STATE UNIVERSITY OF NEW YORK AT STONY BROOK

COLLEGE OF ENGINEERING

REPORT No. 106

ON THE CHEBYSHEV SOLUTION OF INCONSISTENT LINEAR EQUATIONS

by

R. P. Tewarson

MARCH 1, 1968

REPORT NO. 106

ON THE CHEBYSHEV SOLUTION OF INCONSISTENT LINEAR EQUATIONS

by

R. P. Tewarson

MARCH 1, 1968

This research was supported by the National Aeronautics and Space Administration, Washington, D. C., Grant No. NGR-33-015-013.

ON THE CHEBYSHEV SOLUTION OF INCONSISTENT LINEAR EQUATIONS

R. P. Tewarson*

Abstract.

A new formulation of the ascent algorithm for the Chebyshev solution of linear systems is given. This leads to two algorithms, which are similar to the ordinary simplex and the product form of inverse algorithms for the solution of linear programming problems.

1. Introduction.

Let us consider the following system:

$$Ax-b = r(x), \tag{1.1}$$

where A is an m x n matrix with $m > n$, x is an n x 1 vector, b and r(x) are m x 1 vectors. Let the i^{th} element of r(x) be denoted by $r_i(x)$ and the n dimensional Euclidean space by E^n. In this paper, we shall consider the following problem:

$$\min_x \max_i |r_i(x)|, \; i = 1, \ldots, m; \; x \in E^n \tag{1.2}$$

Cheney [3, Chap. 2] gives an excellent discussion of the above problem and also several algorithms for its solution. One of them he calls the ascent algorithm, which is due to Stiefel [7, 8, 9]. We shall give a reformulation of the ascent algorithm. This will make it possible to express the algorithm in the well known tableau forms associated with the solution of linear programming (LP) problems, when using the ordinary [2, Chap. 5] and the revised simplex (product form of inverse) methods [5, p. 200]. The principal advantages of the two algorithms, which result from the reformulation of the ascent algorithm, are as follows: First, the algorithms require somewhat less

*State University of New York at Stony Brook, N. Y. This research was supported by the National Aeronautics and Space Administration, Washington, D.C., Grant No. NGR-33-015-013.

computational work than the ascent algorithm. Second, by making use of the algorithms, linear programming codes, e.g. [4, 5], can be easily adapted for the solution of (1.2) with changes in only few of the pertinent subroutines. Third, the algorithms give a clear and easy representation of the method of solution, especially if the reader is familiar with the solution of LP problems. Finally, one of the algorithms makes it possible to minimize the round-off errors by controlling the pivot choice, and if A is sparse, minimize also the growth of non-zero elements during the computational process.

2. Reformulation of the ascent algorithm.

Let $J = \{i_1, \ldots, i_{n+1}\} \in \{1, \ldots, m\}$. Then, according to [3, p. 36] every solution of (1.2), for appropriate sets of indices J, is a solution of

$$\min_x \max_i |r_i(x)|, \; x \in E^n, \; i \in J. \tag{2.1}$$

Without loss of generality, we can assume that $J = \{1, \ldots, n+1\}$. Now, according to de La Vallée Poussin [3, p. 37], x in (2.1) is the solution of

$$r_i(x) = \sigma_i z, \; i \in J, \; \sigma_i = \pm 1 \tag{2.2}$$

and

$$0 = K(\sigma_1 A_1, \ldots, \sigma_{n+1} A_{n+1}), \tag{2.3}$$

where A_i denotes the i^{th} row of A and the right-hand side of (2.3) denotes the convex hull of the vectors $\sigma_1 A_1, \ldots, \sigma_{n+1} A_{n+1}$. In this paper, we shall assume that the rows of A satisfy the Haar condition viz., every subset of n rows of A is a non-singular matrix. It is possible to relax this condition easily [3, p. 51]. In view of the Haar condition and the fact that in E^n, n+1 vectors have to be linearly dependent, we have

$$\sum_{i=1}^{n+1} \lambda_i A_i = 0, \tag{2.4}$$

where $\lambda_i \neq 0$, $i \in J$. For later use, let us impose the normalization

$$\sum_{i=1}^{n+1} \lambda_i b_i = -1 \tag{2.5}$$

on λ_i; b_i denotes the i^{th} element of b. If we define

$$\operatorname{sgn} \lambda_i = 1, \quad \lambda_i > 0$$
$$\quad\;\; = -1, \quad \lambda_i < 0 \tag{2.6}$$

then we can take $\sigma_i = \operatorname{sgn} \lambda_i$ and from (2.4) we have

$$\sum_{i=1}^{n+1} |\lambda_i| \sigma_i A_i = 0. \tag{2.7}$$

The above equation shows that condition (2.3) is satisfied.

Let us compute the values of the λ_is. From (2.4) and (2.5), we have

$$\begin{bmatrix} A_1^T & \ldots & A_{n+1}^T \\ \\ \\ b_1 & \ldots & b_{n+1} \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \cdot \\ \cdot \\ \cdot \\ \lambda_{n+1} \end{bmatrix} = \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ -1 \end{bmatrix}, \tag{2.8}$$

where A_i^T denotes the transpose of A_i. Now, if we let

$$B = \begin{bmatrix} A_1^T & \cdot\;\cdot\;\cdot & A_{n+1}^T \\ \\ b_1 & \cdot\;\cdot\;\cdot & b_{n+1} \end{bmatrix}, \tag{2.9}$$

$$\begin{bmatrix} \lambda_1, & \ldots, & \lambda_{n+1} \end{bmatrix}^T = \lambda \text{ and } [0, \ldots, 0, 1]^T = e_{n+1} \tag{2.10}$$

then (2.8) can be written as

$$B\lambda = -e_{n+1}. \tag{2.11}$$

In view of (1.1), the vector $[b_1, \ldots, b_{n+1}]$ does not lie in the row space of the matrix $[A_1^T, \ldots, A_{n+1}^T]$, and therefore B is non-singular and (2.11) gives

$$\lambda = B^{-1}(-e_{n+1}). \tag{2.12}$$

To find z we proceed as follows: From (2.2), we have

$$A_i x - b_i = z\,\sigma_i, \; i \in J. \tag{2.13}$$

Multiplying the above equation by λ_i and summing over all $i \in J$, we get

$$\sum_{i=1}^{n+1} \lambda_i A_i x - \sum_{i=1}^{n+1} \lambda_i b_i = z \sum_{i=1}^{n+1} \lambda_i \sigma_i ,$$

which in view of (2.4) and (2.5) yields

$$z = \frac{1}{\sum_{i=1}^{n+1} |\lambda_i|} . \tag{2.14}$$

We are now in a position to solve equation (2.13) for x. We can write it as

$$\begin{bmatrix} A_1 & , & b_1 \\ & \cdot & \\ & \cdot & \\ & \cdot & \\ A_{n+1} & , & b_{n+1} \end{bmatrix} \begin{bmatrix} x \\ -1 \end{bmatrix} = z\,\sigma , \tag{2.15}$$

where $\sigma^T = [\sigma_1 , \ldots, \sigma_{n+1}]^T$. In view of (2.9) the above equation can be written as

$$B^T \begin{bmatrix} x \\ -1 \end{bmatrix} = z\sigma, \tag{2.16}$$

or

$$\begin{bmatrix} x \\ -1 \end{bmatrix} = z \left[B^T\right]^{-1} \sigma . \tag{2.17}$$

Now, x in (2.17) is the solution of (1.2) if

$$|r_j(x)| < z, \text{ for all } i \notin J.$$

Let us compute $r_j(x)$. Using (1.1) we have

$$\begin{aligned} r_j(x) &= A_j x - b_j , \quad j \notin J, \\ &= [A_j, b_j] \begin{bmatrix} x \\ -1 \end{bmatrix} \\ &= [A_j, b_j]\, z\, [B^T]^{-1} \sigma, \text{ using (2.17).} \end{aligned}$$

Since $r_j(x)$ is a scalar, the above equation can be written as

$$r_j(x) = z\,\sigma^T B^{-1}\begin{bmatrix} A_j^T \\ b_j \end{bmatrix},\ j \notin J . \tag{2.18}$$

If

$$\mu_j = B^{-1}\begin{bmatrix} A_j^T \\ b_j \end{bmatrix}, \tag{2.19}$$

then (2.18) becomes

$$r_j(x) = z\,\sigma^T \mu_j,\quad j \notin J. \tag{2.20}$$

In case $|r_j(x)| > z$ for some $j \notin J$, then x, given by (2.17), is not the solution of (1.2). In this case, let

$$z < |r_\alpha(x)| = \max_j |r_j(x)|,\ j \notin J. \tag{2.21}$$

The set of rows $[A_i, b_i]$, $i \in J$ is called a reference set (RS). We shall introduce row $[A_\alpha, b_\alpha]$ and remove the row $[A_\beta, b_\beta]$ from the RS, thus constructing a new RS. The index β is given by

$$\max_i \frac{\mu_{i\alpha}\ \mathrm{sgn}\ r_\alpha(x)}{\lambda_i} = \frac{\mu_{\beta\alpha}\ \mathrm{sgn}\ r_\alpha(x)}{\lambda_\beta}, \tag{2.22}$$

where $\mu_{i\alpha}$ denotes the ith element of μ_α. The reason for the above choice is as follows: From (2.19), we have

$$\begin{bmatrix} A_\alpha^T \\ b_\alpha \end{bmatrix} = B\mu_\alpha = \sum_{i=1}^{n+1} \mu_{i\alpha} \begin{bmatrix} A_i^T \\ b_i \end{bmatrix},$$

which gives

$$A_\alpha = \sum_{i=1}^{n+1} \mu_{i\alpha} A_i . \tag{2.23}$$

Denoting sgn $r_\alpha(x)$ by σ_α and multiplying (2.23) by $\sigma_\alpha\,\gamma$ and adding the result to (2.4), we get

$$\gamma\,\sigma_\alpha A_\alpha + \sum_{i=1}^{n+1} [\lambda_i \sigma_i - \gamma\,\sigma_\alpha\,\sigma_i\,\mu_{i\alpha}]\sigma_i A_i = 0 . \tag{2.24}$$

As in (2.7), zero must lie in the convex hull of the new RS, therefore in equation (2.24) we have

$$\lambda_i \sigma_i > \gamma \, \sigma_\alpha \sigma_i \mu_{i\alpha} \text{ and } \gamma > 0 .$$

Since $\lambda_i \sigma_i = |\lambda_i| > 0$, we have

$$\frac{1}{\gamma} \geq \frac{\sigma_\alpha \sigma_i \mu_{i\alpha}}{\lambda_i \sigma_i} = \frac{\sigma_\alpha \mu_{i\alpha}}{\lambda_i} .$$

Therefore, if we take

$$\frac{1}{\gamma} = \max_i \frac{\sigma_\alpha \mu_{i\alpha}}{\lambda_i} = \frac{\sigma_\alpha \mu_{\beta\alpha}}{\lambda_\beta} ,$$

then in (2.24), the coefficient of $\sigma_\beta A_\beta$ will be zero and the coefficients of all the other $\sigma_i A_i$; $i = 1, \ldots, \beta-1, \beta+1, \ldots, n+1, \alpha$ will be positive. Thus the set $A_1, \ldots, A_{\beta-1}, A_{\beta+1}, \ldots, A_{n+1}, A_\alpha$ is the new RS. Evidently the new x will be a Chebyshev solution of an equation analogus to (2.2). It is easy to show that the new z will be greater than the current z [3, p. 47].

3. First algorithm - updating the whole tableau (UWT).

Let $\hat{B}$ denote the matrix obtained from B by replacing the βth column by $[A_\alpha, b_\alpha]^T$. Then it can be easily shown [10, 2 (p. 197)] that

$$\hat{B}^{-1} = \eta B^{-1} , \tag{3.1}$$

where

$$\eta = \begin{bmatrix} 1 & & & -\frac{\mu_{1\alpha}}{\mu_{\beta\alpha}} & & & \\ & \cdot & & \cdot & & & \\ & & \cdot & \cdot & & & \\ & & 1 & \cdot & & & \\ & & & \frac{1}{\mu_{\beta\alpha}} & & & \\ & & & \cdot & 1 & & \\ & & & \cdot & & \cdot & \\ & & & \cdot & & & \cdot \\ & & & -\frac{\mu_{n+1,\alpha}}{\mu_{\beta\alpha}} & & & 1 \end{bmatrix} . \tag{3.2}$$

Notice that the columns of $\hat{B}$ constitute a new RS. In order to continue with the algorithm outlined in secton 2, we need λ' and μ_j' (the values of λ and μ_j) for the new RS. From (2.12), we have

$$\lambda' = \hat{B}^{-1}\,[-e_{n+1}]$$

$$= \eta\, B^{-1}\,[-e_{n+1}], \text{ using (3.1),}$$

or

$$\lambda' = \eta\,\lambda\,. \tag{3.3}$$

Using (3.2) in the above equation, we get

$$\lambda_i' = \lambda_i - \frac{\lambda_\beta \mu_{i\alpha}}{\mu_{\beta\alpha}}\,,\ i \neq \beta \text{ and } \lambda_\beta' = \frac{\lambda_\beta}{\mu_{\beta\alpha}}\,. \tag{3.4}$$

Once we know λ', we can evaluate z' (the z for the new RS) from (2.14). Furthermore, as in (2.19), associated with the new RS, we have

$$\mu_j' = \hat{\beta}^{-1}\begin{bmatrix} A_j^T \\ b_j \end{bmatrix},\quad j \notin \text{New RS}$$

$$= \eta\, B^{-1}\begin{bmatrix} A_j^T \\ b_j \end{bmatrix}, \text{ using (3.1),}$$

or

$$\mu_j' = \eta\,\mu_j\,. \tag{3.5}$$

Once again, using (3.2) in the above equation, we get

$$\mu_{ij}' = \mu_{ij} - \frac{\mu_{\beta j}}{\mu_{\beta\alpha}}\,\mu_{i\alpha}\,,\ i \neq \beta \text{ and } \mu_{\beta j}' = \frac{\mu_{\beta j}}{\mu_{\beta\alpha}}\,. \tag{3.6}$$

Notice that $\mu_\beta = e_\beta$, since column β was in the old RS, and therefore, from (3.5) we have

$$\mu_\beta' = \eta\, e_\beta\,.$$

In view of (3.2), the above leads to

$$\mu'_{i\beta} = -\frac{\mu_{i\alpha}}{\mu_{\beta\alpha}}\ ,\ i \neq \beta \text{ and } \mu'_{\beta\beta} = \frac{1}{\mu_{\beta\alpha}}\ , \tag{3.7}$$

which is the β^{th} column of η.

We shall now describe the UWT algorithm. A numerical example will be given in section 5.

1. Apply the Gauss-Jordan elimination [6] to the matrix

$$\begin{bmatrix} A_1^T, & \ldots, & A_{n+1}^T, & A_{n+2}^T, & \ldots, & A_m^T, & \uparrow \\ & & & & & & -e_{n+1} \\ b_1, & \ldots, & b_{n+1}, & b_{n+2}, & \ldots, & b_m, & \downarrow \end{bmatrix}, \tag{3.8}$$

such that the RS is transformed to the identity matrix. This is equivalent to premultiplying (3.8) by B^{-1}. There is no loss of generality if we assume that the first n+1 columns in (3.8) constitute the initial RS. Hence we have

$$\begin{bmatrix} 1 & & & & \uparrow & & \uparrow & \lambda_1 \\ & \cdot & & & & & & \cdot \\ & & \cdot & & \mu_{n+2}, & \cdots & \mu_m, & \cdot \\ & & & \cdot & & & & \cdot \\ & & & 1 & \downarrow & & \downarrow & \lambda_{n+1} \end{bmatrix}. \tag{3.9}$$

If the columns in RS are chosen sequentially, but in order to minimize the round-off errors, the largest available element in each column is chosen as a pivot (partial pivoting [12]), then instead of the identity matrix I, a permutation matrix P will replace the RS in (3.9). The affect of this is described in the remarks following the algorithm.

2. Compute z using (2.14) and $r_j(x)$, $j \notin RS$ using (2.20). Determine $\max_j |r_j(x)| = |r_\alpha(x)|$, $j \notin RS$. If $|r_\alpha(x)| \leq z$, go to 4, otherwise go to 3.

3. Find the value of β by using (2.22). Transform column α of (3.9) into e_β (the βth column of the identity matrix) by means of elementary row operations. This is equivalent to the use of the formulas (3.4), (3.6) and (3.7). Now, go to 2.

4. Compute x for the final RS by performing the Gauss-Jordan or the Gaussian elimination [6] on

$$\begin{bmatrix} A_{i_1}, & b_{i_1} \\ \cdot & \\ \cdot & \\ \cdot & \\ A_{i_{n+1}}, & b_{i_{n+1}} \end{bmatrix} \begin{bmatrix} x \\ -1 \end{bmatrix} = z\,\sigma ,$$

where $\sigma^T = [\sigma_{i_1}, \ldots, \sigma_{i_{n+1}}]^T$ and $\{i_1, \ldots, i_{n+1}\}$ are the indicies associated with the final RS.

In using the above algorithm, we have to take the following facts into consideration. If the matrix B associated with the initial or any one of the subsequent reference sets is singular, then $Ax = b$ may have an exact solution. Also all the λ_is associated with each of the reference sets should be non-zero, otherwise the Haar condition will be violated. In (3.9), when using partial pivoting, instead of I we will get P, then in place of (3.9) we will have $[P, P\mu_j, P\lambda]$ and evidently the algorithm remains unchanged. The set of indicies $\{i_1, \ldots, i_{n+1}\}$ used in step 4 of algorithm, is determined by the permutation matrix which is present in the final updated tableau. The index i_k; $k=1, \ldots, n+1$, being that column of $[A,b]^T$ which has been transformed to e_k, the kth column of I.

The UWT algorithm described above is specially advantageous if m is not much greater than n. For $m >> n$, the algorithm given in section 4 is somewhat better. We can now point out the similarity of the UWT algorithm to the ordinary simplex method in linear programming [2, Ch. 5]. The similarities are between λ and the basic solution, z and the objective function, RS and the basic column vectors, μ_js and the coordinates of the non-basic vectors in terms of the basis, the updating rules. Of course, there are differences, viz., computation of z, computation of β, choice of initial RS etc. These differences can be used to make changes to the subroutines of linear programming codes to adapt them for the UWT algorithm.

4. Second algorithm-the product form of inverse (PFI).

As is well known [10, 11] the inverse of B can be expressed as the product of n+1 matrices of the type (3.2). Let

$$B^{-1} = \eta_{n+1} \cdots \eta_2 \eta_1 , \tag{4.1}$$

where η_i denotes the i^{th} matrix factor of B^{-1}. In view of (4.1) and (3.1), we have

$$\hat{B}^{-1} = \eta_{n+2} \cdots \eta_1 . \tag{4.2}$$

Only the elements of the non-trivial columns of each of the η_i along with the corresponding pivot row index β_i are stored. If the non-trivial column of η in (3.2) is denoted by $(\omega_1 , \cdots , \omega_{n+1})^T$, and

$$(v_1' , \ldots , v_{n+1}') = (v_1 , \ldots , v_{n+1}) \eta, \tag{4.3}$$

then

$$\begin{aligned} v_i' &= v_i, \; i \neq \beta \\ \text{and} \quad v_\beta' &= \sum_{i=1}^{n+1} v_i \, \omega_i , \end{aligned} \tag{4.4}$$

on the other hand if

$$\begin{bmatrix} s_1' \\ \cdot \\ \cdot \\ \cdot \\ s_{n+1}' \end{bmatrix} = \eta \begin{bmatrix} s_1 \\ \cdot \\ \cdot \\ \cdot \\ s_{n+1} \end{bmatrix}$$

then

$$\begin{aligned} s_i' &= s_i + \omega_i \, s_\beta, \; i \neq \beta \\ s_\beta' &= \omega_\beta \, s_\beta . \end{aligned} \tag{4.5}$$

Now, from (2.12) and (4.1), we have

$$\lambda = \left[\eta_{n+1} \cdots \eta_1\right] \left[-e_{n+1}\right] . \tag{4.6}$$

Using the updating rule (4.5) in the above equation, λ can be easily evaluated. Now, from (2.18) and (4.1), we have

$$r_j(x) = z\,\sigma^T\,[\eta_{n+1} \cdots \eta_1] \begin{bmatrix} A_j^T \\ \hline b_j \end{bmatrix}. \tag{4.7}$$

The row vector

$$\pi \doteq z\,\sigma^T\,[\eta_{n+1} \cdots \eta_1] \tag{4.8}$$

can be easily evaluated by making use of the updating formulas (4.4). From (4.7) and (4.8), we get

$$r_j(x) = \pi \begin{bmatrix} A_j^T \\ b_j \end{bmatrix}. \tag{4.9}$$

From (2.19) and (4.1), we have

$$\mu_\alpha = [\eta_{n+1} \cdots \eta_1] \begin{bmatrix} A_\alpha^T \\ b_\alpha \end{bmatrix}, \tag{4.10}$$

which can be easily evaluated by means of updating formulas (4.5). Also, from (2.17), (4.1) and (4.8), we get

$$[x^T, -1] = z\,\sigma^T\,[\eta_{n+1} \cdots \eta_1] = \pi. \tag{4.11}$$

We are now in a position to describe the PFI algorithm.

1. Determine an initial RS and find the product form of inverse of B in the form (4.1) by making use of the Gauss-Jordan elimination [6], as shown in [10]. The various techniques for keeping the product form of inverse sparse, if A is sparse, and minimizing the round-off errors can be incorporated [10, 11] in the above inversion.

2. Compute λ from (4.6), using (4.5).

3. Compute the following: z from (2.14), π from (4.8) and $r_j(x)$ from (4.9), $j \notin RS$.

4. Find $\max_j |r_j(x)| = |r_\alpha(x)|$. If $|r_\alpha(x)| \leq z$, go to 6, otherwise, compute μ_α from (4.10), using (4.5), and determine β from (2.22).

5. Evaluate and store the non-trivial column of the new η_t according to (3.7). If there are too many η's, viz., $t >> n+1$, ~~then~~ reinvert the basis. If more accuracy at each step is desired, then always keep only $n+1$ η's. Thus $\eta_{\beta-1}, \ldots, \eta_1$ will remain same when column α of $[A, b]^T$ is replacing column β of B, but $\eta_{n+1}, \ldots, \eta_\beta$ will get modified. A practical method of doing this is described in [1]. Now, go to 2.

6. The first n elements of the current value of π constitute the solution vector x^T (see (4.11)).

Remark: The computation of λ in step 2 of the algorithm, initially, can be done along with the evaluation of B^{-1} (as in UWT algorithm). Also, later on, during the course of the algorithm if η_t is only being prefixed to the other etas, then λ' can be evaluated according to (3.3).

The similarities between the revised simplex algorithm (product form of inverse) and our PFI algorithm are clear, e.g., the pricing vector and π, the update rules (4.4) and (4.5), and of course the similarities mentioned following the UWT algorithm.

5. Numerical Example.

Let

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 6 & 6 & 7 \\ -1 & 2 & 2 \\ 0 & -3 & 0 \end{bmatrix} \text{ and } b = \begin{bmatrix} 2 \\ 1 \\ 1 \\ 5 \\ 29 \\ 3 \\ -4 \end{bmatrix}.$$

Therefore

$$\begin{array}{c} \quad [A,d]^T \qquad\qquad\qquad \lambda \\ \left[\begin{array}{ccccccc|c} 1 & 0 & 0 & 1 & 6 & -1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 6 & 2 & -3 & 0 \\ 0 & 0 & 1 & 1 & 7 & 2 & 0 & 0 \\ 2 & 1 & 1 & 5 & 29 & 3 & -4 & -1 \end{array}\right]. \end{array}$$

Taking the first 4 columns as the initial RS, as in (3.9), we get

$$\left[\begin{array}{ccccccccc} 1 & 0 & 0 & 0 & 2 & -2 & 1 & & 1 \\ 0 & 1 & 0 & 0 & 2 & 1 & -2 & & 1 \\ 0 & 0 & 1 & 0 & \enclose{circle}{3} & 1 & 1 & & 1 \\ 0 & 0 & 0 & 1 & 4 & 1 & -1 & & -1 \end{array}\right].$$

Hence $z = 1/4$, $r_j(x) = 1/4 \ (3, -1, 1)$, $\alpha = 5$, $|r_5(x)| = 3/4 > z$,

$\frac{\mu_{i\alpha}}{\lambda_i} \ \sigma_\alpha = (2, 2, 3, -4)$, $\beta = 3$ and the next tableau is

$$\left[\begin{array}{cccccccc} 1 & 0 & -2/3 & 0 & 0 & \enclose{circle}{-8/3} & 1/3 & 1/3 \\ 0 & 1 & -2/3 & 0 & 0 & 1/3 & -8/3 & 1/3 \\ 0 & 0 & 1/3 & 0 & 1 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & -4/3 & 1 & 0 & -1/3 & -7/3 & -7/3 \end{array}\right],$$

and $z = 3/10$, $r_j(x) = (1/10, -1/2, 1/3)$, $\alpha = 6$,

$|r_6(x)| = 1/2 > 3/10$ and

$\frac{\mu_{i\alpha}}{\lambda_i} \sigma_\alpha = (8, -1, -1, 1/7)$, $\beta = 1$ and we have

$$\left[\begin{array}{cccccccc} -3/8 & 0 & 1/4 & 0 & 0 & 1 & -1/8 & -1/8 \\ 1/8 & 1 & -3/4 & 0 & 0 & 0 & -21/8 & 3/8 \\ 1/8 & 0 & 1/4 & 0 & 1 & 0 & 3/8 & 3/8 \\ -1/8 & 0 & -5/4 & 1 & 0 & 0 & -19/8 & -19/8 \end{array}\right].$$

which gives $z = 4/13$, $r_j(x) = (3/13, 1/26, 1/13)$ and $\max_j |r_j(x)| = 3/13 < z$, hence the final RS has been determined. Evidently,

$\{i_1, i_2, i_3, i_4\} = \{6, 2, 5, 4\}$ and the required x is the solution of

$$\begin{bmatrix} -1 & 2 & 2 & 3 \\ 0 & 1 & 0 & 1 \\ 6 & 6 & 7 & 29 \\ 1 & 1 & 1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ -1 \end{bmatrix} = 4/13 \begin{bmatrix} -1 \\ 1 \\ 1 \\ -1 \end{bmatrix} .$$

The above gives $(x_1, x_2, x_3) = (29/13, 17/13, 15/13)$.

Dr. R. P. Tewarson
Associate Professor
State University of New York
Stony Brook, N. Y., 11790, U.S.A.

REFERENCES

1. G. B. Dantzig, R. P. Harvey and R. D. McKnight, Updating the Product Form of the Inverse for the Revised Simplex Method, University of California at Berkley, Report No. ORC 64-33(RR), December, 1964.

2. ____________, Linear Programming and Extensions, Princeton University Press, Princeton, N. J., 1963.

3. E. W. Cheney, Introduction to Approximation Theory, McGraw-Hill, New York, N. Y., 1966.

4. Honeywell EDP, Advanced Linear Programming System, DSI-275, Wellesley Hills, Mass., 1964.

5. IBM, LP 90 Reference Manual, Share Distr. Agency, No. 1377, White Plains, N. Y., 1962.

6. A. Ralston, A First Course in Numerical Analysis, McGraw-Hill Book Co., New York, N. Y., 1965, pp. 399-401.

7. E. L. Stiefel, Uber Diskrete und Linear Tchebyscheff Approximation, Numer. Math. Vol. 1, 1959, pp. 1-28.

8. ____________, Note on Jordan Elimination Linear Programming and Tchebyscheff Approximation, Numer. Math. Vol. 2, 1960, pp. 1-17.

9. ____________, An Introduction to Numerical Mathematics, Academic Press, New York, N. Y., 1963, pp. 44-50.

10. R. P. Tewarson, On the Product Form of Inverses of Sparse Matrices, SIAM Rev., Vol. 8, 1966, pp. 336-342.

11. ____________, On the Product Form of Inverses of Sparse Matrices and Graph Theory, SIAM Rev., Vol. 9, 1967, pp. 91-99.

12. J. H. Wilkinson, Rounding Errors in Algebraic Processes, Prentice-Hall, Englewood Cliffs, N. J., 1963, p. 97.