The Wolfe Epsilon Steepest Descent Algorithm

Hakima Degaichia
Department of Mathematics, University of Tebessa
12000 Tebessa, Algeria

Rachid Benzine
LANOS, Badji Mokhtar University, Annaba, Algeria

Copyright © 2014 Hakima Degaichia and Rachid Benzine. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract

In this article, we study the unconstrained minimization problem

\[(P) \min \{ f(x) : x \in \mathbb{R}^n \} \]

where \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) is a continuously differentiable function. We introduce a new algorithm which accelerates the convergence of the steepest descent method. We study the global convergence of the new algorithm, named the Wolfe epsilon steepest descent algorithm, by using Wolfe inexact line search ([35],[36]). In [16], [33], Benzine, Djeghaba and Rahali studied the same problem by using exact and Armijo inexact line searches. Numerical tests show that Wolfe Epsilon steepest descent Algorithm accelerates the convergence of the gradient method and is more successful than Armijo Epsilon Steepest descent, Exact Epsilon steepest descent algorithm and Steepest descent algorithm.

Mathematics Subject Classification: 90C26; 65H10

Keywords: Unconstrained optimization. Gradient Method. Epsilon steepest descent Algorithm. Wolfe line search. Armijo line search. Exact line search. Global convergence
1. INTRODUCTION

Our problem is to minimize a function of \( n \) variables

\[
(1.1) \quad \{ \min f(x), \quad x \in \mathbb{R}^n \},
\]

where \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) is smooth and its gradient \( g \) is available. We consider iterations of the form

\[
(1.2) \quad x_{k+1} = x_k + \alpha_k d_k
\]

where \( d_k \) is a search direction and \( \alpha_k \) is a steplength obtained by means of a one-dimensional search. In conjugate gradient methods the search direction is of the form

\[
d_k = -g_k + \beta_k d_{k-1}
\]

where the scalar \( \beta_k \) is chosen so that the method reduces to the linear conjugate gradient method when the function is quadratic and the line search is exact. If \( \beta_k = 0 \), then one obtains the steepest descent method. Another broad class of methods defines the search direction by

\[
(1.3) \quad d_k = -B_k^{-1} g_k
\]

where \( B_k \) is a nonsingular symmetric matrix. Important special cases are given by:

- \( B_k = I \) (the steepest descent method)
- \( B_k = \nabla^2 f(x_k) \) (Newton's method)

Variable metric methods are also of the form (1.3), but in this case \( B_k \) is not only a function of \( x_k \) but depends also on \( B_{k-1} \) and \( x_{k-1} \).

All these methods are implemented so that \( d_k \) is a descent direction i.e. so that

\[
d_k^T g_k < 0,
\]

which guarantees that the function can be decreased by taking a small step along \( d_k \).

The convergence properties of line search methods can be studied by measuring the goodness of the search direction and by considering the length of the step. The quality of the search direction can be studied by monitoring the angle between the steepest descent direction \(-g_k\) and the search direction. Therefore we define

\[
\cos(\theta_k) = \frac{-d_k^T g_k}{\|g_k\| \|d_k\|}
\]

The length of the step is determined by a line search iteration. In computing the step length \( \alpha_k \), we face tradeoff. We would like to choose \( \alpha_k \) to give a substantial reduction to \( f \), but at the same time, we do not want to spend too much time making the choice. The ideal choice would be the global minimizer of the univariate function \( \varphi(\alpha) \) defined by

\[
\varphi(\alpha) = f(x_k + \alpha d_k)
\]
The Wolfe epsilon steepest descent algorithm

Exact line search consists to find $\alpha_k$ as a solution of the following one-dimension optimal problem:
\[
  f(x_k + \alpha_k d_k) = \min \{ f(x_k + \alpha d_k) : \alpha > 0 \}
\]

However, commonly, the exact line search is expensive. Especially, when an iterate is far from the solution of the problem, it is not effective to solve exactly one-dimension subproblem. A strategy that will play a central role in this paper consists in accepting a steplength $\alpha_k$ if it satisfies the two conditions:

\[
  f(x_k + \alpha_k d_k) \leq f(x_k) + \sigma_1 \alpha_k g_k^T d_k
\]

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

where $0 < \sigma_1 < \sigma_2 < 1$. The first inequality (1.4) (Armijo condition ([5])), ensures that the function is reduced sufficiently, and the second prevents the from being too small. We will call these two relations the Wolfe conditions.

One can also choose $\alpha_k$ satisfying the following conditions:

\[
  f(x_k + \alpha_k d_k) \leq f(x_k) + \sigma \alpha_k g_k^T d_k
\]

\[
  f(x_k + \alpha_k d_k) \geq f(x_k) + (1 - \sigma) \alpha_k g_k^T d_k
\]

where $0 < \sigma < \frac{1}{2}$. (1.6) and (1.7) are called Goldstein conditions.

This line search strategy allows us to establish the following result due to Zoutendijk. This result was essentially proved by Zoutendijk ([39]) and Wolfe ([35], [36]). The starting point of the algorithm is denoted by $x_1$.

**Theorem 1.** ([39], [35], [36]). Suppose that $f$ is bounded below in $\mathbb{R}^n$ and that $f$ is continuously differentiable in a neighborhood $\mathcal{N}$ of the level set $\mathcal{L} = \{ x : f(x) \leq f(x_1) \}$. Assume also that the gradient is Lipschitz continuous, i.e., there exists a constant $L > 0$ such that

\[
  \| g(x) - g(y) \| \leq L \| x - y \|
\]

for all $x, y \in \mathcal{N}$. Consider any iteration of the form (1.2), where $d_k$ is a descent direction and $\alpha_k$ satisfies the Wolfe conditions (1.4), (1.5). Then

\[
  \sum_{k=1}^{\infty} \cos^2(\theta_k) \| g_k \|^2 < \infty
\]

We shall call inequality (1.9) the Zoutendijk condition. Suppose that an iteration of the form (1.2) is such that

\[
  \cos(\theta_k) \geq \delta > 0
\]

for all $k$. Then we conclude that from (1.9) that

\[
  \lim_{k \to \infty} \| g_k \| = 0.
\]

The steepest descent method is one of the simplest and the most fundamental minimization methods for unconstrained optimization. Since it uses the negative gradient as its descent direction, it is also called the gradient method.
For many problems, the steepest descent method is very slow. Although
the method usually works well in the early steps, as a stationary point
is approached, it descends very slowly with zigzaguing phenomena. There
are some ways to overcome these difficulties of zigzagging by deflating
the gradient. Rather than moving along 

\[ d_k = -\nabla f(x_k) \] 

([8],[9],[10], [12], [13], [14], [15], [17], [21], [22], [23], [24], [25], [28], [31], [32], [33]),
oralong

\[ d_k = -g_k + h_k \] 

([18], [19],[20],[21],[27], [29],[30]), where \( D_k \) is an appropriate
matrix and \( h_k \) is an appropriate vector.

In [16] and in [33], Benzine, Djeghaba and Rahali provided another solution
to this problem by accelerating the convergence of the gradient method.

They achieved this goal by designing a new algorithm, named the epsilon
steepest descent algorithm, in which the Florent Cordellier Wynn epsilon al-
algorithm ([11], [37], [38]) with exact and Armijo inexact line searches play a
prominent role.

In this work we accelerate the convergence of the gradient method by using
the Florent Cordellier epsilon algorithm ([11]) and the Wolfe inexact line search
([35], [36]).

We study the global convergence of the new algorithm, named the Wolfe
epsilon steepest descent algorithm, by using Wolfe inexact line searches (1.4),
(1.5) ([35], [36]).

Numerical tests show that our algorithm accelerates the convergence of the
gradient method and is at least as efficient as the other Epsilon steepest descent
algorithms ([16], [33]).

This paper is organized as follows. In the next section, the New algorithm is
stated and descent property is presented. The global convergence of the new
algorithm is established. Numerical results are presented in Section 5.

2. THE EPSILON ALGORITHM

The Epsilon Algorithm is due to P. Wynn ([37], [38]).

Given a sequence \( \{ x_k \}_{k \in \mathbb{N}} \), \( x_k \in \mathbb{R}^n \). The coordinates of \( x_k \) will be noted as follows:

\[ x_k = (x^1_k, x^2_k, ..., x^n_k) \in \mathbb{R}^n \]

For \( i \in \{1, 2, ..., n\} \), the Epsilon Algorithm calculates quantities with two
indices \( \varepsilon_{j,k}^i \) as follows:

\[
\begin{align*}
\varepsilon_{-1}^i &= 0 \\
\varepsilon_0^i &= x^i_k \quad k = 0, 1, ...
\end{align*}
\]

\[
\varepsilon_{j+1}^i = \varepsilon_{j}^i + \frac{1}{\varepsilon_{j+1}^k - \varepsilon_{j}^k} \varepsilon_{j}^k 
\quad j, k = 0, 1, ...
\]

For \( i \in \{1, 2, ..., n\} \), these numbers can be placed in an array as follows:
This array is called the ε-array. In this array the lower index denotes a column while the upper index denotes a diagonal.

For $i \in \{1, 2, ..., n\}$, the Epsilon algorithm relates the numbers located at the four vertices of a rhombus:

$$
\begin{array}{c}
\varepsilon_{k+1,i}^j = x_{k+1}^i + \frac{1}{x_{k+2}^i - x_{k+1}^i} - \frac{1}{x_{k+1}^i - x_k^i}^{-1}, & (i = 1, 2, ..., n) \\
\end{array}
$$

To calculate the quantities $\varepsilon_{k+1,i}^j$, we need to know the numbers $\varepsilon_{j-1}^{k+1,i}$, $\varepsilon_{j}^{k+1,i}$ and $\varepsilon_{j}^{k,i}$.

3. THE WOLFE EPSILON STEEPEST DESCENT ALGORITHM

To construct our algorithm, we use the column $\varepsilon_{k+1,i}^j$ ($i = 1, 2, ..., n$). Given a sequence $\{x_k^i\}_{k \in \mathbb{N}}$ ($i = 1, 2, ..., n$), Aitken and F. Cordellier ([1], [6], [11]) proposed another formula to calculate the epsilon algorithm of order 2. The quantities $\varepsilon_{k+1,i}^j$ can be calculated as follows

$$
(3.1) \quad \varepsilon_{k+1,i}^j = x_{k+1}^i + \frac{1}{x_{k+2}^i - x_{k+1}^i} - \frac{1}{x_{k+1}^i - x_k^i}^{-1}, & (i = 1, 2, ..., n) \\
$$

To calculate $\varepsilon_{k+1,i}^j$, we use the elements $x_k^i$, $x_{k+1}^i$ and $x_{k+2}^i$ ($i = 1, 2, ..., n$). Numerical calculations ([11]) showed that the epsilon algorithm of order 2 with the Cordellier formula (3.1) is more stable than Wynn epsilon algorithm (2.1).

We are now in measure to introduce our new algorithm: the Armijo epsilon steepest descent algorithm.

The Wolfe epsilon steepest descent algorithm
**Initialization step:** Choose an initial point \( x_0 \in \mathbb{R}^n \) an initial point. The coordinates of \( x_0 \) will be noted as follows:

\[
x_0 = (x_0^1, x_0^2, \ldots, x_0^i, \ldots, x_0^n) \in \mathbb{R}^n
\]

Let \( k = 0 \) and go to Main step.

**Main Step:** Starting with the vector \( x_k \),

\[
x_k = (x_k^1, x_k^2, \ldots, x_k^i, \ldots, x_k^n)
\]

If \( \| \nabla f(x_k) \| = 0 \), stop. Otherwise, let should be \( r_k = x_k \) and compute the vectors \( s_k \) and \( t_k \) by using twice the steepest descent algorithm, with Wolfe inexact line search

\[
s_k = r_k - \lambda_k \nabla f(r_k),
\]

and

\[
t_k = s_k - \beta_k \nabla f(s_k),
\]

\( \lambda_k \) and \( \beta_k \) are positive scalars obtained by the Wolfe inexact line search (1.4) and (1.5).

If

\[
s_k^i - r_k^i \neq 0, \ t_k^i - s_k^i \neq 0 \text{ and } \frac{1}{t_k^i - s_k^i} - \frac{1}{s_k^i - r_k^i} \neq 0, \ i = 1, \ldots, n
\]

Let

\[
\varepsilon_{k,i}^i = s_k^i + \left( \frac{1}{t_k^i - s_k^i} - \frac{1}{s_k^i - r_k^i} \right)^{-1}, \ i = 1, \ldots, n,
\]

and

\[
\varepsilon_k^i = (\varepsilon_{k,1}^i, \ldots, \varepsilon_{k,i}^i, \ldots, \varepsilon_{k,n}^i).
\]

If \( f(\varepsilon_k^i) < f(t_k) \), let \( x_k = \varepsilon_k^i \). Replace \( k \) by \( k + 1 \) and go to main step.

If \( f(\varepsilon_k^i) \geq f(t_k) \) or if

\[
s_k^{i_0} - r_k^{i_0} = 0 \text{ or } t_k^{i_0} - s_k^{i_0} = 0 \text{ or } \frac{1}{t_k^{i_0} - s_k^{i_0}} - \frac{1}{s_k^{i_0} - r_k^{i_0}} = 0, \ i_0 \in \{1, \ldots, n\}.
\]

Let \( x_k = t_k \). Replace \( k \) by \( k + 1 \) and go to Main step.

### 4. GLOBAL CONVERGENCE OF THE WOLFE EPSILON STEEPEST DESCENT ALGORITHM

The foregoing preparatory results enable us to establish the following theorem

**Theorem 2.** For the unconstrained minimization problem (1.1), we let \( x_1 \) be a starting point of the Wolfe epsilon steepest descent Algorithm, and assume that the following assumptions hold: the \( f \) is bounded below in \( \mathbb{R}^n \) and that \( f \) is continuously differentiable in a neighborhood \( \mathcal{N} \) of the level set.
\[ \mathcal{L} = \{ x : f(x) \leq f(x_1) \} . \] Assume also that the gradient is Lipschitz continuous, i.e., there exists a constant \( L > 0 \) such that
\[ \| g(x) - g(y) \| \leq L \| x - y \| \]
for all \( x, y \in \mathcal{N} \). Then, the sequence \( \{ x_k \}_{k \in \mathbb{N}} \) generated by the Wolfe epsilon steepest descent algorithm must satisfy one of the properties: \( \nabla f(x_{k_0}) = 0 \) for some \( k_0 \in \mathbb{N} \) or \( \| \nabla f(x_k) \| \xrightarrow{k \to \infty} 0 \).

**Proof.** Suppose that an infinite sequence \( \{ x_k \}_{k \in \mathbb{N}} \) is generated by the Wolfe epsilon steepest descent Algorithm. According to the Algorithm, the vectors \( s_k \) and \( t_k \) are obtained by using twice the steepest descent method, with Wolfe inexact line search. Then we have
\[ f(s_k) < f(r_k) = f(x_k) \]
and
\[ f(t_k) < f(s_k) \]
Now, by considering the Algorithm, if the calculation of \( \varepsilon^k_2 \) is possible, two cases are possible:

a) \( f(\varepsilon^k_2) < f(t_k) \). Then we have
\[ f(x_{k+1}) = f(\varepsilon^k_2) < f(x_k) \]

b) \( f(\varepsilon^k_2) \geq f(t_k) \) or if the calculation of \( \varepsilon^k_2 \) is not possible. In this case and according to the algorithm we have
\[ f(x_{k+1}) = f(t_k) < f(x_k) . \]

In conclusion the Wolfe epsilon steepest descent Algorithm guarantees
\[ f(x_{k+1}) < f(x_k) , \quad k = 0, 1, 2, \ldots \]

(4.1)

(4.2) \[ \varepsilon^k_2 = x_{k+1} = x_k + \alpha_k d_k , \]
where \( d_k \) and \( \alpha_k \) depend on \( s_k , r_k , t_k , \lambda_k \) and \( \beta_k \).

(4.1) implies that \( d_k \) is a descent direction. Then by theorem 1, we have
\[ \sum_{k=1}^{\infty} \cos^2(\theta_k) \| g_k \|^2 < \infty \]
(4.3)

The Wolfe Epsilon steepest descent Algorithm is similar to the steepest descent algorithm, then it is not difficult to prove that
\[ \cos(\theta_k) \geq \delta > 0 \]
(4.4)

for all \( k \). Then we conclude that from (4.3) that
\[ \lim_{k \to \infty} \| g_k \| = 0 . \]
(4.5)

\[ \square \]
5. NUMERICAL RESULTS AND COMPARISONS

In this section we report some numerical results obtained with an implementation of the Wolfe Epsilon Steepest Descent algorithm. For our numerical tests, we used test functions and Fortran programs from ([2]). Considering the same criteria as in ([3]), the code is written in Fortran and compiled with f90 on a Workstation Intel Pentium 4 with 2 GHz. We selected a number of 52 unconstrained optimization test functions in generalized or extended form [2] (some from CUTE library [4]). For each test function we have taken twenty (20) numerical experiments with the number of variables increasing as $n = 2, 10, 30, 50, 70, 100, 300, 500, 700, 900, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000$: The algorithm implements the Wolfe line search conditions (1.4), (1.5) ([35], [36]), and the same stopping criterion $\|\nabla f (x_k)\| < 10^{-6}$. In all the algorithms we considered in this numerical study the maximum number of iterations is limited to 100000.

The comparisons of algorithms are given in the following context. Let $f_{i}^{ALG1}$ and $f_{i}^{ALG2}$ be the optimal value found by ALG1 and ALG2, for problem $i = 1, \ldots, 962$, respectively. We say that, in the particular problem $i$, the performance of ALG1 was better than the performance of ALG2 if:

$$|f_{i}^{ALG1} - f_{i}^{ALG2}| < 10^{-3}$$

and the number of iterations, or the number of function-gradient evaluations, or the CPU time of ALG1 was less than the number of iterations, or the number of function-gradient evaluations, or the CPU time corresponding to ALG2, respectively.

In the set of numerical experiments we compare Wolfe Epsilon Steepest Descent algorithm versus Armijo Epsilon Steepest descent, Exact Epsilon steepest descent algorithm and Steepest descent algorithm. Figure 1 shows the Dolan and Moré CPU performance profile of Wolfe Epsilon Steepest Descent algorithm versus Armijo Epsilon Steepest descent, Exact Epsilon steepest descent algorithm and Steepest descent algorithm.

In a performance profile plot, the top curve corresponds to the method that solved the most problems in a time that was within a factor $\tau$ of the best time. The percentage of the test problems for which a method is the fastest is given on the left axis of the plot. The right side of the plot gives the percentage of the test problems that were successfully solved by these algorithms, respectively. Mainly, the right side is a measure of the robustness of an algorithm. When comparing Wolfe Epsilon Steepest Descent algorithm to Armijo Epsilon Steepest descent, Exact Epsilon steepest descent algorithm and Steepest descent algorithm, subject to CPU time metric, we see that Armijo Wolfe Steepest Descent algorithm is top performer. The Wolfe Epsilon Steepest Descent algorithm is more successful than Armijo Epsilon Steepest descent,
The Wolfe epsilon steepest descent algorithm

Exact Epsilon steepest descent algorithm and Steepest descent algorithm.

References


[37] Wynn, P. "On a device for computing the em(Sn) transformation", *M.T.A.C.*, 10 (1956), 91-96.


Received: March 6, 2014