Noname manuscript No.
(will be inserted by the editor)

Rigorous Outer Bounds with the Dual Simplex Algorithm in Tableau Form

Presented at SCAN 2010, Ecole Normale Supérieure Lyon.

Received: date / Accepted: date

Abstract In this paper, we describe how to compute rigorous outer bounds for linear programming problems occuring in linear relaxations derived from the Bernstein polynomials. The computation uses interval arithmetic for the Gauss-Jordan pivoting steps on the tableau. The resulting errors are stored as interval right hand sides. Additionally, we show how to generate a start basis for the linear programs of this type. We give details of the implementation and comment on numerical experiments.

Keywords verified simplex algorithm · interval arithmetic · tableau form

PACS 90C05 · 65G30 · 65N15

1 Introduction

Linear relaxation [6] is a common method to solve non-linear systems over variable intervals $D_i \subset \mathbb{R}$, $i = 1, \ldots, N$. For the system with variables $x_1 \in [0, 1]$ and $x_2 \in [0, 1]$,

$$\begin{aligned} x_1^2 - x_2 &= 0 \\ x_2 - x_1 &\leq 0 \end{aligned}$$

e.g., a linear relaxation derived from the tangent plane in $x_1 = x_2 = 0.5$, for example,

$$\begin{aligned} &(x_1 - 0.5) - (x_2 - 0.5) - 0.25 \geq 0 \\ &x_2 - x_1 \leq 0 \end{aligned}$$

In [3], linear relaxations, derived from the Bernstein polynomials, were used for the monomials x_i^2 and $x_i x_j$, $i < j$ with $x_i \in D_i$. The curves (x_i, x_i^2), and the surfaces $(x_i, x_j, x_i x_j)$, $i < j$ are enclosed in a polytope, called *Bernstein polytope* (Figure 2).

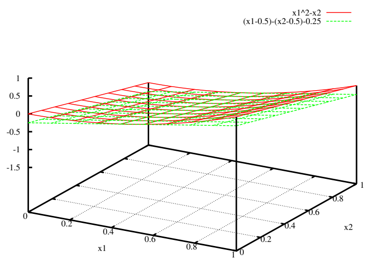
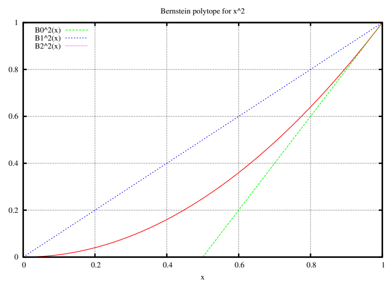


Fig. 1 Linear relaxation (in green) for $x_1^2 - x_2$ (in red).



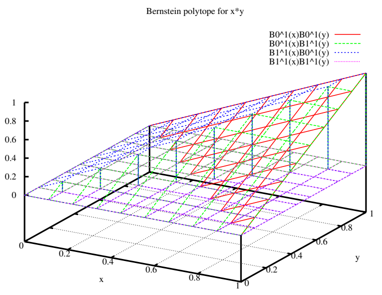


Fig. 2 Bernstein polytope enclosing the curve (x, x^2) (left) and the surface (x, y, xy) (right) on $x, y \in [0, 1]$.

With linear relaxation, the quadratic system

$$F(x) = 0,\ G(x) \geq 0,\ x = (x_1, \ldots, x_N)$$

gives a linear system in the variables x_i, x_{ii}, and x_{ij}, $i < j$. For example, a lower bound on the variable domain D_i can be obtained by solving a linear program: minimize x_i on the linear system and the Bernstein polytope as constraints.

In [3], the revised simplex code *SoPlex* [7] in floating point arithmetic and a backward analysis of the final linear system for the objective value was used. A comparison of linear programming codes using rational arithmetic and floating point arithmetic can be found in [2]. Of course, the use of floating point arithmetic, which is used at least in parts of the code, is significantly faster than exact rational arithmetic. In [1], a method to compute a lower bound is described using an arbitrary linear program solver, which is based on the *weak optimality theorem*: Any feasible point y of the dual problem $A^t y \leq c$ $(\max b^t y)$ gives a lower bound $b^t y$ for the minimum of the primal problem $Ax = b$, $x \geq 0$ $(\min c^t x)$. As outlined in [1], the lower bounds obtained by a verified computation of a feasible point can be much off from the optimum value for ill-conditioned problems. Additionally, the computation of upper bounds is considered.

In this article, we show how to use interval arithmetic in a tableau-form implementation of the dual simplex algorithm (Section 2) to verify computations and to generate a tight lower bound on the minimum value. The tableau has floating point entries and uses an interval right-hand side. In a pivoting operation of the Gauss-Jordan algorithm, rounding errors are collected and stored in the right-hand side intervals (Section 2.2). For the application of linear relaxations using the Bernstein polytope, we give two ways to generate a start basis for the occurring linear programs in Section 2.3. Finally, we conclude on this work in Section 4.

1.1 Notation

Concerning notation, we use $\underline{R}$ for the lower bound of an interval R and $\overline{R}$ for the upper bound. For a real number a, we denote by a^- the largest floating point number smaller or equal to a, and by a^+ the smallest floating-point number larger or equal to a. We denote by e_k the kth vector of the canonical basis with $e_{k,k} := 1$, and 0 otherwise. As usual, an inequality between vectors, like $x \geq 0$, applies to all components i: $x_i \geq 0$.

2 Linear Programming Problem

A linear program in standard form is defined by

$$\begin{aligned}&\min c^t x\\ &Ax = b\\ &x \geq 0\end{aligned}$$

where A is a $m \times n$ real matrix, b is a m-component real vector, and c is a n-component real vector. The system $Ax = b$ contains the linear equality constraints, and the function $c^t x$ defines the linear objective function to be minimized. An inequality $a_1^t x \leq b_1$ is transformed into an equality by a new variable $x_s \geq 0$: $a_1^t x + x_s = b_1$, which is called a *slack variable* [4]. Note that in our case it is $m \leq n$, i.e., our problem has at least as many variables as rows due to the slack variables. Note also that to ensure non-negativity $x \geq 0$, we use symbolic substitutions $x_i \to (x_i - \underline{D_i})$ if $\underline{D_i} < 0$.

The tableau-form implementation of the simplex algorithm (with column basis) selects a maximal subset of m linear independent columns of A (corresponding to components of x), where m is the rank of the matrix A. The subset with index set B is called a *basis*, and the corresponding submatrix $A_{*,B}$ is invertible; the rest is denoted by $A_{*,NB}$. Non-basis variables always have a zero value. A basis update operation maintains the reduced row-echelon form of the tableau

$$\begin{pmatrix} c_B^t A_{*,B}^{-1} b = c_B^t x_B & (c_B - c_B^t A_{*,B}^{-1} A_{*,B})^t = 0 & (c_{NB} - c_B^t A_{*,B}^{-1} A_{*,NB})^t \\ A_{*,B}^{-1} b = x_B & A_{*,B}^{-1} A_{*,B} = I & A_{*,B}^{-1} A_{*,NB} \end{pmatrix}$$

which allows to look up the relative costs in the first row, the objective function value in the first column of the first row, and the basis variable values x_B in the first column below the first row of this tableau [4]. Furthermore, we group matrix rows into equalities (without a slack variable) given by the index set E and inequalities (each having a slack variable) given by the index set NE. Maintaining this form is possible with the Gauss-Jordan algorithm from numerical linear algebra. As tableau rows define basis variables, it is possible to check the non-negativity constraints $x \geq 0$ and to select a leaving variable in the simplex algorithm (*pricing* [4]). The leaving variable is replaced by an entering variable, which can be selected from the reduced costs in the first tableau row (*ratio test* [4]). For applications, this *dual form of the simplex algorithm* is beneficial [5], which changes an infeasible basis with a lower bound value into an optimum, feasible basis. It selects the leaving variable from the infeasible basis first and replaces it by the entering variable. In contrast the *primal form of the simplex algorithm*, selects the entering variable first, then selects the leaving variable until an optimum objective value is reached.

2.1 Pricing Rule and Ratio Test

Important for the performance of the solver is the pricing rule, the ratio test, and the start basis [5]. For the pricing rule, we consider

Definition 1 (Goldfarb-Forrest pricing rule) *Select row r which has the most negative ratio $\frac{x_r}{|e_r^t A_{*B}^{-1}|_2}$.*

For the ratio test, we use

Definition 2 (Harris ratio test) *Select column s so that $a_{r,s}$ is minimum with $\frac{c_j}{a_{r,j}} \geq \theta_r(\epsilon)$, $a_{r,j} < 0$, $\theta_r(\epsilon) := \min\{\frac{c_j+\epsilon}{a_{r,j}} \ : \ a_{r,j} < 0\}$.*

This rule chooses the element $\frac{c_s}{a_{r,s}}$ of the set $\{\frac{c_j}{a_{r,j}} \geq \theta_r(\epsilon) \ : \ a_{r,j} < 0\}$ defined by a small parameter $\epsilon > 0$, so that the denominator $a_{r,s} < 0$ has largest magnitude in order to have only small shifts of the number's significant in the evaluation of the division.

2.2 Pivoting Steps using Interval Arithmetic

For error collection during a pivoting operation of the Gauss-Jordan algorithm, we use interval arithmetic. Let $a_{r,s}$ be the pivot element in row r, and D_i the variable domain for variable x_i. Then the linear equation

$$\sum_j \frac{a_{r,j}}{a_{r,s}} x_j = \frac{b_r}{a_{r,s}}$$

transforms into an interval equation

$$\sum_j R_{r,j} x_j = R_r$$

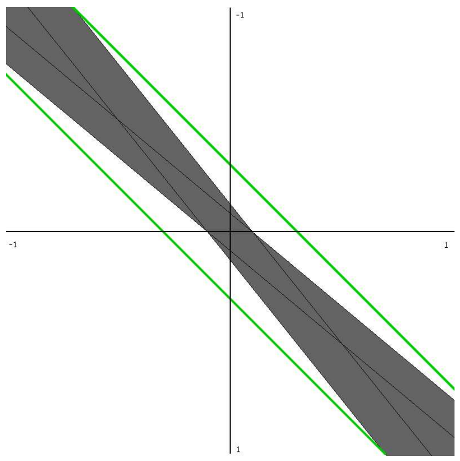


Fig. 3 Hyperplane arrangement (grey) for the interval equation $[0.8, 1.2]x + [1.0, 1.0]y + [-0.5, 0.5] = 0$, $x, y \in [-1, 1]$. A thick hyperplane (green) results from the selection of interval representatives $1.0x + 1.0y + [-0.7, 0.7] = 0$.

where we can select a floating-point value $a_{r,j} \in R_{r,j}$ of the interval so that $R_{r,j} \subset a_{r,j} + [(\underline{R_{r,j}} - a_{r,j})^-, (\overline{R_{r,j}} - a_{r,j})^+]$. The intervals can be collected and stored as an interval right hand side R'_r

$$\sum_j a_{r,j}x_j = R_r - \sum_j [(\underline{R_{r,j}} - a_{r,j})^-, (\overline{R_{r,j}} - a_{r,j})^+]D_j =: R'_r$$

With the representative $a_{r,j} := mid(R_{r,j})^-$, the resulting interval $[(\underline{R_{r,j}} - a_{r,j})^-, (\overline{R_{r,j}} - a_{r,j})^+]$ has smallest width. Figure 3 shows the hyperplane arrangement for an example equation.

Similarly, a row operation as required in the Gauss-Jordan algorithm between row r and row i

$$\sum_j (a_{i,j} - a_{r,j}\frac{a_{i,s}}{a_{r,s}})x_j = b_i - b_r\frac{a_{i,s}}{a_{r,s}}$$

can be performed in interval arithmetic

$$\sum_j R_{i,j}x_j = R_i$$

and rewritten using an interval right-hand side

$$\sum_j a_{i,j}x_j = R_i - \sum_j [\underline{R_{i,j}} - a^-_{i,j}, \overline{R_{i,j}} - a^+_{i,j}]D_j =: R'_i$$

In this form, a sufficient condition for the feasibility of x_i, $i \in B$ is $\underline{R_i} \geq 0$. In case $\overline{R_i} < 0$, it is infeasible and a candidate for the pricing rule. Otherwise some R_i have positive and negative values, in which case we stop the solving process with a lower bound of the optimum value.

In the geometric view of the polytope, the thick hyperplanes bound a set of polytopes, which are not necessarily of the same topology. See Figure 4 for an example, where the minimum-y vertex of the outer hyperplanes is defined by the intersection of r_1 and r_3, but the minimum-y vertex of the inner hyperplanes is defined by the intersection of r_1 and r_2.

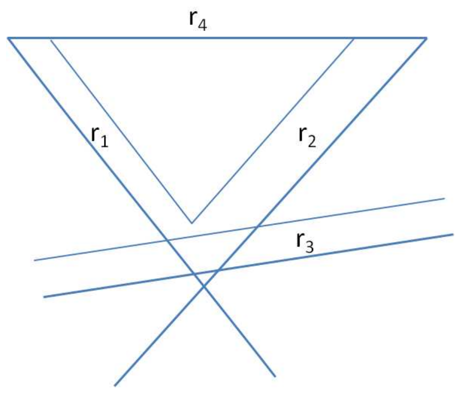


Fig. 4 Polytope bounded by thick hyperplanes r_1, r_2, r_3 and r_4. Note that the topology of the polytope built by the outer hyperplanes (thick) is not the same as the one by the inner hyperplanes (thin).

Note that these topology changes make situations possible, where the outer polytope is non-empty but the inner polytope is empty. In such cases, the algorithm can not decide feasibility of the linear program.

2.3 Start Basis Generation

Inside the non-linear solver, we only have to handle objective functions of the form $x_i = e_i^t x$ for a variable index i. Start basis generation can be done by performing the Gauss-Jordan algorithm on the equation part $A_{E,*}$ of the system. This defines a subset B_E of the basis B. Note that the set B_E depends on the pivot selection strategy in the Gauss-Jordan algorithm.

The first possibility is to select only pivots with column index different from i. In this case, where B_E does not contain variable index i, we can easily complete the basis from the vertex with smallest value x_i of the Bernstein polytope for (x_i, x_i^2).

The second strategy is to select a pivot with column index i. In this case, where B_E contains the variable index i, we can generate a start basis from the equation row k defining variable x_i. Let $x_i + \sum_{j \neq i} a_{k,j} x_j = R_k$ be row k. If there are columns $a_{k,j} > 0$ the current basis part B_E is not optimum, and it can be changed by primal steps into a optimum basis. I.e., for each such column j with $a_{k,j} > 0$ we determine a row r such that $a_{r,0} \geq 0$ and $-a_{r,0}\frac{a_{k,j}}{a_{r,j}} < 0$ is minimum. Both strategies are compared in Section 3 based on a numerical example.

3 Implementation and Numerical Experiments

We have implemented the dual simplex algorithm in C/C++ using the tableau organization shown in Figure 5. Note that the tableau can be physically stored

objective function row	
User equ. rows, n vars	User equ. rows, $u + 3c + 4d$ slack
User inequ. rows, n vars	User inequ. rows, $u + 3c + 4d$ slack
Bernstein rows, n vars	Bernstein rows, $u + 3c + 4d$ slack

Fig. 5 Tableau organisation in regions for the user system and for the Bernstein polytope (u user inequalities, c squared variables, d mixed variables).

as a $m \times n$ array of `double`-entries or in a sparse form as an array of m rows of `index`/`double`-entry pairs.

In the following, we demonstrate the different strategies of start basis generation on the example system *mixed.sys*:

$$\begin{aligned} 0.5x_1 &= x_2x_3 \\ 0.5x_2 &= x_1x_3 \\ 0.5x_3 &= x_1x_2 \end{aligned}$$

on $D_1 = D_2 = D_3 = [-1, 1]$. The basis variables are marked by a box around them. When pivoting with x_{23}, x_{13}, x_{12}, the system is in reduced row-echelon form

$$\begin{aligned} 0.5x_1 &= \boxed{x_{23}} \\ 0.5x_2 &= \boxed{x_{13}} \\ 0.5x_3 &= \boxed{x_{12}} \end{aligned}$$

and can be completed with two inequalities of the Bernstein polytope for x_{ii} into a start basis. Figure 6 shows a statistics of the interval width of the objective value in the course of the pivoting steps. The computation (described in [3]) performs 54 reductions (382 pivot steps in total), 11 bisections, and the solution time is 0.25s (Windows XP, Intel Pentium T7400, 2.2GHz). The worst condition number of the tableau is 373.2.

When pivoting with x_1, x_{13}, x_{12}, the system is in reduced row-echelon form

$$\begin{aligned} 0.5\boxed{x_1} &= x_{23} \\ 0.5x_2 &= \boxed{x_{13}} \\ 0.5x_3 &= \boxed{x_{12}} \end{aligned}$$

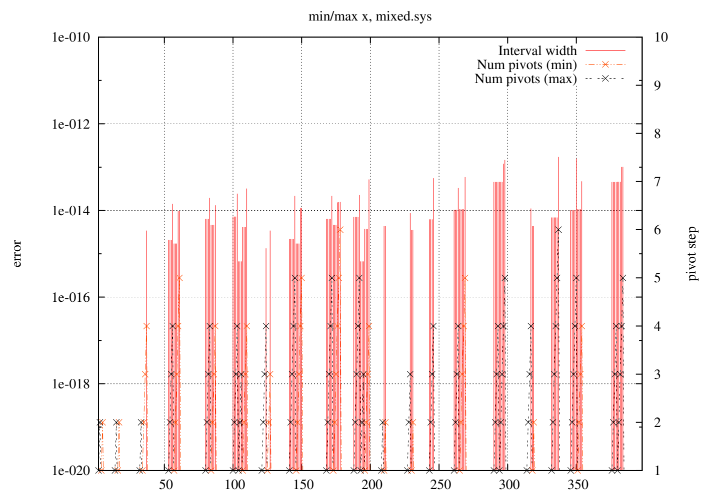


Fig. 6 System *mixed.sys* with start basis from Bernstein polytope for x_{ii}. Errors are derived from the interval width of the objective value. The average number of pivot steps per reduction is 4.07.

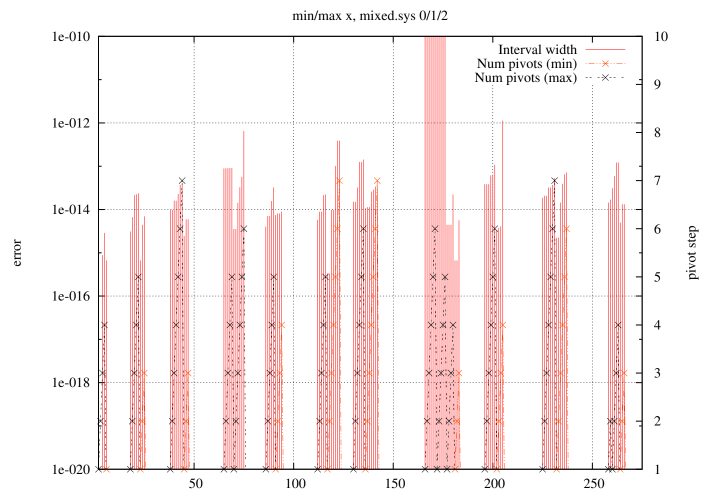


Fig. 7 System *mixed.sys* with start basis from Bernstein polytope for x_{ij}. Errors are derived from the interval width of the objective value. The average number of pivot steps per reduction is 4.53.

and can be completed with three inequalities of the Bernstein polytope x_{23} into a start basis. Note that the reduced row-echelon form for variables x_2 and x_3 is similar and thus omitted here. Figure 7 shows a statistics of the interval width of the objective value in the course of the pivoting steps. The computation (described in [3]) performs 32 reductions (275 pivot steps in total), 8 bisections, and the solution time is 0.15s (Windows XP, Intel Pentium T7400, 2.2GHz). The worst condition number of the tableau is 308800, and it results in larger objective value intervals, i.e., worse lower bounds. In the comparison, the second start basis results in less pivoting steps in the dual simplex iteration. Tableaus of worst condition number normally occur due to the Bernstein inequalities for very small intervals D_i. Such polytopes can be avoided by replacing them with thick planes or lines as described in [3].

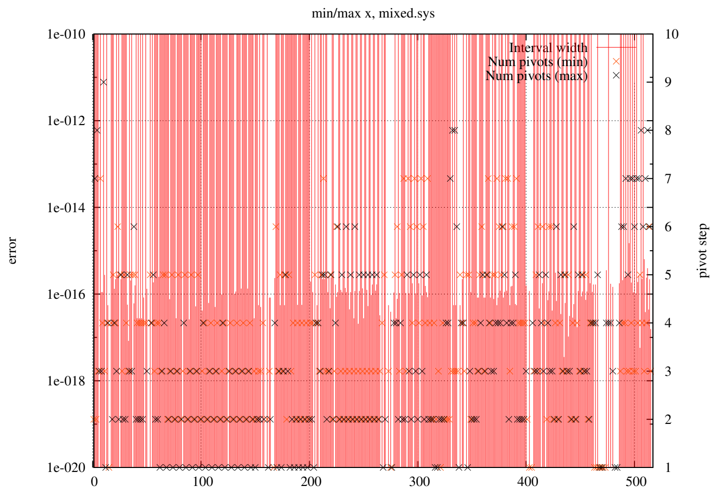


Fig. 8 System *mixed.sys* solved by SoPlex. Errors are derived from the duality gap $(c^t - {y^*}^t A)x$, $x \in D$, where y^* is the corresponding dual vector to the primal solution vector x^*.

In general, the tableau form tends to populate rows quickly. On the system *mixed.sys*, the user and Bernstein region of the tableau get filled approximately 60% to 80%.

For comparison with the revised simplex implementation *SoPlex*, we compute a rigorous lower bound $b^t y + e$ using the duality gap

$$e = \min\{(c^t - {y^*}^t A)x : x \in D\}$$

The primal solution vector x^* is directly available, and the corresponding dual solution vector y^* can be derived from the constraint slacknesses at x^*. Figure 8 shows the statistics of the duality gap sizes during solving the system *mixed.sys*. Large duality gap sizes (larger than 10^{-10}) result from linear programs, where no bound reduction could be achieved due to an early termination of the simplex implementation in *SoPlex*.

4 Conclusion

In this paper, we have presented a new way to implement the dual simplex algorithm in tableau form with interval pivoting steps for the direct computation of a rigorous lower bound. Such an algorithm can be used for example in a polynomial system solver using linear relaxations. Compared to a lower bound computed from the duality gap, it is not so much affected by the condition of the given linear program and an early termination of the simplex code. But due to the use of the tableau form it performs more floating point operations than state-of-the-art revised simplex implementations (e.g., SoPlex [7]). The algorithm can not exploit the system's sparsity easily, so that a pivoting step requires $\Theta(nm)$ interval operations.

Nevertheless, the computation is based on Gauss-Jordan pivoting and has a corresponding memory access pattern. Due to this regularity, it might be a good candidate for a fine-grained parallel implementation of the dual simplex algorithm.

In future work, we want to consider reducing the number of variables x_{ij} by replacing some of them by its interval $D_i \cdot D_j$. Additionally, we would like to look into using an interval right hand side similarly in a revised implementation of the dual simplex algorithm.

References

1. Ch. Keil *Lurupa – Rigorous Error Bounds in Linear Programming*, Algebraic and Numerical Algorithms and Computer-assisted Proofs (B. Buchberger, S. Oishi, M. Plum, S. Rump), Dagstuhl Seminar Proceedings (Number 05391) 2006.
2. Ch. Keil *A Comparison Of Software Packages For Verified Linear Programming*, submitted to SIAM Journal on Optimization 2008.
3. Ch. Fünfzig, D. Michelucci, and S. Foufou *Nonlinear systems solver in Floating-Point Arithmetic using LP Reduction*, ACM/SIAM Symposium on Solid and Physical Modeling 2009.
4. Ch. Papadimitriou, K. Steiglitz *Combinatorial optimization: algorithms and complexity*, Dover 1998.
5. R.E. Bixby *Solving Linear and Integer Programs*, Block Course Combinatorial Optimization at Work, Berlin 2009.
6. R.B. Kearfott *Discussion and Empirical Comparisons of Linear Relaxations and Alternate Techniques in Validated Deterministic Global Optimization*, Optimization Methods and Software 21(5), pp. 715-731, 2006.
7. R. Wunderling *SoPlex Library Version 1.4.2*, Technical Report, Zuse Institute, Berlin 1996.