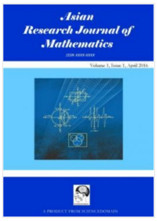

Asian Research Journal of Mathematics

9(1): 1-18, 2018; Article no.ARJOM.39293

ISSN: 2456-477X

Optimal Hybrid Bfgs-Cg Method for Unconstrained Optimization

O. M. Bamigbola[1], O. O. Okundalaye[2*] and C. N. Ejieji[1]

[1]*Department of Mathematics, University of Ilorin, Nigeria.*
[2]*Department of Mathematical Sciences, Adekunle Ajasin University (AAUA), Nigeria.*

Authors' contributions

This work was carried out in collaboration between all authors. Author OMB designed the study, Author OOO performed the statistical analysis, wrote the protocol, and wrote the first draft of the manuscript. Authors OMB and OOO managed the analyses of the study. Author CNE managed the literature searches. All authors read and approved the final manuscript.

Article Information

DOI: 10.9734/ARJOM/2018/39293

Editor(s):

(1) Danilo Costarelli, Department of Mathematics and Computer Science, University of Perugia, Italy.

Reviewers:

(1) Grienggrai Rajchakit, Maejo University, Thailand.
(2) Muhammad Asif Jan, Kohat University of Science & Technology, Pakistan.
(3) Snezana S. Djordjevic, College for Textile, Serbia.

Complete Peer review History: http://sciencedomain.org/review-history/24054

Received: 9th January 2018
Accepted: 9th March 2018
Published: 9th April 2018

Original Research Article

Abstract

In solving unconstrained optimization problems, both quasi-Newton and conjugate gradient methods are known to be efficient methods. Hence, the optimal hybrid Broyden-Fletcher-Goldfarb-Shanno-Conjugate Gradient (OBFGS-CG) method is proposed in this work, which combines the strengths of both BFGS and CG methods. The optimal hybrid BFGS-CG method is based on an existing hybrid BFGS-CG method. The optimal BFGS-CG paramter, when utilised in solving unconstrained optimization problems, resulted in improvement in the total number of iterations and CPU time.

**Corresponding author: ombamigbola@unilorin.edu.ng, okundalaye.oluwaseun@aaua.edu.ng;*

Keywords: Quasi-Newton, hybrid BFGS-conjugate gradient, optimal method, unconstrained optimization.

1 Introduction

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

where R^n is an n-dimensional Euclidean space and $f : R^n \rightarrow R$ is continuously differentiable.

Generally, all iterative techniques for solving(1.1) require an initial starting point x_0 to generate a sequence of points x_i, $i = 1, 2, 3, ..., k$, which represent improved approximations to the solution using the formula

$$x_{i+1} = x_i + \alpha_i d_i \tag{1.2}$$

where d_i denotes the search direction and α_i, the steplength.

A descent approach for solving (1.1) is to iterate in such a way as to decrease the objective function continuously from one step to the next. In this manner, the global convergence, that is convergence from an arbitrary starting point, can be assured, [1]. The descent methods are classified into two groups - first-order methods and second-order methods. First-order methods (also called gradient methods) are those techniques which employ at most the first-order derivative of the function under consideration, [2]. Second-order method or quasi-linearisation methods are those techniques which make use of at most the second-order derivative of the objective function.

The descent methods are steepest descent method, Newton method, conjugate gradient method and quasi-Newton method. Many computational algorithms today are characterized by the steepest descent method (SDM), or some of its modifications. Although the SDM has received a lot of attention and it is easy to apply, it is not recommended for general optimization purpose because of its slow convergence rate [3]. A computational scheme called the conjugate gradient (CG) method was later developed to speed up the convergence rate of the gradient method. The CG method is preferred above other methods for solving large-scale nonlinear optimization problems because of its very low memory requirement as it does not store information about Hessian matrix, [4].

Among the second derivative methods, the best known is the quasi-Newton methods, the most popular being the Broyden-Fletcher-Goldfarb-Shanno (BFGS) version. Some experts in the field of nonlinear unconstrained optimization agree that second derivative methods are more reliable than methods that are solely on gradient information because generally they converge in fewer iterations, [5]. However, the second derivative methods have the disadvantage of requiring substantially more storage. As such, there is sense in combining two methods with the sole purpose of ameliorating the demerits of a good method with the advantageous attributes of the other method. Indeed, there are past research efforts combining other methods with the quasi-Newton method, see [6], [7]. In order to blend the desirable features of the CG method with quasi-Newton method, [8] developed a hybrid BFGS-CG method for solving unconstrained optimization problems. This paper therefore, presents an improvement of the hybrid method by utilizing the optimal search parameter in the initial proposal.

2 Preliminaries

The success of the numerical solution of (1.1) heavily depends on the choice of the step length α_i and search direction d_i i.e., the different choices of the step length and search direction lead to different convergence properties. There are three ways to determine the value of the step length, namely, the exact line search, the inexact line search and use of formula. More often, it is impracticable to use the exact line search. The use of formula is a recent development which efficiency is still being investigated. Among the several inexact line searches available, the Armijo rule is adjudged as one of the most useful and the easiest implementable procedure, [9]. The line search can be described as follows:

Given $s > 0$, $\delta \in (0, 1)$ and $\sigma \in (0, 1)$ find $\alpha_i = max\{s, s\delta, s\delta^2, \ldots\}$ such that

$$f(x_i) - f(x_i + \alpha_i d_i) \geq -\sigma \alpha_i g_i^T d_i, \quad i = 0, 1, 2, ..., n. \tag{2.1}$$

One requirement of the search direction d_i is the satisfaction of descent condition to guarantee the attainment of the minimum value of the objective function $f(x)$. The CG method easily satisfy the descent condition as the current direction to explore for the minimization objective is a linear combination of the gradient vector and the previous search vector i.e.,

$$d_i = \begin{cases} -g_i, & i = 0 \\ -g_i + \beta_i d_{i-1}, & i \geq 1 \end{cases} \tag{2.2}$$

where $g_i = \nabla f(x_i)$ and β_i is known as the CG coefficient. There are many ways to calculate β_i and some well-known formulae are:

$$\beta_i^{FR} = \frac{g_i^T g_i}{||g_{i-1}||^2} \tag{2.3}$$

$$\beta_i^{PR} = \frac{g_i^T (g_i - g_{i-1})}{||g_{i-1}||^2} \tag{2.4}$$

$$\beta_i^{HS} = \frac{g_i^T (g_i - g_{i-1})}{(g_i - g_{i-1})^T d_{i-1}} \tag{2.5}$$

$$\beta_i^{BAN} = \frac{-g_i^T (g_i - g_{i-1})}{g_{i-1}^T (g_i - g_{i-1})} \tag{2.6}$$

$$\beta_i^{HZ} = \left((g_i - g_{i-1}) - \frac{2d_i \|(g_i - g_{i-1})\|^2)}{d_i^T (g_i - g_{i-1})} \right)^T \left(\frac{g_{i+1}}{(d_i^T (g_i - g_{i-1}))} \right) \tag{2.7}$$

$$\beta_i^{DY} = \frac{g_i^T g_i}{d_i^T (g_i - g_{i-1})} \tag{2.8}$$

$$\beta_i^{LS} = \frac{-g_i^T (g_i - g_{i-1})}{d_i^T g_{i-1}} \tag{2.9}$$

$$\beta_i^{CD} = \frac{-g_i^T g_i}{d_i^T (g_i - g_{i-1})} \tag{2.10}$$

$$\beta_i^{NF} = \frac{g_i^T g_{i-1}}{g_{i-1}^T d_{i-1}} \tag{2.11}$$

where g_i and g_{i-1} are gradients of $f(x)$ at the points x_i and x_{i-1}, respectively, while $||.||$ is a norm of vectors and d_{i-1} is a direction for the previous iteration. The above corresponding coefficients are known as [10], [11], [2], [12], [13], [14], [15], [16], [17].

The algorithm for conjugate gradient method is as below.

Algorithm 2.1: The algorithm for conjugate gradient method

Step 1. Start with an arbitrary initial point x_0

Step 2. Set the initial search direction $d_0 = -g_0$
Step 3. Find the point x_1 according to the relation $x_1 = x_0 + \alpha_0 d_0$
where α_0 is the optimal step length in the direction d_0 set i=1 and go to the next step.
Step 4. Find $g_i = g(x_i)$ and set $d_i = -g_i + \beta_i d_{i-1}$. Compute the optimum step length α_i in the direction d_i and find the new point $x_{i+1} = x_i + \alpha_i d_i$.

In quasi-Newton methods, the search direction is given by

$$d_i = -H_i g_i \tag{2.12}$$

where H_i is an approximation of the Hessian. Initial matrix H_0 is chosen as the identity matrix, and subsequently computed with an update formula. The update formula for BFGS is

$$H_{i+1} = H_i - \frac{H_i s_i s_i^T H_i}{s_i^T H_i s_i} + \frac{y_i y_i^T}{s_i^T y_i} \tag{2.13}$$

with $s_i = x_i - x_{i-1}$ and $y_i = g_i - g_{i-1}$. The approximation that the Hessian must fulfill is

$$H_{i+1} s_i = y_i \tag{2.14}$$

This condition is required to hold for the updated matrix H_{i+1} subject to satisfaction of the curvature condition

$$s_i^T y_i > 0. \tag{2.15}$$

Algorithm 2.2 for the BFGS method:

Step 1. Given a starting point x_0 and $H_0 = I_n$, set $i = 1$
Step 2. Terminate if $||g(x_i)|| < 10^{-6}$ or $i > 1000$
Step 3. Calculate the search direction by (2.12) and the step length α_i by (2.1).
Step 4. Compute the difference between $s_i = x_i - x_{i-1}$ and $y_i = g_i - g_{i-1}$
Step 5. Update H_i by (2.13) to obtain H_{i+1}
Step 6. Set $i = i + 1$ and go to Step 2.

3 The Hybrid Methods

The main goal of combining two methods to form an hybrid method is to replace the weakness of a component of the hybrid with the strength of the other component. This is the basis of the hybrid BFGS-CG method proposed by [8] for which the search direction is given as

$$d_i = \begin{cases} -H_i g_i, & i = 0 \\ -H_i g_i + \eta(-g_i + \beta_i d_{i-1}), & i \geq 1 \end{cases} \tag{3.1}$$

where $\eta > 0$, $\quad \beta_i = \frac{g_i^T g_{i-1}}{g_i^T d_{i-1}}$.

In solving (1.1), the BFGS-CG method adopts the Armijo line search in computing the step length with the following computational scheme:

Algorithm 2.3 for BFGS-CG method.

Step 1. Given a starting point x_0 and $H_0 = I_n$, choose values for σ and set $i = 1$.
Step 2. Terminate if $||g(x_i)|| < 10^{-6}$ or $i > 1000$.
Step 3. Calculate the search direction by (3.1) and the step length α_i by (2.1).
Step 4. Compute the difference between $s_i = x_i - x_{i-1}$ and $y_i = g_i - g_{i-1}$.
Step 5. Update H_i by (2.13) to obtain H_{i+1}.

Step 6. Set $i = i + 1$ and go to Step 2.

It was shown in [8], with a numerical test on selected unconstrained problems, that the hybrid BFGS-CG method is globally convergent and that there were significant improvements in the iteration number and execution time, in comparison with some CG methods and some quasi-Newton method. With a mindset to improve the BFGS-CG method, herein is proposed two variants, namely, the Optimal hybrid BFGS-CG (OBFGS-CG) and the Pure hybrid BFGS-CG (PBFGS-CG) methods. The basic difference between the existing and the proposed BFGS-CG hybrids is in the definition of the search direction. The OBFGS-CG utilizes the optimal search parameter to have the search direction as

$$d_i = \begin{cases} -H_i g_i, & i = 0 \\ -H_i g_i + \eta^*(-g_i + \beta_i d_{i-1}), & i \geq 1 \end{cases} \quad (3.2)$$

where η^* obtained from **Lemma 3.3** below, is the optimal value of $\eta > 0$ and β_i is the CG coefficient.

The current search direction for PBFGS-CG is the linear combination of the previous search direction and the quasi-Newton projection of the gradient. Thus, the search direction is given by

$$d_i = \begin{cases} -H_i g_i, & i = 0 \\ -H_i g_i + \eta_i d_{i-1}, & i \geq 1 \end{cases} \quad (3.3)$$

where

$$\eta_i = \frac{(H_i g_i)^T (g_i - g_{i-1})}{d_{i-1}^T (g_i - g_{i-1})}, \quad (3.4)$$

see [18] for details.

4 Analysis of the Hybrid Methods

The following definitions are prerequisites to the proceeding analysis.

Definitions

The search direction d_i is said to satisfy

(i) the descent condition if

$$g_i^T d_i < 0 \quad (4.1)$$

(ii) the sufficient descent condition if there exists a constant $c > 0$ such that

$$g_i^T d_i \leq -c||g_i||^2 \quad (4.2)$$

where g_i denotes the corresponding gradient.

Lemma 3.1 [19]

In the CG method,

$$g_i^T d_{i-1} = 0. \quad (4.3)$$

Lemma 3.2

The hybrid BFGS-CG family is a set of descent methods.

Proof

This is shown for only the OBFGS-CG member of the family. From (3.2),

$$\begin{aligned} g_i^T d_i &= -g_i^T H_i g_i - \eta^* \|g_i\|^2 - \beta_i \eta^* g_i^T d_{i-1} \\ &= -g_i^T H_i g_i - \eta^* \|g_i\|^2, \quad \text{by Lemma 3.1} \\ &\leq -\eta^* \|g_i\|^2, \quad \text{since by the BFGS method, } H_i \text{ is positive definite} \\ &< 0 \quad \textit{since} \quad \eta^* > 0. \end{aligned}$$

Lemma 3.3

The optimal search parameter $\eta^* = \frac{3}{4}$.

Proof

Utilizing Lemma 3.1 in (2.2) yields

$$\beta_i = \frac{-g_{i-1}^T g_i}{g_{i-1}^T d_{i-1}}. \tag{4.4}$$

and thus,

$$g_i^T d_i = \|g_i\|^2 + \beta_i g_i^T d_{i-1} \tag{4.5}$$

Now combining (4.4)the above with the fact that for vectors u and v the inequality

$$u^T v \leq \frac{1}{2}(\|u\|^2 + \|v\|^2)$$

holds, we have that

$$\begin{aligned} \beta_i g_i^T d_{i-1} &= \frac{-g_i^T g_{i-1}.d_{i-1}^T g_{i-1}.g_i^T d_{i-1}}{(g_{i-1}^T d_{i-1})^2} \\ &= \frac{\sqrt{2}}{\sqrt{2}} \frac{-g_i^T g_{i-1}.d_{i-1}^T g_{i-1}.g_i^T d_{i-1}}{(g_{i-1}^T d_{i-1})^2} \\ &= -\frac{1}{\sqrt{2}} \frac{g_i^T d_{i-1}^T g_{i-1}}{g_{i-1} d_{i-1}}.\sqrt{2} \frac{g_{i-1} g_i^T d_{i-1}}{g_{i-1}^T d_{i-1}} \\ &\leq \frac{1}{2}(\| \frac{1}{\sqrt{2}} \frac{g_i^T d_{i-1}^T g_{i-1}}{g_{i-1}^T d_{i-1}} \|^2 + \| \sqrt{2} \frac{g_{i-1} g_i^T d_{i-1}}{g_{i-1}^T d_{i-1}} \|^2, \quad \text{on applying the above inequality} \\ &\leq \frac{1}{2(g_{i-1}^T d_{i-1})^2} \left[(g_{i-1}^T d_{i-1})^2 \frac{1}{(\sqrt{2})^2} \|g_i\|^2 + (\sqrt{2})^2 (g_i^T d_{i-1})^2 \|g_{i-1}\|^2 \right], \quad \text{on simplification} \\ &\leq \frac{1}{4}\|g_i\|^2, \quad \text{using (4.3).} \end{aligned}$$

As such by (4.5),

$$g_i^T d_i \leq -\frac{3}{4}\|g_i\|^2. \tag{4.6}$$

Hence, the result follows from (4.6) and the fourth line in the proof of **Lemma 3.2**.

5 Conclusions

In this section we use a set of selected unconstrained optimization problems from the CUTEr suite [20]. The results obtained using the OBFGS-CG method compared with the BFGS, CG-FR, CG-PR, CG-HS, CG-BAN, BFGS-CG, CG-IBRAH, CG-CD, CG-LS, CG-HZ, CG-DY, and PBFGS-CG methods are tabulated inTables 2 and 3. Each of the test problems is tested with dimensions varying from 2 to 1000. For the Armijo line search, we used $\sigma = 0.1$, the stopping criteria are $||g_i|| \leq 10^{-6}$ and the number of iterations within a limit of 10,000. Performance profile were drawn for the above methods. In general $p(\tau)$ is the fraction of problems with performance ratio τ; thus, a solver with high values of $p(\tau)$ is preferable.

In the implementation, numerical tests were performed on Compaq Presario CQ57-339WM Notebook PC, Windows 7 operating system, and Matlab 2013.

5.1 Remarks on Computational Results

Performance profiles of the methods are illustrated in above figures showing the relative performance of the methods on a set of selected test problems.

From Figs. 1 and 2, the OBFGS-CG method has the best performance both in terms of number of iteration and CPU time since it has the performance metrics (96% and 99%) compared with the BFGS-CG (94% and 97%), PBFGS-CG (91% and 95%), BFGS (82% and 85%), CG-HS (72% and 71%), CG-PR (64% and 56%), CG-FR (57% and 58%), CG-BAN (92% and 84%), CG-DY (89% and 58%), CG-HZ (84% and 82%), CG-IBRAH (83% and 80%), CG-CD (66% and 67%), and CG-LS (49% and 54%).

The computational results show that global convergence was achieved from different starting points on the selected unconstrained optimization problems.

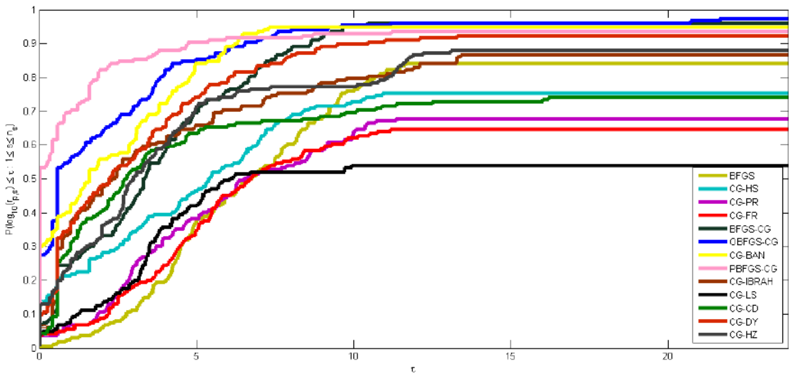

Fig. 1. Performance Profile in a log_{10} scale based on iteration

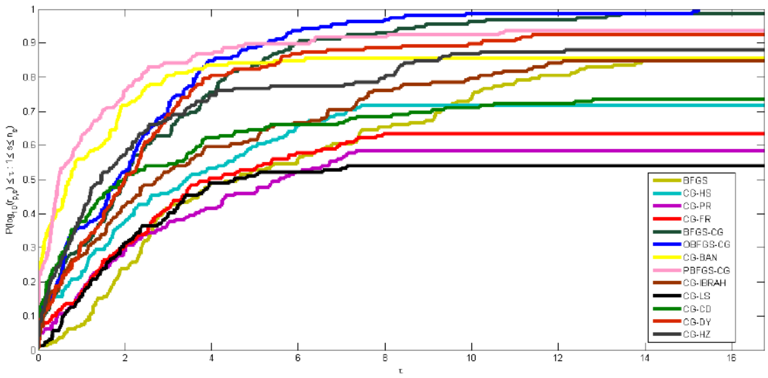

Fig. 2. Performance Profile in a log_{10} scale based on CPU time

Table 1. A list of selected test problems

Test Problem	Dimension	Source
Powell badly scaled	2	[21]
Beale	2	[21]
Biggs Exp	6 6	[21]
Chebyquad	4 6	[21]
Colville polynomial	4	[21]
ARWhead	2, 4	[21]
Freudenstein and Roth	2	[21]
Goldstein price polynomial	2	[22]
Himmelblau	2	[23]
Extended EP1	1 2 4 10 100	[21]
Extended Powell singular	4, 8	[21]
Extended Rosenbrock	2, 10, 100, 200, 500, 1000	[21]
Extended Hebert	2 4 10	[23]
Extended Cliff	2, 4, 10	[21]
Six-hump camel back polynomial	2	[22]
Extended Quadratic Penalty QP1	2, 4, 10, 100, 200, 500, 1000	[1]
Raydan 1	2, 4,	[23]
Raydan 2	2, 4 10 100 200	[23]
De jong	2	[23]
Diagonal 9	2 4 10	[23]
PS1	2	[23]
Cube	2, 10, 100, 200	[21]

Table 2. Number of iterations

PROB	X0	BFGS	CG-HS	CG-PR	CG-FR	BFGS-CG	IBFGS-CG	CG-BAN	PBFGS-CG	CG-IBRAH	CG-LS	CG-CD	CG-DY	CG-HZ
Powell B-S funct N=2	[10 10];	26	33	33	14	12	36	3	3	9	35	75	6	17
Powell B-S funct N=2	[100.,100.];	34	3	33	17	16	4	3	3	4	24	69	4	20
Powell B-S funct N=2	[1000.,1000]	37	3	39	19	19	48	3	3	4	28	76	4	22
Beale funct N=2	[3,3];	21	5	NaN	NaN	6	64	10	NaN	106	36	2391	24	17
Beale funct N=2	[30, 30];	NaN	6	NaN	NaN	6	21	8	NaN	27	27	NaN	1529	9
Beale funct N=2	[15,15];	37	6	NaN	NaN	30	742	8	NaN	15	15	9974	140	8
Biggs EXP6 funct, N =6.	[22]'	289	8	NaN	9	47	5	5	4	4198	NaN	661	5	2
Biggs EXP6 funct, N ¿2.	[15,,15]'	249	NaN	NaN	NaN	40	23	NaN	6	23	8	8	6	2
Biggs EXP6 funct, N ¿2.	[50,..,50]'	NaN	NaN	NaN	NaN	234	28	NaN	2	3	NaN	3	3	2
Chebyquad funct, (N=4)	[10,...,10]	59	3	3	3	12	10	3	3	2	3	NaN	2	2
Chebyquad funct, (N=4)	[100,,100]	77	1	1	1	9	13	1	1	1	1	1	1	1
Chebyquad funct, (N=4)	[1000,,1000]	89	1	1	1	11	16	1	1	1	1	1	1	1
Chebyquad funct, (N=6)	[10,,10]	64	3	NaN	NaN	43	12	3	3	11	NaN	NaN	2	38
Chebyquad funct, (N=6)	[100,,100]	100	3	NaN	NaN	21	11	3	3	16	NaN	23	2	13
Chebyquad funct, (N=6)	[1000,,1000]	NaN	NaN	NaN	NaN	56	18	3	3	11	NaN	NaN	2	14
Colville poly,(N = 4)	[10,,10]	94	7	NaN	NaN	120	4652	5	2	173	NaN	9	NaN	NaN
Colville poly, (N =4)	[100,,100]	95	NaN	NaN	NaN	245	3453678	NaN	2	NaN	NaN	14	NaN	NaN
Colville poly,(N = 4)	[50,,50]	41	13	NaN	NaN	1232	6574356	145	2		NaN	19	NaN	NaN
freud and roth funct , N=2	[2,2]	248	7	109	64	14	35	7	6	17	69	34660	99	61
freud and roth funct , N=2	[10,10]	19	82	1773	706	20	920	51	53	14	NaN	11925	51	141
freud and roth funct , N=2	[200,200]	22	16	196	100	21	295	9	7	17120	192	13375	65	68
Goldstein Price poly' (N=2)	[20,20]	42	6	2	2	20	3	3	3	3	2	3	134	3
Goldstein Price poly' (N=2)	[100,100.]';	59	3	2	2	20	2	3	3	2	2	2	424	3
Goldstein Price poly' (N=2)	[1000,1000]	32	2	2	2	23	2	2	2	2	2	2	2	3
Himmelblau funct, (N = 2)	[200,200]	28	32	22	156	17	73	6	6	19	31	31	106	28
Himmelblau funct, (N = 2)	[500,500]	26	19	34	23	14	4	29	11	39723	73	34	23	23
Himmelblau funct, (N = 2)	[1000,1000]	30	31	31	78	15	4	32	10	33	24	79	6	25
Powell s-q funct, N=4.	[2,...,2]	52	3	NaN	NaN	43	4	3	NaN	25	NaN	5	3	5
Powell s-q funct, N=4.	[100,,100]	60	6	NaN	NaN	50	12	6	6	10	NaN	59	6	NaN
Powell s-q funct, N=4.	[150,,150]	73	NaN	NaN	NaN	45	19	6	6	10	NaN	17	9	NaN
Powell s-q funct, N=8.	[100,,100]	110	6	NaN	NaN	106	12	6	6	10	NaN	59	6	NaN
Powell s-q funct, N=8.	[1000,,1000]	111	NaN	NaN	NaN	103	13	6	6	12	NaN	13	NaN	NaN
Rosenbrock , N=2	[20,20]	61	6	NaN	NaN	53	14	5	112	48	NaN	8	6	5
Rosenbrock funct N=2	[50,50]	148	7	NaN	NaN	167	14	5	306	7	NaN	6	6	7
Rosenbrock funct N=2	[1000,1000]	2571	136	NaN	NaN	1002	16	13	88	NaN	NaN	119	12	11
De Jong funct F2' (N=2)	[35,35]	48	2	NaN	NaN	34	3	2	2	3	NaN	3	2	2
De Jong funct F2' (N=2)	[350,350]	NaN	2	NaN	NaN	1604	3	2	2	3	NaN	3	2	2

PROB	X0	BFGS	CG-HS	CG-PR	CG-FR	BFGS-CG	IBFGS-CG	CG-BAN	PBFGS-CG	CG-IBRAH	CG-LS	CG-CD	CG-DY	CG-HZ
Six-Hump C-B, (N=2)	[30,30]	2653	2	NaN	NaN	1604	3	2	3	11	NaN	5	4098	17478
Six-Hump C-B, (N=2)	[120,120]	37	3	NaN	NaN	28	5	3	3	10	NaN	7	NaN	NaN
Six-Hump C-B (N=2)	[750,750]	56	3	NaN	NaN	30	23	3	3	15	NaN	20	NaN	NaN
Raydan 2, N=2	[20,20]	60	3	NaN	NaN	30	28	3	4	4	27	13	11	9
Raydan 2, N=2	[50,50]	26	4	22	14	53	4	11	3	4	31	19	14	12
Raydan 2, N=2	[200,200]	NaN	17	15	17	24	4	3	3	4	56	31	36	35
Raydan 2, N=4	[15,..,15]	NaN	3	45	39	10	4	3	17	4	29	7	12	8
Raydan 2, N=4	[50,,50]	29	4	23	14	6	4	7	3	4	31	19	14	12
Raydan 2, N=4	[200,,200]	92	3	15	17	20	4	3	3	4	56	31	36	32
Raydan 1 , N=2	[20,20]	NaN	3	45	39	60	4	3	3	4	55	22	23	20
Raydan 1 , N=2	[50,50]	30	4	34	21	23	1	11	3	4	54	34	27	29
Raydan 1 , N=2	[200,200]	NaN	3	3	32	40	1	3	3	6	105	64	74	71
Raydan 1,N=4	[5,,5]	NaN	3	101	74	120	1	3	3	4	24	13	15	7
Raydan 1,N=4	[50,,50]	27	3	13	15	20	1	3	3	4	51	33	36	38
Raydan 1,N=4	[100,,100]	NaN	3	58	37	56	1	3	3	4	75	76	64	76
PSC 1 funct N=2	[5,5]	NaN	3	76	82	120	1	3	26	860	35	38	30	18
PSC 1 funct N=2	[100,100]	22	22	36	24	23	11	12	36	619	38	41	595	21
PSC 1 funct N=2	[500,500]	36	29	38	437	25	34	75	71	619	30	50	45	20
Cube funct,(N=2)	[4,4]	36	40	41	174	26	63	48	24	NaN	83	62899	65	90
Cube funct,(N=2)	[40,40]	298	50	135	485	200	8	7	91	3012	262	NaN	61	275
Cube funct,(N=2)	[100,100]	127	2567	171	1023	167	11	345	279	NaN	797	NaN	1324	335
Cube funct	[50,,50]	NaN	NaN	317	NaN	234	16	56	71	NaN	351	NaN	1178	278
Cube funct	[80,80]	NaN	393	NaN	458	23	4	23	55	NaN	230	NaN	893	413
Ext Rosenbrock	[100,,100]	112	3462	4045	NaN	34	4	67	48	7	NaN	NaN	5	5
Ext Rosenbrock, (N=10)	[20,,20]	NaN	7741	NaN	NaN	20	4	74	167	8	NaN	8	6	5
Ext Rosenbrock, (N=10)	[1000,,1000]	1746	1338	NaN	3415	1324	1342	45	46	54	NaN	1325	12	10
Ext Raydan2, (N=10)	[20,..,20]	6141	344	NaN	NaN	1056	671	12	4	4	29	13	11	9
Ext Raydan2, (N=10)	[50,..,50]	8518	NaN	NaN	NaN	772	543	32	3	4	32	19	14	12
Ext Raydan2, (N=10)	[100,..,100]	624	126	7850	NaN	23	4	4	3	6	27	23	32	19
Cube funct, (N=100)	[10,,10]	NaN	511	1678	2558	16	4	3	NaN	NaN	NaN	NaN	NaN	NaN
Cube funct, (N=100)	[55,,55]	1492	1039	NaN	6877	45	4	56	NaN	NaN	NaN	NaN	NaN	NaN
Cube funct, (N=100)	[300,,300]	3095	134	410	113	1789	1765	285	NaN	NaN	NaN	NaN	NaN	NaN
Ext Raydan2, (N=100)	[20,..,20]	2211	158	292	124	1579	2341	345	4	12	30	14	12	9
Ext Raydan2, (N=100)	[50,..,50]	2070	162	615	146	430	2316	476	3	14	34	17	15	12
Ext Raydan2, (N=100)	[100,..,100]	1780	NaN	17	25	1450	4	423	3	7	32	28	33	19
Cube funct, (N=200)	[10,,10]	1785	NaN	22	15	301	14	321	NaN	NaN	NaN	NaN	NaN	NaN
Cube funct, (N=200)	[55,,55]	4445	NaN	27	35	2654	16	896	NaN	NaN	NaN	NaN	NaN	NaN
Cube funct, (N=200)	[300,,300]	1233	502	1459	1240	10	4	43	NaN	NaN	NaN	NaN	NaN	NaN
Ext Rosenbrock, (N=200)	[55,,55]	2285	NaN	NaN	371	19	4	235	3	5	NaN	2307	4	4
Ext Rosenbrock, (N=200)	[75,,75]	NaN	NaN	NaN	NaN	6	4	74	4	6	NaN	4392	33	8
Ext Rosenbrock, (N=200)	[125,,125]	2408	994	NaN	NaN	234	4	29	9375	8	NaN	8	9	8
Ext Raydan2 , (N=200)	[20,..,20]	3594	454	NaN	1637	2611	6	86	6	5	31	14	12	9
Ext Raydan2 , (N=200)	[50,..,50]	4217	1336	NaN	2688	2433	44	74	3	7	34	18	15	12
Ext Raydan2 , (N=200)	[75,..,75]	727	590	3903	597	132	368	37	4	39	39	21	34	14
Ext Rosenbrock, (N=500)	[55,,55]	359	525	471	787	199	144	54	3	5	NaN	76	5	5

Table 3. Number of iterations

PROB	X0	BFGS	CG-HS	CG-PR	CG-FR	BFGS-CG	IBFGS-CG	CG-BAN	PBFGS-CG	CG-IBRAH	CG-LS	CG-CD	CG-DY	CG-HZ
Ext Rosenbrock, (N=500)	[75,,75]	262	1577	NaN	1695	NaN	N	485	3401	15	NaN	15	44	8
Ext Rosenbrock, (N=500)	[125,,125]	1076	133	146	114	NaN	N	742	6	8	NaN	8	8	10
Ext Rosenbrock, (N=1000)	[100,,100]	1170	145	4052	134	NaN	N	109	4	23	NaN	20	5	5
Ext Rosenbrock, (N=1000)	[125,,125]	128	157	9978	199	NaN	N	46	29	26	NaN	8	6	8
Ext Cliff funct (N=2)	[10,..,10]	4313	NaN	17	26	506	464	104	5	39	64	23	64	14
Ext Cliff funct (N=2)	[50,50]	8314	NaN	23	15	148	243	36	5	235	55	NaN	55	131
Ext Cliff funct (N=2)	[100,..100]	298	NaN	28	34	171	93	42	5	694	68	NaN	115	127
Ext Cliff funct (N=2)	[500,..,500]	127	500	NaN	3745	75	33	60	7	322	66	NaN	270	79
Ext Cliff funct (N=2)	[1000,...1000]	NaN	NaN	NaN	NaN	56	32	56	5	25	130	4	66	64
Ext Cliff funct (N=4)	[10,..,10]	122	NaN	NaN	NaN	408	312	104	5	679	64	23	35	110
Ext Cliff funct (N=4)	[50,50]	273	1610	NaN	5627	148	62	36	5	235	57	NaN	57	166
Ext Cliff funct (N=4)	[100,..100]	393	1703	NaN	NaN	522	93	42	5	694	70	NaN	115	155
Ext Cliff funct (N=4)	[500,..,500]	48	625	NaN	NaN	43	125	60	7	322	77	NaN	270	123
Ext Cliff funct (N=4)	[1000,...1000]	92	112	4329	174	115	127	56	5	25	146	4	66	52
Ext Cliff funct (N=10)	[10,..,10]	109	632	NaN	1679	917	115	134	5	210	72	23	41	38
Ext Cliff funct (N=10)	[50,50]	112	929	NaN	NaN	505	59	144	5	234	67	NaN	57	120
Ext Cliff funct (N=10)	[100,..100]	112	130	392	118	181	342	42	5	822	75	NaN	115	140
Ext Cliff funct (N=10)	[500,..,500]	NaN	141	2696	139	38	94	60	7	1333	65	NaN	285	69
Ext Cliff funct (N=10)	[1000,...1000]	1746	182	4023	178	28	35	56	5	25	156	4	116	61
Ext Hiebert funct (N=2)	[10,..,10]	6141	NaN	18	27	30	3	244	4	50	NaN	NaN	64	7300
Ext Hiebert funct (N=2)	[50,50]	8518	NaN	23	15	49	3	92	11	13092	2513	4	1331	5280
Ext Hiebert funct (N=2)	[100,..100]	NaN	NaN	19	23	4	3	160	7	30239	NaN	NaN	30239	7336
Ext Hiebert funct (N=2)	[500,..,500]	2178	3673	NaN	6983	123	3	11	4	1475	NaN	NaN	1475	14176
Ext Hiebert funct (N=2)	[1000,...1000]	1337	3955	3955	NaN	66	3	180	9	4191	NaN	NaN	4191	7855
Ext Hiebert funct (N=4)	[10,..,10]	257	4959	NaN	NaN	161	3	244	4	50	NaN	NaN	50	6154
Ext Hiebert funct (N=4)	[50,50]	912	132	1735	236	555	3	92	11	13092	2513	4	13092	7945
Ext Hiebert funct (N=4)	[100,..100]	9672	325	699	4334	4	3	160	7	30239	NaN	NaN	867	8321
Ext Hiebert funct (N=4)	[500,..,500]	76	1539	NaN	NaN	587	5	11	4	1475	NaN	NaN	13	13936
Ext Hiebert funct (N=4)	[1000,...1000]	525	132	156	123	2408	6	180	1972	4191	NaN	NaN	1399	6601
Ext Hiebert funct (N=10)	[10,..,10]	282	134	836	137	262	3	266	4	50	NaN	NaN	36	6608
Ext Hiebert funct (N=10)	[50,50]	624	167	NaN	295	713	3	92	11	11377	2931	4	1323	9052
Ext Hiebert funct (N=10)	[100,..100]	NaN	180	24	NaN	4	3	158	7	32266	NaN	NaN	125	9968
Ext Hiebert funct (N=10)	[500,..,500]	1492	369	60	17	2141	5	11	4	1651	NaN	NaN	13	15087
Ext Hiebert funct (N=10)	[1000,...1000]	3095	863	21	NaN	2340	6	240	8	19154	NaN	NaN	965	9154
Ext Quad-P QP1 funct (N=2)	[10,..,10]	2211	NaN	1475	48	5899	113	45	32	3970	29	64	51	27
Ext Quad-P QP1 funct (N=2)	[50,50]	2070	NaN	257	131	286	66	NaN	60	1065	36	41	35	35
Ext Quad-P QP1 funct (N=2)	[100,..100]	1780	24	106	80	NaN	192	NaN	43	2031	33	145	89	25
Ext Quad-P QP1 funct (N=2)	[500,..,500]	1785	58	1926	970	68	1146	29	6	NaN	32	42	31	42
Ext Quad-P QP1 funct (N=2)	[1000,...1000]	4445	63	2183	NaN	8394	1572	NaN	6	NaN	30	160	44	26
Ext EP1 funct(N=2)	[10,..,10]	440	43	NaN	NaN	3	3	2	2	3	NaN	3	3	27
Ext EP1 funct(N=2)	[50,50]	1200	1926	14	34	3	3	2	2	3	NaN	3	3	35

Table 4. Number of iterations

PROB	X0	BFGS	CG-HS	CG-PR	CG-FR	BFGS-CG	IBFGS-CG	CG-BAN	PBFGS-CG	CG-IBRAH	CG-LS	CG-CD	CG-DY	CG-HZ
Ext EP1 funct(N=2)	[100,..100]	2073	NaN	18	35	3	3	2	2	3	NaN	3	3	25
Ext EP1 funct(N=2)	[500,..,500]	276	NaN	12	36	3	3	2	2	3	NaN	3	3	42
Ext EP1 funct(N=2)	[1000,...1000]	392	NaN	13	183	3	3	2	2	3	NaN	3	3	26
Ext EP1 funct(N=4)	[10,..,10]	413	13	15	NaN	3	3	2	2	3	NaN	3	3	8
Ext EP1 funct(N=4)	[50,50]	1233	NaN	19	113	3	3	2	2	3	NaN	3	3	16
Ext EP1 funct(N=4)	[100,..100]	2285	NaN	2762	1254	3	3	2	2	3	NaN	3	3	12
Ext EP1 funct(N=4)	[500,..,500]	NaN	NaN	778	990	3	3	2	2	3	NaN	3	3	6344
Ext EP1 funct(N=4)	[1000,...1000]	2408	NaN	1141	1646	3	3	2	2	3	NaN	3	3	13
Ext EP1 funct(N=10)	[10,..,10]	3594	298	91	216	3	3	2	2	3	NaN	3	3	10
Ext EP1 funct(N=10)	[50,50]	4217	237	102	110	3	3	2	2	3	NaN	3	3	16
Ext EP1 funct(N=10)	[100,..100]	647	262	83	270	3	3	2	2	3	NaN	3	3	12
Ext EP1 funct(N=10)	[500,..,500]	1552	17	182	90	3	3	2	2	3	NaN	3	3	13
Ext EP1 funct(N=10)	[1000,...1000]	NaN	22	161	85	3	3	2	2	3	NaN	3	3	156
Ext EP1 funct(N=100)	[10,..,10]	220	197	137	69	3	3	2	2	3	104	3	3	5
Ext EP1 funct(N=100)	[50,50]	727	82	1058	NaN	3	3	2	2	3	86	3	3	17
Ext EP1 funct(N=100)	[100,..100]	359	79	155	366	3	3	2	2	3	NaN	3	3	13
Ext EP1 funct(N=100)	[500,..,500]	26	60	100	209	3	3	2	2	3	NaN	3	3	17
Ext EP1 funct(N=100)	[1000,...1000]	34	37	790	NaN	3	3	2	2	3	2	3	3	NaN
Diagonal 9 funct (N=2)	[50,50]	37	NaN	5304	175	4	4	38	7	NaN	104	298103	74	105
Diagonal 9 funct (N=2)	[100,..100]	21	40	3325	NaN	4	4	54	107	54	86	266351	471	205
Diagonal 9 funct (N=2)	[500,..,500]	NaN	NaN	69	24	2	2	2	2	2	NaN	3	2	2
Diagonal 9 funct (N=2)	[1000,...1000]	37	NaN	25	NaN	1	1	2	1	2	2	2	2	2
Diagonal 9 funct (N=4)	[50,50]	289	NaN	107	31	1	4	60	11	NaN	67	NaN	64	163
Diagonal 9 funct (N=4)	[100,..100]	249	36	20	NaN	4	4	9	4	88	105	NaN	2826	125
Diagonal 9 funct (N=4)	[500,..,500]	NaN	46	17	NaN	2	2	2	2	2	NaN	3	3	2
Diagonal 9 funct (N=4)	[1000,...1000]	59	33	12	NaN	1	1	2	1	2	2	2	2	2
Diagonal 9 funct (N=10)	[50,50]	77	NaN	18	NaN	4	4	112	7	NaN	141	NaN	137	129
Diagonal 9 funct (N=10)	[100,..100]	89	NaN	17	NaN	4	4	68	4	88	150	NaN	475	77
ARWHEAD funct(N=2)	[10,..,10]	64	NaN	19	NaN	50	51	46	29	NaN	46	45	27	19
ARWHEAD funct(N=2)	[50,50]	100	11	587	267	75	47	100	34	23	45	19	24	25
ARWHEAD funct(N=2)	[100,..100]	NaN	NaN	689	437	66	60	43	19	62	43	35	31	32
ARWHEAD funct(N=2)	[500,..,500]	94	NaN	476	380	52	98	45	29	83	52	31	26	25
ARWHEAD funct(N=2)	[1000,...1000]	95	173	574	301	61	57	34	42	57	43	50	55	24
ARWHEAD funct(N=4)	[10,..,10]	41	489	460	320	NaN	45	16	6	NaN	NaN	91	17	NaN
ARWHEAD funct(N=4)	[50,50]	24	211	637	509	93	110	NaN	10	NaN	NaN	NaN	22	NaN
ARWHEAD funct(N=4)	[100,..100]	28	205	190	793	431	156	NaN	36	12930	NaN	72	47	NaN

Table 5. CPU-time second

PROB	X0	BFGS	CG-HS	CG-PR	CG-FR	BFGS-CG	IBFGS-CG	CG-BAN	PBFGS-CG	CG-IBRAH	CG-LS	CG-CD	CG-DY	CG-HZ
Powell B-S funct N=2	[10 10];	0.0468	0.0721	0.1231	0.0263	0.0808	0.1057	0.1015	0.0024	0.0269	0.0769	0.089	0.0164	0.0295
Powell B-S funct N=2	[100.,100.];	0.0624	0.0126	0.069	0.0274	0.0211	0.0264	0.0021	0.0154	0.0169	0.0479	0.08424	0.0143	0.0357
Powell B-S funct N=2	[1000.,1000]	0.0468	0.0135	0.0859	0.0309	0.0206	0.011	0.0136	0.0152	0.0174	0.0525	0.08687	0.0144	0.037
Beale funct N=2	[3,3];	0.078	0.0457	NaN	NaN	0.0491	0.0996	0.0125	NaN	0.4973	0.0642	1.51374	0.0314	0.5085
Beale funct N=2	[30, 30];	NaN	0.0233	NaN	NaN	0.0383	0.1662	0.1713	NaN	0.1186	0.1219	NaN	2.3623	0.0312
Beale funct N=2	[15,15];	0.0624	0.0277	NaN	NaN	0.0295	0.0669	0.0326	NaN	0.0601	0.0512	7.94324	0.2005	0.0292
Biggs EXP6 funct, N =6.	[22]'	0.078	0.0368	NaN	0.1991	0.0186	0.4119	0.0242	0.0255	26.714	NaN	2.99742	0.0289	0.0204
Biggs EXP6 funct, N ¿2.	[15,,15]'	0.0936	NaN	NaN	NaN	0.0568	0.6784	NaN	0.0129	0.1363	0.0162	0.03135	0.0338	0.0227
Biggs EXP6 funct, N ¿2.	[50,..,50]'	NaN	NaN	NaN	NaN	0.0232	0.9865	NaN	0.0197	0.0071	NaN	0.00735	0.0198	0.0261
Chebyquad funct, (N=4)	[10,...,10]	0.078	0.0105	0.0127	0.0425	0.0624	0.1532	0.0109	0.0133	0.0107	0.0128	NaN	0.0103	0.0116
Chebyquad funct, (N=4)	[100,,100]	0.0624	0.0076	0.0075	0.0073	0.0624	0.0225	0.0077	0.0097	0.0081	0.0074	0.00825	0.0076	0.0081
Chebyquad funct, (N=4)	[1000,,1000]	0.0936	0.0076	0.0074	0.0073	0.0468	0.0232	0.0081	0.0093	0.0261	0.008	0.00809	0.0073	0.0081
Chebyquad funct, (N=6)	[10,,10]	0.0624	0.0108	NaN	NaN	0.0156	0.0281	0.0112	0.0117	0.0157	NaN	NaN	0.0105	0.0714
Chebyquad funct, (N=6)	[100,,100]	0.078	0.0112	NaN	NaN	0.0312	0.0235	0.0107	0.0141	0.0174	NaN	0.03264	0.0104	0.0215
Chebyquad funct, (N=6)	[1000,,1000]	NaN	NaN	NaN	NaN	0.0312	0.0298	0.0107	0.0138	0.0169	NaN	NaN	0.011	0.0267
Colville poly,(N = 4)	[10,,10]	0.0468	0.0216	NaN	NaN	0.078	0.10938	0.0254	0.0087	1.4567	NaN	0.04785	NaN	NaN
Colville poly, (N =4)	[100,,100]	0.078	NaN	NaN	NaN	0.0312	1100.67	NaN	0.042	NaN	NaN	0.0766	NaN	NaN
Colville poly,(N = 4)	[50,,50]	0.0468	0.0608	NaN	NaN	0.0312	336.212	0.6222	0.0087	NaN	NaN	0.10788	NaN	NaN
freud and roth funct , N=2	[2,2]	0.1092	0.0417	0.1886	0.084	0.0226	0.0171	0.0642	0.0348	0.0547	0.1227	24.9489	0.1308	0.0762
freud and roth funct , N=2	[10,10]	0.0624	0.1161	2.1783	1.3452	0.0541	0.1913	0.0589	0.0563	0.0578	NaN	7.236	0.0599	0.1512
freud and roth funct , N=2	[200,200]	0.0624	0.0405	0.3411	0.1108	9.042	6.3771	0.0228	0.0183	49.715	0.276	12.2207	0.0611	0.0882
Goldstein Price poly' (N=2)	[20,20]';	0.078	0.136	0.0134	0.0134	0.0171	0.002	0.0427	0.1706	0.0439	0.057	0.05834	4.2065	0.016
Goldstein Price poly' (N=2)	[100,100.]';	0.0624	0.0556	0.0129	0.0132	0.0177	0.0166	0.0565	0.0505	0.0163	0.0144	0.013	12.434	0.016
Goldstein Price poly' (N=2)	[1000,1000]	0.0468	0.0131	0.0015	0.0129	0.0169	0.0167	0.0134	0.0166	0.0134	0.0136	0.01402	0.0136	0.016
Himmelblau funct, (N = 2)	[200,200]	0.0468	0.0303	0.03	0.1108	0.0779	0.0798	0.0184	0.0212	0.0555	0.0495	0.02836	0.0742	0.0269
Himmelblau funct, (N = 2)	[500,500]	0.0468	0.0301	0.0551	0.0242	0.0205	0.0207	0.0316	0.0221	89.063	0.0944	0.02889	0.0268	0.0265
Himmelblau funct, (N = 2)	[1000,1000]	0.0468	0.0483	0.0378	0.0713	0.0209	0.0207	0.0318	0.0251	0.1306	0.0333	0.05434	0.0467	0.0275
Powell s-q funct, N=4.	[2,...,2]	0.078	0.0468	NaN	NaN	0.0468	0.03946	0.0413	NaN	0.1422	NaN	0.02003	0.0175	0.0119
Powell s-q funct, N=4.	[100,,100]	0.0624	0.026	NaN	NaN	0.0468	0.07634	0.0259	0.0316	0.0657	NaN	0.32055	0.0336	NaN
Powell s-q funct, N=4.	[150,,150]	0.0936	NaN	NaN	NaN	0.0312	0.11722	0.0257	0.0327	0.0523	NaN	0.09204	0.0504	NaN
Powell s-q funct, N=8.	[100,,100]	0.078	0.0343	NaN	NaN	0.0156	0.08336	0.0273	0.0331	0.0541	NaN	0.3566	0.034	NaN
Powell s-q funct, N=8.	[1000,,1000]	0.0624	NaN	NaN	NaN	0.0312	0.09796	0.0296	0.0364	0.0875	NaN	0.08336	NaN	NaN
Rosenbrock , N=2	[20,20]	0.0312	0.0165	NaN	NaN	0.0195	0.0241	0.0158	0.3851	0.1957	NaN	0.05681	0.058	0.0181
Rosenbrock funct N=2	[50,50]	0.0624	0.0178	NaN	NaN	0.0352	0.0357	0.0158	2.841	0.0106	NaN	0.01801	0.0577	0.0223
Rosenbrock funct N=2	[1000,1000]	0.3744	0.2983	NaN	NaN	0.0338	0.0428	0.0268	0.3065	NaN	NaN	0.35883	0.0664	0.0293
De Jong funct F2' (N=2)	[35,35]	0.0468	0.0144	NaN	NaN	0.0468	0.01976	0.0138	0.0171	0.0142	NaN	0.01501	0.0461	0.0196
De Jong funct F2' (N=2)	[350,350]	NaN	0.0161	NaN	NaN	0.1716	0.02062	0.0136	0.0167	0.016	NaN	0.00426	0.0476	0.0203

Table 6. CPU-time second

PROB	X0	BFGS	CG-HS	CG-PR	CG-FR	BFGS-CG	IBFGS-CG	CG-BAN	PBFGS-CG	CG-IBRAH	CG-LS	CG-CD	CG-DY	CG-HZ
Six-Hump C-B poly, (N=2)	[30,30]	0.0468	0.0145	NaN	NaN	0.0312	0.41187	0.0168	0.0218	0.0384	NaN	0.03144	32.764	66.142
Six-Hump C-B poly, (N=2)	[120,120]	0.0624	0.0351	NaN	NaN	0.0312	0.67836	0.0076	0.0223	0.1092	NaN	0.03521	NaN	NaN
Six-Hump C-B poly'(N=2)	[750,750]	0.0312	0.0195	NaN	NaN	0.1092	0.98646	0.0204	0.0231	0.1236	NaN	0.2037	NaN	NaN
Raydan 2, N=2	[20,20]	0.078	0.0201	NaN	NaN	0.0312	0.01487	0.0204	0.0141	0.0355	0.0189	0.01607	0.0131	0.0138
Raydan 2, N=2	[50,50]	NaN	0.0115	0.0445	0.0142	0.0468	0.0187	0.0154	0.0166	0.0161	0.0233	0.02461	0.0213	0.0216
Raydan 2, N=2	[200,200]	NaN	0.0238	0.0227	0.0238	0.0312	0.03702	0.0126	0.0242	0.0279	0.1289	0.13451	0.1239	0.1351
Raydan 2, N=4	[15,..,15]	0.0468	0.0193	0.1365	0.1487	0.0312	0.01452	0.0194	0.0228	0.0134	0.019	0.0131	0.0132	0.0131
Raydan 2, N=4	[50,,50]	0.0624	0.0155	0.0163	0.0149	0.0624	0.01869	0.012	0.0149	0.0181	0.0257	0.02223	0.0199	0.0205
Raydan 2, N=4	[200,,200]	NaN	0.0141	0.0235	0.025	0.312	0.03759	0.0127	0.0231	0.0284	0.1298	0.13047	0.1243	0.1322
Raydan 1 , N=2	[20,20]	0.0468	0.0213	0.1368	0.1459	0.0312	0.00956	0.021	0.0145	0.0284	0.0112	0.02806	0.0184	0.0185
Raydan 1 , N=2	[50,50]	NaN	0.0142	0.0087	0.0186	0.1092	0.01078	0.0162	0.0152	0.0169	0.0382	0.04779	0.0319	0.036
Raydan 1 , N=2	[200,200]	NaN	0.0143	0.0127	0.0351	0.2652	0.08214	0.0131	0.0291	0.0483	0.2711	0.28871	0.2801	0.2873
Raydan 1,N=4	[5,,5]	0.0624	0.0201	0.3069	0.3043	0.0312	0.00923	0.0208	0.0132	0.014	0.0152	0.01752	0.0158	0.014
Raydan 1,N=4	[50,,50]	NaN	0.0109	0.0131	0.0038	0.0312	0.01161	0.0109	0.0174	0.0171	0.0578	0.04454	0.0527	0.0579
Raydan 1,N=4	[100,,100]	NaN	0.0129	0.0413	0.0575	0.1092	0.01376	0.0542	0.0174	0.0232	0.1652	0.18602	0.1535	0.1937
PSC 1 funct N=2	[5,5]	0.0468	0.0196	0.1742	0.2107	0.0302	0.0195	0.0213	0.0338	3.137	0.05	0.0317	0.0282	0.0231
PSC 1 funct N=2	[100,100]	0.0312	0.0258	0.0504	0.0251	0.0483	0.2041	0.0694	0.0551	2.5756	0.1917	0.03334	0.3908	0.0255
PSC 1 funct N=2	[500,500]	0.0468	0.0372	0.0574	0.3623	0.0433	0.1256	0.0544	0.0916	2.6034	0.0419	0.03832	0.0339	0.0617
Cube funct,(N=2)	[4,4]	0.1248	0.0373	0.0693	0.1178	0.0468	6.70268	0.0243	0.0406	NaN	0.0905	28.3409	0.0462	0.0535
Cube funct,(N=2)	[40,40]	0.4836	1.0296	0.0936	0.3588	0.1872	38.6306	0.0415	0.218	6.8042	0.3041	NaN	0.0541	0.2066
Cube funct,(N=2)	[100,100]	0.468	NaN	0.156	NaN	0.0468	0.03003	NaN	0.5708	NaN	1.0277	NaN	1.351	0.2524
Cube funct	[50,,50]	2.3244	1.3728	1.7784	NaN	0.0936	23.518	NaN	0.1583	NaN	0.3876	NaN	1.063	0.217
Cube funct	[80,80]	4.6956	3.3072	NaN	NaN	0.0624	0.02862	NaN	0.1504	NaN	0.2673	NaN	1.001	0.3239
Ext Rosenbrock	[100,,100]	0.1092	0.4368	NaN	1.17	6.9732	0.04142	0.02249	0.1367	0.0225	NaN	NaN	0.0616	0.0072
Ext Rosenbrock, (N=10)	[20,,20]	0.0468	0.0936	NaN	NaN	0.0156	0.0238	0.01705	0.4233	0.0514	NaN	0.06279	0.0594	0.0191
Ext Rosenbrock, (N=10)	[1000,,1000]	NaN	NaN	NaN	NaN	135.05	0.05197	0.03864	0.1038	0.2097	NaN	3.53716	0.0701	0.0247
Ext Raydan2, (N=10)	[20,..,20]	0.0468	NaN	0.0468	0.0468	145.24	0.01578	NaN	0.0144	0.0143	0.0192	0.01584	0.0131	0.014
Ext Raydan2, (N=10)	[50,..,50]	0.0468	NaN	0.0312	0.0312	136.39	0.02059	NaN	0.0154	0.0165	0.0238	0.0258	0.0199	0.0224
Ext Raydan2, (N=10)	[100,..,100]	NaN	NaN	0.0312	0.0312	6.1152	0.02512	NaN	0.0203	0.0297	0.0413	0.04579	0.0457	0.0422
Cube funct, (N=100)	[10,,10]	2.7924	0.5304	1.248	1.014	9.2041	3.06543	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Cube funct, (N=100)	[55,,55]	2.5896	NaN	NaN	0.2652	8.1433	2.10605	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Cube funct, (N=100)	[300,,300]	28.314	NaN	NaN	NaN	2.5584	2.13247	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Ext Raydan2, (N=100)	[20,..,20]	0.9672	NaN	0.0312	0.0312	19.141	0.07678	NaN	0.0265	0.0245	0.0223	0.01865	0.0159	0.015
Ext Raydan2, (N=100)	[50,..,50]	NaN	NaN	0.0624	0.0156	0.0936	0.07075	NaN	0.0541	0.0419	0.0284	0.02779	0.0227	0.025
Ext Raydan2, (N=100)	[100,..,100]	2.1684	NaN	0.0312	0.0312	0.0936	0.03265	NaN	0.0758	0.043	0.0553	0.06714	0.0608	0.0538
Cube funct, (N=200)	[10,,10]	18.439	0.6396	NaN	5.148	0.2184	0.32345	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Cube funct, (N=200)	[55,,55]	13.447	NaN	NaN	NaN	2.106	0.12347	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Cube funct, (N=200)	[300,,300]	18.315	NaN	NaN	NaN	1.3884	12.3425	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Ext Rosenbrock, (N=200)	[55,,55]	10.156	0.6864	NaN	2.1528	0.0571	0.0926	0.01825	0.0616	0.0266	NaN	6.86588	0.0734	0.0503
Ext Rosenbrock, (N=200)	[75,,75]	9.8125	0.702	NaN	NaN	0.0971	0.0949	0.03456	0.0726	0.0261	NaN	14.1144	0.2358	0.03

Table 7. CPU-time second

PROB	X0	BFGS	CG-HS	CG-PR	CG-FR	BFGS-CG	IBFGS-CG	CG-BAN	PBFGS-CG	CG-IBRAH	CG-LS	CG-CD	CG-DY	CG-HZ
Ext Rosenbrock, (N=200)	[125,,125]	28.143	0.234	NaN	NaN	0.0975	0.0704	0.03242	36.465	0.0327	NaN	0.02297	0.1019	0.0272
Ext Raydan2 , (N=200)	[20,..,20]	6.63	NaN	0.0312	0.0312	0.0936	0.03145	0.01224	0.0797	0.0198	0.1039	0.00986	0.0068	0.0158
Ext Raydan2 , (N=200)	[50,..,50]	14.742	NaN	0.0156	0.0156	0.0936	0.06252	0.01437	0.0644	0.0313	0.033	0.03295	0.0273	0.0263
Ext Raydan2 , (N=200)	[75,..,75]	NaN	NaN	0.0312	0.0312	0.2184	0.04337	0.01602	0.0812	0.0473	0.0473	0.04526	0.0456	0.0437
Ext Rosenbrock, (N=500)	[55,,55]	135.07	1.8876	NaN	3.4164	0.1371	0.157	0.025	0.1137	0.102	NaN	0.44938	0.025	0.0263
Ext Rosenbrock, (N=500)	[75,,75]	199.2	1.9032	1.9032	NaN	0.4794	0.3213	0.04651	54.306	0.1619	NaN	0.15501	0.2452	0.0314
Ext Rosenbrock, (N=500)	[125,,125]	227.7	2.5272	NaN	NaN	0.5213	0.2349	0.03947	0.2275	0.0397	NaN	0.06048	0.1204	0.0408
Ext Rosenbrock, (N=1000)	[100,,100]	34.32	0.0624	0.39	0.078	1.0501	0.7344	0.05336	0.4849	0.1583	NaN	0.10517	0.1376	0.0296
Ext Rosenbrock, (N=1000)	[125,,125]	16.957	0.1248	NaN	0.1716	1.0898	0.7205	0.02951	2.2597	0.1955	NaN	0.03355	0.1469	0.0357
Ext Cliff funct (N=2)	[10,..,10]	0.1248	0.0624	0.1092	0.1404	0.9891	1.2983	0.1066	0.0192	0.0473	0.05	0.02697	0.0621	0.0297
Ext Cliff funct (N=2)	[50,50]	0.4836	1.02961	0.0936	0.3588	0.3361	0.6865	0.0279	0.0191	0.6415	0.0564	NaN	0.0571	0.1075
Ext Cliff funct (N=2)	[100,..100]	0.468	NaN	0.156	NaN	0.3447	0.2699	0.0334	0.0196	1.7991	0.0583	NaN	0.0618	0.0706
Ext Cliff funct (N=2)	[500,..,500]	0.0156	0.156	NaN	0.156	0.1492	0.13	0.0557	0.0223	0.916	0.0614	NaN	0.1204	0.0354
Ext Cliff funct (N=2)	[1000,...1000]	2.32442	1.37281	1.77841	NaN	0.1167	0.0667	0.0501	0.0187	0.0703	0.1099	0.01375	0.0396	0.0422
Ext Cliff funct (N=4)	[10,..,10]	4.69563	3.30722	NaN	NaN	0.8544	0.8114	0.072	0.0204	2.0793	0.0657	0.02816	0.0259	0.0674
Ext Cliff funct (N=4)	[50,50]	0.1092	0.4368	NaN	1.17001	0.3494	0.1944	0.0297	0.019	0.6674	0.0367	NaN	0.0225	0.1021
Ext Cliff funct (N=4)	[100,..100]	0.0468	0.0936	NaN	NaN	1.1642	0.2842	0.0351	0.0195	1.9354	0.0665	NaN	0.0705	0.096
Ext Cliff funct (N=4)	[500,..,500]	NaN	NaN	NaN	NaN	0.078	0.3938	0.0429	0.0267	0.9303	0.0714	NaN	0.1321	0.0738
Ext Cliff funct (N=4)	[1000,...1000]	0.0468	0.0468	2.27762	NaN	0.2507	0.3621	0.0534	0.0189	0.067	0.1369	0.01416	0.0382	0.039
Ext Cliff funct (N=10)	[10,..,10]	0.0936	0.1716	0.4212	0.7644	1.7763	0.2507	0.1113	0.0211	0.701	0.0786	0.02972	0.03	0.0347
Ext Cliff funct (N=10)	[50,50]	0.1404	0.2808	NaN	1.96561	1.1805	0.155	0.1023	0.0209	0.7151	0.0649	NaN	0.0358	0.0936
Ext Cliff funct (N=10)	[100,..100]	0.0312	0.0468	0.1092	0.0468	0.4081	1.1131	0.035	0.0207	2.1149	0.0728	NaN	0.0668	0.1209
Ext Cliff funct (N=10)	[500,..,500]	0.0312	0.0468	0.1092	0.0624	0.0667	0.3078	0.0567	0.0232	3.5015	0.064	NaN	0.1405	0.0467
Ext Cliff funct (N=10)	[1000,...1000]	0.0156	0.0624	0.1872	0.0624	0.0587	0.0801	0.0532	0.0198	0.0789	0.1429	0.01532	0.0608	0.0477
Ext Hiebert funct (N=2)	[10,..,10]	0.0468	NaN	0.0468	0.0468	0.1001	0.2327	0.1071	0.0195	0.1411	NaN	NaN	0.0519	6.4004
Ext Hiebert funct (N=2)	[50,50]	0.0468	NaN	0.0312	0.0312	0.1424	0.169	0.0824	0.0324	28.685	2.9442	0.0498	1.4723	4.1492
Ext Hiebert funct (N=2)	[100,..100]	NaN	NaN	0.0312	0.0312	0.0534	0.1529	0.1369	0.0234	58.94	NaN	NaN	62.683	7.1141
Ext Hiebert funct (N=2)	[500,..,500]	2.79242	0.5304	1.24801	1.01401	0.2681	0.1472	0.0205	0.0185	3.5996	NaN	NaN	3.4281	11.168
Ext Hiebert funct (N=2)	[1000,...1000]	2.58962	NaN	NaN	0.2652	0.1389	0.1469	0.152	0.0322	9.4824	NaN	NaN	10.714	7.584
Ext Hiebert funct (N=4)	[10,..,10]	28.3142	NaN	NaN	NaN	0.4315	0.0268	0.2146	0.0197	0.1394	NaN	NaN	0.1655	6.2392
Ext Hiebert funct (N=4)	[50,50]	NaN	0.4992	NaN	NaN	1.16	0.0253	0.0877	0.033	29.328	3.2176	0.0589	30.534	5.8107
Ext Hiebert funct (N=4)	[100,..100]	3.33842	0.1872	NaN	0.6396	0.0199	0.027	0.155	0.0226	66.193	NaN	NaN	0.9705	6.2766
Ext Hiebert funct (N=4)	[500,..,500]	1.82521	0.6084	NaN	1.01401	1.1907	0.0331	0.0211	0.0185	3.619	NaN	NaN	0.0225	11.824
Ext Hiebert funct (N=4)	[1000,...1000]	0.3432	0.0624	1.27921	0.0312	3.9157	0.0377	0.1628	2.6015	10.257	NaN	NaN	1.5246	4.8857
Ext Hiebert funct (N=10)	[10,..,10]	1.23241	0.1872	0.1404	0.2496	0.5897	0.0283	0.2326	0.0196	0.1459	NaN	NaN	0.0452	5.0283
Ext Hiebert funct (N=10)	[50,50]	18.9385	0.6552	NaN	0.546	1.5938	0.0282	0.0888	0.0335	27.595	3.9316	0.05405	1.5886	9.755
Ext Hiebert funct (N=10)	[100,..100]	0.1248	0.0624	0.078	0.0468	0.02	0.0557	0.1442	0.0229	69.525	NaN	NaN	0.143	8.1134
Ext Hiebert funct (N=10)	[500,..,500]	0.78001	0.0936	1.74721	0.0624	4.1744	0.0351	0.0225	0.0189	3.9747	NaN	NaN	0.0242	13.76
Ext Hiebert funct (N=10)	[1000,...1000]	0.4056	0.0936	3.97803	0.156	3.9192	0.0387	0.2144	0.0282	47.441	NaN	NaN	1.1067	8.4396
Ext Quad P-QP1 (N=2)	[10,..,10]	0.96721	NaN	0.0312	0.0312	9.366	0.0698	0.0823	0.036	7.6568	0.0284	0.03687	0.0784	0.0227
Ext Quad P-QP1 (N=2)	[50,50]	NaN	NaN	0.0624	0.0156	0.3107	0.0545	NaN	0.0554	2.4555	0.0324	0.02524	0.0248	0.0251
Ext Quad P-QP1(N=2)	[100,..100]	2.16841	NaN	0.0312	0.0312	NaN	0.1554	NaN	0.0464	4.3217	0.0329	0.09232	0.0462	0.0235

Table 8. CPU-time (second)

PROB	X0	BFGS	CG-HS	CG-PR	CG-FR	BFGS-CG	IBFGS-CG	CG-BAN	PBFGS-CG	CG-IBRAH	CG-LS	CG-CD	CG-DY	CG-HZ
Ext Quad P-QP1 (N=2)	[500,..,500]	18.4393	0.6396	NaN	5.14803	0.0598	1.329	0.0253	0.023	NaN	0.033	0.02893	0.0238	0.0299
Ext Quad P-QP1 (N=2)	[1000,...1000]	13.4473	NaN	NaN	NaN	7.4032	2.5818	NaN	0.0237	NaN	0.0334	0.07485	0.0905	0.0239
Ext EP1 funct(N=2)	[10,..,10]	18.3145	NaN	NaN	NaN	0.0963	0.0201	0.0069	0.0179	0.2032	NaN	0.07336	0.0651	0.0242
Ext EP1 funct(N=2)	[50,50]	10.1557	0.6864	NaN	2.15281	0.1265	0.0201	0.0136	0.017	0.0511	NaN	0.01356	0.1049	0.0254
Ext EP1 funct(N=2)	[100,..100]	9.81246	0.702	NaN	NaN	0.0186	0.0201	0.0132	0.0183	0.0142	NaN	0.01378	0.0645	0.0226
Ext EP1 funct(N=2)	[500,..,500]	28.1426	0.234	NaN	NaN	0.0184	0.0216	0.0135	0.0158	0.014	NaN	0.01345	0.0645	0.0302
Ext EP1 funct(N=2)	[1000,...1000]	2.38682	0.0468	1.41961	0.078	0.0192	0.0201	0.0134	0.0167	0.014	NaN	0.01525	0.0649	0.0276
Ext EP1 funct(N=4)	[10,..,10]	8.20565	0.2964	NaN	0.5928	0.0199	0.0209	0.016	0.0197	0.0164	NaN	0.01435	0.0735	0.0205
Ext EP1 funct(N=4)	[50,50]	11.3881	0.39	NaN	NaN	0.02	0.0063	0.0139	0.0037	0.0167	NaN	0.01551	0.0755	0.032
Ext EP1 funct(N=4)	[100,..100]	1.77841	0.0936	0.2028	0.0624	0.0202	0.0063	0.0139	0.0168	0.0147	NaN	0.01578	0.0706	0.0282
Ext EP1 funct(N=4)	[500,..,500]	2.27762	0.078	1.07641	0.078	0.0196	0.0208	0.0142	0.0175	0.0145	NaN	0.01354	0.0697	15.784
Ext EP1 funct(N=4)	[1000,...1000]	2.27762	0.078	1.59121	0.0936	0.0191	0.02	0.0134	0.0158	0.0158	NaN	0.01419	0.0732	0.0219
Ext EP1 funct(N=10)	[10,..,10]	6.63004	NaN	0.0312	0.0312	0.0192	0.0215	0.0136	0.0165	0.0147	NaN	0.01618	0.0707	0.0276
Ext EP1 funct(N=10)	[50,50]	14.7421	NaN	0.0156	0.0156	0.0203	0.0204	0.0142	0.0169	0.0164	NaN	0.01404	0.0733	0.034
Ext EP1 funct(N=10)	[100,..100]	NaN	NaN	0.0312	0.0312	0.0197	0.0206	0.0123	0.0173	0.0183	NaN	0.01412	0.0712	0.0281
Ext EP1 funct(N=10)	[500,..,500]	135.066	1.88761	NaN	3.41642	0.0206	0.0206	0.0139	0.0173	0.0167	NaN	0.01375	0.0711	0.0244
Ext EP1 funct(N=10)	[1000,...1000]	199.198	1.90321	1.90321	NaN	0.0192	0.0208	0.0141	0.0163	0.0164	NaN	0.01445	0.0745	0.2418
Ext EP1 funct(N=100)	[10,..,10]	227.699	2.52722	NaN	NaN	0.0661	0.0714	0.0147	0.0199	0.0176	0.1306	0.01515	0.0172	0.021
Ext EP1 funct(N=100)	[50,50]	30.1238	0.0468	0.702	0.0936	0.0552	0.2452	0.0147	0.1034	0.0174	0.0996	0.01521	0.096	0.0334
Ext EP1 funct(N=100)	[100,..100]	73.9445	0.1716	0.3432	1.82521	0.0642	0.1959	0.0149	0.0616	0.0172	NaN	0.01497	0.0933	0.0288
Ext EP1 funct(N=100)	[500,..,500]	NaN	0.6864	NaN	NaN	0.0552	0.0597	0.0149	0.0545	0.0166	NaN	0.01509	0.0939	0.0343
Ext EP1 funct(N=100)	[1000,...1000]	10.4677	0.0468	0.078	0.078	0.0902	0.3335	0.0152	0.0554	0.0169	0.0101	0.01677	0.0949	NaN
Diagonal 9 funct (N=2)	[50,50]	34.3202	0.0624	0.39	0.078	0.0243	0.0203	0.0347	0.0479	NaN	0.0932	144.505	0.0576	0.082
Diagonal 9 funct (N=2)	[100,..100]	16.9573	0.1248	NaN	0.1716	0.0337	0.027	0.0569	0.1553	0.3724	0.1	144.179	0.338	0.1977
Diagonal 9 funct (N=2)	[500,..,500]	0.0624	NaN	1.63801	NaN	0.0441	0.0447	0.0361	0.0682	0.0293	NaN	0.0382	0.0416	0.044
Diagonal 9 funct (N=2)	[1000,...1000]	0.0468	NaN	0.0312	0.0312	0.01	0.0089	0.0109	0.0007	0.0103	0.0105	0.01018	0.0103	0.0009
Diagonal 9 funct (N=4)	[50,50]	0.0624	0.0312	0.0312	NaN	0.0096	0.0244	0.0488	0.0375	NaN	0.0758	NaN	0.117	0.117
Diagonal 9 funct (N=4)	[100,..100]	0.0468	0.0312	0.0624	0.0624	0.0334	0.0333	0.0337	0.0296	0.7748	0.1165	NaN	1.9876	0.1137
Diagonal 9 funct (N=4)	[500,..,500]	0.078	0.0468	0.1716	0.0624	0.0482	0.0562	0.0442	0.045	0.0438	NaN	0.04271	0.0753	0.0478
Diagonal 9 funct (N=4)	[1000,...1000]	NaN	0.0312	0.078	0.0624	0.0007	0.0091	0.0104	0.0093	0.0107	0.0104	0.0103	0.0103	0.0116
Diagonal 9 funct (N=10)	[50,50]	0.0624	0.2652	0.0468	0.2028	0.0265	0.0248	0.0781	0.03	NaN	0.1267	NaN	0.0988	0.1014
Diagonal 9 funct (N=10)	[100,..100]	0.078	NaN	0.2652	NaN	0.0207	0.0331	0.0545	0.0151	0.6596	0.1567	NaN	0.3429	0.0881
ARWHEAD funct(N=2)	[10,..,10]	0.0936	NaN	0.3588	NaN	0.0495	0.0719	0.0321	0.0723	NaN	0.0418	0.02703	0.02	0.0189
ARWHEAD funct(N=2)	[50,50]	NaN	NaN	NaN	0.0312	0.0736	0.0756	0.0541	0.0331	0.0706	0.0401	0.0176	0.0189	0.0233
ARWHEAD funct(N=2)	[100,..100]	0.078	0.0312	0.0468	0.0312	0.0495	0.0983	0.0274	0.0261	0.2386	0.0414	0.02403	0.0212	0.0247
ARWHEAD funct(N=2)	[500,..,500]	0.0624	NaN	0.0624	0.0624	0.0547	0.1706	0.0329	0.0398	0.2822	0.0468	0.02246	0.02	0.024
ARWHEAD funct(N=2)	[1000,...1000]	0.0936	NaN	0.0312	0.078	0.0634	0.1066	0.0274	0.0496	0.1647	0.0386	0.03059	0.0313	0.0252
ARWHEAD funct(N=4)	[10,..,10]	0.0624	NaN	0.0312	NaN	NaN	0.1606	0.0292	0.0197	NaN	NaN	0.21399	0.0382	NaN
ARWHEAD funct(N=4)	[50,50]	0.078	NaN	0.0468	0.078	0.128	0.2876	NaN	0.0277	NaN	NaN	NaN	0.0574	NaN
ARWHEAD funct(N=4)	[100,..100]	NaN	0.0624	0.078	0.156	0.8269	0.5063	NaN	0.0807	33.337	NaN	0.16033	0.1073	NaN

Acknowledgement

A brief acknowledgement section may be given after the conclusion section just before the references. The acknowledgments of people who provided assistance in manuscript preparation, funding for research, etc. should be listed in this section.

Competing Interests

Authors have declared that no competing interests exist.

References

[1] Gilbert JC, Nocedal J. Global convergence properties of conjugate gradient methods for optimization. SIAM Journal on Numerical Analysis. 1992;2:21-42.

[2] Hestenes MR, Stiefel E. Method of conjugate gradient for solving linear equation. Journal of Research of the National Bureau of Standards. 1952;49(6):409-436.

[3] Walsh GR. Methods of optimization. John Wiley, New York; 1975.

[4] Yuan G, Wei Z. New line search methods for unconstrained optimization. Journal of the Korean. Statistical Society. 2009;38(1):2939.

[5] RH Byrd, J Nocedal. A tool for the analysis of quasi-Newton methods with application to unconstrained minimization. SIAM Journal on Numerical Analysis. 1989;26(3):727-739.

[6] Luo YZ, Tang GJ, Zhou LN. Hybrid approach for solving systems of nonlinear equations using chaos optimization and quasi-Newton method. Applied Soft Computing. 2008;8(2):10681073.

[7] Ludwig A. The Gauss-Seidel-quasi-Newton method: A hybrid algorithm for solving dynamic economic model. Journal of Economic Dynamics and Control. 2007;31:1610-1632.

[8] Ibrahim MAH, Mamat M, Leong WJ. The Hybrid BFGS-CG method in solving unconstrained optimization problems. Hindawi Publishing Corporation, Article ID 507102. 2014;6.

[9] Shi ZJ. Convergence of quasi-Newton method with new inexact line search. J. Math. Anal. Appl. 2006;315:120-131.

[10] Fletcher R, Reeves RM. Function minimization by conjugate gradients. The Computer Journal. 1964;7(2):149-154.

[11] Polak E, Ribiere G. Note on the convergence of methods of conjugate direction. Revue Francaised Informatique et de Recherche Operationnelle. 1969;3:35-43.

[12] OM Bamigbola, MM Ali, E Nwaeze. A high-order nonlinear conjugate gradient method for unconstrained optimization. Proceedings of International Congress of Mathematics, Hyderabad, India; 2010.

[13] Hager WW, Zhang H. A survey of nonlinear conjugate gradient methods. Pacific Journal of Optimization. 2006;2(1):35-58.

[14] YH Dai, Y Yuan. An efficient hybrid conjugate gradient method for unconstrained optimization. Annals Operation Research. 2001;103:33-47.

[15] Liu Y, Storey C. Efficient generalized conjugate gradient algorithms, part 1: theory. Journal of Optimization Theory and Applications. 1992;69:129137.

[16] Fletcher R. Practical methods of optimization. 2nd edition. John Wiley, New York; 1967.

[17] OM Bamigbola. Efficiency of variants of the classical conjugate gradient method on large-scale optimization problems. Unpublished manuscript; 2016. .

[18] Okundalaye OO. Optimization Lecture note. Department of Mathematics, Adekunle Ajasin University Akungba-Akoko, Nigeria; 2015.

[19] Edwin K, Chong P, Stanislaw H. Zak. An introduction to optimization. Edition 2 Publisher, John Wiley Sons; 2001.

[20] Gould NIM, Orban D, Toint Ph. L. CUTEr and SifDec: A constrained and unconstrained testing environment, Revisted. ACM Transactions on Mathematical Software. 2003;29:373-394.

[21] More JJ, Garbow BS, Hillstrom KE. Testing unconstrained optimization software. ACM Transaction on Mathematical Software. 1981;7:17-41.

[22] Michalewicz Z. Genetic Algorithms+Data Structures=Evolution programs, Springer, Berlin; 1996.

[23] Andrei N. An unconstrained optimization test functions collection. Advanced Modeling and Optimization. 2008;10(1):147-161.

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
http://www.sciencedomain.org/review-history/24054