Yokohama Publishers
ISSN 1338-9151 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 2, No. 1, pp. 81–104, January 2006

PRECONDITIONED CONJUGATE GRADIENT ALGORITHMS FOR NONCONVEX PROBLEMS*

R. Pytlak and T. Tarnawski

Abstract: The paper describes a new conjugate gradient algorithms for large scale nonconvex problems. In order to speed up the convergence the algorithms employ 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. We believe that the preconditioned conjugate gradient algorithms give more flexibility in achieving balance between the computing time and the number of function evaluations in comparison to a limited memory BFGS algorithm. We give some numerical results which support our claim.

Key words: *preconditioned conjugate gradient, large scale problems, unconstrained optimization*

Mathematics Subject Classification: *90C06, 90C26, 65K05*

1 Introduction

In this paper we consider algorithms for the unconstrained minimization problem:

$$\min_{x \in \mathcal{R}^n} f(x). \tag{1.1}$$

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 $f \in C^2$).

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 (1.1). If the number of variables is large then the recommended quasi–Newton method is the limited memory BFGS described in [18] and [30].

We can also use the conjugate gradient algorithm. The direction at the kth step of this algorithm is determined according to the rule:

$$d_k = -g_k + t_k d_{k-1} \tag{1.2}$$

*Some parts of the paper were presented at 43rd IEEE Conference on Decision and Control, December 7–14, The Bahamas, 2004.

where $g_k = g(x_k) = \nabla f(x_k)$ and, e.g.

$$t_k = \frac{\langle g_k - g_{k-1}, g_k \rangle}{\|g_{k-1}\|^2}, \tag{1.3}$$

$$t_k = \frac{\|g_k\|^2}{\|g_{k-1}\|^2}, \tag{1.4}$$

$$t_k = \frac{\langle g_k - g_{k-1}, g_k \rangle}{\langle g_k - g_{k-1}, d_{k-1} \rangle}, \tag{1.5}$$

$$t_k = \frac{\|g_k\|^2}{\langle g_k - g_{k-1}, d_{k-1} \rangle} \tag{1.6}$$

see e.g. [2],[8],[12],[22]; more complicated formulae are also possible ([27],[28]). The first formula in (1.3) is usually called the Polak–Ribiére formula while the second one is the Fletcher-Reeves formula. Hestenes–Stiefel formula (1.5) leads to algorithms as efficient as those supported by Polak–Ribiére formula. Dai–Yuan (1.6)([11]) formula guarantees global convergence of a conjugate gradient algorithm under standard Wolfe conditions (see discussion below). Other formulae are also possible ([11]) although the Polak–Ribiére version is often recommended due to its superior numerical properties emphasized in [23].

The global convergence of the Fletcher–Reeves version was first established in [1] under the assumption that the strong Wolfe conditions are applied in directional minimization. In [11] global convergence of the conjugate gradient algorithm, which is the Fletcher–Reeves method if directional minimization is exact, is established requiring only the Wolfe conditions.

The first globally convergent version of the Polak–Ribiére algorithm is given in [27]. Shanno's method is in fact memoryless quasi–Newton algorithm which is equivalent to the Polak–Ribiére conjugate gradient algorithm if directional minimization is exact. Furthermore, his convergence result is valid under the assumption $\|x_{k+1} - x_k\| \to 0$ where $\{x_k\}$ is the sequence generated by his method. Bridging conjugate gradient and quasi–Newton concepts was further advanced in [4] where variable storage quasi–Newton method was introduced. The algorithm of Buckley–LeNir uses quasi–Newton steps to evaluate a scaling matrix for conjugate gradient iterations which are performed afterwards. The Buckley–LeNir algorithm is globally convergent if the function f is strongly convex. Global convergence of the Polak–Ribiére version of a conjugate gradient algorithm is also analyzed in [13] where several versions of the Polak–Ribiére algorithm are considered along the lines of the analysis initiated by Al–Baali in the context of the Fletcher–Reeves method. Gilbert and Nocedal propose a method which switches between the Polak–Ribiére and the Fletcher–Reeves algorithms in such a way that the coefficient t_k in their method is always bounded by t_k of the Fletcher–Reeves algorithm. Their method is free from the drawback of the Fletcher–Reeves method as described in [23] (if $g_k \approx g_{k-1}$ and $g_{k-1}^T d_{k-1} \approx 0$ then $g_k^T d_k \approx 0$) and at the same time is globally convergent under the strong Wolfe directional minimization rules.

In [17] and [29] a new conjugate gradient method was introduced. Their direction finding subproblem is given by

$$d_k = -\mathbf{Nr}\{g_k, -d_{k-1}\}, \tag{1.7}$$

where $\mathbf{Nr}\{a, b\}$ is defined as the point from a line segment spanned by the vectors a and b which has the smallest norm, i.e.,

$$\| \mathbf{Nr}\{a, b\} \| = \min\{\| \lambda a + (1-\lambda) b \| : 0 \le \lambda \le 1\}, \tag{1.8}$$

and $\|\cdot\|$ is the Euclidean norm. The algorithm proposed in [17] and [29] is in fact the extension of some version of a conjugate gradient algorithm for quadratic function discussed in [15]. Hestenes called this version the method of shortest residuals. 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.

Consider the problem

$$\min_{(\mu,d)\in\mathcal{R}^{n+1}}\{\mu+1/2\|d\|^2\}$$

$$\begin{aligned}\text{s. t. } \langle g_k,d\rangle &\le \mu,\\ -\langle d_{k-1},d\rangle &\le \mu.\end{aligned}\tag{1.9}$$

We obtain the solution of this problem by solving its dual

$$\begin{aligned}\min_{0\le\lambda\le1}\quad &\frac{1}{2}\|\lambda g_k-(1-\lambda)d_{k-1}\|^2\\ =\ &\frac{1}{2}\|\mathbf{Nr}\{g_k,-d_{k-1}\}\|^2\\ =\ &\frac{1}{2}\|\lambda_k g_k-(1-\lambda_k)d_{k-1}\|^2.\end{aligned}$$

Moreover, the optimal value μ_k is

$$\mu_k=-\|d_k\|^2.$$

From this we can easily deduce the following properties

$$\langle g_k,d_k\rangle \le -\|d_k\|^2,\tag{1.10}$$

$$-\langle d_{k-1},d_k\rangle \le -\|d_k\|^2,\tag{1.11}$$

and

$$\langle g_k,d_k\rangle=-\|d_k\|^2,\ \langle d_{k-1},d_k\rangle=\|d_k\|^2,\tag{1.12}$$

if $0<\lambda_k<1$, because $(\lambda_k,\ 1-\lambda_k)$ are the Lagrange multipliers for problem (1.9). From (1.10) we have that if $\|d_k\|\ne0$, d_k is a direction of descent.

In [25] a new family of conjugate gradient algorithms was introduced. The important difference between these methods and the Lemaréchal–Wolfe algorithm lies in a new direction finding subproblem

$$d_k=-\mathbf{Nr}\{g_k,-\beta_k d_{k-1}\}.\tag{1.13}$$

Notice that if $\beta_k=1$ then we have the Wolfe–Lemaréchal algorithm. In [25] it was shown that the Lemaréchal–Wolfe algorithm is in fact the Fletcher–Reeves algorithm when directional minimization is exact. Moreover, the sequence $\{\beta_k\}$ was constructed in such a way that directions generated by (1.13) are equivalent to those provided by the Polak–Ribiére formula (under the assumption that directional minimization is exact). This sequence

$$\beta_k=\frac{\|g_k\|^2}{|\langle g_k-g_{k-1},g_k\rangle|}\tag{1.14}$$

has striking resemblance to the Polak–Ribiére formula and 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., [13], [28]). In [10] global convergence properties were established for the Fletcher–Reeves version of the method (i.e., when $\beta_k \equiv 1$). Therein, it was also shown that the restriction on the scalar λ in the problem (1.8) can be removed. Furthermore, if the restriction is dropped one can use the Wolfe conditions for the step–sizes in the directional minimization instead of the conditions borrowed from algorithms for nondifferentiable optimization (these conditions are discussed in the next section).

Due to strong convergence properties exhibited by the conjugate gradient algorithm defined by (1.13) 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 optimization problem are more clustered (see [21] for the discussion of how eigenvalues clustering influences the behavior of conjugate gradient algorithms).

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

$$\hat{x} = Dx \tag{1.15}$$

then our minimization problem will become

$$\min_{\hat{x}} \left[\hat{f}(\hat{x}) = f(D^{-1}\hat{x})\right] \tag{1.16}$$

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

$$\hat{d}_k = -\mathbf{Nr}\{\nabla \hat{f}(\hat{x}_k), -\hat{\beta}_k \hat{d}_{k-1}\} \tag{1.17}$$

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, due to (1.15) notice that

$$\nabla f(x) = D^T \nabla \hat{f}(\hat{x}) = D^T \hat{g} \tag{1.18}$$

therefore we can write

$$\hat{d}_k = -\mathbf{Nr}\{D^{-T}\nabla f(D^{-1}\hat{x}_k), -\hat{\beta}_k \hat{d}_{k-1}\}. \tag{1.19}$$

If we multiply both sides of (1.19) by D^{-1} we will get

$$d_k = -\mathbf{Nr}\{D^{-1}D^{-T}\nabla f(x_k), -\hat{\beta}_k d_{k-1}\}. \tag{1.20}$$

Eventually, d_k must satisfy

$$d_k = -\lambda_k D^{-1}D^{-T} g_k + (1-\lambda_k)\,\hat{\beta}_k d_{k-1}. \tag{1.21}$$

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

$$\hat{\beta}_k = 1 \tag{1.22}$$

for the Fletcher-Reeves version, or

$$\begin{aligned} \hat{\beta}_k &= \frac{\|\hat{g}_k\|^2}{|\langle \hat{g}_k - \hat{g}_{k-1}, \hat{g}_k\rangle|} \\ &= \frac{g_k^T \left(D^T D\right)^{-1} g_k}{|\left(g_k - g_{k-1}\right)^T \left(D^T D\right)^{-1} g_k|} \end{aligned} \tag{1.23}$$

for the Polak–Ribiére version.

The equation (1.21) can be stated as

$$d_k = -\lambda_k H g_k + (1-\lambda_k)\,\hat{\beta}_k d_{k-1}. \tag{1.24}$$

where $H = (D^T D)^{-1}$. This suggests that D should be chosen in such a way that $D^T D$ is an approximation to $\nabla^2_{xx} f(\bar{x})$ where $\bar{x}$ is a solution of problem (1.1).

Moreover, D should be such that systems of linear equations

$$\begin{aligned} D^T \hat{g}_k &= g_k \quad (1.25)\\ D d_k &= \hat{d}_k, \quad (1.26)\end{aligned}$$

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

It is straightforward to show that for rule (1.17) properties similar to (1.10)-(1.12) hold:

$$\begin{aligned} \langle g_k, d_k\rangle &\le -\|\hat{d}_k\|^2, \quad (1.27)\\ -\hat{\beta}_k d_{k-1}^T H^{-1} d_k &\le -\|\hat{d}_k\|^2, \quad (1.28)\end{aligned}$$

and

$$\langle g_k, d_k\rangle = -\|\hat{d}_k\|^2,\ \hat{\beta}_k d_{k-1}^T H^{-1} d_k = \|\hat{d}_k\|^2, \tag{1.29}$$

if $0 < \lambda_k < 1$.

The aim of the paper is to present two versions of the preconditioned conjugate gradient algorithm. They differ by the way the scaling matrices are built and then used in conjugate gradient iterations. One version follows the strategy proposed by Buckley and LeNir, the other can be regarded as the extension of the limited memory quasi–Newton method. We discuss convergence properties of these methods and provide some numerical results which show that the proposed approach can lead to viable and competitive numerical algorithms.

Finally we note that we are concerned with functions defined over the Euclidean space $\mathcal{R}^n$, $\|\cdot\|$ is the Euclidean norm, $\langle\cdot,\cdot\rangle$ is a scalar product and $\mathrm{rd}(a,b)$ is the angle between two vectors a and b. Throughout the paper we will denote by $g_k = \nabla f(x_k)$ and by $g(x_k+\alpha_k d_k) = \nabla f(x_k + \alpha_k d_k)$ – the same notation applies to $\hat{g}_k$.

$\boxed{2}$ General Algorithm.

The scaling matrix D should be changed frequently to guarantee that it is as close as possible to $\nabla^2_{xx} f(x_k)$. For the simplicity of presentation we assume that it is changed at every iteration – in this section we assume that matrices $\{D_k\}_0^\infty$ are given.

Since $\hat{d}_k$ is calculated in the space determined by D_k and $\hat{d}_k$ is expressed by $\hat{d}_{k-1}$ we need additional notation:

$$\hat{d}^c_{k-1} = D_k d_{k-1}. \tag{2.1}$$

Our general algorithm is as follows.

Algorithm Parameters: $\mu,\ \eta \in (0,1),\ \eta > \mu,\ \epsilon > 0,\ \{\hat{\beta}_k\}_1^\infty$,
$\{D_k\}_1^\infty,\ D_k \in R^{n\times n}$.
Data: x_0

1. Set k=0

2. Compute:

$$d_k = -g_k. \tag{2.2}$$

If $\|d_k\| = 0$ then STOP, if not go to Step 4.

3. Compute:

$$\begin{aligned} D_k^T \hat{g}_k &= g_k & (2.3)\\ \hat{d}_{k-1}^c &= D_k d_{k-1} & (2.4)\\ \hat{d}_k &= -\mathbf{Nr}\{\hat{g}_k, -\hat{\beta}_k \hat{d}_{k-1}^c\} & (2.5)\\ D_k d_k &= \hat{d}_k & (2.6)\end{aligned}$$

if $\|d_k\| = 0$ then STOP.

4. Find a positive number α_k such that:

$$\begin{aligned} f(x_k + \alpha_k d_k) - f(x_k) &\leq -\mu\alpha_k \|\hat{d}_k\|^2 & (2.7)\\ g(x_k + \alpha_k d_k)^T d_k &\geq -\eta\|\hat{d}_k\|^2. & (2.8)\end{aligned}$$

5. Substitute $x_k + \alpha_k d_k$ for x_{k+1}, increase k by one, go to Step 3.

The directional minimization is defined by the expressions (2.7)-(2.8). These rules, which lead to inexact minimization, were taken from the algorithms for nondifferentiable problems - [16],[17],[19]. Let us observe that our conditions for directional minimization are very similar to those well-known in the literature. In order to notice that we have to replace $-\|\hat{d}_k\|^2$ in (2.7) by $g_k^T d_k$ which holds when $0 < \lambda_k < 1$. Furthermore, if we do not impose the restriction on λ we could employ the Wolfe conditions:

$$\begin{aligned} f(x_k + \alpha_k d_k) - f(x_k) &\leq \mu\alpha_k g_k^T d_k & (2.9)\\ g(x_k + \alpha_k d_k)^T d_k &\geq \eta g_k^T d_k. & (2.10)\end{aligned}$$

This version of the preconditioned conjugate gradient algorithm would be in the spirit of the method of shortest residuals discussed in [10].

A procedure which finds α_k satisfying (2.7)-(2.8), in the finite number of operations, can be easily constructed -[25].

Lemma 1 *There exists a procedure which finds α_k satisfying (2.7)-(2.8) in a finite number of operations, or produces $\alpha_k \to \infty$ such that $f(x_k + \alpha_k d_k) \to -\infty$.*

To prove global convergence results we need the following assumptions.

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

$$\|\nabla f(y) - \nabla f(x)\| \leq L\|y - x\| \tag{2.11}$$

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*

$$d_l\|x\|^2 \leq x^T D_k^T D_k x \leq d_u\|x\|^2 \tag{2.12}$$

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

The following lemma plays crucial role in proving global convergence of *Algorithm*.

Lemma 2 *If the direction d_k is determined by (2.3)–(2.6), the step-size coefficient α_k satisfies (2.7)-(2.8) and* Assumptions 1-2 *are satisfied then:*

$$\lim_{k\to\infty} \|d_k\| = 0 \tag{2.13}$$

or

$$\lim_{k\to\infty} f(x_k) = -\infty. \tag{2.14}$$

Proof. Assume that $\{f(x_k)\}$ is bounded. In the first step of the proof we will show that

$$\lim_{k\to\infty} \|\hat{d}_k\|^2 = 0. \tag{2.15}$$

(2.15) follows from the following considerations. We have

$$\langle g_{k+1}, d_k\rangle \geq -\eta\|\hat{d}_k\|^2, \tag{2.16}$$

thus

$$\begin{aligned}\langle g_{k+1} - g_k, d_k\rangle &\geq -\eta\|\hat{d}_k\|^2 - \langle g_k, d_k\rangle \\ &\geq (1-\eta)\|\hat{d}_k\|^2 \end{aligned} \tag{2.17}$$

Since *Assumption 1* is satisfied

$$\begin{aligned}\langle g_{k+1} - g_k, d_k\rangle \leq \alpha_k L\|d_k\|^2 &= \alpha_k L\|D_k^{-1}\hat{d}_k\|^2 \\ &\leq \frac{1}{d_l} L\alpha_k\|\hat{d}_k\|^2 \end{aligned} \tag{2.18}$$

which together with (2.17) imply that

$$\alpha_k \geq \frac{(1-\eta)}{\hat{L}} \tag{2.19}$$

with $\hat{L} = L/d_l$.

From directional minimization rule we also have

$$\begin{aligned} f(x_k + \alpha_k d_k) - f(x_k) &\geq \mu\alpha_k\|\hat{d}_k\|^2 \\ &\geq \mu\frac{(1-\eta)}{\hat{L}}\|\hat{d}_k\|^2 \end{aligned} \tag{2.20}$$

which implies

$$\hat{L}\,[f(x_k + \alpha_k d_k) - f(x_k)]\,/\,[(1-\eta)\mu] \geq \|\hat{d}_k\|^2 \tag{2.21}$$

Eventually we will get

$$\infty > \hat{L}\sum_{k=0}^{\infty}[f(x_k + \alpha_k d_k) - f(x_k)]\,/\,[(1-\eta)\mu] \geq \sum_{k=0}^{\infty}\|\hat{d}_k\|^2 \tag{2.22}$$

since f is bounded. This proves

$$\sum_{k=0}^{\infty} \|\hat{d}_k\|^2 < \infty \tag{2.23}$$

and (2.15).

From (2.15) we also have

$$\lim_{k\to\infty} \|\hat{d}_k\| = 0 \tag{2.24}$$

and since

$$\|d_k\| \leq \frac{1}{\sqrt{d_l}} \|\hat{d}_k\|$$

we also have (2.13). □

The condition (2.13) is not equivalent to the condition:

$$\lim_{k\to\infty} \|g_k\| = 0. \tag{2.25}$$

This is due to the additional vector $\hat{\beta}_k d_{k-1}$ in formula (1.21).

It can happen that (2.13) holds because the vectors d_{k-1} are not appropriately scaled by the $\hat{\beta}_k$. Moreover, we can have $\lim_{k\in K, k\to\infty} \mathrm{rd}(-\nabla f(x_k), d_{k-1}) = \pi$ for certain sequence $\{\nabla f(x_k)\}_{k\in K}$ and inexact line search.

In each of these situations we shall have (2.13). Thus, in order to prove the convergence of *Algorithm* we have to exclude these situations.

Theorem 3 *Suppose that* Assumptions 1-2 *are satisfied and that* $\{\hat{\beta}_k\}$ *is such that*

$$\liminf_{k\to\infty} \left(\hat{\beta}_k \|d_{k-1}\|\right) \geq \nu_1 \liminf_{k\to\infty} \|g_k\| \tag{2.26}$$

where ν_1 *is some positive constant. If there exists a number* ν_2 *such that* $\nu_2 \|D_k\|_2 \|D_k^{-1}\|_2 \in (0,1)$ *and*

$$g_k^T d_{k-1} \leq \nu_2 \|g_k\| \|d_{k-1}\|, \text{ whenever } \lambda_k \in (0,1) \tag{2.27}$$

then $\lim_{k\to\infty} f(x_k) = -\infty$*, or every cluster point* $\bar{x}$ *of the sequence* $\{x_k\}_0^\infty$ *generated by* Algorithm *is such that* $\nabla f(\bar{x}) = 0$.

Proof. Case a). Let us suppose that for infinitely often $k \in K_1$, $\lambda_k \in (0,1)$, thus:

$$\langle \hat{g}_k, \hat{d}_k \rangle = -\|\hat{d}_k\|^2 \text{ and } \hat{\beta}_k \langle \hat{d}_{k-1}^c, \hat{d}_k \rangle = \|\hat{d}_k\|^2. \tag{2.28}$$

Moreover let us assume that $x_k \xrightarrow{K_1} \bar{x}$, $\nabla f(\bar{x}) \neq 0$. From this it follows that $\lim_{k\to\infty, k\in K_1} \|g_k\| \neq 0$. Because of this, the equalities (2.28), and since by *Lemma 2*

$\lim_{k\to\infty} \|\hat{d}_k\| = 0$, we have

$$\lim_{k\to\infty, k\in K_1} \cos \mathrm{rd}\left(-\hat{g}_k, \hat{d}_k\right) \equiv \lim_{k\to\infty, k\in K_1} \cos\phi_k =$$

$$\lim_{k\to\infty, k\in K_1} \left\langle -\frac{\hat{g}_k}{\|\hat{g}_k\|}, \frac{\hat{d}_k}{\|\hat{d}_k\|} \right\rangle = \lim_{k\to\infty, k\in K_1} \frac{\|\hat{d}_k\|}{\|\hat{g}_k\|} = 0. \tag{2.29}$$

$$\lim_{k\to\infty, k\in K_1} \cos \mathrm{rd}(\hat{\beta}_k \hat{d}^c_{k-1}, \hat{d}_k) \equiv \lim_{k\to\infty, k\in K_1} \cos\delta_k =$$

$$\lim_{k\to\infty, k\in K_1} \frac{\|\hat{d}_k\|}{\|\hat{d}^c_{k-1}\|\hat{\beta}_k} = 0 \tag{2.30}$$

which follows from (2.24), (2.26) and the fact that

$$\|\hat{d}^c_{k-1}\| = \|D_k d_{k-1}\| \geq \sqrt{d_l}\|d_{k-1}\|.$$

Since (2.29), (2.30) are satisfied: $\phi_k \to \pi/2,\ \delta_k \to \pi/2$.

Let us consider the angle $\phi_k + \delta_k$. From the usual calculus it follows that

$$\lim_{k\to\infty, k\in K_1} \cos(\phi_k + \delta_k) = \lim_{k\to\infty, k\in K_1} \cos\phi_k \cos\delta_k - \lim_{k\to\infty, k\in K_1} \sin\phi_k \sin\delta_k = -1 \tag{2.31}$$

(see also Figure 1), but this implies that

$$\lim_{k\to\infty, k\in K_1} \left\langle \frac{\hat{g}_k}{\|\hat{g}_k\|}, \frac{\hat{d}^c_{k-1}}{\|\hat{d}^c_{k-1}\|} \right\rangle = 1. \tag{2.32}$$

Let $\nu_2 > 0$ be th number from (2.27). Using (2.27) together with (1.25) and (1.26) we obtain

$$\begin{aligned} \langle \hat{g}_k, \hat{d}^c_{k-1} \rangle &= \langle g_k, d_{k-1} \rangle \\ &\leq \nu_2 \|g_k\| \|d_{k-1}\| \\ &= \nu_2 \|D_k^T \hat{g}_k\| \|D_k^{-1} \hat{d}^c_{k-1}\| \\ &\leq \nu_2 \| \|D_k\|_2 \|D_k^{-1}\|_2 \|\hat{g}_k\| \|\hat{d}^c_{k-1}\| \\ &< \|\hat{g}_k\| \|\hat{d}^c_{k-1}\| \end{aligned} \tag{2.33}$$

since $\nu_2 \|D_k\|_2 \|D_k^{-1}\|_2 \in (0,1)$. Thus equality (2.32) cannot hold and, therefore, $\nabla f(\bar{x}) = 0$.

Case b). Now let consider the case $\lambda_k = 0$ which implies $d_k = \hat{\beta}_{k-1} d_{k-1}$. If it occurs infinitely often for $k \in K_2$ and $x_k \overset{K_2}{\to} \bar{x},\ \nabla f(\bar{x}) \neq 0$ we have that there is a ν_4 such that

$$\liminf_{k\to\infty, k\in K_2} \|g_k\| = \nu_4 > 0$$

and by assumption $\lambda_k = 0$, and (2.26)

$$\liminf_{k\to\infty, k\in K_2} \left(\hat{\beta}_k \|d_{k-1}\|\right) \geq \nu_1 \nu_4 > 0.$$

But $\lim_{k\to\infty, k\in K_2} \|d_k\| = 0 = \lim_{k\to, k\in K_2} \left(\hat{\beta}_k \|d_{k-1}\|\right) \geq \nu_1\nu_4 > 0$ and this is impossible.

Case c). If we have the case $\lambda_k = 1$ for $k \in K_3$ then $-d_k = g_k$ for $k \in K_3$. If $x_k \overset{K_3}{\to} \bar{x},\ \nabla f(\bar{x}) \neq 0$ then $\lim_{k\to\infty, k\in K_3} \|g_k\| > 0$ but this is a contradiction to $\lim_{k\to\infty} \|d_k\| = 0$. This completes our proof. □

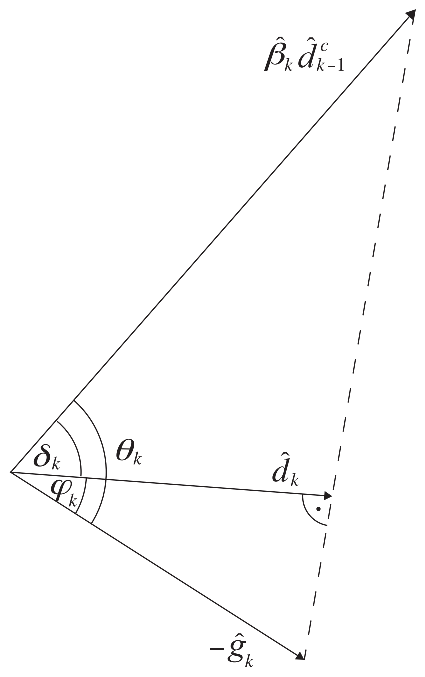

Figure 1: Calculation of d_k.

$\boxed{3}$ The Globally Convergent Conjugate gradient algorithm.

In this section we examine *Algorithm* with the sequence $\{\hat{\beta}_k\}_1^\infty$ defined by

$$\hat{\beta}_k = \frac{\|\hat{g}_k\|^2}{|\langle \hat{g}_k - \hat{g}^c_{k-1}, \hat{g}_k \rangle|}. \tag{3.1}$$

Here, $\hat{g}^c_{k-1} = D_k^{-T} g_{k-1}$ (cf, (2.1)).

We can prove the theorem:

Theorem 4 *If* Assumptions 1-2 *are satisfied then* Algorithm *gives*

$$\lim_{k\to\infty} f(x_k) = -\infty, \text{ or } \lim_{k\to\infty} \|g_k\| = 0 \tag{3.2}$$

provided that:

i) $\hat{\beta}_k$ is given by (3.1),

ii) there exists $M < \infty$ such that $\alpha_k \leq M$, $\forall k$.

Proof. From *Assumptions 1-2* we have:

$$\begin{aligned}\hat{\beta}_k \|d_{k-1}\| &= \frac{\|\hat{g}_k\|^2 \|d_{k-1}\|}{|\langle \hat{g}_k - \hat{g}^c_{k-1}, \hat{g}_k \rangle|} \\ &\geq \frac{\|\hat{g}_k\|^2 \|d_{k-1}\|}{\|\hat{g}_k - \hat{g}^c_{k-1}\| \|\hat{g}_k\|} \\ &\geq \frac{d_l \|g_k\|^2 \|d_{k-1}\|}{d_u L \alpha_{k-1} \|g_k\| \|d_{k-1}\|} \\ &\geq \frac{d_l \|g_k\|}{d_u L M}. \end{aligned} \tag{3.3}$$

Thus (2.26) holds. If f is bounded from below then, because $\|x_k - x_{k-1}\| \leq M \|d_{k-1}\|$ and $\|d_k\| \to 0$ we have

$$\lim_{k \to \infty} \|x_{k+1} - x_k\| = 0. \tag{3.4}$$

Now, we assume that for some infinite set K we have

$$\lim_{k \to \infty, k \in K} \|g_k\| = \alpha > 0 \tag{3.5}$$

Theorem 3 implies that for every $\nu \in (0,1)$ such that $\nu \|D_k\|_2 \|D_k^{-1}\|_2 \in (0,1)$ we have

$$g_k^T d_{k-1} > \nu \|g_k\| \|d_{k-1}\|.$$

But for sufficiently large $k \in K$, since $\|x_{k+1} - x_k\| \to 0$, we will achieve the relation (from (1.27) and (3.5))

$$\begin{aligned} 0 < \alpha\nu &\leq \lim_{k \to \infty, k \in K} \left\langle g_k, \frac{d_{k-1}}{\|d_{k-1}\|} \right\rangle \\ &= \lim_{k \to \infty, k \in K} \left\langle g_{k-1}, \frac{d_{k-1}}{\|d_{k-1}\|} \right\rangle \\ &\leq \lim_{k \to \infty, k \in K} \left(-\frac{\|\hat{d}_{k-1}\|^2}{\|d_{k-1}\|} \right) \leq 0. \end{aligned}$$

This is impossible, thus (3.5) cannot happen. □

$\boxed{4}$ Scaling Matrices

In the previous section we showed that for a given sequence of nonsingular matrices $\{D_k\}$, where D_k satisfies (2.12), the preconditioned conjugate gradient algorithm is globally convergent. In addition, on the matrices D_k we should impose the condition that the matrix $H_k = (D_k^T D_k)^{-1}$ is such that H_k^{-1} is as close as possible to the Hessian $\nabla^2_{xx} f(x_k)$. Furthermore, H_k should be easily factorized as $(D_k^T D_k)^{-1}$ where D_k is a nonsingular matrix.

In the paper we present the preconditioned conjugate gradient algorithm based on the BFGS updating formula. Suppose that B_k is an approximation of $\nabla^2_{xx} f(x_k)$ then the update of B_k to an approximation of the Hessian at the point x_{k+1} is given by

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k^T}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} \tag{4.6}$$

where $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$. To maintain that B_{k+1} is positive definite we require that $s_k^T y_k > 0$.

We propose two strategies of using B_k matrices in a preconditioned conjugate gradient algorithm.

S1 We start with some diagonal matrix

$$B_r = \gamma_r I_n, \tag{4.7}$$

and then apply formula (4.6) for the next m quasi–Newton iterations to obtain B_{r+m}. Then we factorize B_{r+m} as $B_{r+m} = D_{r+m}^T D_{r+m}$ and D_{r+m} is used for the next $n_r - m$ iterations in the preconditioned conjugate gradient algorithm defined by the search direction rule (2.3)–(2.6). This strategy corresponds to Buckley–LeNir algorithm if we assume that

$$\gamma_r = \frac{y_{r-1}^T y_{r-1}}{s_{r-1}^T s_{r-1}}. \tag{4.8}$$

S2 On each iteration the search direction is calculated by (2.3)–(2.6). Every n_r iterations the scaling matrix D_k is recalculated based on the m most recent vector pairs $\{s_i, y_i\}_{i=k-m}^{k-1}$ for which $s_i^T y_i > 0$, $i = k-m, \ldots, k-1$ was fulfilled (assuming $k \geq m$). In order to calculate D_k we start with the matrix $B_k^0 = \gamma_k I_n$ with γ_k given by (4.8) (with r replaced by k) and apply BFGS updates with these vector pairs to B_k^0. The resulting matrix B_k is then factorized as $B_k = D_k^T D_k$.

The convergence analysis of the versions of *Algorithm* employing B_k obtained by the BFGS updates requires some properties of these updates when applied to functions which satisfy the following assumption.

Assumption 3 *The level set*

$$\mathcal{L} = \{x \in \mathcal{R}^n : \ f(x) \leq f(x_1)\} \tag{4.9}$$

is convex and there exist positive constants M_l *and* M_u *such that* $0 < M_l < M_u < +\infty$ *and*

$$M_l \|z\|^2 \leq z^T \nabla_{xx}^2 f(x) z \leq M_u \|z\|^2 \tag{4.10}$$

for all $x \in \mathcal{N}$ *and all* $z \in \mathcal{R}^n$. *Here,* $\mathcal{N}$ *is an open set containing* $\mathcal{L}$.

Under *Assumption 3* we have

Lemma 5 *Suppose that* Assumption 3 *holds. If* B_k *is updated according to the strategy* S1, *or strategy* S2 *then there exist positive constants* c_1 *and* c_2 *such that*

$$\begin{aligned} \operatorname{trace}(B_k) &\leq c_1 & (4.11)\\ \det(B_k) &\geq c_2. & (4.12)\end{aligned}$$

Proof. We show the proof for the case of strategy *S2* (the proof for the case *S1* is analogical). Following [6] (see also [5] and [21]) we can show that following inequalities hold.

$$\begin{aligned} \operatorname{trace}(B_{k+1}) &= \operatorname{trace}(B_k) - \frac{\|B_k s_k\|^2}{s_k^T B_k s_k} + \frac{\|y_k\|^2}{s_k^T y_k}, & (4.13)\\ \det(B_{k+1}) &= \det(B_k) \frac{s_k^T y_k}{s_k^T B_k s_k} & (4.14)\end{aligned}$$

where trace(A) and det(A) denote the trace and the determinant of the matrix A.

Define

$$\bar{G}_k = \int_0^1 \nabla^2_{xx} f(x_k + \tau\alpha_k d_k) d\tau.,$$

then

$$y_k = \bar{G}_k s_k$$

and

$$M_l \le \frac{\|y_k\|^2}{s_k^T y_k} = \frac{s_k^T \bar{G}_k^2 s_k}{s_k^T \bar{G}_k s_k} \le M_u, \tag{4.15}$$

$$M_l \le \frac{s_k^T y_k}{\|s_k\|^2} \le M_u. \tag{4.16}$$

Since

$$B_k^0 = \frac{y_{k-1}^T y_{k-1}}{s_{k-1}^T y_{k-1}} I_n \tag{4.17}$$

we have

$$\begin{aligned} \text{trace}(B_k^0) &\le nM_u, &(4.18)\\ \det(B_k^0) &\ge M_l^n. &(4.19)\end{aligned}$$

Now we take into account (4.13) and (4.18) which give the estimate

$$\text{trace}(B_k) \le nM_u + mM_u. \tag{4.20}$$

Here, m denotes the number of pairs (y_i, s_i) which build the matrix B_k. Thus we take $c_1 = (n+m)M_u$.

In order to show (4.12) we notice that (see [18])

$$\det(B_{k+1}) = \det(B_k)\frac{y_k^T s_k}{s_k^T B_k s_k} = \det(B_k)\frac{y_k^T s_k}{s_k^T s_k}\frac{s_k^T s_k}{s_k^T B_k s_k}. \tag{4.21}$$

and

$$\frac{s_k^T s_k}{s_k^T B_k s_k} \ge \frac{1}{c_1}. \tag{4.22}$$

Thus, from (4.16), we have

$$\det(B_k) \ge \left(\frac{M_l}{c_1}\right)^m \det(B_k^0) \ge \left(\frac{M_l}{c_1}\right)^m M_l^n = c_2. \tag{4.23}$$

□

It remains to show that matrices B_k could be factorized in such a way that $B_k = D_k^T D_k$. To this end we recall compact representations of quasi–Newton matrices proposed in [7].

Suppose that the k vector pairs $\{s_i, y_i\}_{i=k-m}^{k-1}$ satisfy $s_i^T y_i > 0$ for $i = k-m, \dots, k-1$. Let B_k be obtained by applying k BFGS updates with these vector pairs to B_0. We then have that

$$B_k = B_0 - [B_0 S_k \ Y_k] \begin{bmatrix} S_k^T B_0 S_k & L_k \\ L_k^T & -G_k \end{bmatrix}^{-1} \times \begin{bmatrix} S_k^T B_0 \\ Y_k^T \end{bmatrix} \tag{4.24}$$

where S_k and Y_k are the $n \times k$ matrices defined by

$$S_k = [s_{k-m}, \dots, s_{k-1}], \ Y_k = [y_{k-m}, \dots, y_{k-1}] \tag{4.25}$$

while L_k and G_k are the $k \times k$ matrices

$$\begin{aligned} (L_k)_{i,j} &= \begin{cases} s_{k-i}^T y_{k-j} & \text{if } i > j \\ 0 & \text{otherwise} \end{cases} \\ G_k &= \operatorname{diag}\left[s_{k-m}^T y_{k-m}, \dots, s_{k-1}^T y_{k-1}\right]. \end{aligned} \tag{4.26}$$

If we assume that $B_0 = \gamma_k I_n$ 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 (4.24) can be written as

$$B_k = \gamma_k I - M_k W_k M_k^T. \tag{4.27}$$

In order to transform the matrix B_k to the form $D_k^T D_k$ we do the QR factorization of the matrix M_k:

$$M_k^T = R_k Q_k \tag{4.28}$$

where Q_k is $n \times n$ orthogonal matrix and R_k the $k \times n$ matrix which has zero elements except the elements constituting the left $m \times m$ submatrix ([14]). Taking into account that $Q_k^T Q_k = I_n$ we can write (4.27) as

$$B_k = Q_k^T \left[\gamma_k I - R_k^T W_k R_k\right] Q_k. \tag{4.29}$$

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. 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

$$B_k = Q_k^T F_k^T F_k Q_k \tag{4.30}$$

with

$$F_k = \begin{bmatrix} C_k & 0 \\ 0 & \sqrt{\gamma_k} I_{n-k} \end{bmatrix}. \tag{4.31}$$

The desired decomposition of the matrix B_k is thus given by

$$B_k = D_k^T D_k, \ D_k = F_k Q_k \tag{4.32}$$

where the matrix D_k is nonsingular provided that $s_i^T y_i > 0$ for $i = k-m, \ldots, k-1$. Notice that the matrix Q_k does not have to be stored explicitly 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 ([14]).

Recall the relations (1.25)–(1.26) which now can be written as

$$Q_k^T F_k^T \hat{g}_k = g_k \tag{4.33}$$

$$F_k Q_k d_k = \hat{d}_k. \tag{4.34}$$

Solving these equations requires multiplication of vectors in $\mathcal{R}^n$ by the orthogonal matrix Q_k (or Q_k^T), and this can be achieved by the sequence of m multiplications of the Householder matrices H_k^i, $i = 1, \ldots, m$ such that $Q_k = H_k^1 H_k^2 \cdots H_k^m$. The cost of these multiplications is proportional to n. Furthermore, we have to solve the set on n linear equations with the upper triangular matrix F_k, or its transpose. The cost of these operations is also proportional to n since the matrix F_k is of the form (4.31) and we assume that $m \ll n$.

$\boxed{5}$ Conjugate Gradient Algorithm with BFGS Scaling Matrices

Algorithm has to be specified in order to take into account two strategies outlined in the previous section. If we change the scaling matrix at every iteration (it corresponds to $n_r = 1$) then it does not make sense to use the previous direction through the term $\hat{\beta}_k d_{k-1}$ to calculate d_k since it corresponds to the situation when the conjugate gradient algorithm is restarted at every iteration. If we apply *Strategy S2* then we have a limited memory quasi–Newton method.

Another possibility is to update D_k every n_r iterations. Then between the consecutive updates of the matrix D_k we apply conjugate gradient iteration as stated in (1.24). If we use *Strategy S1* we have the following algorithm.

CG–LBFGS–1 Algorithm
Parameters: $\mu,\ \eta \in (0,1),\ \eta > \mu,\ \epsilon > 0,\ \{\hat{\beta}_k\}_1^\infty,\ m,\ n_r,\ m < n_r$
Data: x_0

1. Set $k = 0$, $r = 0$.

2. Set $B_k = I$.

3. If $k < (r+1)n_r$ and $k > rn_r + m$ go to Step 5. Otherwise compute d_k according to

$$B_k d_k = -g_k \tag{5.35}$$

   If $\|d_k\| = 0$ then STOP.

4. Find a positive number α_k according to the Wolfe conditions (2.9)–(2.10). Go to Step 7.

5. Compute

$$D_r^T \hat{g}_k = g_k \tag{5.36}$$

$$\hat{d}_k = -\mathbf{Nr}\{\hat{g}_k, -\hat{\beta}_k \hat{d}_{k-1}\} \tag{5.37}$$

$$D_r d_k = \hat{d}_k \tag{5.38}$$

   if $\|d_k\| = 0$ then STOP.

6. Find a positive number α_k such that:

$$\begin{aligned} f(x_k+\alpha_k d_k)-f(x_k) &\le -\mu\alpha_k\|\hat{d}_k\|^2 && (5.39)\\ \langle g(x_k+\alpha_k d_k), d_k\rangle &\ge -\eta\|\hat{d}_k\|^2. && (5.40)\end{aligned}$$

7. Substitute $x_k+\alpha_k d_k$ for x_{k+1}, calculate $s_k = x_{k+1}-x_k$, $y_k = g(x_k+\alpha_k d_k)-g_k$.
8. If $k=(r+1)n_r$ then substitute $B_{k+1}=\gamma_{k+1}I_n$ according to (4.8) and increase r by one.
9. If $k > rn_r$ and $k < rn_r + m$ then calculate B_{k+1} according to (4.27) with k replaced by $k+1$.
10. If $k = rn_r + m$ then determine D_r by (4.32) and assume $\hat{\beta}_{k+1}=0$.
11. Increase k by one and go to Step 3.

Notice that in Step (5) we use $\hat{d}_{k-1}$, in the formula (5.37), instead of $\hat{d}^c_{k-1}$ which is justified by the fact that $D_k = D_{k-1} = D_r$.

If *Strategy S2* is applied we end up with the following version of a preconditioned conjugate gradient algorithm.

CG–LBFGS–2 Algorithm
Parameters: $\mu,\ \eta\in(0,1),\ \eta>\mu,\ \epsilon>0,\ \{\hat{\beta}_k\}_1^\infty,\ m,\ n_r,$
Data: x_0

1. Set $k=0$, $r=0$, $D_r=I_n$
2. Compute:

$$\begin{aligned} d_k &= -g_k && (5.41)\\ \hat{d}_k &= D_r d_k && (5.42)\end{aligned}$$

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

$$\begin{aligned} D_r^T\hat{g}_k &= g_k && (5.43)\\ \hat{d}_k &= -\mathbf{Nr}\{\hat{g}_k, -\hat{\beta}_k\hat{d}_{k-1}\} && (5.44)\\ D_r d_k &= \hat{d}_k && (5.45)\end{aligned}$$

   if $\|d_k\|=0$ then STOP.
4. Find a positive number α_k such that:

$$\begin{aligned} f(x_k+\alpha_k d_k)-f(x_k) &\le -\mu\alpha_k\|\hat{d}_k\|^2 && (5.46)\\ \langle g(x_k+\alpha_k d_k), d_k\rangle &\ge -\eta\|\hat{d}_k\|^2. && (5.47)\end{aligned}$$

5. Substitute $x_k+\alpha_k d_k$ for x_{k+1}, calculate $s_k = x_{k+1}-x_k$, $y_k = g(x_k+\alpha_k d_k)-g_k$.
6. If $k=(r+1)n_r$ then increase r by one and calculate new scaling matrix D_r according to (4.24)–(4.32) by taking into account the values of $s_i,\ y_i,\ i=k,\dots,k-m+1$. Assume $\hat{\beta}_{k+1}=0$.

7. Increase k by one and go to Step 3.

Following the proof of Theorem 4 in [25] the global convergence of both algorithms is a straightforward conclusion of *Theorem 3*, *Lemma 5*. Notice that due to *Assumption 3* the theorem establishes that $\{x_k\}$ converges to the unique minimizer of f.

Theorem 6 *Suppose that $\{x_k\}$ is generated by CG–BFGS–1(2) Algorithm and*

i) Assumption 3 *and Assumption 1 are satisfied,*

ii) $\hat{\beta}_k$ is given by

$$\hat{\beta}_k = \frac{\|\hat{g}_k\|^2}{|\langle \hat{g}_k - \hat{g}_{k-1}, \hat{g}_k \rangle|}. \tag{5.48}$$

Then $\{x_k\}$ converges to the minimizer of f.

Proof. B_k is a symmetric positive definite matrix since $s_k^T y_k > 0$ according to the relation

$$\langle g_{k+1}, d_k \rangle \geq -\eta \|\hat{d}_k\|^2 \geq \eta \langle g_k, d_k \rangle > \langle g_k, d_k \rangle .$$

Furthermore, we have

$$\lambda_n(B_k)\|x\|^2 \leq x^T B_k x \leq \lambda_1(B_k)\|x\|^2 \tag{5.49}$$

where $\lambda_1(B_k)$ and $\lambda_n(B_k)$ are the largest and the smallest eigenvalues of B_k.

Consider now the sequence $\{B_k\}$. From *Lemma 5* there exists positive constants d_1 and d_2 such that

$$\lambda_1(B_k) \leq d_2, \tag{5.50}$$
$$\lambda_n(B_k) \geq d_1 \tag{5.51}$$

for all k. It follows from the fact that

$$\mathrm{trace}(B_k) = \sum_{i=1}^{n} \lambda_i(B_k), \tag{5.52}$$
$$\det(B_k) = \prod_{i=1}^{n} \lambda_i(B_k) \tag{5.53}$$

where $\lambda_1(B_k), \ldots, \lambda_n(B_k)$ are eigenvalues of B_k.

(5.50)–(5.51) imply that

$$d_1\|x\|^2 \leq x^T D_r^T D_r x \leq d_2\|x\|^2 \tag{5.54}$$

for all r and $x \in \mathcal{R}^n$. This together with *Theorem 3* and arguments given in the proof of Theorem 4 in [25] imply the theorem's thesis □

The proof of *Theorem 6* establishes also that the matrices D_r generated by *CG–BFGS–1(2) Algorithm* satisfy *Assumption 2*. In general, nonlinear case, we cannot guarantee that matrices D_k are uniformly bounded. However, notice that d_k is always a direction of descent and that D_r can be substituted by the matrix I_n if the Cholesky factor F_k cannot be determined.

6 Numerical Experiments

In order to verify the effectiveness of our algorithm we have tested it on problems from the CUTE collection ([3]). We tried it on problems with various dimension although its application is recommended for solving large scale problems.

We present numerical results for *CG–LBFGS–1(2) Algorithms* which have been implemented in C on Intel PC under Linux operating system. In the implementation we used directional minimization procedure described in [20] - *CG–LBFGS–1(2) Algorithm* required several obvious modifications of the procedure in [20] due to the fact that its direction minimization rules are different from the standard Wolfe conditions. L-BFGS-B code was used with the parameter $m = 5$ and we applied $m = 4$, $n_r = 10$ in *CG-LBFGS–1 Algorithm* and $m = 3$ and $n_r = 5$ in *CG-LBFGS–2 Algorithm* (these combinations of parameters seem to best suit the discussed algorithms as far as the compromise between the number of function evaluation and CPU time is concerned).

The stopping criterion was $\|\nabla f(x)\|/\max(1, \|x\|) \leq 10^{-7}$.

The performance comparison of *CG–LBFGS–1 Algorithm* is given in Figure 2. where we compare it with the code L-BFGS-B presented in [30]. For each problem the bars represent the ratio of the number of iterations (LIT), number of function evaluations (IF) and computing time (CPU) needed by the *CG–LBFGS–1 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.

In Figure 3 the comparison between *CG–LBFGS–2 Algorithm* and the code L-BFGS-B is shown – again values above one testify in favor of the L-BFGS-B. Finally, Figure 4 illustrates how preconditioning in the conjugate gradient algorithm improves its performance. In Figure 4 values above one shows superiority of *CG–LBFGS–2 Algorithm* against *Algorithm* with $D_k = I_n$.

The results are given for problems from the CUTE collection with dimensions specified in Table 1 which presents also IF and CPU for *CG–LBFGS–1(2) Algorithms* and L-BFGS-B. $>$ in the table means that the algorithm terminated without satisfying the specified stopping criterion. Results for the same problems but with different dimensions basically follow the pattern of Figures 2–4. and for that reason are not presented here.

These results show that proposed preconditioned conjugate gradient algorithms are viable methods for solving large scale unconstrained optimization problems, competitive to the benchmark code L-BFGS-B.

7 Conclusions and Future Works

The paper presents a new approach to solving large scale nonlinear problems without constraints. Our numerical results show that it is a viable technique and competitive with the code L-BFGS-B. L-BFGS-B is a benchmark method for solving large scale problems when we cannot assume a special structure of a minimizing problem such as the partial separability of f. We think that our method gives the flexibility in achieving balance between the computing time and the number of function evaluations. We believe that the performance of the proposed method can be improved by using other updating scheme for matrices B_k and by ignoring pairs $\{s_i, y_i\}$ from the sequence $\{s_k, y_k\}, \ldots \{s_{k-m+1}, y_{k-m+1}\}$ if they are linearly dependent with the others. Notice that our approach is well–suited to this performance enhancement since during QR factorization we can identify these pairs.

The method has been extended to problems with box constraints following the approach

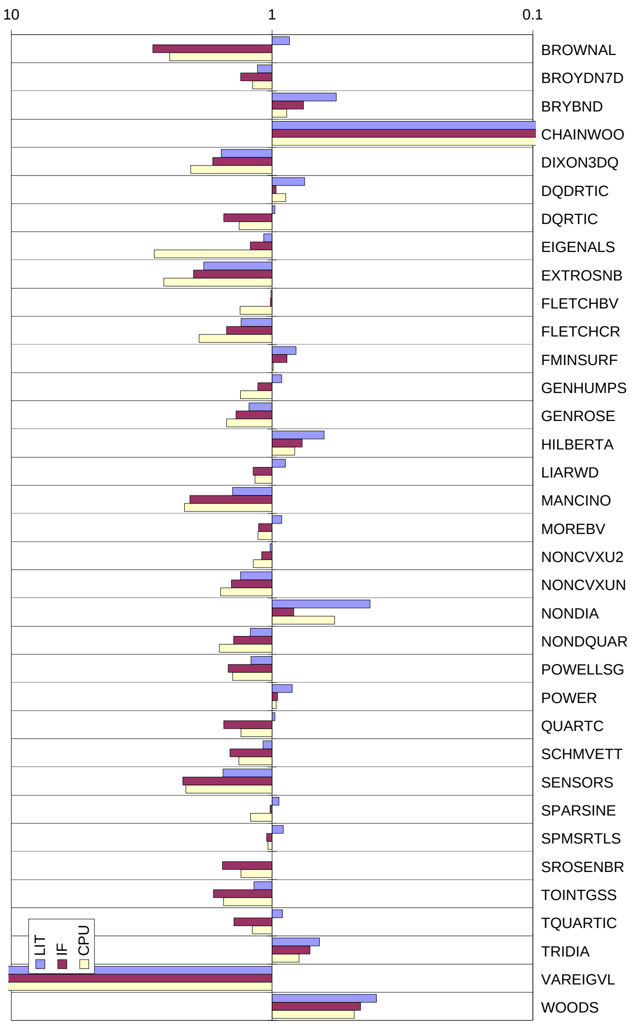

Figure 2: Performance comparison of *CG–LBFGS–1 Algorithm* against the L-BFGS-B code.

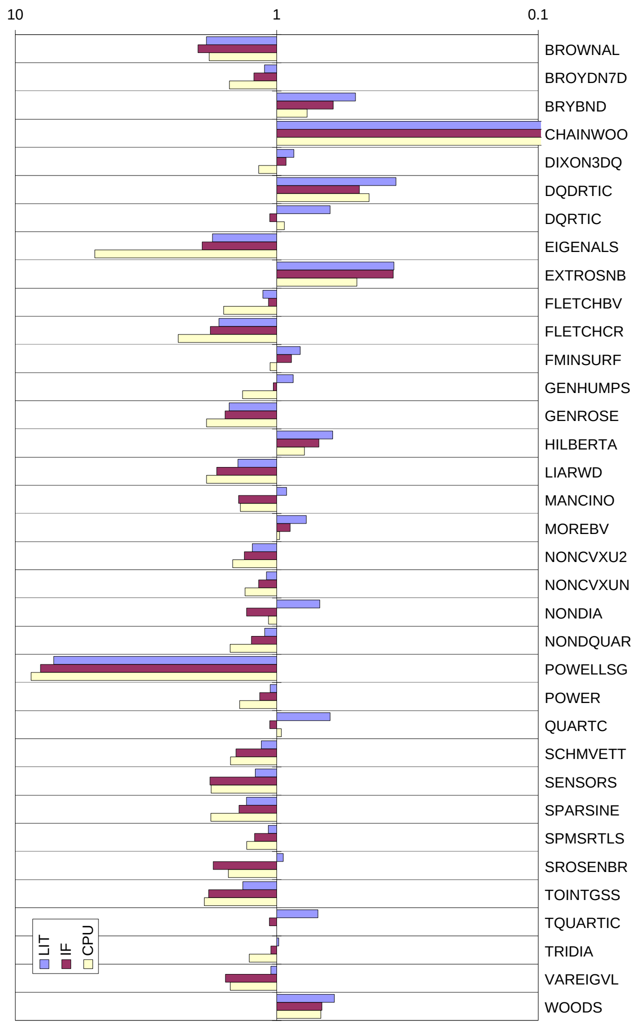

Figure 3: Performance comparison of *CG–LBFGS–2 Algorithm* against the L-BFGS-B code.

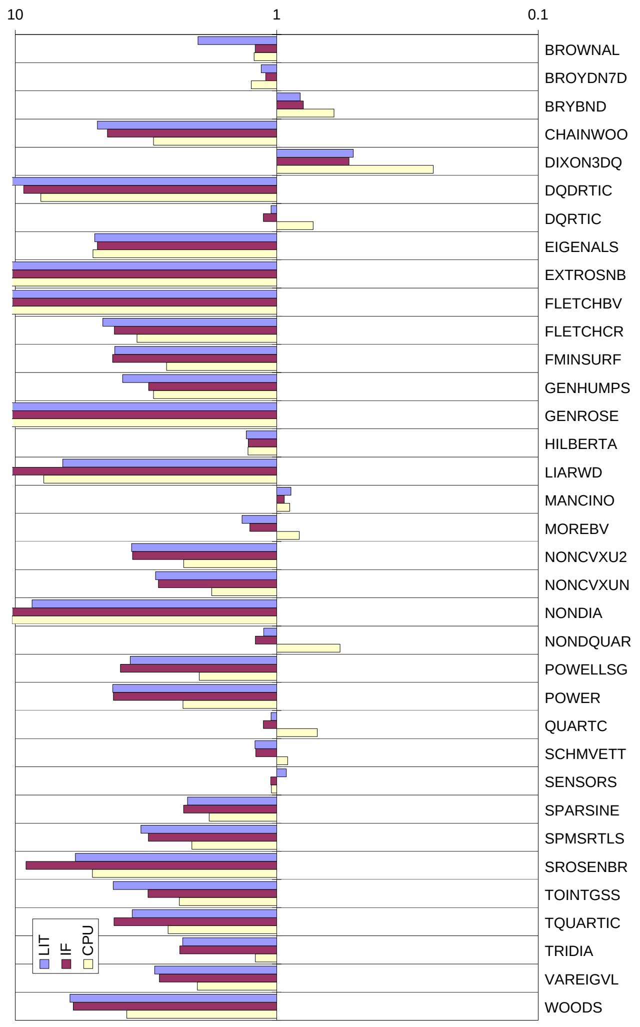

Figure 4: Performance comparison of *CG–LBFGS–2 Algorithm* against the non-preconditioned *Algorithm* (i.e. with $D_k = I_n$).

Problem	n	CG-LBFGS-1		CG-LBFGS-2		L-BFGS-B	
		IF	CPU	IF	CPU	IF	CPU
BROWNAL	1000	30	3.19	43	4.35	15	1.76
BROYDN7D	1000	473	3.14	511	2.46	387	2.07
BRYBND	10000	73	4.73	91	5.43	120	6.18
CHAINWOO	10000	590	38.10	310	19.13	>34386	>2125.00
DIXON3DQ	10000	4059	198.35	7444	347.09	4408	169.20
DQDRTIC	10000	14	0.55	28	1.10	29	1.24
DQRTIC	10000	50	1.58	72	2.26	47	1.69
EIGENALS	110	1746	2.63	1097	1.50	905	0.53
EXTROSNB	1000	4695	19.89	26207	104.88	13095	40.31
FLETCHBV	1000	3518	21.01	3318	17.46	3270	13.16
FLETCHCR	1000	9706	43.69	8084	34.93	5408	18.34
FMINSURF	10000	969	64.13	967	59.87	1103	60.47
GENHUMPS	1000	3213	17.47	3533	17.09	3116	12.93
GENROSE	10000	37817	2358.43	33004	1898.72	24001	1270.34
HILBERTA	100	593	3.58	659	3.74	860	4.57
LIARWD	10000	56	2.97	39	1.86	33	1.60
MANCINO	100	21	1.24	31	1.95	15	0.90
MOREBV	10000	56	3.15	71	3.66	63	3.23
NONCVXU2	10000	4314	275.32	3556	220.65	3243	186.83
NONCVXUN	10000	4769	299.08	5825	356.70	4067	226.30
NONDIA	10000	30	1.14	19	0.61	23	1.06
NONDQUAR	10000	1655	79.69	1860	84.28	1324	52.88
POWELLSG	10000	897	38.47	165	6.26	112	4.42
POWER	10000	755	37.41	619	26.04	650	26.99
QUARTC	10000	50	1.70	72	2.33	47	1.77
SCHMVETT	10000	73	5.38	74	4.80	51	3.58
SENSORS	100	45	1.39	55	1.67	25	0.78
SPARSINE	1000	13488	72.06	9841	48.81	9671	40.35
SPMSRTLS	10000	299	21.59	258	17.18	246	16.57
SROSENBR	10000	35	1.21	31	1.04	20	0.79
TOINTGSS	10000	51	2.48	47	2.01	28	1.31
TQUARTIC	10000	32	1.21	42	1.44	30	1.21
TRIDIA	10000	5999	306.65	4082	189.54	5699	240.61
VAREIGVL	10000	33	2.44	295	24.61	21	1.62
WOODS	10000	88	3.81	60	2.72	131	5.62
Total		100292	3938.86	112410	3518.17	116649	4546.21

Table 1: Numerical results

presented in [26]. The first numerical results of the application of this method are encouraging. They will be presented elsewhere.

Acknowledgements

The authors were very grateful to the referees for their very valuable comments and suggestions.

References

[1] M. Al-Baali, Decent property and Global Convergence of the Fletcher-Reeves Method with Inexact Line Search, *IMA J. Numer. Anal.* 5 (1985) 121–124.

[2] D.P. Bertsekas, *Constrained Optimization and Lagrange Multipliers Methods*, NY. Academic Press, New York, 1982.

[3] 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 Heights, NY, 1994.

[4] A. Buckley and A. LeNir, QN–like variable storage conjugate gradients, *Math. Program.* 27 (1983) 155–175.

[5] R.H. Byrd and J. Nocedal, A tool for the analysis of quasi–Newton methods with application to unconstrained minimization, *SIAM J. Numer. Anal.* 26 (1989) 727–739.

[6] R.H. Byrd, J. Nocedal, and Y. Yuan, Global convergence of a class of quasi–Newton methods on convex problems, *SIAM J. Numer. Anal.* 24 (1987) 1171–1190.

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

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

[9] Y. Dai and Y. Yuan, Convergence properties of the Fletcher–Reeves method, *IMA J. Numer. Anal.* 16 (1996) 155–164.

[10] Y. Dai and Y. Yuan, Global convergence of the method of shortest residuals, *Numer. Math.* 83 (1999) 581–598.

[11] Y. Dai and Y. Yuan, A nonlinear conjugate gradient method with a strong global convergence property, *SIAM J. Optim.* 10 (1999) 177–182.

[12] R. Fletcher and C.M. Reeves, Function Minimization by Conjugate Gradients, *Comput. J.* 7 (1964) 149–154.

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

[14] D. Golub and Ch.F. Van Loan, *Matrix Computations*, The John Hopkins University Press, Baltimore, 1996.

[15] M. Hestenes, *Conjugate Direction Methods in Optimization*, Springer–Verlag, Berlin Heidelberg, 1980.

[16] K. Kiwiel, *Methods of Descent for Nondifferentiable Optimization*, Lect. Notes in Math. 1133, Springer, New York, 1985.

[17] C. Lemaréchal, An Extension of Davidon Methods to Nondifferentiable Problems, in *Mathematical Programming Study* 3, M.L. Balinski and P. Wolfe (eds.), North-Holland, Amsterdam, 1975, pp. 95–109.

[18] D.C. Liu and J. Nocedal, On the Limited Memory BFGS Method for Large Scale Optimization, *Math. Program.* 45 (1989) 503–528.

[19] R. Mifflin, An Algorithm for Constrained Optimization with Semismooth Functions, *Math. Oper. Res.* 2 (1977) 191–207.

[20] J.J Morè and D.J. Thuente, Line Search Algorithms with Guaranteed Sufficient Decrease, *ACM Trans. Math. Software*, 20 (1994) 286–307.

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

[22] E. Polak and C. Ribiére, Note sur la Convergence de Methods de Directions Conjugres, *Revue Francaise Informat. Recherche Operationalle*, 16 (1969) 35–43.

[23] M.J.D. Powell, Restart Procedures for the Conjugate Gradient Method, *Math. Program.* 12 (1977) 241–254.

[24] R. Pytlak, Numerical Experiments with New Conjugate Direction Methods for Nondifferentiable Optimization, *Proceedings of the 28th IEEE CDC Conference*, December 12-15 1989, Tampa.

[25] R. Pytlak, On the convergence of conjugate gradient algorithms, *IMA J. Numer. Anal.* 14 (1994) 443–460.

[26] R. Pytlak, An efficient algorithm for large-scale nonlinear programming problems with simple bounds on the variables, *SIAM J. Optim.* 8 (1998) 532–560.

[27] D.F. Shanno, Conjugate Gradient Methods with Inexact Searches, *Math. Oper. Res.* 3 (1978) 244–256.

[28] D.F. Shanno, On the Convergence of a New Conjugate Gradient Algorithm, *SIAM J. Numer. Anal.* 15 (1978) 1247–1257.

[29] 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.

[30] C. Zhu, R.H. Byrd, P. Lu and J. Nocedal, Algorithm 778: L-BFGS-B, FORTRAN subroutines for large scale bound constrained optimization, *ACM Trans. Math. Software*, 23 (1997) 550–560.

Manuscript received 12 November 2004
revised 16 October 2005
accepted for publication 21 November 2005

R. Pytlak
Military University of Technology, Faculty of Cybernetics, ul. Kaliskiego 2, 00-908 Warsaw Poland
E-mail address: radoslaw.pytlak@isi.wat.edu.pl

T. Tarnawski
Military University of Technology, Faculty of Cybernetics, ul. Kaliskiego 2, 00-908 Warsaw Poland
E-mail address: tarni@isi.wat.edu.pl