Techniques of the simplex basis LU factorization update

**Daniela Renata Cantane**¹
Electric Engineering and Computation School (FEEC), State University of Campinas (UNICAMP), São Paulo, Brazil

**Aurelio Ribeiro Leite de Oliveira**²
Institute of Mathematics, Statistics and Science of Computation (IMECC), State University of Campinas (UNICAMP), São Paulo, Brazil

**Christiano Lyra Filho**³
Electric Engineering and Computation School (FEEC), State University of Campinas (UNICAMP), São Paulo, Brazil

**Abstract**

The objective of this work is to compare the developed LU factorization update with results from MINOS. This technique, uses a matrix columns static reordering and they are rearranged in accordance with the increasing number of nonzero entries and triangularized, leading to sparse basis factorization without computational effort to reorder the columns. Only the columns factorizations actually modified by the change of basis are carried through due to matrix sparse structure. Computational results in Matlab for problems from the Netlib show that this is a very promising idea, since there is no need to refactorize the matrix in the tested problems.

**Keywords:** linear optimization, sparse matrix, factorization update, simplex.

1. Introduction

The efficient solution of large-scale linear systems is very important for solving linear optimization problems. These systems can be approached through generic methods as, for example, LU factorization of the basis and its update, or through the problem specific structure exploitation, as in the network flow problem [5].

In [2] the column reordering in the combined cutting stock and lot sizing problems [8] is done in such a way that the resulting matrix is block diagonal. Therefore, it is easy to factor the base, resulting in little fill-in. Through the static reordering of columns, it is possible to obtain a sparse factorization of the basis, without any overhead to determine the order of the columns, since the base columns follow the given ordering.

In the LU update, the floating-point operations originated by the column that leaves the base are undone, in the reverse order of the factorization, always considering sparsity.

Finally, it is necessary to compute the factored columns after the entering column, also considering the sparse pattern.

¹ Av. Albert Einstein, 400, CEP: 13083-852, Campinas, SP, Brazil. E-mail: dcantane@densis.fee.unicamp.br
² Sérgio Buarque de Holanda, 651, CEP: 13081-970, Campinas, SP, Brazil. E-mail: aurelio@ime.unicamp.br
³ Av. Albert Einstein, 400, CEP: 13083-852, Campinas, SP, Brazil. E-mail: chrlyra@densis.fee.unicamp.br
For the combined problem, results obtained in [2] have shown that this approach is fast and robust, introducing insignificant accumulated rounding errors in worst case situations, after thousands of iterations.

2. LU Factorization Update Methods

The LU factorization update technique with partial pivot was proposed by [1]. Two variants of this algorithm proposed in [9] aim to balance the sparsity and numeric stability in the factorization. The latter variant is an improvement over the former.

Several basic matrix LU factorization update was implemented [10] (in portuguese) with the objective to balance data original sparsity, it is conclude that the LU factorization update number carried through in the following iterations it influences considerably in the solved time of the problems, in the fill-ins and even iterations numbers of the simplex method.

In [7], the LU factorization update proposed by [6] and its application in the simplex update basis method is described in detail. Moreover, it presents the ideas of an efficient implementation proposed by [11], which is a good combination of the symbolic and numerical phase of the pivoting and presents a compromise between sparsity and numerical stability. Other techniques updates are describe in [4].

3. Implementation Issues

The objective of the implementation developed here is to simulate simplex iterations using the static reordering and the LU update adopted in [2] for general linear optimization problems.

The base sequence obtained from MINOS is used in the simulation. A procedure for finding the initial basis, “Crash” base described in [7] and the leaving and entering columns of the basis from MINOS’ output was implemented. The matrix is triangularized, the columns are reordered according to the number of nonzero entries, in increasing order, leading to sparse factorizations without computational effort to obtain the order of columns, since the reordering of the matrix is static and basis columns follow this ordering.

Two column update approaches are used in the implementation and the number of nonzero entries for each one is compared. When the entering column \( e \) in the basis follows the static ordering the approach is called \( U_1 \). In the second approach, called \( U_2 \), the entering column \( e \) entries in the basis in the position of the leaving column \( l \). The notation \( U_{\text{MINOS}} \) is the used for MINOS results.

In the \( U_1 \) approach, a LU factorization of the basis considering its sparsity, called \( F_1 \), is performed. Only the factored columns actually modified by the change of the basis are carried through. Notice that there is no changes in column \( k \) located after the entering and/or leaving column if \( LU(k,e) = 0 \) or \( LU(k,l) = 0 \) due to sparse structure of columns involved.

In the \( U_{\text{MINOS}} \) approach, a complete LU factorization of the basis to simulate the MINOS’ factorization, called \( F_{\text{MINOS}} \), is carried through.

The introduced error estimation in the factorization was computed with the objective of verify the robustness of the LU factorization method. Each factorization is completely undone in every simplex method iteration. Thus, the operations are performed in the reverse order of the LU factorization and the matrix that will simulate the basis factorization update is obtained from the one being factored from the very first column. Therefore, the original basis
is obtained with some amount of numerical error and the accumulated errors are a worst case estimation for each iteration.

4. Numerical Experiments

Initially, it was carried through numeric experiments for verifying the efficiency of the basis column update approaches presented in the previous section. Table 1 shows iterations (phase 2) and factorizations number of a subset of Netlib problems. Tables 2 and 3 show the nonzero number entries of the basis factorization of the same linear optimization problems using and not using the Crash basis, respectively.

<table>
<thead>
<tr>
<th>LO Problems</th>
<th>Updates</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>With Crash</td>
</tr>
<tr>
<td>kb2</td>
<td>3</td>
<td>55</td>
</tr>
<tr>
<td>adlittle</td>
<td>3</td>
<td>67</td>
</tr>
<tr>
<td>sc205</td>
<td>2</td>
<td>52</td>
</tr>
<tr>
<td>israel</td>
<td>3</td>
<td>235</td>
</tr>
<tr>
<td>scsd1</td>
<td>8</td>
<td>314</td>
</tr>
<tr>
<td>bandm</td>
<td>3</td>
<td>297</td>
</tr>
<tr>
<td>scfxm1</td>
<td>2</td>
<td>138</td>
</tr>
<tr>
<td>beaconfd</td>
<td>1</td>
<td>26</td>
</tr>
<tr>
<td>scsd6</td>
<td>14</td>
<td>917</td>
</tr>
</tbody>
</table>

Table 1: Linear optimization problems data.

<table>
<thead>
<tr>
<th>Nonzero entries</th>
<th>Nonzero entries</th>
<th>Maximum</th>
<th>Minimum</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$U_1$</td>
<td>$U_2$</td>
<td>$U_{\text{MINOS}}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$U_1$</td>
<td>$U_2$</td>
<td>$U_{\text{MINOS}}$</td>
</tr>
<tr>
<td>kb2</td>
<td>353</td>
<td>460</td>
<td>406</td>
<td>257</td>
</tr>
<tr>
<td>adlittle</td>
<td>406</td>
<td>501</td>
<td>501</td>
<td>367</td>
</tr>
<tr>
<td>sc205</td>
<td>1242</td>
<td>1795</td>
<td>897</td>
<td>644</td>
</tr>
<tr>
<td>israel</td>
<td>2187</td>
<td>6116</td>
<td>3536</td>
<td>2173</td>
</tr>
<tr>
<td>scsd1</td>
<td>689</td>
<td>1274</td>
<td>907</td>
<td>602</td>
</tr>
<tr>
<td>bandm</td>
<td>5508</td>
<td>9674</td>
<td>4163</td>
<td>3076</td>
</tr>
<tr>
<td>scfxm1</td>
<td>2250</td>
<td>4721</td>
<td>2059</td>
<td>1692</td>
</tr>
<tr>
<td>beaconfd</td>
<td>1423</td>
<td>1845</td>
<td>1217</td>
<td>1194</td>
</tr>
<tr>
<td>scsd6</td>
<td>1618</td>
<td>2997</td>
<td>1934</td>
<td>1168</td>
</tr>
</tbody>
</table>

Table 2: Nonzero entries number with crash basis.
Table 3: Nonzero entries number without crash basis.

<table>
<thead>
<tr>
<th>Nonzero entries</th>
<th>Maximum</th>
<th>Minimum</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$U_1$</td>
<td>$U_2$</td>
<td>$U_{MINOS}$</td>
</tr>
<tr>
<td>kb2</td>
<td>357</td>
<td>612</td>
<td>387</td>
</tr>
<tr>
<td>adlittle</td>
<td>399</td>
<td>697</td>
<td>528</td>
</tr>
<tr>
<td>sc205</td>
<td>1273</td>
<td>2094</td>
<td>1221</td>
</tr>
<tr>
<td>israel</td>
<td>1994</td>
<td>5840</td>
<td>2933</td>
</tr>
<tr>
<td>scsd1</td>
<td>859</td>
<td>1215</td>
<td>1013</td>
</tr>
<tr>
<td>bandm</td>
<td>5287</td>
<td>22422</td>
<td>4173</td>
</tr>
<tr>
<td>scfxm1</td>
<td>2316</td>
<td>4718</td>
<td>2090</td>
</tr>
<tr>
<td>beaconfd</td>
<td>1615</td>
<td>3744</td>
<td>1485</td>
</tr>
<tr>
<td>scsd6</td>
<td>1744</td>
<td>3644</td>
<td>1975</td>
</tr>
</tbody>
</table>

The $U_1$ basis update approach reduces up to 52% for the problem scfxm1 and 78% for the problem bandm nonzero entries, in comparison to the $U_1$ approach in the Tables 2 and 3, respectively. Thus, $U_1$ is used to carry through the $LU$ factorization proposed in this work. The $U_1$ update approach is a little more dense that the $U_{MINOS}$ update in a few cases, but as we shall see it not need to refactorize the base. Thus, it probably will be faster than other approaches.

The Table 4 shows the flop number (floating point operation count) of each factorization presented in the previous section. MINOS' Crash option is turned off.

Table 4: Flops number of $LU$ factorization.

<table>
<thead>
<tr>
<th>Flop Number</th>
<th>Maximum</th>
<th>Average</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$F_1$</td>
<td>$F_{MINOS}$</td>
<td>$F_1$</td>
</tr>
<tr>
<td>kb2</td>
<td>2289</td>
<td>3624</td>
<td>1462</td>
</tr>
<tr>
<td>adlittle</td>
<td>2492</td>
<td>5358</td>
<td>2103</td>
</tr>
<tr>
<td>israel</td>
<td>12585</td>
<td>30456</td>
<td>10010</td>
</tr>
<tr>
<td>scsd1</td>
<td>5279</td>
<td>11829</td>
<td>3635</td>
</tr>
<tr>
<td>bandm</td>
<td>41178</td>
<td>88358</td>
<td>27961</td>
</tr>
<tr>
<td>scfxm1</td>
<td>13919</td>
<td>21071</td>
<td>12436</td>
</tr>
<tr>
<td>beaconfd</td>
<td>9701</td>
<td>9948</td>
<td>8898</td>
</tr>
<tr>
<td>scsd6</td>
<td>10298</td>
<td>23511</td>
<td>6199</td>
</tr>
</tbody>
</table>

The $F_1$ factorization, proposed in this work, reduces the flop number with respect to the $F_{MINOS}$ factorization.

In Table 5, the error estimate introduced by these operations was verified, computing the norm of the difference between the basis obtained by completely undoing the factorization and the original basis obtained directly from the constraint matrix.
<table>
<thead>
<tr>
<th>Error</th>
<th>Maximum</th>
<th>Minimum</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>kb2</td>
<td>1.7e-14</td>
<td>0</td>
<td>1.0e-14</td>
</tr>
<tr>
<td>adlittle</td>
<td>3.4e-16</td>
<td>1.1e-16</td>
<td>2.5e-16</td>
</tr>
<tr>
<td>sc205</td>
<td>3.5e-16</td>
<td>0</td>
<td>1.2e-16</td>
</tr>
<tr>
<td>israel</td>
<td>4.6e-12</td>
<td>0</td>
<td>2.4e-13</td>
</tr>
<tr>
<td>scsd1</td>
<td>7.3e-16</td>
<td>1.4e-16</td>
<td>3.3e-16</td>
</tr>
<tr>
<td>bandm</td>
<td>1.9e-13</td>
<td>5.7e-14</td>
<td>1.1e-14</td>
</tr>
<tr>
<td>scfxm1</td>
<td>1.1e-14</td>
<td>2.1e-15</td>
<td>7.5e-15</td>
</tr>
<tr>
<td>beaconfd</td>
<td>2.4e-16</td>
<td>2.8e-17</td>
<td>7.0e-17</td>
</tr>
<tr>
<td>scsd6</td>
<td>1.0e-15</td>
<td>1.4e-16</td>
<td>3.2e-16</td>
</tr>
</tbody>
</table>

Table 5: Error estimate of updated basis without crash basis.

It can be concluded that the factorization update proposed method is very robust. The maximum accumulated error in the worst case is of the order of $10^{-12}$, that is, in this approach it is not necessary at all to refactorize the basis any time, as it is usually done by the tradicional methods due to robustness and sparsity considerations.

5. Conclusions

In this work a static reordering of matrix columns is proposed, leading to simplex base with sparse $LU$ factorizations and inexpensive factorization updates. The reordering has no initialization or updating costs since there is no need to reorder the columns in the factorization.

Stable results are present in Table 5. It is safe to conclude that no periodical factorization is needed for these problems as it is usually for updating schemes [4] and [7]. As a result, if the starting basis is the identity, is not necessary to compute any $LU$ factorization at all.

In spite of the new basis update approach obtain a little less sparse basis than MINOS, it does not need to perform periodic refactorizations. Thus, it will probably obtain better running time in the same computational environment since it takes less computational effort. For future work this approach and factorization will be integrated to an implementation such as MINOS or GLPK.

Acknowledgement

This research was sponsored by the Foundation for the Support of Research of the State of São Paulo (FAPESP) and the Brazilian Council for the Development of Science and Technology (CNPq).

References


