

Gradient method based on epsilon algorithm for large-scale nonlinear optimization

Jianliang Li , * Lian Lian, Jun Xu

School of Science, Nanjing University of Science & Technology, Nanjing 210094, P. R. China

(Received October 22 2007, Accepted December 2 2007)

Abstract. Based on epsilon vector sequences extrapolation, in this paper we apply the sequences acceleration principle to the gradient method on large-scale nonlinear optimization, and present many new accelerating methods for the sequences of gradient method. Finally, the experiment results are showed to compare the new accelerating sequences algorithm with the conjugate gradient method.

Keywords: gradient method, vector sequences extrapolation, epsilon algorithm, numerical experiment

1 Introduction

Nonlinear optimization problem

$$\min_{x \in D} f(x) \quad D \subseteq R^n \quad (1)$$

exist widely in scientific research and engineering. Generally, the algorithm with superlinear convergence, for example, quasi-Newton method based on Newton method or collinear scale method, is effective in solving problem (1)^[4, 13]. For large-scale optimization problem, and complicated optimizations like artificial neural networks or portfolio investment, conjugated gradient method with superlinear convergence is the first choice because there is no need to compute the approximation of Hessian matrix. Conjugate gradient method is a kind of generalization form of deepest descent method. Its essence is to use an appropriate linear combination of two adjacent negative gradient directions as its new search direction. Consequently, this class of algorithms is superlinear convergent and is common used in science research and engineering technology.

In addition, sequence extrapolation principle involves linear or nonlinear combinations of series of approximations, and is applied to many branches of numeric computation because it can obtain required precise predicted results with relatively lower computation complexity. Since we can get superlinear convergent property using the combination of adjacent search directions of deepest descent method, we can also expect to accelerate convergence by applying extrapolation algorithms to vector sequences generated by deepest descent method.

2 Epsilon extrapolation for gradient method

2.1 Gradient method based on epsilon algorithm

The gradient method is the simplest optimization algorithm. Its iterative format is

* Corresponding author. Tel.: +86-25-85408876; E-mail address: ljl6006@hotmail.com.

$$\begin{cases} x_{k+1} = x_k + \alpha_k d_k, \\ d_k = -\nabla f(x_k), \\ \alpha_k : \min_{\alpha > 0} f(x_k + \alpha d_k) \end{cases} \quad k = 0, 1, \dots \tag{2}$$

Though this algorithm has simple form, the generated vector sequence $\{x_k\}$ only has linear convergence because two adjacent search directions are orthogonal. This drawback limits its application.

Many extrapolation algorithms have been studied. Appropriate combinations of them can speed up sequential convergence. We can even get convergent result from divergent iterative sequence using algorithms such as Aitkin extrapolation algorithm^[3, 5, 12]. Epsilon algorithm is the most commonly used sequence extrapolation method, as a more general form of Aitkin extrapolation. That epsilon algorithm can accelerate general linear convergent sequence has been proved in theory [15], while sequences generated by deepest descent method are exactly linear convergent.

Epsilon algorithm is defined by

$$\begin{cases} \varepsilon_{-1}^{(n)} = 0, \varepsilon_0^{(n)} = s_n \\ \varepsilon_{k+1}^{(n)} = \varepsilon_{k-1}^{(n+1)} + [\varepsilon_k^{(n+1)} - \varepsilon_k^{(n)}]^{-1} \end{cases} \quad k, n = 0, 1, \dots \tag{3}$$

Vector epsilon algorithm is a vector-form generalization of epsilon algorithms. The key for generalizing ε -algorithm to the vector case is how to deal with the inverse of the vector sequence. Different methods dealing with the inverse make different generalizing items.

For vector sequence $\{x_k\}$, one can apply the scalar epsilon algorithm to each component of x_n . This method is called the scalar algorithm (SEA).

If we choose the vector inverse as the Samelon inverse

$$z^{-1} = z / \|z\|^2, \quad z \in R^N \tag{4}$$

We can get the vector algorithm (VEA) defined by

$$\begin{cases} \varepsilon_{-1}^{(n)} = 0, \varepsilon_0^{(n)} = x_n \\ \varepsilon_{k+1}^{(n)} = \varepsilon_{k-1}^{(n+1)} + \Delta \varepsilon_k^{(n)} / \|\Delta \varepsilon_k^{(n)}\|_2^2 \end{cases} \quad k, n = 0, 1, \dots \tag{5}$$

Another possibility is to use the inverse defined by

$$\begin{aligned} [\Delta \varepsilon_{2k}^{(n)}]^{-1} &= \frac{y}{(y^T \Delta \varepsilon_{2k}^{(n)})} \\ y^{-1} &= \frac{\Delta \varepsilon_{2k}^{(n)}}{y^T \Delta \varepsilon_{2k}^{(n)}} \\ [\Delta \varepsilon_{2k+1}^{(n)}]^{-1} &= \frac{y^{-1}}{\Delta \varepsilon_{2k+1}^{(n)T} y^{-1}} = \frac{\Delta \varepsilon_{2k}^{(n)}}{\Delta \varepsilon_{2k}^{(n)T} \Delta \varepsilon_{2k+1}^{(n)}} \end{aligned} \tag{6}$$

where y is any non-zero vector in n -dimensional space and $\Delta \varepsilon_k^n = \varepsilon_k^{n+1} - \varepsilon_k^n$. This leads to the topological ε -algorithm (TEA).

$$\begin{aligned} \varepsilon_{-1}^{(n)} &= 0, \varepsilon_0^{(n)} = x_n & n = 0, 1, \dots \\ \varepsilon_{2k+1}^{(n)} &= \varepsilon_{2k-1}^{(n+1)} + \frac{y}{(y, \Delta \varepsilon_{2k}^{(n)})} \\ \varepsilon_{2k+2}^{(n)} &= \varepsilon_{2k}^{(n+1)} + \frac{\Delta \varepsilon_{2k}^{(n)}}{(\Delta \varepsilon_{2k+1}^{(n)}, \Delta \varepsilon_{2k}^{(n)})} \end{aligned} \quad n, k = 0, 1, \dots \tag{7}$$

In this case, for the vector sequence $\{x_n\}$ generated by deepest descent method, we get ε -extrapolation gradient algorithm according to vector algorithm.

Algorithms (ε -extrapolation gradient algorithm)

Step 1. Choose ε, e , stepwise α ;

Step 2. Iterate by formula (2) and make \tilde{x}_0 satisfy $\|g(\tilde{x}_0)\| \leq \varepsilon$, let $k = 0$;

Step 3. Let $x_0 = \tilde{x}_k$, update $2K$ times by formula (2) and get vector sequence $\{x_l\}_{l=0}^{2K}$, if $\|g(\tilde{x}_l)\| \leq \varepsilon$ during the iteration process, go to Step 5;

Step 4. For vector sequence $\{x_k\}_{k=0}^{2K}$, apply vector extrapolation(SEA,VEA or TEA) to get $x_{k+1} = \varepsilon_{2K}^0$. If $\|g(x_{k+1})\| \leq \varepsilon$, go to Step 6;

Step 5. $k = k + 1$, go to Step 2;

Step 6. Stop.

2.2 Conclusions of the accelerated convergence algorithms

Assume that the initial point is x_0 , let x_1, x_2, \dots be the vector sequence generated by x_0

$$x_{j+1} = F(x_j)$$

F is a vector-valued function defined on an open and connected domain D in n -dimensional space, which has a Lipschitz continuous derivative. If $x^* = F(x^*)$ is a fixed point in D and $F'(x^*)$ is the Jacobian matrix of F at x^* , then for all $x \in D$

$$F(x) - x^* = F'(x^*)(x - x^*) + O(\|x - x^*\|^2) \quad (8)$$

We also assume $F'(x^*)$ doesn't have 1 as an eigenvalue.

Lemma 1. ^[2, 14] For the F and x^* , if k is chosen on the t th circle to be the degree of the minimal polynomial of $F'(x^*)$ with respect to $x_i - x^*$ in every algorithm above making use of $2k + 1$ vectors, and if x_0 is sufficiently close to x^* , then

$$\|x_{i+1} - x^*\| = O(\|x_i - x^*\|^2) \quad (9)$$

Theorem 1. For nonlinear optimization, if x^* is a strict minimum point, and $f(x)$ has quadratic derivative $G(x)$ and continuous, second order derivative $G'(x)$, which is Lipschitz continuous. For x^* which is sufficiently close to \tilde{x}_0 , if K_i is chosen on every cycle to be the degree of the minimal polynomial of $G(x)$ with respect to $\tilde{x}_i - x^*$ in every algorithm above(SEA, VEA or TEA) making use of $2K_i + 1$ vectors, then the vector sequence generated by the vector extrapolation algorithm is quadratic convergence

$$\|x_{i+1} - x^*\| = O(\|\tilde{x}_i - x^*\|^2) \quad (10)$$

3 Experiments and numerical analysis

In order to show the efficiency of our accelerating algorithm, we test it with example problems and compare the results with that generated by conjugate gradient method. We use $E(x) = \|\nabla f(x)\|_2$ as the error function and error less than 10^{-6} as the stopping criterion in the test, All the experiments were implemented in C++, and the computer we used was Pentium 4 CPU2.40GHz. As the program ran under Windows XP and there were only 100 variables in the experimental functions, we may not be able to distinguish the small difference before and after we applied our accelerated algorithms to the problems if the CPU time required for running operation system was considered. Because the computation varies little in iteration, we just compared iteration times in the numerical experiments. Moreover, different vector ε -extrapolation gradient algorithms vary only in the computation of the inverse of matrices, so in the following comparisons we just use vector ε -algorithm (VEA) as an example.

Test problem 1 (Extended Rosenbrock Function^[10])

$$f(x) = \sum_{i=1}^n f_i^2(x)$$

where $x \in R^N$ (n is even)and

$$f_{2i-1}(x) = 10(x_{2i} - x_{2i-1}^2)$$

$$f_{2i}(x) = 1 - x_{2i-1}$$

We chose $x_0 = \{-1.2, 1, -1.2, 1, \dots, -1.2, 1\}^T$ as the initial point, the minimum of f is 0. We chose $n = 100$ and the error of one-dimension search less than $1e-4$. We started the extrapolation when the error was less $2e - 5$. We chose $K = 2$. The numerical results are shown in Tab.1.

Table 1. The comparison of VEA and conjugate gradient method for test 1

Algorithm	Iteration	Error
gradient method	11039	9.99984e-7
vector ε -algorithm	123	6.70704e-7
FR conjugate gradient method	10600	9.91765e-7

Test problem 2 (Penalty Function ^[10])

$$f(x) = \sum_{i=1}^{n+1} f_i^2(x)$$

where $x \in R^N$ and

$$f_i(x) = \alpha^{1/2}(x_i - 1), \quad 1 \leq i \leq n$$

$$f_{n+1}(x) = \left(\sum_{j=1}^n x_j^2 \right) - \frac{1}{4}$$

where $\alpha = 10^{-5}$. We chose $x_0 = (1, 2, 3, \dots, n)^T$ as the initial point. We chose $n = 100$ and used error less than $1e - 6$ as the stopping criterion and error of one-dimension search less than $1e - 3$. We stated the extrapolation when the error was less than $1e - 1$. We chose $K = 10$. The numerical results are shown in Tab. 2.

Table 2. The comparison of VEA and conjugate gradient method for test 2

Algorithm	Iteration	Error
gradient method	2759	9.99718e-7
vector ε -algorithm	15	7.46054e-7
FR conjugate gradient method	200	3.82106e-7

The numerical results from Tab. 1 and Tab. 2 show us that gradient method based on vector extrapolation greatly reduced iteration times and saved CPU time for computation. Also the error was reduced. Meanwhile, from the comparison of accelerated sequence algorithm and conjugate gradient method, we found that the former one could even get better numeric results than the latter one with superlinear convergence which was often applied to large-scale optimization.

4 Conclusion

In this paper, we discussed the accelerated convergence of the vector sequences generated by gradient method on large-scale nonlinear optimization. These new algorithms based on epsilon algorithm needn't complicated calculation and can accelerate gradient method significantly and is much faster than conjugate gradient method obviously. The results of numerical experiments shown that can reduce CPU time for computation.

References

- [1] C. Brezinski. Convergence acceleration during the 20th century. *J. Comp. Appl. Math.*, 2000, **122**: 1–21.
- [2] C. Brezinski, G. de la Transformation de Shanks, de la table de Pade. ε -algorithme. *Calcolo*, 1975, **12**: 317–360.
- [3] Cabay, Jackson. A polynomial extrapolation method for finding limits and anti-limits of vector sequence. *SIAM J. Numer. Anal.*, 1976, **13**: 734–752.
- [4] J. E. Dennis, J. J. More. Quasi-newton methods, motivation and theory. *SIAM Rev.*, 1997, **19**: 46–89.
- [5] R. P. Eddy. *Extrapolation to the Limit of a Vector Sequence*, p.c.c wang edn. Information Linkage Between Applied Mathematics and Industry, Academic Press, New York, 1979. 387-396.
- [6] R. Fletcher. Practical methods of optimization ,unconstrained optimization. *John Wiley & Sons*, 1980.
- [7] E. Gekeler. On the solution of system of equation by the epsilon algorithm of wynn. *Math. Comp.*, 1972, **26**: 427–436.
- [8] K. Jbilou, H. Sadok. Some results about vector extrapolation methods and related fixed-point iteration. *J. Comp. Appl. Math.*, 1991, **36**: 385–398.
- [9] K. Jbilou, H. Sadok. Vector extrapolation methods applications and numerical comparison. *J. Comp. Appl. Math.*, 2000, **122**: 149–165.
- [10] J. J. More, et al. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software*, 1981, **3**(7): 17–41.
- [11] A. Sidi. Convergence and stability properties of minimal polynomial and reduced rank extrapolation algorithms. *SIAM J. Numer. Anal.*, 1986, **23**: 197–209.
- [12] A. Sidi, et al. Acceleration of convergence of vector sequences. *SIAM J. Numer. Anal.*, 1986, **23**: 176–196.
- [13] D. C. Sorensen. The q-superlinear convergence of a collinear scaling algorithm for unconstrained optimization. *SIAM, Numer. Anal.*, 1980, **12**: 84–114.
- [14] P. Wynn. On a device for computing the transformation. *MATC*, 1956, **10**: 91–96.
- [15] P. Wynn. Acceleration techniques for iterated vector and matrix problems. *Math. Comp.*, 1966, **12**: 331–348.
- [16] Y. Yuan. A modified bfgs algorithm for unconstrained optimization. *IMA Journal of Numerical Analysis*, 1991, **11**: 325–332.