A SURVEY OF GRADIENT METHODS
FOR SOLVING NONLINEAR OPTIMIZATION

PREDRAG S. STANIMIROVIĆ
University of Niš, Faculty of Sciences and Mathematics
Višegradska 33, 18000 Niš, Serbia

BRANISLAV IVANOV1, HAIFENG MA2 AND DJANA MOSIĆ3
1 Technical Faculty in Bor, University of Belgrade
Vojske Jugoslavije 12, 19210 Bor, Serbia
2 School of Mathematical Science, Harbin Normal University
Harbin 150025, China
3 University of Niš, Faculty of Sciences and Mathematics
Višegradska 33, 18000 Niš, Serbia

ABSTRACT. The paper surveys, classifies and investigates theoretically and numerically main classes of line search methods for unconstrained optimization. Quasi-Newton (QN) and conjugate gradient (CG) methods are considered as representative classes of effective numerical methods for solving large-scale unconstrained optimization problems. In this paper, we investigate, classify and compare main QN and CG methods to present a global overview of scientific advances in this field. Some of the most recent trends in this field are presented. A number of numerical experiments is performed with the aim to give an experimental and natural answer regarding the numerical one another comparison of different QN and CG methods.

1. Introduction and preliminaries. In this survey, we focus on solving the unconstrained nonlinear optimization problem

$$\min f(x), \ x \in \mathbb{R}^n,$$

(1.1)

where the function $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and bounded from below. The general iterative rule for solving (1.1) starts from an initial approximation $x_0 \in \mathbb{R}^n$ and generates a sequence \{x_k, k \geq 0\} using the general iterative scheme

$$x_{k+1} = x_k + \alpha_k d_k, \ k \geq 0,$$

(1.2)

where the step-size $\alpha_k$ is a positive real parameter determined after the exact or inexact line search, $x_k$ is the last generated iterative point, $x_{k+1}$ is the current iterative point, and $d_k$ is an appropriate search direction. General class of algorithms of the form (1.2) is known as the line search algorithms. These algorithms require only the search direction $d_k \in \mathbb{R}^n$ and the step-size $\alpha_k \in \mathbb{R}$.

The following notations will be used, as usual:

$$g(x) = \nabla f(x), \ G(x) = \nabla^2 f(x), \ g_k = \nabla f(x_k), \ G_k = \nabla^2 f(x_k),$$

2020 Mathematics Subject Classification. Primary: 65K05, 90C30; Secondary: 90C06, 90C26.

Key words and phrases. Gradient methods, conjugate gradient methods, unconstrained optimization, nonlinear programming, sufficient descent condition, global convergence, line search.

* Corresponding author: Predrag S. Stanimirović.
where $\nabla f(x)$ denotes the gradient and $\nabla^2 f(x)$ denotes the Hessian of $f$. For the sake of simplicity, the notation $f_k$ will point to $f(x_k)$. Further, $x^T$ denotes the transpose of $x \in \mathbb{R}^n$. Taylor’s approximation of the function $f$ at the point $x_{k+1} = x_k + \alpha_k d_k$ is defined by

$$f(x_{k+1}) \approx f(x_k) + \alpha_k g_k^T d_k.$$  

Therefore, an appropriate descent search direction $d_k$ must be determined on the basis of the descent condition

$$g_k^T d_k < 0, \text{ for all } k.$$  

Primary choice for descent direction is $d_k = -g_k$, which reduces the general line search iterations (1.2) into the gradient descent (GD) iterative scheme

$$x_{k+1} = x_k - \alpha_k g_k.$$  

In this paper, we survey gradient methods satisfying the descent condition (1.4). If there exists a constant $c > 0$ such that

$$g_k^T d_k < -c\|g_k\|^2, \text{ for all } k,$$

where $c$ is a positive constant independent of $k$, then it is said that the vector $d_k$ satisfies the sufficient descent condition.

As for the choice of search direction, one of the possible choices for the search direction in unconditional optimization is to move from the current point along the negative gradient in each iteration, which correspond to $d_k = -g_k$. This choice of search direction leads us to a class of methods known as gradient descent methods. One negative feature of gradient methods is relatively frequent occurrence of, so called, zig-zagging phenomenon, which initiates very slow convergence of GD algorithms to the optimal point, or even divergence [90].

Advantages and disadvantages of GD methods can be summarized as follows.

1. GD methods are globally convergent, i.e., converge to a local minimizer regardless of the starting point.
2. Many optimization methods switch to GD rule when they do not make sufficient progress to the convergence.
3. The convergence is linear and usually very slow.
4. Numerically, GD methods are often not convergent.

Another important direction of the search is the Newton’s direction $d_k = -G_k^{-1} g_k$, obtained from the second-order Taylor-development, assuming that Hessian $G_k$ is positive-definite. The pure Newton method (without line search) for minimization of a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is defined using a quadratic approximation of $f(x_{k+1})$:

$$\Phi(d) := f(x_k + d) \approx f(x_k) + g_k^T d + \frac{1}{2} d^T G_k d.$$  

The solution $d_k = \min_d(\Phi(d))$ is given by

$$d_k = -G_k^{-1} g_k.$$  

So, the pure Newton method is defined by

$$x_{k+1} = x_k - G_k^{-1} g_k.$$  

The Newton method with line search uses an appropriate step-size $\alpha_k$ in (1.8) with the aim to ensure global stability. The resulting iterations are of the form

$$x_{k+1} = x_k - \alpha_k G_k^{-1} g_k,$$  

(1.9)
wherein the step-size $\alpha_k$ is computed performing a line search.

The Newton method exhibits three major drawbacks in practical applications.

1. The descent (and convergence) may not be achieved if the iterations (1.8) are started far away from the local minimizer.

2. Another drawback is numerically expensive and tedious necessity to compute the second derivative matrix (Hessian) and its inverse in every iteration. Moreover, the second derivatives may be sometimes unavailable.

3. The main disadvantages of the Newton method are the possibility that the Hessian $G_k$ is not positive definite.

Due to that, numerous modifications of it were created, which can be globally divided into two large groups: modified Newton’s methods and quasi-Newton (QN) methods. The QN methods are aimed to address all the above difficulties of the Newton method. The first drawback is overcome by taking an appropriately defined positive definite matrix $B_k$ that approximates the Hessian $G_k$ or an appropriately defined positive definite matrix $H_k$ that approximates the true Hessian inverse $G^{-1}$ and then performing a line search at each iteration. For a given initial point $x_0 \in \mathbb{R}^n$ and a symmetric positive definite matrix $H_0$, the search direction in the $k$th iteration of the quasi-Newton method is defined by $d_k = -H_k g_k$, where $H_k$ is a symmetric and positive-definite matrix.

QN methods and modified Newton methods belong to most powerful methods for solving unconditional optimization and applicable in many nonlinear optimization problems. A survey of QN methods for solving nonlinear least-squares problems was considered in [86]. The optimization methods have found a number of applications in fluid mechanics [44], free surface flow and solid body contact [13], finding the optimal trajectory for an aircraft or a robot arm, designing a portfolio of investments, controlling a chemical process, computing the optimal shape of an automobile. See [90] for more details. A modification of the quasi-Newton method in defining the two-phase approximate greatest descent was used in [71]. Several variants of multi-step spectral gradient methods for solving large scale unconstrained optimization problems were proposed in [111]. Usage of an optimization algorithm in artificial neural networks was considered in [75]. Properties of Hessian matrix which appear in distributed gradient-based multi-agent control systems was considered in [117]. A survey of derivative-free optimization methods was given in [127]. An application of unconstrained optimization in solving the risk probability was presented in [76].

The study of conjugate gradient (CG) methods was started by Hestenes and Stiefel in 1952 in [60], and the development of CG methods for solving large-scale unconstrained optimization problems is still ongoing. After all these years, there is still a need to find a more efficient CG method that will solve unconstrained optimization problems with thousands of variables in the shortest possible time interval as well as with a minimal number of iterations and function evaluations.

CG methods construct a sequence of approximations $x_k$ by the line search rule (1.2), such that the search directions $d_k$ are generated by

$$d_k := d_k := d(\beta_k, g_k, d_{k-1}) = \begin{cases} -g_0, & k = 0, \\ -g_k + \beta_k d_{k-1}, & k \geq 1, \end{cases} \tag{1.10}$$

where $\beta_k$ is the real value which is known as the conjugate gradient update parameter (CGUP). More precisely, the search direction $d_k$ of the CG method is defined as a proper linear combination of the gradient descent direction and a positive multiple of the direction used in the previously finished iteration. From (1.10) and (1.2), it
clearly follows that the CG methods are defined simply only by the gradient direction $g_k$ and by the CGUP $\beta_k$. Different CG methods arise from proper choices of the scalar $\beta_k$. According to the common agreement, $\beta_k^M$ denotes the parameter $\beta_k$ of the CG method $\mathcal{M}$. It is important to mention that some researchers propose usage of $\beta_{k+}^M = \max\{\beta_k^M, 0\}$. So, it is possible to use $\beta_{k+}^M$ instead of $\beta_k^M$ and generate corresponding dual method.

Popularity of CG methods is confirmed by a number of recent surveys and book chapters [42, 57, 85, 87, 88]. In addition to this basic information on the chronological development of the CG methods, it is also important to mention its applications. In general, CG methods are important in solving large-scale optimization problems. CG methods iterates are characterized by low memory allocation and strong local and global convergence properties. Based on this fact, these methods become useful in all areas where optimization problems of any kind are involved. The CG methods have wide use in solving systems of equations and image restoration problem [12, 16, 70, 78, 128, 135, 136, 140], the linear response eigenvalue problem [74], in regression analysis [108, 143]. On that way, CG methods have the influence on the development of an artificial neural networks learning algorithms [46, 75]. A unique approach to the ABS type CG methods was proposed in [1]. Application of CG methods in solving very large symmetric positive semi definite linear systems that appear in optimal surface parameterizations are described in [64]. Also, it is possible to mention application in data analysis [110]. A variant of the projected preconditioned conjugate gradient method and its application in solving the linear response eigenvalue problem was investigated in [74].

Main goals leading current research paper can be highlighted as follows.

1. A survey and specific classifications of CG and QN methods for nonlinear unconstrained optimization is presented.
2. Convergence properties of CG methods are investigated.
3. Specific numerical testings are performed on both the CG and QN methods. Numerical testing on some classes of CG methods and hybrid CG methods as well as on some QN methods is presented. A numerical experiment about the influence of the scalar $\tau$ in Dai-Liao CG methods is performed and analysed. Also, gradient descent methods defined by appropriate acceleration parameter are tested and compared.

The overall structure of the paper based on contents of each section is described as follows. Section 1 describes basic notation, introductory notions, preliminaries and motivation. Global algorithms and various line search variants are presented in Section 2. Overview of QN methods and their classification are considered in Section 3. Section 4 gives a specific overview of CG methods. Convergence properties of considered CG methods are investigated in Section 5. According to the presented taxonomy of basic CG methods, properties of CG methods with $y_{k-1}^T g_k$ in the numerator of $\beta_k$ are considered in Subsection 5.1, properties of CG methods involving $\|g_k\|^2$ in the numerator of $\beta_k$ are given in Subsection 5.2, while the convergence properties of DL methods are presented in Subsection 5.3. Numerical experiments are performed in Section 6. In details, Subsection 6.1 arranges numerical results on QN methods with constant diagonal Hessian approximation, Subsection 6.2 compares numerically basic CG methods involving $y_{k-1}^T g_k$ in the numerator of $\beta_k$, Subsection 6.3 compares basic CG methods with $\|g_k\|^2$ in the
A SURVEY OF GRADIENT METHODS FOR NONLINEAR OPTIMIZATION

numerator of $\beta_k$, while numerical experiments on the hybrid CG methods are presented in Subsection 6.4. Finally, Subsection 6.5 describes numerical experiments on the modified Dai-Liao methods. Concluding remarks are given in Section 7.

2. Global algorithms and line search variants. First, we present an algorithm that describes the general scheme of line search methods

**Algorithm 1** Global line search algorithm

Require: Objective $f(x)$, initial point $x_0 \in \mathbb{R}^n$ and the tolerance $0 < \varepsilon \ll 1$.

1: $k := 0$.
2: while $\|g_k\| > \varepsilon$ do
3: Determine the vector $d_k$ which represents the search direction.
4: Compute the step length $\alpha_k$
5: Compute the new approximation of the minimum $x_{k+1} := x_k + \alpha_k d_k$
6: $k := k + 1$
7: end while

Ensure: $x_{k+1}, f(x_{k+1})$

2.1. Line search procedures. To achieve the global convergence of iterative methods, an appropriate step-size $\alpha_k$ is required. The most promising at first glance is the exact line search (ELS), which assumes the unidimensional function

$$\Phi(\alpha) := f(x_k + \alpha d_k)$$

and the step-size is defined after the unidimensional optimization of the form

$$f(x_k + \alpha_k d_k) = \min_{\alpha > 0} \Phi(\alpha).$$

The ELS rule may give the most precise minimum. However, ELS is too expensive in practice or even impossible to implement, especially in situations where the iteration is far from the exact solution.

Applying the iterative procedure (1.2), it is most logical to choose a new point so that the step length $\alpha_k$ reduces the value of the goal function:

$$\Phi(\alpha_k) = f(x_{k+1}) < \Phi(0) = f(x_k).$$

Methods that in each iterative step require the fulfillment of conditions (2.3), that is, the reduction of the value of the objective function, define iterations that in each step approach the minimum of the given function. The methods conceived in this way belong to the class of methods of monotone line search. Many variants of inexact line search (ILS) rules are proposed and dominant in the nonlinear optimization. The most commonly used ILS techniques are Armijo, Goldstein, Wolfe, Powel, Fletcher and other [4, 8, 19, 50, 55, 56, 109, 126]. In most conjugate gradient methods, one of the next ILS procedures methods is used to calculate the step length $\alpha_k$: Wolfe line search developed in [55, 56], strong Wolfe line search, or backtracking line search from [4].

In contrast to the monotonic line search, the non-monotonic line search is also known in the literature, where it is not necessary to reduce the value of the objective function in each iteration [51, 52, 53]. Although non-monotone techniques do not provide a minimum approach to the function in each iteration, they are very common in practical applications and have very good convergence properties. A
number of nonmonotonic linear search methods have been proposed recently (see, for example, [119, 120]).

2.1.1. Backtracking line search. A backtracking line search scheme based on the Armijo condition, is aimed to determining the maximal value during the moving along a given search vector. It starts with a relatively large step-size estimate and iteratively reduces the step-size value until a decrease in the value of the objective function is observed, according to the local gradient of the goal function. Let \( \beta \in (0, 1) \), \( \phi \in (0, 1) \) and \( \alpha > 0 \) be given. Then there exists a smallest nonnegative integer \( m_k \) satisfying

\[
f(x_k + \beta^m t d_k) \leq f(x_k) + \phi \beta^m t g_k^T d_k, \quad t > 0.
\] (2.4)

The procedure for backtracking line search proposed in [4] starts from the initial value \( \alpha = 1 \) and its output values are defined such that it decreases the goal function. Consequently, Algorithm 2 from [112] is used in numerical experiments as the implementation of the ILS which defines the principal step-size \( \alpha_k \).

**Algorithm 2** The backtracking line search.

**Require:** Nonlinear multivariate function \( f(x) \), the vector \( d_k \), previous approximation \( x_k \), and the real numbers \( 0 < \omega < 0.5 \) and \( \varphi \in (0, 1) \).

1: \( \alpha = 1 \).
2: While \( f(x_k + \alpha d_k) > f(x_k) + \omega \alpha g_k^T d_k \), take \( \alpha := \alpha \varphi \).
3: Return \( \alpha_k = \alpha \).

2.1.2. Goldstein line search. In order to ensure a sufficient decrease of the objective function, Goldstein rule for ILS requires the following conditions:

\[
f(x_k + \alpha d_k) \leq f(x_k) + \rho t g_k^T d_k \] (2.5)

and

\[
f(x_k + \alpha d_k) \geq f(x_k) + (1 - \rho) t g_k^T d_k, \] (2.6)

where \( 0 < \rho < \frac{1}{2} \) and \( t > 0 \). Conditions (2.5) and (2.6) define the Goldstein rule for inexact line search.

2.1.3. Wolfe line search. Wolfe line search conditions are well-known and these are given by

\[
f(x_k + \alpha d_k) \leq f(x_k) + \eta \alpha g_k^T d_k, \] (2.7)

\[
g(x_k + \alpha d_k)^T d_k \geq \sigma_1 g_k^T d_k, \] (2.8)

where \( 0 < \eta < \sigma_1 < 1 \). In addition, for conjugate gradient methods, the generalized strong Wolfe conditions, which are a conjunction of (2.7) and

\[
-\sigma_2 g_k^T d_k \geq g(x_k + \alpha_k d_k)^T d_k \geq \sigma_1 g_k^T d_k \] (2.9)

are often used, where \( \sigma_1 > 0 \). In the case \( \sigma_1 = \sigma_2 \), the generalized strong Wolfe conditions reduce to the strong Wolfe conditions, which are a conjunction of (2.7) and

\[
|g(x_k + \alpha d_k)^T d_k| \leq -\sigma_1 g_k^T d_k. \] (2.10)

The condition (2.7) of the Wolfe conditions is called the Armijo condition, which is often used apart or in the form of its variants.
3. Quasi-Newton methods and their classification. The most general iterative rule of QN type with line search is of the form
\[ x_{k+1} = x_k - \alpha_k H_k g_k, \]
(3.1)
such that \( H_k \) is an approximation of the inverse Hessian \( G_k^{-1} \). Further, it is assumed that \( B_k \) is an appropriately generated symmetric positive definite approximation of \( G_k \) [118]. The following notations in defining an appropriate updating formula
\[ s_k = x_{k+1} - x_k, \quad y_k = g_{k+1} - g_k \]
(3.2)
are typical. The update \( B_{k+1} \) of \( B_k \) is defined using the rule
\[ B_{k+1} = B_k + E_k, \]
(3.3)
where \( E_k \) is defined on the basis of the quasi-Newton property (secant condition)
\[ B_{k+1}s_k = y_k. \]
(3.4)
The quasi-Newton condition for the matrix \( H_k \) is given by
\[ H_{k+1}y_k = s_k. \]
(3.5)
Methods that require the calculation or approximation of the Hessian matrix and its inverse belong to the class of QN methods as well as its numerous modifications. The pure Newton method requires calculation of second derivatives matrix, which is avoided in QN methods. As a consequence, the Newton method is computationally expensive and exhibits slow computation, while QN methods are computationally cheap and of faster computation. On the other hand, the Newton method requires lesser number of iterative steps and generates more precise convergence path than QN methods.

3.1. Symmetric Rank-One update. The Symmetric Rank-One update (SR1) assumes the matrix \( H_{k+1} \) in the form
\[ H_{k+1} = H_k + E_k, \]
where \( E_k \) is assumed to be a symmetric rank-one matrix. Therefore,
\[ H_{k+1} = H_k + u_k v_k^T, \]
(3.6)
where
\[ u_k, v_k \in \mathbb{R}^n. \]
The quasi-Newton condition (3.5) initiates
\[ H_{k+1}y_k = (H_k + u_k v_k^T)y_k = s_k, \]
that is
\[ (v_k^T y_k)u_k = s_k - H_k y_k. \]
(3.7)
The conclusion is that \( u_k \) must be in the direction \( s_k - H_k y_k \). Suppose that
\[ s_k - H_k y_k \neq 0, \]
(otherwise \( H_k \) would satisfy the quasi-Newton equation) and the vector \( v_k \) satisfies \( v_k^T y_k \neq 0 \). Then, on the basis of (3.6) and (3.7), it follows that
\[ H_{k+1} = H_k + \frac{1}{v_k^T y_k} (s_k - H_k y_k) v_k^T. \]
(3.8)
The condition that the approximation $H_{k+1}$ of the Hessian inverse is symmetric requires us to take $v_k = s_k - H_k y_k$, so it was obtained

$$H_{k+1} = H_k + \frac{(s_k - H_k y_k)(s_k - H_k y_k)^T}{(s_k - H_k y_k)^T y_k},$$

(3.9)

which is the Symmetric Rank One (SR1) update.

For general functions, Conn, Gould and Toint in [20] proved that the sequence of SR1 Hessian approximations converges to the true Hessian provided that the steps are uniformly linearly independent.

3.2. DFP update. The Hessian update $B_{k+1}$ is defined as the solution to the problem

$$\min_B \| B - B_k \|, \quad \text{s.t. } B = B^T, \ B s_k = y_k.$$  

(3.10)

The solution to (3.10) is equal to

$$B_{k+1}^{\text{DFP}} = (I - \rho_k y_k s_k^T) B_k^{\text{DFP}} (I - \rho_k s_k y_k^T) + \rho_k y_k y_k^T, \quad \rho_k = \frac{1}{y_k s_k}.$$  

The inverse Hessian update can be generated using the Sherman - Morrison - Woodbury.

Moreover, the DFP update is known as a method of updating, of rank 2, that is, $H_{k+1}$ is formed by adding the matrix $H_k$ with two symmetric matrices, each of which is rank 1:

$$H_{k+1} = H_k + a u_k u_k^T + b v_k v_k^T,$$

where $u_k, v_k \in \mathbb{R}^n$, and $a, b$ are scalars. From the quasi-Newton condition (3.5) it follows that

$$H_k y_k + a u_k u_k^T y_k + b v_k v_k^T y_k = s_k.$$  

(3.11)

The vectors $u_k$ and $v_k$ are not unique, but they can obviously be determined in the following way

$$u_k = s_k, \quad v_k = H_k y_k.$$  

Now, (3.11) implies

$$a = \frac{1}{u_k^T y_k} = \frac{1}{s_k^T y_k}, \quad b = -\frac{1}{v_k^T y_k} = -\frac{1}{y_k^T H_k y_k}.$$  

So we get

$$H_{k+1}^{\text{DFP}} = H_k + \frac{s_k s_k^T}{s_k^T y_k} - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k}.$$  

(3.12)

Formula (3.12) was proposed by Davidon and later was developed by Fletcher and Powell, so that it is called DFP update.

3.3. BFGS update. One famous broadly used updating formula is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) rule. The inverse Hessian update $H_{k+1}$ is defined as the solution to

$$\min_H \| H - H_k \|, \quad \text{s.t. } H = H^T, \ H y_k = s_k.$$  

(3.13)

Certainly, the BFGS update is overtly known as

$$H_{k+1}^{\text{BFGS}} = (I - \rho_k s_k y_k^T) H_k^{\text{BFGS}} (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T, \quad \rho_k = \frac{1}{y_k s_k}.$$  

(3.14)
where \( s_k \) and \( y_k \) are defined in (3.2).

The update BFGS formula for the Hessian matrix can be generated using the Sherman-Morrison-Woodbury formula. A rank-one-modification (or perturbation) \( M = A + bc^* \) of a matrix \( A \in \mathbb{C}^{m \times n} \) uses two vectors \( b \in \mathbb{C}^{m \times 1} \) and \( c \in \mathbb{C}^{n \times 1} \). The Sherman-Morrison formula establishes a relationship between \( M^{-1} \) and \( A^{-1} \) as follows [45]:

\[
M^{-1} = A^{-1} - (1 + c^* A^{-1} b)^{-1} A^{-1} bc^* A^{-1}.
\] (3.15)

As a result, the following update for \( B_k \) is obtained:

\[
B_{k+1}^{\text{BFGS}} = B_k^{\text{BFGS}} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T y_k}.
\] (3.16)

3.4. Broyden family of methods. The weighted combinations of DFP and BFGS updates give the whole update class, which is known as the Broyden class. This class of update is defined by

\[
H_{k+1}^\phi = (1 - \phi) H_{k+1}^{\text{DFP}} + \phi H_{k+1}^{\text{BFGS}},
\] (3.17)

where \( \phi \) is a real parameter. If \( \phi \in [0, 1] \), then (3.17) is called the Broadden convex update class. It is obvious that Broyden’s class (3.17) satisfies the quasi-Newton equation (3.4). Also, the expression (3.17) can be rewritten in the following form

\[
H_{k+1}^\phi = H_{k+1}^{\text{DFP}} + \phi y_k y_k^T v_k
\] (3.18)

\[
= H_{k+1}^{\text{BFGS}} + (\phi - 1) v_k v_k^T
\]

\[
= H_k + \frac{s_k s_k^T}{s_k^T y_k} - \frac{H_k y_k y_k^T H_k}{y_k^T y_k} + \phi y_k y_k^T v_k,
\]

where

\[
v_k = (y_k^T H_k y_k)^{1/2} \left[ \frac{s_k}{s_k^T y_k} - \frac{H_k y_k}{y_k^T H_k y_k} \right].
\] (3.19)

If we put in (3.18):

- \( \phi = 0 \), then we will obtain DFP update (3.12)
- \( \phi = 1 \), then we will obtain BFGS update (3.14)
- \( \phi = \frac{s_k^T y_k}{s_k^T y_k - h_k y_k^T y_k} \), then we will obtain SR1 update (3.9).

The Broyden class of methods can be derived directly from the quasi-Newton equation. Consider the general formula for updating rank 2, which contains \( s_k \) and \( H_k y_k \):

\[
H_{k+1} = H_k + a s_k s_k^T + b (H_k y_k s_k^T + s_k y_k^T H_k) + c H_k y_k y_k^T H_k,
\] (3.20)

where \( a, b, c \) are unknown scalars. We obtain

\[
1 = a s_k^T y_k + b y_k^T H_k y_k,
\]

\[
0 = 1 + c s_k^T y_k + c y_k^T H_k y_k.
\] (3.21)

Here we have two equations with three unknowns, so we can introduce the replacement

\[
b = -\phi / s_k^T y_k,
\]

where \( \phi \) is a parameter. Solving system (3.21) and substituting the obtained result in (3.20), we obtain

\[
H_{k+1}^\phi = H_k + \frac{s_k s_k^T}{s_k^T y_k} - \frac{H_k y_k y_k^T H_k}{y_k^T y_k H_k} + \phi y_k y_k^T v_k = H_{k+1}^{\text{DFP}} + \phi y_k y_k^T v_k.
\]
where $v_k$ is defined by means of (3.19). Previous expression is identical to (3.18).

3.5. Some modifications of quasi-Newton methods. A great effort has been invested to discover QN methods that do not merely possess convergence but it is also better from the BFGS update in the numerical performance. Table 1 shows some of these modifications of the quasi-Newton equations.

**Table 1. Some modifications of quasi-Newton equations.**

<table>
<thead>
<tr>
<th>Quasi-Newton Eqs. $y_{k-1}$</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B_k s_{k-1} = y_{k-1}$</td>
<td>$y_{k-1} = \varphi_{k-1} y_{k-1} + (1 - \varphi_{k-1}) B_{k-1} s_{k-1}$</td>
</tr>
<tr>
<td>$B_k s_{k-1} = \tilde{y}_{k-1}$</td>
<td>$\tilde{y}<em>{k-1} = y</em>{k-1} + \frac{r(f_k)}{|y_{k-1}|^2} s_{k-1}$</td>
</tr>
<tr>
<td>$B_k s_{k-1} = \tilde{y}_{k-1}$</td>
<td>$\tilde{y}<em>{k-1} = y</em>{k-1} + \frac{2r(f_k)}{|y_{k-1}|^2} s_{k-1}$</td>
</tr>
<tr>
<td>$B_k s_{k-1} = \hat{y}_{k-1}$</td>
<td>$\hat{y}<em>{k-1} = y</em>{k-1} + \frac{\max(0, 2r(f_k))}{|y_{k-1}|^2} s_{k-1}$</td>
</tr>
<tr>
<td>$B_k s_{k-1} = \hat{y}_{k-1}$</td>
<td>$\hat{y}<em>{k-1} = y</em>{k-1} + \frac{\max(0, 2r(f_k))}{|y_{k-1}|^2} s_{k-1}$</td>
</tr>
<tr>
<td>$B_k s_{k-1} = \hat{y}_{k-1}$</td>
<td>$\hat{y}<em>{k-1} = \frac{1}{3} y</em>{k-1} + \frac{(f_{k-1} - f_k) - 2g^T s_{k-1}}{s_{k-1}^T y_{k-1}} y_{k-1}$</td>
</tr>
</tbody>
</table>

Also, it is important to state spectral gradient method. Therein the updating of the formula for $B_{k+1}$ is done in the following way [111]

$$B_{k+1} = \text{diag}(r_k^{(i)})$$

(3.22)

with

$$r_k^{(i)} = \frac{1}{\sqrt{\sum_{i=1}^n (s_k^{(i)})^2 + \sum_{i=1}^n (y_k^{(i)}) (s_k^{(i)}) (\hat{s}_k^{(i)})^2)}}$$

$$\hat{s}_k = s_k - \frac{7}{11} s_{k-1} + \frac{2}{11} s_{k-2},$$

$$\hat{y}_k = y_k - \frac{7}{11} y_{k-1} + \frac{2}{11} y_{k-2}.$$  

The general framework of the QN algorithm is given in Algorithm 3.

**Algorithm 3** General framework of the quasi-Newton algorithm

**Require:** Objective $f(x)$, initial point $x_0 \in \mathbb{R}^n$, initial inverse Hessian approximation $H_0 \in \mathbb{R}^{n \times n}$, and the tolerance $0 < \varepsilon \ll 1$.

1. $k := 0$.
2. while $\|g_k\| > \varepsilon$ do

3. Compute

$$d_k = -H_k g_k.$$  

(3.23)

4. Compute $\alpha_k > 0$ using exact or inexact line search.

5. Compute $x_{k+1} := x_k + \alpha_k d_k$.

6. Update $H_k$ into $H_{k+1}$, such that (3.5) holds.

7. $k := k + 1$.

8. end while

**Ensure:** $x_{k+1}, f(x_{k+1})$. 

3.6. QN methods based on constant diagonal matrix approximation. The general QN iterative rule with line search

\[ x_{k+1} = x_k - \alpha_k H_k g_k \]  

assumes that \( B_k \) (resp. \( H_k = B_k^{-1} \)) is a positive definite approximation of \( G_k \) (resp. \( G_k^{-1} \)) [118]. The update \( B_{k+1} \) of \( B_k \) is defined on the basis of the quasi-Newton property (3.4) or (3.5).

According to Brezinski’s classification in [15], the structure of updating \( B_k \) can be divided into three categories: scalar matrix \( B_k = \lambda_k I \), diagonal matrix \( B_k = \text{diag} \ (\lambda_1, \ldots, \lambda_n) \) and an appropriate full matrix \( B_k \). Optimization methods included in this class of iterations are based on simplest approximation of the Hessian and its inverse as

\[ B_k = \gamma_k I \approx G_k, \quad \gamma_k > 0, \]  

where \( I \) is a proper \( n \times n \) identity matrix and and \( \gamma_k > 0 \) is a parameter. Such choice leads to the iterative rule

\[ x_{k+1} = x_k - \gamma_k^{-1} \alpha_k g_k. \]  

Usually, the parameter \( \alpha_k \) is defined using an available ILS, and \( \gamma_k \) is defined according to the Taylor’s development of \( f(x_k) \). The iterations (3.26) are termed as improved gradient descent (IGD) methods in [63].

Andrei in [4, 6] defined iterations

\[ x_{k+1} = x_k - \theta_k \alpha_k g_k. \]  

Usage of random values of \( \theta_k \) was proposed in [6]. Later, Andrei in [4] proposed appropriate algorithm for defining \( \theta_k \) in (3.27). The iterative rule (3.27) was called in [4] as Accelerated Gradient Descent (AGD):

\[ x_{k+1}^{AGD} = x_k^{AGD} - \theta_k^{AGD} \alpha_k g_k^{AGD}. \]  

A few modifications of the scheme (3.26) were promoted in [94, 96, 97, 112, 114]. The iterations defined in [112] are of the form (3.26), in which \( \gamma_k I \) is the Hessian approximation, where \( \gamma_k = \gamma(x_k, x_{k-1}) > 0 \) is the parameter. The SM method from [112] was defined by the iterations

\[ x_{k+1}^{SM} = x_k^{SM} - \alpha_k (\gamma_k^{SM})^{-1} g_k^{SM}, \]  

where \( \gamma_k^{SM} > 0 \) is the acceleration parameter defined using the Taylor’s development of the objective \( f \) at the point \( x_{k+1}^{SM} \), as follows:

\[ \gamma_k^{SM} = 2\gamma_k^{SM} [f(x_{k+1}^{SM}) - f(x_k^{SM})] + \alpha_k \|g_k^{SM}\|^2. \]  

The Double direction and double step-size accelerated methods, termed as ADSS and ADD, respectively, were originated in [94, 96].

The next iterations are known as Accelerated double step-size (ADSS) iterations [94]:

\[ x_{k+1}^{ADSS} = x_k^{ADSS} - (\alpha_k (\gamma_k^{ADSS})^{-1} + l_k) g_k^{ADSS}, \]  

where \( \alpha_k \) and \( l_k \) are step-sizes, derived by two independent backtracking procedures. The TADSS method from [114] is proposed using the assumption \( \alpha_k + l_k = 1 \), which gives

\[ x_{k+1}^{TADSS} = x_k^{TADSS} - \psi_k g_k^{TADSS}, \]  

where \( \psi_k = \alpha_k (\gamma_k^{TADSS})^{-1} - 1 \). An application of the TADSS iterations in aviation industry was investigated in [68].
The particular choice $\gamma_k = 1$ transforms the IGD iterations (3.26) into the GD iterative rule (1.5). Further, the IGD iterations (3.26) in the case $\alpha_k = 1$ can be viewed as the GD iterations
\[ x_{k+1} = x_k - \gamma_k^{-1} g_k, \] (3.32)
where $\gamma_k$ becomes the primary step length which should be appropriately defined.
Barzilai and Borwein in [11] originated two well known variants of the GD method, known as BB method, with the step length $\gamma_{BB}^k := \gamma_k^{-1}$ in (3.32). The step length $\gamma_{BB}^k$ in the first case is defined by the vector minimization $\min_{\gamma} \|s_{k-1} - \gamma y_{k-1}\|^2$, which yields
\[ \gamma_{BB}^k = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}}. \] (3.33)
The symmetric case assumes the minimization $\|\gamma s_{k-1} - y_{k-1}\|^2$, which produces
\[ \gamma_{BB}^k = \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T y_{k-1}}. \] (3.34)
The BB iterations are defined using $\gamma_{BB}^k$ as follows:
\[ x_{BB}^{k+1} = x_{BB}^k - \gamma_{BB}^k g_{BB}^k. \]
The BB method was improved in a number of research articles, main of which are [22, 24, 34, 35, 36, 37, 106, 107, 122, 137].

Another member of the IGD iterates is the Scalar Correction (SC) method [84], defined in (3.32) by the rule
\[ \gamma_{SC}^{k+1} = \begin{cases} \frac{s_{k-1}^T r_k}{\|s_{k-1}\|} & y_{k-1}^T r_k > 0, \\ \frac{y_{k-1}^T r_k}{\|y_{k-1}\|} & y_{k-1}^T r_k \leq 0, \end{cases} \quad r_k = s_k - \gamma_k y_k. \] (3.35)
Accordingly, the SC iterations are defined by the relation
\[ x_{SC}^{k+1} = x_{SC}^k - \gamma_{SC}^k g_{SC}^k. \]

Relaxed BB method by an additional step $\theta_k \in (0, 2)$ is proposed in [105].

A modification of GD method (1.5) was proposed in [63]. It is defined by $MGD = M(GD)$ with
\[ x_{k+1} = M(GD)(x_k) = x_k - (\alpha_k + \alpha_k^2 - \alpha_k^3) g_k. \] (3.36)
Further, the next scheme was proposed as the modified SM (MSM) method in [63]:
\[ x_{MSM}^{k+1} = x_{k}^{MSM} - (\alpha_k + \alpha_k^2 - \alpha_k^3)(\gamma_{MSM}^k)^{-1} g_{MSM}^k. \] (3.37)
The leading principle used in defining the iterations (3.37) is the replacement of $\alpha_k$ in the GD methods (1.5) by the slightly longer step $\alpha_{MSM}^k = \alpha_k + \alpha_k^2 - \alpha_k^3$. The underlying idea in defining $\alpha_{MSM}^k$ is the observation $\alpha_{MSM}^k > \alpha_k$, which means that MSM method proposes a slightly longer step with the aim to additionally accelerate the method. As before, $\alpha_k \in (0, 1)$ is defined by Algorithm 2. The rationale of this modification lies in the inequalities
\[ \alpha_k \leq \alpha_k + \alpha_k^2 - \alpha_k^3 \leq \alpha_k + \alpha_k^2. \]
So, (3.37) is based on a small increase of $\alpha_k$ inside the interval $[\alpha_k, \alpha_k + \alpha_k^2]$. 
The efficiency of IGD methods was numerically tested in [98]. The real number Υ ∈ (0, 1) from (3.39) is termed as the correction parameter in [97]. Instead of (3.39), it suffices to use an equivalent iteration given by

\[ x_{k+1} = H(T)(x_k) = T[(1 - \Upsilon_k)x_k + \Upsilon_kTx_k], \quad k \in \mathbb{N}. \]  

(3.40)

3.7. Gradient methods accelerated by Picard-Mann hybrid iterative process. An application of the Picard-Mann hybrid iterative process from [66] is another possibility to accelerate iterations for solving nonlinear optimization problems. The function \( T : C \to C \) in (3.40) is defined on a convex subset \( C \) of a normed space \( E \). The hybrid iterations define two sequences \( x_k, y_k \) by the rules:

\[
\begin{cases}
  x_1 = x \in C, \\
  x_{k+1} = Ty_k, \\
  y_k = (1 - \Upsilon_k)x_k + \Upsilon_kTx_k, \quad k \in \mathbb{N}.
\end{cases}
\]  

(3.39)

The following algorithm is known as the SM method and introduced in [112].

**Algorithm 4** The SM method.

**Require:** Objective function \( f(x) \) and chosen initial point \( x_0 \in \text{dom}(f) \).

1. Set \( k = 0 \) and compute \( f(x_0), g_0 = \nabla f(x_0) \) and take \( \gamma_0 = 1 \).
2. If stopping criteria are satisfied, then STOP; otherwise, go to the next step.
3. Find the step-size \( \alpha_k \in (0, 1) \) using Algorithm 2 with \( d_k = -g_k^{-1}\).
4. Compute \( x_{k+1} = x_k - \xi_k\alpha_k\), \( f(x_{k+1}) \) and \( g_{k+1} = \nabla f(x_{k+1}) \).
5. Define \( \gamma_{k+1} \) using (3.30).
6. If \( \gamma_{k+1} < 0 \), then put \( \gamma_{k+1} = 1 \).
7. Set \( k := k + 1 \), go to Step 2.
8. Return \( x_{k+1} \) and \( f(x_{k+1}) \).
The iterates \((3.40)\) are denoted by \(H(T, x_k) = H(T)(x_k)\).

In [66] it was proposed a set of constant values \(\alpha = \Upsilon_k \in (0, 1), \forall k\) in numerical experiments and concluded that the process \((3.40)\) converges faster than the Picard, Mann and Ishikawa iterations from [62, 83, 101]. Further, \((3.40)\) was applied in [97] for a hybridization of the SM method, known as HSM. Using the mapping \(T\) in \((3.39)\) or \((3.40)\) to coincide with the SM rule \((3.29)\):

\[ T(x_k) := x_{SM}^k - (\Upsilon_{SM} k + 1)^{-1} \alpha_k g_{SM}^k, \]

the iterations \((3.40)\) become the so called HSM iterative rule given as

\[ x_{HSM}^{k+1} := H(SM)(x_k) = x_{HSM}^k - (\Upsilon_{HSM} k + 1)^{-1} \alpha_k g_{HSM}^k, \tag{3.41} \]

where \(\Upsilon_{HSM} > 0\) is the acceleration defined by

\[ \Upsilon_{HSM} = 2 \gamma_{HSM} \frac{f(x_{HSM}^{k+1}) - f(x_{HSM}^k)}{(\Upsilon_{HSM} k + 1)^2 t_k^2 \| g_{HSM}^k \|^2} \cdot \]

A modified HSM (MHSM) method is defined in [92] by proposing an appropriate initial value in the backtracking procedure.

A hybridization of the ADD method was considered in [99] in the form

\[ x_{HADD}^{k+1} = x_{HADD}^k - (\Upsilon_{HADD} + 1) t_k (\gamma_{HADD})^{-1} g_{HADD}^k + (\Upsilon_{HADD} + 1) t_k^2 d_k, \]

wherein

\[ \gamma_{HADD} = 2 \left( \frac{f(x_{HADD}^{k+1}) - f(x_{HADD}^k)}{(\Upsilon_{HADD} + 1)^2 t_k^2 \| g_{HADD}^k \|^2} \right) \cdot \]

Recently, the hybridization HTADSS \(= H(TADSS)\) was proposed, investigated and tested in [95].

4. Overview of conjugate gradients methods. Nonlinear conjugate gradient (CG) methods form a class of important methods for solving unconstrained nonlinear optimization and solving system of nonlinear equations. Nonlinear CG methods are defined by the line search iterates \((1.2)\) where the search direction \(d_k\) is defined by \((1.10)\) and the CGUP \(\beta_k\) is given using one of many available rules.

In this article, a review on CG methods for unconstrained optimization is given. Main convergence theorems are provided for the conjugate gradient method assuming the descent property of each search direction. Some research issues on conjugate gradient methods are mentioned.

In [21], the nonlinear CG methods are divided into three classes: early conjugate gradient methods, descent conjugate gradient methods, and sufficient descent conjugate gradient methods. Andrei classified the CG methods in three classes: classical CG methods, scaled CG methods, and hybrid and parameterized CG methods.

The classification presented in this paper divides CG methods into the following categories: basic conjugate gradients methods, considered in Subsection 4.1, Dai-Liao class of methods, presented in Subsection 4.2, hybrid conjugate gradient methods, described in Subsection 4.3, and combined BFGS-CG iterations, in Subsection 4.4.
TABLE 2. Some modifications of quasi-Newton equations.

<table>
<thead>
<tr>
<th>$\beta_k$</th>
<th>Title</th>
<th>Year</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta_k^{HS}$</td>
<td>$\frac{y_{k-1}^T g_k}{y_{k-1}^T d_{k-1}}$</td>
<td>1952</td>
<td>[60]</td>
</tr>
<tr>
<td>$\beta_k^{FR}$</td>
<td>$\frac{g_k}{</td>
<td></td>
<td>g_k</td>
</tr>
<tr>
<td>$\beta_k^T$</td>
<td>$\frac{g_k^T G_{k-1} d_{k-1}}{d_{k-1}^T G_{k-1} d_{k-1}}$</td>
<td>1967</td>
<td>[38]</td>
</tr>
<tr>
<td>$\beta_k^{PRP}$</td>
<td>$\frac{y_{k-1}^T g_k}{</td>
<td></td>
<td>g_k</td>
</tr>
<tr>
<td>$\beta_k^{CD}$</td>
<td>$\frac{</td>
<td></td>
<td>y_k^T g_k</td>
</tr>
<tr>
<td>$\beta_k^{LS}$</td>
<td>$\frac{y_{k-1}^T g_k}{g_{k-1}^T d_{k-1}}$</td>
<td>1991</td>
<td>[79]</td>
</tr>
<tr>
<td>$\beta_k^{DY}$</td>
<td>$\frac{</td>
<td></td>
<td>y_k^T g_k</td>
</tr>
</tbody>
</table>

4.1. Basic conjugate gradients methods. The CG methods included in Table 2 are known as early or classical conjugate gradient methods.

where $y_{k-1} = g_k - g_{k-1}$, $s_{k-1} = x_k - x_{k-1}$, $G_{k-1} = \Delta^2 f(x_{k-1})$ and $|| \cdot ||$ stands for the Euclidean vector norm.

In the listed CG methods, the numerator of the update parameter $\beta_k$ is either $||g_k||^2$ or $y_{k-1}^T g_k$ and the denominator is either $||g_{k-1}||^2$ or $y_{k-1}^T d_{k-1}$ or $-g_{k-1}^T d_{k-1}$. Two possible choices for the numerator and the 3 possible choices for the denominator lead to 6 different choices for $\beta_k$.

TABLE 3. Classification of CG methods.

<table>
<thead>
<tr>
<th>Denominator</th>
<th>Numerator</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y_{k-1}^T d_{k-1}$</td>
<td>$</td>
</tr>
<tr>
<td>$y_{k-1}^T g_k$</td>
<td>$</td>
</tr>
</tbody>
</table>

Define the following functions

$n_1 := ||g_k||^2$, $n_2 = y_{k-1}^T g_k$, $d_1 := ||g_{k-1}||^2$, $d_2 = y_{k-1}^T d_{k-1}$, $d_3 = -g_{k-1}^T d_{k-1}$.

Then

$\beta_k^{FR} = \frac{n_1}{d_1}$, $\beta_k^{PRP} = \frac{n_2}{d_1}$, $\beta_k^{DY} = \frac{n_1}{d_2}$, $\beta_k^{HS} = \frac{n_2}{d_2}$, $\beta_k^{CD} = \frac{n_1}{d_3}$, $\beta_k^{LS} = \frac{n_2}{d_3}$.

But, there exist exceptions to these rules. One example is given in [38]

$$\beta_k^D = \frac{g_k^T G_{k-1} d_{k-1}}{d_{k-1}^T G_{k-1} d_{k-1}} (1967).$$

From the presented chronological development of the CGUP, we can see that the $\beta_k^D$ choice of the CG parameter differs structurally from the other choices.

For a large-scale unconstrained nonlinear optimization problem, in practice, choices for updating a CG parameter that do not require computation or approximation of the Hessian and its inverse are preferred over methods that require Hessian or its approximation in each iteration.
Wei et al. [124] gave a variant of the PRP method which we call the VPRP method, with the parameter $\beta_k$:

$$\beta_k^{\text{VPRP}} = \frac{\|g_k\|^2 - \|g_k\| g_k^T g_{k-1}}{\|g_{k-1}\|^2}.$$  

The VPRP method was extended to a variant of the HS method by Yao et al. in [132],

$$\beta_k^{\text{VHS}} = \frac{\|g_k\|^2 - \|g_k\| g_k^T g_{k-1}}{d_{k-1}^T y_{k-1}}.$$  

Zhang [138] took a little modification to the VPRP method and constructed the NPRP method as follows,

$$\beta_k^{\text{NPRP}} = \frac{\|g_k\|^2 - \|g_k\| g_k^T g_{k-1}}{\|g_{k-1}\|^2}.$$  

Moreover, Zhang [138] extended this result to the HS method and proposed the NHS method as follows,

$$\beta_k^{\text{NHS}} = \frac{\|g_k\|^2 - \|g_k\| g_k^T g_{k-1}}{d_{k-1}^T y_{k-1}}.$$  

Recently, Wei et al. [125] proposed a variation of the FR method which we call the VFR method. the parameter $\beta_k$ in the VFR method is given by

$$\beta_k^{\text{VFR}} = \frac{\mu_1 \|g_k\|^2}{\mu_2 \|g_k\| d_{k-1} + \mu_3 \|g_{k-1}\|^2},$$

where $\mu_1 \in (0, +\infty)$, $\mu_2 \in (\mu_1 + \epsilon_1, +\infty)$, $\mu_3 \in (0, +\infty)$ and $\epsilon_1$ is any given positive constant. Motivated by these modifications, in [29] the authors defined the modified PRP method as

$$\beta_k^{\text{DPRP}} = \frac{\|g_k\|^2 - \|g_k\| g_k^T g_{k-1}}{\|g_{k-1}\|^2}.$$  

Recently, Wei et al. [18] gave a variant of the PRP method which we call the WYL method, that is,

$$\beta_k^{\text{WYL}} = \frac{g_k^T \left( g_k - \|g_k\| g_{k-1} \right)}{\|g_{k-1}\|^2}.$$  

The WYL method was extended to a variant of the LS method by Yao et al. [132], that is,

$$\beta_k^{\text{MLS}} = \frac{g_k^T \left( g_k - \|g_k\| g_{k-1} \right)}{-g_k^T d_{k-1}}.$$  

Also, the following function will be useful:

$$\mathcal{N}_1 = g_k^T \left( g_k - \|g_k\| g_{k-1} \right) = n_1 - \|g_k\| g_k^T g_{k-1} = g_k^T y_{k-1}$$  

$$\mathcal{N}_2 = n_1 - \|g_k\| \|g_{k-1}\|^2.$$
Dai-Liao method and its variants.

Expression (4.7) for defining $\beta_k$ was studied by Perry [93]. Perry tried to incorporate the second-order information of the objective function into the CG method to accelerate it. Specifically, by replacing the conjugacy condition (4.2) by the following condition:

$$\|g_k\|^2 - \|g_{k-1}\|^2 + d_{k-1}^T g_k (4.1)$$

Some particular CG variants are $\beta_k^{\text{DHS}}$ [29] and $\beta_k^{\text{DLS}}$ [139], defined by

$$\beta_k^{\text{DHS}} = \frac{\|g_k\|^2 - \|g_{k-1}\|^2}{\mu |g_k^T d_{k-1}| + d_{k-1}^T y_{k-1}} \quad (2012)$$

$$\beta_k^{\text{DLS}} = \frac{\|g_k\|^2 - \|g_{k-1}\|^2}{\mu |g_k^T d_{k-1}| - d_{k-1}^T g_k} \quad (4.2)$$

If the functions $\mathcal{D}_1(\mu) = \mu |g_k^T d_{k-1}| + d_{k-1}^T y_{k-1}$ and $\mathcal{D}_2(\mu) = \mu |g_k^T d_{k-1}| - d_{k-1}^T g_k$ are defined, then

$$\beta_k^{\text{DHS}} = \frac{\mathcal{D}_2}{\mathcal{D}_1(\mu)}, \quad \beta_k^{\text{DLS}} = \frac{\mathcal{D}_2}{\mathcal{D}_2(\mu)}.$$

4.2. Dai-Liao method and its variants. An extension of the conjugacy condition

$$d_k^T y_{k-1} = 0 \quad (4.2)$$

was studied by Perry [93]. Perry tried to incorporate the second-order information of the objective function into the CG method to accelerate it. Specifically, by using the secant condition and the search direction of the QN methods, which are respectively defined by

$$B_k s_{k-1} = y_{k-1} \quad \text{and} \quad B_k d_k = -g_k,$$

the following relation is obtained:

$$d_k^T y_{k-1} = d_k^T (B_k s_{k-1}) = (B_k d_k)^T s_{k-1} = -g_k^T s_{k-1} \quad (4.4)$$

where $B_k$ is a symmetric approximation to the Hessian matrix $G_k$. Then Perry accordingly replaced the conjugacy condition (4.2) by the following condition

$$d_k^T y_{k-1} = -g_k^T s_{k-1}. \quad (4.5)$$

Furthermore, Dai and Liao [26] included a nonnegative parameter $t$ into Perry’s condition and gave

$$d_k^T y_{k-1} = -t g_k^T s_{k-1}. \quad (4.6)$$

In order to find the search direction $d_k$ in (1.10) which satisfies the conjugacy condition (4.6), it suffices to multiply (1.10) by $y_{k-1}$ and use (4.6), yielding

$$\beta_k^{\text{DL}} = \frac{g_k^T y_{k-1} - t g_k^T s_{k-1}}{d_k^T y_{k-1}} = \frac{g_k^T (y_{k-1} - ts_{k-1})}{d_k^T y_{k-1}} \quad (4.7)$$

Expression (4.7) for defining $\beta_k$ characterizes the Dai and Liao (DL) CG method. Later, motivated by the DL method, researchers in papers [18, 80, 100, 129, 131, 139, 141], suggested modified variants of the DL method. Some well-known formulas for $\beta_k$ were created modifying the CG parameter $\beta_k^{\text{DL}}$ [18, 26, 80, 100, 129, 131, 139].
Main of them are $\beta_k^{\text{DL}}$ \cite{26}, $\beta_k^{\text{DHS SDL}}$ \cite{139}, $\beta_k^{\text{DL SDL}}$ \cite{139}, $\beta_k^{\text{MHSDL}}$ \cite{131}, defined as

$$
\beta_k^{\text{DL}} = \frac{y_k^T s_{k-1}}{y_k^T d_{k-1}} - t \frac{g_k^T s_{k-1}}{d_{k-1}^T y_{k-1}}
$$

where $t > 0$ is a scalar and $\hat{y}_{k-1} = g_k - \frac{\|g_k\|}{\|s_{k-1}\|} s_{k-1}$.

Clearly $\beta_k^{\text{DL}}$ with $t > 0$ defines a class of nonlinear CG methods. Moreover, in the case of the exact line search, i.e., $g_k^T s_{k-1} = 0$, then $\beta_k^{\text{DL}} = \beta_k^{\text{HS}}$.

Some additional CG methods from the DL class are $\beta_k^{\text{MLSDL}}$ \cite{18} and $\beta_k^{\text{ZZDL}}$ \cite{141}, defined as follows:

$$
\beta_k^{\text{MLSDL}} = \frac{g_k^T y_{k-1}}{z_k^T d_{k-1}} - t \frac{g_k^T s_{k-1}}{d_{k-1}^T y_{k-1}}
$$

where $t > 0$ is a scalar, $z_{k-1} = y_{k-1} + C\|s_{k-1}\|s_{k-1}$ and $\hat{y}_{k-1} = g_k - \frac{\|g_k\|}{\|s_{k-1}\|} s_{k-1}$.

In order to characterize this family of CG methods, define the mapping

$$
\bar{f}(\beta_k^M, t) = \beta_k^M - t \frac{g_k^T s_{k-1}}{d_{k-1}^T y_{k-1}}.
$$

Then

$$
\beta_k^{\text{DL}} = \bar{f}(\beta_k^{\text{HS}}, t), \quad \beta_k^{\text{DHS SDL}} = \bar{f}(\beta_k^{\text{DHS}}, t), \quad \beta_k^{\text{DL SDL}} = \bar{f}(\beta_k^{\text{DLS}}, t), \quad \beta_k^{\text{MHSDL}} = \bar{f}(\beta_k^{\text{DHS}}, t).
$$

The research for the best values of the parameter $t$ was divided into two directions. One direction was to find the best fixed value for $t$ and the other the best approximation for $t$ in each iteration. Analyzing the results from \cite{18, 26, 131, 139}, we conclude that the scalar $t$ was defined by a fixed value of 0.1 in numerical experiments. Also, numerical experience related to the fixed valued $t = 1$ was reported in \cite{26}. Common numerical experience is that different choices of $t$ initiate totally
A SURVEY OF GRADIENT METHODS FOR NONLINEAR OPTIMIZATION 1591
different numerical experience. This was the reason for further research to focus on the values of $t$ that change through iterations. The value of $t$ in arbitrary $k$th iteration will be denoted by $t_k$. Hager and Zhang in [55] defined $t_k$ by the rule

$$t_k = \frac{2 \| y_{k-1} \|^2}{y_{k-1}^T s_{k-1}}.$$  

Babaie-Kafaki and Ghanbari [9] presented two appropriate choices of the parameter $t$ in (4.7):

$$t_k = \frac{y_{k-1}^T}{\| s_{k-1} \|} s_{k-1} + \frac{y_{k-1}^T}{\| s_{k-1} \|} s_{k-1}$$

and

$$t_k = \frac{\| y_{k-1} \| }{\| s_{k-1} \|}.$$  

Andrei in [5] suggested the following value for $t$ in (4.7) which becomes a variant of the DL method, denoted by DLE:

$$t_k = \frac{y_{k-1}^T}{\| s_{k-1} \|} s_{k-1}.$$  

Lotfi and Hosseini in [81] discovered the most recent approximations of the parameter $t_k$.

4.3. Hybrid conjugate gradient methods. In the subsequent sections, we will survey recent advances in CG methods. Two main research streams can be observed: the algorithms which improve the scalar parameter $t$ and the algorithms which improve the CG parameter $\beta_k$.

Hybrid CG methods can be segmented into two classes: mixed methods as well as methods combined together by introducing one or more parameters. The following hybrid CG method was suggested in [121]:

$$\beta_k = \max \{ 0, \min \{ \beta_{\text{PRP}}, \beta_{\text{FR}} \} \}.$$  

When a jam in iterations occurs again, the PRP update parameter is used. Hu and Storey in [61] had a similar motivation and suggested the following rule

$$\beta_k = \max \{ 0, \min \{ \beta_{\text{PRP}}, \beta_{\text{FR}} \} \}. $$

In [49] it is pointed out that $\beta_{\text{PRP}}$ can be negative, even for strongly convex functions. In an effort to extend the allowed choices for the PRP update parameter, while retaining global convergence, Nocedal and Gilbert [49] suggested the choice

$$\beta_k = \max \{ -\beta_{\text{FR}}, \min \{ \beta_{\text{PRP}}, \beta_{\text{FR}} \} \}. $$

Dai and Yuan [31] combined the DY method with other CG methods, which leads to the following CGUP parameters:

$$\beta_k = \max \{ 0, \min \{ \beta_{\text{HS}}, \beta_{\text{DY}} \} \}$$

and

$$\beta_k = \max \{ -c \beta_{\text{DY}}, \min \{ \beta_{\text{HS}}, \beta_{\text{DY}} \} \}, \quad c = \frac{1 - \sigma}{1 + \sigma}. $$

In [28], they tested different CG methods for large-scale unconstrained optimization problems and concluded that the hybrid CG method (4.18) gave the best results.
Next hybrid CG method, proposed by Dai in [23], employs either the DY scheme or the CD scheme:

$$
\beta_k = \frac{\|g_k\|^2}{\max\{d_{k-1}^T y_{k-1}, -g_{k-1}^T d_{k-1}\}}. \tag{4.20}
$$

A modified CG method defined as the hybridization of known LS and CD conjugate gradient methods is presented and analyzed in [130] by the rule

$$
\beta_k^{LSCD} = \max\{0, \min\{\beta_k^{LS}, \beta_k^{CD}\}\}. \tag{4.21}
$$

CG methods can be combined together by introducing one or more parameters. In [32, 33], Dai and Yuan proposed a one-parameter family of CG methods with

$$
\beta_k = \frac{\|g_k\|^2}{\theta_k \|g_{k-1}\|^2 + (1 - \theta_k) d_{k-1}^T y_{k-1}}, \tag{4.22}
$$

where $\theta_k \in [0, 1]$ is a parameter. Note that $\beta_k = \beta_k^{FR}$ in the case $\theta_k = 1$, and $\beta_k = \beta_k^{DY}$ if $\theta_k = 0$.

Another hybrid method, proposed by Delladji, Belloufi and Sellami in [39], exploits either the PRP scheme or the HZ scheme, as

$$
\beta_k^{PRPHZ} = \theta_k \beta_k^{PRP} + (1 - \theta_k) \beta_k^{HZ}
$$

in which $\theta_k \in [0, 1]$ is called the hybridization parameter. Note that if $\theta_k = 1$ then $\beta_k^{PRPHZ} = \beta_k^{PRP}$, and if $\theta_k = 0$ then $\beta_k^{PRPHZ} = \beta_k^{HZ}$.

Nazareth in [89] proposed a two-parameter family of CGUP parameters using convex combinations of numerators and denominators, as

$$
\beta_k = \frac{\nu_k \|g_k\|^2 + (1 - \nu_k) g_{k-1}^T y_{k-1}}{\theta_k \|g_{k-1}\|^2 + (1 - \theta_k) d_{k-1}^T y_{k-1}}, \tag{4.24}
$$

where $\nu_k, \theta_k \in [0, 1]$. This two-parameter family includes FR, DY, PRP, and HS methods as extreme cases.

In [113], the authors proposed hybrid CG methods where the search direction $d_k := \delta_k$, $k \geq 1$, is improved using one of the rules

$$
\begin{align*}
\delta_k := \mathcal{D}(\beta_k, g_k, d_{k-1}) &= \left(1 + \beta_k \frac{g_k^T d_{k-1}}{\|g_k\|^2}\right) g_k + \beta_k d_{k-1}, \tag{4.25} \\
\delta_k := \mathcal{D}_1(\beta_k, g_k, d_{k-1}) &= -B_k g_k + \mathcal{D}(\beta_k, g_k, d_{k-1}) \tag{4.26}
\end{align*}
$$

and $\beta_k$ is determined using appropriate combinations of $\beta_k$ used in Table 2 and/or previously defined hybridizations. In [113], the authors defined a modification of LSCD method, defined in [130] by

$$
\begin{align*}
\beta_k^{LSCD} &= \max\{0, \min\{\beta_k^{LS}, \beta_k^{CD}\}\}, \\
\mathcal{D}(\beta_k^{LSCD}, g_k, d_{k-1}) &= \mathcal{D}(\beta_k^{LSCD}, g_k, d_{k-1}). \tag{4.27}
\end{align*}
$$

The resulting method is known as the MLSCD method with the search direction

$$
\delta_k := \mathcal{D}(\beta_k^{LSCD}, g_k, d_{k-1}). \tag{4.28}
$$

In general, the idea is based on the replacement of $\delta_k = \mathcal{D}(\beta_k^{LSCD}, g_k, d_{k-1})$ from [130] by $\delta_k = \mathcal{D}(\beta_k^{LSCD}, g_k, d_{k-1})$.

Now we give the general framework of the CG class of methods.
Algorithm 5 Algorithm of CG methods.

Require: A starting point $x_0$, and real quantities $0 < \epsilon < 1$, $0 < \delta < 1$.

1: Put $k = 0$ and compute $d_0 = -g_0$.
2: If $\|g_k\| \leq \epsilon$ and $|f(x_{k+1}) - f(x_k)| \leq \delta$, STOP; else go to Step 3.
3: Compute $\alpha_k \in (0, 1]$ using Algorithm 2.
4: Compute $x_{k+1} = x_k + \alpha_k d_k$.
5: Calculate $g_{k+1}, y_k = g_{k+1} - g_k$ and go to Step 6.
6: Calculate $\beta_{k+1}$.
7: Compute the search direction $d_{k+1} = \mathcal{D}_1(\beta_{k+1}, g_{k+1}, d_k)$ or $d_{k+1} = \mathcal{D}(\beta_{k+1}, g_{k+1}, d_k)$.
8: Put $k = k + 1$, and go to Step 2.

The first iteration in CG methods is a gradient step. Also, it is common to restart the algorithm periodically by taking the gradient step.

4.4. Broyden-Fletcher-Goldfarb-Shanno conjugate gradient methods.

Known fact is that CG iterates are better than QN methods in terms of the CPU time. Moreover, BFGS updates require greater memory space usage than CG. On the other hand, the QN methods require lesser number of iterations as well as the number of function evaluations. For this purpose, one of the modern trends in defining new CG methods is usage of the BFGS update in defining new rules for defining $\beta_k$. A hybrid method which solves the system of nonlinear equations combining the QN method with chaos optimization was discovered in [82]. In [58], the authors defined a combination of a QN and the Cauchy descent method for solving unconstrained optimization problems, which is known as the quasi-Newton SD method.

A hybrid direction defined as a combination of the BFGS update of $B_k$ and the CGUP $\beta_k$ was considered in [10, 67]. The DFP-CG method was originated in [91]. A three-term hybrid BFGS-CG method (termed as H-BFGS-CG1) was proposed in [113] by the search direction

$$d_k := \begin{cases} -B_k g_k, & k = 0, \\ \mathcal{D}_1(\beta_k, g_k, d_{k-1}), & k \geq 1. \end{cases}$$

In [113], the authors investigated hybrid CG algorithms based on the modified search direction which is defined using one of the following two hybridizations:

$$d_k := \mathcal{D}(\beta_k, g_k, d_{k-1}) = - \left( 1 + \beta_k \frac{g_k^T d_{k-1}}{\|g_k\|^2} \right) g_k + \beta_k d_{k-1},$$

as well as on the usage of $\beta_k$ defined using convenient combinations of the parameters $\beta_k$ involved in Table 2 and previously defined hybridizations. The matrix $B_k$ in (4.30) is defined as an appropriate Hessian approximation by the BFGS update. A three-term BFGS-CG method, known as H-BFGS-CG1, was defined in [113] using

$$d_k := \begin{cases} -B_k g_k, & k = 0, \\ \mathcal{D}_1(\beta_k, g_k, d_{k-1}), & k \geq 1. \end{cases}$$
5. Convergence properties of CG methods. Below we give some assumptions related to line search procedures.

Assumption 5.1. (1) The level set $S = \{ x \in \mathbb{R}^n \mid f(x) \leq f(x_0) \}$ is bounded, where $x_0$ is an initial point of the iterative method (1.2).

(2) The objective function $f$ is continuous and differentiable in a neighborhood $N$ of $S$, and its gradient $g$ is Lipschitz continuous. So, there exists a positive constant $L > 0$, satisfying
\[ \| g(u) - g(v) \| \leq L \| u - v \|, \quad \forall u, v \in N. \] (5.1)

Assumption 5.1 initiates the existence of a positive constants $D$ and $\gamma$ satisfying
\[ \| u - v \| \leq D, \quad \forall u, v \in N \] (5.2)
and
\[ \| g(u) \| \leq \gamma, \quad \forall u \in N. \] (5.3)

The proof of Lemma 5.1 is given in [142] and known as the Zoutendijk condition.

Lemma 5.1. [17, 142] Let Assumption 5.1 be accomplished and the points $\{ x_k \}$ be generated by the method (1.2) and (1.10). Then it holds
\[ \sum_{k=0}^{\infty} \frac{\| g_k \|^4}{\| d_k \|^2} < +\infty. \] (5.4)

5.1. Properties of CG methods with $\gamma_{k-1}^T g_k$ in the numerator of $\beta_k$. In this subsection, we recall properties of the HS, PRP and LS methods. If we look at the chronological development presented in Table 2, a clear observation is that HS, PRP and LS methods involve the expression $\gamma_{k-1}^T g_k$ in the numerator of the parameter $\beta_k$. We first mention the Property (*) for $\beta_k$ given by Gilbert and Nocedal [49]. The Property (*) implies that $\beta_k$ is bounded and small when the step $s_{k-1}$ is small.

Property (*) [49] Let a method defined by (1.2) and (1.10) satisfies
\[ 0 < \gamma \leq \| g_k \| \leq \bar{\gamma}, \quad \forall k \geq 0 \] (5.5)
for all $k \geq 0$. Under this assumption we say that the method possesses the Property (*) if there exist constants $b > 1$ and $\lambda > 0$ such that for all $k$:
\[ |\beta_k| \leq b, \] (5.6)
and
\[ \| s_{k-1} \| \leq \lambda \Rightarrow |\beta_k| \leq \frac{1}{2b}. \] (5.7)

In order to prove that conjugate gradient methods have Property (*), it suffices to show that there exists a constant $c > 0$ such that
\[ |\beta_k| \leq c \| s_{k-1} \| \] for all $k$,
under the assumption (5.5). Then, by putting $\lambda = \frac{1}{2bc}$, we have $|\beta_k| \leq \max\{1, 2bc\} \equiv b$ and
\[ \| s_{k-1} \| \leq \lambda \Rightarrow |\beta_k| \leq \frac{1}{2b}, \] (5.9)
which confirms the Property (*). It is easily shown that (5.8) holds for the HS, PRP and LS methods, and thus these methods have Property (*).

Next, we give the global convergence theorem of CG methods satisfying Property (*). The proof of Theorem 5.2 is given in [49].
Theorem 5.2. Consider any conjugate gradient method (1.2), (1.10) that satisfies the following conditions:

(a) $\beta_k \geq 0$.

(b) The search directions satisfy the sufficient descent condition (1.6)

(c) The Zoutendijk condition holds.

(d) The Property (*) holds.

If the Lipschitz and Boundedness Assumptions hold, then the iterates are globally convergent.

5.2. Properties of CG methods involving $\|g_k\|^2$ in the numerator of $\beta_k$.
In this section, we recall properties of the FR, CD and DY methods. If we look at the chronological development presented in Table 2, it is observable that FR, CD and DY methods involve the value $\|g_k\|^2$ in the numerator of the parameter $\beta_k$. If the step-size $\alpha_k$ satisfies the generalized strong Wolfe conditions (2.7) and (2.9), the following properties are obtained.

Proposition 1. The following statements hold:

(a) For the FR method, if $\alpha_k$ satisfies the generalized strong Wolfe conditions (2.7) and (2.9) with $\sigma_1 + \sigma_2 < 1$, then

$$-\frac{1}{1-\sigma_1} \leq \frac{g_k^T d_k}{\|g_k\|^2} \leq -1 + \frac{\sigma_2}{1-\sigma_1}.$$  (5.10)

(b) For the DY method, if $\alpha_k$ satisfies the generalized strong Wolfe conditions (2.7) and (2.9), then

$$-\frac{1}{1-\sigma_1} \leq \frac{g_k^T d_k}{\|g_k\|^2} \leq -\frac{1}{1+\sigma_2}.$$  (5.11)

(c) For the CD method, if $\alpha_k$ satisfies the generalized strong Wolfe conditions (2.7) and (2.9) with $\sigma_2 < 1$, then

$$-1 - \sigma_1 \leq \frac{g_k^T d_k}{\|g_k\|^2} \leq -1 + \sigma_2.$$  (5.12)

Proposition 1 implies that the FR, CD and DY methods satisfy the sufficient descent condition (1.6), dependent on line searches.

We now give the global convergence properties of the FR and DY methods, which were proven in [2] and [30], respectively.

Theorem 5.3. Suppose that Assumption 5.1 holds. Let the sequence $\{x_k\}$ be generated by the conjugate gradient method of the form (1.2)-(1.10).

(a) If $\beta_k = \beta_k^{FR}$ and $\alpha_k$ satisfies the generalized strong Wolfe conditions (2.7) and (2.9) with $\sigma_1 + \sigma_2 < 1$, then $\{x_k\}$ converges globally with the limit

$$\lim_{k \to \infty} \|g_k\| = 0.$$  (5.13)

(b) If $\beta_k = \beta_k^{DY}$ and $\alpha_k$ satisfies the Wolfe conditions (2.7) and (2.8), then $\{d_k\}$ satisfies the descent condition (1.4) and $\{x_k\}$ converges globally in the sense that (5.13) holds.

Methods surveyed in Subsection 5.2 exhibit strong convergence properties, but they may not be efficient in practice due to appearance of jamming. On the other hand however, methods given in Subsection 5.1 may not be convergent in general, they often perform better than the methods restated in Subsection 5.2. Details about this fact are given in [65]. This clearly implies that combinations of these
methods have been proposed with the aim of exploiting attractive features of each family of methods.

5.3. Convergence of DL methods. The following assumptions will be commonly used in the subsequent convergence analysis of DHSDL, DLSDL, MHSDL and MLSDL methods.

It is supposed that the conditions in Assumption 5.1 hold. Assumption 5.1 initiates the existence of positive constants \( D \) and \( \gamma \) satisfying (5.2) and (5.3).

By the uniform convexity of \( f \), there exists a constant \( \theta > 0 \) such that

\[
(g(u) - g(v))^T(u - v) \geq \theta \|u - v\|^2, \quad \text{for all } u, v \in S,
\]

or equivalently,

\[
f(u) \geq f(v) + g(v)^T(u - v) + \frac{\theta}{2} \|u - v\|^2, \quad \text{for all } u, v \in S.
\]

It follows from (5.14) and (5.15) that

\[
s_k^T s_{k-1} y_{k-1} \geq \theta \|s_{k-1}\|^2
\]

(5.16) and

\[
f_{k-1} - f_k \geq -g_k^T s_{k-1} + \frac{\theta}{2} \|s_{k-1}\|^2.
\]

By (5.1) and (5.16), we have

\[
\theta \|s_{k-1}\|^2 \leq s_k^T s_{k-1} y_{k-1} \leq L \|s_{k-1}\|^2,
\]

(5.18) where the inequalities imply \( \theta \leq L \).

Further, (5.18) implies

\[
s_k^T s_{k-1} y_{k-1} = \alpha_{k-1} d_k^T y_{k-1} > 0.
\]

(5.19) From \( \alpha_{k-1} > 0 \) and (5.19), it follows that

\[
d_k^T y_{k-1} > 0.
\]

(5.20)

In order to improve presentation, an arbitrary method defined by (1.2) and (1.10) will be denoted by \( M(\alpha_k, \beta_k) \). In our current investigation, it will be assumed \( \beta_k \in \{\beta_{\text{DHSDL}}, \beta_{\text{DLSDL}}, \beta_{\text{MHSDL}}, \beta_{\text{MLSDL}}\} \). It is assumed that \( \alpha_k \) satisfies the backtracking condition. Further, we will use the notation \( \beta'_k \in \{\beta_{\text{DHS}}, \beta_{\text{DLS}}, \beta_{\text{MHS}}, \beta_{\text{MLS}}\} \).

Clearly,

\[
\beta_k = \beta'_k - t \frac{g_k^T s_{k-1}}{d_k^T y_{k-1}}.
\]

(5.21)

Lemma 5.4. [17, 142] Let Assumption 5.1 be accomplished and the points \( \{x_k\} \) be generated by the method \( M(\alpha_k, \beta_k) \). Then (5.4) holds

Lemma 5.5. The parameters \( \beta'_k \) in \( M(\alpha_k, \beta_k) \) satisfy

\[
0 \leq \beta'_k \leq \frac{\|g_k\|^2}{\lambda \|g_k d_{k-1}\|}, \quad \lambda \geq 1
\]

(5.22) in each iterative step \( k \).

Proof. In the case \( \beta'_k \in \{\beta_{\text{DHS}}, \beta_{\text{DLS}}\} \) the inequalities

\[
0 \leq \beta_{\text{DHS}} \leq \beta_{\text{DLS}} \leq \frac{\|g_k\|^2}{\mu \|g_k d_{k-1}\|}
\]

are known from [29].
For $\beta_k \equiv \beta_k^{\text{MHS}}$, it is possible to verify
\[ 0 \leq \beta_k^{\text{MHS}} = \frac{\|g_k\|^2 - \|g_k\| \|g_k^T g_{k-1}\|}{d_{k-1}^T y_{k-1}} \leq \frac{\|g_k\|^2}{|g_k^T d_{k-1}|}. \]
For $\beta_k' \equiv \beta_k^{\text{MLS}}$, we have
\[ 0 \leq \beta_k^{\text{MLS}} = \frac{\|g_k\|^2 - \|g_k\| \|g_k^T g_{k-1}\|}{-d_{k-1}^T g_{k-1}} \leq \frac{\|g_k\|^2}{|g_k^T d_{k-1}|}. \]
Since $\mu > 1$ the proof is completed. \qed

Lemma 5.6. The iterations $M(\alpha_k, \beta_k)$ satisfy
\[ g_k^T d_k \leq -c \|g_k\|^2 \] (5.23)
for some $0 \leq c \leq 1$, all $k \geq 0$ and arbitrary $\beta_k$. 

Proof. The inequality (5.23) will be verified by induction. In the initial situation $k = 0$, one obtains $g_0^T d_0 = -\|g_0\|^2$. Since $c \leq 1$, obviously (5.23) is satisfied in the basic case. Suppose that (5.23) is valid for some $k \geq 1$. By taking the inner product of the left and right hand side in (1.10) with the vector $g_k^T$, it can be obtained
\[ g_k^T d_k = -\|g_k\|^2 + \beta_k g_k^T d_{k-1}. \] (5.24)
An application of (5.21) and $s_{k-1} = \alpha_{k-1} d_{k-1}$ leads to further conclusions:
\[ g_k^T d_k = -\|g_k\|^2 + \left(\beta_k - t \frac{g_k^T s_{k-1}}{d_{k-1}^T y_{k-1}}\right) g_k^T d_{k-1} \]
\[ = -\|g_k\|^2 + \beta_k g_k^T d_{k-1} - t \frac{\alpha_{k-1} g_k^T d_{k-1}}{d_{k-1}^T y_{k-1}} \]
\[ = -\|g_k\|^2 + \beta_k g_k^T d_{k-1} - t \frac{\alpha_{k-1} (g_k^T d_{k-1})^2}{d_{k-1}^T y_{k-1}}. \]

From (5.20), $t > 0$ and $\alpha_{k-1} > 0$, one obtains
\[ t \frac{\alpha_{k-1} (g_k^T d_{k-1})^2}{d_{k-1}^T y_{k-1}} > 0. \] (5.25)
Now (5.25) in conjunction with (5.22) implies
\[ g_k^T d_k \leq -\|g_k\|^2 + \beta_k g_k^T d_{k-1} \]
\[ \leq -\|g_k\|^2 + \frac{\|g_k\|^2}{\lambda |g_k^T d_{k-1}|} |g_k^T d_{k-1}| \]
\[ \leq -\|g_k\|^2 + \frac{\|g_k\|^2}{\lambda} \]
\[ = -\left(1 - \frac{1}{\lambda}\right) \|g_k\|^2. \]
In view of $\lambda \geq 1$, the inequality (5.23) is satisfied for $c = (1 - \frac{1}{\lambda})$ and arbitrary $k \geq 0$. \qed

Lemma 5.7. The parameter $\beta_k \in \{\beta_k^{\text{DHS}, \text{DLSDL}}, \beta_k^{\text{MHS}, \text{DLSDL}}\}$ satisfies
\[ \beta_k \leq \frac{\|g_k\|^2 - \|g_k\| \|g_k^T g_{k-1}\| |g_k^T s_{k-1}|}{d_{k-1}^T y_{k-1}}. \] (5.26)
Therefore, the inequality (5.26) follows from (5.27), (5.29) and (5.30).

The following conclusion is valid in the case $\beta_k \equiv \beta_k^{\text{DSDL}}$.

Assume that Assumption 5.1 is true and Theorem 5.8.

Proof. For $\beta_k \equiv \beta_k^{\text{DSDL}}$, in view of (5.20), it follows that

$$
\beta_k^{\text{DSDL}} = \frac{\|g_k\|^2 - \|g_k\|_k^T g_k k - 1}{\mu g_k^T d_{k-1}^T - \|g_k\|_k^T g_k - 1} - t g_k^T s_{k-1} \frac{d_{k-1}^T y_{k-1}}{d_{k-1}^T d_{k-1} y_{k-1}}
$$

(5.27)

As $\mu > 1$ we conclude

$$
d_{k-1}^T y_{k-1} = d_{k-1}^T (g_k - g_{k-1})
= d_{k-1}^T g_k - d_{k-1}^T g_{k-1}
\leq |d_{k-1}^T g_k| - |d_{k-1}^T g_{k-1}|
\leq \mu |d_{k-1}^T g_k| - |d_{k-1}^T g_{k-1}|.
$$

(5.28)

Using inequalities (5.20) and (5.28) in $\beta_k \equiv \beta_k^{\text{DSDL}}$, one obtains

$$
\beta_k^{\text{DSDL}} = \frac{\|g_k\|^2 - \|g_k\|_k^T g_k k - 1}{\mu g_k^T d_{k-1}^T - \|g_k\|_k^T g_k - 1} - t g_k^T s_{k-1} \frac{d_{k-1}^T y_{k-1}}{d_{k-1}^T d_{k-1} y_{k-1}}
$$

(5.29)

$$
= \frac{\|g_k\|^2 - \|g_k\|_k^T g_k k - 1}{\mu g_k^T d_{k-1}^T - \|g_k\|_k^T g_k - 1} - t g_k^T s_{k-1} \frac{d_{k-1}^T y_{k-1}}{d_{k-1}^T d_{k-1} y_{k-1}}.
$$

(5.30)

The following conclusion is valid in the case $\beta_k \equiv \beta_k^{\text{MHSDDL}}$.

$$
\beta_k^{\text{MHSDDL}} = \frac{\|g_k\|^2 - \|g_k\|_k^T g_k k - 1}{\mu g_k^T d_{k-1}^T - \|g_k\|_k^T g_k - 1} - t g_k^T s_{k-1} \frac{d_{k-1}^T y_{k-1}}{d_{k-1}^T d_{k-1} y_{k-1}}
$$

(5.31)

Therefore, the inequality (5.26) follows from (5.27), (5.29) and (5.30).

The global convergence of the proposed methods is confirmed by Theorem 5.8.

**Theorem 5.8.** Assume that Assumption 5.1 is true and $f$ is a uniformly convex function. Then the sequence $\{x_k\}$ generated by the $\mathcal{M}(\alpha_k, \beta_k)$ method fulfills

$$
\liminf_{k \to \infty} \|g_k\| = 0.
$$

(5.31)

Proof. Assume that (5.31) is not true. This implies the existence of a constant $c_1 > 0$ such that

$$
\|g_k\| \geq c_1, \text{ for all } k.
$$

(5.32)

Squaring both sides of (1.10) implies

$$
\|d_k\|^2 = \|g_k\|^2 - 2\beta_k g_k^T d_{k-1} + (\beta_k)^2 \|d_{k-1}\|^2.
$$

(5.33)
Taking into account (5.22), one can obtain
\[ -2\beta_k g_k^T d_{k-1} = -2 \left( \beta_k^* - t \frac{g_k^T s_{k-1}}{d_{k-1}^T y_{k-1}} \right) g_k^T d_{k-1} \]
\[ = -2 \left( \beta_k^* g_k^T d_{k-1} - t \frac{\alpha_k (g_k^T d_{k-1})^2}{d_{k-1}^T y_{k-1}} \right). \]

Now from (5.25), with respect to \( t \frac{\alpha_{k-1} (g_k^T d_{k-1})^2}{d_{k-1}^T y_{k-1}} > 0 \), the following inequalities hold:
\[ -2\beta_k g_k^T d_{k-1} \leq 2|\beta_k^*| |g_k^T d_{k-1}| \]
\[ \leq 2 \left( \frac{\|g_k\|^2}{\lambda} |g_k^T d_{k-1}| \right) \]
\[ \leq \frac{2 \|g_k\|^2}{\lambda}. \]

**Case 1.** In the cases \( \beta_k \in \{ \beta_k^{\text{DSDL}}, \beta_k^{\text{DLSDL}}, \beta_k^{\text{MHS}} \} \) from Lemma 5.7, we conclude
\[ \beta_k \leq \frac{\|g_k\|^2 - \|g_k\| \|g_k^T s_{k-1}\|}{\|g_k^T s_{k-1}\|} - t g_k^T s_{k-1} \]
\[ \leq \frac{\|g_k^T g_{k-1} - \|g_k\| g_{k-1} + t g_k^T s_{k-1}\|}{\|g_k^T s_{k-1}\|} \]
\[ \leq \frac{\|g_k\| (\|g_k - g_{k-1}\| + \|g_k - s_{k-1}\|)}{\|g_k^T s_{k-1}\|} \]
\[ \leq \frac{\|g_k\| (\|g_k - g_{k-1}\| + \|s_{k-1}\|)}{\|g_k^T s_{k-1}\|} \]
\[ \leq \frac{\|g_k\| (\|g_k - g_{k-1}\| + \|s_{k-1}\|)}{\|g_k^T s_{k-1}\|} \]
\[ \leq (2 L + t) \frac{\|g_k\| \|s_{k-1}\|}{\theta \alpha_k \|d_{k-1}\|}. \]

So,
\[ \beta_k \leq \frac{\|g_k\| (2 \|g_k - g_{k-1}\| + t \|s_{k-1}\|)}{\theta \alpha_k \|d_{k-1}\|} \]
\[ \leq \frac{\|g_k\| (2L \|s_{k-1}\| + t \|s_{k-1}\|)}{\theta \alpha_k \|d_{k-1}\|} \]
\[ \leq \frac{(2L + t) \|g_k\| \|s_{k-1}\|}{\theta \alpha_k \|d_{k-1}\|}. \]
Using (5.35) and (5.37) in (5.33), we obtain

\[
\|d_k\|^2 \leq \|g_k\|^2 + 2 \frac{\|g_k\|^2}{\lambda} + \frac{(2L + t)^2 \|g_k\|^2}{\theta^2 \|d_k-1\|^2} \|d_k-1\|^2
\]

\[
\leq \|g_k\|^2 + 2 \frac{\|g_k\|^2}{\lambda} + \frac{(2L + t)^2 \|g_k\|^2}{\theta^2}
\]

\[
\leq \left(1 + \frac{2}{\lambda} + \frac{(2L + t)^2}{\theta^2}\right) \|g_k\|^2
\]  
(5.38)

\[
\leq \left(\lambda + 2 + \frac{(2L + t)^2}{\theta^2}\right) \|g_k\|^2
\]

\[
\leq \frac{(\lambda + 2)\theta^2 + \lambda(2L + t)^2}{\lambda \theta^2} \|g_k\|^2.
\]

Next, dividing both sides of (5.38) by \(\|g_k\|^4\) and using (5.32), it can be concluded

\[
\frac{\|d_k\|^2}{\|g_k\|^4} \leq \frac{(\lambda + 2)\theta^2 + \lambda(2L + t)^2}{\lambda \theta^2 c_1^2} \cdot \frac{1}{c_1^2}
\]

\[
\frac{\|g_k\|^4}{\|d_k\|^2} \geq \frac{\lambda \theta^2 \cdot c_1^2}{(\lambda + 2) \theta^2 + \lambda(2L + t)^2}.
\]

The inequalities in (5.39) imply

\[
\sum_{k=0}^{\infty} \frac{\|g_k\|^4}{\|d_k\|^2} \geq \sum_{k=0}^{\infty} \frac{\lambda \theta^2 \cdot c_1^2}{(\lambda + 2) \theta^2 + \lambda(2L + t)^2} = \infty.
\]

(5.40)

Therefore, \(\|g_k\| \geq c_1\) causes a contradiction to (5.4). Consequently, (5.31) is confirmed for Case 1.

**Case 2.** In the cases \(\beta_k = \beta_k^{\text{MLSDL}}\), applying Lemma 5.6 and Assumption 5.1, we have

\[
\beta_k^{\text{MLSDL}} \leq \left| \frac{\|g_k\|^2 - \frac{\|g_k\|^2}{\|g_{k-1}\|} \|g_{k-1}\|^2}{-d_{k-1}^T g_{k-1}} - t \frac{g_{k}^T s_{k-1}}{d_{k-1}^T y_{k-1}} \right|
\]

\[
= \left| \frac{g_k^T (g_k - \frac{g_k}{\|g_{k-1}\|} g_{k-1})}{-d_{k-1}^T g_{k-1}} + \frac{g_{k}^T s_{k-1}}{d_{k-1}^T y_{k-1}} \right|
\]

\[
= \left| \frac{g_k^T (g_k - g_{k-1} + g_{k-1} - \frac{g_k}{\|g_{k-1}\|} g_{k-1})}{-d_{k-1}^T g_{k-1}} + t \frac{g_{k}^T s_{k-1}}{d_{k-1}^T y_{k-1}} \right|
\]

\[
= \left| \frac{g_k^T (g_k - g_{k-1})}{-d_{k-1}^T g_{k-1}} + \frac{g_{k}^T s_{k-1}}{d_{k-1}^T y_{k-1}} \right|
\]
From (5.18), (5.19) and (5.41), we conclude

\[
\|d\| \leq \left\| \frac{\|g_k - g_{k-1}\| \|g_{k-1}\|}{\|d_{k-1}\|} \right\| + t \|g_k\| \|s_{k-1}\| \|d_{k-1}\| y_{k-1}
\]

\[
\leq \left\| \frac{\|g_k - g_{k-1}\| \|g_{k-1}\|}{\|d_{k-1}\|} \right\| + t \|g_k\| \|s_{k-1}\| \|d_{k-1}\| y_{k-1}
\]

\[
\leq \left\| \frac{\|g_k\|}{\|d_{k-1}\|} \right\| + t \|g_k\| \|s_{k-1}\| \|d_{k-1}\| y_{k-1}
\]

\[
\leq 2 \cdot \frac{\|g_k\|}{\|s_{k-1}\|} \|d_{k-1}\| y_{k-1}
\]

From (5.18), (5.19) and (5.41), we conclude

\[
\beta_k^{MLSDL} \leq 2 \cdot \frac{\|g_k\|}{\|s_{k-1}\|} \|d_{k-1}\| y_{k-1}
\]

\[
= \frac{2L \cdot \|g_k\| \|s_{k-1}\|}{\|g_{k-1}\|^2} + t \frac{\|g_k\| \|s_{k-1}\|}{\|d_{k-1}\| y_{k-1}}
\]

\[
\leq \frac{2L \cdot \|g_k\| \|s_{k-1}\|}{\|g_{k-1}\|^2} + t \frac{\|g_k\|}{\|s_{k-1}\|}
\]

\[
\leq \frac{2L \cdot \|g_k\| \|s_{k-1}\|}{\|g_{k-1}\|^2} + t \frac{\|g_k\|}{\|s_{k-1}\|}
\]

Replacement of (5.35) and (5.42) in (5.33) leads to

\[
\|d_k\|^2 \leq \|g_k\|^2 + 2 \frac{\|g_k\|^2}{\lambda} + \frac{\|g_k\|^2}{\|d_{k-1}\|^2} \|d_{k-1}\|^2
\]

\[
= \|g_k\|^2 + 2 \frac{\|g_k\|^2}{\lambda} + \frac{\|g_k\|^2}{\|d_{k-1}\|^2} \|d_{k-1}\|^2
\]

\[
= \left(1 + 2 \lambda \frac{\|g_k\|^2}{\|d_{k-1}\|^2} \right) \|g_k\|^2
\]

\[
= \left(1 + 2 \lambda \frac{\|g_k\|^2}{\|d_{k-1}\|^2} \right) \|g_k\|^2
\]

\[
= \left(1 + 2 \lambda \frac{\|g_k\|^2}{\|d_{k-1}\|^2} \right) \|g_k\|^2
\]

\[
= \left(1 + 2 \lambda \frac{\|g_k\|^2}{\|d_{k-1}\|^2} \right) \|g_k\|^2
\]
Next, dividing both sides of (5.43) by $\|g_k\|^4$ and using (5.32), it can be concluded

$$
\frac{\|d_k\|^2}{\|g_k\|^4} \leq \frac{(\lambda + 2) \left(c \cdot c_1^2 \cdot \theta \cdot \alpha_{k-1}\right)^2 + \lambda \left(2L\theta \cdot D^2 + c \cdot c_1^2 \cdot t\right)^2}{\lambda (c \cdot c_1^2 \cdot \theta \cdot \alpha_{k-1})^2} \cdot \frac{1}{c_1^4},
$$

(5.44)

The inequalities in (5.44) imply

$$
\sum_{k=0}^{\infty} \frac{\|g_k\|^4}{\|d_k\|^2} \geq \sum_{k=0}^{\infty} \frac{\lambda (c \cdot c_1^2 \cdot \theta \cdot \alpha_{k-1})^2 \cdot c_1^2}{(\lambda + 2) (c \cdot c_1^2 \cdot \theta \cdot \alpha_{k-1})^2 + \lambda (2L\theta \cdot D^2 + c \cdot c_1^2 \cdot t)^2} \cdot \frac{1}{c_1^4} > \infty.
$$

(5.45)

Therefore, $\|g_k\| \geq c_1$ causes a contradiction to (5.4). Consequently, (5.31) is confirmed for Case 2. The proof is complete. \(\square\)

6. **Numerical experiments.** The code used in the testing experiments is written in the software Matlab R2017a, and executed on the personal computer Workstation Intel Core i3 2.0 GHz, 8GB of RAM memory, and Windows 10 operating system. Three important criteria: the number of iterations (IT), number of function evaluations (FE) and CPU time (CPU) in all tested methods are analyzed.

The numerical experiments are performed on contains functions presented in [3], where much of the problems are taken over from CUTEr collection [14]. Each test function is tested 10 times with a gradually increasing values of the dimension by the rule $n = 10, 50, 100, 200, 300, 500, 700, 800, 1000$ and $1500$.

Strong Wolfe line search use the following choice of parameters for all algorithms $\sigma_1 = 0.0001$ and $\sigma_2 = 0.5$.

We utilized the performance profile given in [43] to compare numerical results (IT, FE and CPU) for all tested methods. The upper curve of the selected performance profile corresponds to the method that shows the best performance.

6.1. **Numerical experiments on QN methods with constant diagonal Hessian approximation.** BFGS, DFP, SR1 updates in QN methods with respect to different ILS strategies are compared in [40]. The main conclusion is that the BFGS method is superior to the others. Continuing such research, we compare the numerical performances obtained from AGD, MSM and SM methods, i.e, gradient methods with acceleration parameter. The numerical experiment contains 25 test functions proposed in [3]. For each of tested functions, we performed 12 numerical experiments with $100, 200, 300, 500, 1000, 2000, 3000, 5000, 7000, 8000, 10000$, and $15000$ variables. Tested algorithms are based on the same implementation of the backtracking line search (Algorithm 2), which we set $\omega = 0.0001$ and $\varphi = 0.8$.

The uniform stopping criteria in this numerical experiments are

$$
\|g_k\| \leq 10^{-6} \quad \text{and} \quad \frac{|f_{k+1} - f_k|}{1 + |f_k|} \leq 10^{-16}.
$$

Summary numerical data generated by AGD, MSM and SM method, tried on 25 test functions, are arranged in Table 4.

Table 4 contains numerical results corresponding to IT, FE and CPU criteria for the AGD, MSM and SM methods. Figures 1, 2, 3 illustrate the performance profiles corresponding to the results in Table 4 corresponding to the criterion IT, GE and CPU, respectively.
Table 4. Summary numerical results of the AGD, MSM and SM methods with respect to IT, FE and CPU.

<table>
<thead>
<tr>
<th>Test function</th>
<th>IT profile</th>
<th>FE profile</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AGD</td>
<td>MSM</td>
<td>SM</td>
</tr>
<tr>
<td>Perturbed Quadratic</td>
<td>353897</td>
<td>349828</td>
<td>99908</td>
</tr>
<tr>
<td>Raydan 1</td>
<td>22620</td>
<td>20694</td>
<td>14018</td>
</tr>
<tr>
<td>Diagonal 2</td>
<td>129416</td>
<td>152892</td>
<td>183445</td>
</tr>
<tr>
<td>Generalized Tri diagonal 1</td>
<td>9795</td>
<td>8468</td>
<td>325</td>
</tr>
<tr>
<td>Extended Tri diagonal 1</td>
<td>3564</td>
<td>1370</td>
<td>14292</td>
</tr>
<tr>
<td>Extended FPE</td>
<td>141</td>
<td>316</td>
<td>134</td>
</tr>
<tr>
<td>Diagonal 4</td>
<td>129</td>
<td>96</td>
<td>90</td>
</tr>
<tr>
<td>Extended Rimmelstein</td>
<td>396</td>
<td>280</td>
<td>196</td>
</tr>
<tr>
<td>Perturbed quadratic diagonal</td>
<td>2542050</td>
<td>37454</td>
<td>44903</td>
</tr>
<tr>
<td>Quadratic QF1</td>
<td>366183</td>
<td>36169</td>
<td>62927</td>
</tr>
<tr>
<td>Extended quadratic penalty QP1</td>
<td>280</td>
<td>369</td>
<td>271</td>
</tr>
<tr>
<td>Extended quadratic penalty QP2</td>
<td>395887</td>
<td>1674</td>
<td>3489</td>
</tr>
<tr>
<td>Quadratic QF2</td>
<td>100286</td>
<td>32727</td>
<td>64076</td>
</tr>
<tr>
<td>Extended quadratic exponential EP1</td>
<td>48</td>
<td>100</td>
<td>73</td>
</tr>
<tr>
<td>Extended Tri diagonal 2</td>
<td>1657</td>
<td>659</td>
<td>543</td>
</tr>
<tr>
<td>ARWHEAD (CUTE)</td>
<td>5667</td>
<td>430</td>
<td>270</td>
</tr>
<tr>
<td>Almost Perturbed Quadratic</td>
<td>356094</td>
<td>33652</td>
<td>60789</td>
</tr>
<tr>
<td>LIARWHD (CUTE)</td>
<td>1054019</td>
<td>3029</td>
<td>18691</td>
</tr>
<tr>
<td>ENGVAL1 (CUTE)</td>
<td>743</td>
<td>461</td>
<td>375</td>
</tr>
<tr>
<td>QUARTC (CUTE)</td>
<td>174</td>
<td>267</td>
<td>200</td>
</tr>
<tr>
<td>Generalized Quartic</td>
<td>187</td>
<td>181</td>
<td>180</td>
</tr>
<tr>
<td>Diagonal 7</td>
<td>141</td>
<td>158</td>
<td>158</td>
</tr>
<tr>
<td>Diagonal 8</td>
<td>290</td>
<td>290</td>
<td>290</td>
</tr>
<tr>
<td>Full Hessian FH3</td>
<td>45</td>
<td>96</td>
<td>90</td>
</tr>
</tbody>
</table>

From Table 4, we conclude that the AGD, MSM and SM methods have successfully solved all test functions.

Figure 1 presents the performance profiles of the IT of the AGD, MSM and SM methods. In this figure, it is observable that MSM method is best in 52.00% of the test functions compared with: AGD (24.00%) and SM (32.00%).

From Figure 1, it is observable that the graph of the MSM method comes first to the top, which means that the MSM is superior compared to other considered methods with respect to the IT profile.

Figure 2 presents the performance profiles of the FE of the AGD, MSM and SM methods. It is observable that MSM method is best in 64.00% of tested functions compared with: AGD (12.00%) and SM (32.00%). In view of Figure 2, the MSM graph first comes to the top, which means that the MSM is winer with respect to the FE profile.

![Figure 1. IT performance profile for AGD, MSM and SM methods.](image-url)
Figure 2. FE performance profile for AGD, MSM and SM methods.

Figure 3 presents the performance profiles of the CPU of the AGD, MSM and SM methods. It is obvious that MSM is winner in 56.00% of the test functions with respect to: AGD (4.00%) and SM (44.00%). Figure 3 demonstrates that the graph of the MSM method first comes to the highest level, which signifies that the MSM is winner with respect to the CPU.

From the previous analysis of the results shown in Table 4 and Figures 1-3, we can conclude that the MSM iterates are most efficient in terms of all three basic metrics: IT, FE and CPU. The MSM method has the smallest IT, FE and the CPU time compared to the other two methods on the most test functions.

6.2. Numerical experiments on the CG methods with $y^T_{k-1} g_k$ in the numerator of $\beta_k$. The uniform stopping criterion during testing CG methods is

$$\|g_k\| \leq \epsilon,$$

where $\epsilon = 10^{-6}$ or when the number of function evaluations becomes greater than 1000000.
In this subsection, we compare the numerical results obtained from HS, PRP and LS methods, i.e., methods with $\gamma_k^{T\gamma_{k-1}} \beta_k$ in the numerator of $\beta_k$. The numerical experiment is based on 26 test functions. Summary numerical results for HS, PRP and LS method, tried on 26 test functions, are presented in Table 5.

Table 5 shows the numerical results (IT, FE and CPU) for the HS, PRP and LS methods.

<table>
<thead>
<tr>
<th>Test function</th>
<th>IT profile</th>
<th>FE profile</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HS</td>
<td>PRP</td>
<td>LS</td>
</tr>
<tr>
<td>Perturbed Quadratic</td>
<td>1157</td>
<td>1157</td>
<td>6662</td>
</tr>
<tr>
<td>Raydan 2</td>
<td>NaN</td>
<td>474</td>
<td>474</td>
</tr>
<tr>
<td>Diagonal 2</td>
<td>NaN</td>
<td>1721</td>
<td>5067</td>
</tr>
<tr>
<td>Extended Triagonal 1</td>
<td>NaN</td>
<td>779</td>
<td>779</td>
</tr>
<tr>
<td>Diagonal 4</td>
<td>NaN</td>
<td>196</td>
<td>1927</td>
</tr>
<tr>
<td>Diagonal 5</td>
<td>NaN</td>
<td>154</td>
<td>30</td>
</tr>
<tr>
<td>Extended Himmelblau</td>
<td>159</td>
<td>512</td>
<td>210</td>
</tr>
<tr>
<td>Pult Bessel FH2</td>
<td>5966</td>
<td>5966</td>
<td>14490</td>
</tr>
<tr>
<td>Perturbed quadratic diagonal</td>
<td>1074</td>
<td>11120</td>
<td>21667</td>
</tr>
<tr>
<td>Quadratic QF1</td>
<td>1158</td>
<td>1158</td>
<td>5612</td>
</tr>
<tr>
<td>Extended quadratic penalty QP2</td>
<td>NaN</td>
<td>11456</td>
<td>5395</td>
</tr>
<tr>
<td>Quadratic QF2</td>
<td>2056</td>
<td>2311</td>
<td>9682</td>
</tr>
<tr>
<td>Extended quadratic exponential EPI</td>
<td>NaN</td>
<td>NaN</td>
<td>1350</td>
</tr>
<tr>
<td>TRIDIA (CUTE)</td>
<td>633</td>
<td>6744</td>
<td>NaN</td>
</tr>
<tr>
<td>Almost Perturbed Quadratic</td>
<td>1158</td>
<td>11158</td>
<td>5996</td>
</tr>
<tr>
<td>LIARWHD (CUTE)</td>
<td>NaN</td>
<td>408</td>
<td>11498</td>
</tr>
<tr>
<td>POWER (CUTE)</td>
<td>174</td>
<td>1389</td>
<td>150528</td>
</tr>
<tr>
<td>NOSCOMP (CUTE)</td>
<td>654</td>
<td>9647</td>
<td>NaN</td>
</tr>
<tr>
<td>QUARTIC (CUTE)</td>
<td>NaN</td>
<td>165</td>
<td>1347</td>
</tr>
<tr>
<td>Diagonal 6</td>
<td>NaN</td>
<td>174</td>
<td>139</td>
</tr>
<tr>
<td>DIXONDOQ (CUTE)</td>
<td>NaN</td>
<td>12599</td>
<td>12039</td>
</tr>
<tr>
<td>BIGGSR1 (CUTE)</td>
<td>NaN</td>
<td>11444</td>
<td>11517</td>
</tr>
<tr>
<td>Generalized Quadratic</td>
<td>NaN</td>
<td>124</td>
<td>139</td>
</tr>
<tr>
<td>Diagonal 8</td>
<td>NaN</td>
<td>51</td>
<td>80</td>
</tr>
<tr>
<td>PLETCHCR (CUTE)</td>
<td>18292</td>
<td>19084</td>
<td>20354</td>
</tr>
</tbody>
</table>

Figures 4, 5 and 6 plot the performance profiles for the results in Table 5 with respect to IT, FE and CPU criterion, respectively.

Figure 4 presents the performance profiles of the IT correspondig to the HS, PRP and LS methods. It is observable that PRP method is best in 61.54% of the test functions compared with: HS (26.92%) and LS (23.08%). From Figure 4, it is observable that the graph of the PRP method comes first to the top, which signifies that the PRP outperforms other considered methods with respect to the IT criterion.

Figure 5 presents the performance profiles of the FE of the HS, PRP and LS methods. It is observable that PRP method is best in 69.32% of the test functions compared with: HS (23.08%) and LS (23.08%). From Figure 5, it is observed that the PRP graph first comes to the top, which signifies that the PRP is the best with respect to the FE.

Figure 6 presents the performance profiles of the CPU of the HS, PRP and LS methods. It is observable that PRP method is best in 69.23% of the test functions compared with: HS (23.08%) and LS (23.08%). From Figure 6, it is observed that the PRP graph first comes to the top, which signifies that the PRP is the best with respect to the CPU.

From the previous analysis of the results shown in Table 5 and Figures 4-6, we can conclude that the PRP method achieved the best and most efficient results in terms of all three basic metrics: IT, FE and CPU.
Figure 4. IT performance profile for HS, PRP and LS methods.

Figure 5. FE performance profile for HS, PRP and LS methods.

Figure 6. CPU time performance profile for HS, PRP and LS methods.
6.3. Numerical experiments on CG methods with $\|g_k\|^2$ in the numerator of $\beta_k$. In this subsection, we compare the numerical results obtained from DY, FR and CD methods, i.e., methods with $\|g_k\|^2$ in the numerator of $\beta_k$. The numerical experiment contains 25 test functions. Summary numerical results for DY, FR and CD method, tried on 25 test functions, are presented in Table 6.

Table 6 contains numerical results (IT, FE and CPU) for the DY, FR and CD methods. Figures 7, 8 and 9 plot the performance profiles for the results in Table 6 with respect to profiles IT, FE and CPU, respectively.

<table>
<thead>
<tr>
<th>Test function</th>
<th>IT profile</th>
<th>FE profile</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perturbed Quadratic</td>
<td>1157</td>
<td>1157</td>
<td>1157</td>
</tr>
<tr>
<td>Raydan 2</td>
<td>86</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>Diagonal 2</td>
<td>6366</td>
<td>3440</td>
<td>2068</td>
</tr>
<tr>
<td>Extended TriDiagonal 1</td>
<td>2081</td>
<td>990</td>
<td>1140</td>
</tr>
<tr>
<td>Diagonal 4</td>
<td>70</td>
<td>70</td>
<td>200</td>
</tr>
<tr>
<td>Diagonal 5</td>
<td>40</td>
<td>124</td>
<td>155</td>
</tr>
<tr>
<td>Extended Himmelblau</td>
<td>583</td>
<td>339</td>
<td>207</td>
</tr>
<tr>
<td>Full Hessian FH2</td>
<td>4682</td>
<td>4868</td>
<td>4784</td>
</tr>
<tr>
<td>Perturbed quadratic diagonal</td>
<td>1036</td>
<td>1084</td>
<td>1276</td>
</tr>
<tr>
<td>Quadratic QF1</td>
<td>1158</td>
<td>1158</td>
<td>1158</td>
</tr>
<tr>
<td>Quadratic QF2</td>
<td>NaN</td>
<td>NaN</td>
<td>2349</td>
</tr>
<tr>
<td>Extended quadratic exponential EP1</td>
<td>NaN</td>
<td>NaN</td>
<td>1255</td>
</tr>
<tr>
<td>Almost Perturbed Quadratic</td>
<td>NaN</td>
<td>NaN</td>
<td>10901</td>
</tr>
<tr>
<td>LIARWHD (CUTE)</td>
<td>2812</td>
<td>1202</td>
<td>1255</td>
</tr>
<tr>
<td>POWER (CUTE)</td>
<td>1779</td>
<td>7781</td>
<td>7825</td>
</tr>
<tr>
<td>NONSCOMP (CUTE)</td>
<td>2558</td>
<td>13483</td>
<td>10901</td>
</tr>
<tr>
<td>QUARTC (CUTE)</td>
<td>134</td>
<td>94</td>
<td>99</td>
</tr>
<tr>
<td>Diagonal 6</td>
<td>86</td>
<td>40</td>
<td>192</td>
</tr>
<tr>
<td>DIXON3DQ (CUTE)</td>
<td>16047</td>
<td>18776</td>
<td>19376</td>
</tr>
<tr>
<td>BIGGSB1 (CUTE)</td>
<td>15274</td>
<td>17835</td>
<td>18374</td>
</tr>
<tr>
<td>Generalized Quartic</td>
<td>142</td>
<td>214</td>
<td>173</td>
</tr>
<tr>
<td>Diagonal 7</td>
<td>50</td>
<td>50</td>
<td>160</td>
</tr>
<tr>
<td>Diagonal 8</td>
<td>50</td>
<td>40</td>
<td>190</td>
</tr>
<tr>
<td>Full Hessian FH3</td>
<td>43</td>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td>FLETCHCR (CUTE)</td>
<td>NaN</td>
<td>NaN</td>
<td>20793</td>
</tr>
</tbody>
</table>

From Figure 7, it is observable that the graph of the CD method comes first to the top, which signifies that the CD outperforms other considered methods with respect to the IT.

Figure 8 presents the performance profiles of the FE of the DY, FR and CD methods. From Figure 8, it is observed that the CD graph first comes to the highest level, which means that the CD possesses best performances with respect to the criterion FE.

Figure 9 presents the performance profiles of the CPU of the DY, FR and CD methods. Figure 9 demonstrates that the graph of the CD method first achieves the top level, so that the CD is a winner with respect to the CPU.

From the previous analysis of the results shown in Table 6 and Figures 7-9, it is clear that the CD method achieved most efficient results in terms of all three basic metrics: IT, FE and CPU.

6.4. Numerical experiments on the hybrid conjugate gradient methods. This subsection analyses numerical results obtained by running a MATLAB implementation with predefined conditions given at the beginning section. The following
ten hybrid CG methods in the form of (1.2) and (1.10), which differ only in the choice of the CG parameter $\beta_k$, are tested:

- HCG1: The CG method with $\beta_k$ defined by (4.15).
- HCG2: The CG method with $\beta_k$ defined by (4.16).
- HCG3: The CG method with $\beta_k$ defined by (4.17).
- HCG4: The CG method with $\beta_k$ defined by (4.18).
- HCG5: The CG method with $\beta_k$ defined by (4.19) in which $c = \frac{1-\sigma}{1+\sigma}$.
- HCG6: The CG method with $\beta_k$ defined by (4.20).
- HCG7: The CG method with the parameter $\beta_k$ defined by (4.27).
- HCG8: The CG method with the parameter $\beta_k$ defined by (4.22) in which $\theta_k \in [0, 1]$.
- HCG9: The CG method with the parameter $\beta_k$ defined by (4.23) in which $\theta_k \in [0, 1]$.
- HCG10: The CG method with the parameter $\beta_k$ defined by (4.24) in which $\nu_k, \theta_k \in [0, 1]$.

The numerical experiment contains 25 test functions.
Summary numerical results for hybrid CG methods, tried on 25 test functions, with respect to IT, FE and CPU profiles are presented in Table 7.

Table 7. Summary numerical results of the hybrid CG methods HCG1–HCG10 with respect to IT.

<table>
<thead>
<tr>
<th>Test function</th>
<th>HCG1</th>
<th>HCG2</th>
<th>HCG3</th>
<th>HCG4</th>
<th>HCG5</th>
<th>HCG6</th>
<th>HCG7</th>
<th>HCG8</th>
<th>HCG9</th>
<th>HCG10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perturbed Quadratic</td>
<td>1157</td>
<td>1157</td>
<td>1157</td>
<td>1157</td>
<td>1157</td>
<td>1157</td>
<td>1157</td>
<td>1157</td>
<td>1157</td>
<td>1157</td>
</tr>
<tr>
<td>Raydan 2</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Diagonal 2</td>
<td>1355</td>
<td>1355</td>
<td>1355</td>
<td>1355</td>
<td>1355</td>
<td>1355</td>
<td>1355</td>
<td>1355</td>
<td>1355</td>
<td>1355</td>
</tr>
<tr>
<td>Extended Tridiagonal 1</td>
<td>312</td>
<td>312</td>
<td>312</td>
<td>312</td>
<td>312</td>
<td>312</td>
<td>312</td>
<td>312</td>
<td>312</td>
<td>312</td>
</tr>
<tr>
<td>Diagonal 4</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>Diagonal 5</td>
<td>124</td>
<td>124</td>
<td>124</td>
<td>124</td>
<td>124</td>
<td>124</td>
<td>124</td>
<td>124</td>
<td>124</td>
<td>124</td>
</tr>
<tr>
<td>Extended Himmelblau</td>
<td>145</td>
<td>145</td>
<td>145</td>
<td>145</td>
<td>145</td>
<td>145</td>
<td>145</td>
<td>145</td>
<td>145</td>
<td>145</td>
</tr>
<tr>
<td>Full Hessian FH2</td>
<td>4036</td>
<td>4036</td>
<td>4036</td>
<td>4036</td>
<td>4036</td>
<td>4036</td>
<td>4036</td>
<td>4036</td>
<td>4036</td>
<td>4036</td>
</tr>
<tr>
<td>Perturbed quadratic diagonal 2</td>
<td>428</td>
<td>428</td>
<td>428</td>
<td>428</td>
<td>428</td>
<td>428</td>
<td>428</td>
<td>428</td>
<td>428</td>
<td>428</td>
</tr>
<tr>
<td>Quadratic QP1</td>
<td>515</td>
<td>515</td>
<td>515</td>
<td>515</td>
<td>515</td>
<td>515</td>
<td>515</td>
<td>515</td>
<td>515</td>
<td>515</td>
</tr>
<tr>
<td>Quadratic QP2</td>
<td>2125</td>
<td>2125</td>
<td>2125</td>
<td>2125</td>
<td>2125</td>
<td>2125</td>
<td>2125</td>
<td>2125</td>
<td>2125</td>
<td>2125</td>
</tr>
<tr>
<td>TRIDIA (CUTE)</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0.210</td>
<td>0.210</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0.210</td>
</tr>
<tr>
<td>Almost Perturbed Quadratic</td>
<td>1355</td>
<td>1355</td>
<td>1355</td>
<td>1355</td>
<td>1355</td>
<td>1355</td>
<td>1355</td>
<td>1355</td>
<td>1355</td>
<td>1355</td>
</tr>
<tr>
<td>Liarwhid (CUTE)</td>
<td>1.677</td>
<td>1.677</td>
<td>1.677</td>
<td>1.677</td>
<td>1.677</td>
<td>1.677</td>
<td>1.677</td>
<td>1.677</td>
<td>1.677</td>
<td>1.677</td>
</tr>
<tr>
<td>Power (CUTE)</td>
<td>1275</td>
<td>1275</td>
<td>1275</td>
<td>1275</td>
<td>1275</td>
<td>1275</td>
<td>1275</td>
<td>1275</td>
<td>1275</td>
<td>1275</td>
</tr>
<tr>
<td>Quartic (CUTE)</td>
<td>2.4</td>
<td>2.4</td>
<td>2.4</td>
<td>2.4</td>
<td>2.4</td>
<td>2.4</td>
<td>2.4</td>
<td>2.4</td>
<td>2.4</td>
<td>2.4</td>
</tr>
<tr>
<td>Diagonal 6</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Dixon3d2q (CUTE)</td>
<td>0.2152</td>
<td>0.2152</td>
<td>0.2152</td>
<td>0.2152</td>
<td>0.2152</td>
<td>0.2152</td>
<td>0.2152</td>
<td>0.2152</td>
<td>0.2152</td>
<td>0.2152</td>
</tr>
<tr>
<td>Hyp2 (CUTE)</td>
<td>1.0664</td>
<td>1.0664</td>
<td>1.0664</td>
<td>1.0664</td>
<td>1.0664</td>
<td>1.0664</td>
<td>1.0664</td>
<td>1.0664</td>
<td>1.0664</td>
<td>1.0664</td>
</tr>
<tr>
<td>Generalized Quartic</td>
<td>1.29</td>
<td>1.29</td>
<td>1.29</td>
<td>1.29</td>
<td>1.29</td>
<td>1.29</td>
<td>1.29</td>
<td>1.29</td>
<td>1.29</td>
<td>1.29</td>
</tr>
<tr>
<td>Diagonal 7</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>Full Hessian FH3</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Fletcher (CUTE)</td>
<td>1.7621</td>
<td>1.7621</td>
<td>1.7621</td>
<td>1.7621</td>
<td>1.7621</td>
<td>1.7621</td>
<td>1.7621</td>
<td>1.7621</td>
<td>1.7621</td>
<td>1.7621</td>
</tr>
</tbody>
</table>

Figure 10 plot corresponding performance profiles IT for the results included in Table 7, in three columns denoted by IT.

From Figure 10, it is observable that the graph of the HCG6 method comes first to the top, which signifies that the HCG6 outperforms other considered methods with respect to the IT. However, if we look in more detail Figure 10, we can see that the HCG6 method does not have the best results, it has the best results in only (16%), while the HCG7 method has the best results in (52%) on the number of functions tested. The reason for such behavior lies in the fact that the HCG6 method is the only one that has successfully solved all the test problems.

The numerical results of the hybrid CG methods with respect to the FE are arranged in Table 8.
Figure 10. IT performance profile for hybrid CG methods HCG1–HCG10.

Table 8. Summary numerical results of the hybrid CG methods HCG1–HCG10 with respect to FE.

<table>
<thead>
<tr>
<th>Test function</th>
<th>HCG1</th>
<th>HCG2</th>
<th>HCG3</th>
<th>HCG4</th>
<th>HCG5</th>
<th>HCG6</th>
<th>HCG7</th>
<th>HCG8</th>
<th>HCG9</th>
<th>HCG10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perturbed Quadratic</td>
<td>3481</td>
<td>3481</td>
<td>3481</td>
<td>3481</td>
<td>3481</td>
<td>3481</td>
<td>3481</td>
<td>3481</td>
<td>3481</td>
<td>3481</td>
</tr>
<tr>
<td>Raydan 2</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>134</td>
<td>176</td>
<td>182</td>
<td>100</td>
<td>158</td>
<td>NaN</td>
<td>282</td>
</tr>
<tr>
<td>Diagonal 1</td>
<td>2136</td>
<td>9317</td>
<td>5006</td>
<td>5923</td>
<td>5944</td>
<td>9281</td>
<td>5944</td>
<td>6142</td>
<td>4822</td>
<td>9711</td>
</tr>
<tr>
<td>Extended Tridiagonal 1</td>
<td>12609</td>
<td>1991</td>
<td>2274</td>
<td>1878</td>
<td>2924</td>
<td>2941</td>
<td>3119</td>
<td>2661</td>
<td>NaN</td>
<td>8909</td>
</tr>
<tr>
<td>Diagonal 4</td>
<td>170</td>
<td>170</td>
<td>200</td>
<td>170</td>
<td>200</td>
<td>200</td>
<td>170</td>
<td>200</td>
<td>NaN</td>
<td>340</td>
</tr>
<tr>
<td>Diagonal 5</td>
<td>238</td>
<td>88</td>
<td>206</td>
<td>98</td>
<td>270</td>
<td>228</td>
<td>98</td>
<td>292</td>
<td>338</td>
<td>210</td>
</tr>
<tr>
<td>Extended Himmelblau</td>
<td>1565</td>
<td>687</td>
<td>584</td>
<td>704</td>
<td>813</td>
<td>961</td>
<td>767</td>
<td>1013</td>
<td>767</td>
<td>594</td>
</tr>
<tr>
<td>Full Hessian FH2</td>
<td>13115</td>
<td>15515</td>
<td>18515</td>
<td>34467</td>
<td>44467</td>
<td>44467</td>
<td>14467</td>
<td>44467</td>
<td>14467</td>
<td>34467</td>
</tr>
<tr>
<td>Perturbed quadratic diagonal</td>
<td>9686</td>
<td>9647</td>
<td>9805</td>
<td>2805</td>
<td>3282</td>
<td>2967</td>
<td>2993</td>
<td>3053</td>
<td>NaN</td>
<td>8044</td>
</tr>
<tr>
<td>Quadratic QF1</td>
<td>1584</td>
<td>1484</td>
<td>1484</td>
<td>1484</td>
<td>1484</td>
<td>1484</td>
<td>1484</td>
<td>1484</td>
<td>1484</td>
<td>1484</td>
</tr>
<tr>
<td>Quadratic QP2</td>
<td>1845</td>
<td>9202</td>
<td>9201</td>
<td>9216</td>
<td>9054</td>
<td>10229</td>
<td>10886</td>
<td>9511</td>
<td>9585</td>
<td>20260</td>
</tr>
<tr>
<td>TRIDIA (CUTE)</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>18640</td>
<td>18640</td>
<td>16792</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>20260</td>
</tr>
<tr>
<td>Almost Perturbed Quadratic</td>
<td>1845</td>
<td>1484</td>
<td>1484</td>
<td>1484</td>
<td>1484</td>
<td>1484</td>
<td>1484</td>
<td>1484</td>
<td>1484</td>
<td>1484</td>
</tr>
<tr>
<td>LIARWHD (CUTE)</td>
<td>1712</td>
<td>5931</td>
<td>9275</td>
<td>6165</td>
<td>8113</td>
<td>9395</td>
<td>9554</td>
<td>10305</td>
<td>NaN</td>
<td>8488</td>
</tr>
<tr>
<td>POWER (CUTE)</td>
<td>23356</td>
<td>23356</td>
<td>23356</td>
<td>23356</td>
<td>23356</td>
<td>23356</td>
<td>23356</td>
<td>23356</td>
<td>23356</td>
<td>23356</td>
</tr>
<tr>
<td>NONSCOMP (CUTE)</td>
<td>41353</td>
<td>42211</td>
<td>27961</td>
<td>52705</td>
<td>51458</td>
<td>40687</td>
<td>54017</td>
<td>23411</td>
<td>13367</td>
<td>35106</td>
</tr>
<tr>
<td>QUARCC (CUTE)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Diagonal 6</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>DIXONJXQ (CUTE)</td>
<td>16098</td>
<td>17534</td>
<td>157355</td>
<td>15594</td>
<td>15594</td>
<td>46073</td>
<td>15594</td>
<td>57234</td>
<td>NaN</td>
<td>50796</td>
</tr>
<tr>
<td>HCG6H1 (CUTE)</td>
<td>13999</td>
<td>15534</td>
<td>31427</td>
<td>55554</td>
<td>83427</td>
<td>10846</td>
<td>15554</td>
<td>48620</td>
<td>NaN</td>
<td>31469</td>
</tr>
<tr>
<td>Generalized Quartic</td>
<td>477</td>
<td>371</td>
<td>370</td>
<td>371</td>
<td>481</td>
<td>529</td>
<td>471</td>
<td>439</td>
<td>446</td>
<td>467</td>
</tr>
<tr>
<td>Diagonal 7</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>130</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Diagonal 8</td>
<td>180</td>
<td>130</td>
<td>130</td>
<td>190</td>
<td>NaN</td>
<td>NaN</td>
<td>190</td>
<td>190</td>
<td>130</td>
<td>130</td>
</tr>
<tr>
<td>Full Hessian FH3</td>
<td>119</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>FLETCHCR (CUTE)</td>
<td>166463165774168759175559175559184939174406125687</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 11 plots the performance profiles for the results in Table 8, in three columns denoted by FE.

From Figure 11, it is observable that the graph of the HCG6 method comes first to the top, which signifies that the HCG6 outperforms other considered methods with respect to the FE. However, if we look in more detail Figure 11, we can see an identical situation as in Figure 10 that the HCG6 method does not have the best results, it has the best results in (16%), while the HCG2 method has the best results in (48%) on the number of functions tested.

Table 9 contains numerical results of the hybrid CG methods with respect to the CPU.

Figure 12 plots the performance profiles for the results in Tables 9.

From Figure 12, it is observable that the graph of the HCG6 method comes first to the top, which signifies that the HCG6 outperforms other considered methods
with respect to the CPU. However, if we look in more detail Figure 12, we can see an identical situation as in the figures 10 and 11 that the HCG6 method does not have the best results, it has the best results in only (8%), while the HCG7 and HCG10 methods has the best results in (20%) on the number of functions tested.

### 6.5. Numerical experiments on the modified Dai-Liao methods.

The numerical experiments presented in this subsection investigate the influence of the scalar size in the modified Dai-Liao methods. The previously mentioned variants of the DL method use a fixed value of the parameter $t$. It can also be seen that the scalar $t$ in all the above papers are greater than 0 and is less than 1. Analyzing the results from [18, 26, 131, 139], we conclude that the scalar $t$ was defined by a fixed value of 0.1 in numerical experiments. Also, numerical experience related the fixed

### Table 9. Summary numerical results of the hybrid CG methods HCG1–HCG10 with respect to the CPU (sec).

<table>
<thead>
<tr>
<th>Test function</th>
<th>HCG1</th>
<th>HCG2</th>
<th>HCG3</th>
<th>HCG4</th>
<th>HCG5</th>
<th>HCG6</th>
<th>HCG7</th>
<th>HCG8</th>
<th>HCG9</th>
<th>HCG10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perturbed Quadratic</td>
<td>0.656</td>
<td>0.516</td>
<td>0.781</td>
<td>0.719</td>
<td>0.594</td>
<td>0.698</td>
<td>0.688</td>
<td>0.844</td>
<td>0.688</td>
<td>0.844</td>
</tr>
<tr>
<td>Raydan 2</td>
<td>0.034</td>
<td>0.234</td>
<td>0.154</td>
<td>0.234</td>
<td>0.154</td>
<td>0.234</td>
<td>0.154</td>
<td>0.234</td>
<td>0.154</td>
<td>0.234</td>
</tr>
<tr>
<td>Diagonal 2</td>
<td>1.453</td>
<td>1.328</td>
<td>1.656</td>
<td>1.797</td>
<td>1.813</td>
<td>1.266</td>
<td>1.250</td>
<td>1.141</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extended Tridiagonal 1</td>
<td>1.016</td>
<td>1.125</td>
<td>1.359</td>
<td>2.250</td>
<td>2.375</td>
<td>7.578</td>
<td>1.672</td>
<td>1.375</td>
<td>0.922</td>
<td></td>
</tr>
<tr>
<td>Diagonal 4</td>
<td>0.831</td>
<td>0.311</td>
<td>0.311</td>
<td>0.311</td>
<td>0.311</td>
<td>0.311</td>
<td>0.311</td>
<td>0.311</td>
<td>0.311</td>
<td>0.311</td>
</tr>
<tr>
<td>Extended Himmelblau</td>
<td>0.144</td>
<td>0.364</td>
<td>0.156</td>
<td>0.244</td>
<td>0.124</td>
<td>0.109</td>
<td>0.085</td>
<td>0.219</td>
<td>0.166</td>
<td></td>
</tr>
<tr>
<td>Full Hessian FH2</td>
<td>0.078</td>
<td>0.176</td>
<td>0.176</td>
<td>0.176</td>
<td>0.176</td>
<td>0.176</td>
<td>0.176</td>
<td>0.176</td>
<td>0.176</td>
<td>0.176</td>
</tr>
<tr>
<td>Perturbed quadratic diagonal</td>
<td>0.406</td>
<td>0.609</td>
<td>0.641</td>
<td>0.375</td>
<td>0.563</td>
<td>0.359</td>
<td>0.328</td>
<td>0.344</td>
<td>0.734</td>
<td></td>
</tr>
<tr>
<td>Quadratic QF1</td>
<td>0.359</td>
<td>0.438</td>
<td>0.422</td>
<td>0.422</td>
<td>0.406</td>
<td>0.391</td>
<td>0.484</td>
<td>0.422</td>
<td>0.281</td>
<td></td>
</tr>
<tr>
<td>Quadratic QF2</td>
<td>0.047</td>
<td>0.133</td>
<td>0.203</td>
<td>0.156</td>
<td>0.063</td>
<td>0.136</td>
<td>0.090</td>
<td>0.104</td>
<td>0.047</td>
<td></td>
</tr>
<tr>
<td>TRIDIA (CUTE)</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>1.588</td>
<td>1.391</td>
<td>1.759</td>
<td>0.978</td>
<td>1.076</td>
<td>0.682</td>
<td></td>
</tr>
<tr>
<td>Almost Perturbed Quadratic</td>
<td>0.406</td>
<td>0.438</td>
<td>0.316</td>
<td>0.564</td>
<td>0.250</td>
<td>0.359</td>
<td>0.406</td>
<td>0.344</td>
<td>0.322</td>
<td></td>
</tr>
<tr>
<td>CDWAVI (CUTE)</td>
<td>0.586</td>
<td>0.586</td>
<td>0.586</td>
<td>0.586</td>
<td>0.586</td>
<td>0.586</td>
<td>0.586</td>
<td>0.586</td>
<td>0.586</td>
<td></td>
</tr>
<tr>
<td>POWER (CUTE)</td>
<td>1.064</td>
<td>1.672</td>
<td>1.750</td>
<td>1.609</td>
<td>1.625</td>
<td>1.575</td>
<td>1.625</td>
<td>1.888</td>
<td>1.463</td>
<td></td>
</tr>
<tr>
<td>NONSOMP (CUTE)</td>
<td>1.047</td>
<td>1.484</td>
<td>1.967</td>
<td>1.766</td>
<td>1.422</td>
<td>1.719</td>
<td>1.546</td>
<td>1.663</td>
<td>1.203</td>
<td>1.703</td>
</tr>
<tr>
<td>QUARTC (CUTE)</td>
<td>0.550</td>
<td>0.580</td>
<td>0.609</td>
<td>0.569</td>
<td>0.515</td>
<td>0.735</td>
<td>0.938</td>
<td>0.703</td>
<td>1.156</td>
<td>0.633</td>
</tr>
<tr>
<td>Diagonal 6</td>
<td>0.078</td>
<td>0.078</td>
<td>0.078</td>
<td>0.094</td>
<td>0.063</td>
<td>0.016</td>
<td>0.125</td>
<td>0.109</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIKONRDQ (CUTE)</td>
<td>2.047</td>
<td>2.047</td>
<td>2.047</td>
<td>2.047</td>
<td>2.047</td>
<td>2.047</td>
<td>2.047</td>
<td>2.047</td>
<td>2.047</td>
<td>2.047</td>
</tr>
<tr>
<td>BIGGSBI (CUTE)</td>
<td>1.875</td>
<td>2.047</td>
<td>2.359</td>
<td>1.750</td>
<td>2.250</td>
<td>2.391</td>
<td>1.422</td>
<td>2.072</td>
<td>2.047</td>
<td>2.047</td>
</tr>
<tr>
<td>Generalized Quartic</td>
<td>0.063</td>
<td>0.125</td>
<td>0.131</td>
<td>0.156</td>
<td>0.125</td>
<td>0.094</td>
<td>0.078</td>
<td>0.169</td>
<td>0.172</td>
<td>0.109</td>
</tr>
<tr>
<td>Diagonal 7</td>
<td>0.063</td>
<td>0.016</td>
<td>0.016</td>
<td>0.016</td>
<td>0.016</td>
<td>0.016</td>
<td>0.016</td>
<td>0.016</td>
<td>0.016</td>
<td>0.016</td>
</tr>
<tr>
<td>Diagonal 8</td>
<td>0.078</td>
<td>0.125</td>
<td>0.078</td>
<td>0.031</td>
<td>0.063</td>
<td>0.031</td>
<td>0.063</td>
<td>0.047</td>
<td>0.109</td>
<td>0.063</td>
</tr>
<tr>
<td>Full Hessian FH3</td>
<td>0.063</td>
<td>0.047</td>
<td>0.199</td>
<td>0.047</td>
<td>0.031</td>
<td>0.063</td>
<td>0.047</td>
<td>0.109</td>
<td>0.063</td>
<td></td>
</tr>
<tr>
<td>Pleuchech (CUTE)</td>
<td>0.666</td>
<td>0.616</td>
<td>0.616</td>
<td>0.616</td>
<td>0.616</td>
<td>0.616</td>
<td>0.616</td>
<td>0.616</td>
<td>0.616</td>
<td>0.616</td>
</tr>
</tbody>
</table>

Figure 11. FE performance profile for hybrid CG methods HCG1–HCG10.
valued \( t = 1 \) was reported in [26]. Common numerical experience is that different choice of \( t \) initiate totally different numerical experience.

That is why we come to the next question. What is the ‘best’ value of \( t \in (0,1) \) from the computational point of view? Because of that, our intention is to investigate numerically and theoretically behavior of different variants of the DL conjugate gradient framework with respect to various values \( t \). For this reason, we started this research with the aim to find answer to the aforementioned question. Our strategy is to select several values of the parameter \( t \) within the interval \((0,1)\) and to compare the obtained results based on different criteria. In that way, we will get the answer to the question: whether it is better to take values closer to zero or closer to one.

6.5.1. Motivations and the corresponding algorithm. As we have already indicated in the previous section, the aim of this subsection is to answer to the question: what is the ‘best’ value of \( t \in (0,1) \) in DL CG computational scheme? Our plan is to examine numerically the influence of the scalar \( t \) in the DL class of iterations and determine some rules for its appropriate choice and, if possible, find the best value. The detailed research plan is to find the answer to two challenging questions:

- Does and how much the values of the scalar \( t \) affect each of the methods DHSDL, DLSDL, MHSDL and MLSDL, which are observed individually, with respect to IT, FE and the CPU time (CPU)?
- Does the choice of \( t \) favor one (or some) of the considered methods?

To give an answer to these questions, we would have to test all the methods under the same conditions. During testing, we will compare all the considered methods with the same values of required scalars. The Algorithm 2, i.e. the backtracking line search, determines the step-size \( \alpha_k \) in (1.2).

Algorithm 6 gives the corresponding general framework for DHSDL, DLSDL, MHSDL and MLSDL methods.

Values of \( t \) used during the testing of the observed methods are given in the Table 10. Each particular value of the scalar \( t \) is marked with one of the labels \( T1 \) to \( T6 \), corresponding to a joined value of \( t \). Five of the six given values are fixed during testing of the observed methods. Only the value of \( t \) labeled by \( T1 \) is variable during the iterations, where the value in \( k \)th iteration is denoted by \( t_k \).
Algorithm 6 Modified Dai-Liao conjugate gradient methods.

Require: A starting point $x_0$, real numbers $0 < \epsilon < 1$, $0 < \delta < 1$, $\mu > 1$ and $t > 0$.

1: Set $k = 0$ and compute $d_0 = -g_0$.
2: If 
   \[ \|g_k\| \leq \epsilon \quad \text{and} \quad \frac{|f_{k+1} - f_k|}{1 + |f_k|} \leq \delta \]
   STOP;
   else perform Step 3.
3: Determine $\alpha_k \in (0, 1)$ using backtracking in Algorithm 2.
4: Compute $x_{k+1} = x_k + \alpha_k d_k$.
5: Calculate $g_{k+1}, y_k = g_{k+1} - g_k, s_k = x_{k+1} - x_k$.
6: Calculate $\beta_k$ by (4.9) or (4.10) or (4.11) or (4.12).
7: Compute $d_k = -g_k + \beta_k d_{k-1}$.
8: $k := k + 1$, and go to Step 2.

In this case, the value $t_k$ is obtained from backtracking line search, i.e., $t_k = \alpha_k$ inherits a new value obtained from Algorithm 2 in each iteration.

The reason why we decided to define values $t_k$ from the backtracking line search algorithm is:
- $t_k \in (0, 1)$;
- $t_k$ also affects the iterative steps when computing the next value for $x_k$.

<table>
<thead>
<tr>
<th>Label</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value of the scalar $t_k$</td>
<td>$t_k = \alpha_k$</td>
<td>0.05</td>
<td>0.1</td>
<td>0.2</td>
<td>0.5</td>
<td>0.9</td>
</tr>
</tbody>
</table>

The convergence of the above methods has already considered in the mentioned references. Our goal is to give a unified analysis of the convergence of proposed DL methods. The aim of our research is to investigate the influence of the scalar $t$ in the DL iterates.

Each particular value of the scalar $t$ is marked with one of the labels T1 to T6, corresponding to a joined value of $t$. Five of the six given values are fixed during testing of the observed methods. Only the value of $t$ labeled by T1 is variable during the iterations, where the value in $k$th iteration is denoted by $t_k$. In this case, the value $t_k$ is obtained from backtracking line search, i.e., $t_k = \alpha_k$ inherits a new value obtained from Algorithm 2 in each iteration.

The convergence of the above methods has already considered in the mentioned references. Our goal is to give a unified analysis of the convergence of the proposed DL methods. The aim of our research is to investigate the influence of the scalar $t$ in the DL iterates.

Arranged numerical results are generated by testing and comparing DHSDL, DLSDL, MHSDL and MLSDL methods on 6 different values $t$, denoted by T1-T6. Three important criteria (IT, CPU, FE) in all tested methods are analyzed. Numerical report is based on 22 test functions proposed in [3]. For each of tested functions, we performed 10 numerical experiments with 100, 500, 1000, 3000, 5000, 7000, 8000,
10000, 15000 and 20000 variables. Summary results for DHSDL, DLSDL, MHSDL and MLSDL methods, tried on 22 tests, are presented.

The uniform stopping criteria are (refer to previous)

\[ \|g_k\| \leq 10^{-6} \quad \text{and} \quad \frac{|f_{k+1} - f_k|}{1 + |f_k|} \leq 10^{-16}. \]

The backtracking parameters for all algorithms are \( \omega = 0.0001 \) and \( \varphi = 0.8 \).

During the testing of the DHSDL and DLSDL methods, the constant parameter \( \mu = 1.2 \) was used in each iteration.

6.5.2. Numerical experiments on DHSDL method. Figure 13 indicates the performance profiles of the IT criterion with respect to the DHSDL method depending on the scalar value \( t \). This figure exhibits that the DHSDL method successfully solved all the problems for all values of the scalar \( t \). Also, the DHSDL-T2 method is finest in 45.5\% of testings with respect to DHSDL-T1 (18.2\%), DHSDL-T3 (9.1\%), DHSDL-T4 (4.5\%), DHSDL-T5 (13.6\%) and DHSDL-T6 (13.6\%). It can be noticed that the graph of DHSDL-T2 reaches the top first, which signifies that DHSDL-T2 is the best with respect to IT.

Figure 13. Performance profiles of DHSDL (T1,T2,T3,T4,T5,T6) method based on IT.

Figure 14 shows the performance profile given by FE of the DHSDL solver with respect to \( t \). Evidently, DHSDL solves all test problems for all values of the scalar \( t \), and the DHSDL-T2 method is winer in 50.0\% of the test problems compared to the DHSDL-T1 (18.2\%), DHSDL-T3 (9.1\%), DHSDL-T4 (0\%), DHSDL-T5 (13.6\%) and DHSDL-T6 (9.1\%). From Figure 14, it is notifiable that the DHSDL-T2 is the best with respect to FE.

Figure 15 illustrates the CPU criterion spanned by the DHSDL method depending on the metrics \( t \). Again, the DHSDL method is able to solve all the tested problems for all values of the scalar \( t \). Further, the DHSDL-T6 method is superior in 40.9\% of tests with respect to the DHSDL-T1 (4.5\%), DHSDL-T2 (27.3\%), DHSDL-T3 (9.1\%), DHSDL-T4 (4.5\%) and DHSDL-T5 (13.6\%). We also observed that at the beginning, DHSDL-T2 does not perform well but as the number of problems increases, its graph crosses other graphs and achieves the top first, which means that the DHSDL-T2 is dominant with respect to CPU.
Figure 14. Performance profiles of DHSDL (T1,T2,T3,T4,T5,T6) method based on FE.

Figure 15. Performance profile of DHSDL (T1,T2,T3,T4,T5,T6) method based on CPU time.

The graphs displayed in figures 13–15 show that the DHSDL method has achieved superior results for $t \equiv T_2 = 0.05$.

6.5.3. Numerical experiments on DLSDL method. Figure 16 illustrates the IT criterion in the DLSDL method depending on the scalar value $t$. It is observable that DLSDL successfully solves all the problems for all values of the scalar $t$. Moreover, the DLSDL-T2 method is winner in 36.4% of the tests compared to DLSDL-T1 (22.7%), DLSDL-T3 (13.6%), DLSDL-T4 (9.1%), DLSDL-T5 (9.1%) and DLSDL-T6 (13.6%). Figure 16 (left) exhibits that the graph DLSDL-T2 achieves the top first, which means that DLSDL-T2 outperforms all the other methods with respect to IT.

Figure 17 shows the performance profiles of the criterion FE corresponding to the DLSDL method and the scalar value $t$. In this Figure, it is observable that DLSDL for all values of the scalar $t$ successfully solves all tests, and DLSDL-T2 is
best in 36.4% of the test functions in comparison to DLSDL-T1 (18.2%), DLSDL-T3 (4.5%), DLSDL-T4 (18.2%), DLSDL-T5 (13.6%) and DLSDL-T6 (9.1%). From Figure 17, it is observed that DLSDL-T2 is the best with respect to FE.

Figure 18 shows the performance profiles of the CPU time of the DLSDL method depending on $t$. The graphs in this figure indicate that DLSDL method solved all the problems for all values of $t$, and the DLSDL-T4 method is superior in 31.8% of the test problems compared to DLSDL-T1 (13.6%), DLSDL-T2 (22.7%), DLSDL-T3 (18.2%), DLSDL-T5 (9.1%) and DLSDL-T6 (4.6%). We also observed that at the beginning, DLSDL-T2 does not perform well; but as the number of problems increases, its graph crosses other graphs and comes to the top which signifies that, with respect to CPU, the DLSDL-T2 is the best.

Based on figures (16–18) analysis, we come to the conclusion that the DLSDL method has achieved best responses for $t = T2 = 0.05$.

6.5.4. Analysis of average values. In Subsection 6.5.1, we have defined two questions. Our aim in this subsection is to give answers. In order to answer properly to the first question, in Tables 11, 12, 13 are collected average values for all three considered criteria.
Figure 18. Performance profile of DLSDL (T1,T2,T3,T4,T5,T6) method based on CPU.

Table 11. Average IT values for 22 test functions tested on 10 numerical experiments.

<table>
<thead>
<tr>
<th>Method</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
</tr>
</thead>
<tbody>
<tr>
<td>DHSDL</td>
<td>12980.14</td>
<td>31281.32</td>
<td>33640.45</td>
<td>32942.36</td>
<td>34448.32</td>
<td>33872.36</td>
</tr>
<tr>
<td>DLSDL</td>
<td>30694.00</td>
<td>28701.14</td>
<td>31048.32</td>
<td>30594.77</td>
<td>31926.59</td>
<td>31573.05</td>
</tr>
<tr>
<td>MHSDL</td>
<td>29289.73</td>
<td>27653.64</td>
<td>29660.00</td>
<td>29713.50</td>
<td>30491.18</td>
<td>30197.27</td>
</tr>
<tr>
<td>MLSDL</td>
<td>25398.82</td>
<td>22941.77</td>
<td>24758.27</td>
<td>2450.68</td>
<td>25722.64</td>
<td>25032.64</td>
</tr>
</tbody>
</table>

Table 12. Average values of FE for 22 test functions tested on 10 numerical experiments.

<table>
<thead>
<tr>
<th>Method</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
</tr>
</thead>
<tbody>
<tr>
<td>DHSDL</td>
<td>1228585.50</td>
<td>1191960.55</td>
<td>1252957.09</td>
<td>1238044.36</td>
<td>1271176.59</td>
<td>1255710.45</td>
</tr>
<tr>
<td>DLSDL</td>
<td>1131421.41</td>
<td>1083535.14</td>
<td>1149482.41</td>
<td>1134315.00</td>
<td>1167030.14</td>
<td>1158554.77</td>
</tr>
<tr>
<td>MHSDL</td>
<td>1089700.41</td>
<td>1036710.32</td>
<td>1089777.64</td>
<td>1091985.41</td>
<td>1105299.91</td>
<td>1101380.18</td>
</tr>
<tr>
<td>MLSDL</td>
<td>904217.14</td>
<td>845017.55</td>
<td>891669.50</td>
<td>879473.14</td>
<td>913165.68</td>
<td>895652.36</td>
</tr>
</tbody>
</table>

Table 13. Average CPU time for 22 test functions tested on 10 numerical experiments.

<table>
<thead>
<tr>
<th>Method</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
</tr>
</thead>
<tbody>
<tr>
<td>DHSDL</td>
<td>902.06</td>
<td>894.73</td>
<td>917.77</td>
<td>930.56</td>
<td>911.28</td>
<td>870.93</td>
</tr>
<tr>
<td>DLSDL</td>
<td>816.68</td>
<td>790.63</td>
<td>804.69</td>
<td>816.28</td>
<td>803.84</td>
<td>809.67</td>
</tr>
<tr>
<td>MHSDL</td>
<td>770.78</td>
<td>751.65</td>
<td>728.61</td>
<td>749.70</td>
<td>712.64</td>
<td>720.57</td>
</tr>
<tr>
<td>MLSDL</td>
<td>573.14</td>
<td>587.41</td>
<td>581.50</td>
<td>576.32</td>
<td>582.62</td>
<td>580.96</td>
</tr>
</tbody>
</table>

After the numerical testing of the compared methods and the individual analysis for each method, we can now give a global conclusion of the behavior of the observed methods. The first conclusion is that the value of the scalar $t$ significantly affects each of the DHSDL, DLSDL, MHSDL and MLSDL methods with respect to all three criteria. Further, a common conclusion is that all the methods give the best performance profiles for the scalar value $t = 0.05$. We can also give an answer to another question. According to the previous analysis, a particular selection of the
scalar value $t$ can give priority to one of the observed method. All methods do not
behave identically for the same values of the scalar $t$. If we observe Table 11 which
contains the average number of iterations, we can notice that the difference between
the smallest and the biggest average result of the observed method is in the range
of 10.1\% to 12.1\% in relation to the minimum obtained value. Table 12 shows the
average results related FE. An individual comparison of considered methods leads
to the conclusion that the difference between the smallest and the biggest average
results is in the range of 6.6\% to 8.9\% in relation to the minimum obtained value.
This brings us to the same conclusion once again, that is, the value of the scalar
$t$ affects the methods in a given percentage. Also, if we observe Table 13 with the
average CPU time, we can notice that the difference between the smallest and the
biggest average CPU time observed method is in the range of 2.5\% to 10.6\% in
relation to the minimum value.

7. Conclusion. Overview of QN methods, CG methods and their classification
are presented. Section 5 investigates convergence properties of CG methods, fol-
lowing the CG classes in accordance with the presented taxonomy of basic CG
methods. Numerical experiments compare main classes of QN and CG methods.
More precisely, main QN methods with constant diagonal Hessian approximation
are compared as well as two classes of basic CG methods, hybrid CG methods, and
finally some variants of modified Dai-Liao methods.

The problem of defining further improvements of the CGUP $\beta_k$ is still open.
Moreover, new CG methods could be defined using appropriate updates of the
parameter $t$. Another research stream includes various hybridizations of so far
proposed CG methods. On the other hand, there are open possibilities for defining
new updates of the matrices $B_k$ and $H_k$, used in defining QN methods. Continuing
research on some composite definitions of $\beta_k$ based on CG and BFGS updates, it is
possible to discover new three-term (or even different) CG variants.

One of prospective fields for further research includes a generalization of the
discrete-time approach to continuous-time approach, considered in [69]. Another
possibility for further research is extension of gradient methods to tensor case. This
possibility was exploited in [77] on solving $M$-tensor equations.

Moreover, an application of low rank updates used in optimization can be trans-
ferrred to appropriate numerical methods for computing generalized inverses. Ap-
plications of rank-one update formula are investigated in [115, 116].

Acknowledgments. Predrag Stanimirović and Dijana Mosić are supported by the
Ministry of Education, Science and Technological Development, Grants No. 174013
and 174007. Haifeng Ma is supported by the National Natural Science Foundation
of China under grant 11971136 and the bilateral project between China and and
Poland (no. 37-18).

REFERENCES

[2] M. Al-Baali, Descent property and global convergence of the Fletcher-Reeves method with
(2008), 147–161.


1622 P. S. STANIMIROVIĆ, B. IVANOV, H. MA AND D. MOSIĆ


A SURVEY OF GRADIENT METHODS FOR NONLINEAR OPTIMIZATION 1623


[113] P. S. Stanimirović, V. N. Katsikis and D. Pappas, Computation of \( \{2, 4\} \) and \( \{2, 3\} \)-inverses based on rank-one updates, Linear Multilinear Algebra, 66 (2018), 147–166.


Received October 2020; revised November 2020.

E-mail address: pecko@pmf.ni.ac.rs
E-mail address: ivanov.branislav@gmail.com
E-mail address: haifengma.aliyun.com
E-mail address: dijana@pmf.ni.ac.rs