LEAST-SQUARES PROBLEMS *

STEFANIA BELLAVIA †, JACEK GONDZIO ‡, AND BENEDETTA MORINI §

Abstract. We discuss the solution of large-scale box-constrained linear least-squares problems by two recent affine-scaling methods: a cyclic Barzilai-Borwein strategy and an Inexact Newton-like method where a preconditioning technique allows for an efficient computation of the steps. A robust globally and fast locally convergent method based on the combination of the two procedures is presented along with extensive numerical results.

1. Introduction. We address the solution of the box-constrained least-squares problem

$$\min_{l\le x\le u} q(x) = \frac{1}{2}\|Ax-b\|_2^2 + \frac{\mu}{2}\|x\|_2^2, \tag{1.1}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ are given, $m \ge n$, and $\mu \ge 0$. In order to ensure that the problem has a unique solution, we assume that A has full rank whenever $\mu = 0$, [4]. The lower and upper bounds l, $u \in \mathbb{R}^n$ are such that $l < u$ and some components may be equal to minus or plus infinity.

To simplify the presentation, we will mainly focus on the case where one-sided bounds apply and outline the generalization to the case of two-sided (box) constraints. Then problem (1.1) can be stated as

$$\min_{x\ge 0} q(x) = \frac{1}{2}\|Ax-b\|_2^2 + \frac{\mu}{2}\|x\|_2^2. \tag{1.2}$$

A number of different methods for the solution of (1.2) have been proposed during the last years [2, 3, 4, 5, 6, 13, 17, 15, 18]. In this work we follow the so-called Affine-scaling Interior-Point approach based on the papers [6, 7] by Coleman and Li. This approach relies on the observation that the first-order optimality conditions (KKT conditions) for (1.2) can be written as a nonlinear system of equations

$$D(x)g(x) = 0, \tag{1.3}$$

where $x \ge 0$, $g(x) = \nabla q(x) = (A^TA+\mu I)x - A^Tb$, and $D(x) = diag(d_1(x), \ldots, d_n(x))$ has entries of the form

$$d_i(x) = \begin{cases} x_i & \text{if} \quad g_i(x) \ge 0, \\ 1 & \text{otherwise,} \end{cases} \tag{1.4}$$

with x_i and $g_i(x)$ denoting the i-th component of the vectors x and $g(x)$. Since the variable x in (1.3) is subject to the nonnegativity constraint, affine-scaling methods for solving (1.3) are iterative procedures generating strictly feasible iterates $x_k > 0$.

*Work supported by MIUR, Roma, Italia, through PRIN 2007, "Sviluppo ed analisi di modelli matematici e di metodi numerici per equazioni alle derivate parziali per le applicazioni a problemi ambientali ed industriali" and INDAM-GNCS.

†Dipartimento di Energetica "S. Stecco", Università di Firenze, via C. Lombroso 6/17, 50134 Firenze, Italia, stefania.bellavia@unifi.it

‡School of Mathematics, The University of Edinburgh, Mayfield Road, Edinburgh EH9 3JZ, Scotland, UK, J.Gondzio@ed.ac.uk

§Dipartimento di Energetica "S. Stecco", Università di Firenze, via C. Lombroso 6/17, 50134 Firenze, Italia, benedetta.morini@unifi.it

Given $x_k > 0$, by formal application of the product rule, the kth iteration of the Newton method for (1.3), takes the form

$$(D(x_k)\,(A^TA + \mu I) + diag(g(x_k)^+))p = -D(x_k)g(x_k), \tag{1.5}$$

where $(A^TA + \mu I)$ is the Hessian of the function q, and $g_i(x)^+ = \max\{0, g_i(x)\}$, $i = 1, ..., n$. Note that the not everywhere existing derivatives of $\dfrac{\partial d(x)}{\partial x_i}$, $i = 1, ..., n$, are substituted by the real valued functions $\max\{0, sign(g_i(x))\}$. Strict feasibility of the new iterate is enforced by either a projection or a truncation strategy ([6, 17]).

In case problem (1.2) is large, solving the linear system may be computationally costly and time consuming. This occurrence is addressed in some recent papers adopting different approaches. An efficient solution of the linear systems (1.5) is proposed in [2, 3]; the resulting method is a Newton-like procedure which converges locally fast to the solution. An alternative approach, presented by Hager, Maier and Zhang in [15], is the Affine-Scaling Cyclic Barzilai-Borwein (AS_CBB) method where the matrix in (1.5) is replaced by a diagonal matrix and the solution of the linear system is not required.

The above methods were combined with globalization strategies. In particular, in [3] the Newton-like method was embedded into a simple globalization strategy which uses the Newton step and a constrained scaled Cauchy step; here the resulting method will be denoted Globally convergent REgularized Newton-like (GREN) method. On the other hand, the AS_CBB method employs a nonmonotone linesearch strategy and has local R-linear convergence to a nondegenerate solution.

The numerical experiments carried out in [2] showed that GREN method is quite successful in practice. However, in some cases, the repeated use of a step bent towards the scaled Cauchy step yields a lack of robustness. In this work we propose a new procedure that combines the GREN method and the AS_CBB method. The new approach is based on the use of the GREN procedure as long as its iterates differ considerably from the iterates generated by the scaled Cauchy step. Otherwise AS_CBB is applied for some iterations. This choice is motivated by the fact that AS_CBB method is superior to classical steepest descent method. A suitable combination of these strategies enhances robustness of both the GREN and the AS_CBB procedures and favourably compares with them in terms of computational cost. In Sections 2 and 3 we introduce the AS_CBB and the GREN method, respectively. In Section 4 we describe the new hybrid method which combines the above procedures. Section 5 is devoted to the numerical solution of the linear systems arising in the hybrid method. The extension of the preconditioner given in [2] to the case $\mu > 0$ in (1.1) is described along with the study of its spectral properties. Further relevant implementation issues are presented in Sections 6 and 7. The algorithm used for the numerical comparison with the hybrid method is summarized in Section 8. The results of numerical experiments are discussed in Section 9.

1.1. Notations. We use the subscript k as index for any sequence and for any function f we denote $f(x_k)$ by f_k. The symbol x_i or $(x)_i$ denotes the i-th component of a vector x. For any vector t, we let $t^+ = \max\{0, t\}$, where max is meant componentwise. Finally, the symbol I denotes the identity matrix of dimension inferred by the context.

2. The AS_CBB method. The general framework of AS_CBB method was proposed for general box-constrained optimization problems but here we restrict to the case where the objective function is the one of problem (1.2).

The AS_CBB method replaces the Hessian matrix $(A^TA + \mu I)$ in (1.5) by the matrix $\lambda_k I$ where λ_k is a positive scalar. Then, the step b_k used solves the equation

$$(\lambda_k D_k + diag(g_k^+))b_k = -D_k g_k,$$

and the ith component of b_k is given by

$$(b_k)_i = -\left(\frac{1}{\lambda_k + (g_k)_i^+/(x_k)_i}\right)(g_k)_i. \tag{2.1}$$

Remarkably, $x_k + b_k$ is strictly feasible [15, Lemma 3.4].

The positive scalar λ_k is computed using a cyclic version of the Barzilai-Borwein (BB) stepsize rule. In particular, let

$$\lambda_0^{BB} = \max\{\bar{\lambda},\, \|g_0\|_\infty\}, \tag{2.2}$$

$$\lambda_k^{BB} = \underset{\lambda \geq \bar{\lambda}}{\operatorname{argmin}} \|\lambda s_{k-1} - y_{k-1}\|_2 = \max\left\{\bar{\lambda},\, \frac{s_{k-1}^T y_{k-1}}{s_{k-1}^T s_{k-1}},\right\}, \quad k \geq 1, \tag{2.3}$$

where $\bar{\lambda}$ is a fixed positive parameter, $s_{k-1} = x_k - x_{k-1}$, $y_{k-1} = g_k - g_{k-1}$, [1]. These BB stepsizes ensure that the denominator in (2.1) is bounded away from zero. The cyclic BB strategy consists in reusing the BB stepsize for several iterations and performs better than the standard BB strategy [9]. In practice, letting $c \geq 1$ be the cycle length and $l \geq 0$ be the cycle number, the value of the scalars λ_k is assigned by the following rule

$$\lambda_{cl+i} = \lambda_{cl+1}^{BB}, \quad i = 1, \ldots, c. \tag{2.4}$$

The choice of the steplength (2.3) is superior to the choice of the classical Cauchy steepest descent steplength along $-D_k g_k$ both in theory and practice. Since the BB method does not monotonically reduce the value of the objective function, AS_CBB method generates a new iterate of the form

$$x_{k+1} = x_k + \zeta_k b_k, \tag{2.5}$$

where the stepsize $\zeta_k \in (0, 1]$ is computed by a nonmonotone linesearch strategy. This way, the generated sequence is strictly feasible, due to the property that $x_k + b_k$ is strictly feasible.

The algorithm for the kth iteration is given below.

`AS_CBB ALGORITHM.` k`th iteration`
Given $c \geq 1$, $C > 0$, δ, γ, $\bar{\lambda} \in (0, 1)$.
1. Let λ_k be given by (2.2)-(2.4).
2. Compute b_k by (2.1).
3. Set

$$q_k^{max} = \max\{q_{k-i},\ 0 \leq i \leq \min\{k-1, C-1\}\}. \tag{2.6}$$

4. If $k = 0$, set $q_0^r = q_0$
   Else choose q_k^r so that $q_k \leq q_k^r \leq \max\{q_{k-1}^r, q_k^{max}\}$ and $q_k^r \leq q_k^{max}$ infinitely often.
5. Let q_R be either q_k^r or $\min\{q_k^{max}, q_k^r\}$.
6. If $q_k \leq q_R + \delta g_k^T b_k$, set $\zeta_k = 1$;
   Else find the smallest integer j such that

$$q(x_k + \gamma^j b_k) \leq q_R + \delta \gamma^j g_k^T b_k, \tag{2.7}$$

and set $\zeta_k = \gamma^j$.
7. Set $x_{k+1} = x_k + \zeta_k b_k$

The linesearch performed along b_k makes use of the local maximum function q_k^{max} defined in (2.6) where the scalar C is a fixed integer memory. In particular, q_k^{max} is used in Steps 4-5 to choose the scalar q_R employed in (2.7). Various strategies for setting q_R have been proposed and we refer to [16, Appendix A]. The scalars $\delta \in (0,1)$, $\gamma \in (0,1)$ are the Armijo parameters.

The AS_CBB method is globally convergent with R-linear asymptotic rate of convergence to a nondegenerate solution [15].

3. The GREN method. On the base of the analysis conducted in [17], the authors proposed a Globally convergent REgularized Newton-like (GREN) method suited for bound-constrained least-squares problems, [2]. Letting

$$E(x) = diag(e_1(x), \ldots, e_n(x)), \tag{3.1}$$

with

$$e_i(x) = \begin{cases} g_i(x) & \text{if} \quad 0 \le g_i(x) < x_i^2 \text{ or } g_i(x)^2 > x_i, \\ 0 & \text{otherwise,} \end{cases} \tag{3.2}$$

and

$$W(x) = diag(w_1(x), \ldots, w_n(x)), \qquad w_i(x) = \frac{1}{d_i(x) + e_i(x)}\,, \tag{3.3}$$

the Newton equation (1.5) is replaced with the linear system

$$W_k D_k N_k p_k = -W_k D_k g_k, \tag{3.4}$$

where

$$N_k = A^T A + \mu I + D_k^{-1} E_k + \Delta_k, \tag{3.5}$$

$$\Delta_k = diag(\delta_{k,1}, \delta_{k,2}, \ldots, \delta_{k,n}), \qquad \delta_{k,i} \in [0,\, 1),\ i = 1, \ldots, n, \tag{3.6}$$

The linear system (3.4) can be reformulated as the symmetric and positive definite system

$$S_k N_k S_k \tilde{p}_k = -S_k g_k, \tag{3.7}$$

with

$$S(x) = W(x)^{\frac{1}{2}}\, D(x)^{\frac{1}{2}}, \qquad \tilde{p}_k = S_k^{-1} p_k, \tag{3.8}$$

and the corresponding augmented system takes the form

$$\begin{pmatrix} I & AS_k \\ S_k A^T & -C_k \end{pmatrix} \begin{pmatrix} \tilde{q}_k \\ \tilde{p}_k \end{pmatrix} = \begin{pmatrix} -(Ax_k - b) \\ \mu S_k x_k \end{pmatrix}, \tag{3.9}$$

$$C_k = W_k E_k + (\mu I + \Delta_k) S_k^2, \tag{3.10}$$

The use of the matrices E and W allows to develop fast convergent methods without assuming strict complementarity at the solution ([3, 17]) while the regularizing

matrix Δ_k has been introduced in [2] with the aim to design an efficient preconditioner for the augmented system and to avoid the potential ill-conditioning of the augmented system in the case $\mu = 0$.

To enforce strict feasibility of the iterates, the projected Newton step $\hat{p}_k$

$$\hat{p}_k = \max\{\sigma,\, 1 - \|P(x_k + p_k) - x_k\|_2\,\}\,(P(x_k + p_k) - x_k), \tag{3.11}$$

is formed. Here $\sigma \in (0, 1)$ is close to one, and $P(x)$ is the projection of x onto the feasible set, i.e. $P(x) = x^+$.

The GREN method consists of the above described Newton method and of a globalization strategy which provides a sufficient decrease in the value of q with respect to a constrained scaled Cauchy step p_k^C, [3]. Let ψ_k be the following quadratic function

$$\psi_k(p) = \frac{1}{2} p^T N_k p + p^T g_k,$$

whose minimizer is the Newton step p_k. Further, let p_k^C be an approximate solution to the problem

$$\text{argmin}\{\psi_k(p) : p = -c_k D_k g_k, c_k > 0, x_k + p > 0\}.$$

In practice, p_k^C is given by

$$p_k^C = -c_k D_k g_k, \tag{3.12}$$

with

$$c_k = \begin{cases} \dfrac{g_k^T D_k g_k}{g_k^T D_k N_k D_k g_k}, & \text{if} \quad x_k - \dfrac{g_k^T D_k g_k}{g_k^T D_k N_k D_k g_k} D_k g_k > 0 \\ \theta\, \text{argmin}\{l > 0, x_k - l D_k g_k \geq 0\}, & \theta \in (0,1), \text{ otherwise} \end{cases}. \tag{3.13}$$

Then, the new iterate x_{k+1} has the form

$$x_{k+1} = x_k + t p_k^C + (1 - t)\hat{p}_k, \tag{3.14}$$

and it is required to satisfy

$$\frac{\psi_k(x_{k+1} - x_k)}{\psi_k(p_k^C)} \geq \beta, \qquad \beta \in (0,\, 1). \tag{3.15}$$

This ensures global convergence of the procedure. If the point $x_{k+1} = x_k + \hat{p}_k$ satisfies (3.15), t is simply taken equal to zero, otherwise a scalar $t \in (0, 1]$ is computed in order to satisfy

$$\frac{\psi_k(t p_k^C + (1 - t)\hat{p}_k)}{\psi_k(p_k^C)} - \beta = 0. \tag{3.16}$$

It is easy to see that this problem amounts to finding the smallest root of the above scalar quadratic equation.

The convergence analysis of GREN method carried out in [2] is valid for the case $\mu = 0$ and in an inexact framework i.e. an Inexact Newton step can replace the step $p_k = S_k \tilde{p}_k$ provided that $\tilde{p}_k$ satisfies:

$$S_k N_k S_k \tilde{p}_k = -S_k g_k + \tilde{r}_k, \tag{3.17}$$

with

$$\|\tilde{r}_k\|_2 \leq \eta_k \|W_k D_k g_k\|_2, \qquad \eta_k \in [0,1), \tag{3.18}$$

Such convergence analysis can be easily extended to the case $\mu > 0$. Hence the GREN method results to be globally convergent and choosing $\|\Delta_k\|_2 \leq \Lambda_1 \|W_k D_k g_k\|_2$ and $\eta_k \leq \Lambda_2 \|W_k D_k g_k\|_2$ for some positive Λ_1 and Λ_2 and for k sufficiently large, it converges quadratically to the solution of (1.2) even if it is degenerate.

4. The hybrid method. The globalization strategy employed in GREN is cheap but may be ineffective in some occurrences. In particular, if the projected Newton step $\hat{p}_k$ is a poor direction then the step used in (3.14) is likely to be bent towards the Cauchy step p_k^C. The repeated use of such step may produce a very slow progress towards the solution. In fact, the projected Newton step is guaranteed to provide a sufficient reduction of the objective function only when the current iterate is close enough to the solution. On the other hand, the Barzilai-Borwein method AS_CBB is superior to standard steepest descent method both in theory and practice. Therefore, here we propose a hybrid algorithm based on a suitable combination of the GREN and AS_CBB methods. In other words, the GREN method is enriched with a further globalization strategy with the aim to avoid unnecessary computations of the Newton step, whenever the projected Newton step does not provide reduction of the objective function.

The combination of the GREN method and the AS_CBB method is given in the following algorithm and it is based on the preceding discussion. The logical variable `m_GREN` has value true if the method to be applied is the GREN method. At the first iteration `m_GREN` is set to true, and then it is adaptively modified.

`HYBRID ALGORITHM.` k`th iteration`
Given `m_GREN`, β, $t_u \in (0,1)$, $\omega_1 > 0$ (for GREN algorithm)
$c \geq 1$, $C > 0$, δ, γ, $\bar{\lambda} \in (0,1)$, I_{CBB}, I_{CBB}^{max} (for AS_CBB algorithm).

1. If `m_GREN`
   1.1 Choose the matrix Δ_k and the forcing term η_k.
   1.2 Solve (3.17) and (3.18) for $\tilde{p}_k$. Set $p_k = S_k \tilde{p}_k$.
   1.3 Let $\hat{p}_k$ be given by (3.11).
   1.4 If $\psi_k(\hat{p}_k) \geq \beta \psi_k(p_k^C)$, then set $x_{k+1} = x_k + \hat{p}_k$.
   Else compute the smallest root t of (3.16).
   If $t \leq t_u$ then set $x_{k+1} = x_k + t p_k^C + (1-t)\hat{p}_k$.
   Else compute x_{k+1} by the AS_CBB algorithm.
   1.5 If $\dfrac{\psi_k(\hat{p}_k)}{\psi_k(p_k^C)} < -1$ and $\min\limits_{1 \leq i \leq n} |(x_k)_i| < \omega_1$ then
   set `m_GREN=false`, $I_{CBB} = 1$.
   Compute x_{k+1} by the AS_CBB algorithm.
   Else
   1.6 Compute x_{k+1} by the AS_CBB algorithm,
   set $I_{CBB} = I_{CBB} + 1$.
   If $I_{CBB} > I_{CBB}^{max}$ set `m_GREN=true`.

We remark that the AS_CBB strategy is activated either in Step 1.4 or in Step 1.5. In the first case, the parameter t_u is assumed to be near to one and the AS_CBB strategy is performed if the step $(t p_k^C + (1-t)\hat{p}_k)$ is bent towards p_k^C. In the second case, the switch to the AS_CBB strategy is performed if x_k is near to the boundary of

the feasible set and the projected Newton step $\hat{p}_k$ does not provide a decrease of the quadratic model ψ_k. Since one iteration of the AS_CBB strategy is cheap, if in Step 1.5 we activate this method then I_{CBB}^{max} successive AS_CBB iterations are performed.

We underline that the hybrid method inherits all the asymptotic convergence properties of the GREN method. This is due to the fact that in [2, Theorem 3.1] it is proved that eventually $\hat{p}_k$ satisfies $\psi_k(\hat{p}_k) \geq \beta\psi_k(p_k^C)$. Then eventually, only the GREN method is applied.

We conclude this section presenting the generalization of the hybrid method to the case where the unknown is bounded on both sides. In this case everything follows as in the nonnegative case, once suitable adaptations of the matrices $D(x)$ and $E(x)$ and the Cauchy step are introduced. The KKT conditions are written as in (1.3) letting $l \leq x \leq u$ and $D(x) = diag(d_1(x), \ldots, d_n(x))$ with

$$d_i(x) = \begin{cases} u_i - x_i & \text{if } \; g_i(x) < 0, u_i < \infty \\ x_i - l_i & \text{if } \; g_i(x) \geq 0, l_i > -\infty \\ 1 & \text{otherwise,} \end{cases} \tag{4.1}$$

Let $J_k^d \in \mathbb{R}^{n\times n}$ be the diagonal matrix whose i-th row is given either by the gradient of $d_i(x)$ (whenever $d_i(x)$ is differentiable) or by the null vector (when $g_i(x) = 0$). Then, the k-th iteration of the Newton method for (1.3) takes the form

$$(D(x_k)\,(A^TA + \mu I) + diag(g(x_k))J_k^d)p = -D(x_k)g(x_k). \tag{4.2}$$

The diagonal entries e_i of the matrix E used in (3.5) are:

$$e_i(x) = \begin{cases} |g_i(x)| & \text{if } \; |g_i(x)| < \min\{x_i - l_i, u_i - x_i\}^2 \text{ or} \\ & \quad |g_i(x)|^2 > \min\{x_i - l_i, u_i - x_i\}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

Then, the projected Newton step has the form (3.11) where $P(x) = \max\{l, \min\{x, u\}\}$. The Cauchy step p_k^C is given by (3.12) with

$$c_k = \begin{cases} \dfrac{g_k^T D_k g_k}{g_k^T D_k N_k D_k g_k}, & \text{if } \; l < x_k - \dfrac{g_k^T D_k g_k}{g_k^T D_k N_k D_k g_k} D_k g_k < u \\ \theta\,\text{argmin}\{\tau > 0, l \leq x_k - \tau D_k g_k \leq u\}, \quad \theta \in (0,1), & \text{otherwise.} \end{cases}$$

Finally, we refer to [15, §8] for the generalization of AS_CBB to handle box constrained problems.

5. Solution of the Newton equation. In this section we address the computation of the Newton step, when the dimension of the problem is large. For sake of simplicity, we focus on problem (1.2). First, we assume to use an iterative linear solver and to compute an inexact Newton step $\tilde{p}_k$ satisfying (3.17) and (3.18). In the following, we let $p_k = S_k\tilde{p}_k$.

In order to choose an iterative method and a suitable preconditioner, we follow the ideas of [2] and extend the approach proposed in [2] to the case $\mu > 0$. We take advantage of the splitting of the indices $\{1, \ldots, n\}$ into two sets. From (3.1), (3.3) and (3.8) it is easy to note that $(w_k)_i(e_k)_i + (s_k)_i^2 = 1$, for $i = 1, \ldots, n$. Moreover $(s_k)_i$ either tends to zero or one whenever $\{x_k\}$ approaches the solution x^*. In fact, if $(x^*)_i$ is active and non-degenerate, we have that $(x_k)_i \to 0$ and from (3.2) it follows $(e_k)_i \to g_i(x^*) \neq 0$. This implies that $(s_k)_i \to 0$. On the other hand, if $(x^*)_i$ is

inactive, we must have $g_i(x^*) = 0$ and from the definition of the matrix E it follows $(e_k)_i \to g_i(x^*) = 0$ and this yields $(s_k)_i \to 1$. In the case $(x_k)_i$ is degenerate, it is easy to see, again using (3.2) that if $g_i(x_k) = o(\sqrt{(x_k)_i})$ then $(s_k)_i \to 1$, otherwise $(s_k)_i \to 0$.

Therefore, given a small positive threshold $\tau \in (0,1)$, at each iteration we let

$$\mathcal{L}_k = \{i \in \{1,2,\ldots,n\},\ s.t.\ \ (s_k)_i^2 \geq 1-\tau\}, \quad n_1 = card(\mathcal{L}_k), \tag{5.1}$$

where $card(\mathcal{L}_k)$ is the cardinality of the set $\mathcal{L}_k$.

If the set $\mathcal{L}_k$ is empty then in (3.9) we have $\|S_k\|_2 \leq 1-\tau$. Moreover, $\|S_k\|_2$ is expected to be small if x_k is close to x^*. In such a case, to use a short-recurrence method, we apply the Conjugate-Gradient (CG) method to (3.4) without preconditioner and solve the linear system so that

$$S_k N_k S_k \tilde{p}_k = -S_k g_k + \tilde{r}_k, \tag{5.2}$$

where the residual vector $\tilde{r}_k$ is required to satisfy (3.18).

If the set $\mathcal{L}_k$ is nonempty, we proceed as follows. Let us omit permutations and assume that

$$S_k = \begin{pmatrix} (S_k)_1 & 0 \\ 0 & (S_k)_2 \end{pmatrix}, \tag{5.3}$$

$$(S_k)_1 = diag_{i\in\mathcal{L}_k}\left((s_k)_i\right) \in \mathbb{R}^{n_1\times n_1},$$

$$(S_k)_2 = diag_{i\notin\mathcal{L}_k}\left((s_k)_i\right) \in \mathbb{R}^{(n-n_1)\times(n-n_1)}. \tag{5.4}$$

Analogously for any diagonal matrix $G \in \mathbb{R}^{n\times n}$ we let $(G)_1 \in \mathbb{R}^{n_1\times n_1}$ be the submatrix formed by the first n_1 rows and n_1 columns and $(G)_2 \in \mathbb{R}^{(n-n_1)\times(n-n_1)}$ be the submatrix formed by the remaining rows and columns. Finally, we consider the partitioning $A = (A_1, A_2)$, $A_1 \in \mathbb{R}^{m\times n_1}$, $A_2 \in \mathbb{R}^{m\times(n-n_1)}$ and $x_k = ((x_k)_1^T, (x_k)_2^T)$, $(x_k)_1 \in \mathbb{R}^{n_1}$, $(x_k)_2 \in \mathbb{R}^{n-n_1}$. Then, the augmented system (3.9) takes the form

$$\begin{pmatrix} I & A_1(S_k)_1 & A_2(S_k)_2 \\ (S_k)_1 A_1^T & -(C_k)_1 & 0 \\ (S_k)_2 A_2^T & 0 & -(C_k)_2 \end{pmatrix} \begin{pmatrix} \tilde{q}_k \\ (\tilde{p}_k)_1 \\ (\tilde{p}_k)_2 \end{pmatrix} = \begin{pmatrix} -(Ax_k - b) \\ \mu(S_k)_1(x_k)_1 \\ \mu(S_k)_2(x_k)_2 \end{pmatrix}, \tag{5.5}$$

and eliminating $(\tilde{p}_k)_2$ from the first equation we get

$$\underbrace{\begin{pmatrix} I + Q_k & A_1(S_k)_1 \\ (S_k)_1 A_1^T & -(C_k)_1 \end{pmatrix}}_{\mathcal{A}_k} \begin{pmatrix} \tilde{q}_k \\ (\tilde{p}_k)_1 \end{pmatrix} = \begin{pmatrix} -(Ax_k - b) + R_k \\ \mu(S_k)_1(x_k)_1 \end{pmatrix}, \tag{5.6}$$

where $Q_k = A_2(S_k C_k^{-1} S_k)_2 A_2^T \in \mathbb{R}^{m\times m}$ and $R_k = \mu A_2(S_k C_k^{-1} S_k)_2 (x_k)_2 \in \mathbb{R}^m$. The preconditioner we use for (5.6) is the matrix

$$\mathcal{P}_k = \begin{pmatrix} I & A_1(S_k)_1 \\ (S_k)_1 A_1^T & -((\mu I + \Delta_k)S_k^2)_1 \end{pmatrix}. \tag{5.7}$$

Here we point out that, by (5.1) we have $\|(W_k E_k)_1\|_2 \leq \tau$, $\|(C_k)_2^{-1}\|_2 \leq 1/\tau$, $\|Q_k\|_2 \leq (1-\tau)\|A_2\|_2^2/\tau$. Further, we note that when $\{x_k\}$ approaches the solution x^* we have $(S_k)_1 \to I$, $(S_k)_2 \to 0$, $\|Q_k\|_2 \to 0$ and $\|(W_k E_k)_1\|_2 \to 0$. Therefore, $\|\mathcal{P}_k - \mathcal{A}_k\|_2$ tends to zero.

In the following theorems we characterize the spectral properties of the matrix $\mathcal{P}_k^{-1}\mathcal{A}_k$ for varying iterative linear solvers. Iterative methods for indefinite systems such as BiCGSTAB, GMRES, QMR can be applied to solve the preconditioned augmented system. Alternatively, in case A is full rank, the specific structure of (5.6) and (5.7) allows the use of the short-recurrence Projected Preconditioned Conjugate-Gradient (PPCG) method developed in [10, 12]. While PPCG would have the disadvantage that A must be full rank, it has the relevant features to satisfy a minimization property and to require a fixed amount of work per iteration.

First, we analyze the spectral properties of the preconditioned system in case iterative methods for indefinite systems are applied. This analysis was first provided in [2] for the case $\mu = 0$. However, the proof given in [2, Theorem 4.1] had a gap and characterized only the *real* eigenvalues of $\mathcal{P}_k^{-1}\mathcal{A}_k$. In fact the eigenvalues of $\mathcal{P}_k^{-1}\mathcal{A}_k$ may be *complex*. Below we generalize this theorem to the case $\mu \neq 0$ and provide a new proof which corrects the gap of [2].

THEOREM 5.1. *Let $\mathcal{A}_k$ and $\mathcal{P}_k$ be the matrices given in (5.6) and (5.7). Then at least $m - n + n_1$ eigenvalues of $\mathcal{P}_k^{-1}\mathcal{A}_k$ are unit and the other eigenvalues have positive real part.*

Letting $(u^T, v^T)^T$, $u \in \mathbb{C}^m$, $v \in \mathbb{C}^{n_1}$, be an eigenvector of $\mathcal{P}_k^{-1}\mathcal{A}_k$ associated to $\lambda = Re(\lambda) + iIm(\lambda)$, and Z be the imaginary part of $u^H A_1 (S_k)_1 v$, then λ has the form

$$\lambda = 1 + \gamma,$$

with

$$Re(\gamma) = \frac{(u^H Q_k u + v^H (W_k E_k)_1\, v)(u^H u + v^H ((\mu I + \Delta_k) S_k^2)_1\, v)}{(u^H u + v^H ((\mu I + \Delta_k S_k^2)_1\, v)^2 + 4Z^2}, \tag{5.8}$$

and

$$Im(\gamma) = -\frac{2Z(u^H Q_k u + v^H (W_k E_k)_1\, v)}{(u^H u + v^H ((\mu I + \Delta_k) S_k^2)_1\, v)^2 + 4Z^2}, \tag{5.9}$$

Proof. The eigenvalues and eigenvectors of matrix $\mathcal{P}_k^{-1}\mathcal{A}_k$ satisfy

$$\begin{pmatrix} I + Q_k & A_1 (S_k)_1 \\ (S_k)_1 A_1^T & -(C_k)_1 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \lambda \begin{pmatrix} I & A_1 (S_k)_1 \\ (S_k)_1 A_1^T & -((\mu I + \Delta_k) S_k^2)_1 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}.$$

If $\lambda = 1$ we get

$$\begin{aligned} (I + Q_k)u &= u \\ (C_k)_1 v &= ((\mu I + \Delta_k) S_k^2)_1 v \end{aligned}$$

i.e. u belongs to the null space of Q_k and v belongs to the null space of $(W_k E_k)_1$. As the rank of Q_k is at most $n - n_1$, it follows that there are at least $m - (n - n_1)$ unit eigenvalues. If $\lambda \neq 1$, denoting $\lambda = 1 + \gamma$ we have

$$\begin{aligned} u^H Q_k u &= \gamma u^H u + \gamma u^H A_1 (S_k)_1 v \\ v^H (W_k E_k)_1\, v &= -\gamma (u^H A_1 (S_k)_1 v)^H + \gamma v^H ((\mu I + \Delta_k) S_k^2)_1 v \end{aligned}$$

Then, adding these two equations we obtain

$$\underbrace{u^H Q_k u + v^H (W_k E_k)_1\, v}_{\alpha} = \gamma\, (\underbrace{u^H u + v^H ((\mu I + \Delta_k) S_k^2)_1 v}_{\beta} + 2iZ) \tag{5.10}$$

where Z is the imaginary part of $u^H A_1 (S_k)_1 v$. Note that Q_k, $(W_k E_k)_1$, $((\mu I + \Delta_k) S_k^2)_1$ are positive semidefinite and α and β are real and strictly positive (if $\alpha = 0$ then u belongs to the null space of Q_k and v belongs to the null space of $(W_k E_k)_1$. These conditions imply $\lambda = 1$ and this is a contradiction. Analogously, if $\beta = 0$ then $u = 0$ and consequently $Z = 0$; thus (5.10) reduces to $v^H (W_k E_k)_1 v = 0$ and this is a contradiction because if $u = 0$ and v belongs to the null space of $(W_k E_k)_1$ then $\lambda = 1$). Thus,

$$\gamma = \frac{\alpha}{\beta + 2iZ} = \frac{\alpha\,(\beta + 2iZ)^H}{\beta^2 + 4Z^2} = \frac{\alpha\beta}{\beta^2 + 4Z^2} - i\frac{2\alpha Z}{\beta^2 + 4Z^2} \tag{5.11}$$

Then, the real and complex part of γ are such that $Re(\gamma) = \frac{\alpha\beta}{\beta^2 + 4Z^2}$ and $Im(\gamma) = -\frac{2\alpha Z}{\beta^2 + 4Z^2}$ which give (5.8) and (5.9), respectively. Since $\alpha > 0$ and $\beta > 0$ we conclude that $Re(\gamma)$ is positive. □.

Clearly, if the magnitude $|\gamma|$ of γ is small it means that the eigenvalues of $\mathcal{P}_k^{-1}\mathcal{A}_k$ are clustered around one and fast convergence of Krylov methods can be expected. This is the case when x_k is close to the solution. On the other hand, when x_k is still far away from x^*, the following bounds for $|\gamma|$ can be derived.

COROLLARY 5.1. *Let $\mathcal{A}_k$ and $\mathcal{P}_k$ be the matrices given in (5.6) and (5.7), τ be the scalar in (5.1), $\bar{\mathcal{L}}_k = \{i \in \mathcal{L}_k : \ (s_k)_i^2 \neq 1\}$.*

If the elements $\delta_{k,i}$ in (3.6) are such that $\delta_{k,i} = \delta > 0$ for $i \in \mathcal{L}_k$, and $\delta_{k,i} = 0$ for $i \notin \mathcal{L}_k$, then the eigenvalues of $\mathcal{P}_k^{-1}\mathcal{A}_k$ have the form $\lambda = 1 + \gamma$ and

$$|\gamma| \leq \frac{\|A_2 (S_k)_2\|_2^2}{\tau} + \frac{\tau}{(\delta + \mu)(1 - \tau)}. \tag{5.12}$$

If the elements $\delta_{k,i}$ in (3.6) are such that

$$\delta_{k,i} = \begin{cases} \max\{(w_k)_i (e_k)_i - \mu, 0\} & \text{if} \quad i \in \bar{\mathcal{L}}_k \\ \delta & \text{if} \quad i \in \mathcal{L}_k \backslash \bar{\mathcal{L}}_k \quad \text{and } \mu = 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.13}$$

then the eigenvalues of $\mathcal{P}_k^{-1}\mathcal{A}_k$ have the form $\lambda = 1 + \gamma$ and

$$|\gamma| \leq \frac{\|A_2 (S_k)_2\|_2^2}{\tau} + \frac{1}{1 - \tau}. \tag{5.14}$$

Proof. Suppose $u \neq 0$ and $v \neq 0$. Then, from (5.11) we have

$$|\gamma| = \frac{|\alpha|}{|\beta + 2iZ|} = \frac{\alpha}{\sqrt{\beta^2 + 4Z^2}} \leq \frac{\alpha}{\beta}.$$

Then, from the definition of α and β we get:

$$\begin{aligned} |\gamma| &\leq \frac{u^H Q_k u + v^H (W_k E_k)_1\, v}{u^H u + v^H ((\mu I + \Delta_k) S_k^2)_1 v} \\ &\leq \frac{u^H Q_k u}{u^H u} + \frac{v^H (W_k E_k)_1\, v}{v^H ((\mu I + \Delta_k) S_k^2)_1\, v}. \end{aligned} \tag{5.15}$$

Also observe that (5.1) implies

$$\min_{i \in \mathcal{L}_k} (s_k)_i^2 \geq 1 - \tau, \quad \|(W_k E_k)_1\|_2 \leq \tau, \quad \|(C_k)_2^{-1}\|_2 \leq \frac{1}{\tau}.$$

Then, when $\delta_{k,i} = \delta > 0$ for $i \in \mathcal{L}_k$, we obtain

$$|\gamma| \leq \|(C_k)_2^{-1}\|_2 \, \|A_2(S_k)_2\|_2^2 + \frac{\tau}{(\mu+\delta)(1-\tau)},$$

which yields (5.12).

Consider the case when $\delta_{k,i}$ is determined by (5.13). For $i \in \mathcal{L}_k \backslash \bar{\mathcal{L}}_k$, we have $(w_k)_i (e_k)_i = 0$. Then, we get

$$\begin{aligned}
|\gamma| &\leq \|(C_k)_2^{-1}\|_2 \, \|A_2(S_k)_2\|_2^2 + \frac{\sum_{i\in\mathcal{L}_k} (w_k)_i (e_k)_i v_i^2}{\sum_{i\in\mathcal{L}_k} (\delta_{k,i}+\mu)(s_k)_i^2 v_i^2} \\
&= \|(C_k)_2^{-1}\|_2 \, \|A_2(S_k)_2\|_2^2 + \frac{\sum_{i\in\bar{\mathcal{L}}_k} (w_k)_i (e_k)_i v_i^2}{\sum_{i\in\bar{\mathcal{L}}_k} \max\{(w_k)_i (e_k)_i, \mu)\}(s_k)_i^2 v_i^2 + \sum_{i\in\mathcal{L}_k\backslash\bar{\mathcal{L}}_k} (\delta_{k,i}+\mu)(s_k)_i^2 v_i^2} \\
&\leq \frac{\|A_2(S_k)_2\|_2^2}{\tau} + \frac{\sum_{i\in\bar{\mathcal{L}}_k} (w_k)_i (e_k)_i v_i^2}{\sum_{i\in\bar{\mathcal{L}}_k} (s_k)_i^2 (w_k)_i (e_k)_i v_i^2} \\
&\leq \frac{\|A_2(S_k)_2\|_2^2}{\tau} + \frac{1}{\min_{i\in\bar{\mathcal{L}}_k} (s_k)_i^2}.
\end{aligned}$$

Then (5.14) trivially follows from (5.1).

Finally, if either u or v is null the bound (5.15) simplifies to one of the two terms and the thesis still holds. □

The previous result (5.14) shows that the spectral properties of the preconditioned matrix are affected by the choice of the regularization term Δ_k. In fact, (5.13) indicates that if μ is sufficiently large it is not necessary to introduce the regularization term in order to get good spectral properties of the preconditioned system. In other words, in this latter case Δ_k may be chosen as the null matrix at each iteration. On the other hand, when μ is small compared to $(w_k)_i (e_k)_i$, letting $\delta_{k,i} = (w_k)_i (e_k)_i - \mu$ for $i \in \mathcal{L}_k$ we have a better distribution of the eigenvalues of $\mathcal{P}_k^{-1}\mathcal{A}_k$. Note that for any regularization used, it is essential to keep the term $\|A_2(S_k)_2\|_2^2/\tau$ as small as possible. Hence, we advise scaling matrix A at the beginning of the solution process to guarantee that the norm $\|A\|_2$ is small.

Let us now consider the application of PPCG method for solving the preconditioned iterative system. PPCG is a conjugate-gradient like method for solving preconditioned block symmetric indefinite linear systems that arise from saddle-point problems and our preconditioned augmented systems fall in this class of problems. Taking into account that PPCG requires the second block of the right hand side to be the null vector, we introduce the vector $p_k^* \in \mathbb{R}^{n_1}$ defined as $p_k^* = -\mu (C_k)_1^{-1} (S_k)_1 (x_k)_1$ and rewrite system (5.6) in the following form

$$\underbrace{\begin{pmatrix} I + Q_k & A_1(S_k)_1 \\ (S_k)_1 A_1^T & -(C_k)_1 \end{pmatrix}}_{\mathcal{A}_k} \begin{pmatrix} \tilde{q}_k \\ (\tilde{p}_k)_1 - p_k^* \end{pmatrix} = \begin{pmatrix} -(Ax_k - b) + \hat{R}_k \\ 0 \end{pmatrix}, \tag{5.16}$$

where $\hat{R}_k = R_k - A_1 (S_k)_1 p_k^*$. Solving (5.16) with preconditioner $\mathcal{P}_k$ by PPCG is equivalent to applying Preconditioned Conjugate Gradient (PCG) method to the system

$$\underbrace{(I + Q_k + A_1(S_k C_k^{-1} S_k)_1 A_1^T)}_{\mathcal{F}_k} \tilde{q}_k = -(Ax_k - b) + \hat{R}_k, \tag{5.17}$$

using the preconditioner

$$\mathcal{G}_k = I + A_1(\Delta_k + \mu I)_1^{-1} A_1^T, \tag{5.18}$$

see [11]. The distribution of the eigenvalues for matrix $\mathcal{G}_k^{-1}\mathcal{F}_k$ is summarized in the following theorem.

THEOREM 5.2. *Let $\mathcal{F}_k$ and $\mathcal{G}_k$ be the matrices given in (5.17) and (5.18), $\mathcal{L}_k$ be the set given in (5.1). If $\delta_{k,i}$ are given by (5.13) then the eigenvalues λ of $\mathcal{G}_k^{-1}\mathcal{F}_k$ satisfy*

$$1 - \frac{1}{2-\tau} \leq \lambda \leq 1 + \frac{\|A_2(S_k)_2\|_2^2}{\tau}. \tag{5.19}$$

Proof. The proof follows straightforwardly from the proof of Theorem 4.2 in [2] . □

6. Scaling of the problem. Numerical experience with the GREN and the AS_CBB methods shows that scaling of the problem is crucial for the performance of such algorithms. As will be shown in §9, robustness and efficiency of the AS_CBB method seem to be favourably affected by the scaling. Further, a proper scaling enhances the linear algebra phase of GREN method and influences the convergence of the sequence $\{x_k\}$ generated by the hybrid method.

Without lack of generality, let us assume that all the columns of A contain at least one nonzero entry. We apply a scaling from the right-hand side and divide each column of A by the sum of the absolute values of its entries. Let $F \in \mathbb{R}^{n\times n}$ be the diagonal matrix where jth entry is the sum of the absolute values of the jth column of A. Thus, our method is applied to the following equivalent problem

$$\min_{x\geq 0} q(x) = \min_{\bar{x}\geq 0} \bar{q}(\bar{x}) = \frac{1}{2}\|AF^{-1}\bar{x} - b\|_2^2 + \frac{\mu}{2}\|F^{-1}\bar{x}\|_2^2, \qquad \bar{x} = Fx. \tag{6.1}$$

The relation between $g(x) = \nabla q(x)$ and $\bar{g}(\bar{x}) = \nabla \bar{q}(\bar{x})$ is

$$\bar{g}(\bar{x}) = F^{-1} g(x). \tag{6.2}$$

Thus $\|\bar{g}(\bar{x})\|_2 = \|g(x)\|_{F^{-2}}$.

When such a scaling of the problem is performed, the hybrid method generates the sequence $\{\bar{x}_k\}$, the augmented system (5.16) takes the form

$$\underbrace{\begin{pmatrix} I + Q_k & A_1 F_1^{-1}(S_k)_1 \\ (S_k)_1 F_1^{-1} A_1^T & -(\tilde{C}_k)_1 \end{pmatrix}}_{\mathcal{A}_k} \begin{pmatrix} \tilde{q}_k \\ (\tilde{p}_k)_1 - p_k^* \end{pmatrix} = \begin{pmatrix} -(AF^{-1}\bar{x}_k - b) + \hat{R}_k \\ 0 \end{pmatrix},$$

where

$$\begin{aligned} \hat{R}_k &= \mu A_2 F_2^{-1}(S_k \tilde{C}_k^{-1} S_k)_2 F_2^{-2}(\bar{x}_k)_2 - A_1 F_1^{-1}(S_k)_1 p_k^*, \\ Q_k &= A_2 F_2^{-1}(S_k \tilde{C}_k^{-1} S_k)_2 F_2^{-1} A_2^T, \\ \tilde{C}_k &= W_k E_k + (\mu F^{-2} + \Delta_k) S_k^2, \\ p_k^* &= -\mu F_1^{-2}(\tilde{C}_k^{-1} S_k)_1 \bar{x}_k, \end{aligned}$$

and the preconditioner $\mathcal{P}_k$ is given by

$$\mathcal{P}_k = \begin{pmatrix} I & A_1 F_1^{-1}(S_k)_1 \\ (S_k)_1 F_1^{-1} A_1^T & -((\mu F^{-2} + \Delta_k) S_k^2)_1 \end{pmatrix}. \tag{6.3}$$

7. Implementation of the hybrid method. In this section we address the implementation issues, the choice of default parameters for the hybrid method, the preconditioner's factorization, the choice of the regularization matrix and the stopping criteria.

The parameter $\beta = 0.1$ is used within the GREN algorithm. For the AS_CBB algorithm we followed [15] and set

$$c = 4, \quad C = 6, \quad \delta = 10^{-4}, \quad \gamma = 0.5, \quad \bar{\lambda} = 10^{-2}. \tag{7.1}$$

A maximum number of 10 backtracks were allowed in Step 4 of the AS_CBB algorithm. If the criterion (2.7) was not satisfied within 10 backtracks we proceeded with the last computed iterate.

The switching to the AS_CBB algorithm was performed using the parameters

$$t_u = 0.8, \quad \omega_1 = \sqrt{\epsilon_m}, \quad I_{CBB}^{max} = 10.$$

Further, in our implementation the switch to I_{CBB}^{max} AS_CBB iterations was activated in the case where

$$\frac{q_k - q_{k-1}}{1 + q_k} \leq 10^{-4}.$$

This rule is motivated by the fact that the nonmonotone globalization strategy can be beneficial to convergence if we are far from the solution and a stagnation occurs.

The factorization of the preconditioner $\mathcal{P}_k$ given in (6.3) can be accomplished based on the identity

$$\mathcal{P}_k = \begin{pmatrix} I & 0 \\ 0 & (S_k)_1 \end{pmatrix} \underbrace{\begin{pmatrix} I & A_1 F_1^{-1} \\ F_1^{-1} A_1^T & -(\mu F^{-2} + \Delta_k)_1 \end{pmatrix}}_{\Pi_k} \begin{pmatrix} I & 0 \\ 0 & (S_k)_1 \end{pmatrix}, \tag{7.2}$$

and factorizing Π_k. The LDL^T factorization of Π_k is carried out via the Cholesky factorization of the matrix $A_1^T A_1$. We stress that if the set $\mathcal{L}_k$ and the matrix Δ_k remain unchanged for a few iterations, the factorization of the matrix Π_k does not have to be updated. In fact, eventually $\mathcal{L}_k$ is expected to settle down as it contains the indices of all the components of x^* such that $(s_k)_i$ tends to one.

The rules we used to update the matrix Δ_k and the set $\mathcal{L}_k$ are the following. The entries of $\Delta_k = diag(\delta_{k,1}, \delta_{k,2}, \ldots, \delta_{k,n})$, $\delta_{k,i} \in [0,\, 1),\ i = 1, \ldots, n$ are given by

$$\delta_{k,i} = \begin{cases} 0, & \text{if } i \notin \mathcal{L}_k \text{ and } \max\{F_{ii}^{-2}\mu, (w_k)_i(e_k)_i\} > 10^{-8} \\ 10^{-8}, & \text{if } i \notin \mathcal{L}_k \text{ and } \max\{F_{ii}^{-2}\mu, (w_k)_i(e_k)_i\} \leq 10^{-8} \\ 0, & \text{if } i \in \mathcal{L}_k \text{ and } F_{ii}^{-2}\mu > \max\{10^{-8}, (w_k)_i(e_k)_i)\} \\ \min\{\max\{10^{-8}, (w_k)_i(e_k)_i - F_{ii}^{-2}\mu\}, 10^{-2}\}, & \text{otherwise.} \end{cases}$$

Further, we freeze the set $\mathcal{L}_k$ and $(\Delta_k)_1$ either if at kth iteration the iterative linear solver has succeeded within 30 iterations and

$$|card(\mathcal{L}_{k+1}) - card(\mathcal{L}_k)| \leq 10, \quad \|(\Delta_k + \mu F^{-2})_1^{-1}(W_k E_k)_1\|_\infty \leq 100 \tag{7.3}$$

or if at kth iteration the iterative linear solver has not succeeded within 30 iterations but

$$\mathcal{L}_{k+1} = \mathcal{L}_k, \quad \|(\Delta_k + \mu F_1^{-2})^{-1}(W_k E_k)_1\|_\infty \leq 100 \tag{7.4}$$

Some comments on the choice of the regularization term and conditions (7.3)-(7.4) are in order. The quality of the preconditioner depends on the size of the regularization term and on the ratio $\dfrac{(w_k e_k)_i}{\delta_{k,i} + \mu F_{i,i}^{-2}}$ (see §5 and the analysis performed in [2]). Taking into account the presence of the matrix F, it seems that the best choice is $\delta_{k,i} = \max\{(w_k e_k)_i - \mu F_{i,i}^{-2}, 0\}$ for any $i \in \bar{\mathcal{L}}_k$. This choice also guarantees fast convergence as the regularization sequence converges to zero when x_k approaches the solution (see [2], Theorem 3.1). In practice, we impose the threshold 10^{-8} in order to avoid too small regularization terms. Moreover the set $\mathcal{L}_k$ and the regularization $(\Delta_k)_1$ are frozen only if the ratio $(w_k e_k)_i/(\delta_{k,i} + \mu F_{i,i}^{-2})$ is less than or equal to 100. This way the quality of the preconditioner is preserved. Finally, it must be taken into account that, after a freezing, it may happen that an index i such that $(w_k)_i(e_k)_i = 0$ does not belong to $\mathcal{L}_k$ and this yields the singularity of $(C_k)_2$ if $\mu = 0$. In order to prevent this latter situation, we set the regularization term to 10^{-8}, whenever $i \notin \mathcal{L}_k$ and $\max\{(w_k)_i(e_k)_i, \mu F_{i,i}^{-2}\} \leq 10^{-8}$.

We conclude this section by discussing the stopping criteria that we used. Due to (6.2), the norms of $P(\bar{x}_k - \bar{g}(\bar{x}_k)) - \bar{x}_k$ and $P(x_k - g_k) - x_k$ may differ significantly. Since we are solving the scaled problem (6.1), it is appropriate to check $\|P(\bar{x}_k - \bar{g}(\bar{x}_k)) - \bar{x}_k\|_\infty$ as a measure of optimality but to avoid stopping at a point x_k where $\|P(x_k - g_k) - x_k\|_\infty$ is large, we control both $\|P(\bar{x}_k - \bar{g}(\bar{x}_k)) - \bar{x}_k\|_\infty$ and $\|P(x_k - g_k) - x_k\|_\infty$.

At kth iteration we test either if

$$\begin{cases} q_{k-1} - q_k < \tau_1(1 + q_{k-1}), \\ \|P(\bar{x}_k - \bar{g}(\bar{x}_k)) - \bar{x}_k\|_\infty < \tau_1(1 + \min\{\|\bar{g}(\bar{x}_0)\|_\infty, \tau_u\}), \\ \|P(x_k - g_k) - x_k\|_\infty < \tau_g, \end{cases} \tag{7.5}$$

or

$$\begin{cases} \|P(\bar{x}_k - \bar{g}(\bar{x}_k)) - \bar{x}_k\|_\infty < \tau_2(1 + \min\{\|\bar{g}(\bar{x}_0)\|_\infty, \tau_u\}), \\ \|P(x_k - g_k) - x_k\|_\infty < \tau_g, \end{cases} \tag{7.6}$$

A failure is declared when the above conditions are not satisfied within a fixed number of nonlinear iterations.

8. The algorithms used for the comparison. The numerical performance of the hybrid method proposed was compared with the performance of the AS_CBB method described in §2, the GREN method described in §3 and with the BCLS software package [13, 14].

The GREN method was implemented as the hybrid method inhibiting the switch to AS_CBB strategy. The AS_CBB method was implemented in `Matlab` with the algorithmic parameters (7.1) and two possible strategies for setting the reference function value q_k^r. The first strategy sets $q_k^r = q_k^{max}$ at each iteration; this yields $q_R = q_k^{max}$ in (2.7). The second strategy is the one described in [16, Appendix A] where q_k^r is set equal to q_k^{max} after a prescribed number of iterations if a sufficient reduction of q has been achieved. We underline that all the numerical results reported in the next session are obtained using the first strategy for selecting q_k^r since in our runs this choice turned out to be slightly more effective than the other.

The stopping criteria for successful termination are given in (7.6). A maximum number of 10 backtracks is allowed in Step 4 of the AS_CBB algorithm. If the criterion (2.7) is not satisfied within 10 backtracks we declare a failure.

The comparison of the hybrid method with the AS_CBB and GREN algorithms allows us to establish if our strategy strengthens such approaches and is competitive from a computational point of view.

BCLS is an ISO C-code for solving bound-constrained least-squares problems of the form

$$\min_{l \le x \le u} \frac{1}{2}\|Ax - b\|_2^2 + c^T x + \frac{1}{2}\mu\|x\|_2^2,$$

where the m-by-n matrix can be any shape, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ and μ is a nonnegative regularization parameter [13, 14].

The BCLS algorithm is based on a two-metric projection method where the variables are partitioned into two sets. Variables that are well within the interior of the feasible set are classified as free (B); variables that are at or near the bounds are labelled as fixed (N). Two independent directions Δx_B and Δx_N for the free and fixed components of x are generated at each iteration. The step Δx_B is a Newton step and Δx_N is a scaled steepest descent step. Then, the aggregated step $(\Delta x_B, \Delta x_N)$ is projected into the feasible region and the first minimizer is computed along the piecewise linear projected-search direction, [8]. The step Δx_B is computed as a solution of a least-squares problem by applying the LSQR method [18] and a preconditioner can be supplied by the user. Note that BCLS reduces the dimension of the linear system to be solved at each iteration but it does not provide an internal preconditioner.

BCLS has been run using a precompiled Mex interface available in [13]. A preconditioner for LSQR was not supplied and default values for the parameters were used. In particular, BCLS successfully terminates if

$$\|\hat{g}_k\|_\infty < \tau(1 + \|A^T b\|_2), \tag{8.1}$$

where $\tau = 10^{-3}$ and

$$(\hat{g}_k)_i = \begin{cases} (g_k)_i \min\{(x_k)_i - l_i, 1\} & \text{if } (g_k)_i \ge 0, \\ -(g_k)_i \min\{u_i - (x_k)_i, 1\} & \text{if } (g_k)_i < 0. \end{cases}$$

A failure is declared if (8.1) is not satisfied within $5n$ nonlinear iterations or $10n$ linear iterations.

The comparison of the hybrid method with the BCLS code allows to assess if our interior-point approach is competitive with a strategy that identifies active set variables.

9. Numerical results. All tests were performed on a Intel Xeon (TM) 3.4 Ghz, 1GB RAM using MATLAB 7.6 with machine precision $\epsilon_m \sim 2 \cdot 10^{-16}$.

The problems selected include several tests where both the GREN method and the AS_CBB method failed to converge. This allows us to highlight the good properties of the hybrid method. We considered 60 matrices from The University of Florida Sparse Matrix Collection [19]; these matrices belong to 21 different groups, are full rank and such that $n \ge 4000$ and $\|A^T b\|_2 \le 10^{20}$. In case of nonsquare matrix with $m < n$, we used the transpose. Table (9.1) displays the name of the matrix and its group, the dimensions m, n and the number nnz of nonzero entries of A. Moreover, the vector b is set equal to $b = -Ae$, where e is the vector of all ones and μ is set to zero. In the following we refer to each obtained test problem with the name of the matrix A used.

All algorithms used for the comparison were applied with and without the scaling described in §6. The initial guess x_0 is the vector $F^{-1}e$, that is we start the iterative process with $\bar{x}_0 = e$.

The tolerances for the stopping criteria (7.5) and (7.6) are

$$\tau_1 = 10^{-6},\ \tau_2 = 10^{-9},\ \tau_u = 10^{3},\ \tau_g = 10^{-2} \tag{9.1}$$

The maximum number of nonlinear iterations allowed for the hybrid method, the AS_CBB method and the GREN method is 5000, 20000 and 500, respectively.

When $\mathcal{L}_k \neq \emptyset$, the linear system (5.6) is solved using PPCG. The linear systems are solved with a low accuracy when the current iterate is far from the solution and increased accuracy as the solution is approached. Specifically, following [2], the forcing term η_k in (3.18) is

$$\eta_0 = 0.5, \quad \eta_k = \max\{500\,\epsilon_m, \min\{10^{-3}, 10^{-2}\|W_k D_k g_k\|_2\}\},\ k \geq 1, \tag{9.2}$$

thus η_k decreases as $\|W_k D_k g_k\|_2$ and fast convergence is ensured.

As already underlined in §5, when $\mathcal{L}_k = \emptyset$, $\tilde{p}_k$ is computed approximately solving (3.4) by the unpreconditioned CG method.

We restricted the CG and PPCG methods to use a maximum of 100 iterations. If the stopping criterion is not satisfied within 100 iterations, the algorithm proceeds with the last computed iterate.

The successful runs of the hybrid method on the scaled problems (6.1) are shown in details in Table 9.2. For every problem we report: the total number `it` of iterations performed; the number `it_new` of iterations where the Newton step was computed; the number `h_bt` of backtracks performed in the AS_CBB phase; the overall number `prec` of preconditioner updates and factorizations; the average number `PPCG_a` of PPCG iterations performed; the number `Aprod` of matrix-vector products computed and, in brackets, the number of matrix-vector products needed excluding the linear algebra phase; the values of q at the computed solution x_c and the infinity norm $g_p(x_c)$ of the projected gradient at x_c, i.e. $g_p(x_c) = \|P(x_c - g(x_c)) - x_c\|_\infty$. All failures of the hybrid method occurred as the maximum number of allowed nonlinear iterations was reached.

The hybrid method was able to solve 46 problems (77% of the problem set). For some problems, our solver returns quite a large value $g_p(x_c)$. We solved all the problems for which $g_p(x_c) \in [10^{-2}, 10^{-1})$ again with a higher accuracy, that is, using $\tau_1 = 10^{-9}$ instead of 10^{-6}. From the results obtained, we can conclude that the approximate solutions computed with the tolerances (9.1) are accurate, except for one test, even if $g_p(x_c)$ is not small and that the implemented stopping criteria works properly. In fact, the relative error between the two solutions, computed using both $\tau_1 = 10^{-9}$ and $\tau_1 = 10^{-6}$, is at least of order 10^{-6} for all tests except for `Oberwolfach/flowmeter0` where the lower tolerance produces a more accurate solution.

On successful runs, typically the number `it_new` of iterations where the Newton step was computed is considerably smaller than the total number `it` of iterations performed. Further, 16 problems were solved using only CG for computing the Inexact Newton step and therefore without using preconditioning strategy. On the remaining 30 successfully solved tests, comparing the values of `it_new` and `prec`, we see that freezing of the preconditioner occurred a few times in 11 problems.

The combination of the GREN method with the Barzilai-Borwein strategy considerably enhances the performance of the GREN method. In fact, the GREN method solves 34 problems (57% of the problem set) and in case of successful runs the elapsed time is higher than that of the hybrid method. The elapsed time performance profile for the successful runs is displayed in Figure 9.1. In this figure for each algorithm, we

plot the fraction of problems for which the algorithm is within a given factor of the best CPU time.

The AS_CBB method was able to solve 36 problems (60% of the problem set). We underline that all the failures of AS_CBB are due to the fact that the maximum number of allowed nonlinear iterations was performed without satisfying the stopping criteria. The numerical behaviour of the AS_CBB method on succesfull runs is reported in Table 9.3. For every problem we report: the number `it` of iterations performed; the number `Aprod` of matrix-vector products computed, the values of q and the infinity norm $g_p(x_c)$ of the projected gradient $P(x - g(x)) - x$ at the computed solution x_c.

The results obtained show that the hybrid method is more robust than AS_CBB method. Concerning the computational cost, we note that there are several problems where AS_CBB converges very fast; in particular for 21 problems it requires less than 50 matrix-vector products and clearly, it solves these problems with a very low computational cost. On the other hand, the hybrid method favourably compares with the AS_CBB method in terms of computational time on more difficult tests. Table 9.4 displays the time in seconds required by the hybrid method and the AS_CBB method; in this table we restrict the comparison to problems solved successfully by both algorithms and such that at least one algorithm requires less than 1 second. Table 9.5 shows the time in seconds for runs where both algorithms require more than 1 second. Focusing on these tests, we can see from Table 9.5, that the hybrid method is more efficient than AS_CBB on 10 out of 13 tests and the execution time is more than halved for six tests. On the other hand, we note the disappointing performance of the hybrid method on two examples from the Parsec family.

We would like to stress that for the hybrid method the number of matrix-vector products is not the only factor which determines its computational cost. The overall effort of the method is also influenced by the number `prec` of preconditioner factorizations. On the other hand, the number of matrix-vector products is an accurate measure of the cost of the AS_CBB method. However, it should be taken into account that the execution time depends on the number of matrix-vector products and on the number of nonzero elements of the matrix A as well. For example, if we compare the behaviour of AS_CBB on `Parsec/Na5` and `Shenk_IBMNA/c-42`, the substantial execution time to perform 74720 matrix-vector products, compared to the execution time of solving `Shenk_IBMNA/c-42`, can be explained by the 305000 nonzero entries in the matrix A of `Parsec/Na5` against the 110000 nonzeros entries of the matrix A of `Shenk_IBMNA/c-42`.

Finally, in the upper portion of Figure 9.2 we compare the behaviour of the hybrid method and AS_CBB method in terms of accuracy of the computed solution in case of successful runs for both methods. Let x_{AS_CBB} and x_{hybrid} be the computed solutions provided by AS_CBB and the hybrid method, respectively. In this figure, we plot, with a logarithmic (base 10) scale for the y-axis, the value of the ratio:

$$r_{AS\_CBB} = \frac{g_p(x_{AS\_CBB})}{g_p(x_{hybrid})}$$

for each problem successfully solved by both methods. The figure clearly shows that the hybrid method reaches a lower level of accuracy than AS_CBB only for eight tests.

Next we report results obtained with BCLS. BCLS seems to be slightly more robust and computationally cheaper if scaling is applied. In particular, it solves 45 problems (75%) when no scaling is applied and 48 problems (80%) when scaling is

applied. Proper scaling of the problem affects the linear algebra phase; if the problem is badly scaled, BCLS may fail as the maximum number of allowed LSQR inner iterations is reached.

Let us focus on the application of BCLS on scaled problems. In Table 9.6 we report solution statistics for BCLS method: the number `major-itns` of iterations performed; the number `minor-itns` of linear iterations performed; the number `Aprod` of matrix-vector products computed and, in brackets, the number of matrix-vector products needed to perform the globalization strategy; the values of q and the infinity norm g_p of the projected gradient at the computed solution x_c. We report statistics only for problems successfully solved by BCLS; all the failures declared by BCLS code are due to the fact that the maximum number of linear iterations allowed was reached.

BCLS is able to solve 48 problems (80%) of the problem set but some comments on the returned norm $g_p(x_c)$ are in order. For 14 problems the value of g_p is equal to zero and this feature can be ascribed to the active set strategy employed as the solution of these tests is the null vector and therefore all the constraints are active at the solution. On the other hand, in case of 10 successful runs we have obtained $g_p(x_c) > 10^{-1}$ and in seven among these tests, BCLS has declared successful termination although $g_p(x_c) \geq 1$. We explain these occurrences noting that BCLS does not apply an internal scaling and in (8.1) the gradient $\bar{g}$ of the scaled problem is checked.

A comparison of the values of the norm of the projected gradient at the solution computed by the hybrid method and BCLS method is shown in the bottom portion of Figure 9.2. In this figure, for each problem successfully solved by both methods (runs where the norm of projected gradient at the computed solution is zero are excluded), we plot, with a logarithmic (base 10) scale for the y-axis, the value of the ratio:

$$r_{bcls} = \frac{g_p(x_{bcls})}{g_p(x_{hybrid})}$$

where x_{bcls} is the approximate solution provided by BCLS. We note that the level of accuracy reached by the hybrid method is lower than that reached by BCLS only in six tests.

Performing further runs with the tighter tolerance $\tau_1 = 10^{-9}$, on problems where $g_p(x_c) > 10^{-1}$, we observed that the level of accuracy in the computed solution increases considerably when tighter tolerances are used. We think that providing a scaling of the problem and a control of the original gradient $\nabla q(x)$ is a good feature of our implementation which distinguishes it from BCLS.

Focusing on the iterations where a Newton step is computed, we see that the number `it_new` of iterations performed by our code is similar to the number `major-itns` of major iterations performed by BCLS. In Figure 9.3 we draw the performance profile comparing the values of `it_new` and `major-itns` on successful runs.

Regarding the number of matrix-vector products `Aprod`, our method requires a lower number of products than BCLS. This saving can be ascribed to the use of our built-in preconditioner but it is not easy to compare the computational effort of the two codes as the cost of our preconditioning technique may vary substantially and is difficult to estimate. A fair comparison of the computational cost can be performed restricting to problems where our code did not require preconditioning. We recall that this is the case when the set $\mathcal{L}_k$ defined in (5.1) is empty, see (5.2). For such problems, in Figure 9.4 we draw the ratio of the number `Aprod` of matrix-vector products required by BCLS and our code respectively; the height of the dotted line is equal to 50. This ratio is always greater than one and lower than 50 for 8 problems.

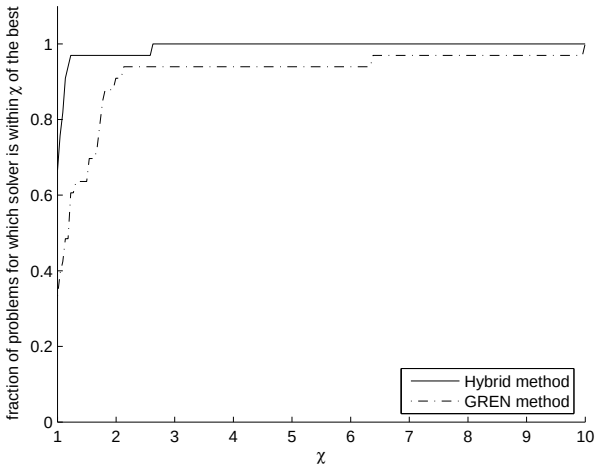

FIG. 9.1. *Performance profile: elapsed times over successful runs*

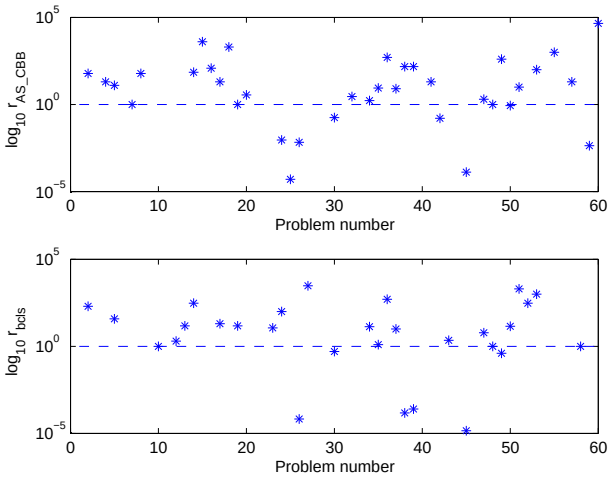

FIG. 9.2. *Accuracy comparison on successful runs*

From Figure 9.4 we can see that in the solution of these tests our method needs considerably fewer matrix-vector products than BCLS.

10. Conclusions. A method for box-constrained linear least-squares has been discussed in this paper. It combines ideas of a cyclic Barzilai-Borwein and globally convergent regularized Newton-like method to guarantee global and fast local convergence. Its major computational step consists in the solution of the Newton equation. This step is executed by a suitably preconditioned iterative method. The preconditioner combines the idea of regularization and makes guess of active constraints. Extensive computational results based on Matlab implementation illustrate that the new method strengthens both the cyclic Barzilai-Borwein method and the regularized Newton-like method and that it compares favourably with the BCLS code.

Group/Test name	m	n	nnz
Bai/cryg10000	10000	10000	49699
Bai/dw8192	8192	8192	41746
Bai/olm5000	5000	5000	19996
Bai/rw5151	5151	5151	20199
Boeing/bcsstm39	46772	46772	46772
Boeing/msc23052	23052	23052	1142686
Boeing/pcrystk03	24696	24696	1751178
Boeing/pct20stif	52329	52329	2698463
DRIVCAV/cavity16	4562	4562	137887
DRIVCAV/cavity26	4562	4562	138040
GHS_indef/bloweybq	10001	10001	49999
GHS_indef/bratu3d	27792	27792	173796
GHS_indef/sit100	10262	10262	61046
GHS_indef/spmsrtls	29995	29995	229947
GHS_psdef/copter1	17222	17222	211064
GHS_psdef/ford1	18728	18728	101576
GHS_psdef/jnlbrng1	40000	40000	199200
GHS_psdef/opt1	15449	15449	1930655
Gset/G67	10000	10000	40000
HB/bcspwr10	5300	5300	21842
HB/gemat1	10595	4929	46591
HB/gemat12	4929	4929	33044
HB/sherman3	5005	5005	20033
LPnetlib/lp_dfl001	12230	6071	35632
LPnetlib/lp_osa_60	243246	10280	1408073
LPnetlib/lp_qap15	22275	6330	94950
LPnetlib/lp_ stocfor3	23541	16675	72721
Mallya/lhr07	7337	7337	18427
Mathworks/Kuu	7102	7102	340200
Mathworks/Muu	7102	7102	170134
Mathworks/Pd	8081	8081	13036
Meszaros/bas1lp	9852	5411	587775
Meszaros/co5	12325	5774	57993
Meszaros/cq9	21534	9278	96653
Meszaros/deter1	15737	5527	32187
Meszaros/nl	15325	7039	47035
Mittelmann/fome13	97840	48568	285056
Mittelmann/nug08-3rd	29856	19728	148416
Nasa/barth5	15606	15606	61484
Nasa/nasa4704	4704	4704	104756
Nasa/shuttle_eddy	10429	10429	103599
Nasa/skirt	12598	12598	196520
Oberwolfach/flowmeter0	9669	9669	67391
Oberwolfach/gyro	17361	17361	1021159
Oberwolfach/rail_5177	5177	5177	35185
Parsec/Benzene	8219	8219	242669
Parsec/Na5	5832	5832	305630
Parsec/SiH4	5041	5041	171903
Rajat/rajat09	24482	24482	105573
Schenk_IBMNA/c-42	10471	10471	110285
Schenk_IBMNA/c-48	18354	18354	166080
Schenk_IBMNA/c-53	30235	30235	355139
Schenk_IBMNA/c-61	43618	43618	310016
Schenk_IBMSDS/2D_27628_bjtcai	27628	27628	206670
Simon/appu	14000	14000	1853104
Tkk/cbukle	13681	13681	676515
Tkk/g3rmt3m3	5357	5357	207695
Tkk/t2d_q4	9801	9801	87025
Tkk/tube1	21498	21498	897056
VanHenkelum/cage11	39082	39082	559722

TABLE 9.1
Test problems

Group/Test name	it	it_new	h_bt	prec	PPCG_a	Aprod	$q(x_c)$	$g_p(x_c)$
Bai/dw8192	324	34	880	34	7	3696(3158)	2.00420E+3	1.E-5
Bai/rw5151	6	5	1	0	2	62(36)	2.57550E+3	2.E-9
Boeing/bcsstm39	6	5	1	3	2	75(41)	2.55329E+5	8.E-7
Boeing/pcrystk03	3	3	0	0	1	32(20)	6.45662E+7	4.E-10
Boeing/pct20stif	4	3	1	0	1	36(24)	7.71187E+7	5.E-7
DRIVCAV/cavity26	3378	310	9261	310	9	39460(32964)	3.46916E+3	1.E-4
GHS_indef/bratu3d	76	31	90	31	4	901(579)	8.21465E+2	2.E-5
GHS_indef/sit100	646	71	1491	71	8	6997(5771)	4.22135E+2	4.E-6
GHS_indef/spmsrtls	94	14	141	14	1	1010(700)	2.56428E+5	1.E-6
GHS_psdef/copter1	4	3	1	0	1	36(24)	1.57801E+6	1.E-10
GHS_psdef/ford1	5	4	1	0	1	48(30)	3.00930E+5	5.E-12
GHS_psdef/jnlbrng1	118	18	184	18	6	1138(890)	2.10955E+2	1.E-6
GHS_psdef/opt1	3	3	0	0	1	32(20)	1.34586E+8	1.E-10
Gset/G67	38	8	36	8	10	424(250)	1.90595E+4	2.E-5
HB/bcspwr10	5	4	1	0	1	48(30)	5.05190E+4	2.E-11
HB/gemat1	408	38	1040	38	10	4630(3824)	2.76050E+8	8.E-3
HB/sherman3	195	45	303	40	9	2393(1523)	8.11138E+5	7.E-2
LPnetlib/lp_dfl001	36	7	29	7	11	405(233)	1.61473E+4	1.E-4
LPnetlib/lp_osa_60	3	2	1	0	1	26(18)	3.73599E+7	4.E-9
LPnetlib/lp_qap15	4	4	0	2	2	52(30)	1.76400E+5	3.E-6
LPnetlib/lp_ stocfor3	3171	294	9611	252	5	36096(32788)	3.58317E+8	1.E-2
Mathworks/Muu	12	9	3	8	4	170(82)	3.30168E-4	4.E-8
Meszaros/bas1lp	6	2	4	0	2	44(34)	3.89051E+8	7.E-7
Meszaros/cq9	197	27	481	27	10	2415(1833)	7.55190E+6	3.E-2
Meszaros/deter1	66	16	100	16	9	842(512)	4.25901E+1	8.E-5
Meszaros/nl	405	45	931	43	7	4367(3617)	1.37931E+5	4.E-5
Mittelmann/fome13	36	7	33	7	11	403(233)	1.29178E+5	1.E-4
Mittelmann/nug08-3rd	11	4	7	2	2	94(70)	3.43392E+5	2.E-7
Nasa/barth5	5	4	1	0	1	48(30)	1.24284E+5	4.E-12
Nasa/shuttle_eddy	4	4	0	0	1	42(26)	5.41577E+5	3.E-11
Nasa/skirt	5	3	2	0	3	42(30)	1.78356E+6	5.E-6
Oberwolfach/flowmeter0	265	249	38	249	5	5033(1885)	1.39057E+2	9.E-2
Oberwolfach/rail_5177	37	10	69	8	17	625(265)	4.94290E-6	7.E-7
Parsec/Benzene	71	41	62	41	7	1191(533)	9.68629E+3	6.E-4
Parsec/Na5	61	23	66	23	8	851(447)	1.95586E+3	5.E-5
Parsec/SiH4	47	17	59	17	6	601(359)	6.40972E+3	3.E-4
Rajat/rajat09	4	4	0	0	1	44(26)	2.42823E+5	5.E-8
Schenk_IBMNA/c-42	266	39	599	39	13	3447(2371)	1.79401E+8	7.E-2
Schenk_IBMNA/c-48	1041	96	2951	96	8	12042(10352)	2.50827E+11	1.E-2
Schenk_IBMNA/c-53	3481	323	13484	318	8	47499(41853)	1.42336E+7	3.E-2
Schenk_IBMNA/c-61	395	37	1167	34	8	4655(4013)	2.42598E+8	2.E-4
Simon/appu	4	3	1	0	1	36(24)	1.69528E+6	1.E-10
Tkk/g3rmt3m3	4	3	1	0	1	36(24)	4.15968E+6	1.E-11
Tkk/t2d_q4	39	11	43	8	14	605(273)	1.82855E+1	1.E-5
Tkk/tube1	3	3	0	0	1	32(20)	1.90115E+7	7.E-10
VanHenkelum/cage11	6	5	1	0	1	60(36)	1.97780E+4	2.E-11

TABLE 9.2
Results obtained with the hybrid method on scaled problems

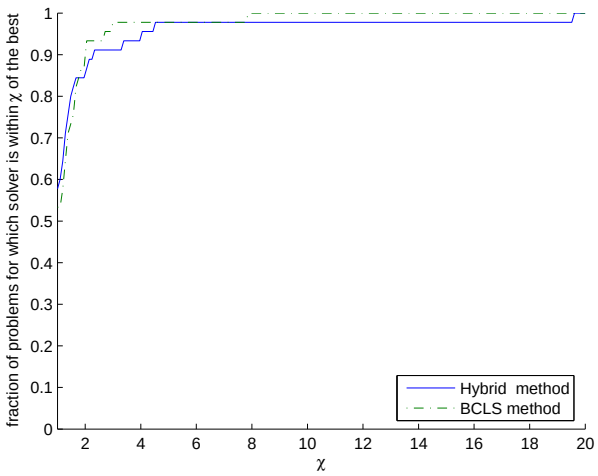

FIG. 9.3. *Performance profile in terms of Newton step computations (i.e. linear systems solved)*

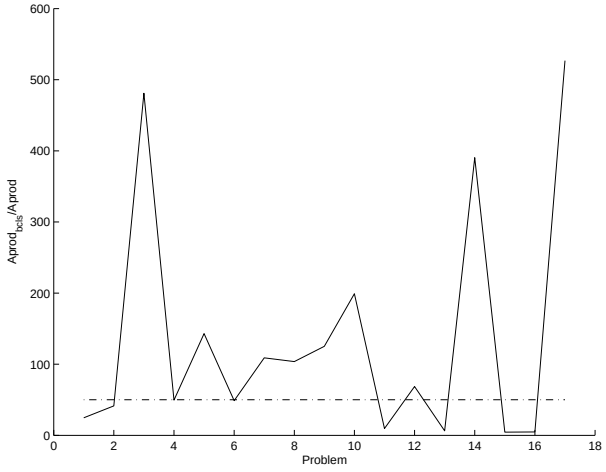

FIG. 9.4. *Ratio of matrix-vector products required by BCLS and the hybrid method on problems where preconditioning is not applied.*

Acknowledgments. We would like to thank Alessandro Melani for implementing the hybrid method in a `Matlab` code.

REFERENCES

[1] J. Barzilai, J.M. Borwein, *Two point step size gradient methods*, IMA J. Numer. Anal. 8, pp. 141-148, 1988.

[2] S. Bellavia, J. Gondzio, B. Morini, *Regularization and preconditioning of KKT systems arising in nonnegative least-squares problems*, Numerical Linear Algebra with Applications, 16, pp. 39-61, 2009.

[3] S. Bellavia, M. Macconi, B. Morini, *An interior Newton-like method for nonnegative least-squares problems with degenerate solution*, Numerical Linear Algebra with Applications, 13, pp. 825-846, 2006.

Group/Test name	it	Aprod	$q(x_c)$	$g_p(x_c)$
Bai/dw8192	1622	9128	2.00420E+3	6.E-4
Bai/rw5151	7	16	2.57550E+3	4.E-8
Boeing/bcsstm39	16	34	2.55329E+5	1.E-5
Boeing/pcrystk03	6	14	6.45642E+7	4.E-10
Boeing/pct20stif	8	18	7.71187E+7	3.E-5
GHS_indef/spmsrtls	797	3886	2.56428E+5	7.E-5
GHS_psdef/copter1	6	14	1.57801E+6	4.E-7
GHS_psdef/ford1	8	18	3.00930E+5	6.E-10
GHS_psdef/jnlbrng1	1032	5194	2.10955E+2	2.E-5
GHS_psdef/opt1	6	14	1.34586E+8	2.E-7
Gset/G67	97	310	1.90595E+4	2.E-5
HB/bcspwr10	8	18	5.05190E+4	7.E-11
LPnetlib/lp_dfl001	216	826	1.61473E+4	9.E-7
LPnetlib/lp_osa_60	6	14	3.73599E+7	2.E-13
LPnetlib/lp_qap15	11	24	1.76400E+5	2.E-8
Mathworks/Muu	22	46	3.30168E-4	7.E-9
Meszaros/bas1lp	10	22	3.89051E+8	2.E-6
Meszaros/cq9	7296	40920	7.55190E+6	5.E-2
Meszaros/deter1	8386	44846	4.25901E+1	7.E-4
Meszaros/nl	6657	33610	1.37931E+5	2.E-2
Mittelmann/fome13	226	840	1.29178E+5	8.E-4
Mittelmann/nug08-3rd	11	24	3.43392E+5	3.E-5
Nasa/barth5	7	16	1.24284E+5	6.E-10
Nasa/shuttle_eddy	7	16	5.41577E+5	6.E-10
Nasa/skirt	9	20	1.78356E+6	8.E-7
Oberwolfach/rail_5177	2634	10958	2.40883E-5	9.E-11
Parsec/Na5	14840	74720	1.95586E+3	1.E-4
Parsec/SiH4	3113	14860	6.40972E+3	3.E-4
Rajat/rajat09	8	18	2.42823E+5	2.E-5
Schenk_IBMNA/c-42	8676	46710	1.79401E+8	6.E-2
Schenk_IBMNA/c-48	12526	81802	2.50827E+11	1.E-1
Schenk_IBMNA/c-61	3946	24530	2.42598E+8	2.E-2
Simon/appu	6	14	1.69528E+6	1.E-7
Tkk/g3rmt3m3	7	16	4.15968E+6	2.E-10
Tkk/tube1	6	14	1.90115E+7	3.E-12
VanHenkelum/cage11	7	16	1.97780E+4	9.E-7

TABLE 9.3
Results obtained with AS_CBB method on scaled problems

[4] A. Björck, *Numerical methods for least squares problems*, SIAM Philadelphia, 1996.

[5] D. Chen, R.J. Plemmons, *Nonnegativity constraints in numerical analysis*, Paper presented at the Symposium on the Birth of Numerical Analysis, Leuven Belgium, October 2007. To appear in the Conference Proceedings, to be published by World Scientific Press, A. Bultheel and R. Cools, Eds. (2009)

[6] T.F. Coleman, Y.Li, *An interior trust-region approach for nonlinear minimization subject to bounds*, SIAM Journal on Optimization 6, pp. 418-445, 1996.

[7] T.F. Coleman, Y. Li, *A reflective Newton method for minimizing a quadratic function subject to bounds on some of the variables*, SIAM Journal on Optimization, 6, pp. 1040-1058, 1996.

[8] A.R. Conn, N.I.M. Gould, Ph.L. Toint, *Trust-region methods*, SMPS/SIAM Series on Optimization, 2000.

[9] Y-H. Dai, W. W. Hager, K. Schittkowski, H. Zhang, *The cyclic Barzilai-Borwein method for unconstrained optimization*, IMA Journal on Numerical Analysis, 26, pp. 604-627, 2006.

[10] H.S. Dollar, *Iterative linear algebra for constrained optimization*. Oxford University Computing

elapsed time in seconds					
Group/Test name	Hybrid method	AS_CBB	Group/Test name	Hybrid method	AS_CBB
Bai/rw5151	0.32	0.05	Mathworks/Muu	4.65	0.10
Boeing/bcsstm39	1.16	0.22	Meszaros/bas1lp	0.42	0.11
Boeing/pcrystk03	1.22	0.21	Mittelmann/nug08-3rd	1.07	0.09
Boeing/pct20stif	1.22	0.46	Nasa/barth5	0.49	0.04
GHS_psdef/copter1	0.39	0.05	Nasa/shuttle_eddy	0.28	0.03
GHS_psdef/ford1	0.42	0.06	Nasa/skirt	0.40	0.06
GHS_psdef/opt1	1.05	0.21	Rajat/rajat09	0.54	0.07
Gset/G67	1.03	0.39	Simon/appu	1.12	0.25
HB/bcspwr10	0.11	0.02	Tkk/g3rmt3m3	0.19	0.04
LPnetlib/lp_dfl001	0.71	0.77	Tkk/tube1	0.69	0.12
LPnetlib/lp_osa_60	1.02	0.35	VanHenkelum/cage11	1.26	0.15
LPnetlib/lp_stocfor3	0.18	0.04			

TABLE 9.4
Elapsed time in seconds

elapsed time in seconds					
Group/Test name	Hybrid method	AS_CBB	Group/Test name	Hybrid method	AS_CBB
Bai/dw8192	7.02	7.38	Oberwolfach/rail_5177	6.93	6.97
GHS_indef/spmsrtls	6.39	15.27	Parsec/Na5	674.06	198.44
GHS_psdef/jnlbrng1	38.23	22.48	Parsec/SiH4	238.91	24.53
Meszaros/cq9	6.94	59.76	Schenk_IBMNA/c-42	58.41	68.27
Mesaros/deter1	5.19	29.64	Schenk_IBMNA/c-48	59.32	195.61
Meszaros/nl	6.36	30.90	Schenk_IBMNA/c-61	50.69	129.15
Mittelmann/fome13	6.69	7.03			

TABLE 9.5
Elapsed time in seconds

Laboratory 2005.

[11] H.S. Dollar, N.I.M. Gould, W.H.A. Schilders, A.J. Wathen, *Using constraint preconditioners with regularized saddle-point problems*, Computational Optimization and Applications 36, pp. 249-270, 2007.

[12] H.S. Dollar, N.I.M. Gould, W.H.A. Schilders, A.J. Wathen, *Implicit-Factorization Preconditioning and Iterative Solvers for Regularized Saddle-Point Systems*, SIAM Journal on Matrix Analysis and Applications, 28 pp. 170-189, 2006.

[13] M.P. Friedlander, *BCLS: a large-scale solver for bound-constrained least-squares*, http://www.cs.ubc.ca/ mpf/bcls/, 2007.

[14] M.P. Friedlander, K. Hatz, *Computing nonnegative tensor factorizations* Optimization Methods and Software 23, pp. 631-647, 2008.

[15] W.W. Hager, B.A. Mair, H. Zhang, *An Affine-scaling Interior-point CBB Method for Box-Constrained Optimization*, Mathematical Programming, DOI 10.1007/s10107-007-0199-0.

[16] W.W. Hager, H. Zhang, *A new active set algorithm for box constrained optimization*, SIAM J. Optim. 17, pp. 526-557, 2006.

[17] M. Heinkenschloss, M. Ulbrich, S. Ulbrich, *Superlinear and quadratic convergence of affine-scaling interior-point Newton methods for problems with simple bounds without strict complementarity assumptions*, Mathematical Programming 86, pp. 615-635, 1999.

[18] C.C. Paige, M.A. Saunders, *LSQR: an algorithm for sparse linear equations and sparse least squares*, ACM Transactions on Mathematical Software, pp. 43-71, 1982.

[19] T.A. Davis, *The University of Florida sparse matrix collection*. NA Digest, vol. 92, no. 42, October 16, 1994, NA Digest, vol. 96, no. 28, July 23, 1996, and NA Digest, vol. 97, no. 23, June 7, 1997, `http://www.cise.ufl.edu/research/sparse/matrices`.

Group/Test name	major-itns	minor-itns	Aprod	$q(x_c)$	$g_p(x_c)$
Bai/cryg10000	17	1475	4240(1219)	2.37768E+1	3E+2
Bai/dw8192	24	708	6279(4184)	2.00420E+3	2.E-3
Bai/rw5151	4	8	1531(1495)	2.57550E+3	0.E+0
Boeing/bcsstm39	6	6	519(12)	2.55329E+5	3.E-5
Boeing/pcrystk03	6	16	1323(120)	6.45642E+7	0.E+0
Boeing/pct20stif	8	29	17313(5329)	7.71187E+7	0.E+0
DRIVCAV/cavity26	69	11122	27954(5431)	3.46916E+3	1.E-4
GHS_indef/bratu3d	47	7182	17491(2322)	8.21465E+2	4.E-5
GHS_indef/sit100	57	4648	22441(12913)	4.22135E+2	6.E-5
GHS_indef/spmsrtls	13	234	26808(26285)	2.56428E+5	3.E-4
GHS_psdef/copter1	5	28	1781(1446)	1.57801E+6	0.E+0
GHS_psdef/ford1	5	31	6867(4980)	3.00930E+5	0.E+0
GHS_psdef/jnlbrng1	20	873	10797(5384)	2.10955E+2	2.E-5
GHS_psdef/opt1	6	36	1560(1)	1.34586E+8	0.E+0
Gset/G67	11	70	7640(6451)	1.90595E+4	3.E-4
HB/bcspwr10	4	10	5227(5083)	5.05190E+4	0.E+0
HB/sherman3	50	14586	32728(3351)	8.11138E+5	8.E-1
LPnetlib/lp_dfl001	9	89	5830(5605)	1.61473E+4	1.E-2
LPnetlib/lp_osa_60	2	3	2695(1264)	3.73599E+7	0.E+0
LPnetlib/lp_qap15	1	1	10(0)	1.76400E+5	2.E-10
LPnetlib/lp_ stocfor3	15	798	3668(1437)	3.58318E+8	3.E+1
Mathworks/Muu	6	22	4413(2129)	3.32036E-4	2.E-8
Mathworks/Pd	10	1538	4530(938)	2.37584E+4	7.E+0
Meszaros/bas1lp	6	34	5506(5415)	3.89051E+8	0.E+0
Meszaros/co5	54	18356	42974(5873)	1.47088E+7	2.E+0
Meszaros/cq9	23	995	11030(8819)	7.55190E+6	4.E-1
Meszaros/deter1	14	818	6444(4748)	4.25901E+1	1.E-4
Meszaros/nl	20	878	7963(5989)	1.37931E+5	2.E-2
Mittelmann/fome13	13	190	6085(0)	1.29178E+5	1.E-3
Mittelmann/nug08-3rd	2	4	21(0)	3.43392E+5	3.E-11
Nasa/barth5	4	10	9559(8857)	1.24284E+5	1.E-15
Nasa/shuttle_eddy	3	10	405(15)	5.41577E+5	0.E+0
Nasa/skirt	6	47	2882(814)	1.78356E+6	0.E+0
Oberwolfach/flowmeter0	426	84980	616410(444743)	1.38994E+2	2.E-1
Oberwolfach/rail_5177	6	3320	6667(0)	2.78271E-6	1.E-11
Parsec/Benzene	59	6583	19816(6411)	9.68629E+3	3.E-4
Parsec/Na5	30	2166	7334(2879)	1.95586E+3	3.E-4
Parsec/SiH4	17	708	2993(1443)	6.40972E+3	3.E-4
Rajat/rajat09	3	7	280(189)	2.42823E+5	2.E-8
Schenk_IBMNA/c-42	65	1818	15382(11457)	1.79401E+8	1.E+0
Schenk_IBMNA/c-48	29	1994	21739(17604)	2.50827E+11	2.E+1
Schenk_IBMNA/c-53	153	27335	86948(29013)	1.42336E+7	9.E+0
Schenk_IBMNA/c-61	26	1012	39563(37292)	2.42598E+8	2.E-1
Simon/appu	4	17	14053(14000)	1.69528E+6	0.E+0
Tkk/g3rmt3m3	5	24	163(2)	4.15968E+6	0.E+0
Tkk/t2d_q4	86	10185	35708(13889)	1.82855E+1	1.E-5
Tkk/tube1	3	13	153(0)	1.90115E+7	0.E+0
VanHenkelum/cage11	4	16	31607(27820)	1.97780E+4	0.E+0

TABLE 9.6
Results obtained with BCLS on scaled problems