On Extensions of Limited Memory Steepest Descent Method

Qinmeng Zou and Frédéric Magoulès
CentraleSupélec, Université Paris-Saclay
91190 Gif-sur-Yvette, France
frederic.magoules@hotmail.com

Abstract—We present some extensions to the limited memory steepest descent method based on spectral properties and cyclic iterations. Our aim is to show that it is possible to combine sweep and delayed strategies for improving the performance of gradient methods. Numerical results are reported which indicate that our new methods are better than the original version. Some remarks on the stability and parallel implementation are shown in the end.

Keywords-limited memory steepest descent; Ritz values; QR factorization; spectral properties.

I. INTRODUCTION

Unconstrained nonlinear optimization is an exciting field and has been the subject of research for centuries. One of the most widely used method is the gradient-based iterative method

$$x_{n+1} = x_n - \alpha_n g_n, \quad (1)$$

where $g_n = \nabla f(x_n)$ is the gradient vector and $\alpha_n > 0$ is viewed as a steplength. Consider a strongly convex quadratic function $f : \mathbb{R}^N \to \mathbb{R}$ in the form

$$f(x) = \frac{1}{2} x^\mathsf{T} A x - b^\mathsf{T} x, \quad (2)$$

where A is an N-dimensional symmetric positive definite (SPD) matrix and b is an N-dimensional vector. The unique minimizer

$$\min_{x \in \mathbb{R}^N} f(x)$$

can be obtained by proper gradient methods. In the past three decades, gradient methods have regained interest with the work of Barzilai and Borwein [1], in which a novel approach, the so-called Barzilai-Borwein method (BB), was proposed in the form

$$\alpha_n^{\mathrm{BB}} = \frac{s_{n-1}^\mathsf{T} s_{n-1}}{s_{n-1}^\mathsf{T} y_{n-1}},$$

where $s_{n-1} = x_n - x_{n-1}$ and $y_{n-1} = g_n - g_{n-1}$. This method is nonmonotone in regard to both the sequences $\{f(x_n)\}$ and $\{\|g_n\|\}$ where $\|g_n\|$ denotes the 2-norm of vector g_n. Many generalizations of BB can thus be considered, see [2] and the references therein.

In 2012, Fletcher [3] proposed a sweep method called limited memory steepest descent (LMSD). During a sweep, spectral information is exploited in order to approximate the eigenvalues of the Hessian matrix A. It follows from (1) and (2) that

$$g_{n+1} = g_n - \alpha_n A g_n. \quad (3)$$

Let $\{\lambda_i\}$ denote the set of eigenvalues of A with $i \in \{1, \ldots, N\}$ and $\{v_i\}$ denote the set of corresponding eigenvectors. We assume that $\lambda_1 \leq \cdots \leq \lambda_N$ and notice that g_n can de decomposed as

$$g_n = \sum_{i=1}^N \zeta_{i,n} v_i,$$

where $\zeta_{i,n}$ is the ith spectral component of g_n. It follows that

$$\zeta_{i,n+1} = (1 - \alpha_n \lambda_i) \zeta_{i,n} = \prod_{j=0}^n (1 - \alpha_j \lambda_i) \zeta_{i,0}.$$

If there exists any $\alpha_j = 1/\lambda_i$, then the ith spectral component vanishes for all subsequent iterations. In particular, if A is a diagonal matrix, then $\zeta_{i,n}$ can be denoted by $g_{i,n}$, expressing the ith component of g_n. Fletcher came up with the idea of using the back gradient vectors to compute the Ritz values, regarded as estimates of the eigenvalues of A. Consider the most recent m back gradient vectors

$$G = [g_{n-m}, g_{n-m+1}, \ldots, g_{n-1}].$$

The QR factorization of G yields an $N \times m$ matrix Q with orthonormal columns and an $m \times m$ upper triangular matrix R. By (3), we know that the columns of G lie in the Krylov subspace

$$\mathrm{span}\left\{ g_{n-m}, A g_{n-m}, \ldots, A^{m-1} g_{n-1} \right\}.$$

Hence, the orthogonalization can be regarded as a Lanczos process (see, e.g., [4]) which leads to a tridiagonal matrix

$$T = Q^\mathsf{T} A Q. \quad (4)$$

The eigenvalues of T, namely, the Ritz values of Hessian matrix A, can be easily solved, for instance by QL algorithm with implicit shift as mentioned in [3].

For general unconstrained optimization problems, A is not available. The following equations provide another way to formulate this method:

$$T = [R, r_n] J R^{-1}, \quad J = \begin{pmatrix} \alpha_{n-m}^{-1} & & & \\ -\alpha_{n-m}^{-1} & \ddots & & \\ & \ddots & \alpha_{n-1}^{-1} & \\ & & -\alpha_{n-1}^{-1} & \end{pmatrix},$$

where R and r can be computed through the partially extended Cholesky factorization

$$G^\mathsf{T}[G, g_n] = R^\mathsf{T}[R, r]. \quad (5)$$

It is noteworthy that in the case of minimizing quadratic function (2) and with $m = 1$, LMSD is mathematically equivalent to BB.

II. EXTENSIONS OF LMSD METHOD

Extensions to LMSD can be thought of by using techniques in the field of delayed gradient methods. The first approach is motivated by the spectral properties. Consider the Yuan steplength [5] in the form

$$\alpha_n^{\mathrm{Y}} = \frac{2}{\alpha_n^{\mathrm{RA}} + \sqrt{(\alpha_n^{\mathrm{RA}})^2 - 4\Gamma_n}},$$

where

$$\alpha_n^{\mathrm{RA}} = \frac{1}{\alpha_{n-1}^{\mathrm{SD}}} + \frac{1}{\alpha_n^{\mathrm{SD}}}, \quad \Gamma_n = \frac{1}{\alpha_{n-1}^{\mathrm{SD}} \alpha_n^{\mathrm{SD}}} - \frac{\|g_n\|^2}{\left(\alpha_{n-1}^{\mathrm{SD}} \|g_{n-1}\|\right)^2}.$$

Under some assumptions without loss of generality, the following limit holds:

$$\lim_{n \to \infty} \alpha_n^{\mathrm{Y}} = \frac{1}{\lambda_N}.$$

We refer the reader to [6] for more details. On the other hand, the most basic gradient method is steepest descent (SD) which can be written in the form

$$\alpha_n^{\mathrm{SD}} = \arg\min_\alpha f(x_n - \alpha g_n).$$

A method that relies on these two stepelengths was proposed in [7] which can be summarized as follows:

1) Execute several gradient iterations based on SD stepelength.
2) Compute α_n^{Y} as a constant using data from the current and last iterations.
3) Execute several gradient iterations based on this constant steplength.

For the quadratic case, this method can foster the sequence $\{1/\alpha_n^{\mathrm{Y}}\}$ to approximate some largest eigenvalues, and thus reduce the search spaces into smaller and smaller dimensions (see [7]).

Our main idea is to consider the SD iterations being used for both fostering alignment and generating m-dimensional subspaces. Here we report the main steps:

1) Execute m gradient iterations based on SD steplength.
2) Compute α_n^{Y} as a constant using data from the current and last iterations.
3) Execute d gradient iterations based on this constant steplength.
4) Compute m Ritz values of the Hessian matrix.
5) Execute m gradient iterations based on these Ritz values.

We observe that the algorithm outlined above is a combination of alignment method and LMSD. This might take advantage of both strategies and improvements might be expected. There exist other constant stepelengths that could be exploited for the purpose of alignment, see [8] for more details.

Another line of extensions starts with the cyclic steplength

$$\alpha_n = \alpha_{n-1}.$$

A theorem provided in [9] could prove its convergence when α_n follows the framework of gradient method with retards. It was recently shown in [10], [11] that cyclic formulation may accelerate the iteration process both in terms of sequential and parallel computation. The main steps based on this strategy can be stated as follows:

1) Compute m Ritz values of the Hessian matrix.
2) Execute $k \cdot m$ gradient iterations based on these Ritz values.

Each Ritz value would be used for k times for updating iterates. Only m back gradient vectors will be stored for computing the next set of Ritz values.

The previous methods are called LMSD, LMSD with constant steplength (LMSDC), LMSD with retards (LMSDR), respectively. For LMSD, Fletcher [3] has further studied two questions: how can the first m back gradient vectors be initialized and in what order should Ritz values be used for performing the gradient iterations within a sweep? He suggested to initialize one or more Ritz values and proceed in a greedy way, namely, in what follows one could use all back gradient vectors for constructing G until it is possible to use m back vectors. For LMSDC, it is clear that SD stepelengths must be employed in the first iterations. For LMSDR, the initialization remains the same as that of LMSD and thus could be carried out by greedy choices as mentioned above.

In [3], Ritz values are selected in decreasing order, which gives the best chance that the monotonicity could be retained in early steps. In addition, if nonmonotonicity is detected in terms of $f(x_n)$ or $\|g_n\|$, then the current sweep is terminated. Obviously, these approaches can be adapted for LMSDC without additional operations. For LMSDR, since m Ritz values are employed in $k \cdot m$ iterations, the most reliable approach is to execute all these iterations in decreasing order with the monotonicity detector mentioned above.

III. NUMERICAL EXPERIMENTS

We implement these algorithms by a linear algebra programming library [12]. QR factorization combining with (4) is used for computing the Ritz values. There are two reasons: the first one is our programming environment has enough working memory that can ensure the storage of several long vectors; the second one is the implementation based on Cholesky factorization suffers from numerical instabilities since R can be numerically ill-conditioned, see [3] for more details. Hence, a positive definite quadratic function is used as test case. We choose the same problem as that in [3], which is based on an SPD matrix of size 20 with $\lambda_1 = 1$. Other eigenvalues are distributed in geometric progression with ratio $\sqrt{2}$. More specifically, the Hessian matrix can be written as follows:
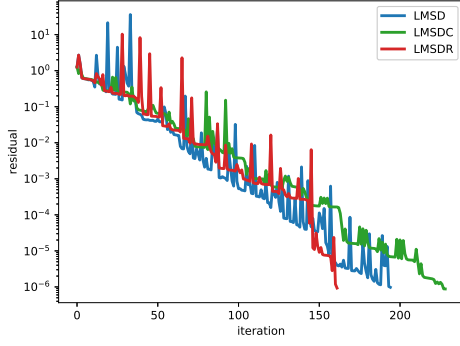
$$A = \mathrm{diag}(1, 1.414, \ldots, 724.077).$$

Figure 1. Convergence history of gradient norm (residual) $\|g_n\|$

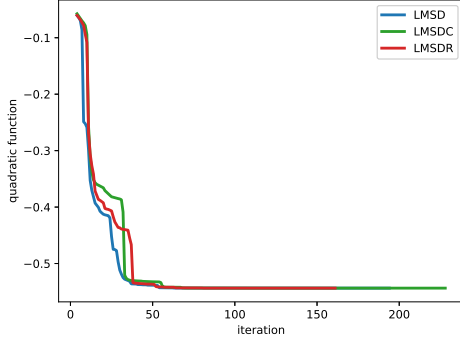Figure 3. Comparison of BB and nonmonotone LMSD with $m = 1$

Figure 2. Convergence history of quadratic function $f(x_n)$

Table I
AVERAGE RUNNING TIME AMONG TEN TESTS AND COUNTS OF
NONMONOTONE TERMINATION WITHIN A SWEEP

Method	LMSD	LMSDC	LMSDR
Time (s)	0.049	0.030	0.031
$f(x_{n+1}) > f(x_n)$	16	1	12
$\|g_{n+1}\| > \|g_n\|$	20	10	15

The right hand side b is selected as a random vector in which the elements range from 10 to 20. The stopping criterion is $\|g_n\| < 10^{-6} \|g_0\|$ and x_0 is fixed with zero.

In Figures 1 and 2 we show the history of residual and quadratic function, respectively. We choose $m = 4$, $d = 4$, and $k = 2$. The figures show that the convergence behaviors of LMSD and its two variants are about the same. Although $\|g_n\|$ is nonmonotone throughout iterations, $f(x_n)$ is monotonically decreased. Table I provides computation times, from which we find that LMSDC and LMSDR are significantly better than the original version. This observation is consistent with our expectations, which shows better performance of cyclic iterations in terms of computation costs compared with classical iterations.

We give the results of nonmonotone termination within a sweep, shown in Table I. We can see that LMSDC almost preserves monotonicity even without local termi-
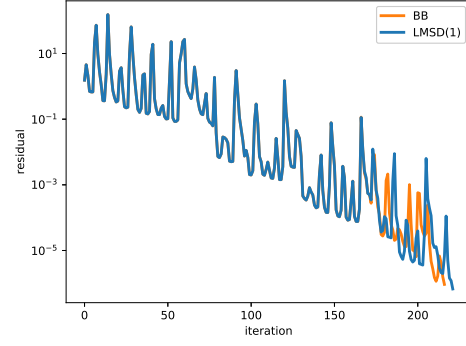
Table II
HIGHLIGHT OF RESIDUALS AND THEIR ABSOLUTE DIFFERENCE FOR
SOME BB AND NONMONOTONE LMSD(1) ITERATIONS

Iteration	100	150	200
BB (p_1)	2.05×10^{-3}	1.49×10^{-2}	2.39×10^{-5}
LMSD(1) (p_2)	2.05×10^{-3}	1.49×10^{-2}	6.45×10^{-6}
$|p_1 - p_2|$	5.57×10^{-11}	2.12×10^{-7}	1.75×10^{-5}

nation criterion. On the other hand, we need to use the termination strategy mentioned above to provide a certain monotonicity for LMSD and LMSDC.

It is known that LMSD is mathematically equivalent to BB for $m = 1$ if nonmonotone sweeps are considered. However, such behavior may not occur in experiments which is subject to rounding errors. An example is shown in Figure 3. Note that we use here the same Hessian matrix and right hand side as the preceding test. In the figure we observe that the convergence results of BB and LMSD are about the same in the beginning. From about iteration 170 we can see some significant difference in colors which indicates the effect of rounding errors. In Table II we highlight some points in the convergence history and notice that when $n = 200$ the absolute difference of residuals even exceeds the second residual value. This observation is consistent with the two curves in Figure 3, which proves that BB and nonmonotone LMSD with $m = 1$ are not numerically equivalent but only mathematically equivalent.

IV. PRACTICAL CONSIDERATIONS

Experience shows that computing Ritz values according to (4) can be a better choice in terms of stability at the expense of more arithmetic operations and extra long vectors. Fletcher [3] gave some remedies which consist of discarding the oldest back gradient vectors and recomputing matrix T. These approaches were also reported in [2] as parts of the implementation. Note that cyclic formulation may amplify the rounding errors. As a result, we could not choose a large k in practice. On the other hand, a smooth reduction in residual can be expected for the alignment process, and in fact the local termination

detector can be removed without changing monotonicity.

The matrix G with $N \gg m$ can be factorized by parallel algorithms. For example, Golub and Van Loan [13] provided some classical strategies for parallel factorizations. One could employ asynchronous iterations (see ,e.g, [14], [15]) in an iterative methods for improving the parallel performance. However, it is difficult to handle the effect of dot product operations. For the communication-avoiding techniques, Demmel et al. [16] discussed parallel QR factorization for tall and skinny matrices which was compared with the classical Householder QR. The storage of long vectors can be avoided by using Cholesky factorization (5), which has also been discussed in the previous references. On the other hand, the communication cost is reduced by cyclic formulation in LMSDR, which may lead to better parallel performance.

We have already implemented the parallel variants by JACK [17], [18], which is an MPI-based communication library with several choices of convergence detectors [19] for both classical and asynchronous iterative computing. Experiments show that LMSDR often gives better convergence results. There exist other possibilities that one could implement parallel LMSD variants by overlapping communication phases with computations. Specifically, the bottleneck in these methods is the dot product operations, which can be tackled by synchronization-reducing techniques. We refer to [20] and the references therein for more detailed discussion on this topic.

V. CONCLUSION

We have presented two variants for the LMSD method, called LMSDC and LMSDR, respectively. The main ingredients that are exploited when formulating the new methods are alignment and cyclic strategies, which have been widely used in delayed gradient methods. The comparison on the problem used in [3] reveals that our new methods are superior in terms of computation time and generate similar convergence curves as the original version.

REFERENCES

[1] J. Barzilai and J. M. Borwein, "Two-point step size gradient methods," *IMA J. Numer. Anal.*, vol. 8, no. 1, pp. 141–148, 1988.

[2] D. di Serafino, V. Ruggiero, G. Toraldo, and L. Zanni, "On the steplength selection in gradient methods for unconstrained optimization," *Appl. Math. Comput.*, vol. 318, pp. 176–195, 2018.

[3] R. Fletcher, "A limited memory steepest descent method," *Math. Program.*, vol. 135, no. 1, pp. 413–436, 2012.

[4] F. Magoulès, F.-X. Roux, and G. Houzeaux, *Parallel Scientific Computing.* Wiley-ISTE, 2015.

[5] Y.-X. Yuan, "A new stepsize for the steepest descent method," *J. Comput. Math.*, vol. 24, no. 2, pp. 149–156, 2006.

[6] Q. Zou and F. Magoulès, "Parameter estimation in the Hermitian and skew-Hermitian splitting method using gradient iterations," *preprint available at arXiv:1909.01481*, 2019.

[7] R. De Asmundis, D. di Serafino, W. W. Hager, G. Toraldo, and H. Zhang, "An efficient gradient method using the Yuan steplength," *Comput. Optim. Appl.*, vol. 59, no. 3, pp. 541–563, 2014.

[8] Q. Zou and F. Magoulès, "Fast gradient methods with alignment for symmetric linear systems without using Cauchy step," *preprint available at arXiv:1909.01479*, 2019.

[9] A. Friedlander, J. M. Martínez, B. Molina, and M. Raydan, "Gradient method with retards and generalizations," *SIAM J. Numer. Anal.*, vol. 36, no. 1, pp. 275–289, 1999.

[10] Q. Zou and F. Magoulès, "A new cyclic gradient method adapted to large-scale linear systems," in *Proceedings of the 17th International Symposium on Distributed Computing and Applications to Business, Engineering and Science.* Wuxi, China: IEEE, 2018, pp. 196–199.

[11] Q. Zou and F. Magoulès, "Reducing the effect of global synchronization in delayed gradient methods for symmetric linear systems," *Adv. Eng. Softw.*, submitted.

[12] F. Magoulès and A.-K. Cheik Ahamed, "Alinea: An advanced linear algebra library for massively parallel computations on graphics processing units," *Int. J. High Perform. Comput. Appl.*, vol. 29, no. 3, pp. 284–310, 2015.

[13] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th ed. Johns Hopkins University Press, 2013.

[14] F. Magoulès and C. Venet, "Asynchronous iterative substructuring methods," *Math. Comput. Simul.*, vol. 145, pp. 34–49, 2018.

[15] F. Magoulès, G. Gbikpi-Benissan, and Q. Zou, "Asynchronous iterations of Parareal algorithm for option pricing models," *Mathematics*, vol. 6, no. 4, pp. 1–18, 2018.

[16] J. W. Demmel, L. Grigori, M. Hoemmen, and J. Langou, "Communication-optimal parallel and sequential QR and LU factorizations," *SIAM J. Sci. Comput.*, vol. 34, no. 1, pp. A206–A239, 2012.

[17] F. Magoulès and G. Gbikpi-Benissan, "JACK: An asynchronous communication kernel library for iterative algorithms," *J. Supercomput.*, vol. 73, no. 8, pp. 3468–3487, 2017.

[18] F. Magoulès and G. Gbikpi-Benissan, "JACK2: An MPI-based communication library with non-blocking synchronization for asynchronous iterations," *Adv. Eng. Softw.*, vol. 119, pp. 116–133, 2018.

[19] F. Magoulès and G. Gbikpi-Benissan, "Distributed convergence detection based on global residual error under asynchronous iterations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 4, pp. 819–829, 2018.

[20] Q. Zou and F. Magoulès, "Recent developments in iterative methods for reducing synchronization," in *Proceedings of the 18th International Symposium on Distributed Computing and Applications to Business, Engineering and Science.* Wuhan, China: IEEE, 2019.