QUASI-NEWTON TYPE OF DIAGONAL UPDATING FOR THE L-BFGS METHOD

M. L. SAHARI AND R. KHALDI

Abstract. The use of the L-BFGS method is very efficient for the resolution of large scale optimization problems. The techniques to update the diagonal matrix seem to play an important role in the performance of overall method. In this work, we introduce some methods for updating the diagonal matrix derived from quasi-Newton formulas (DFP, BFGS). We compare their performances with the Oren-Spedicato update proposed by Liu and Nocedal (1989) and we get considerable amelioration in the total running time. We also study the convergence of L-BFGS method if we use the BFGS and inverse BFGS update of the diagonal matrix on uniformly convex problems.

1. Introduction

The aim of this paper is to introduce some methods for updating the diagonal matrix obtained from quasi-Newton formulas and then after applying Oren-Spedicato update proposed in [4] by Liu and Nocedal to study their performances. We show that the total running time is improved considerably. Using the BFGS and inverse BFGS update of diagonal matrix on uniformly convex problems, we study the convergence of L-BFGS method.

More precisely, we consider the unconstrained optimization problem

\[ \min_{x \in \mathbb{R}^n} f(x), \]

where \( f \) is a real valued function on \( \mathbb{R}^n \). We assume throughout that both the gradient \( g(x) = \nabla_x f(x) \) and the Hessian matrix \( H(x) = \nabla_{xx} f(x) \) of \( f \) exist and are continuous. To solve the problem (1.1) one uses an algorithm that generates a sequence of iterates \( x_k \) according to

\[ x_{k+1} = x_k + t_k d_k, \]

where \( x_1 \in \mathbb{R}^n \) is given, \( d_k \in \mathbb{R}^n \) is the search direction and \( t_k \) is a step length which minimizes \( f \) along \( d_k \) from the point \( x_k \). In this paper we suppose that \( t_k \)
satisfies the Wolfe conditions (see [1], [6])

(1.3) \[ f(x_k + t_k d_k) - f(x_k) \leq c_1 t_k g(x_k) d_k, \]

(1.4) \[ g(x_k + t_k d_k) d_k \geq c_2 g(x_k) d_k, \]

where \( 0 < c_1 < 1/2, c_1 < c_2 < 1 \). If \( n \) is not large \((n \leq 100)\), the BFGS method is very efficient (see [1], [6]). In this case the direction \( d_k \) is defined by

(1.5) \[ d_k = -S_k \cdot g(x_k) \]

where \( S_k \) is an inverse Hessian approximation updated at every iteration by means of the formula

(1.6) \[ S_{k+1} = V_k^T S_k V_k + \rho_k \delta_k \delta_k^T \]

where

(1.7) \[ \rho_k = \frac{1}{\gamma_k \delta_k}, \]

\( V_k = I - \rho_k \gamma_k \delta_k^T \)

and

(1.8) \[ \delta_k = x_{k+1} - x_k, \quad \gamma_k = g(x_{k+1}) - g(x_k). \]

But if \( n \) is very large when the matrix \( S_k \) cannot be computed or stored, it is desirable to use limited memory BFGS (L-BFGS) (see [1], [2], [3]). The implementation described by Liu and Nocedal [4] is almost identical to that of the BFGS method, the only difference is in the matrix update. Instead of storing the matrix \( S_k \), one stores a certain number, say \( m \) of pairs \( \{\delta_k, \gamma_k\} \). The product \( S_k \cdot g(x_k) \) is obtained by performing a sequence of inner products involving \( g(x_k) \) and the \( m \) most recent vector pairs \( \{\delta_k, \gamma_k\} \). After computing the new iterate, the oldest pair is deleted from the set \( \{\delta_k, \gamma_k\} \) and replaced by the newest one. The algorithm therefore always keeps the \( m \) most recent pairs \( \{\delta_k, \gamma_k\} \) to define the iteration matrix. This strategy is suitable for large scale problems because it has been observed in practice that small values of \( m \) (say \( m \in (3, 8) \)) give satisfactory results [4].

The structure of this paper is the following: In the next section we describe the L-BFGS update process. Section three deals with updating the diagonal matrix, for this we propose several updating methods for the diagonal matrix \( D_k \) derived from the quasi-Newton methods [3], [6].

In section four, we study the convergence, we show that the L-BFGS method with BFGS and inverse BFGS update formulas of the diagonal matrix are globally convergent on uniformly convex problems if the line search is Wolfe type. Section five concerns numerical tests on some problems proposed in [5], using FORTRAN 90. Then we give tables that compare the performance of the L-BFGS method with Oren-Spedicato scaling and other updates of the diagonal matrix proposed in this work (DFP, BFGS and Inverse BFGS). We achieve this work with a conclusion and some references.
2. L-BFGS Method

Now, we give a precise description of the L-BFGS update process in more details. Let $x_1$ be given as the first element of the iteration and suppose that we have stored the $m$ pairs $\{\delta_k, \gamma_k\}$, $k = 1, \ldots, m$. We choose a “basic matrix” $S_k = D_k$ (usually a diagonal with all positive entries) and from (1.6) the $S_k$ update can be written as

$$
S_{k+1} = V_k^T S_k V_k + \rho_k \delta_k \delta_k^T, \quad \text{for } 1 \leq k \leq m
$$

$$
S_1 = D_1
$$

$$
S_i^{m+1} = V_{k-m+i-1}^T S_k V_{k-m+i-1} + \rho_k \delta_{k-m+i-1} \delta_{k-m+i-1}^T, \quad 1 \leq i \leq m, \quad k \geq m+1.
$$

In practice we prefer to use the following more explicit formula

$$
S_k = \sum_{i=\tilde{m}+1}^i \rho_{k-i} \left( \prod_{j=0}^{j=i-1} V_{k-j}^T \right) \delta_{k-i} \delta_{k-i}^T \left( \prod_{j=1}^{j=i} V_{k+j-1} \right),
$$

with $\tilde{m} = \min\{k-1, m\}$ and

$$
\begin{align*}
\rho_{k-\tilde{m}-1} &= 1, \\
\delta_{k-\tilde{m}+1} \delta_{k-\tilde{m}+1}^T &= D_k, \\
V_k &= I.
\end{align*}
$$

3. Updating the diagonal matrix

The diagonal matrix $D_k$ has to be ensured by the weak quasi-Newton condition

$$(3.1) \quad \gamma_{k-1}^T S_k \gamma_{k-1} = \gamma_{k-1}^T \delta_{k-1}.$$

Liu and Nocedal [4] recommend the choice of the multiple of the unit matrix

$$(3.2) \quad D_k = \zeta_{k-1} I,$$

where $I$ is the identity matrix and $\zeta_{k-1} = \gamma_{k-1}^T \delta_{k-1} / \gamma_{k-1} \delta_{k-1}$. This scaling is suggested by Oren and Spedicato [7], using only the last pair $(\delta_i, \gamma_i)$. This strategy does not allow a well initialization of the L-BFGS method. In the following we propose several methods of updating the diagonal matrix $D_k$, derived from the quasi-Newton methods, i.e. DFP and BFGS methods.

3.1. The DFP type of diagonal update

The DFP diagonal update formula is obtained by taking the diagonal of the matrix $D_k$ with the DFP formula. If

$$
D_k = \text{diag} \left[ D_k^{(1)}, D_k^{(2)}, \ldots, D_k^{(i)}, \ldots, D_k^{(m)} \right]
$$
and \( \{e_1, e_2, \ldots, e_n\} \) is the canonical basis of \( \mathbb{R}^n \), the \( i \)-th update component is

\[
D^{(i)}_{k+1} = D^{(i)}_k + \frac{\left( \delta^T_k e_i \right)^2}{\gamma_k \delta_k} - \frac{\left( D^{(i)}_k (\gamma_k e_i) \right)^2}{\gamma_k D_k \gamma_k}.
\]

### 3.2. The BFGS type of diagonal update

By taking the diagonal of the matrix obtained by updating \( D_k \) with (1.6) formula, the \( i \)-th update component is given by

\[
D^{(i)}_{k+1} = D^{(i)}_k + \left[ 1 + \frac{\gamma_k^T D_k \gamma_k}{\gamma_k \delta_k} \right] \frac{\left( \delta^T_k e_i \right)^2}{\gamma_k \delta_k} - 2 \frac{D^{(i)}_k (\delta^T_k e_i) (\gamma_k e_i)}{\gamma_k \delta_k}.
\]

### 3.3. The inverse BFGS type of diagonal update

If we use the BFGS formula for the Hessian approximation update given by

\[
B_{k+1} = B_k + \frac{\gamma_k \gamma_k^T}{\gamma_k \delta_k} - \frac{B_k \delta_k \delta_k^T B_k}{\delta_k^T B_k \delta_k}
\]

and due to the fact that

\[
D^{-1}_{k+1} = \text{diag} \left[ (D^{(1)}_{k+1})^{-1}, (D^{(2)}_{k+1})^{-1}, \ldots, (D^{(i)}_{k+1})^{-1}, \ldots, (D^{(n)}_{k+1})^{-1} \right],
\]

by (3.5), we have

\[
D^{-1}_{k+1} = D^{-1}_k + \frac{\gamma_k \gamma_k^T}{\gamma_k \delta_k} - \frac{D^{-1}_k \delta_k \delta_k^T D^{-1}_k}{\delta_k^T D^{-1}_k \delta_k},
\]

finally

\[
D^{(i)}_{k+1} = \left( \frac{1}{D^{(i)}_k} + \frac{(\gamma_k e_i)^2}{\gamma_k \delta_k} - \frac{(\delta^T_k e_i)^2}{(D^{(i)}_k)^2 (\delta_k^T D^{-1}_k \delta_k)} \right)^{-1}.
\]

### Algorithm 1 (The L-BFGS method with diagonal update)

Choose \( \varepsilon > 0, x_1 \in \mathbb{R}^n \), set \( S_1 = D_1 = I \) and \( k = 1 \)

**Repeat**

Starting

If \( \|g(x_k)\| < \varepsilon \) then

\( x^* = x_k \)

stop.

Else

Compute direction search: \( d_k = -S_k \cdot g(x_k) \)

Compute the step length \( t_k \) satisfies the Wolfe conditions (1.3)–(1.4)

Update \( D_k \) matrix according to (3.2), (3.3), (3.4) or (3.6)

Update \( S_k \) matrix according to (2.1)

Set \( x_{k+1} = x_k + t_k d_k \)

\( k \leftarrow k + 1 \)

End if

**End repeat**
4. Convergence Analysis

Now, we show that the L-BFGS method with (3.2), (3.4) and (3.6) updates of the diagonal matrix is globally convergent on uniformly convex problems if the line search is Wolfe type. In the case of (3.3) update formula, we can deduce this analysis if the step length $t_k$ is determined by the exact line search

$$f(x_k + t_k d_k) = \min_{t \geq 0} f(x_k + td_k).$$

Assumptions A

(1) The function $f$ is twice continuously differentiable.
(2) The level set $\mathcal{L} = \{x \in \mathbb{R}^n : f(x) \leq f(x_1)\}$ is convex.
(3) There exist constants $M_1, M_2 > 0$ such that

$$M_1 \|d\|^2 \leq d^\top H(x)d \leq M_2 \|d\|^2$$

for all $d \in \mathbb{R}^n$ and all $x \in \mathcal{L}$.

Lemma 4.1. Let $x_1$ be a starting point for which $f$ satisfies Assumptions A. Let $\{x_k\}$ be generated by Algorithm 1 using (3.2), (3.4) or (3.6) update formulas for the diagonal matrix, then $\{D_k\}$ is bounded for each $k > 0$.

Proof. We define

$$\overline{H}_k = \int_0^1 H(x_k + \tau \delta_k)d\tau,$$

then

$$\gamma_k = \overline{H}_k \delta_k.$$

Thus (4.1) and (4.2) give

$$M_1 \|\delta_k\|^2 \leq \gamma_k^\top \delta_k \leq M_2 \|\delta_k\|^2.$$

Since $\overline{H}_k$ is positive definite, then

$$\frac{\|\gamma_k\|^2}{\gamma_k^\top \delta_k} = \frac{\delta_k^\top \overline{H}_k \delta_k}{\delta_k^\top \overline{H}_k \delta_k} \leq \frac{M_2 \|\overline{H}_k^{1/2} \delta_k\|^2}{\|\overline{H}_k^{1/2} \delta_k\|^2}.$$

So,

$$\frac{\|\gamma_k\|^2}{\gamma_k^\top \delta_k} \leq M_2.$$

Let $J_k$ denote the inverse of the diagonal matrix $D_k$, then (3.5) can be rewritten in the next form

$$J_{k+1}^{(i)} = J_k^{(i)} + \frac{(\gamma_k^\top e_i)^2}{\gamma_k^\top \delta_k} - \frac{(J_k^{(i)}(\delta_k^\top e_i))^2}{\delta_k^\top J_k \delta_k}.$$
From (4.5) and by simple expression for the trace and the determinant of the matrix (see [3], [6])

\[
\sum_{i=1}^{i=n} J^{(i)}_{k+1} = \sum_{i=1}^{i=n} J^{(i)}_1 - \sum_{k=1}^{k=m} \frac{\|J_k\delta_k\|^2}{\gamma_k} + \sum_{k=1}^{k=m} \frac{\gamma_k}{\gamma_k}\delta_k = \sum_{i=1}^{i=n} J^{(i)}_1 + \sum_{k=1}^{k=m} \frac{\gamma_k}{\gamma_k}\delta_k.
\]

Let \( j_n \) denote the greatest component of \( J_{k+1} \), from (4.4), (4.6) and the boundlessness of \( J_k \)

\[
j_n \leq \sum_{i=1}^{i=n} J^{(i)}_1 + \sum_{k=1}^{k=m} \frac{\gamma_k}{\gamma_k}\delta_k \leq M_3,
\]

for some positive constant \( M_3 \). From (4.7) we have

\[
\prod_{i=1}^{i=n} J^{(i)}_{k+1} = \left( \prod_{i=1}^{i=n} J^{(i)}_1 \right) \prod_{k=1}^{k=m} \left( \frac{\gamma_k}{\gamma_k}\delta_k \delta_k \right) \cdot \frac{\delta_k}{\delta_k J_k\delta_k} = M_3^{-1}
\]

and using (4.9)

\[
\prod_{i=1}^{i=n} J^{(i)}_{k+1} \geq \left( \prod_{i=1}^{i=n} J^{(i)}_1 \right) \left( \frac{M_4}{M_3} \right)^m \geq M_4
\]

for some positive constant \( M \), from (4.7), (4.10) and if \( j_1 \) denotes the smallest component of \( J_{k+1} \), then

\[
j_1 \geq \frac{M_4}{j_1 \cdots j_n} \geq \frac{M_4}{(M_3)^{n-1}} = M_5,
\]

when \( j_2, \ldots, j_{n-1} \) denote all other components of \( J_{k+1} \). From (4.7) and (4.11) we conclude that for every \( i = 1, 2, \ldots, n \) we have

\[
M_5 \leq j_i \leq M_3,
\]

which implies that the diagonal matrix \( J_{k+1} \) is bounded. \( \square \)

**Theorem 4.1 ([4]).** Under assumptions A for every start point \( x_1 \). Then if \( \{D_k\}, k = 1, 2, \ldots \) is a bounded, positive definite matrix, the Algorithm 1 generates a sequence \( \{x_k\} \) which converges to unique minimum \( x^* \) and there exists a constant \( r, (0 \leq r \leq 1) \) such that

\[
f(x_k) - f(x^*) \leq r^k[f(x_1) - f(x^*)],
\]

i.e. the sequence \( \{x_k\} \) converges to \( x^* \) at a \( r \)-linear rate.
5. Experiments with selected problems

We have focused on the five problems listed in Table 1, these problems were proposed by J. J. Moré, B. S. Garbow and K. E. Hillstrom in [5]. We used FORTRAN 90 code package documented and available in [9]. For the line searches we used $c_1 = 0.3$, $c_2 = 0.7$ for Wolfe line search and $c_1 = 0.3$ for Armijo line search (the step length satisfies only (1.3) condition [1], [6]). The optimization iteration was terminated when

$$\|g(x_k)\| \leq \varepsilon, \quad (\varepsilon \approx 10^{-8})$$

using $m = 5$ and the number of variable $n$ on all these test problems is between 500 and 10,000.

<table>
<thead>
<tr>
<th>Problems</th>
<th>Problem’s name</th>
</tr>
</thead>
<tbody>
<tr>
<td>P.I</td>
<td>Extended Dixon Function</td>
</tr>
<tr>
<td>P.II</td>
<td>Extended Oren Function</td>
</tr>
<tr>
<td>P.III</td>
<td>Extended Powell Singular Function</td>
</tr>
<tr>
<td>P.IV</td>
<td>Extended Rosenbrock Function</td>
</tr>
<tr>
<td>P.V</td>
<td>Extended Wood Function</td>
</tr>
</tbody>
</table>

In the following tables we compare the performance of the L-BFGS method with Oren-Spedicato scaling and others updates where of the diagonal matrix proposed in this work (DFP, BFGS and Inverse BFGS) $n$ denotes the number of variables and the results are reported in the form: number of iterations/total time.

<table>
<thead>
<tr>
<th>Problems</th>
<th>$n$</th>
<th>Oren-Sped.</th>
<th>DFP</th>
<th>BFGS</th>
<th>Inverse BFGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>P.I</td>
<td>500</td>
<td>360/1.53</td>
<td>398/1.76</td>
<td>382/1.50</td>
<td>355/1.48</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>575/2.86</td>
<td>558/2.91</td>
<td>566/2.66</td>
<td>533/2.01</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td>1437/17.41</td>
<td>1328/19.41</td>
<td>1443/17.36</td>
<td>1192/14.82</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>1909/42.26</td>
<td>1720/42.33</td>
<td>1922/40.02</td>
<td>1754/32.62</td>
</tr>
<tr>
<td>P.II</td>
<td>500</td>
<td>47/1.68</td>
<td>342/8.68</td>
<td>–</td>
<td>49/1.64</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>51/4.89</td>
<td>496/24.84</td>
<td>–</td>
<td>57/4.85</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td>229/208.15</td>
<td>–</td>
<td>–</td>
<td>611/208.48</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>301/970.43</td>
<td>–</td>
<td>–</td>
<td>298/763.52</td>
</tr>
<tr>
<td>P.III</td>
<td>500</td>
<td>170/0.25</td>
<td>341/0.36</td>
<td>341/0.29</td>
<td>246/0.28</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>365/2.92</td>
<td>353/5.76</td>
<td>–</td>
<td>357/2.20</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td>536/6.43</td>
<td>3880/9.22</td>
<td>–</td>
<td>356/4.72</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>477/9.89</td>
<td>335/20.43</td>
<td>–</td>
<td>539/10.85</td>
</tr>
</tbody>
</table>
should be noted that in generally the Wolfe line search gives better results than
the BFGS update (3.4) performes well sometimes, but is inefficient in some cases.

Our numerical study indicates that updating the diagonal matrix of the L-BFGS
with inverse BFGS performs better than the Oren-Spedicato scaling (3.2), and
the BFGS update (3.4) performes well sometimes, but is inefficient in some cases. The numerical and theoretical results lead to reject the DFP formula (3.3). It should be noted that in generally the Wolfe line search gives better results than

<table>
<thead>
<tr>
<th>Problems</th>
<th>n</th>
<th>Oren-Sped.</th>
<th>DFP</th>
<th>BFGS</th>
<th>Inverse BFGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>P.IV</td>
<td>500</td>
<td>36/0.16</td>
<td>37/0.60</td>
<td>35/0.11</td>
<td>38/0.16</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>36/1.17</td>
<td>36/0.50</td>
<td>33/0.16</td>
<td>36/0.14</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td>37/0.44</td>
<td>38/1.98</td>
<td>33/0.40</td>
<td>38/0.45</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>37/0.77</td>
<td>37/3.24</td>
<td>36/0.78</td>
<td>36/0.70</td>
</tr>
</tbody>
</table>

Table 3. L-BFGS with Wolfe line search.

<table>
<thead>
<tr>
<th>Problems</th>
<th>n</th>
<th>Oren-Sped.</th>
<th>DFP</th>
<th>BFGS</th>
<th>Inverse BFGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>P.I</td>
<td>500</td>
<td>372/1.57</td>
<td>344/1.37</td>
<td>359/1.55</td>
<td>340/1.19</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>491/2.7</td>
<td>496/3.79</td>
<td>503/3.10</td>
<td>544/3.05</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td>1480/15.18</td>
<td>1353/23.67</td>
<td>1480/15.13</td>
<td>1194/12.90</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>2153/58.61</td>
<td>1911/56.08</td>
<td>2103/52.57</td>
<td>1935/41.79</td>
</tr>
<tr>
<td>P.II</td>
<td>500</td>
<td>39/5.22</td>
<td>–</td>
<td>–</td>
<td>39/5.01</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>88/23.45</td>
<td>–</td>
<td>–</td>
<td>95/23.50</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td>178/839.59</td>
<td>–</td>
<td>–</td>
<td>422/839.54</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>329/1079.55</td>
<td>–</td>
<td>–</td>
<td>402/1022.65</td>
</tr>
<tr>
<td>P.III</td>
<td>500</td>
<td>204/0.99</td>
<td>103/0.93</td>
<td>131/0.77</td>
<td>138/0.75</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>298/5.83</td>
<td>301/3.84</td>
<td>254/2.50</td>
<td>282/2.47</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td>561/10.49</td>
<td>543/9.99</td>
<td>–</td>
<td>484/9.40</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>519/11.35</td>
<td>713/20.54</td>
<td>–</td>
<td>461/10.03</td>
</tr>
<tr>
<td>P.IV</td>
<td>500</td>
<td>35/0.17</td>
<td>29/0.16</td>
<td>36/0.16</td>
<td>35/0.14</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>36/0.22</td>
<td>35/0.22</td>
<td>34/0.21</td>
<td>36/0.21</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td>36/0.81</td>
<td>35/1.27</td>
<td>53/0.83</td>
<td>34/0.76</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>36/1.65</td>
<td>36/2.04</td>
<td>35/1.62</td>
<td>36/1.61</td>
</tr>
<tr>
<td>P.V</td>
<td>500</td>
<td>80/0.33</td>
<td>64/0.32</td>
<td>54/0.27</td>
<td>90/0.33</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>82/0.82</td>
<td>70/0.83</td>
<td>54/0.70</td>
<td>95/0.77</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td>82/1.59</td>
<td>67/1.97</td>
<td>53/1.41</td>
<td>95/1.56</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>59/2.64</td>
<td>37/3.35</td>
<td>46/2.12</td>
<td>52/2.01</td>
</tr>
</tbody>
</table>

6. Conclusions
the Armijo line search type. Finally, it is not clear that the same conclusion would be valid for the case of the constrained optimization problems, but this is certainly an issue that deserves further research.

References


M. L. Sahari, Département de Mathématiques, Université Badji-Mokhtar, Annaba, Algérie, e-mail: mlsahari@yahoo.fr

R. Khaldi, Département de Mathématiques, Université Badji-Mokhtar, Annaba, Algérie, e-mail: rkhadi@yahoo.fr