Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Preconditioned conjugate gradient algorithm for large scale problems with box constraints

R. Pytlak
Faculty of Cybernetics
Military University of Technology
00-908 Warsaw, Poland
rpytlak@isi.wat.waw.pl

T. Tarnawski
Faculty of Cybernetics
Military University of Technology
00-908 Warsaw, Poland
tarni@isi.wat.waw.pl

***Abstract*— The paper describes a new conjugate gradient algorithm for large scale nonconvex problems with box constraints. In order to speed up the convergence the algorithm employs a scaling matrix which transforms the space of original variables into the space in which Hessian matrices of functionals describing the problems have more clustered eigenvalues. This is done efficiently by applying limited memory BFGS updating matrices. Once the scaling matrix is calculated, the next few iterations of the conjugate gradient algorithms are performed in the transformed space. The box constraints are treated efficiently by the projection. We believe that the preconditioned conjugate gradient algorithm is competitive to the L-BFGS-B algorithm. We give some numerical results which support our claim.**

I. INTRODUCTION

We consider the problem

$$\min_{x\in\mathcal{R}^n} f(x) \tag{1}$$

$$\text{subject to the simple bounds } l \le x \le u, \tag{2}$$

where we assume that l, u are fixed vectors and the inequalities are taken componentwise. In general, we assume that the function f is continuously differentiable, i.e., $f \in \mathrm{C}^1$ (however in some cases we will apply a stronger assumption that $f \in C^2$).

Assume that constraints (2) are not present. If second order derivatives of f are not available, or the evaluation of the Hessian matrix of f is not cheap, but gradients of f are available, then we can either use quasi–Newton or conjugate gradient algorithms to solve the problem (1). If the number of variables is large then the recommended quasi–Newton method is the limited memory BFGS described in [15] and [23].

We can also use the conjugate gradient algorithm. In [18] a new family of conjugate gradient algorithms was introduced based on methods proposed in [14] and [22]. Their direction finding subproblem is given by

$$d_k = -\mathbf{Nr}\{g_k, -\beta_k d_{k-1}\}, \tag{3}$$

where $g_k = g(x_k) = \nabla f(x_k)$ and $\mathbf{Nr}\{a,b\}$ is defined as the point from a line segment spanned by the vectors a and b which has the smallest norm, i.e.,

$$\| \mathbf{Nr}\{a,b\} \| = \min\{\| \lambda a + (1-\lambda)b \| : 0 \le \lambda \le 1\}, \tag{4}$$

and $\| \cdot \|$ is the Euclidean norm. Let us notice that the operation $\mathbf{Nr}\{\cdot,\cdot\}$ can be easily performed. This is a simple univariate quadratic problem with box constraints and can be solved analytically.

If $\beta_k = 1$ then we have the Lemaréchal-Wolfe algorithm and when

$$\beta_k = \frac{\|g_k\|^2}{|\langle g_k - g_{k-1}, g_k\rangle|} \tag{5}$$

directions generated by (3) are equivalent to those provided by the Polak–Ribiére formula (under the assumption that directional minimization is exact). (3) with (5) give the algorithm that has not only superior numerical properties but has also convergence properties better than that of all existing versions of the Polak–Ribiére algorithm (see, e.g., [12]).

Due to strong convergence properties exhibited by the conjugate gradient algorithm defined by (3) it is tempting to extend it by introducing its preconditioned version. The idea behind preconditioned conjugate gradient algorithm is to transform the decision vector by linear transformation D such that after the transformation the nonlinear problem is *easier* to solve–eigenvalues of Hessian matrices of the objective function of the new minimization are more clustered (see [17] for the discussion of how eigenvalues clustering influences the behaviour of conjugate gradient algorithms).

If $\hat{x}$ is transformed x:

$$\hat{x} = Dx \tag{6}$$

then our minimization problem will become

$$\min_{\hat{x}} \left[\hat{f}(\hat{x}) = f(D^{-1}\hat{x})\right] \tag{7}$$

and for this problem the search direction will be defined as follows

$$\hat{d}_k = -\mathbf{Nr}\{\nabla\hat{f}(\hat{x}_k), -\hat{\beta}_k\hat{d}_{k-1}\} \tag{8}$$

Since we want to avoid to minimize $\hat{f}$ with respect to $\hat{x}$ we need expressing the above search direction rule in terms of f and x. First of all, notice that

$$\nabla\hat{f}(\hat{x}) = D^{-T}\nabla f(x) \tag{9}$$

therefore we can write

$$\hat{d}_k = -\mathbf{Nr}\{D^{-T}\nabla f(D^{-1}\hat{x}_k), -\hat{\beta}_k\hat{d}_{k-1}\}. \tag{10}$$

0-7803-9568-9/05/$20.00 ©2005 IEEE

If we multiply both sides of (10) by D^{-1} we will get

$$d_k = -\lambda_k D^{-1}D^{-T}\nabla f(x_k) + (1-\lambda_k)\,\hat{\beta}_k d_{k-1}, \quad (11)$$

where $0 \le \lambda_k \le 1$ and either

$$\hat{\beta}_k = 1 \quad (12)$$

for the Fletcher-Reeves version, or

$$\begin{aligned}\hat{\beta}_k &= \frac{\|\hat{g}_k\|^2}{|\langle \hat{g}_k - \hat{g}_{k-1}, \hat{g}_k\rangle|} \\ &= \frac{g_k^T D^{-1}D^{-T} g_k}{|\left(g_k - g_{k-1}\right)^T D^{-1}D^{-T} g_k|}\end{aligned} \quad (13)$$

for the Polak-Ribiere version.

The equation (11) can be stated as

$$d_k = -\lambda_k H\nabla f(x_k) + (1-\lambda_k)\,\hat{\beta}_k d_{k-1}. \quad (14)$$

where $H = D^{-1}D^{-T}$. This suggests that D should be chosen in such a way that D^TD is an approximation to $\nabla^2_{xx} f(\bar{x})$ where $\bar{x}$ is a solution of problem (1).

Moreover, D should be such that systems of linear equations

$$D^T\hat{g}_k = g_k \quad (15)$$

$$Dd_k = \hat{d}_k \quad (16)$$

which we have to solve at every iteration are *easy* to solve.

II. A general convergence theory

Now consider the problem with box constraints (2). For the moment assume that the scaling matrix is an identity matrix. This will be helpful in explaining the way we handle the box constraints. To cope efficiently with these constraints the projection algorithm is applied.

Let the projection operator $P[\cdot]$ be defined as follows

$$(P[x])_i = \begin{cases} (l)_i & \text{if } (x)_i < (l)_i \\ (x)_i & \text{if } (l)_i \le (x)_i \le (u)_i \\ (u)_i & \text{if } (x)_i > (u)_i \end{cases}.$$

The projection algorithm was studied, among others, in [1], [2], [3], [8], [9], [10], [5], [6].

Before stating our algorithm we have to specify necessary optimality conditions which must be satisfied by any minimizer of problem (1)–(2).

Lemma 1: Let $x^\star$ be a feasible point. Then $x^\star$ is a critical point for problem (1)–(2) if and only if

$$P[x^\star - \alpha\nabla f(x^\star)] = x^\star, \quad (17)$$

for all $\alpha > 0$. Moreover condition (17) can be checked only for $\alpha = 1$.

Proof: The proof can be found in [1]. ∎

In order to simplify the presentation we will consider the problem of minimizing f subject to the constraints

$$x \ge 0. \quad (18)$$

As far as a line search of our algorithm is concerned we follow [19]. First, we notice that the function $f(P[x_k + \alpha d_k])$, of variable α, can be interpreted as a composition of two functions: the first one is Lipschitzian and the second one continuously differentiable. Let us consider a piecewise linear *arc* of the form

$$\begin{aligned} x_k(\alpha) &= x_k + d_k(\alpha), \quad \text{where} \\ (d_k(\alpha))_i &:= \begin{cases} \alpha(d_k)_i & \text{if } \alpha \le \alpha_k^i \\ \alpha_k^i(d_k)_i & \text{if } \alpha > \alpha_k^i \end{cases} \end{aligned} \quad (19)$$

where the breakpoints $\{\alpha_k^i\}_1^n$ are calculated as follows

$$\alpha_k^i := -\frac{(x_k)_i}{(d_k)_i}, \quad i = 1, \ldots, n. \quad (20)$$

Throughout the paper we apply the following convention

$$(g_k(\alpha))_i := \begin{cases} \nabla_{x_i} f(x_k + d_k(\alpha)) & \text{if } \alpha < \alpha_k^i \\ 0 & \text{if } \alpha \ge \alpha_k^i \end{cases}, \quad (21)$$

$i = 1, \ldots, n$.

Having in mind that we consider simpler constraints (18) we define the set of indices I_k^+

$$I_k^+ := \{i \in \overline{1,n} : \ (x_k)_i \le \varepsilon_k \text{ and } \nabla_{x_i} f(x_k) > 0\}, \quad (22)$$

where $\{\varepsilon_k\}$ is such that $\varepsilon_k > 0$ and

$$\lim_{k\in K} \|x_k - P[x_k - \nabla f(x_k)]\| = 0 \quad \Leftrightarrow \quad \lim_{k\in K} \varepsilon_k = 0. \quad (23)$$

for any subsequence $\{x_k\}_{k\in K}$.

Let us notice that if $\{\varepsilon_k\}$ is calculated in the following way

$$w_k = x_k - P[x_k - M\nabla f(x_k)], \quad \varepsilon_k = \min(\varepsilon, \|w_k\|), \quad (24)$$

$\varepsilon > 0$, where M is a diagonal positive definite matrix, then condition (23) will be satisfied.

The sets I_k^+ are used to modify the direction finding subproblem. Instead of solving problem (3) we find a new direction according to the rule

$$d_k = -\mathbf{Nr}\{\nabla f(x_k), -\beta_k d_{k-1}^+\}. \quad (25)$$

Here d_{k-1}^+ is defined by

$$(d_{k-1}^+)_i := \begin{cases} (d_{k-1})_i & \text{if } i \notin I_k^+ \\ - \nabla_{x_i} f(x_k)/\beta_k & \text{if } i \in I_k^+ \end{cases}. \quad (26)$$

Let us notice that according to (25) and (26) we have

$$(d_k)_i = -\nabla_{x_i} f(x_k) \quad \text{for } i \in I_k^+. \quad (27)$$

Next, we introduce, for a given α, the set of all indices such that $i \notin I_k^+$, or $i \in I_k^+$ and $\alpha_k^i > \alpha$:

$$\begin{aligned} I_k(\alpha) := \ & \{i \in \overline{1,n} : \ i \notin I_k^+\} \bigcup \\ & \{i \in \overline{1,n} : \ i \in I_k^+ \text{ and } \alpha_k^i > \alpha\}. \end{aligned} \quad (28)$$

Having defined $I_k(\alpha)$ we can partition d_k into two vectors in such a way that one of them, $d_k^-(\alpha)$, is as follows

$$d_k^-(\alpha) := \{(d_k)_i\}_{i\in I_k(\alpha)}.$$

Eventually we can state our directional minimization rule which we borrow from [19]: find a positive number α_k such that

$$f(P[x_k+\alpha_k d_k]) - f(x_k) \leq$$
$$-\mu\langle d_k, P[x_k+\alpha_k d_k] - x_k\rangle, \quad (29)$$
$$\langle g_k(\alpha_k), d_k\rangle \geq -\eta\|d_k^-(\alpha_k)\|^2,$$
$$0<\mu<\eta<1. \quad (30)$$

According to Bertsekas ([2]) the scaling matrix should be applied only to variables with indices corresponding to the set I_k^+. Therefore, we choose the scaling matrix as

$$D_k = \begin{bmatrix} D_k^+ & 0 \\ 0 & I_{n_k} \end{bmatrix}.$$

Here, D_k^+ is a nonsingular matrix of dimension $n_k^+ = n - n_k$ and I_{n_k} is the identity matrix of dimension $n_k = |I_k^+|$.

We use the matrix D_k^+ to transform x variables to $\hat{x}$ variables resulting in equations:

$$D_k^T \hat{g}_k = g_k \quad (31)$$
$$D_k d_k = \hat{d}_k \quad (32)$$

Notice that, as in (27), we have

$$(\hat{d}_k)_i = (d_k)_i, \quad (33)$$
$$\nabla_{x_i}\hat{f}(\hat{x}_k) = \nabla_{x_i} f(x_k), \quad i \in I_k^+. \quad (34)$$

and $\hat{d}_k$ is defined by

$$\hat{d}_k = -\mathbf{Nr}\{\hat{g}_k, -\hat{\beta}_k \hat{d}_{k-1}^+\}. \quad (35)$$

Our general algorithm is as follows:

Algorithm Parameters: $\mu,\ \eta \in (0,1),\ \eta > \mu,\ \epsilon > 0,\ \{\hat{\beta}_k\}_0^\infty, \varepsilon > 0, \{D_k\}_1^\infty,\ D_k \in R^{n\times n}$, diagonal positive definite matrix M.

Data: x_0

1) Set k=0
2) Compute:

$$d_k = -g_k$$
$$\hat{d}_k = D_k d_k$$

If $\|d_k\| = 0$ then STOP, otherwise go to Step 4).

3) Compute:

$$D_k^T \hat{g}_k = g_k \quad (36)$$
$$w_k = x_k - P[x_k - M\nabla f(x_k)]. \quad (37)$$
$$\varepsilon_k = \min(\varepsilon, \|w_k\|), \quad (38)$$
$$\hat{d}_k = -\mathbf{Nr}\{\hat{g}_k, -\hat{\beta}_k \hat{d}_{k-1}^+\} \quad (39)$$
$$D_k d_k = \hat{d}_k \quad (40)$$

If $w_k = 0$ then STOP.

4) Find α_k according to (29)–(30).
5) Substitute $P[x_k+\alpha_k d_k]$ for x_{k+1}, increase k by one, go to Step 3).

We can prove the lemma

Lemma 2: Assume that x_k is a noncritical point, $D_k^T D_k$ is positive definite and $d_k \neq 0$ is calculated in Step 2 of *Algorithm*. Then there exists a positive α_k such that (29)–(30) are satisfied, or

$$\lim_{\alpha\to\infty} f(P[x_k+\alpha d_k]) = -\infty. \quad (41)$$

To investigate the convergence of *Algorithm* we begin by providing a crucial lemma which requires the following assumptions.

Assumption 1: There exists $L < \infty$ such that

$$\|\nabla f(y) - \nabla f(x)\| \leq L\|y - x\|$$

for all $x,\ y$ from a bounded set.

Assumption 2: There exist $d_l,\ d_u$ such that $0 < d_l < d_u < +\infty$ and

$$d_l\|x\|^2 \leq x^T D_k^T D_k x \leq d_u\|x\|^2 \quad (42)$$

for all $x \in \mathcal{R}^n$ and k.

Lemma 3: If *Assumptions 1–2* hold, the direction d_k is determined by (36)–(40) and the step–size coefficient α_k is calculated according to (29)–(30), then for any bounded subsequence $\{x_k\}_{k\in K}$

$$\lim_{k\in K}\|x_k - P[x_k + d_k]\| = 0. \quad (43)$$

Proof: The proof of the lemma, as the proofs of other results presented in the paper, is given in [21]. ■

For the convenience of future notations we assume that variables $(x)_i$ have been reordered in such a way that d_k can be partitioned into two vectors (d_k^1, d_k^2) where the first vector d_k^1 is represented by

$$d_k^1 := \{(d_k)_i\}_{i\notin I_k^+}.$$

The same convention applies to other vectors, for example

$$\nabla f(x_k) = ((\nabla f(x_k))^1, (\nabla f(x_k))^2)$$
$$= (\nabla^1 f(x_k), \nabla^2 f(x_k)),$$
$$d_{k-1}^+ = ((d_{k-1}^+)^1, (d_{k-1}^+)^2)$$
$$= (d_{k-1}^1, (d_{k-1}^+)^2) \quad (44)$$

and to vectors such as $\hat{x}_k = (\hat{x}_k^1, \hat{x_k}^2)$ since we have $x_k^2 = \hat{x}_k^2$. It is important to remember that this **partitioning is always related to** I_k^+ and not to I_{k-1}^+ as the second part of (44) would suggest. This partitioning simplifies statements and proofs of several convergence results presented later in the paper. It also enables the adaptation of the convergence analysis stated in [18] to the new algorithm.

The condition (43) stated in *Lemma 3* unfortunately is not equivalent to the condition

$$\lim_{k\in K}\|x_k - P[x_k - \nabla f(x_k)]\| = 0$$

which we have to prove. This is due to the additional vector $\hat{\beta}_k \hat{d}_{k-1}^+$ in the formula (39).

Theorem 4: Suppose that *Assumptions 1–2* are satisfied. Moreover, assume that for any convergent subsequence $\{x_k\}_{k\in K}$ whose limit is not a critical point

1) $\{\hat{\beta}_k\}$ is such that

$$\liminf_{k\to\infty}\left(\hat{\beta}_k\|d^1_{k-1}\|\right) \geq \nu_1 \liminf_{k\to\infty}\|\nabla^1 f(x_k)\| \quad (45)$$

where ν_1 is some positive constant,

2) there exists a number ν_2 such that $\nu_2\|D_k^{-T}\|_2\|D_{k-1}\|_2 \in (0,1)$ and

$$\langle\nabla^1 f(x_k), d^1_{k-1}\rangle \leq \nu_2\|\nabla^1 f(x_k)\|\|d^1_{k-1}\|, \quad (46)$$

whenever $\lambda_k \in (0,1)$.

Then $\lim_{k\to\infty} f(x_k) = -\infty$, or every accumulation point of the sequence $\{x_k\}_0^\infty$ generated by *Algorithm* is a critical point.

The condition (46) is independent of the choice of the sequence $\{\hat{\beta}_k\}$ and is related to the directional minimization. Therefore it is not surprising that in some important situation it can be substituted by another, more easily verifiable, condition.

Lemma 5: Suppose that *Assumptions 1–2* are satisfied. Assume that $\{x_k\}$ is generated by *Algorithm* and $\{\hat{\beta}_k\}$ satisfies (45) for any convergent subsequence whose limit is not a critical point. Then either

$$\lim_{k\to\infty} f(x_k) = -\infty, \quad (47)$$

or for every convergent subsequence $\{x_k\}_{k\in K}$ such that

$$\lim_{k\in K}\|x_{k+1} - x_k\| = 0 \quad (48)$$

we have

$$\lim_{k\in K}\|x_k - P[x_k - \nabla f(x_k)]\| = 0. \quad (49)$$

III. The globally convergent Polak–Ribiere algorithm.

In this section we examine *Algorithm* with the sequence $\{\hat{\beta}_k\}$ defined by

$$\hat{\beta}_k = \|\nabla^1\hat{f}((\hat{x}^1_k,\hat{x}^2_{k-1}))\|^2 / |\langle\nabla^1\hat{f}((\hat{x}^1_k,\hat{x}^2_{k-1})) - \nabla^1\hat{f}(\hat{x}_{k-1}), \nabla^1\hat{f}((\hat{x}^1_k,\hat{x}^2_{k-1}))\rangle|, \quad (50)$$

where, as usual, by $\hat{x}^1$ we mean a vector defined by indices not belonging to I_k^+ and the vector $\nabla^1\hat{f}((\hat{x}^1_k,\hat{x}^2_{k-1}))$ is as follows

$$\nabla^1\hat{f}((\hat{x}^1_k,\hat{x}^2_{k-1})) = \{\nabla_{x_i}\hat{f}((\hat{x}^1_k,\hat{x}^2_{k-1}))\}_{i\notin I_k^+}.$$

In [18] we proved that $\{\hat{\beta}_k\}$ constructed in this way guarantees that directions generated by *Algorithm* are equivalent to those generated by the Polak–Ribiére algorithm, if directional minimization is exact and I_k^+ are empty for all k.

We can prove the theorem.

Theorem 6: If *Assumptions 1–2* are satisfied, then *Algorithm* gives

$$\lim_{k\to\infty} f(x_k) = -\infty,$$

or for any convergent subsequence $\{x_k\}_{k\in K}$

$$\lim_{k\in K}\|x_k - P[x_k - \nabla f(x_k)]\| = 0 \quad (51)$$

provided that:

1) $\hat{\beta}_k$ is given by (50),
2) there exists $S < \infty$ such that $\alpha_k \leq S\ \forall k$.

The next result applies to problems which satisfy sufficient optimality conditions together with the strict complementarity condition. We show the important property that *Algorithm* determines the set of bounds that are active at the solution in a finite number of iterations.

Assumption 3: Let $x^\star$ be a point which satisfies necessary optimality conditions for problem (1), (18). Assume also that f is twice continuously differentiable and there exist $m_2 > m_1 > 0$ such that

$$m_1\|z\|^2 \leq z^T\nabla^2_{xx}f(x^\star)z \leq m_2\|z\|^2 \quad (52)$$

for all z satisfying

$$z \neq 0: \quad (z)_i = 0 \quad \forall i \in A(x^\star), \quad (53)$$

where

$$A(x) := \{i \in \overline{1,n}: \quad (x)_i = 0\}. \quad (54)$$

Moreover

$$\nabla_{x_i} f(x^\star) > 0 \ \ \forall i \in A(x^\star). \quad (55)$$

We can prove the following theorem.

Theorem 7: Let $\bar{x}$ be a point at which Assumption 3 is satisfied. Assume that $\{x_k\}$ is generated by *Algorithm*, the assumptions of *Theorem 6* hold and ε_k is calculated according to (24). Then there exists $\delta > 0$ such that if

$$\|x_{\bar{k}} - \bar{x}\| \leq \delta$$

for some $\bar{k}$, then $\{x_k\}$ converges to $\bar{x}$ and

$$A(\bar{x}) = A(x_k) = I_k^+ \ \ \forall k \geq \bar{k}+1.$$

IV. Scaling matrices

In the previous section we showed that for a given nonsingular matrix $H^{-1} = D^TD$ the preconditioned conjugate gradient algorithm is globally convergent. The use of constant scaling matrix is likely to be inefficient since the function f we minimize is nonlinear. Therefore, we are looking at the sequence of matrices $\{H_k\}$ such that each H_k^{-1} is as close as possible to the Hessian $\nabla^2_{xx}f(x_k)$ and can be easily factorized as $D_k^{-1}D_k^{-T}$ where D_k is a nonsingular matrix.

In the paper we present the method of obtaining the scaling matrix that is based on the compact representation of the L-BFGS matrix proposed in [7] (see also [23]). In that method on kth iteration the Hessian matrix is approximated with B_k:

$$B_k = B_0 - [B_0S_k\ Y_k]\begin{bmatrix} S_k^TB_0S_k & L_k \\ L_k^T & -G_k\end{bmatrix}^{-1} \times \times \begin{bmatrix} S_k^TB_0 \\ Y_k^T\end{bmatrix} \quad (56)$$

where S_k and Y_k are $n \times m$ matrices (m being a constant parameter that controls the amount of memory used by the algorithm) defined by

$$S_k = [s_{k-m-1}, \ldots, s_{k-1}],\ Y_k = [y_{k-m-1}, \ldots, y_{k-1}] \quad (57)$$

with $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$, while L_k and G_k are the $m \times m$ matrices

$$\begin{aligned} (L_k)_{i,j} &= \begin{cases} s_{i-1}^T y_{i-1} & \text{if } i > j \\ 0 & \text{otherwise} \end{cases} \\ G_k &= \text{diag}\left[s_{k-m-1}^T y_{k-m-1}, \ldots, s_{k-1}^T y_{k-1}\right] \end{aligned} \quad (58)$$

and where B_k^0 is usually a diagonal matrix. A method for choosing B_k^0 that has proved to be effective in practice is to set $B_k^0 = \gamma_k I$ with

$$\gamma_k = \frac{y_{k-1}^T y_{k-1}}{s_{k-1}^T s_{k-1}}. \quad (59)$$

If we assume then that $B_0 = \gamma_k I$ and introduce matrices $M_k = [\gamma_k S_k\ Y_k]$ and

$$W_k = \begin{bmatrix} \gamma_k S_k^T S_k & L_k \\ L_k^T & -G_k \end{bmatrix}^{-1}$$

then (56) can be written as $B_k = \gamma_k I - M_k W_k M_k^T$.

Now we have to recall, that the scaling matrix will be applied only to the variables with indices not in the set I_k^+. Therefore, instead of full scaling matrix D_k (so that $B_k = D_k^T D_k$) it will be needed to evaluate the "reduced" scaling matrix D_k^+ corresponding to B_k^+, with B_k^+ defined analogically to 56, but in terms of S_k^+, Y_k^+, L_k^+ and G_k^+. The matrices S_k^+, Y_k^+ are submatrices of S_k, Y_k with rows with indices not belonging to I_k^+. In turn L_k^+ and G_k^+ are equivalent to L_k and G_k, but defined in terms of s^1 and y^1. Having said all that we write $B_k^+ = \gamma_k^+ I - M_k^+ W_k^+ M_k^{+T}$.

From here the transformation of the matrix B_k^+ to the form $D_k^{+T} D_k^+$ will proceed along the lines presented in [20]. First we do the QR factorization of the matrix M_k^+: $M_k^{+T} = Q_k R_k$, where Q_k is $n_k^+ \times n_k^+$ orthogonal matrix and R_k the $n \times m$ matrix which has zero elements except the elements constituting the upper $m \times m$ submatrix([13]). Taking into account that $Q_k^T Q_k = I$ we can represent B_k^+ as

$$B_k^+ = Q_k^T \left[\gamma_k I - R_k^T W_k^+ R_k\right] Q_k. \quad (60)$$

Notice that the matrix $R_k^T W_k^+ R_k$ has zero elements except those lying in the upper left $m \times m$ submatrix. We denote this submatrix by T_k and we can easily show that it is a positive definite matrix. If we compute the Cholesky decomposition of the matrix $\gamma_k I_k - T_k$, $\gamma_k I_k - T_k = C_k^T C_k$ then eventually we come to the relation $B_k^+ = Q_k^T F_k^T F_k Q_k$ with

$$F_k = \begin{bmatrix} C_k & 0 \\ 0 & \sqrt{\gamma_k} I_{n_k^+ - k} \end{bmatrix}. \quad (61)$$

The desired decomposition of the matrix B_k^+ is thus given by $B_k^+ = D_k^{+T} D_k^+$, $D_k^+ = F_k Q_k$, where the matrix D_k^+ is nonsingular provided that $s_i^{1T} y_i^1 > 0$ for $i = 0, \ldots, m-1$. Notice that the matrix Q_k does not have to be stored since it can be easily evaluated from the Householder vectors which have been used in the QR factorization. These vectors can be stored in zero elements of the R_k matrix ([13]).

Recall the relations (15)–(16) which now can be written as

$$\begin{aligned} Q_k^T F_k^T \hat{g}_k^1 &= g_k^1 \quad (62) \\ F_k Q_k d_k^1 &= \hat{d}_k^1. \quad (63) \end{aligned}$$

Solving these equations requires the number of floating point operations proportional to n_k^+ (see, e.g., [20]).

V. Numerical Experiments

In order to verify the effectiveness of the proposed algorithm it has been tested on problems from the CUTE collection ([4]). The aim was to try it on problems as large as possible, therefore choosing the problems with the `select` tool was done with the criteria, that the problem's dimension had to exceed 1000 or the number of variables had to be modifiable. The names and dimensions of bound constrained problems used in the comparison, are listed below in Table I.

Problem	Dimension
S368, SCOND1LS, SINEALI, SPECAN	8 – 12
CHEBYQAD, HS110	50
CVXBQP1, HARKERP2	100
EXPLIN, EXPLIN2, EXPQUAD, QRTQUAD	120
QR3DLS	155
GRIDGENA	170
CHARDIS0	400
PROBPENL	500
BIGGSB1, CHENHARK, PENTDI,	1000
LINVERSE	1999
BDEXP	5000
MINSURFO	5306
JNLBRNG1, JNLBRNG2, JNLBRNGA, JNLBRNGB	5625
MCCORMCK, NCVXBQP1, NCVXBQP2, NCVXBQP3, NOBNDTOR, NONSCOMP, OBSTCLAE, OBSTCLAL, OBSTCLBL, OBSTCLBM, OBSTCLBU, TORSION1, TORSION2, TORSION3, TORSION4, TORSION5, TORSION6, TORSIONA, TORSIONB, TORSIONC, TORSIOND, TORSIONE, TORSIONF	10000

TABLE I
Dimensions of bound-constrained problems used for testing the algorithm.

The implemented code (*CG Algorithm*) was compared against the benchmark for bound-constrained large scale optimization: the L-BFGS-B routine presented in [23]. Both programs were compiled on Intel PC machine under Linux operating system. For ensuring fair comparison the common parameters in both algorithms and the stopping criterion were unified. Both algorithms were to terminate on: $\|\nabla^1 f(x)\|_\infty / \max(1, \|x^1\|) \leq 10^{-7}$.

The parameters for the directional minimization rule (29)-(30) were set as: $\mu = 10^{-3}$, $\eta = 0.9$. The line search

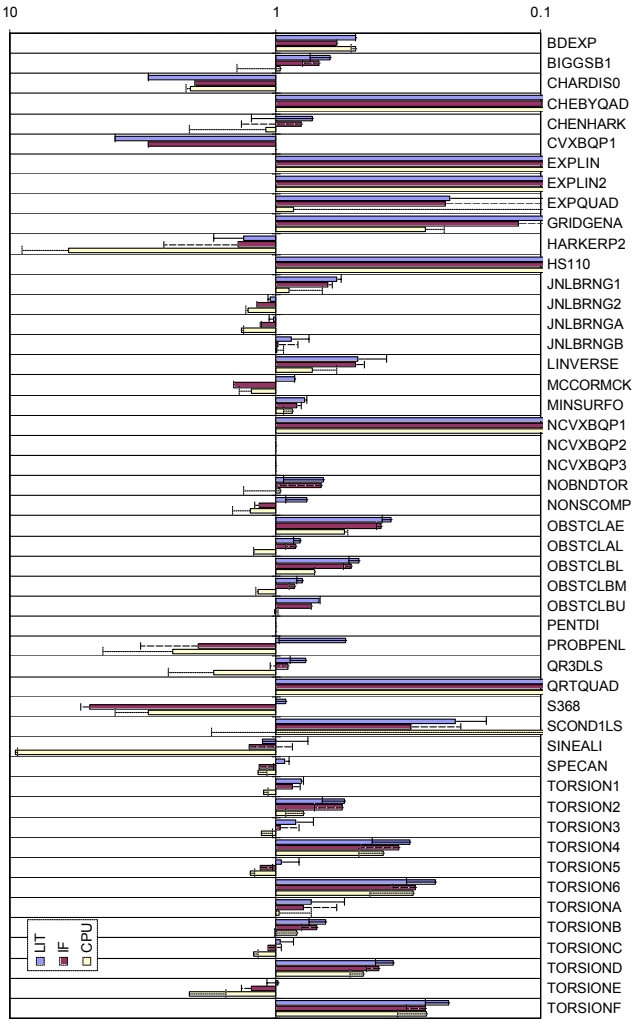

Fig. 1. Performance comparison of the proposed *CG Algorithm* against the L-BFGS-B code for bound-constrained problems (shown in table I). The bars correspond to algorithms performance with m set to 5, while the additional lines show analogous comparison for $m = 3$.

routine implemented in the code was augmented with the steplength selection algorithm of Morè & Thuente described in [16].

The results of the comparison are presented graphically. On the figure a set of three bars is drawn for each problem. The bars on a logarithmic scale show the ratio of the number of iterations (LIT), number of function+gradient evaluations (IF) and computing time (CPU) needed by the *CG Algorithm* divided by those from the executions of the L-BFGS-B code. Therefore values above one testify in favor of the L-BFGS-B and below one – in favor of the *CG Algorithm*. For some problems one or the other algorithm has failed and it is indicated by all tree bars drawn past the maximum (minimum) value.

References

[1] D.P. Bertsekas, *On the Goldstein–Levitin–Polyak gradient projection method*, IEEE Trans. Automat. Control 21 (1974), pp. 174–184.

[2] D.P. Bertsekas, *Projected Newton methods for optimization problems with simple constraints*, SIAM J. Control and Optimization 20 (1982), pp. 221–245.

[3] D.P. Bertsekas and E.M. Gafni, *Projected Newton methods and optimization of multicommodity flows*, IEEE Trans. Automat. Control 28 (1983), pp. 1090–1096.

[4] I. Bongartz, A.R. Conn, N.I.M. Gould and Ph.L. Toint, *CUTE: Constrained and Unconstrained Testing Environment*, Research Report RC 18860, IBM T.J. Watson Research Center, Yorktown, USA, 1994.

[5] J.V. Burke and J.J. More, *On the identification of active constraints*, SIAM J. Numer. Anal., 25 (1988), pp. 1197–1211.

[6] J.V. Burke and J.J. More, *Exposing constraints*, SIAM J. Optimization 4 (1994), pp. 573–595.

[7] R.H. Byrd, J. Nocedal and R.B. Schnabel, *Representations of quasi–Newton matrices and their use in the limited memory methods*, Technical Report NAM–03, Northwestern University, 1996.

[8] A.R. Conn, N.I.M. Gould and Ph.L. Toint, *Global convergence of a class of trust region algorithms for optimization with simple bounds*, SIAM J. Numerical Analysis 28 (1988), pp. 433–460.

[9] A.R. Conn, N.I.M. Gould and Ph.L. Toint, *Testing a class of methods for solving minimization problems with simple bounds on the variables*, Mathematics of Computation 50 (1988), pp. 399–430.

[10] J.C. Dunn, *Gradient projection methods for systems optimization problems*, Control and Dynamic Systems 29 (1988), pp. 135–195.

[11] R. Fletcher, *Practical Methods of Optimization*, J. Wiley: Chichester, 1987.

[12] J.Ch. Gilbert and J. Nocedal, *Global convergence properties of conjugate gradient methods for optimization*, SIAM J. Optimization 2 (1992), pp. 21–42.

[13] G.H. Golub and Ch.F. Van Loan, *Matrix computations*, The Johns Hopkins University Press, Baltimore, 1996.

[14] C. Lemaréchal, *An extension of Davidon methods to nondifferentiable Problem*, in Mathematical Programming Study 3 , M.L. Balinski and P. Wolfe eds., North–Holland: Amsterdam, 1975, pp. 95–109.

[15] D.C. Liua and J. Nocedal, *On the limited memory BFGS method for large scale optimization problems*, Mathematical Programming 45 (1989), pp. 03–528.

[16] J.J. Morè and D.J. Thuente, *Line Search Algorithms with Guaranteed Sufficient Decrease*, ACM Transactions on Mathematical Software, Vol. 20, No. 3, (1994) pp. 286–307.

[17] J. Nocedal and S.J. Wright, *Numerical optimization*, Springer–Verlag, New York, 1999.

[18] R. Pytlak, *On the convergence of conjugate gradient algorithms*, IMA J. Numerical Analysis 14 (1994), pp. 443–460.

[19] R. Pytlak, *An efficient Algorithm for Large Scale Problems with Simple Bound on the Variables*, SIAM J. on Optimization, Vol. 8, (1998) pp. 632–560.

[20] R. Pytlak and T. Tarnawski, *Preconditioned Conjugate Gradient Algorithms for Nonconvex Problems*, Research Report RR/ISI/WCY/WAT/01/2004, Military University of Technology, Warsaw, also 43rd IEEE CDC, December, Bahamas (2004) pp. 3191-3196.

[21] R. Pytlak and T. Tarnawski, *Preconditioned Conjugate Gradient Algorithms for Large Scale Problems with Box Constraints*, Research Report RR/ISI/WCY/WAT/01/2005, Military University of Technology, Warsaw, 2005.

[22] P. Wolfe, *A method of conjugate subgradients for minimizing nondifferentiable functions*, in Mathematical Programming Study 3, M.L. Balinski, P. Wolfe, eds., North–Holland: Amsterdam, 1975, pp. 145–173.

[23] C. Zhu, R.H. Byrd, P. Lu and J. Nocedal, *L–BFGS–B — FORTRAN Subroutines for Large–Scale Bound Constrained Optimization*, Research Report, Northwestern University, Department of Electrical Engineering and Computer Science, 1994.