TR/02/87 February 1987

An Investigation of Pricing Strategies Within Simplex

by

Gautam Mitra
Ivonia Rebelo
Mehrdad Tamiz

(Pcr)
QA
1
B7C

AN INVESTIGATION OF PRICING STRATEGIES WITHIN SIMPLEX

by

Gautam Mitra*

Ivonia Rebelo**

Mehrdad Tamiz*

February1 987

*Department of Mathematics and Statistics, Brunel University.

**Economics Department, City of London Polytechnic.

ABSTRACT

The PRICE step within the revised simplex method for the LP problems is considered in this report. Established strategies which have proven to be computationally efficient are first reviewed. A method based on the internal rate of return is then described. The implementation of this method and the results obtained by experimental investigation are discussed.

CONTENTS

1. Introduction

Several approaches have been suggested to improve the pricing step in the simplex method which involves substantial computational effort. Among the major developments which have now become accepted industry's standard are the multiple and partial pricing methods, the devex method and the steepest-edge algorithm. More recently an alternative method, which considers the rate of return of each non-basic variable with negative reduced cost, has been suggested by Dr. Keyzer. The method has been found to have an efficiency gain of 20 to 60 percent over the simplex pricing method on some medium to large scale LP problems.

Experimental tests were undertaken using a set of small but representative industrial test problems to investigate whether this result held more generally. The purpose of the present paper is to report on these investigations

The paper is in five parts. The present section includes a definition of the notation and terminology, and an overview description of the simplex method. It is followed in section 2 by a survey of three of the major alternative pricing methods. The partial pricing and multiple pricing methods are presented in section 2.1. The devex method for pricing and the steepest-edge algorithms are described in section 2.2 and 2.3, respectively. These algorithms have proved, in practice, to give a worthwhile overall gain over the original pricing step of the simplex. The method recently published by Maros [MARO 86] appears to be promising. A description and investigation of this work is postponed until more experimental work is undertaken and the next version of this report.

Section 3 presents the alternative method developed by Dr. Keyzer at the Centre for World Food studies. Results of the investigation are presented in section 4, and finally some overall conclusions are made in section 5. Appendix contains a description of zero tolerances which determine logic tests used in the simplex steps.

Before proceeding to section 2 first the notation and terminology used in the paper is defined and then the revised simplex method is described in order to provide the framework within which the subsequent algorithms are described.

Notation and terminology

A	LP coefficient matrix
A_i	Matrix of non-basic columns
a_{ij}	The (i, j)th element of the original A
$\bar{a}_{ij}$	The (i, j)th element of A after one or more linear transformations
$\bar{a}_{ij}$	The (i, j)th element after one pivotal transformat ion of the above
$a_{.j}$	The jth column of A. For notational simplicity this is represented as a_j
B	Basis matrix
b_i	The ith value of the original rhs

$\bar{b}_i$	The ith value of the rhs after one or more linear transformations
$\bar{b}_i$	The ith value of the rhs after one pivotal transformation of the above
BTRAN	Backward transformation
C	Indices of the non-basic variables forming the reference frame (devex)
c_j	Cost coefficient
$\bar{c}_j$ or dj	Reduced cost of the jth non -basic variable
COL	Set of indices {1,2,........,n}
COL^+	Set of indices {l, 2,….,n, 0, n+1, ... , n+m}
E_i	The ith transformation matrix (eta-vector)
eta-file	The set of eta-vectors
FTRAN	Forward transformation
G_i	Set of column indices for partial pricing
LP	Linear programming

MROW, m	Number of rows in A (and B)
NCOL, n	Number of columns in A
$\Pi_{p\cdot}$, Π_p , Π	The pth row of the basis inverse. For notational simplicity this is represented as Π_p. or more simply as Π
R	Pivot positions for basic variables in C (devex)
rhs	Right hand side values
ROW	Set of indices {1,2,......,m}
ROW^+	Set of indices {0,1,2,......,m}
T_j	Weight assigned to the column j
XTOLDJ	Tolerance used for d_j
XTOLIN	Tolerance used for pivots during reinversion
XTOLPV	Tolerance used for pivots during simplex iterations
XT0LV	Tolerance used for rhs values
XTOLZE	Tolerance used for zero

The revised simplex method

The revised simplex mathod is stated in the following compact form which takes into account phase 1 and phase 2 (for the definition of phase 1 and phase 2 see [OHAY 68]).

Assume that B_k^{-1} is the inverse matrix at the kth iteration, that is

$$B_k^{-1} = E_k E_{k-1} \ldots . E_1 ,$$

then the steps of the revised simplex algorithm may be stated as:

step 1 :

a: Compute the form vector e.

This is a row vector of (m+1) components which takes the value 0 or ± 1.

b: Compute the pi-vector.

Obtain the pi-vector by which the reduced cost coefficient of a variable (column) is computed. This is done by performing a BTRAN operation on vector e:

$$\Pi^k = eB_k^{-1} = e\, E_k E_{k-1} \ldots .. E_1$$

step 2 : Pricing operation (PRICE)

Price out the variables (columns of A) by computing the inner product $\bar{a}_{oj} = \bar{c}_j = \Pi^k a_j$, ($\bar{w}_j$ in phase 1). Choose a column q for which $\bar{c}_q$ has the most negative value. If no such column exists go to EXIT.

step 3 : Column transformation (update)

Update the column q by performing an FTRAN operation on the column, that is

$$\bar{a}_q = B_k^{-1} a_q$$

or $$\bar{a}_q = E_k E_{k-1} \ldots\ldots E_1 a_q$$

step 4 : Choose a row (CHUZRO)

Choose a row p such that

$$\frac{\bar{b}_p}{\bar{a}_{pq}} = \min_{i \in ROW} \left\{ \frac{\bar{b}_p}{\bar{a}_{pq}} \middle| \bar{b}_i > 0 \text{ and } \bar{a}_{iq} > 0, \text{ or } \bar{b}_i < 0 \text{ and } \bar{a}_{iq} < 0 \right\}$$

if no such a row exists 20 to EXIT

step 5 : Update the solution values and the basis inverse B_k^{-1}

The solution vector is updated by the relation:

$$\bar{b}'_p = E_{k+1} \bar{b}$$

where $\bar{b}\,'$ denotes the new solution values .The basis inverse is updated by the relation

$$(B_{k+1})^{-1} = E_{k+1} B_k^{-1}$$

thus eta-file is augmented by the new eta E_{k+1}

go to step 1.

EXIT :

If control is transferred from step 2 and current status is phase 2 then optimum solution for the LP problem is found, else it is phase 1 and there cannot be a feasible solution to the problem. If control is transferred from step 4 (only possible in phase 2) then the optimum solution is not bounded.

For the computer implementation of this method it is also necessary to consider the zero tolerances for $\bar{c}_j$, $\bar{b}_i$ and $\bar{a}_{pq}$. These tolerances are described in the appendix A.l.

Within revised simplex, the pricing step involves substantial computational effort. This is specially true for long thin matrices, since a vector multiplication operation has to be carried out to obtain the reduced cost for each variable (column). Based on actual experience, it has long been recognised that computing the inner product Πa_j for all the columns a_j not in the basis and selecting the most negative reduced cost (dj) is not always the best computational strategy.

2. Pricing methods: a survey

2.1 Multiple and partial pricing [OHAY 68]

In the multiple pricing method at most p columns ($2 \leq p \leq 10$) with most negative d_j's are selected during the price pass. Subsequently this sub-matrix ($m \times p$) is optimised in explicit tableau format which does not require further FTRAN and BTRAN operations. To optimise over this ($m \times p$) sub-matrix, however, requires as many floating point words (work areas) in the main memory. This strategy has proved very effective in reducing both the total computational time and the average time per iteration for solving large scale problems. The steps of the revised simplex method using the multiple pricing strategy are as follows:

Step 1 Set up the pi-vector.

Step 2 Select at most p columns with an ordered set of negative d_j's. If p=0 go to EXIT.

Step 3 Update the sub-matrix using the FTRAN operation on the p Selected columns.

Step 4 Optimise over the sub-matrix:

(i) Select the column with most negative reduced cost. If none selected go to step 1.

(ii) Perform pivot selection on this column to determine the leaving basic variable. If none found go to EXIT.

(iii) Update the eta-file by this new eta, or update the solution values by the bound value if there is a bound change.

(iv) Update the sub-matrix and rhs values, go to (i).

EXIT If control is transferred from step 2 and current status is phase 2 the optimum solution for the LP problem is found, else it is phase 1 and there cannot be a feasible solution to the problem. If control is transferred from step 4 (ii) (only possible in phase 2) then the optimum solution is not bounded.

Another refinement to pricing operation is the partial pricing, where the columns of the A-matrix are partitioned into k portions, defining k sets of column indices.

$$G_1, G_2, \ldots.., G_k$$

such that

$$\bigcup_{t=1}^{k} G_t = COL$$

In general there are q columns in each of these portions thus

$$|G_1| \simeq |G_2| \simeq |G_3| \simeq \cdots\cdots \simeq |G_3| \simeq q.$$

Also in practice q is much larger than p.

During the price pass, the best p columns are selected from one portion beginning with the portion it left off with at the last pricing pass. Thus not all of the A-matrix need to be scanned in one price pass. This appears to achieve a direct saving on the total time required to optimise the problem, however, it is somewhat offset by the fact that column selections are based on only a portion of the matrix, whereby the total number of iterations usually increase.

<u>2.2 Devex method for pricing</u> [HARR 73]

In the simplex method, the value of the objective function differs at every iteration by

$$x_0 = x_0 - \left[d_j \quad \frac{\bar{b}_p}{\bar{a}_{pq}} \right]$$

which leads to the definition of gain

$$\text{gain} = \left[d_j \frac{\bar{b}_p}{\bar{a}_{pq}} \right]$$

The largest gain at any step depends on the reduced cost d_j (rate of objective change) and the resulting value of the entering non-basic variable. Unfortunatly this value is not known until the basic variables are interrogated. Substantial computational effort is needed if the non- basic variable with the greatest gain were to be chosen to enter the basis. Greenberg [GREE 78] states that, generally, a "best gain criterion" is effective in keeping the total number of iterations nearest to minimal in comparison with the strategy of selecting the non-basic variable with largest (absolute) reduced cost. A procedure for obtaining a good gain criterion is the devex method which was suggested by Harris [HARR 73]. This method is based on the concept of the gradient,

$$\text{gradient} = \frac{d_j}{||\bar{a}_j||} \quad ,$$

which is the rate of change in objective function value as defined using the norm of the updated column.

Choice of the most negative d_j (standard simplex rule) corresponds to choosing the largest gradient in the space of the current non-basic variables; a reference frame that changes from iteration to iteration, and which constantly discounts previous and future decisions. Harris's method is to maintain a constant reference frame (usually the initial set of non-basic variables) and compute the radients in this space by means of a set of (dynamic) weighting or scaling factors applied to the reduced costs.

Suppose that the original non-basic variables form the reference frame and assign unit column weights to all the columns, T_j=l, $j \in COL$. Let the set C contain the indices of these non-basic variables, and the set R contain the pivot positions of these reference variables. Initially R=Φ. The gradient in the space of these reference variables after a number of iterations is no longer simply d_j but d_j/T_j where

$$T_j = \surd\ (k_j + \sum_{i \in ROW} k_i\, \bar{a}_{ij}^{\,2}) \qquad ,$$

where k_j and k_i take the value 1 or 0 depending upon whether the pivot row i and the column j , respectively, belong to variables of the reference frame or not.

The T_j factors can be approximated from iteration to iteration by means of an updating formula which uses only the pivot row and the updated column at each iteration. Suppose that basic variable in the p^{th} row position is to be exchanged with the non-basic variable in the qth column position then

$$\begin{aligned} \bar{a}'_{pj} &= \bar{a}_{pj}/\bar{a}_{pq} \\ \bar{a}'_{pj} &= \bar{a}_{ij} - \bar{a}_{iq}\ \bar{a}'_{pj} \qquad , \end{aligned} \qquad (2.2,1)$$

suppose further that $p \notin R$ and $q \in C$ then

$$(T'_j)^2 = k_j + \sum_{i \in R}\ (\bar{a}'_{ij})^2 + (\bar{a}'_{pj})^2 \quad , \quad p \notin R \qquad (2.2.2)$$

from (2.2.1) and (2.2.2) it can be shown that

$$(T'_j)^2 = k_j + \sum_{i \in R} (\bar{a}_{ij})^2 + \sum_{i \in R} (\bar{a}_{ip}\bar{a}'_{pj})^2 \ + \ (\bar{a}'_{pj})^2 \ - 2\sum_{i \in R} \bar{a}'_{pj}\bar{a}_{ij}\bar{a}_{iq}$$

$$= k_j + \sum_{i \in R} (\bar{a}_{ij})^2 + (\bar{a}'_{pj})^2 \ \ (1 + \sum_{i \in R}\ (\bar{a}_{pj})^{2)} \ - 2\sum_{i \in R} \bar{a}'_{pj}\bar{a}_{ij}\bar{a}_{iq}$$

from which we have

$$T_j' = \sqrt{[T_j^2 + (\bar{a}_{pj}')^2 T_q^2]}$$

which roughly approximates to

$$T_j' = \max \quad (T_j \; , \; |\bar{a}_{pj}'| T_q) \; . \qquad (2.2.3)$$

The same approximation is obtained if $p \in R$ and $q \in C$ and for $q \notin C$ the slightly different approximation is given by.

$$T_j' = \max \; [T_j \; , \left|\bar{a}_{pj}'\right| \sqrt{(1+T_q^2)} \;] \; . \qquad (2.2.4)$$

The T_j' as given by (2.2.3) and (2.2.4) are used to update the weighting factors at every iteration before the pricing operation. This is accomplished by using a unit row vector e_p with unity in the pth row position and computing $\bar{e}_p = e_p B^{-1}$ and then $\bar{a}_{pj}' = \bar{e}_p a_j$ for $j \in COL$

Harris reports that the ratio between the calculated and the estimated value of T_q for the updated column q rarely exceeds 2.0, and its most usual value lies between 0.7 and 1.3. But if it falls below 0.2 a new reference frame is set up using the current non-basic variables and the weighting factors are reset to unity.

Many industry standard LP systems include this pricing strategy, or its variant, as a system defined algorithmic option.

2.3 Steepest-edge algorithm [GORE 77]

The steepest-edge method is based on approximate calculation of the gradient in the space of a fixed framework by a recurrence relation.

Let the A-matrix be partitioned into an mxm basis matrix B and a matrix of the non-basic columns A_1 such that $A = [B/A_1]$.

Now consider the matrix N where

$$N = \begin{vmatrix} B & A_1 \\ 0 & I \end{vmatrix}$$

where I is an identity matrix of order n-m. The inverse of this matrix is then

$$N^{-1} = \begin{vmatrix} B^{-1} & -B^{-1}A_1 \\ 0 & I \end{vmatrix}$$

If η_j (j >m) is the jth column of N^{-1}, then the reduced cost d_j of a non-basic variable is

$$d_j = c^T \eta_j .$$

Choosing the steepest-edge in the space of all the vriables is equivalent to minimising the normalised reduced cost

$$\frac{c^T \eta_j}{||\eta_j||} \tag{2.3.1}$$

Let $\lambda_j = ||\eta_j||^2 = \eta_j^T \eta_j$, then (2.3.1) becomes

$$\frac{c^T \eta_j}{\lambda_j^{1/2}} \quad . \quad j>m$$

Explicit computation of all $\lambda_j^{1/2}$, j>m , at each step of the simplex method is prohibitively expensive. The recurrence relations derived by Goldfarb and Reid calculates this value accurately without an enormous increase in the computational time per iteration.

Suppose the non-basic variable x_q is to be exchanged with the basic variable x_p, in the pth row position, at the next iteration. Then the recurrences giving the new values $\bar{\lambda}_j$ in terms of the old values λ_j are

$$\lambda_q = 1 + ||B^{-1}a_q||^2 \qquad (2.3.2)$$

$$\bar{\lambda}_q = \lambda_q \ / \ \bar{a}_q^2$$

$$\bar{\lambda}_j = \lambda_j - 2 \ \bar{a}'_{pj} \ a_j^T B^{-T} B^{-1} a_q + (\bar{a}'_{pj}) \ 2\lambda_q. \qquad (j \neq q)$$

Formula (2.3.2) fives a current non-recursive calculation for the variable leaving the basis. It may be used to provide a simple check on round-off errors by comparing it with the recurred value.

Goldfarb and Reid reported that the total number of iterations and the total time needed to solve their six real-life test problems required 33% less iterations and 7% less time than the Harris's devex algorithm. They conclude that the steepest edge and the Harris algorithm show a worthwhile overall gain over the original simplex algorithm.

3. An alternative pricing method [KEYZ 84]

Recently Keyzer [KEYZ 84] has suggested an approach which considers the rate of return of each non—basic variable with negative reduced cost d_j (which is the net return). Define the following notations:

$c_j+ \geq 0$ The original cost c_j, $c_j+ = c_j$ if $c_j \geq -\text{XTOLDJ}$. $c_j+ = 0$ otherwise

$c_j- \geq 0$ The original cost c_j, $c_j- = \text{abs}(c_j)$ if $c_j < -\text{XTOLDJ}$. $c_j- = 0$ otherwise

$\Pi_i+ \geq 0$ The positive entry in the ith row position of the pi-vector, $\Pi_i + \Pi_i$ if $\Pi_i \geq -\text{XTOLZE}$ $i \in \text{ROW}$. $\Pi_i+ = 0$ otherwise.

$\Pi_i- \geq 0$ The absolute value of the negative entry in the ith row position of the pi-vector, $\Pi_i^- = \text{abs}(\Pi_i)$ if $\Pi_i < -\text{XTOLZE}$, $i \in \text{ROW}$. $\Pi_i- = 0$ otherwise

$a_{ij}+ \geq 0$ The positive entry in the ith row position of the jth column, $a_{ij}+ = a_{ij}$ if $a_{ij} > -\text{XTOLZE}$, $i \in \text{ROW}$. $a_{ij}+ = 0$ otherwise

$a_{ij}- \geq 0$ The absolute value of the negative entry in the ith row position of the jth column, $a_{ij}- = \text{abs}(a_{ij})$ if $a_{ij} < -\text{XTOLZE}$, $i \in \text{ROW}$. $a_{ij}- = 0$ otherwise

c_j, a_{ij} and Π_i can now be expressed in terms of these notations as

$$\left.\begin{array}{lcl} c_j &=& c_j+ - c_j- \\ a_{ij} &=& a_{ij}+ - a_{ij}- \\ \Pi_i &=& \Pi_i+ - \Pi_i- \end{array}\right\} \quad \begin{array}{l} i \in \text{ROW} \\ j \in \text{COL+} \end{array}$$

For every non-basic variable x_j with $d_j < -\text{XTOLDJ}$ compute α_j^+ and α_j^- such that

$$\left.\begin{array}{ll} \alpha_j^+ = \Pi_i + a_{ij}^+ + \Pi_i^- a_{ij}^- & i \in \text{ROW} \\ \alpha_j^- = \Pi_i + a_{ij}^- + \Pi_i^- a_{ij}^+ & i \in \text{ROW} \\ \text{whereby} \quad d_j = \alpha_j^+ - \alpha_j^- + c_j . & \end{array}\right\} \qquad (3.1)$$

Taking into account the economist's interpretation of the LP model whereby the internal rate of return is considered, the variable choice may be stated as: find the variable q such that

$$\max\ (c_j^+ + \alpha_j^+)x_j$$

$$\text{subject to } (c_j^- + \alpha_j^-)x_j \leq \delta \quad ,$$

for a given arbitrary finite positive value δ. This can be easily shown to be equivalent to choosing the variable x_q for which the expression

$$\frac{c_q^+ + \alpha_q^+}{c_q^- + \alpha_q^-} \quad ,$$

takes the maximum value, Thus.

$$\frac{c_q^+ + \alpha_q^+}{c_q^- + \alpha_q^-} = \max_j \left\{ \frac{c_j^+ + \alpha_j^+}{c_j^- + \alpha_j^-} \,|\, d_j < -\text{XTOLDJ} \right\} \qquad (3.2)$$

If in (3.2), $0 \leq c_j^+ + \alpha_j^+ \leq \text{XTOLDJ}$ then the numerator is taken as C where C>XTOLDJ. If $0 \leq c_j^- + \alpha_j^- \leq \text{XTOLDJ}$ then the denominator is set to 1.

(3.1) is computed for each column during the price pass of the A-matrix. As it involves processing the two vectors a_j and Π. The procedure is easily implemented using the "price" routine of the simplex method.

4. Experimental results

Five test problems were used to investigate the performance of Keyzer's alternative pricing procedure, their characteristics are summarized in table 4.1. The first four problems are representative industrial test problems and are taken from the lower end of the collection of bench mark problems compiled for the validation of the FORTLP system [MITY 86], [TAMI 86].

TABLE 4.1: CHARACTERISTICS OF TEST PROBLEMS.

NO	NAME	SOURCE	NO OF ROWS	NO OF COL.	NO OF BOUNDS	NO OF NON-ZEROS	DENSITY IN %	NO OF DISTINCT NON-ZEROS
1	BEALE	LBU*	171	303	38	901	1.7%	19
2	BERGER	LBU	65	133	133	415	4.8%	16
3	BASEIN	SIA	48	60	28	209	7.3%	116
4	FULLJV29	SIA	201	230	163	864	1.9%	281
5	TESTIN	BRUNEL	5	7	0	12	34.3%	5

* Loughborough University.

Three alternative strategies were tried:

Strategy 1:

Use Keyzer in both phase 1 and phase 2 of the simplex replacing a zero $(c_j^+ + \alpha_j^+)$ by a small constant C which is assigned a value greater than the d_j tolerance XTOLDJ. In the present computations C was assigned the value 10 x XTOLDJ.

Strategy 2:

Use Keyzer in both phase 1 and phase 2 of the simplex bypassing the j th non-basic variable if $(c_j^+ + \alpha_j^+) = 0$.

Strategy 3:

Use Keyzer in phase 1 bypassing the j^{th} non-basic variable if $(c_j^{+} + \alpha_j^{+}) = 0$. In phase 2, apply the original simplex pricing method.

The experimental results using the alternative strategies are presented in table 4.2.

TABLE 4.2: EXPERIMENTAL RESULTS

NO	OBJECTIVE FUNCTION	OPTIMUM VALUE	NO OF ITERATIONS			
			SIMPLEX	STRATEGIES		
				1	2	3
1	Min	0.0	138	138	>>146	146
2	Min	811.84	81	80	64	104
3	Min	127286.51	35	35	29	35
4	Min	NO-FEAS	87	87	82	82
5	Min	23.0	7	5	5	5

The results in table 4.2 indicate that for strategy 1 both the simplex method and Keyzer perform equally well in most cases. In the case of strategy 3 Keyzer performs rather poorly, however, in the case of strategy 2 Keyzer's method performs better for three of the test problems.

5. Conclusion

In this paper, alternative pricing methods have been discussed, it is well accepted that the devex and steepest descent methods are superior to the standard pricing algorithm which is used by the simplex method. Experimental tests on Keyzer's approach suggests that its performance may depend on the problem structure and the choice of strategy.

i) The structure of the problem.

Keyzer [KEYZ 84] states that: the return-over-cost selection rule appears in practice to function better. In practical medium to large scale LP applications the change has produced efficiency gain from 20 to 60 percent depending on the problem (seven LP models for the agricultural sector of developing countries, six of about 100 rows and 150 columns and one of 1100 rows and 900 columns). The return-over-cost rule may not always prove to be more efficient.

ii) The strategy employed.

For example, it was found that if, in the case of the fourth test problem, before optimisation, ZCRASH was used to pivot out the artificial variables and then Keyzer was applied then in this case, 77 iteration were required to obtain the non-feasible solution.

REFERENCES

[APEX 77] APEX III, "Reference Manual, version 1.1", Control Data Corporation, Minneapolis, USA, (1977) .

[GORE 77] Goldfarb, D., and Reid, J.K., "A practical steepest-edge simplex algorithm", Maths. Prog., Vol. 12, No. 3, June (1977).

[GREE 78] Greenberg, H.J., "Pivot selection techniques", Design and implementation of optimisation software, Greenberg, H.J., (ed.), Sijthoff and Noordhoff, (1978).

[HARR 73] Harris, P.M.J., "Pivot Selection Methods of the Devex LP Code", Mathematical Programming, vol. 5, pp. 1-28, (1973).

[KEYZ 84] Keyzer, M.A., "Distorted linear programming with applications to linear economic models", Staff working paper SOW-84-08, Centre for World Food Studies, Vrije Universiteit, Amsterdam, Holland, (1984) .

[MARO 86] Maros, I., "A general phase-I method in linear programming", European Journal of Operations Research, Vol. 23, (1986).

[MIBE 69] Mitra, G.,and Beale, E.M.L., private communications, (1969).

[MITY 85] Mitra, G., Tamiz, M., and Yadegar, J., "FORTLP: A Linear, Integer and Nonlinear Programming System, Preliminary User Manual", Brunel University, July 1985.

[MPSX 71] MPSX, "Mathematical Programming System Extended", Program number 5734 XM4, IBM Trade Corporation, New York, USA, (1971).

[MUAT 81] Murtagh, B.A., "Advanced Linear Programming: Computation and Practice", McGraw-Hill, New York, (1981).

[OHAY 68] Orchard-Hays, W., "Advanced Computing Techniques in Linear Programming", McGraw-Hill, New York, (1968).

[TAMI 86] Tamiz, M., "Design, implementation and testing of a general LP system exploiting sparsity", PhD Thesis, Brunel University, (1986).

APPENDIX Zero tolerances for algorithmic steps [OHAY 68]

In this appendix some zero tolerances which determine logic tests used in the simplex steps of the LP computations are discussed and their suggested numerical values given.

Zero tolerances are introduced in to LP optimisers in order to control the degree of accuracy obtained in the numerical computations and to improve computing time. Their purpose is to eliminate noise that would otherwise be introduced indefinitely by algebraic operations.

The most basic tolerance is the threshold value, XTOLZE, for the magnitudes of real numbers. Two real numbers a and b are said to be equal if their difference is within this tolerance, that is

$$-\mathrm{XTOLZE} \leq a-b \leq \mathrm{XTOLZE} \quad .$$

Similarly a real number c is considered to take zero value if

$$-\mathrm{XTOLZE} < c < \mathrm{XTOLZE} \quad .$$

This tolerance is normally set to 10^{-12}.

Another important tolerance is the pivot (rejection) tolerance, XTOLPV, which prevents a coefficient very close to zero from being considered as a pivot element in a simplex iteration, hence directly affecting the stability of the computational steps. The value $10^{-\theta}$ is normally used for this purpose. If this tolerance is increased then it is known to increase the total run time for a solution [MUAT 81].

A second pivot tolerance XTOLIN is used during the reinversion of the basis matrix to test the size of a proposed pivot. The value of this tolerance is related to the value of XTOLPV. Normally XTOLIN is larger than XTOLPV to ensure that at the end of the reinversion the basis is not singular. The value of 10^{-6} is usually used for this tolerance.

Feasibility tolerances XTOLV (for primal solution values) and XTOLDJ (for dual solution values) cure designed to check whether or not the values of the basic variables (rhs values) and the reduced cost coefficients are feasible, respectively. A larger value is used for the reduced cost coefficient than for the rhs values since computations with the pie-vector tend to have more error. The values 10^{-7} and 10^{-6} are normally used for XTOLV and XTOLDJ respectively. Increasing the value of XTOLV may give a shorter total run time, but increasing the value of XTOLDJ may cause unnecessary extra iterations.

In general it is well known among computational LP specialists [see for example MIBE 69] that the following ordered values are used for tolerances

$$\text{XTOLDJ} > \text{XTOLV} > \text{XTOLIN} > \text{XTOLPV} > \text{XTOLZE} .$$

In most LP optimisers the value of these tolerances can be altered by the user to suit the environment. Typical tolerance values used for a 60 bit floating point word giving approximately 15 decimal places accuracy are:

$$\text{XTOLDJ}=10^{-5}$$

$$\text{XTOLV}=10^{-6}$$

$$\text{XTOLPV}=10^{-8}$$

$$\text{XTOLZE}=10^{-10}$$

Tolerances as used in MPSX [MPSX 71] and APEX [APEX 77] are set out in table A. 1. Note that IBM uses 64 bit double precision floating point representation, and CDC uses 60 bit double precision floating point representation.

TABLE A.1: TOLERANCES IN MPSX AND APEX

	Tolerance variable	Identifier	Default value
MPS X	zero	XTOLZE	10^{-30}
	reduced cost	XTOLDJ	10^{-5}
	rhs value	XTOLV	10^{-6}
	pivot value during primal iterations	XTOLPIV	10^{-6}
	pivot value during reinversion	XTOLINV	10^{-6}
APEX	zero	RTPACK	10^{-13}
	reduced cost	RTDINF	10^{-5}
	rhs value	RTINFZ	10^{-6}
	pivot value during primal iterations and reinversion	RTPVMIN	10^{-8}