1985

An Efficient Parallel Solution for Euclidean Shortest Paths in Three Dimensions

Chanderjit Bajaj

Report Number:
85-530
AN EFFICIENT PARALLEL SOLUTION FOR EUCLIDEAN
SHORTEST PATHS IN THREE DIMENSIONS

Chanderjit Bajaj

CSD-TR-530
August 1985
An Efficient Parallel Solution for Euclidean Shortest Paths in Three Dimensions

Chanderjit Bajaj
Department of Computer Science,
Purdue University,
West Lafayette, IN 47907

ABSTRACT

We describe an efficient parallel solution for the problem of finding the shortest Euclidean path between two points in three dimensional space in the presence of polyhedral obstacles. We consider the important case where the order in which the obstacles are encountered in this shortest path is known. In particular for this case we describe an efficient parallel numerical iterative method on a concurrent-read-exclusive-write synchronous shared-memory model. The iterations are essentially convergent non-linear block Gauss-Seidel. For special relative orientations of the, say \( n \), polyhedral obstacles, we further describe a direct method that gives the exact solution in \( O(\log n) \) time using \( n \) processors.

Keywords: Shortest path, polyhedral obstacles, parallel computation
An Efficient Parallel Solution for Euclidean Shortest Paths in Three Dimensions

Chanderjit Bajaj
Department of Computer Science,
Purdue University,
West Lafayette, IN 47907

1. Introduction

The problem of finding the shortest Euclidean path between two points in 3-dimensions, bounded by a finite collection of polyhedral obstacles is a special case of the more general problem of planning optimal collision-free paths for a given robot system. In Euclidean 2-space (the Euclidean plane) the problem is easy to solve and the shortest path is polynomial time computable, Lozano-Perez, Wesley [4]. This problem for Euclidean 3-space is difficult and known shortest path sequential computations are of doubly exponential time, Sharir, Schorr [6].

In this paper we are concerned about the efficient use of parallelism for solving this fundamental motion planning problem. We consider the important subproblem where the order in which the obstacles are encountered in this shortest path is known. Even this is seemingly intractable. The shortest path solution for this problem in general has been shown to be not solvable by radicals\(^\dagger\), and furthermore an exact solution is shown to be not possible even under models of computation where you additionally allow roots of the general fifth degree polynomial equation, Bajaj [1]. This leaves only numerical or symbolic approximations to the shortest path. This paper describes an efficient parallel numerical iterative method for this problem. We assume the concurrent-read exclusive-write synchronous shared-memory parallel machine model of Borodin, von zur Gathen, Hopcroft [3], where on each step each processor can do a single rational addition, subtraction, multiplication or division.

\(^\dagger\) A real number \(\alpha\) is expressible in terms of radicals if there is a sequence of expressions \(\beta_1, \ldots, \beta_n\), where \(\beta_i \in Q\), and each \(\beta_i\) is either a rational or the sum, difference, product, quotient or the \(k^{th}\) root of preceding \(\beta\)'s and the last \(\beta_n\) is \(\alpha\).
The shortest path between two given points, in the presence of polyhedral obstacles, is known to be *polygonal* lines (piecewise straight lines, as for the planar 2-dimensional problem), with break points that lie on the edges of the given polyhedral obstacles. Since the edges of the polyhedral obstacles are arbitrary lines in Euclidean 3-space, the problem of determining the points of contact of the shortest path with these edges can without loss of generality be versed as follows.

**[Shortest Path Problem]** Given a sequence \( L=(L_1, L_2, \ldots, L_n) \) of lines in 3-dimensional space, find the shortest path from a source point \( X \) to a destination point \( Y \) constrained to pass through interior points of each of the lines \( L_1, L_2, \ldots, L_n \) in this order.

We identify three different cases of the relative positions of the lines. All the various configurations of the \( n \) lines in 3-space consist of combinations of these basic orientations between pairs of lines, \( L_i, L_{i+1}, i=1..n-1 \).

(a) Pairwise lines are parallel to each other.

(b) Pairwise lines are not parallel but intersect.

(c) Pairwise lines are skew and do not intersect.

In § 2, we show that when the lines are oriented as a combination of the cases (a) and (b), then the shortest path solution can be computed exactly in time \( O(\log n) \) using \( n \) processors. This corresponds to the solution being algebraically simple, [1]. However in § 3, for the case (c) of skew lines we describe an efficient parallel numerical iterative method. The iterations are essentially convergent non-linear block Gauss-Seidel, [5].

2. Exact Direct Method

Between pairs of lines in 3-space which are parallel to each other there exists a unique plane which contains both of them. The same applies to pairs of lines in 3-space which intersect. Also a point and a line in 3-space define a unique plane between them. The problem of finding the shortest path between \( X \) and \( Y \) in 3-space for cases (a) and (b), then reduces to a constrained 2-1/2 dimensional space problem as follows. Let the point \( X \) and line \( l_1 \) define the plane \( P_1 \), the lines \( L_i \) and \( L_{i+1} \) define the planes \( P_{i+1}, i=1..n-1 \), and the line \( l_n \) and the point \( Y \) define the plane \( P_{n+1} \). The original problem is now reduced to finding the shortest path between two points \( X \) and
Y in 3-space with the path constrained to the planes $P_i, i=1..n+1$. Then the points of contact of the shortest path with the lines $L_i$, the edges of the planes, are determined by first unfolding all the planes $P_i$ so that they all lie on the common plane defined by say, plane $P_1$ containing point $X$. In practice we compute for each plane $P_i$ the displacement $d_i$ and orientation $\theta_i$ defining its position in the common plane $P_1$, relative to some standard fixed plane representation of $P_i$. The shortest path joining $X$ and $Y$ now becomes the shortest plane path that is the straight line, connecting $X$ and $Y'$, (the transformed point $Y$ now on the common plane $P_1$ and thus coplanar with $X$). The points of intersection of this straight line with the unfolded lines $L_i'$, when transformed back, give the points of contact with the lines $L_i$ of our original problem. To prove correctness we note that the length of the the shortest path is kept invariant under such simple planar unfoldings and thus these unfoldings give the unique shortest path.

The unfoldings can be done in $O(\log n)$ parallel time using $n$ processors as follows. Initially there are a sequence of $n$ planes $P_i, i=2..n$, which need to be unfolded onto the chosen common plane $P_1$. There is a processor $m_{pi}$ assigned to each plane. In the first pass every adjacent pair of planes $P_i, P_{i+1}$ is unfolded to become coplanar with $P_1$. In practice processor $m_{pi+1}$ computes the displacement $d_i$ and orientation $\theta_i$ defining the new position of $P_{i+1}$, essentially solving a quadratic equation independent of $n$. At the end of the pass there are now a sequence of $|n/2|$ planes which need to be unfolded. The process is repeated at most $\lfloor \log n \rfloor$ times when all the planes become unfolded on to the common plane $P_1$. As each pass requires $O(1)$ time per processor the entire unfolding is complete in $O(\log n)$ time using $n$ processors. Each processor $m_{pi}$ now computes under the shared-memory model in $O(1)$ time the point of intersection of the straight line shortest path with the unfolded line $L_i'$, and then transforms it back to give the points of contact with the original lines $L_i$ all in a grand total of $O(\log n)$ time.

3. Iterative Numerical Method

Whenever any two adjacent lines $L_i$ and $L_{i+1}$ are skew to one another, there exists no common plane containing both of them. Looking at it differently, the locus (or envelope) of all possible straight line segments connecting skew lines $L_i$ and $L_{i+1}$ is no longer a planar surface but a 3-dimensional volume. Hence the above exact unfolding method does not work. Nevertheless when the lines are skew there exist generalized unfoldings which however provide iterative
approximations of the shortest path solution, Bajaj [2]. Here we describe a parallel numerical iterative method where each iteration is essentially a convergent non-linear block Gauss-Seidel iteration and takes only $O(1)$ time and $|n/2|$ processors.

Let the interior points of each of the given (skew) lines $L_i$ of the shortest path solution which needs to be determined, be respectively $X_i$, $i=1..n$. Each of the points $X_i$ is a vector in 3-dimensional space. Further let $l_i$ be the corresponding unit vectors along the lines $L_i$ and the vectors $X_0, X_{n+1}$ equal the initial given 3-dimensional points $X$ and $Y$ respectively. We wish to minimize the function $D(X_1, ..., X_n) : R^n \rightarrow R^1$ defined below, subject to the $X_i$ constrained to their respective lines $L_i$.

$$D(X_1, ..., X_n) = \sum_{i=0..n} d(X_i, X_{i+1}) \quad (1)$$

Here the $d(X_i, X_{i+1}) = |X_i - X_{i+1}|$, the Euclidean distance between the two points. The above function is strictly convex (see also [6]) and hence the unique solution is given by the following necessary and sufficient conditions. For $i = 1..n$

$$\frac{\delta D}{\delta X_i} = (X_{i-1} - X_i).l_i/|X_{i-1} - X_i| - (X_i - X_{i+1}).l_i/|X_i - X_{i+1}| \quad (2)$$

The above optimality conditions can be interpreted geometrically. For each $i$, the vectors $(X_{i-1} - X_i)$ and $(X_i - X_{i+1})$ subtend equal angles at the lines $L_i$ for the shortest path solution. As $X_i$ is constrained to the line $L_i$ with unit vector $l_i$ we let $X_i = \alpha_i l_i + \beta_i$, where $\alpha_i$ is a scalar and $\beta_i$ is some vector point on the line $L_i$. On substituting this in (2) and simplifying (see Appendix A), we obtain for each $i = 1..n$,

$$a_i \alpha_i^2 + b_i \alpha_i + c_i = 0 \quad (3)$$

where the $a_i$, $b_i$ and $c_i$ are at most fourth degree polynomial functions of $X_{i-1}$ and $X_{i+1}$ as shown below.

$$a_i = ((X_{i-1} - \beta_i).l_i)^2 + (\beta_i - X_{i+1}).(\beta_i - X_{i+1}) - (\beta_i - X_{i+1}).l_i)^2 - (\beta_i - X_{i-1}).(\beta_i - X_{i-1})$$

$$b_i = 2((X_{i-1} - \beta_i).l_i)((X_{i-1} - \beta_i).l_i)((\beta_i - X_{i+1}).l_i) - (\beta_i - X_{i+1}).(\beta_i - X_{i+1}))$$
We now obtain our parallel iterative computation of the $X_i$, $i=1..n$, as follows. (In the following let $m$ take on all the odd values from 1..n and $n$ take on all the even values from 1..n.)

Given initial values of each of the $X_i^0$, in the general $k$th iteration,

1. Compute for all the $m$ in parallel, the values $a_{m}^{k+1}$ and set $X_{m}^{k+1} = a_{m}^{k+1}l_{m} + \beta_{m}$.

2. Then compute for all the $n$ in parallel, the values $a_{n}^{k+1}$ and set $X_{n}^{k+1} = a_{n}^{k+1}l_{n} + \beta_{n}$.

Each of the $\alpha_i$ is governed by a quadratic equation (3) above and can be computed exactly by the quadratic formula. The two solutions of $\alpha_i$ obtained is a direct consequence of the squaring of equation (a) in Appendix A during simplification of (2) to remove the square roots of the Euclidean distance norm. One of these solutions corresponds to the right hand side of equation (a) being taken positive and the other to the right hand side being taken negative. We of course at each stage, conforming to our original minimality condition (2), select the $\alpha_i$ solution corresponding to the positive right hand side of the equation (a). The correctness of the above parallel algorithm follows directly from the fact that each of the $\alpha_i$ is a function of only $X_{i-1}$ and $X_{i+1}$.

From Ortega, Rheinboldt [5, p219], we obtain the following facts. If $F: \mathbb{R}^n \to \mathbb{R}^n$ has components $f_1, \ldots, f_n$ then the basic step of the non-linear Gauss-Seidel iteration is to solve the $i$th equation

$$f_i(X_{i}^{k+1}, \ldots, X_{i-1}^{k+1}, X_{i}, X_{i+1}, \ldots, X_{n}^{k}) = 0$$

for $X_i$ and then to set $X_i^{k+1} = X_i$. Thus in order to obtain all the $X_{i}^{k+1}$ from the $X_{i}^{k}$ we solve successively the $n$ one-dimensional non-linear equations above for $i=1..n$. Since in our case each of the $f_i$ is a function of only $X_{i-1}$, $X_i$ and $X_{i+1}$ and since all the $f_i$ are identical, we can solve $\lfloor n/2 \rfloor$ of them in parallel at a time. Also as is the Gauss-Seidel principle of using new information as soon as it is available, in our case step (2) of the algorithm uses the most recent iterate.
values of the $X_i$. To draw the analogy tighter between the above parallel algorithm and the Gauss-Seidel method, consider further the non-linear block Gauss-Seidel procedure, Ortega, Rheinboldt [5, p225]. In this a complete iteration requires the solution of $s$ non-linear system of dimension $q_i$, $i=1..s$. The components $f_i$ of $F$ are grouped into mappings $F_i: R^n \rightarrow R^{q_i}$, and the $X$ partitioned into $(X^1, \ldots, X^s)$ with $X^i \in R^{q_i}$. Then

$$F_i((X^1)^{k+1}, \ldots, (X^i)^{k+1}, (X^{i+1})^k, \ldots, (X^s)^k) = 0$$

The above algorithm is thus non-linear block Gauss-Seidel for $q_i=|n/2|$ and $m=2$, the two blocks being the systems for the odd and even values of $i=1..n$ of (3) above.

$$a_1a_2 + b_1a_1 + c_1 = 0$$
$$a_3a_2 + b_3a_3 + c_3 = 0$$
$$a_5a_2 + b_5a_5 + c_5 = 0$$
$$a_2a_2 + b_2a_2 + c_2 = 0$$
$$a_4a_2 + b_4a_4 + c_4 = 0$$
$$a_6a_2 + b_6a_6 + c_6 = 0$$

The above iterative parallel procedure was implemented and efficiently converged to the optimal solution for a variety of input skew lines and various values of precision and error bounds. The results are summarized here for both the $l_1$ and the $l_\infty$ error norms. The initial values for these iterations were computed in a single parallel step using equation (3) and in each case letting $X_{i-1} = X_0$ and $X_{i+1} = X_{n+1}$, the two given input points.
<table>
<thead>
<tr>
<th>Error Bound</th>
<th>N=2</th>
<th>N=3</th>
<th>N=4</th>
<th>N=5</th>
<th>N=6</th>
<th>N=7</th>
<th>N=8</th>
<th>N=9</th>
<th>N=10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0e-01</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>1.0e-02</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>1.0e-03</td>
<td>5</td>
<td>7</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>1.0e-04</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>1.0e-05</td>
<td>6</td>
<td>10</td>
<td>9</td>
<td>11</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>1.0e-06</td>
<td>7</td>
<td>11</td>
<td>11</td>
<td>13</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>1.0e-07</td>
<td>8</td>
<td>13</td>
<td>12</td>
<td>15</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>1.0e-08</td>
<td>9</td>
<td>14</td>
<td>14</td>
<td>17</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>1.0e-09</td>
<td>10</td>
<td>16</td>
<td>15</td>
<td>19</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>1.0e-10</td>
<td>11</td>
<td>18</td>
<td>17</td>
<td>22</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>1.0e-11</td>
<td>12</td>
<td>19</td>
<td>18</td>
<td>24</td>
<td>21</td>
<td>21</td>
<td>21</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>1.0e-12</td>
<td>13</td>
<td>21</td>
<td>20</td>
<td>26</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>1.0e-13</td>
<td>14</td>
<td>22</td>
<td>21</td>
<td>28</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>1.0e-14</td>
<td>15</td>
<td>24</td>
<td>23</td>
<td>30</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>1.0e-15</td>
<td>16</td>
<td>25</td>
<td>24</td>
<td>32</td>
<td>28</td>
<td>28</td>
<td>28</td>
<td>28</td>
<td>28</td>
</tr>
</tbody>
</table>

Table 1: Summary of results of the parallel program under the $l_1$ norm
<table>
<thead>
<tr>
<th>Error Bound</th>
<th>N=2</th>
<th>N=3</th>
<th>N=4</th>
<th>N=5</th>
<th>N=6</th>
<th>N=7</th>
<th>N=8</th>
<th>N=9</th>
<th>N=10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0e-01</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>1.0e-02</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>1.0e-03</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>1.0e-04</td>
<td>5</td>
<td>8</td>
<td>7</td>
<td>9</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>1.0e-05</td>
<td>6</td>
<td>10</td>
<td>9</td>
<td>11</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>1.0e-06</td>
<td>7</td>
<td>11</td>
<td>10</td>
<td>13</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>1.0e-07</td>
<td>8</td>
<td>13</td>
<td>12</td>
<td>15</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>1.0e-08</td>
<td>9</td>
<td>14</td>
<td>13</td>
<td>17</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>1.0e-09</td>
<td>10</td>
<td>16</td>
<td>15</td>
<td>19</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>1.0e-10</td>
<td>11</td>
<td>17</td>
<td>16</td>
<td>21</td>
<td>18</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>1.0e-11</td>
<td>12</td>
<td>19</td>
<td>18</td>
<td>23</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>1.0e-12</td>
<td>13</td>
<td>21</td>
<td>19</td>
<td>25</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>1.0e-13</td>
<td>14</td>
<td>22</td>
<td>21</td>
<td>28</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>1.0e-14</td>
<td>15</td>
<td>24</td>
<td>22</td>
<td>30</td>
<td>25</td>
<td>26</td>
<td>25</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>1.0e-15</td>
<td>16</td>
<td>25</td>
<td>24</td>
<td>32</td>
<td>27</td>
<td>27</td>
<td>26</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>1.0e-16</td>
<td>16</td>
<td>27</td>
<td>26</td>
<td>34</td>
<td>28</td>
<td>30</td>
<td>30</td>
<td>29</td>
<td>29</td>
</tr>
</tbody>
</table>

Table 2: Summary of results of the parallel program under the $l_{\infty}$ norm

Acknowledgements: I wish to thank Walter Gautschi, Elias Houstis, and John Rice for enlightening discussions in numerical analysis and Myung Soo Kim for his timely help with the program.

4. References


We now rationalize and simplify the optimality equation (2) of the text.

\[(X_{i-1}-X_i).l_i/|X_{i-1}-X_i| = (X_i-X_{i+1}).l_i/|X_i-X_{i+1}| \quad (a)\]

As \(X_i\) is constrained to the line \(L_i\) with unit vector \(l_i\) we let \(X_i=\alpha_i l_i+\beta_i\), where \(\alpha_i\) is a scalar and \(\beta_i\) is some vector point on the line \(L_i\). On substituting this in (a) and simplifying we obtain

\[(X_{i-1}-\alpha_i l_i-\beta_i).l_i/|X_{i-1}-\alpha_i l_i-\beta_i| = (\alpha_i l_i+\beta_i-X_{i+1}).l_i/|\alpha_i l_i+\beta_i-X_{i+1}|\]

Squaring both sides of the above equation and rationalizing

\[((X_{i-1}-\beta_i).l_i-\alpha_i)^2((X_{i+1}-\beta_i).(X_{i+1}-\beta_i) - 2\alpha_i(X_{i+1}-\beta_i).l_i + \alpha_i^2)\]

\[= ((\beta_i-X_{i+1}).l_i+\alpha_i)^2((X_{i+1}-\beta_i).(X_{i+1}-\beta_i) - 2\alpha_i(X_{i+1}-\beta_i).l_i + \alpha_i^2)\]

Simplifying the above equation we obtain a polynomial equation in terms of the scalar \(\alpha_i\),

\[a_4\alpha_i^4 + a_3\alpha_i^3 + a_2\alpha_i^2 + a_1\alpha_i + a_0 = 0\]

where the coefficients are given by

\[a_4 = 0 \quad a_3 = 0\]

\[a_2 = ((X_{i-1}-\beta_i).l_i)^2 + (\beta_i-X_{i+1}).(\beta_i-X_{i+1}) - ((\beta_i-X_{i+1}).l_i)^2 - (\beta_i-X_{i-1}).(\beta_i-X_{i-1})\]
\[ a_1 = 2((X_{i-1}-\beta_i).l_i)((X_{i-1}-\beta_i).l_i)((\beta_i-X_{i+1}).l_i) - (\beta_i-X_{i+1}).(\beta_i-X_{i+1})] \\
- 2((X_{i+1}-\beta_i).l_i)((X_{i+1}-\beta_i).l_i)((\beta_i-X_{i-1}).l_i) - (\beta_i-X_{i-1}).(\beta_i-X_{i-1})] \\
\]
\[ a_0 = ((X_{i-1}-\beta_i).l_i)^2(\beta_i-X_{i+1}).(\beta_i-X_{i+1}) - ((X_{i+1}-\beta_i).l_i)^2(\beta_i-X_{i-1}).(\beta_i-X_{i-1})] \]