Proceedings of the
47th IEEE Conference on Decision and Control
Cancun, Mexico, Dec. 9-11, 2008

Preconditioned conjugate gradient algorithms with column scaling

R. Pytlak
Institute of Automatic Control and Robotics
Warsaw University of Technology
02-525 Warsaw, Poland
r.pytlak@mchtr.pw.edu.pl

***Abstract*— The paper describes new conjugate gradient algorithms which use preconditioning. The algorithms are intended for general nonlinear unconstrained problems. In order to speed up the convergence the algorithms employ scaling matrices which transform 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 BFGS or 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 unique feature of these algorithms is the application of the reduced–Hessian approach to evaluate directions of descent and the use of column scaling to improve the conditioning. We believe that the proposed algorithms are competitive to limited memory quasi–Newton, or to other preconditioned conjugate gradient algorithms.**

I. Introduction

We consider the problem

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

In general, we assume that the function f is continuously differentiable, i.e., $f \in \mathrm{C}^1$.

We can use the conjugate gradient algorithm to solve problem (1). In [9] a new family of conjugate gradient algorithms was introduced based on methods proposed in [16]. Their direction finding subproblem is given by

$$d_k = -\mathbf{Nr}\{g_k, -\beta_k d_{k-1}\}, \tag{2}$$

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.,

$$\| \mathbf{Nr}\{a, b\} \| = \min\{\| \lambda a + (1-\lambda)b \| : 0 \le \lambda \le 1\}, \tag{3}$$

and $\| \cdot \|$ is the Euclidean norm. Let us notice that the operation $\mathbf{Nr}\{\cdot,\cdot\}$ can be easily performed.

If $\beta_k = 1$ then we have the algorithm given in [16] and when

$$\beta_k = \frac{\|g_k\|^2}{|\langle g_k - g_{k-1}, g_k\rangle|} \tag{4}$$

directions generated by (2) are equivalent to those provided by the Polak–Ribiére formula (under the assumption that directional minimization is exact). (2) with (4) 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.

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 problem are more clustered which is beneficial for the performance of a conjugate gradient algorithm.

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

$$\hat{x} = Dx \tag{5}$$

then our minimization problem will become

$$\min_{\hat{x}} \left[\hat{f}(\hat{x}) = f(D^{-1}\hat{x})\right] \tag{6}$$

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

$$\hat{d}_k = -\mathbf{Nr}\{\nabla \hat{f}(\hat{x}_k), -\hat{\beta}_k \hat{d}_{k-1}\} \tag{7}$$

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

$$\nabla \hat{f}(\hat{x}) = D^{-T}\nabla f(x) \tag{8}$$

therefore we can write

$$\hat{d}_k = -\mathbf{Nr}\{D^{-T}\nabla f(D^{-1}\hat{x}_k), -\hat{\beta}_k \hat{d}_{k-1}\}. \tag{9}$$

If we multiply both sides of (9) by D^{-1} we will get

$$d_k = -\lambda_k D^{-1}D^{-T}\nabla f(x_k) + (1-\lambda_k)\,\hat{\beta}_k d_{k-1}, \tag{10}$$

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

$$\hat{\beta}_k = 1 \tag{11}$$

for the Fletcher-Reeves version, or

$$\begin{aligned}\hat{\beta}_k &= \frac{\|\hat{g}_k\|^2}{|\langle \hat{g}_k - \hat{g}_{k-1}, \hat{g}_k\rangle|} \\ &= \frac{g_k^T D^{-1}D^{-T} g_k}{|\left(g_k - g_{k-1}\right)^T D^{-1}D^{-T} g_k|}\end{aligned} \tag{12}$$

for the Polak–Ribière version.

The equation (10) can be stated as

$$d_k = -\lambda_k H \nabla f(x_k) + (1-\lambda_k)\,\hat{\beta}_k d_{k-1}. \tag{13}$$

where $H = D^{-1}D^{-T}$. 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).

Moreover, D should be such that systems of linear equations

$$D^T \hat{g}_k = g_k \tag{14}$$
$$D d_k = \hat{d}_k \tag{15}$$

which we have to solve at every iteration are *easy* to solve. One can show that $g_k^T d_k \leq -\|\hat{d}_k\|^2$–see [12] for details.

Now, let turn our attention to the BFGS updating scheme:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k^T}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T y_k} \tag{16}$$

Here, $y_k = g_{k+1} - g_k$, $s_k = x_{k+1} - x_k$. The multiplicative form of the BFGS updating formula is given by

$$B_{k+1}^{-1} = \left(I - s_k z_k^T\right) B_k^{-1} \left(I - z_k s_k^T\right) \tag{17}$$

with

$$z_k = \frac{B_k s_k}{\sqrt{(s_k^T y_k)(s_k^T B_k s_k)}} + \frac{y_k}{s_k^T y_k}. \tag{18}$$

Powell in [8] assumes that B_k^{-1} can be factorized as $D_k D_k^T$, then the corresponding updating formula for D_{k+1} is given by

$$D_{k+1} = \left(I - s_k z_k^T\right) D_k. \tag{19}$$

Powell in [8] takes formula (19) as the starting point for his analysis. After some rather lengthy considerations he arrives at the updating scheme for D_{k+1}:

$$d_{k+1}^i = \begin{cases} \frac{s_k}{\sqrt{s_k^T y_k}}, & i = 1, \\ d_k^i - \left(\frac{y_k^T d_k^i}{s_k^T y_k}\right) s_k, & i = 2,3,\ldots,n. \end{cases} \tag{20}$$

A quasi–Newton algorithm with the factorization of B_k^{-1} given by $D_k D_k^T$, where D_k is updated according to (20) in exact arithmetic, is equivalent to the BFGS method. However, Powell observes that if we rescale columns of D_k of small length the numerical behavior of the quasi–Newton method significantly improves. Although such rescaling is the departure from the BFGS variable metric algorithm it does not destroy the finite termination on a quadratic ([8]).

Powell proposes the following column rescaling scheme. First, the matrix obtained in (20) we denote by $\hat{D}_{k+1}$ while D_{k+1} has columns defined by

$$d_{k+1}^i = \begin{cases} \hat{d}_{k+1}^i, & i = 1, \\ \hat{d}_{k+1}^i \max\left[1, \frac{\sigma_k}{\|\hat{d}_{k+1}^i\|}\right] \hat{d}_{k+1}^i, & i = 2,\ldots,n, \end{cases}$$

with

$$\sigma_k = \max_{1 \leq l \leq k+1} \|\hat{d}_{k+1}^l\|.$$

Another column scaling strategy is proposed in [5]. Lalee and Nocedal follow original work by Powell but they as the starting point take the factorization of the matrix B_k in the form

$$B_k = V_k V_k^T, \tag{21}$$

where V_k is a lower Hessenberg matrix. Then they construct an orthogonal matrix Q_k which transforms V_k to the lower triangular matrix $L_k = V_k Q_k$. Since Q_k is orthogonal we have

$$B_k = V_k Q_k Q_k^T V_k = L_k L_k^T. \tag{22}$$

It means that d_k can be easily evaluated by using forward and backward substitutions:

$$d_k = -L_k^{-T} L_k^{-1} g_k. \tag{23}$$

Next, the BFGS update to the matrix B_k is made:

$$B_{k+1} = W_k W_k^T \tag{24}$$

in such a way that the matrix W_k is lower Hessenberg. To this end vector r_k is determined by $r_k = L_k^T s_k$ and then an orthogonal matrix Ω_k is found such that

$$\|r_k\| e_1 = \Omega_k^T r_k. \tag{25}$$

The obvious proposition for Ω_k is the product of at most n Givens rotations. Eventually, columns of matrix W_k are evaluated by

$$w_k^i = \begin{cases} \frac{y_k}{\sqrt{y_k^T s_k}}, & i = 1, \\ L_k \Omega_k e_i, & i = 2,3,\ldots,n. \end{cases} \tag{26}$$

One can show the matrix W_k with columns defined by (26) satisfies (24).

Next, some columns of W_k are scaled giving the matrix V_{k+1}. Two parameters $\sigma_k > 0$ and $\rho_k > 0$ such that $\sigma_k \leq \rho_k$ are used to define scaling factors c_k^i:

$$c_k^i = \begin{cases} \frac{\sigma_k}{\sqrt{\|w_k^i\|}} & \text{if } \|w_k^i\| < \sigma_k, \\ \frac{\rho_k}{\|w_k^i\|} & \text{if } \|w_k^i\| > \rho_k \\ 1 & \text{otherwise.} \end{cases} \tag{27}$$

The coefficients c_k^i, $i = 1,\ldots,n$ define a scaling diagonal matrix $C_k = \mathrm{diag}[c_k^1,\ldots,c_k^n]$ which applied to matrix W_k gives the lower Hessenberg matrix V_{k+1}:

$$V_{k+1} = W_k C_k. \tag{28}$$

Lalee and Nocedal show in [5] that if the sequences $\{\sigma_k\}$ and $\{\rho_k\}$ are bounded–for all k

$$\sigma_k \leq \sigma_{\max}, \quad \rho_k \geq \rho_{\min}, \tag{29}$$

for some positive constants $\sigma_{\max}$, $\rho_{\min}$, then $\{x_k\}$ generated by quasi–Newton algorithm with the Wolfe line search rules is globally convergent for strictly convex functions.

II. Preconditioned shortest residuals algorithm with column scaling

Incorporating column scaling to conjugate gradient algorithms could follow two strategies discussed in [12]. We remind that strategy *S1* in [12] assumes that during some number of iterations, say m, positive definite matrices B_k are updated according to the BFGS rule and then applied in n_r preconditioned conjugate gradients iterates. Column scaling is used while updating B_k.

If we use strategy *S1* we have the following algorithm.

PSRCS Algorithm (The preconditioned method of shortest residuals with column scaling)

Parameters: $\mu,\ \eta \in (0,1)$, $\eta > \mu$, $\{\hat{\beta}_k\}$, m, n_r, $m < n_r$

1. Choose an arbitrary $x_1 \in \mathcal{R}^n$, compute

$$d_1 = -g_1 \tag{30}$$

and set $k = 1$, $r = 0$.

2. Find a positive number α_k satisfying the Wolfe conditions, i.e., α_k which satisfies

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

(the use of $\|\hat{d}_k\|^2$ instead of $g_k^T d_k$ is explained in [12]). Substitute $x_k + \alpha_k d_k$ for x_{k+1}.

3. If $\|g_{k+1}\| = 0$ then STOP.
If $k+1 < (r+1)n_r$ and $k+1 > rn_r + m$ go to Step 4).
If $k+1 = (r+1)n_r$ then substitute $V_{k+1} = \sqrt{\gamma_{k+1}} I_n$, $Q_{k+1} = I$ and increase r by one.
If $k+1 = (r+1)n_r + m$ then substitute $L_r = L_{k+1}$. Compute d_{k+1} by solving the equations

$$L_{k+1} z = -g_{k+1} \tag{31}$$

$$L_{k+1}^T d_{k+1} = z \tag{32}$$

where L_{k+1} is such that $V_{k+1}Q_{k+1} = L_{k+1}$, $V_{k+1} = W_{k+1}C_{k+1}$, $B_{k+1} = W_{k+1}W_{k+1}^T$, B_{k+1} is obtained from $L_k L_k^T$ by the BFGS formula and C_{k+1} is given by (27). Go to Step 5).

4. Solve the following equations with respect to d_{k+1}:

$$L_r \hat{g}_{k+1} = g_{k+1} \tag{33}$$

$$\hat{d}_{k+1} = -\mathbf{Nr}\{\hat{g}_{k+1}, -\hat{\beta}_{k+1}\hat{d}_k\} \tag{34}$$

$$L_r^T d_{k+1} = \hat{d}_{k+1} \tag{35}$$

5. Increase k by one and go to Step 2).

The global convergence of *PSRCS Algorithm* is a straightforward conclusion of Theorem 6 in [12] and the following lemma.

Lemma 1: Assume that the level set

$$\mathcal{D} = \{x \in \mathcal{R}^n :\ f(x) \leq f(x_1)\} \tag{36}$$

is convex and there exist positive constants m and M such that

$$m\|z\|^2 \leq z^T \nabla^2 f(x) z \leq M\|z\|^2 \tag{37}$$

for all $x \in \mathcal{D}$ and $z \in \mathcal{R}^n$. Let α_k be chosen according to the Wolfe conditions. If matrix B_k is obtained from the positive definite matrix B_1 by at most m updates using the scheme: 1) B_{k+1} is the BFGS update of the matrix $L_k L_k^T$; 2) $B_{k+1} = W_{k+1}W_{k+1}^T$; 3) $V_{k+1} = W_{k+1}C_{k+1}$ where C_{k+1} is the diagonal matrix based on σ_{k+1} and ρ_{k+1} such that $0 < \rho_{\min} \leq \rho_{k+1} < \infty$, $0 < \sigma_{k+1} \leq \sigma_{\max} < \infty$; 4) $V_{k+1}Q_{k+1} = L_{k+1}$ where Q_{k+1} is an orthogonal matrix and L_{k+1} upper triangular. Then, there exist positive constants c_1 and c_2 such that

$$\operatorname{trace}(L_k L_k^T) \leq c_1 \tag{38}$$

$$\det(L_k L_k^T) \geq c_2. \tag{39}$$

Proof: The proof heavily borrows from the analysis presented in the proofs of Lemma 5 in [12] and Lemma 3.3 in [5] and so it is omitted. ∎

Under the assumptions of *Lemma 1* eigenvalues of matrices $L_k L_k^T$ are uniformly bounded and are greater than zero. The direct consequence of that is the following theorem.

Theorem 2: Suppose that $\{x_k\}$ is generated by PSRCS Algorithm and

(i) the assumptions of *Lemma 1* hold,
(ii) β_k is given by (12).

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

PSRCS Algorithm is not intended for problems with many variables. Since matrices L_k are in general dense the cost of one iteration is of order n^2. In the next section we introduce an algorithm which could be efficiently applied to large scale unconstrained problems.

III. Preconditioned conjugate gradient based reduced–Hessian algorithm

The algorithm presented in the previous section can be inefficient when applied to large scale problems since it is based on the full scaling matrix. The approach related to the algorithm based on column scaling technique but intended for large scale problems is discussed in [13], [14], [15], [2] and [3].

Siegel in [13] and independently Fenelon ([1]) adopt the conjugate–direction scaling algorithm to large scale problems by referring directly to the concept of the gradient space on which the curvature information is accumulated by quasi–Newton iterates. In the case of large scale problems the number of iterations performed by quasi–Newton algorithms before the approximate solution is found is usually much smaller than the number of variables. This means that the dimension of the gradient space which is relevant in quasi–Newton algorithms for large scale problems is significantly smaller than n and can be represented by its basis consisting of few, say m, vectors.

Siegel in [13] (see also [1]) shows that the gradient space can be associated with the space spanned by the gradients evaluated up to the kth iteration–$\mathcal{G}_k = \operatorname{span}\{g_1, g_2, \ldots, g_k\}$. For further discussion crucial is the following result which we present in more general setting

than it is stated in [13]–its proof is an obvious modification of the proof of Lemma 5.1 given in [14]).

Lemma 3: Consider an algorithm applied to minimize function f whose second order derivatives are continuous and which has the kth iteration defined as

$$d_k = -B_k^{-1} g_k + \beta_k d_{k-1} \tag{40}$$

where β_k is some number, B_k is obtained by applying the BFGS update to the matrix B_{k-1} with $s_k^T y_k > 0$ and beginning from the diagonal matrix $B_1 = \sigma I$. Then, $d_k \in \mathcal{G}_k$ and if $z \in \mathcal{G}_k$, $w \in \mathcal{G}_k^{\perp}$ we have

$$B_k z \in \mathcal{G}_k, \quad B_k w = \sigma w \tag{41}$$

$$H_k z \in \mathcal{G}_k, \quad H_k w = \frac{1}{\sigma} w \tag{42}$$

where H_k is the BFGS matrix for an approximation of the inverse of the Hessian matrix beginning from $H_1 = \frac{1}{\sigma} I$. Here, $\mathcal{G}_k^{\perp}$ is the subspace orthogonal to $\mathcal{G}_k$.

Lemma 3 is the basis of the reduced–Hessian algorithm introduced in [2]. Suppose that $h_k = \dim(\mathcal{G}_k)$ and that the matrix $T_k \in \mathcal{R}^{n \times h_k}$ has columns which form the basis of $\mathcal{G}_k$. Since $\operatorname{rank}(T_k) = h_k$ there exist matrices $Q_k \in \mathcal{R}^{n \times n}$ and $R_k \in \mathcal{R}^{n \times h_k}$ such that Q_k is an orthogonal matrix and R_k is a nonsingular upper triangular matrix and the following holds ([4])

$$T_k = Q_k \begin{bmatrix} R_k \\ 0 \end{bmatrix}. \tag{43}$$

If we divide Q_k into full rank matrices Z_k and W_k: $Q_k = [Z_k \ W_k]$ and transform the space of x variables to the space of variables x^Q by $x = Q_k x^Q$ then the matrix B_k in the new space is stated as

$$Q_k^T B_k Q_k = \begin{bmatrix} Z_k^T B_k Z_k & 0 \\ 0 & \sigma I_{n-h_k} \end{bmatrix} \tag{44}$$

and the gradient g_k as

$$Q_k^T g_k = \begin{bmatrix} Z_k^T g_k \\ 0 \end{bmatrix}. \tag{45}$$

Lemma 4: Suppose that Z_k is the result of orthogonalizing the gradients $g_1, \ldots, g_k$ generated by the method based on the direction formula (40) where B_k are defined as in Lemma 3. If D_k and G_k are matrices of search directions and gradients ($D_k = [d_1 \ d_2 \cdots d_k]$, $G_k = [g_1 \, g_2 \cdots g_k]$), then there are nonsingular upper triangular matrices S_k and $\bar{S}_k$ such that

$$G_k = Z_k S_k, \quad D_k = Z_k \bar{S}_k. \tag{46}$$

Lemma 4 can be used to show that Z_k provides also the basis for the space $\mathcal{D}_k$ of search directions $d_1, \ldots, d_k$.

Theorem 5: The spaces $\mathcal{G}_k$ and $\mathcal{D}_k$ generated by gradients and search directions of the methods based on (40) with B_k defined as in Lemma 3 are identical.

From now on we assume that the dimension of the reduced space is not greater than m. *Theorem 5* states that both directions d_1, $d_2, \ldots, d_m$ and gradients g_1, $g_2, \ldots, g_m$ can be used to built the current reduced space (notice that $p_1 \parallel g_1$). However, when we move to iteration $m+1$ then in order to keep the dimension of the reduced space unchanged the new gradient g_{m+1} has to replace one of the gradients g_i, $i = 1, \ldots, k$. If we substitute g_1 and p_1 by g_{m+1} then the spaces spanned by $g_2, g_3, \ldots, g_{m+1}$ and by $d_2, d_3, \ldots, g_{m+1}$ can no longer be the same. Notice that $d_m \in \mathcal{G}_m$ can have components associated with g_1 and since $d_m \in \mathcal{D}_m$ it follows that $\operatorname{range}([g_2 \ \cdots \ g_m \ g_{m+1}]) \neq \operatorname{range}([d_2 \ \cdots \ d_m \ g_{m+1}])$.

The choice of the space used in the limited memory reduced–Hessian method influences the efficiency of the method. In [3] (see also [6]) the convergence in a finite number of iterations, in the case of strongly convex quadratic functions and with the exact line searches, is established for the method based on the space generated by directions d_k. Furthermore, it is claimed therein that the method exhibits inferior performance if the space defined by gradients g_k is used instead.

The scheme for updating the used space goes as follows. At the start of iteration k the reduced space used to compute d_k has the basis consisted of columns of the matrix

$$D_k^b = [d_l \ d_{l+1} \cdots d_{k-1} \ g_k] \tag{47}$$

where $l = k-m+1$ (we assume that $k > m+1$). Then the creation of D_{k+1}^b is done in three stages (for the simplicity of presentation we assume that g_{k+1} is accepted–see [3] for the explanation when it happens)

1. d_k is evaluated using D_k^b–the basis contains g_k in order to utilize the most recent information on function f,
2. g_k in D_k^b is replaced by d_k giving $\bar{D}_k^b$–notice that $\bar{D}_k^b$ and D_k^b differ by a single column yet according to *Theorem 5* they span the same subspace,
3. the first column is removed from $\bar{D}_k^b$ and the gradient g_{k+1} is added to $\bar{D}_k^b$ as its last column creating D_{k+1}^b.

In Stage 1) the following equations are solved

$$C_k^T z_k = -Z_k^T g_k \tag{48}$$

$$C_k d_k = z_k \tag{49}$$

with respect to z_k and d_k. C_k is the Cholesky factor of the reduced BFGS approximation to the Hessian matrix:

$$C_k^T C_k = Z_k^T B_k Z_k \tag{50}$$

and Z_k is such that

$$D_k^b = Z_k S_k \tag{51}$$

with $S_k \in \mathcal{R}^{m \times m}$ being an upper triangular matrix.

In Stage 2) the matrix Z_k does not have to be changed (according to *Lemma 4*) but the last column of S_k, which is equal to $Z_k^T g_k$, has to be replaced by $Z_k^T d_k$ forming the matrix $\bar{S}_k$. The form of the last column in S_k and $\bar{S}_k$ is the consequence of using the Gram–Schmidt orthogonalization method in the basis creation. Suppose that g_k is added to the basis. As the result matrices $\bar{S}_k$ and $\bar{D}_k^b$ are created–see [3] for details.

In Stage 3) the gradient g_{k+1} is added as the last column to $\bar{D}_k^b$ yielding $\breve{D}_k^b$ and $\breve{Z}_k$ (cf. [3]). In the next step of Stage 3) one column from $\bar{D}_k^b$ has to be removed– we assume that as the rule the column d_l leaves $\bar{D}_k^b$. In that case we have

$$\breve{D}_k^b = \left[d_l \; D_{k+1}^b \right] \tag{52}$$

and we find Z_{k+1} and S_{k+1} (cf. [3]) such that

$$D_{k+1}^b = Z_{k+1} S_{k+1}. \tag{53}$$

The approach based on the limited memory reduced–Hessian can be applied in a preconditioned conjugate gradient algorithm. If we want to use the shortest residuals algorithm as a conjugate gradient algorithm in which we apply preconditioning we have to define scaling matrix D_k such that $\hat{x}_k = D_k x_k$ at every iteration. The matrix should be nonsingular and $D_k^T D_k$ should be equal to the quasi–Newton approximation of the Hessian matrix: $B_k = D_k^T D_k$. We define D_k noting that the reduced–Hessian quasi–Newton method also is based on the transformation of variables by orthogonal matrices Q_k which are factors in the QR decompositions of the basis matrices. In the space of x^{Q_k} variables defined by the equation $x = Q_k x^{Q_k}$ the approximation to the Hessian is represented by $Q_k^T B_k Q_k$ and has the form (44) and the gradient is represented by $Q_k^T g_k$ as given by (45). The next transformation $\hat{x} = \bar{C}_k x^{Q_k}$, with $\bar{C}_k$ being the Cholesky factor of $Q_k^T B_k Q_k$, brings the Hessian approximation in the space of $\hat{x}$ variables to the identity matrix. Eventually, the transformation from the space of x variables to the space of $\hat{x}$ variables is expressed by

$$\hat{x} = \bar{C}_k Q_k^T x \tag{54}$$

and the matrix D_k used in the preconditioned shortest residuals method is stated as

$$D_k = \bar{C}_k Q_k^T. \tag{55}$$

The kth iteration of the preconditioned shortest residuals algorithm is as follows

$$\hat{d}_k = -\lambda_k \hat{g}_k + (1-\lambda_k)\hat{\beta}_k \tilde{d}_{k-1} \tag{56}$$

(here $\tilde{d}_{k-1}$ is vector transformed by the transformation (55)) while in space of x variables is expressed (as can be easily shown) by

$$d_k = -\lambda_k B_k^{-1} g_k + (1-\lambda_k)\hat{\beta}_k d_{k-1}. \tag{57}$$

In order to get a viable method we have to show that equations analogous to (14)–(15) are easy to solve. We have

$$\begin{aligned} Q_k \bar{C}_k \hat{g}_k &= g_k & (58)\\ \bar{C}_k Q_k^T d_k &= \hat{d}_k & (59) \end{aligned}$$

and equations (58), due to *Lemma 3*, reduce to the equations

$$\bar{C}_k^T \hat{g}_k = \begin{bmatrix} Z_k^T g_k \\ 0 \end{bmatrix}, \tag{60}$$

where Z_k is such that $Q_k = [Z_k \; W_k]$ and $Z_k^T W_k = 0$. On the other hand, due to *Lemma 4*,

$$\begin{aligned} D_k d_{k-1} &= \bar{C}_k Q_k^T d_{k-1} \\ &= \bar{C}_k \begin{bmatrix} Z_k^T d_{k-1} \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} C_k Z_k^T d_{k-1} \\ 0 \end{bmatrix} \end{aligned} \tag{61}$$

and we can show that equations (59) can be restated as

$$d_k = Z_k C_k^{-1} \hat{d}_k^Z, \tag{62}$$

where $\hat{d}_k^Z$ is the part of $\hat{d}_k$ corresponding to Z_k (the other part is equal to zero).

The above analysis shows that the equation (56) can be substituted by the equation

$$\hat{d}_k^Z = -\lambda_k C_k^{-1} C_k^{-T} \hat{g}_k^Z + (1-\lambda_k)\hat{\beta}_k \tilde{d}_{k-1}^Z \tag{63}$$

since the parts of all vectors appearing in (56) corresponding to the subspace spanned by columns of matrix W are zero.

It remains to specify the coefficient $\hat{\beta}_k$ in (56). We can assume that $\hat{\beta}_k = 1$ and then we have the Fletcher–Reeves version of the algorithm, or we evaluate $\hat{\beta}_k$ according to the rule

$$\hat{\beta}_k = \frac{\|\hat{g}_k\|^2}{\hat{g}_k^T (\hat{g}_k - \tilde{g}_{k-1})} \tag{64}$$

where $\tilde{g}_{k-1}$ is defined by

$$\bar{C}_k^T \tilde{g}_{k-1} = \begin{bmatrix} Z_k^T g_{k-1} \\ 0 \end{bmatrix}. \tag{65}$$

to have the Polak–Ribiére version. The main effort in solving equations (65) is associated with the calculation of $Z_k^T g_{k-1}$. Notice, however, that from the previous iteration we have at our disposal $Z_{k-1}^T g_{k-1}$ and that matrix Z_k differs from Z_k by last column so we need to evaluate only $z_k^T g_{k-1}$ at the cost of n multiplications and additions.

The evaluation of vector $\tilde{d}_{k-1}$ in equation (56) requires the same computational effort–it is obtained, according to (61), by the formula

$$\tilde{d}_{k-1} = \begin{bmatrix} C_k Z_k^T d_{k-1} \\ 0 \end{bmatrix} \tag{66}$$

which implies that $\tilde{d}_{k-1}^Z = C_k Z_k^T d_{k-1}$. Since $Z_{k-1}^T d_{k-1}$ is calculated in iteration $k-1$, the main extra effort while evaluating $\tilde{d}_{k-1}$ is associated with $z_k^T d_{k-1}$.

If the limited memory reduced–Hessian method is considered then computations required for getting $\tilde{d}_{k-1}$ and $\tilde{g}_{k-1}$ differ from that described above since Z_{k+1} is not simple enlargement of Z_k by adding a new column z_{k+1} and can be described by the formula

$$Z_{k+1} = Z_k \breve{Q}_k^T. \tag{67}$$

$\breve{Q}_k$ is the product of m Givens rotations so the cost of its applying to a given vector is proportional to m. Thus, the evaluation

$$\tilde{d}_{k-1} = \bar{C}_k Z_k d_{k-1} = \bar{C}_k Z_{k-1} \breve{Q}_{k-1}^T d_{k-1} \tag{68}$$

bears the cost proportional to mn, while $\tilde{g}_{k-1}$ satisfying the equation

$$\bar{C}_k^T \tilde{g}_{k-1} = \begin{bmatrix} \breve{Q}_k Z_{k-1}^T g_{k-1} \\ 0 \end{bmatrix} \tag{69}$$

can be determined at the cost proportional to m^2 since $Z_{k-1}^T g_{k-1}$ is known from iteration $k-1$.

We have all necessary ingredients to state the preconditioned conjugate gradient based reduced–Hessian algorithm.

R-HPCG Algorithm (Reduced–Hessian preconditioned conjugate gradient algorithm)

Parameters: $\mu,\ \eta \in (0,1)$, $\mu < \eta$, $\sigma_1 > 0$, positive integer number m.

1. Choose an arbitrary x_1, set $h_1 = 1$, $D_1^b = [g_1]$, $Z_1 = [g_1/\|g_1\|]$, $S_1 = [\|g_1\|]$, $C_1 = [\sqrt{\sigma_1}]$ and $k = 1$.

2. Calculate d_k from $\hat{d}_k$ that satisfies (56) where $\hat{g}_k$ is obtained according to formula (60), $\tilde{d}_{k-1}$ due to (68).
If g_k is accepted then swap the last column of D_k^b with d_k getting $\bar{D}_k^b$ and $\bar{S}_k$. Otherwise set $\bar{D}_k^b = D_k^b$, $\bar{S}_k = S_k$. Substitute C_k for $\bar{C}_k$.

3. Find α_k according to the Wolfe conditions. Substitute $x_k + \alpha_k d_k$ for x_{k+1}.

4. If $\|g_{k+1}\| = 0$ then STOP.
Otherwise do orthogonalization on the vectors from D_k^b and the gradient g_{k+1}.
If g_{k+1} is accepted form $\breve{D}_k^b$ from $\bar{D}_k^b$ and $\breve{Z}_k$ from Z_k and $\breve{S}_k$ from $\bar{S}_k$. Set $h_k^{'} = h_k + 1$
If g_{k+1} is rejected substitute $\bar{D}_k^b$, $\bar{Z}_k$, $\bar{S}_k$ for $\breve{D}_k^b$, $\breve{Z}_k$, $\breve{S}_k$. Set $h_k^{'} = h_k$, update $\bar{C}_k$ to $\breve{C}_k$.
Compute σ_{k+1} and do reinitialization by replacing σ_k with σ_{k+1} (improve the Hessian approximation in the space orthogonal to the reduced space by changing the diagonal matrix).
If $h_k^{'} = m+1$ then remove the first column of matrix $\breve{D}_k^b$ and create D_{k+1}^b from $\breve{D}_k^b$, S_{k+1} from $\breve{S}_k$, Z_{k+1} from $\breve{Z}_k$ and C_{k+1} from $\breve{C}_k$).
If $h_k^{'} < m+1$ substitute $\breve{D}_k$, $\breve{S}_k$, $\breve{Z}_k$ and $\breve{C}_k$ for D_{k+1}^b, S_{k+1}, Z_{k+1} and C_{k+1} respectively.
Calculate $\hat{\beta}_{k+1}$ according to (64) using $\tilde{g}_k$ defined by (69) (with $k-1$ replaced by k).
Increase k by one and go to Step 2).

The numerical algebra related to the update of $\bar{C}_k$ to $\breve{C}_k$, $\breve{D}_k^b$ to D_{k+1}^b, $\breve{Z}_k$ to Z_{k+1} and $\breve{C}_k$ to C_{k+1} is given in [3].

IV. Conclusions

The comparison in [3] shows the superiority of L-RHR (in comparison to the code based on L-BFGS algorithm) in terms of CPU time and the number of function evaluations while requiring half the storage. *R-HPCG Algorithm* is the extension of the L-HRH algorithm by giving more flexibility with the inclusion in the formula for d_k the scaled previous direction. Furthermore, it is rather straightforward to extend it to problems with box constraints along the lines stated in [11]–the method in [11] outperforms the benchmark code L-BFGS-B ([17]) on problems from the CUTE collections. The method presented in the paper should inherit superior numerical properties of the algorithm from [11] but at the same time requires less computational effort.

References

[1] M.C. Fenelon, Preconditioned conjugate–gradient–type algorithm for large–scale unconstrained optimization, Ph.D. thesis, Department of Operations Research, Stanford University, Stanford, CA, 1981.

[2] P.E. Gill and M.W. Leonard, Reduced–Hessian quasi–Newton methods for unconstrained optimization, *SIAM J. Optimization*, Vol. 12, 209–237, 2001.

[3] P.E. Gill and M.W. Leonard, 2003 Limited–memory reduced–Hessian methods for large–scale unconstrained optimization, *SIAM J. Optimization*, Vol. 14, 380–401, 2003.

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

[5] M. Lalee and J. Nocedal, Automatic column scaling strategies for quasi–Newton methods, *SIAM J. Optimization*, Vol. 3, 637–653, 1993.

[6] M.W. Leonard, Reduced Hessian quasi–Newton methods for optimization, Ph.D. thesis, Department of Mathematics, University of California, San Diego, CA, 1995.

[7] J. Nocedal, Updating quasi–Newton matrices with limited storage, *Mathematics of Computation* 35, 773–782, 1980.

[8] M.J.D. Powell, Updating conjugate directions by the BFGS formula, *Math. Programming*, Vol. 38, 29–46, 1987.

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

[10] R. Pytlak, An efficient Algorithm for large scale problems with simple bound on the variables, *SIAM J. on Optimization*, Vol. 8, 632–560, 1998.

[11] R. Pytlak and T. Tarnawski, Preconditioned conjugate gradient algorithms for large scale problems with box constraints, Proceedings of American Control Conference, July 11–13, New York, 1257–1262, 2007.

[12] R. Pytlak and T. Tarnawski, Preconditioned conjugate gradient algorithms for nonconvex problems, *Pacific Journal of Optimization*, Vol. 2, 81–104, 2006.

[13] D. Siegel, Implementing and modifying Broyden class updates for large scale optimization, Report DAMPT/1992/NA12, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, 1992.

[14] D. Siegel, The use of conjugate direction matrices in quasi-Newton methods for nonlinear optimization, Ph.D. thesis, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, 1992.

[15] D. Siegel, Modifying the BFGS update by a new column scaling technique, *Mathematical Programming*, Vol. 66, 45–78, 1993.

[16] 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, 145–173.

[17] 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.