Mathematical Programming Study 31 (1987) 167–174
North-Holland

ON PRACTICAL STOPPING RULES FOR THE SIMPLEX METHOD

Włodzimierz OGRYCZAK

Department of Mathematics, Informatics and Mechanics, University of Warsaw, 00-901 Warsaw, Poland

Received 19 November 1984
Revised manuscript received 22 December 1985

Well defined feasibility tolerances are necessary to guarantee reliable results for the simplex algorithm. This note presents some formulae for dynamic definition of suitable values of the tolerances at each simplex step. They are based on error analysis techniques and seem to be applicable in the standard simplex codes. The tolerances proved to be useful in practice when lexicographic LP problems were solved.

Key words: Linear programming, simplex method, round-off errors, tolerances.

1. Introduction

Well defined tolerances for primal and dual feasibility are necessary in order to guarantee reliable results from the simplex algorithm. They define practical stopping rules for the simplex algorithm. If the tolerances are too small then no solution may be found to be optimal. On the other hand, if the tolerances are too large then many basic solutions may be declared as optimal. Thus, the tolerances must be large enough that the stopping rules (i.e., primal and dual feasibility conditions) are satisfied for the computed values of the original optimal solution and small enough that any current basic solution which satisfies the stopping rules can be taken as a good approximation to the original optimal solution.

Most linear programming codes require the definition of such feasibility tolerances as well as several others. In commercial systems default values are supplied for the tolerances but they are proposed regardless of the special structure of problem and therefore poor results can be obtained for some harder problems. LP problems with dense matrices are usually considered among the most difficult. However, the most spectacular example of failure of the feasibility tolerances mechanism we noticed occurred while solving a typical sparse LP problem (815 rows, 1050 columns, density = 0.3%). The model was connected with optimal districting, Its constraints took fom of a transportation problem with some additional linear inequalities (for details of the model see [5]). While solving the problem with the standard MPSX/370 strategy (see [3]) we got a permanent cycle on two bases (see Table 1). This cycle

Table 1

MPSX/370 R1.6 PTF9 MPSCL EXECUTION							
ITER NUMBER	VECTOR OUT	VECTOR IN	REDUCED COST	NUMBER NONOPT	FUNCTION VALUE	NUMBER INFEAS	SUM INFEAS
M 57	1579	945	0.50000-	1	135 646.2481	1	49.3750
M 58	945	1579	0.69500-	1	135 596.2080	2	66.7500
M 59	1579	945	0.50000-	1	135 646.2481	1	49.3750
M 60	945	1579	0.69500-	1	135 596.2080	2	66.7500
M 61	1579	945	0.50000-	1	135 646.2481	1	49.3750
M 62	945	1579	0.69500-	1	135 596.2080	2	66.7500
M 63	1579	945	0.50000-	1	135 646.2481	1	49.3750
M 64	945	1579	0.69500-	1	135 596.2080	2	66.7500
M 65	1579	945	0.50000-	1	135 646.2481	1	49.3750
M 66	945	1579	0.69500-	1	135 596.2080	2	66.7500
M 67	1579	945	0.50000-	1	135 646.2481	1	49.3750
M 68	945	1579	0.69500-	1	135 596.2080	2	66.7500
M 69	1579	945	0.50000-	1	135 646.2481	1	49.3750
M 70	945	1579	0.69500-	1	135 596.2080	2	66.7500
M 71	1579	945	0.50000-	1	135 646.2481	1	49.3750
M 72	945	1579	0.69500-	1	135 596.2080	2	66.7500
M 73	1579	945	0.50000-	1	135 646.2481	1	49.3750
M 74	945	1579	0.69500-	1	135 596.2080	2	66.7500
M 75	1579	945	0.50000-	1	135 646.2481	1	49.3750
M 76	945	1579	0.69500-	1	135 596.2080	2	66.7500
M 77	1579	945	0.50000-	1	135 646.2481	1	49.3750
M 78	945	1579	0.69500-	1	135 596.2080	2	66.7500
M 79	1579	945	0.50000-	1	135 646.2481	1	49.3750
M 80	945	1579	0.69500-	1	135 596.2080	2	66.7500
M 81	1579	945	0.50000-	1	135 646.2481	1	49.3750
M 82	945	1579	0.69500-	1	135 596.2080	2	66.7500
M 83	1579	945	0.50000-	1	135 646.2481	1	49.3750
M 84	945	1579	0.69500-	1	135 596.2080	2	66.7500
M 85	1579	945	0.50000-	1	135 646.2481	1	49.3750
M 86	945	1579	0.69500-	1	135 596.2080	2	66.7500
M 87	1579	945	0.50000-	1	135 646.2481	1	49.3750
M 88	945	1579	0.69500-	1	135 596.2080	2	66.7500
M 89	1579	945	0.50000-	1	135 646.2481	1	49.3750
M 90	945	1579	0.69500-	1	135 596.2080	2	66.7500
M 91	1579	945	0.50000-	1	135 646.2481	1	49.3750
M 92	945	1579	0.69500-	1	135 596.2080	2	66.7500
M 93	1579	945	0.50000-	1	135 646.2481	1	49.3750
M 94	945	1579	0.69500-	1	135 596.2080	2	66.7500
M 95	1579	945	0.50000-	1	135 646.2481	1	49.3750
M 96	945	1579	0.69500-	1	135 596.2080	2	66.7500
M 97	1579	945	0.50000-	1	135 646.2481	1	49.3750
M 98	945	1579	0.69500-	1	135 596.2080	2	66.7500
M 99	1579	945	0.50000-	1	135 646.2481	1	49.3750
M 100	945	1579	0.69500-	1	135 596.2080	2	66.7500
M 101	1579	945	0.50000-	1	135 646.2481	1	49.3750
M 102	945	1579	0.69500-	1	135 596.2080	2	66.7500

was evidently caused by failure of the feasibility tolerances mechanism.[1] Moreover, it turned out that both the bases were well-conditioned.

The need for reliable feasibility tolerances setting is accepted by all researchers and practitioners. Nevertheless, error analysis of the corresponding quantities has not been performed up to now. What is more, a formula based on error analysis has been proposed for only one specific tolerance connected with updating of basis factorization in the Forrest-Tomlin algorithm (see [9]), whereas a typical simplex code uses at least ten various tolerances.

There exist some experimental investigations of round-off error propagation in the simplex method (see e.g. [6]) but they do not offer any formula for proper values of the feasibility tolerances. They are rather concerned with the so-called general zero tolerance for elimination all non-significant elements. This tolerance is proposed regardless of the specific quantities calculated in feasibility criteria and is usually too small for a feasibility tolerance. It is not used as a feasibility tolerance in any commercial simplex code. On the other hand, some authors propose several numbers as proper feasibility tolerances. For instance in [8] 10^{-8} is suggested as a proper value for feasibility tolerances whereas 10^{-5} is proposed in [7]. Both the values are proposed for the same computation model with the relative computer precision 10^{-15}.

Rigorous feasibility tests with tolerances based on error analysis are especially needed in lexicographic optimization. Lexicographic linear programming leads to solving a sequence of ordinary LP problems constructed in such a way that optimal set to the previous problem is treated as feasible set to the next one (see [4]). Due to this construction the whole optimization process strongly depends on proper definition of primal as well as dual feasibility tolerances. What is more, the lexicographic optimization is, in general, unstable.

The purpose of this note is to present some formulae for feasibility tolerances based on global error bounds. We carefully analyze relations between the feasibility tolerances and the accuracy of the computed basic solutions. We get formulae for dynamic definition of suitable values of the tolerances at each simplex step. The formulae seem to be applicable in the standard simplex codes.

2. The formulae

Consider a linear programming problem in the standard form

P: $\min\{cx: Ax = b, x \geq 0\}$

and its dual

D: $\max\{yb: yA \leq c\}$

where:

[1] Data file is available on request.

x is an $n \times 1$ vector of (primal) variables,
y is a $1 \times m$ vector of (dual) variables,
b is an $m \times 1$ vector,
c is a $1 \times n$ vector,
A is an $m \times n$ matrix of rank m ($m \leq n$).

While solving the problem P with the simplex method a sequence of basic solutions is constructed and their optimality properties are examined. The current basic solutions x and y are defined as follows:

- the basic part of x (denoted by x_B) is the solution of the linear system $Bx_B = b$,
- the nonbasic part of x (denoted by x_N) is the zero vector,
- y is the solution of the linear system $yB = c_B$,

where B is the current basis and c_B denotes the vector of cost coefficients c_j corresponding to basic columns.

Optimality criteria for the basic solutions can be written as

$$x_B \geq 0, \tag{1}$$

$$c_N - yN \geq 0, \tag{2}$$

where N is the nonbasic part of the matrix A, and c_N denotes the vector of cost coefficients corresponding to nonbasic columns. The inequalities (1)-(2) may be considered as stopping rules for the simplex method.

While computing in finite arithmetic, round-off errors may cause that the inequalities (1) and (2) are not satisfied for any computed basic solution. In other words, it is possible that no basic solution will be found as optimal. Therefore in practice it is necessary to introduce some tolerances into the simplex stopping rules. Finally, we obtain practical stopping rules for the simplex method as the inequalities

$$x_j \geq -pft_j \qquad \text{for } j \in J_B, \tag{3}$$

$$c_j - yA_j \geq -dft_j \qquad \text{for } j \notin J_B, \tag{4}$$

where J_B denotes the set of basic indices, A_j denotes the j-th column of the matrix A, pft_j and dft_j are the j-th tolerances for primal and dual feasibility, respectively. The tolerances, obviously, take some positive values. They may be defined once for the whole algorithm or redefined at each simplex step.

The feasibility tolerances should be so large that the computed vectors of the original optimal solutions satisfy the stopping rules (3) and (4). This requirement may be formulated as follows:

- at each simplex step the primal feasibility tolerances should be no less than the errors of the computed primal basic solution, and similarly,
- at each simplex step the dual feasibility tolerances should be no less than the errors of the computed reduced costs ($d_j = c_j - yA_j$).

So, the problem of the minimal values of feasibility tolerances is equivalent to the problem how large errors can occur during the computations of basic solutions. Narrow bounds on these errors will give some formulae for proper values of the feasibility tolerances.

Consider the computed basic solution $\tilde{x}$. The following formula for the error is valid

$$ex_j \equiv \tilde{x}_j - (B^{-1})_j b = (B^{-1})_j (B\tilde{x}_B - b) \quad \text{for } j \in J_B$$

where $(B^{-1})_j$ denotes the row of the inverse that corresponds to the variable x_j. Thus, the errors of the computed primal basic solutions can be bounded by

$$|ex_j| \leq \|(B^{-1})_j\| \, \|B\tilde{x}_B - b\| \quad \text{for } j \in J_B \tag{5}$$

where $\|\cdot\|$ denotes any matrix p-norm.

Now concentrate on computing the reduced costs d_j. For examining optimality the reduced costs are usually computed by the so-called backward transformation (see [8]). This is two stage technique. The reduced costs are calculated there as $d_j = c_j - yA_j$ after the dual vector y has been found using some basis factorization. The computational error of the reduced cost is then a sum of two errors. The first one, ∂_j, is due to replacing the exact dual solution $y = c_B B^{-1}$ by the computed approximation $\tilde{y}$. The second one δ_j is due to computing in finite arithmetic the value of the term $c_j - \tilde{y}A_j$. So, the following formula for the error of reduced costs is valid

$$\begin{aligned} ed_j &\equiv \tilde{d}_j - (c_j - c_B B^{-1} A_j) = \partial_j + \delta_j \\ &= (c_j - \tilde{y}A_j) - (c_j - c_B B^{-1} A_j) + \delta_j \\ &= (\tilde{y}B - c_B \mid \delta_j) \begin{pmatrix} -B^{-1}A_j \\ 1 \end{pmatrix}. \end{aligned}$$

Finally,

$$ed_j = (\tilde{y}B - c_B \mid \delta_j e_j) \begin{pmatrix} -B^{-1}A_j \\ e_j \end{pmatrix},$$

where e_j denotes the $m-n$ dimensional unit vector (in the space of nonbasic variables) corresponding to the variable x_j.

Let ε_j denote an upper bound on the error δ_j. Then, similarly to (5), we obtain

$$|ed_j| \leq \|B^{-1}A_j\| \, \|\tilde{y}B - c_B\| + \varepsilon_j \quad \text{for } j \notin J_B. \tag{6}$$

The basic residual vectors $B\tilde{x}_B - b$ and $\tilde{y}B - c_B$ may be computed at each simplex step for control of the so-called reinvert mechanism. Large values of their norms mean the loss of accuracy of basic solutions and indicate the need of refactorization of the current basis.

However, if we use computed residual vector instead of the exact one we must take into consideration the errors of computations since while computing residual vector in finite arithmetic a large relative error can occur. The theoretical inequalities (5) and (6) then take the following form:

$$|ex_j| \leq \|(B^{-1})_j\| \, (\|\mathrm{fl}(B\tilde{x}_B - b)\| + r^P) \qquad \text{for } j \in J_B, \tag{7}$$

$$|ed_j| \leq \left\| \begin{pmatrix} -B^{-1}A_j \\ e_j \end{pmatrix} \right\| (\|\mathrm{fl}(\tilde{y}B - c_B)\| + r^d) \qquad \text{for } j \notin J_B, \tag{8}$$

where $\mathrm{fl}(\cdot)$ denotes quantities computed in floating-point arithmetic; r^p and r^d denote bounds on computational error for the primal and dual residual vectors, respectively (the latter bound cover also the error δ_j).

The factors $\|(B^{-1})_j\|$ and $\|\binom{-B^{-1}A_j}{e_j}\|$ can be regarded as available in practical simplex algorithms since they are used in some pivot selection mechanisms. The vectors

$$\alpha_j = \begin{pmatrix} -B^{-1}A_j \\ e_j \end{pmatrix} \quad (\text{for } j \notin J_B)$$

point down the $n-m$ edges of feasible region that emanate from the current vertex x. Similarly, the vectors $\beta_j = (B^{-1})_j$ (for $j \in J_B$) point down the m edges that emanate from the current vertex y. Their spectral norms are used as normalizing scales in the so-called steepest edge strategy which is known to be very effective to reduce the number of simplex steps (see [10]). Direct computations of all the norms of the α and β vectors at each simplex step is too expensive. They can be cheaply computed, however, by special updating formulae, especially when a triangular basis factorization is used (see [1]). Most linear programming codes compute the normalizing scales only approximately using the so-called DEVEX technique (see [2]).

Returning to our tolerances we obtain, finally, the following requirements:

$$pft_j / \|\beta_j\| \geq \|\mathrm{fl}(B\tilde{x}_B - b)\| + r^p \quad \text{for } j \in J_B, \tag{9}$$

$$dft_j / \|\alpha_j\| \geq \|\mathrm{fl}(\tilde{y}B - c_B)\| + r^d \quad \text{for } j \notin J_B. \tag{10}$$

These inequalities state necessary relations between the accuracy of the basic solutions and the feasibility tolerances.

Note that the right-hand sides of the inequalities (9) and (10) are independent of the index j. So, the minimal values of the feasibility tolerances are proportional to the corresponding vectors of normalizing scales, i.e., they can be expressed as

$$pft_j = t\,\|\beta_j\| \quad \text{for } j \in J_B, \tag{11}$$

$$dft_j = t^*\,\|\alpha_j\| \quad \text{for } j \notin J_B, \tag{12}$$

where

$$t = \|\mathrm{fl}(B\tilde{x}_B - b)\| + r^p, \tag{13}$$

$$t^* = \|\mathrm{fl}(\tilde{y}B - c_B)\| + r^d. \tag{14}$$

In standard simplex codes we are usually interested in the highest accuracy. Thus, the feasibility tolerances should be defined then as dynamic quantities according to the equalities (11) and (12) where scalars t and t^* are calculated at each simplex step from the formulae (13) and (14).

In simplex codes the maximal norms of the basic residual vectors are usually bounded by some algorithmic parameters referred to as check tolerances (see [8]), i.e.,

$$\|\mathrm{fl}(B\tilde{x}_B - b)\| \leq pcht, \tag{15}$$

$$\|\mathrm{fl}(\tilde{y}B - c_B)\| \leq dcht. \tag{16}$$

Thus, due to (15) and (16) the check tolerances can be used in formulae (13) and (14) instead of the norms of computed residual vectors. Moreover, proper values for check tolerances fulfil usually inequalities

$$r^p \leq pcht \quad \text{and} \quad r^d \leq dcht.$$

The scalars t and t^* can be defined then as the doubled check tolerances.

Stopping rules defined according to the above scheme seem to be quite easy for handling in simplex codes which use the steepest-edge strategy or some approximation to it. Namely, if normalizing scales are used during pricing then the quantities $d_j/\|\alpha_j\|$ or $x_j/\|\beta_j\|$ are explicitly computed. For such normalized quantities our tolerances take the values t^* and t, respectively. These values are independent of index j and can be defined as a sum of the check tolerance and the corresponding error bound (r^p or r^d, respectively). The check tolerances are, certainly, explicitly available as algorithmic parameters. The second term of the sum can be easily computed or simply treated as a parameter depending on the class of LP problem (in particular equaled to the check tolerance). The latter way seems to be more applicable in the standard simplex codes.

The above formulae for feasibility tolerances was used while solving practical large-scale lexicographic LP problems (about 1000 rows and 2000 variables). Namely, we were solving these problems with the MPSX/370 package (see [3]) using the feasibility tolerances defined according to the formulae (11)—(14). We observed good performances of these tolerances whereas using default tolerances we came across numerical troubles in a few runs. More precisely, while using default tolerances the loss of feasibility was frequently observed. Moreover, during some runs the problem was classified as infeasible after loss of feasibility whereas a feasible solution was found several simplex iterations backward.

3. Conclusion

Careful analysis of bounds on round-off errors in the simplex stopping rules showed that the feasibility tolerances should be proportional to product of the corresponding normalizing scales and the check tolerances. We used this fact for construction some formulae for dynamic definition of the feasibility tolerances at each simplex step. These formulae proved to be useful while solving practical lexicographic LP problems. The formulae were based on worst case error analysis and they can generate overestimated tolerances for some problems. For this reason they need further experiments on various practical LP problems.

References

[1] D. Goldfarb and J.K. Reid, "A practicable steepest-edge simplex algorithm," *Mathematical Programming* 12 (1977) 361-371.

[2] P.M.J. Harris, "Pivot selection methods of the DEVEX LP code," *Mathematical Programming* 5 (1973) 1-28.

[3] IBM Mathematical Programming System—Extended/370 (MPSX/370), Program reference manual, SH19-1095 (IBM, 1979).

[4] J.P. Ignizio and J.H. Perlis, "Sequential linear goal programming: Implementation via MPSX," *Computers and Operations Research* 3 (1980) 217-225.

[5] J. Malczewski and W. Ogryczak, "A multi-objective approach to optimization of health care districts" (To appear).

[6] H. Mueller-Merbach, *On Round-off Errors in Linear Programming* (Springer-Verlag, Berlin, 1970).

[7] B.A. Murtagh, *Advanced linear programming: Computation and Practice* (McGraw-Hill, New York, 1981).

[8] W. Orchard-Hays, *Advanced Linear Programming Computing Techniques* (McGraw-Hill, New York, 1968).

[9] J.A. Tomlin, "An accuracy test for updating triangular factors," *Mathematical Programming Study* 4 (1975) 142-145.

[10] P. Wolfe and L. Cutler, "Experiments in linear programming," in: R. Graves and P. Wolfe, eds., *Recent Advances in Mathematical Programming* (McGraw-Hill, New York, 1963) 177-200.