Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

AMBROS M. GLEIXNER

Factorization and update of a reduced basis matrix for the revised simplex method

ZIB-Report 12-36 (October 2012)

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7
D-14195 Berlin-Dahlem

Telefon: 030-84185-0
Telefax: 030-84185-125

e-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

Factorization and update of a reduced basis matrix for the revised simplex method

Ambros M. Gleixner*

October 2012

Abstract

In this paper, we describe a method to enhance the FTRAN and BTRAN operations in the revised simplex algorithm by using a reduced basis matrix defined by basic columns and nonbasic rows. This submatrix of the standard basis matrix is potentially much smaller, but may change its dimension dynamically from iteration to iteration.

For the classical product form update ("eta update"), the idea has been noted already by Zoutendijk, but only preliminarily tested by Powell in the early 1970s. We extend these ideas to Forrest-Tomlin type update formulas for an LU factorization of the reduced basis matrix, which are suited for efficient implementation within a state-of-the-art simplex solver. The computational advantages of the proposed method apply to pure LP solving as well as to LP-based branch-cut-and-price algorithms. It can easily be integrated into existing simplex codes.

1 Introduction

Since Dantzig's initial formulation of the simplex method for solving linear programs (LPs), the algorithm has seen numerous computational improvements. One example is the so-called revised simplex method given by Dantzig [2] himself. For an LP with m inequality constraints and n nonnegative variables, the original tableau simplex has to maintain a matrix of dimension $m \times (n+m)$. In the revised simplex, a matrix of dimension $m \times m$, a representation of the so-called basis inverse, suffices.

This paper discusses a method to employ an even smaller basis matrix defined by the basic columns and nonbasic rows of the constraint matrix. The dimension of this *reduced basis matrix* varies throughout the algorithm and never exceeds $\min\{m,n\} \times \min\{m,n\}$. This approach unifies the simplex variants applied to the different computational representations of an LP—column and row form—as considered by Nazareth [8] and Wunderling [11]. Both for pure LP solving, in particular for problems with more rows than columns, as well as for LP-based branch-cut-and-price algorithms, it shows various computational advantages.

For the classical product form update, this idea has been noted already in the early 1970s by Zoutendijk [13]. Soon after, Powell [9] has reported on a simplex implementation featuring the reduced basis matrix and has given computational evidence based on a small set of three instances showing that it helps to reduce the number of nonzeros in the factorization significantly. For the next decades, however, these results have seemingly been disregarded and the idea has not been adopted by commercial simplex codes.

*Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany, gleixner@zib.de.

In this paper, we extend the approach to Forrest-Tomlin type update formulas for an LU factorization of the reduced basis matrix, which are suited for efficient implementation within a state-of-the-art simplex solver. This research is conducted independently from recent developments of Wunderling [12], who has presented a new implementation of the simplex algorithm based on the idea of the reduced basis matrix. While he proposes to redesign the algorithmic structure of the simplex algorithm, we emphasize that the factorization of the reduced basis matrix and its update can be integrated seamlessly into existing simplex codes.

The paper is organized as follows: In Sec. 2, we introduce notation and revisit the linear systems of equations solved during the revised simplex method. Sec. 3 presents the basic idea of how to avoid the artificial unit structure in the column and row basis matrices by using the reduced basis matrix. In Sec. 4, we provide update formulas for an LU factorization of the reduced basis matrix, in particular for the cases when its dimension changes after a pivot step. To conclude, Sec. 5 discusses the computational benefits of the new method.

2 The revised simplex algorithm

2.1 Notation

For clarity of presentation, let us consider a linear program in form

$$\min\{c'x : Ax \geqslant b,\, x \geqslant 0\} \tag{P}$$

with $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. Its dual reads

$$\max\{b'y : A'y \leqslant c,\, y \geqslant 0\}. \tag{D}$$

The feasible regions of (P) and (D) each form a polyhedron defined by $m + n$ inequalities. For given index sets $\mathcal{I} \subseteq \{1,\dots,m\}$ and $\mathcal{J} \subseteq \{1,\dots,n\}$ of constraints and variables of problem (P), respectively, we denote the corresponding submatrix of A by $A_{\mathcal{I},\mathcal{J}} := (a_{ij})_{i\in\mathcal{I},j\in\mathcal{J}} \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{J}|}$. The index set of all columns or rows is abbreviated by '$\cdot$'. Hence, the i-th row vector and j-th column vector are denoted by $A_{i\cdot}$ and $A_{\cdot j}$, respectively. For a vector $d \in \mathbb{R}^n$, we write $d_{\mathcal{I}} := (d_i)_{i\in\mathcal{I}} \in \mathbb{R}^{|\mathcal{I}|}$. The identity matrix is denoted by I, the i-th unit column vector by e_i, the vector of all zeros by 0. Dimensions are usually clear from the context. The transpose of a vector or matrix is denoted by '$'$'.

Definition 1 (basis, basic solution). *Let $\mathcal{C} \subseteq \{1,\dots,n\}$ and $\mathcal{R} \subseteq \{1,\dots,m\}$ be index sets of variables and constraints of* (P)*, respectively.*

1. *We call $(\mathcal{C},\mathcal{R})$ a* basis *of* (P) *if $|\mathcal{C}| + |\mathcal{R}| = m$. Variables and constraints with index in $\bar{\mathcal{C}} := \{1,\dots,n\} \setminus \mathcal{C}$ and $\bar{\mathcal{R}} := \{1,\dots,m\} \setminus \mathcal{R}$, respectively, are called* nonbasic*.*

2. *We call a basis $(\mathcal{C},\mathcal{R})$* regular *if the vectors $A_{\cdot j}$, $j \in \mathcal{C}$, and e_i, $i \in \mathcal{R}$, are linearly independent.*

3. *We call a primal-dual pair $(x, y) \in \mathbb{R}^n \times \mathbb{R}^m$ a* basic solution *of* (P) *if there exists a regular basis $(\mathcal{C},\mathcal{R})$ such that*

$$x_j = 0,\ j \notin \mathcal{C}, \tag{1}$$
$$A_{i\cdot}x = b_i,\ i \notin \mathcal{R}, \tag{2}$$
$$y'A_{\cdot j} = c'_j,\ j \in \mathcal{C}, \tag{3}$$
$$y_i = 0,\ i \in \mathcal{R}. \tag{4}$$

4. *A basic solution* (x, y) *is called* primal feasible *if* $Ax \geqslant b$, $x \geqslant 0$ *and* dual feasible *if* $A'y \leqslant c$, $y \geqslant 0$.

While the primal simplex algorithm moves between primal feasible basic solutions trying to decrease their objective function value $c'x$ in each step, the dual simplex proceeds along basic solutions which are dual feasible and tries to increase the objective of the dual $b'y$. As soon as a solution is reached that is both primal and dual feasible, the algorithm concludes optimality. Infeasibility and unboundedness can be detected and certified.

A characteristic feature of the simplex algorithm is the interplay between its discrete and continuous nature. While the basic solutions live in continuous space, they have a discrete representation in the form of a basis. The following lemma shows that any regular basis uniquely determines a basic solution.

Lemma 1. Let $(\mathcal{C}, \mathcal{R})$ be a basis of (P), then the vectors $A_{\cdot j}$, $j \in \mathcal{C}$, and e_i, $i \in \mathcal{R}$, are linearly independent if and only if the vectors $A_{i\cdot}$, $i \notin \mathcal{R}$, and e'_j, $j \notin \mathcal{C}$, are linearly independent.

Proof. $\det(A_{\cdot\mathcal{C}} \,|\, I_{\cdot\mathcal{R}}) = \det(A_{\bar{\mathcal{R}},\mathcal{C}}) \det(I) = \det \begin{pmatrix} A_{\bar{\mathcal{R}}\cdot} \\ I_{\bar{\mathcal{C}}\cdot} \end{pmatrix}$. □

Hence, for a regular basis the system of equations 1–4 has full rank and a unique solution (x, y).

2.2 Computational forms

Implementations of the simplex algorithm typically cast (P) in a computationally suited form. They differ in how the values of a basic solution (x, y) are computed or updated, given a regular basis $(\mathcal{C}, \mathcal{R})$.

Dantzig [3] designed the simplex method for what we call the *column form* of (P), which continues to be the basis for most state-of-the-art implementations. Here, we introduce slack variables $s \in \mathbb{R}^m$ in order to obtain equality constraints:

$$\min\{c'x : Ax - s = b,\, x, s \geqslant 0\}.$$

Given a regular basis $(\mathcal{C}, \mathcal{R})$, the variables x_j, $j \notin \mathcal{C}$, and s_i, $i \notin \mathcal{R}$, are set to zero as prescribed by eqs. 1 and 2. The m remaining columns $A_{\cdot j}$, $j \in \mathcal{C}$, and $-e_i$, $i \in \mathcal{R}$, form the (full rank) basis matrix $\mathbf{M} = (A_{\cdot\mathcal{C}} \,|\, -I_{\cdot\mathcal{R}}) \in \mathbb{R}^{m\times m}$. The values of the basic solution can then be computed by solving the two systems of linear equations

$$\mathbf{M} \begin{pmatrix} x_{\mathcal{C}} \\ s_{\mathcal{R}} \end{pmatrix} = b$$

and

$$\mathbf{M}'y = \begin{pmatrix} c_{\mathcal{C}} \\ 0 \end{pmatrix},$$

called FTRAN and BTRAN, respectively.

Treating variable bounds as inequality constraints, we obtain the so-called *row form*, which was first investigated and implemented by Wunderling [11]:

$$\min\left\{c'x : \begin{pmatrix} I \\ A \end{pmatrix} x \geqslant \begin{pmatrix} 0 \\ b \end{pmatrix}\right\}.$$

Here, we form the basis matrix $\mathbf{N} \in \mathbb{R}^{n\times n}$ using rows e'_j, $j \notin \mathcal{C}$ and $A_{i\cdot}$, $i \notin \mathcal{R}$. This lets us compute the primal vector x by solving

$$\mathbf{N}x = \begin{pmatrix} 0 \\ b_{\bar{\mathcal{R}}} \end{pmatrix},$$

which is identical to eqs. 1 and 2. If $z \in \mathbb{R}^n$ denotes the dual multipliers associated with the bound constraints $x \geqslant 0$, then the dual vector y is computed solving

$$\mathbf{N}' \begin{pmatrix} z_{\bar{\mathcal{C}}} \\ y_{\bar{\mathcal{R}}} \end{pmatrix} = c,$$

complemented with $y_i = 0$ for $i \in \mathcal{R}$. Note that if the numbers of variables and constraints differ widely, so do the dimensions of $\mathbf{M}$ and $\mathbf{N}$ and subsequently the effort for computing and updating the basic solution values.

To refer to the basic and nonbasic columns of the auxiliary matrix $(A\,|\,-I)$ directly, we define the index set $\mathcal{B}{:=}\mathcal{C} \cup \{i+n \mid i \in \mathcal{R}\} \subseteq \{1,\ldots,m+n\}$. Then $\mathbf{M}$ can be written as $(A\,|\,-I)_{\cdot\mathcal{B}}$ and $\mathbf{N}' = (I\,|\,A')_{\cdot\bar{\mathcal{B}}}$, where $\bar{\mathcal{B}} = \{1,\ldots,m+n\} \setminus \mathcal{B}$. We denote the position of the j-th column of $(A\,|\,-I)$ in $\mathbf{M}$ by $\pi(j)$, if basic. The position in $\mathbf{N}$ of row $(I\,|\,A')_{i\cdot}$, $i \in \bar{\mathcal{B}}$, is $\sigma(i)$.

To summarize, the left columns of Figs. 1 and 2 outline the primal and dual simplex algorithm when using column and row form.

3 The reduced basis matrix

A basis $(\mathcal{C},\mathcal{R})$ of (P) partitions the columns and rows of the constraint matrix A, i.e., after reordering we can assume

$$A = \begin{pmatrix} A_{\mathcal{R},\mathcal{C}} & A_{\mathcal{R},\bar{\mathcal{C}}} \\ A_{\bar{\mathcal{R}},\mathcal{C}} & A_{\bar{\mathcal{R}},\bar{\mathcal{C}}} \end{pmatrix}.$$

In column form, the basis matrix then reads

$$\mathbf{M} = \begin{pmatrix} A_{\mathcal{R},\mathcal{C}} & -I \\ A_{\bar{\mathcal{R}},\mathcal{C}} & 0 \end{pmatrix}.$$

Traditionally, simplex solvers apply a black-box LU factorization to $\mathbf{M}$ followed by triangular solves to perform FTRAN and BTRAN. However, if we exploit that

$$\mathbf{M}u = p \Longleftrightarrow A_{\bar{\mathcal{R}},\mathcal{C}}u_{\mathcal{C}} = p_{\bar{\mathcal{R}}},\; u_{\bar{\mathcal{C}}} = A_{\mathcal{R},\mathcal{C}}u_{\mathcal{C}} - p_{\mathcal{R}},$$

and

$$\mathbf{M}'v = q \Longleftrightarrow v_{\mathcal{R}} = -q_{\bar{\mathcal{C}}},\; A'_{\bar{\mathcal{R}},\mathcal{C}}v_{\bar{\mathcal{R}}} = q_{\mathcal{C}} - A'_{\mathcal{R},\mathcal{C}}v_{\mathcal{R}},$$

we only need to factorize the smaller matrix $A_{\bar{\mathcal{R}},\mathcal{C}}$. In other words,

$$\mathbf{M}^{-1} = \begin{pmatrix} 0 & A_{\bar{\mathcal{R}},\mathcal{C}}^{-1} \\ -I & A_{\mathcal{R},\mathcal{C}}A_{\bar{\mathcal{R}},\mathcal{C}}^{-1} \end{pmatrix}.$$

Then we can perform FTRAN by solving for $u_{\mathcal{C}}$ and subtracting $p_{\mathcal{R}}$ from a matrix-vector product. For BTRAN, we first compute a matrix-vector product, subtract it from $q_{\mathcal{C}}$, and then solve for $v_{\bar{\mathcal{R}}}$.

The idea applies similarly to the row form where the basis matrix would read

$$\mathbf{N} = \begin{pmatrix} 0 & I \\ A_{\bar{\mathcal{R}},\mathcal{C}} & A_{\bar{\mathcal{R}},\bar{\mathcal{C}}} \end{pmatrix}.$$

A factorization of $A_{\bar{\mathcal{R}},\mathcal{C}}$ and some matrix-vector computations suffice.

The right column of Figs. 1 and 2 presents the primal and dual simplex algorithm using the reduced basis matrix. The hinge clearly lies in the update of the LU factorization in line 19, which will be explained in detail in the following section.

Figure 1: Three versions of the primal simplex algorithm starting from a primal feasible basis using column form (left), row form (center), and the reduced basis matrix (right).

begin

factorize $\mathbf{M} = (A|-I)_{\cdot\mathcal{B}}$

$p_{\mathcal{B}} \leftarrow \mathbf{M}^{-1}b$

(void)

$y' \leftarrow (c'|0')_{\mathcal{B}}\mathbf{M}^{-1}$

$r'_{\bar{\mathcal{B}}} \leftarrow (c'|0')_{\bar{\mathcal{B}}} - y'(A|-I)_{\bar{\mathcal{B}}\cdot}$

while $\min_{k\in\bar{\mathcal{B}}} r_k < 0$ **do**

  choose $k \in \bar{\mathcal{B}}$ with $r_k < 0$

  $\hat{p}_{\mathcal{B}} \leftarrow \mathbf{M}^{-1}(A|-I)_{\cdot k}$

  (void)

  if $\hat{p}_{\mathcal{B}} \leqslant 0$ **then**

    stop: LP unbounded

  else

    $l \leftarrow \arg\min_{i\in\mathcal{B}:\hat{p}_i>0} p_i/\hat{p}_i$

    $\hat{y}' \leftarrow e'_{\pi(l)}\mathbf{M}^{-1}$

    $\hat{r}_{\bar{\mathcal{B}}} \leftarrow \hat{y}'(A|-I)_{\cdot\bar{\mathcal{B}}}$

    $\mathcal{B} \leftarrow (\mathcal{B}\cup\{k\})\setminus\{l\}$

    update p, r

    update factor. of $\mathbf{M}$

end

begin

factorize $\mathbf{N} = \begin{pmatrix} I \\ A \end{pmatrix}_{\bar{\mathcal{B}}\cdot}$

$x \leftarrow \mathbf{N}^{-1}\tilde{b}_{\bar{\mathcal{B}}\cdot}$ with $\tilde{b} := \begin{pmatrix} 0 \\ b \end{pmatrix}$

$p_{\mathcal{B}} \leftarrow \begin{pmatrix} I \\ A \end{pmatrix}_{\mathcal{B}\cdot} x$

$r'_{\bar{\mathcal{B}}} \leftarrow (z'|y')_{\bar{\mathcal{B}}} \leftarrow c'\mathbf{N}^{-1}$

(void)

while $\min_{k\in\bar{\mathcal{B}}} r_k < 0$ **do**

  choose $k \in \bar{\mathcal{B}}$ with $r_k < 0$

  $\hat{x} \leftarrow \mathbf{N}^{-1}e_{\sigma(k)}$

  $\hat{p}_{\mathcal{B}} \leftarrow -\begin{pmatrix} I \\ A \end{pmatrix}_{\mathcal{B}\cdot} \hat{x}$

  if $\hat{p}_{\mathcal{B}} \leqslant 0$ **then**

    stop: LP unbounded

  else

    $l \leftarrow \arg\min_{i\in\mathcal{B}:\hat{p}_i>0} \dfrac{p_i - \tilde{b}_i}{\hat{p}_i}$

    $\hat{r}'_{\bar{\mathcal{B}}} \leftarrow \begin{pmatrix} I \\ A \end{pmatrix}_{l\cdot} \mathbf{N}^{-1}$

    (void)

    $\mathcal{B} \leftarrow (\mathcal{B}\cup\{k\})\setminus\{l\}$

    update p, r

    update factor. of $\mathbf{N}$

end

begin

factorize $\mathbf{B} = A_{\bar{\mathcal{R}},\mathcal{C}}$

$x_{\mathcal{C}} \leftarrow \mathbf{B}^{-1}b_{\bar{\mathcal{R}}}$

$s_{\mathcal{R}} \leftarrow A_{\mathcal{R},\mathcal{C}}x_{\mathcal{C}}$

$y'_{\bar{\mathcal{R}}} \leftarrow c'_{\mathcal{C}}\mathbf{B}^{-1}$

$r'_{\bar{\mathcal{C}}} \leftarrow c'_{\bar{\mathcal{C}}} - y'_{\bar{\mathcal{R}}}A_{\bar{\mathcal{R}},\bar{\mathcal{C}}}$

while $\min_{i\in\bar{\mathcal{R}}} y_i < 0$ *or* $\min_{j\in\bar{\mathcal{C}}} r_j < 0$ **do**

  choose k from $\{i \in \bar{\mathcal{R}} : y_i < 0\}$ or $\{j \in \bar{\mathcal{C}} : r_j < 0\}$

  $\hat{x}_{\mathcal{C}} \leftarrow$ (**if** k *is column index* **then** $\mathbf{B}^{-1}A_{\bar{\mathcal{R}},k}$ **else** $\mathbf{B}^{-1}e_{\sigma(k)}$)

  $\hat{s}_{\mathcal{R}} \leftarrow A_{\mathcal{R},\mathcal{C}}\hat{x}_{\mathcal{C}}$

  if $\hat{x}_{\mathcal{C}} \leqslant 0$ *and* $\hat{s}_{\mathcal{R}} \leqslant 0$ **then**

    stop: LP unbounded

  else

    $l \leftarrow \arg\min\left\{\dfrac{x_j}{\hat{x}_j} : j \in \mathcal{C}, \hat{x}_j > 0\right\} \cup \left\{\dfrac{s_i - b_i}{\hat{s}_i} : i \in \mathcal{R}, \hat{s}_i > 0\right\}$

    $\hat{y}'_{\bar{\mathcal{R}}} \leftarrow$ (**if** l *is column index* **then** $e'_{\pi(l)}\mathbf{B}^{-1}$ **else** $A_{l,\mathcal{C}}\mathbf{B}^{-1}$)

    $\hat{r}'_{\bar{\mathcal{C}}} \leftarrow \hat{y}'_{\bar{\mathcal{R}}}A_{\bar{\mathcal{R}},\bar{\mathcal{C}}} -$ (**if** l *is row index* **then** $A_{l,\bar{\mathcal{C}}}$ **else** $0'$)

    if k *is column index* **then** $\mathcal{C} \leftarrow \mathcal{C}\cup\{k\}$ **else** $\mathcal{R} \leftarrow \mathcal{R}\cup\{k\}$

    if l *is column index* **then** $\mathcal{C} \leftarrow \mathcal{C}\setminus\{l\}$ **else** $\mathcal{R} \leftarrow \mathcal{R}\setminus\{l\}$

    update $x_{\mathcal{C}}$, $s_{\mathcal{R}}$, $y_{\bar{\mathcal{R}}}$, $r_{\bar{\mathcal{C}}}$

    update factorization of $\mathbf{B}$

end

Figure 2: Three versions of the dual simplex algorithm starting from a dual feasible basis using column form (left), row form (center), and the reduced basis matrix (right).

begin

factorize $\mathbf{M} = (A|-I)_{\cdot\mathcal{B}}$

$y' \leftarrow (c'|0')_{\mathcal{B}}\mathbf{M}^{-1}$

$r'_{\bar{\mathcal{B}}} \leftarrow (c'|0')_{\bar{\mathcal{B}}} - y'(A|-I)_{\bar{\mathcal{B}}\cdot}$

$p_{\mathcal{B}} \leftarrow \mathbf{M}^{-1}b$

(void)

while $\min_{l\in\mathcal{B}} p_l < 0$ **do**

  choose $l \in \mathcal{B}$ with $p_l < 0$

  $\hat{y}' \leftarrow e'_{\pi(l)}\mathbf{M}^{-1}$

  $\hat{r}_{\bar{\mathcal{B}}} \leftarrow -\hat{y}'(A|-I)_{\cdot\bar{\mathcal{B}}}$

  if $\hat{r}_{\bar{\mathcal{B}}} \leqslant 0$ **then**

    stop: LP infeasible

  else

    $k \leftarrow \arg\min_{i\in\bar{\mathcal{B}}:\hat{r}_i>0} r_i/\hat{r}_i$

    $\hat{p}_{\mathcal{B}} \leftarrow \mathbf{M}^{-1}(A|-I)_{\cdot k}$

    (void)

    $\mathcal{B} \leftarrow (\mathcal{B} \cup \{k\}) \setminus \{l\}$

    update p, r

    update factor. of $\mathbf{M}$

end

begin

factorize $\mathbf{N} = \begin{pmatrix} I \\ A \end{pmatrix}_{\bar{\mathcal{B}}\cdot}$

$r'_{\bar{\mathcal{B}}} \leftarrow (z'|y')_{\bar{\mathcal{B}}} \leftarrow c'\mathbf{N}^{-1}$

(void)

$x \leftarrow \mathbf{N}^{-1}\tilde{b}_{\bar{\mathcal{B}}\cdot}$ with $\tilde{b} := \begin{pmatrix} 0 \\ b \end{pmatrix}$

$p_{\mathcal{B}} \leftarrow \begin{pmatrix} I \\ A \end{pmatrix}_{\mathcal{B}\cdot} x$

while $\min_{l\in\mathcal{B}} p_l < \tilde{b}_l$ **do**

  choose $l \in \mathcal{B}$ with $p_l < \tilde{b}_l$

  $\hat{r}'_{\bar{\mathcal{B}}} \leftarrow \begin{pmatrix} I \\ A \end{pmatrix}_{l\cdot} \mathbf{N}^{-1}$

  (void)

  if $\hat{r}_{\bar{\mathcal{B}}} \leqslant 0$ **then**

    stop: LP infeasible

  else

    $k \leftarrow \arg\min_{i\in\bar{\mathcal{B}}:\hat{r}_i>0} r_i/\hat{r}_i$

    $\hat{x} \leftarrow \mathbf{N}^{-1}e_{\sigma(k)}$

    $\hat{p}_{\mathcal{B}} \leftarrow \begin{pmatrix} I \\ A \end{pmatrix}_{\mathcal{B}\cdot} \hat{x}$

    $\mathcal{B} \leftarrow (\mathcal{B} \cup \{k\}) \setminus \{l\}$

    update p, r

    update factor. of $\mathbf{N}$

end

begin

factorize $\mathbf{B} = A_{\bar{\mathcal{R}},\mathcal{C}}$

$y'_{\bar{\mathcal{R}}} \leftarrow c'_{\mathcal{C}}\mathbf{B}^{-1}$

$r'_{\bar{\mathcal{C}}} \leftarrow c'_{\bar{\mathcal{C}}} - y'_{\bar{\mathcal{R}}}A_{\bar{\mathcal{R}},\bar{\mathcal{C}}}$

$x_{\mathcal{C}} \leftarrow \mathbf{B}^{-1}b_{\bar{\mathcal{R}}}$

$s_{\mathcal{R}} \leftarrow A_{\mathcal{R},\mathcal{C}}x_{\mathcal{C}}$

while $\min_{j\in\mathcal{C}} x_j < 0$ *or* $\min_{i\in\mathcal{R}}(s_i - b_i) < 0$ **do**

  choose l from $\{j \in \mathcal{C} : x_j < 0\}$ or $\{i \in \mathcal{R} : s_i < b_i\}$

  $\hat{y}'_{\bar{\mathcal{R}}} \leftarrow$ (**if** l *is column index* **then** $e'_{\pi(l)}\mathbf{B}^{-1}$ **else** $A_{l,\mathcal{C}}\mathbf{B}^{-1}$)

  $\hat{r}'_{\bar{\mathcal{C}}} \leftarrow -\hat{y}'_{\bar{\mathcal{R}}}A_{\bar{\mathcal{R}},\bar{\mathcal{C}}} +$ (**if** l *is row index* **then** $A_{l,\bar{\mathcal{C}}}$ **else** $0'$)

  if $\hat{y}_{\bar{\mathcal{R}}} \leqslant 0$ *and* $\hat{r}_{\bar{\mathcal{C}}} \leqslant 0$ **then**

    stop: LP infeasible

  else

    $k \leftarrow \arg\min \left\{ \frac{y_i}{\hat{y}_i} : i \in \bar{\mathcal{R}}, \hat{y}_i > 0 \right\} \cup \left\{ \frac{r_j}{\hat{r}_j} : j \in \bar{\mathcal{C}}, \hat{r}_j > 0 \right\}$

    $\hat{x}_{\mathcal{C}} \leftarrow$ (**if** k *is column index* **then** $\mathbf{B}^{-1}A_{\bar{\mathcal{R}},k}$ **else** $\mathbf{B}^{-1}e_{\sigma(k)}$)

    $\hat{s}_{\mathcal{R}} \leftarrow A_{\mathcal{R},\mathcal{C}}\hat{x}_{\mathcal{C}}$

    if l *is column index* **then** $\mathcal{C} \leftarrow \mathcal{C} \setminus \{l\}$ **else** $\mathcal{R} \leftarrow \mathcal{R} \setminus \{l\}$

    if k *is column index* **then** $\mathcal{C} \leftarrow \mathcal{C} \cup \{k\}$ **else** $\mathcal{R} \leftarrow \mathcal{R} \cup \{k\}$

    update $y_{\bar{\mathcal{R}}}$, $r_{\bar{\mathcal{C}}}$, $x_{\mathcal{C}}$, $s_{\mathcal{R}}$

    update factorization of $\mathbf{B}$

end

4 Updating an LU factorization of $A_{\bar{\mathcal{R}},\mathcal{C}}$

Solving FTRAN and BTRAN accounts for a major part of the computational effort in each simplex iteration, see, e.g., Hall and McKinnon [6]. Competitive implementations employ a factorized representation of the basis matrix or its inverse. Since the basis matrix is only slightly modified from iteration to iteration—traditionally exchanging a column of $\mathbf{M}$ or a row of $\mathbf{N}$—it is more efficient to update the factorization instead of recomputing it. This section shows how a factorization of the reduced basis matrix $A_{\bar{\mathcal{R}},\mathcal{C}}$ can be updated also in the case when its dimension changes.

4.1 Product form update

The first update procedure suggested by Dantzig and Orchard-Hays [4] was based on the product form of the inverse. Suppose for a regular basis matrix $\mathbf{B} = A_{\bar{\mathcal{R}},\mathcal{C}} \in \mathbb{R}^{p\times p}$, $p = |\mathcal{C}| = |\bar{\mathcal{R}}|$, we have computed a factorized representation of its inverse $\mathbf{B}^{-1}$. Now there are four types of basis changes that may occur:

Case 1. If both a column enters and leaves the basis, i.e., $\mathcal{C} \leftarrow (\mathcal{C} \cup \{k\}) \setminus \{l\}$, then the dimension of $\mathbf{B}$ is invariant and the new basis matrix reads

$$\mathbf{B}^{(1)} = \mathbf{B} + (A_{\bar{\mathcal{R}},k} - \mathbf{B}e_{\pi(l)})e'_{\pi(l)} = \mathbf{B}\underbrace{(I + (\eta - e_{\pi(l)})e'_{\pi(l)})}_{=:E} \tag{5}$$

with $\eta = \mathbf{B}^{-1}A_{\bar{\mathcal{R}},k}$, which is available as product of the last BTRAN operation. Matrices like E, i.e., identity matrices with one column or row replaced by a vector, are called (column respectively row) eta matrices. Their inverse is

$$E^{-1} = I - (\eta - e_{\pi(l)})e'_{\pi(l)}/\eta_{\pi(l)},$$

which yields an updated representation of the basis inverse

$$\mathbf{B}^{(1)^{-1}} = E^{-1}\mathbf{B}^{-1},$$

as described by Dantzig and Orchard-Hays [4].

Case 2. Similarly, if one row is replaced by another, i.e., $\mathcal{R} \leftarrow (\mathcal{R} \cup \{k\}) \setminus \{l\}$, we can write $\mathbf{B}^{(1)} = F\mathbf{B}$ with

$$F = I + e_{\sigma(k)}(A_{l,\mathcal{C}}\mathbf{B}^{-1} - e'_{\sigma(k)})$$

Then the basis inverse is updated by multiplication from the right, $\mathbf{B}^{(1)^{-1}} = \mathbf{B}^{-1}F^{-1}$.

Case 3. If a column enters the basis and a row leaves the basis, i.e., $\mathcal{C} \leftarrow \mathcal{C} \cup \{k\}$, $\mathcal{R} \leftarrow \mathcal{R} \setminus \{l\}$, the dimension of $\mathbf{B}$ increases by one. Suppose that column k and row l are added to the right and bottom, respectively, then

$$\mathbf{B}^{(1)} = \begin{pmatrix} \mathbf{B} & A_{\bar{\mathcal{R}},k} \\ A_{l,\mathcal{C}} & A_{lk} \end{pmatrix} = \underbrace{\begin{pmatrix} I & 0 \\ A_{l,\mathcal{C}}\mathbf{B}^{-1} & 1 \end{pmatrix}}_{=:F} \begin{pmatrix} \mathbf{B} & 0 \\ 0' & \kappa \end{pmatrix} \underbrace{\begin{pmatrix} I & \mathbf{B}^{-1}A_{\bar{\mathcal{R}},k} \\ 0 & 1 \end{pmatrix}}_{=:E}$$

with $\kappa := A_{lk} - A_{l,\mathcal{C}}\mathbf{B}^{-1}A_{\bar{\mathcal{R}},k}$. Thus, the new basis inverse can be written as

$$\mathbf{B}^{(1)^{-1}} = E^{-1}\begin{pmatrix} \mathbf{B}^{-1} & 0 \\ 0' & 1/\kappa \end{pmatrix} F^{-1}.$$

Case 4. Finally, if a column leaves the basis and a row enters the basis, i.e., $\mathcal{C} \leftarrow \mathcal{C} \setminus \{l\}$, $\mathcal{R} \leftarrow \mathcal{R} \cup \{k\}$, the dimension of $\mathbf{B}$ decreases by one. Let $E = I + (\mathbf{B}^{-1}e_{\sigma(k)} - e_{\pi(l)})e'_{\pi(l)}$, $F = I + e_{\sigma(k)}(e'_{\pi(l)}\mathbf{B}^{-1} - e'_{\sigma(k)})$, and denote by $D_j := (e_1 \,|\, \ldots \,|\, e_{j-1} \,|\, e_{j+1} \,|\, \ldots \,|\, e_p) \in \mathbb{R}^{p\times(p-1)}$ the identity matrix with its j-th column removed. Then the new basis inverse is given by

Lemma 2. $\mathbf{B}^{(1)^{-1}} = D'_{\pi(l)}E^{-1}\mathbf{B}^{-1}F^{-1}D_{\sigma(k)}$.

Proof. $\mathbf{B}^{(1)} = A_{\bar{\mathcal{R}}\setminus\{k\},\mathcal{C}\setminus\{l\}} = D'_{\sigma(k)}A_{\bar{\mathcal{R}},\mathcal{C}}D_{\pi(l)} = D'_{\sigma(k)}\mathbf{B}D_{\pi(l)}$ yields

$$\begin{aligned}\mathbf{B}^{(1)}D'_{\pi(l)}E^{-1}\mathbf{B}^{-1}F^{-1}D_{\sigma(k)} &= D'_{\sigma(k)}\mathbf{B}\underbrace{D_{\pi(l)}D'_{\pi(l)}E^{-1}}_{I-\mathbf{B}^{-1}e_{\sigma(k)}e'_{\pi(l)}/(\mathbf{B}^{-1}e_{\sigma(k)})_{\pi(l)}}\mathbf{B}^{-1}F^{-1}D_{\sigma(k)}\\ &= D'_{\sigma(k)}\mathbf{B}I\mathbf{B}^{-1}F^{-1}D_{\sigma(k)} - \underbrace{D'_{\sigma(k)}\mathbf{B}\mathbf{B}^{-1}e_{\sigma(k)}}_{=0}(\cdots) = I \in \mathbb{R}^{(p-1)\times(p-1)},\end{aligned}$$

and analogously $D'_{\pi(l)}E^{-1}\mathbf{B}^{-1}F^{-1}D_{\sigma(k)}\mathbf{B}^{(1)} = I$. □

4.2 LU update

Current state-of-the-art implementations, see, e.g., [11, 7] are based on an LU factorization of $\mathbf{B}$,

$$\mathbf{B} = \tilde{L}\tilde{U},$$

where $\tilde{L} = P'LP$ and $\tilde{U} = P'UP$, are permuted lower and upper triangular matrices, respectively, $P \in \mathbb{R}^{p\times p}$ a permutation matrix. (Because the columns in the basis matrix can be reordered, we may assume a symmetric permutation.) Pioneered by Bartels and Golub [1], the most successful update formulas used in practice are by Forrest and Tomlin [5] with an improvement by Suhl and Suhl [10]. We now provide Forrest-Tomlin like update formulas for $A_{\bar{\mathcal{R}},\mathcal{C}}$.

Case 1. Using eq. 5, we obtain

$$\begin{aligned}V &:= L^{-1}P\mathbf{B}^{(1)}P' = L^{-1}P\mathbf{B}(I + (\mathbf{B}^{-1}A_{\bar{\mathcal{R}},k} - e_{\pi(l)})e'_{\pi(l)})P'\\ &= U + (\underbrace{L^{-1}PA_{\bar{\mathcal{R}},k}}_{=:\alpha} - UPe_{\pi(l)})(Pe_{\pi(l)})' = \begin{pmatrix} u_{11} & \ldots & \alpha_1 & \ldots & u_{1p}\\ & \ddots & \vdots & & \vdots\\ & & \alpha_t & \cdots & u_{tp}\\ & & \vdots & \ddots & \vdots\\ & & \alpha_p & & u_{pp}\end{pmatrix},\end{aligned}$$

Suppose $Pe_{\pi(l)} = e_t$, then V is the original factor U which has its t-th column replaced by the so-called *spike* α. To restore triangularity, we shift the t-th column and row to the last position by multiplying with the permutation matrix $\Pi := (e_1 \,|\, \ldots \,|\, e_{t-1} \,|\, e_p \,|\, e_{t+1} \,|\, \ldots \,|\, e_{p-1})$. Subsequently, we eliminate the resulting nonzeros in the last row: $U^{(1)} := \Lambda\Pi V\Pi'$ is upper triangular, where the nonzeros of $\Lambda = I + e_p\lambda'$ are computed by solving

$$\begin{pmatrix} u_{t+1,t+1} & & \\ \vdots & \ddots & \\ u_{t+1,p} & \cdots & u_{p,p}\end{pmatrix}\begin{pmatrix}\lambda_t\\ \vdots\\ \lambda_{p-1}\end{pmatrix} = -\begin{pmatrix}u_{t,t+1}\\ \vdots\\ u_{t,p}\end{pmatrix}.$$

Using the definition of V and $U^{(1)}$ and $P^{(1)} = \Pi P$ we obtain

$$\begin{aligned}\mathbf{B}^{(1)} &= P'LVP = P'L(\Pi'\Lambda^{-1}U^{(1)}\Pi)P \\ &= \underbrace{P'LP}_{\tilde{L}}\ \underbrace{{P^{(1)}}'\Lambda^{-1}P^{(1)}}_{\tilde{R}^{(1)}}\ \underbrace{{P^{(1)}}'U^{(1)}P^{(1)}}_{\tilde{U}^{(1)}}.\end{aligned}$$

Case 2. For a row exchange, we can write

$$\begin{aligned}V{:=}P\mathbf{B}^{(1)}P'U^{-1} &= P(I + e_{\sigma(k)}(A_{l,\mathcal{C}}\mathbf{B}^{-1} - e'_{\sigma(k)}))\mathbf{B}P'U^{-1} \\ = L + Pe_{\sigma(k)}(\underbrace{A_{l,\mathcal{C}}P'U^{-1}}_{=:\alpha} - (Pe_{\sigma(k)})'L) &= \begin{pmatrix} l_{11} & & & & \\ \vdots & \ddots & & & \\ \alpha_1 & \cdots & \alpha_t & \cdots & \alpha_p \\ \vdots & & \vdots & \ddots & \\ l_{p1} & \cdots & l_{pt} & \cdots & l_{pp} \end{pmatrix},\end{aligned}$$

where $Pe_{\sigma(k)} = e_t$. For suitable Λ as above, $L^{(1)}{:=}\Pi V\Pi'\Lambda'$ is lower triangular. With $P^{(1)} = \Pi P$ we get

$$\begin{aligned}\mathbf{B}^{(1)} &= P'VUP = P'(\Pi'L^{(1)}{\Lambda^{-1}}'\Pi)UP \\ &= \underbrace{{P^{(1)}}'L^{(1)}P^{(1)}}_{\tilde{L}^{(1)}}\ \underbrace{{P^{(1)}}'{\Lambda^{-1}}'P^{(1)}}_{\tilde{R}^{(1)}}\ \underbrace{P'UP}_{\tilde{U}}.\end{aligned} \tag{6}$$

Case 3. If a column enters and a row leaves the basis, i.e., the dimension of the basis matrix increases by one, it is straightforward to maintain triangular factors by extending $\tilde{L}$ to the bottom and $\tilde{U}$ to the right:

$$\mathbf{B}^{(1)} = \begin{pmatrix} \mathbf{B} & A_{\bar{\mathcal{R}},k} \\ A_{l,\mathcal{C}} & A_{lk} \end{pmatrix} \tag{7}$$

$$= \underbrace{{P^{(1)}}'\begin{pmatrix} L & 0 \\ A_{l,\mathcal{C}}P'U^{-1} & 1 \end{pmatrix}P^{(1)}}_{=:\tilde{L}^{(1)}}\ \underbrace{{P^{(1)}}'\begin{pmatrix} U & L^{-1}PA_{\bar{\mathcal{R}},k} \\ 0 & \kappa \end{pmatrix}P^{(1)}}_{=:\tilde{U}^{(1)}}. \tag{8}$$

Case 4. If column l leaves and row k enters the basis, then the new basis matrix is obtained by deleting column $\pi(l)$ and row $\sigma(k)$ of $\mathbf{B}$. First, however, imagine that we replace these columns by unit vectors, forming $\mathbf{B}^{(0)}{:=}(I + e_{\sigma(k)}(e'_{\pi(l)}\mathbf{B}^{-1} - e'_{\sigma(k)}))\mathbf{B}(I + (\mathbf{B}^{-1}e_{\sigma(k)} - e_{\pi(l)})e'_{\pi(l)})$. As described for cases 1 and 2 above, we can update the factorization to obtain

$$\mathbf{B}^{(0)} = \underbrace{{P^L}'L^{(0)}P^L}_{\tilde{L}^{(0)}}\ \underbrace{{P^L}'{\Lambda^{L^{-1}}}'P^L}_{\tilde{R}^L}\ \underbrace{{P^U}'\Lambda^{U^{-1}}P^U}_{\tilde{R}^U}\ \underbrace{{P^U}'U^{(0)}P^U}_{\tilde{U}^{(0)}}, \tag{9}$$

where the $\pi(l)$-th column and $\sigma(k)$-th row of $\mathbf{B}^{(0)}$ are permuted to the last position p in $U^{(0)}$ and $L^{(0)}$, respectively. Hence, deleting the last row and column in $L^{(0)}$ and $U^{(0)}$, deleting the last (unit) columns of $\Lambda^{L^{-1}}$ and $\Lambda^{U^{-1}}$ and adjusting the permutation matrices gives a factorized representation of $\mathbf{B}^{(1)} = D'_{\sigma(k)}\mathbf{B}D_{\pi(l)}$.

The improvements of Suhl and Suhl [10] can be incorporated easily and have been left out merely for clarity of presentation.

5 Concluding remarks

We conclude by discussing the computational advantages of using the reduced basis matrix. Note that for clarity, Figs. 1 and 2 present the FTRAN and BTRAN operations using the reduced basis matrix partitioned by $\mathcal{C}$ and $\mathcal{R}$. However, this can easily be hidden behind a black-box implementation of the linear algebra routines and thus seamlessly integrated into existing state-of-the-art simplex codes based on row or column form, yielding all benefits outlined in the following.

Faster factorization and updates. Since $A_{\bar{\mathcal{R}},\mathcal{C}}$ is potentially much smaller than $\mathbf{M}$ and $\mathbf{N}$, both the initial factorization and subsequent updates may be expected to involve less nonzeros and perform faster.

Faster solves. Using an LU factorization of $\mathbf{M}$, we need to perform two triangular solves of dimension m for FTRAN or BTRAN. The new technique only requires two triangular solves of dimension $|\mathcal{C}| \leqslant m$. The remaining $m - |\mathcal{C}|$ solution values can be computed in one sweep. The same effect holds when using the row form.

Smaller memory consumption. First, since $A_{\mathcal{R},\mathcal{C}}$ is no longer part of the basis matrix, it can be read directly from the constraint matrix and is not stored a second time in U. Second, since we factorize and update a smaller matrix, we may expect the fill-in to grow at a smaller rate, an effect that has been observed by Powell [9] and confirmed by Wunderling [12].

Solving from scratch. The impact of the advantages listed above will be more prominent the smaller $A_{\bar{\mathcal{R}},\mathcal{C}}$ is on average when compared to $\mathbf{M}$ or $\mathbf{N}$, i.e., when many of the vertex solutions traversed by the simplex algorithm are predominantly determined by tight column bounds. Note that, even if the optimal basis might have many basic columns, the simplex algorithm is often started from the slack basis $(\emptyset, \{1, \ldots, m\})$. In this case, $A_{\bar{\mathcal{R}},\mathcal{C}}$ necessarily has small dimension during the first iterations.

Furthermore, it is a common situation that, for example, problems from combinatorial optimization or relaxations solved within an LP-based branch-and-cut algorithm exhibit more rows than columns. In this case, the number of basic (not tight) rows $|\mathcal{R}|$ is at least $m - n$, i.e., the dimension of $A_{\bar{\mathcal{R}},\mathcal{C}}$ is guaranteed to be at most $n = \dim(\mathbf{N}) < \dim(\mathbf{M})$.

Reoptimizing within branch-cut-and-price. Last, but not least, when used in an LP-based branch-cut-and-price algorithm, the reduced basis matrix gives an advantage independent of the proportions of the constraint matrix. When using column form the factorization must be computed from scratch after adding a row; in row form after adding a column. Since reoptimization often only takes few iterations, having to recompute the factorization of the basis matrix each time can make up a significant amount of the computational effort performed.

In contrast, adding new columns or rows to the problem does not affect $A_{\bar{\mathcal{R}},\mathcal{C}}$ since a new column is initialized as nonbasic, i.e., added to $\bar{\mathcal{C}}$, and a new row is initialized as basic, i.e., added to $\mathcal{R}$. Hence, the factorization is not invalidated and can be reused.

Acknowledgements

Many thanks to Daniel Steffy, Timo Berthold, Thorsten Koch, and Matthias Miltenberger for their valuable comments on this paper.

References

[1] Richard H. Bartels and Gene H. Golub. The simplex method of linear pogramming using LU decomposition. *Communications of the ACM*, 12:266–268, May 1969. doi:10.1145/362946.362974.

[2] George B. Dantzig. Notes on linear programming – Part III: Computational algorithm of the revised simplex method. Research Memorandum RM-1266, RAND, October 1953. http://www.rand.org/pubs/research_memoranda/RM1266.

[3] George B. Dantzig. *Linear programming and extensions.* Princeton Univ. Press, Princeton, NJ, 1963.

[4] George B. Dantzig and William Orchard-Hays. Notes on linear programming – Part V: Alternate algorithm for the revised simplex method using a product form for the inverse. Research Memorandum RM-1268, RAND, November 1953. http://www.rand.org/pubs/research_memoranda/RM1268.

[5] John J.H. Forrest and John A. Tomlin. Updated triangular factors of the basis to maintain sparsity in the product form simplex method. *Mathematical Programming*, 2:263–278, 1972. doi:10.1007/BF01584548.

[6] Julian A. Hall and Ken I.M. McKinnon. Hyper-sparsity in the revised simplex method and how to exploit it. *Computational Optimization and Applications*, 32:259–283, 2005. doi:10.1007/s10589-005-4802-0.

[7] Achim Koberstein. *The Dual Simplex Method. Techniques for a fast and stable implementation.* PhD thesis, Universität Paderborn, 2005. urn:nbn:de:hbz:466-20050101272.

[8] John L. Nazareth. *Computer solutions of linear programs.* Monographs on Numerical Analysis. Oxford Univ. Press, New York, Oxford, 1987.

[9] Susan Powell. A development of the product form algorithm for the simplex method using reduced transformation vectors. In Michel L. Balinski and Eli Hellerman, editors, *Computational Practice in Mathematical Programming*, volume 4 of *Mathematical Programming Studies*, pages 93–107. North-Holland, Amsterdam, 1975. doi:10.1007/BFb0120713.

[10] Leena M. Suhl and Uwe H. Suhl. A fast LU update for linear programming. *Annals of Operations Research*, 43:33–47, 1993. doi:10.1007/BF02025534.

[11] Roland Wunderling. *Paralleler und objektorientierter Simplex-Algorithmus.* PhD thesis, Technische Universität Berlin, 1996. http://www.zib.de/Publications/abstracts/TR-96-09.

[12] Roland Wunderling. The kernel simplex method. Talk at the 21st International Symposium on Mathematical Programming, Berlin, Germany, August 2012.

[13] Guus Zoutendijk. A product-form algorithm using contracted transformation vectors. In Jean Abadie, editor, *Integer and Nonlinear Programming, 511-523 (1970).* North-Holland, Amsterdam, 1970.