ECONOMETRIC INSTITUTE

AN EFFICIENT WAY OF PROGRAMMING EAVES' FIXED POINT ALGORITHM

R. JANSEN and A.S LOUTER

REPORT 7713/EM

ERASMUS UNIVERSITY ROTTERDAM, P.O. BOX 1738, ROTTERDAM, THE NETHERLANDS

AN EFFICIENT WAY OF PROGRAMMING EAVES' FIXED POINT ALGORITHM

by R. Jansen and A.S. Louter

1. ABSTRACT.

The recently developed fixed point algorithms have provided the possibility of applying advanced numerical computation procedures in a broad field of economics. In particular, the pioneer work of Scarf [4,5], has shown its importance in the computation of economic equilibria. Moreover, Eaves [3] has succeeded in constructing an algorithm based more or less on the same principles as used by Scarf but providing in general a more efficient way in computing fixed points. Therefore his algorithm may be preferred especially in its application to rather extensive problems.

This paper deals with the formulation of Eaves' method such that with a minimum of storage and computation time his algorithm may be prepared for the computer. The algorithm as described in [3] can substantially be improved by reformulating the replacement step and by the application of an efficient inverse routine developed by Bartels [1] on the pivot step.

A very concise introduction to fixed point algorithms is given in section 2, where the basic thoughts behind this methods are pictured, omitting for the sake of legibility the mathematical justifications. For those who are interested in the latter aspect we refer to the original publications.

In section 3 Eaves' method is considered with an explanation of the replacement step in section 3.1 and its reformulation in section 3.2. Sections 3.3 and 3.4 are dealing with the pivot step and the labeling respectively, while in section 4 a complete FORTRAN computer program of the algorithm is given.

Aug. 1977

CONTENTS.

2. INTRODUCTION TO FIXED POINT ALGORITHMS.

Since the algorithms of Scarf and Eaves are developed to approximate fixed points we first summarize two important fixed point theorems, which are often used to prove the existence of a general equilibrium solution in economic models.

Th.1 Brouwer: If S is a non-empty-compact, convex subset of $\mathbb{R}^n$ and if f is a continuous function from S to S, then f has a fixed point, i.e. there is a $x^* \in S$ such that $f(x^*) = x^*$. (fig. 1A).

Th.2 Kakutani If S is a non-empty-compact, convex subset of $\mathbb{R}^n$ and if ϕ is an upper semicontinuous correspondence from S to S such that for all $x \in S$ the set $\phi(x)$ is convex (non-empty), then ϕ has a fixed point, i.e. there is a $x^* \in S$ such that $x^* \in \phi(x^*)$. (fig. 1B).

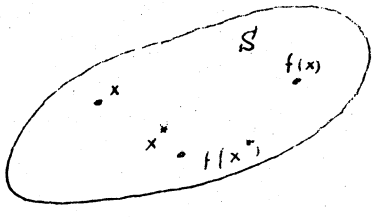


fig. 1A

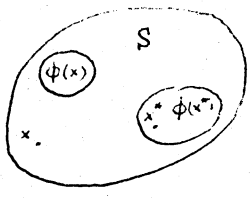


fig. 1B

The algotihms of Scarf and Eaves are both dealing with unit simplices; defined as the collection of vectors $x = \{x_1,\ldots,x_n\}$ with $x_i \geq 0$ for all i, and $\sum_{i=1}^{n} x_i = 1$ (fig. 1C).

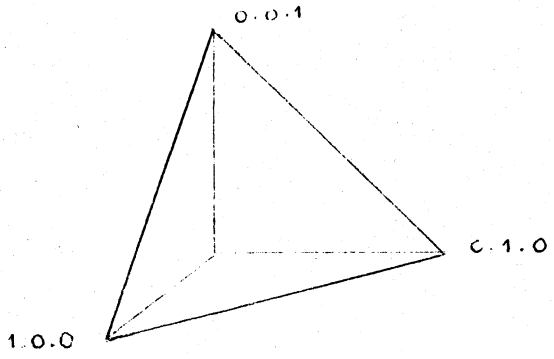


fig. 1C

On the simplex we define a collection of points, which play an important rôle in the search procedure in finding an approximate fixed point.

Def.1 Regular-grid. A regular grid defined on the simplex S is the collection of points x in S of the form $x = \frac{1}{N}(m_1, m_2, \ldots, m_n)$ with m_i non-negative integers summing to N. (fig.2a; n = 3, N = 10)

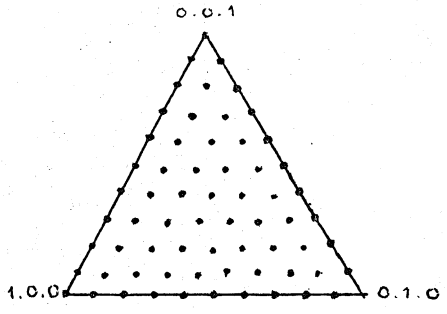


fig. 2a

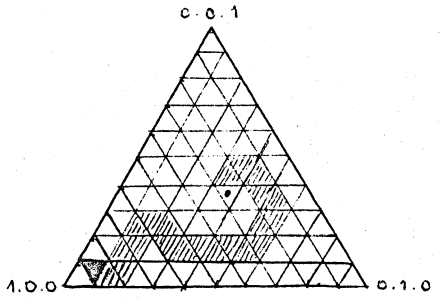


fig. 2b

If we connect these points as in fig.2b the simplex S is divided by this triangulation in regular subsimplices, which are conformable to the definition of primitive sets introduced by Scarf each including n-vectors of the regular grid. His idea in finding the fixed point is a walk through the simplex, starting in a vertex of the simplex passing from one primitive set to an other as illustrated in fig. 2.

The algorithm of Scarf is completely described by the following procedures.

a. a replacement step on primitive sets

b. a proper labeling of the vectors in each primitive set

ad a. The replacement step describes how to find a new primitive set if we eliminate one vector from the original primitive set. In case of a regular grid this procedure turns out to be very simple.

Def.2 Replacement step. If we put the vertices of the primitive set in a matrix and order them in a lexicographically increasing [1] way and suppose that the j-th column is eliminated, the new vector to be introduced must be calculated as follows.

$$x'^{1} = x^{n} + x^{2} - x^{1};\ j = 1$$
$$x'^{j} = x^{j-1} + x^{j+1} - x^{j};\ j = 2, \ldots, n-1$$
$$x'^{n} = x^{n-1} + x^{1} - x^{n};\ j = n$$

Ex. 1 If we consider the following primitive set

$$\frac{1}{10}\begin{bmatrix} 3 & 3 & 4 \\ 5 & 6 & 5 \\ 2 & 1 & 1 \end{bmatrix}$$

we may form a new primitive set by replacing the second vector; yielding

$$\frac{1}{10}\begin{bmatrix} 3 & 4 & 4 \\ 5 & 4 & 5 \\ 2 & 2 & 1 \end{bmatrix}$$

or by replacing the last vector:

$$\frac{1}{10}\begin{bmatrix} 3 & 3 & 2 \\ 5 & 6 & 6 \\ 2 & 1 & 2 \end{bmatrix}$$

or rearranged

$$\begin{bmatrix} 2 & 3 & 3 \\ 6 & 5 & 6 \\ 2 & 2 & 1 \end{bmatrix}$$

1) The vector $a = (a_1, a_2, \ldots, a_n)$ is lexicographically larger than $b = (b_1, b_2, \ldots, b_n)$ if the first nonzero element in the sequence $a_1 - b_1,\ a_2 - b_2,\ \ldots,\ a_n - b_n$ is positive.

ad b. A label of a primitive set vector is a characteristic of each vector in the primitive set, which makes it possible to select the vector to be eliminated, and assures the algorithm to converge to the neighbourhood of a fixed point.

We will elucidate this matter by means of an example of labeling described by Scarf:

Labeling: Suppose that we have a regular grid on our simplex with a certain grid-size. We associate a label with each vector x in the grid by the following rule:

Compute $f(x) - x = \begin{pmatrix} f_1(x) - x_1 \\ f_2(x) - x_2 \\ \cdot \\ \cdot \\ f_n(x) - x_n \end{pmatrix}$; if $f_j(x) - x_j$ is the

first non-negative element of $f(x) - x$, x is given the label j.

This labeling is chosen such that the fixed point is reached as soon as a primitive set is "completely labeled", which means that the vectors in that primitive set have all different labels.

Ex.2. We shall try to find the fixed point in the following simple problem

$$f\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ x_1 \end{pmatrix} \; ; \; x_i \geq 0, \; i = 1,2; \; \sum_i x_i = 1$$

We define a regular grid on our simplex with a grid-size of ten, and start the algorithm with a primitive set consisting of a vertex of the simplex and a point in the grid, which is nearest to this vertex. So the initial primitive set is

$\frac{1}{10}\begin{bmatrix} 9 & 10 \\ 1 & 0 \end{bmatrix}$. The labels of the vectors $\begin{pmatrix} 9 \\ 1 \end{pmatrix}$ and $\begin{pmatrix} 10 \\ 0 \end{pmatrix}$ are computed by the described procedure, yielding for both vectors the label 2.

In the initial position of the algorithm the first vector in the primitive set to the replaced is the side vector. So by applying the replacement step we find the next primitive set.

$\begin{bmatrix} 8 & 9 \\ 2 & 1 \end{bmatrix}$. Since the new vector $\begin{pmatrix} 8 \\ 2 \end{pmatrix}$ has also label 2 we replace the second vector, yielding: $\begin{bmatrix} 7 & 8 \\ 3 & 2 \end{bmatrix}$. The subsequent steps are: $\begin{bmatrix} 6 & 7 \\ 4 & 3 \end{bmatrix}$ and finally $\begin{bmatrix} 5 & 6 \\ 5 & 4 \end{bmatrix}$. The labeling

is now complete and the fixed point will be in the neighbourhood of this primitive set, which is correct since $x_1 = 5 = x_2$ is the fixed point. Scarf pointed out that it would be more efficient in general to associate vector labels with the primitive set vectors instead of integer labels. In a crucial theorem he stated that the fixed point is approximated with a primitive set of which the corresponding associated label vectors form a feasible basis [1] for a system of linear equalities $Ay = b$, with A the matrix of associated (label) vectors and b some non-negative vector. For the associated vectors he chose $f(x^i) - x^i + \iota$, with x^i the i-the primitive set vector, not on the boundary of the simplex, and $\iota = (1,1,\ldots,1)'$ and b is defined to be ι.

For the primitive set vectors representing points on the boundary of the simplex unit vectors are associated by the following rule: Each primitive set vector with the first zero element on the i-th place will be associated with the i-th unit vector. The procedure to be followed in finding the fixed point can be described as follows.

The aim of the algorithm is to find a primitive set of which the corresponding label vectors form a feasible basis for the equations $Ay = \iota$. In order to reach this correspondence we start the algorithm with a feasible basis for $Ay = \iota$ formed by $n - 1$ label vectors corresponding with $n - 1$ vectors in the initial primitive set and one additional label vector generally not corresponding with the remaining primitive set vector. During the algorithm we follow a path through the simplex running from one primitive set to another such that the associated matrices (consisting throughout of $n - 1$ corresponding label vectors and one additional vector) form a feasible basis for $Ay = \iota$. As soon as we have found a primitive set completely corresponding with the associated feasible basis the algorithm is terminated, since then the fixed point is reached.

1) The columns $j_1, \ldots, j_n$ of the matrix A form a feasible basis if the equations $Ay = b$ have a unique, non-negative solution with $y_j = 0$ unless $j = j_1, \ldots, j_n$.

In order to elucidate the procedure with vector labels we follow the algorithm in a number of subsequent steps. The initial primitive set is formed near a vertex of the simplex (say near the vertex $\begin{pmatrix} 1 \\ 0 \\ \cdot \\ \cdot \\ 0 \end{pmatrix}$ and consists of n - 1 side vectors and one vector interior to the simplex (in this case the vector in the grid with the largest first coordinate). In the 3-dimensional case with a grid-size of 10 we have:

$$\begin{bmatrix} 8 & 9 & 9 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}.$$

The initial associated matrix is formed completely by unit vectors, since this matrix forms a feasible basis for $Ay = \iota$. In doing so the associated matrix corresponds with the primitive set except for the vector interior to the simplex. (see the labeling rule for side vectors).

The initial position of the algorithm may then be described as follows:

Primitive set consisting of vectors $x^j, x^2, \ldots, x^n$

Associated matrix consisting of n unit vectors $a^1, \ldots, a^n$

In the primitive set x^j is the interior vector and $x^2, \ldots, x^n$ the n - 1 side vectors. The associated label vectors of the primitive set vectors are recognized by the same indices in this notation.

In order to reach complete correspondence between the primitive set and the associated matrix we introduce the label vector of x^j in the associated matrix by eliminating one of the unit vectors in this matrix such that the new associated matrix again form a feasible basis for $Ay = \iota$. This operation is carried out by means of a pivot step (familiar to the pivot step known from linear programming, discussed in detail in section 3.3).

If the vector to be eliminated turns out to be the first unit vector the algorithm in terminated, since correspondence is reached. If another vector is eliminated (say the i-th one) we are in the next stage:

Primitive set	$x^j, x^2, \ldots, \ldots, x^n$
Associated matrix	$a^j, a^2, \ldots, a^{i-1}, a^{i+1}, \ldots, a^n$

We now replace the i-th primitive set vector by applying the replacement step from def. 2. In doing so another vector is introduced and if this is not a first side vector (= a vector with zero first coordinate) we insert its corresponding label vector in the feasible basis by means of a pivot step, and so on.

The algorithm of Scarf and Eaves are both based on the discribed procedure. Scarf, however, deals with a specific grid size and tries to reach the fixed point by starting in general near a vertex of the simplex. Since the accuracy of the approximation of the fixed point depends on the chosen grid-size this algorithm should be combined with same numerical optimization method in order to refine the last primitive set at the required accuracy-level. Moreover the number of iterations (replacement and pivot steps) might be substantial since the starting point may lie relatively far beyond the definite fixed point.

Eaves, therefore developed an algorithm using more or less the same procedures but without the disadvantages just described, at the price that his method is much more complicated than Scarf's method.

3. EAVES' METHOD.

The principle of Eaves' algorithm is based on a subsequent extension of the grid size during the algorithm.

The crucial idea behind this method is that knowing the final primitive set with respect to a certain grid size it would cost less effort in finding another primitive set, which is the best approximation of the fixed point in a grid size finer than the preceeding one. Therefore the original simplex is multiplied by 2^k, k= 0,1,2, ..., obtaining an infinite sequence of n-simplices[1] S_k. In fig. 3 this is illustrated in case n = 3.

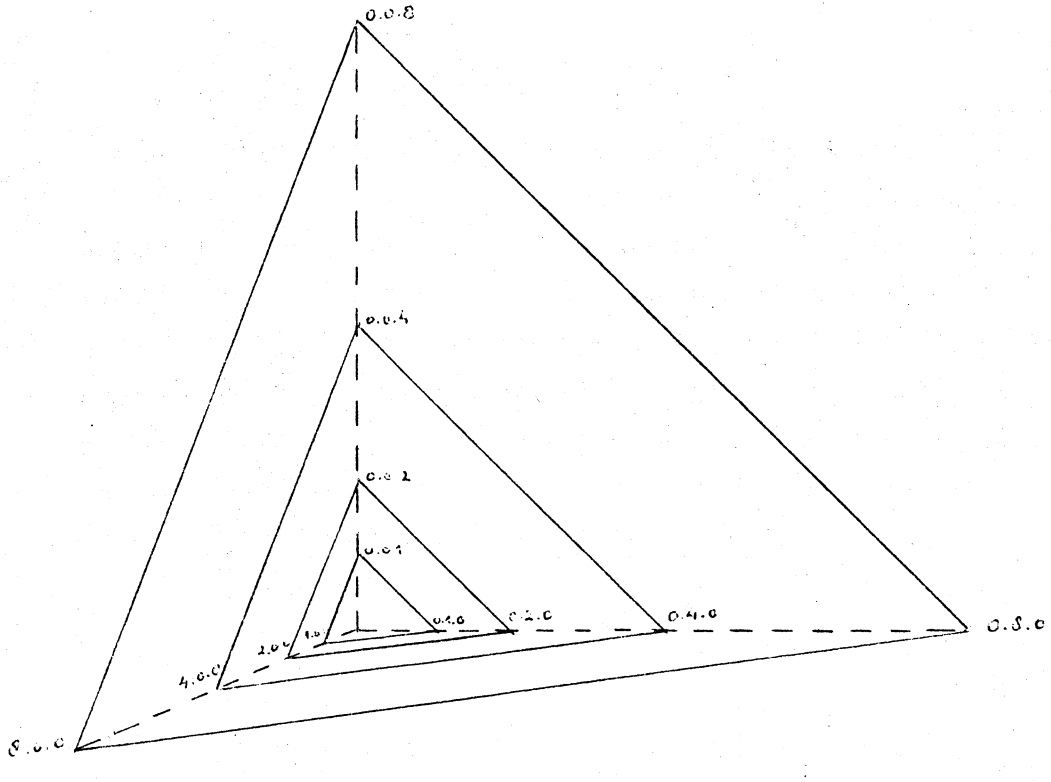


fig. 3

1) An n-simplex is defined to be any simplex having n-vertices.

On each simplex a regular grid is defined in the usual way. In S_k the grid consists of all vectors, that can be written as $(m_1,\ldots,m_n)$ with $m_1,\ldots, m_n$ non negative integers summing to 2^k. In doing so a triangulation of each n-simplex S_k in subsimplices is obtained as shown in fig. 4 for n = 3 and k = 2.

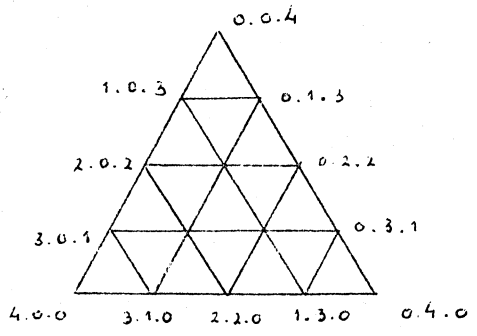


fig. 4

The algorithm is aimed at finding a path through the positive orthant of the n-dimensional space (denoted by D), running from primitive sets in S_k to primitive sets in S_{k+1} for abritrary k. We therefore have to extend the triangulation such that the spaces between two adjacent simplices are partitioned too. This extension is performed by dividing the space between S_k and S_{k+1} into (n+1)-subsimplices with vertices corresponding, to the vertices of the subsimplices (primitive sets) in S_k and S_{k+1}.

Eaves has defined such a triangulation on

$D' = \{x \in D; \Sigma x_i \leq 2\}$ in the following way:

If τ is a (n+1)-subsimplex in D' consisting of n + 1 vertices $v^i = (v^i_1,\ldots,v^i_n)$; i = 1,...n+1 which are ordered in a lexicographically decreasing way ($v^1 > v^2 > \ldots > v^{n+1}$) with v^i in S_o or S_1 for all i and let v^i be generated according to the following scheme:

$$v^{i+1} = v^i + q(\gamma_i) \; ; \; i = 1, \ldots, n$$

with $\gamma_1, \ldots, \gamma_n$ some permutation on 1, ..., n and q(j) the j-th column of the nxn-matrix;

$$\begin{bmatrix} -1 & 0 & \cdots & 0 & 0 \\ +1 & -1 & \cdots & 0 & \vdots \\ 0 & +1 & & \vdots & \vdots \\ \vdots & 0 & & \vdots & \vdots \\ \vdots & \vdots & & -1 & \vdots \\ 0 & 0 & \cdots & +1 & -1 \end{bmatrix}$$

such that the generated v is non negative or zero
then the collection M of all such τ in D' form a triangulation of D' and each τ is completely described by v^1 and γ and we may characterize τ therefore by $\tau(v^1,\gamma)$.
Furthermore we notice that any n-simplex σ in S_k may be characterized by $\sigma(u^1,\beta)$, where σ is generated from u^1 in S_k according to:

$$u^{i+1} = u^i + p(\beta_i) \; , \; i = 1, \ldots, n-1$$

with $\beta_1, \ldots, \beta_{n-1}$ same permutation on 1, ..., n-1 such that the generated u is non negative or zero and p(j) the j-th column of the n x (n-1)-matrix

$$\begin{bmatrix} -1 & \cdots & 0 \\ +1 & & \vdots \\ 0 & & \vdots \\ \vdots & & \vdots \\ \vdots & & -1 \\ 0 & \cdots & +1 \end{bmatrix}$$

The complete triangulation on D' is pictured in fig.5 for the 3-dimensional case.

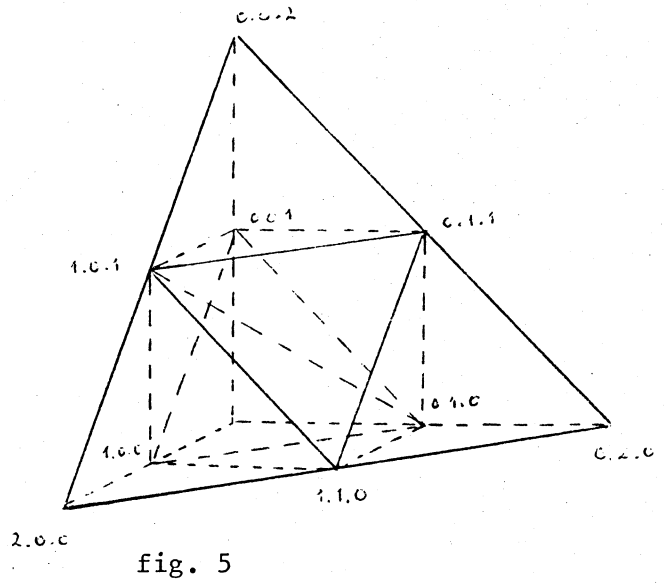


fig. 5

It appears that in this example 7 possible τ-simplices may be distinguished, which are summarized below with their corresponding γ-vector

Ex. 3

$$\tau_1 = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \gamma = (1,2,3)$$

$$\tau_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \gamma = (1,2,3)$$

$$\tau_3 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}, \quad \gamma = (2,1,3)$$

$$\tau_4 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \gamma = (2,3,1)$$

$$\tau_5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 2 & 1 \end{bmatrix}, \quad \gamma = (1,2,3)$$

$$\tau_6 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}, \quad \gamma = (1,3,2)$$

$$\tau_7 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, \quad \gamma = (3,1,2)$$

We are now prepared to extend the complete triangulation on D' to D. Namely if σ is any n-simplex in S_k, k = 0,1,2, ... and τ is any (n+1)-simplex in M then all (n+1)-simplices in D are of the form $T_\sigma(\tau)$, since if we have a $\sigma(u^1,\beta)$ and a $\tau(v^1,\gamma)$ we may find a (n+1)-simplex $T_\sigma(\tau)$ in D by:

$$(3.1)\quad T_\sigma(\tau) = (T_\sigma(v^1),\ldots,T_\sigma(v^{n+1})) =$$

$$= \left(\sum_{i=1}^{n} u^i v_i^1,\ldots,\sum_{i=1}^{n} u^i v_i^{n+1}\right)$$

Ex.4 For example if $\sigma = \begin{bmatrix} 4 & 3 & 3 \\ 2 & 3 & 2 \\ 2 & 2 & 3 \end{bmatrix}$ and $\tau = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

we obtain $T_\sigma(\tau) = \begin{bmatrix} 8 & 7 & 7 & 4 \\ 4 & 5 & 4 & 2 \\ 4 & 4 & 5 & 2 \end{bmatrix}$

3.1. Replacement step on D.

Eaves pointed out that a replacement step on the (n+1)-simplices in D may be carried out by a replacement step in D', since there exists a correspondence $T_\sigma(\tau)$ between all τ in D' and the (n+1)-simplices in D.

Eaves describes this replacement step in the following way:

Suppose that we find ourselves in a certain stage of the algorithm and that we are in the position to move from $T_\sigma(\tau)$ to $T_{\sigma'}(\tau')$ and suppose that the vector $T_\sigma(v^i)$ should be replaced.

We then firstly generate a vector $v = (v_1,\ldots,v_n)$ by means of the replacement step stated in Def.2 on $\tau = \{v^1,\ldots,v^{n+1}\}$, replacing v^i.

There are four cases that might occur.

(1) $v \in D'$

(2) $\Sigma v_i > 2$

(3) $v_j < 0$, for some $j = 1, \ldots, n$ and

(4) $v = 0$

We illustrate the possible occurrence of these four cases in the following example:

Ex.5.

a. Suppose that in τ_3 of Ex.3 v_3 is replaced, we then obtain by applying the replacement operation on v_3:

$$v = v_2 + v_4 - v_3 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad \text{, which refer to case (1)}$$

b. if we replace v_4 in τ_1 we obtain:

$$v = v_3 + v_1 - v_4 = \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} \quad \text{, which lead us to case (2)}$$

c. if we replace v_3 in τ_1 we are in case (3), since

$$v = \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix}$$

d. and if v_1 in τ_7 is replaced v becomes the zero vector.

Since $\sigma(u^1,\beta)$ and $\tau(v^1,\gamma)$ are known in any stage of the algorithm and since u^1,β and v^1,γ define σ and τ completely, Eaves described the procedure of the replacement step in the four cases in terms of changes in u^1,β and v^1,γ instead of considering the complete σ and τ.
Case (1) ($v \in D'$): (u^1,β) are unchanged and the new (v^1,γ) are computed according to table 1.

Table 1

index of the replaced vector	v^1 becomes	γ becomes
$i = 1$	$v^1 + q(\gamma_1)$	$(\gamma_2,\ldots,\gamma_n,\gamma_1)$
$2 \leq i \leq n$	v^1	$(\gamma_1,\ldots,\gamma_i,\gamma_{i-1},\ldots,\gamma_n)$
$i = n + 1$	$v^1 - q(\gamma_n)$	$(\gamma_n,\gamma_1,\ldots,\gamma_{n-1})$

Case (2) ($\Sigma v_i > 2$): u^1 becomes $\Theta_1 u^1 + \sum_{i=1}^{n-1} \Theta_{i+1} q(\beta_i)$ where

$$\Theta_i = \sum_{j=i}^{n} v_j^1 \quad , \quad i = 1, \ldots, n$$

β becomes $(\beta_{\gamma_1}, \ldots, \beta_{\gamma_{n-1}})$, v^1 becomes $(1,0,\ldots,0,1)$ and γ becomes $(n,1,\ldots,n-1)$.

Case (3) ($v_j < 0$): First we note that j is unique. The new (u^1, β) is computed according to table 2 where only u^j is replaced. The new (v^1, γ) is computed according to table 3 (the (i,j) combinations not listed can not occur)

Table 2

index of negative entry	u^1 becomes	β becomes
$j = 1$	$u^1 + q(\beta_1)$	$(\beta_2, \ldots, \beta_{n-1}, \beta_1)$
$1 < j < n$	u^1	$(\beta_1, \ldots, \beta_j, \beta_{j-1}, \ldots, \beta_{n-1})$
$j = n$	$u^1 - q(\beta_{n-1})$	$(\beta_{n-1}, \beta^1, \ldots, \beta_{n-2})$

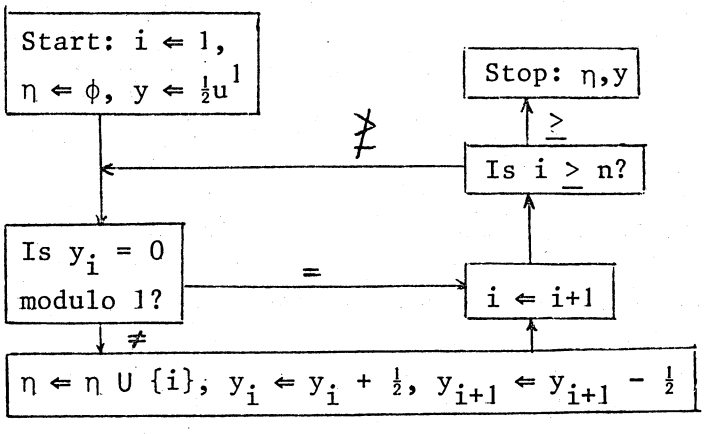
Table 3

index of negative entry j and index of replaced vector i	v becomes	γ becomes
$j = 1$ $i = 1$	$(v_2^2, v_3^2, \ldots, v_n^2, 0)$	$(\gamma_2 - 1, \ldots, \gamma_{k-1} - 1, n-1, \gamma_k,$ $\gamma_{k+1} - 1, \ldots, \gamma_n - 1)$ where $\gamma_k = n$
$1 < j < n$ $1 \leq i \leq n+1$	v^1	γ
$j = n$ $2 \leq i \leq n$	$(1, v_1^1 - 1, v_2^1, \ldots, v_{n-1}^1)$	$(1, \gamma_1 + 1, \ldots, \gamma_{i-2} + 1, n, \gamma_{i+1} + 1,$ $\ldots \gamma_n + 1)$

Case (4) ($v = 0$): There is a unique set $\eta \subseteq \{1, \ldots, n-1\}$ such that

$$y = \tfrac{1}{2}(u^1 - \sum_{i \eta} q(i)) \text{ has integer components}$$

The flow chart generates η and y; "$\Leftarrow$" means "becomes". Observe that components of $\frac{1}{2} u^1$ are 0 or $\frac{1}{2}$ modulo 1.



Let $\xi = \{1,\ldots,n-1\} \sim \eta$ and let $\eta = (\eta_1,\ldots,\eta_k)$ and

$\xi = (\xi_1,\ldots,\xi_{n-k-1})$ inherit order from $\beta = (\beta_1,\ldots,\beta_{n-1})$.

After computing the new (v^1,γ) according to table 4, the new (u^1,β) becomes $(y,(\eta,\xi))$.

Table 4

	v^1 becomes	γ becomes
$k = 0$	$(2,0,\ldots,0)$	$(1,\ldots,n)$
$k > 0$	$(z_o,\ldots,z_n)$ where $z_\ell = \begin{cases} 1 & \ell = 0,\ldots,k \\ 0 & \text{else} \end{cases}$	$(m_1,\ldots,m_{n-1},n)$ where $m_\ell = \begin{cases} h & \beta_\ell = \eta_h \\ k+h & \beta_\ell = \xi_h \end{cases}$

We will try to elucidate the various operations in the replacement step by means of the following examples for the 3-dimensional space

Ex.6

Suppose that in some stage of the algorithm σ, τ and $T_\sigma(\tau)$ are:

$$\sigma = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \qquad \tau = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad T_\sigma\,\tau = \begin{bmatrix} 2 & 2 & 1 & 1 \\ 2 & 1 & 2 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$\beta = (2,1)$ $\qquad$ $\gamma = (1,2,3)$

and suppose that we want to replace $T_\sigma(v^4)$.

We firstly compute v:

$v = v^3 + v^1 - v^4 = (2,0,1)$, which corresponds with case (2). We follow the scheme valid for this case:

$$\Theta_1 = 2 \rightarrow u^1 \Rightarrow 2u^1 = \begin{pmatrix} 2 \\ 2 \\ 0 \end{pmatrix} \text{ ; and } \beta \Rightarrow (2,1);$$

$$v^1 \Rightarrow \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \text{ ; and } \gamma \Rightarrow (3,1,2)$$

So the new σ, τ and $T_\sigma(\tau)$ become:

$$\sigma \qquad \tau \qquad T_\sigma(\tau)$$

$$\begin{bmatrix} 2 & 2 & 1 \\ 2 & 1 & 2 \\ 0 & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 3 & 2 & 2 & 1 \\ 4 & 2 & 1 & 2 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

$\beta = (2,1)$ $\qquad\qquad$ $\gamma = (3,1,2)$

We now want to replace $T_\sigma(v^3)$. Computing v yields:

$v = (1,-1,1)$; corresponding with case (3); $j = 2$, $i = 3$

From table 2 it is seen that u^1 remains unchanged while β becomes (1,2). Table 3 shows that v^1 and γ do not change. So the following situations has been achieved.

$$\sigma \qquad \tau \qquad T_\sigma(\tau)$$

$$\begin{bmatrix} 2 & 1 & 1 \\ 2 & 3 & 2 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 3 & 2 & 1 & 1 \\ 4 & 2 & 3 & 2 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Suppose $T_\sigma(v^1)$ has to be replaced. In that case $v = (0,0,0)$; which lead us to case (4).

Following the flow chart we find:

$i = 1$, $\eta = \phi$, $y = (1,1,0)$; so $y_1 = 0$ modulo 1 which holds for all y_j; $j = 2,3$ and therefore $\eta = \phi$ and $\xi = (1,2)$ computing v^1 and γ according to table 4; we have

$$v^1 \Rightarrow \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix} \quad \text{and } \gamma \Rightarrow (1,2,3)$$

$$\text{Finally } u^1 \Rightarrow \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix} \quad \text{and } \beta \Rightarrow (1,2)$$

Summarizing the new situation becomes:

$$\sigma \qquad\qquad \tau \qquad\qquad T_\sigma(\tau)$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 2 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 2 & 1 & 1 & 1 \\ 2 & 3 & 2 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$\beta = (1,2)$ $\qquad\qquad$ $\gamma = (1,2,3)$

The quite compact description of the replacement operation might unjustly suggest that the derivation of a new subsimplex cost extensively computational effort, since it seems to be necessary to store u^1, β, v^1 on γ at each step of the algorithm and moreover that in each step $T_\sigma(\tau)$ should be computed, according to its definition stated in eq. 3.1.

Since it is seen that a replacement step on $T_\sigma(\tau)$ influences only one vector in $T_\sigma(\tau)$ one might wonder if it is not possible to formulate the replacement step in such a way that the costly operations on σ en τ could be avoided. It is therefore that we have searched for a reformulation of the replacement step in which the calculations are pictured more directly on the subsimplices $T_\sigma(\tau)$ we are interested in.

3.2. Reformulation of the replacement step.

The reformulated replacement step is pictured in the flow chart in fig. 6.

The symbols used in this flow chart have the following meaning:

JMOUT = the index of the vector in $T_\sigma(\tau)$ to be replaced.

JIN = the index of the vector to be introduced in $T_{\sigma'}(\tau')$

T^i = the i-th vector of $T_\sigma(\tau)$

TIN = the vector to be introduced in $T_\sigma(\tau')$ after replacing T^{JMOUT} in $T_\sigma(\tau)$.

NRLH = the index of the last vector on the highest level.

The other symbols correspond with the notation used in the previous text.

The equivalence of the reformulated replacement step with the replacement step discussed in section 3.1 is proved in Appendix A.

From the flow chart in fig.6 it may be noticed that with the knowledge of γ and the structure of $T_\sigma(\tau)$ we are able to calculate the vector to be introduced in $T_{\sigma'}(\tau')$, by applying a number of simple tests.
In order to adapt γ in each step of the algorithm we follow Eaves' tables from section 3.1.
The cases 1,2,3 and 4 to which these tables correspond, are explicitely given in the flow chart.
When case (4) appears we have to apply table 4 in which is shown that in this case we must know u^1 and β. Fortunately, however, this problem can be solved immediately since the last n vectors of $T_\sigma(\tau)$ in this case are the vectors that form the σ.
Summarizing we may conclude that since only storage and adaptation is required with respect to γ the reformulated replacement step requires less information compared with the description of Eaves' replacement step.
And from a computational point of view important progress has been made by avoiding the extensive calculation of $T_{\sigma'}(\tau')$ through eq. 3.1.

3.3. Pivot step.

As already stated in section 2.the pivot step is used in case of vector labeling and may provide a procedure to find the vector

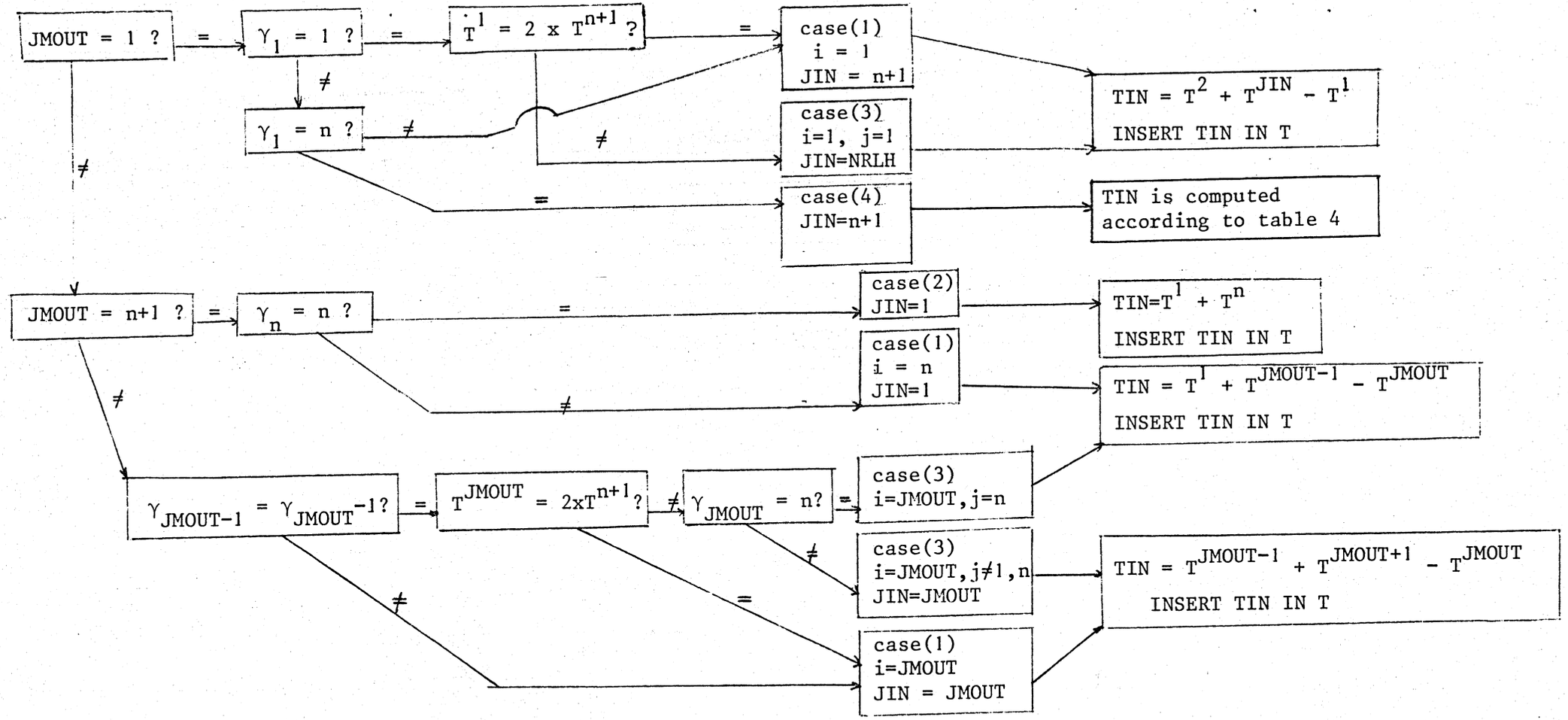


fig. 6

FLOW CHART OF THE

REPLACEMENT STEP

to be replaced in the primitive set. This procedure will be elucidated by means of the following example.

Suppose we are in a certain stage of the algorithm with a primitive set consisting of n + 1 vectors in which the j-th vector is just introduced by means of a replacement step. So the associated matrix A consists of vectors corresponding with the primitive set vectors except for the j-th primitive set vector. (a^1). With the notation according to page 6, we have:

primitive set	$x^i, x^2, \ldots, x^{n+i}$
associated matrix	$a^1, a^2, \ldots, a^{n+i}$

The matrix A is such that it fulfils Ay = 1, with y strictly positive.

Since the j-th primitive set vector is just introduced, the associated vector $(a^*)^j$ will be introduced in the A-matrix such that the new A-matrix (say A*) forms a feasible basis for $A^*_p = 1$.

We are now at the problem to select the vector in A to be eliminated. Therefore we write $(a^*)^j$ as a linear combination of the original associated vectors; so we have:

$$(3.3.1) \qquad a^1 z_1 + a^2 z_2 + \ldots a^k z_k \ldots + a^{n+1} z_{n+1} = (a^*)^j$$

Moreover from Ay = 1, we may write:

$$(3.3.2) \qquad a^1 y_1 + a^2 y_2 + \ldots + a^{n+1} y_{n+1} = 1$$

Reformulating 3.3.1 yields:

$$(3.3.3) \qquad -a^1 \frac{z_1}{z_k} - a^2 \frac{z_2}{z_k} - \ldots + \frac{(a^*)^j}{z_k} - \ldots - a^{n+1} \frac{z_{n+1}}{z_k} = a^k; \; z_k \neq 0$$

and substituting a^k in 3.3.2 yields

$$(3.3.4) \qquad a^1 y_1 (1 - \frac{z_1}{y_1} - \frac{y_k}{z_k}) + a^2 y_2 (1 - \frac{z_2}{y_2} \cdot \frac{y_k}{z_k}) + \ldots + (a^*)^j \frac{y_k}{z_k} + \ldots$$
$$+ a^{n+1} y_{n+1} (1 - \frac{z_{n+1}}{y_{n+1}} \cdot \frac{y_k}{z_k}) = 1$$

which must be equivalent with $A'p = 1$ if a^k is the vector to be eliminated from A in favour of $(a^*)^j$. Since all p_i should be strictly positive, the first thing to be noticed is:

(3.3.5) $$p_k = \frac{y_k}{z_k} > 0 \quad \text{. which implies}$$

(3.3.6) $$z_k > 0 \text{ , since } y_k > 0$$

and moreover:

(3.3.7) $$p_i = y_i(1 - \frac{z_i}{y_i}\frac{y_k}{z_k}) > 0; \quad i \neq k$$

or equivalent:

(3.3.8) $$\frac{z_i}{y_i} < \frac{z_k}{y_k} \quad ; \; i \neq k$$

3.3.6 and 3.3.8 may be fulfilled by selecting k such that $z_k > 0$ and

$$\frac{z_k}{y_k} = \max_m(\frac{z_m}{y_m}) \quad ; \; m = 1, \ldots, n+1$$

As a result of linear programming such a k can in general always be found. If it appeared not to be unique a criteria must be formulated such that the pivot step is carried out consequently during the algorithm.

Since there is no need in reordering the A-matrix after any pivot step this may be accomplished easily by selecting the first k for which $\frac{z_k}{y_k} = \max_m(\frac{z_m}{y_m})$.

It is noticed that in finding the pivot column we have to invert A in order to compute $y = A^{-1}1$ and $z = A^{-1}(a^*)^j$. Dohmen and Schoeber [2] suggested in their paper that by applying a result of Bartels [1] a procedure may be carried out avoiding the

computation of the complete A^{-1} at each pivot step.

Bartels pointed out that if A^{-1} is known and the k-th column of A is replaced by $(a^*)^k$. $(A^*)^{-1}$ may be computed by Gauss-Jordan elimination as $(A^*)^{-1} = DA^{-1}$ with

$$D = I - \frac{1}{z_k}(z - e^{(k)})e^{(k)^T} \quad ; \text{ in which } e^{(k)} \text{ is k-th unit vector.}$$

In avoiding too large round-off errors it is recommendable to invert the associated matrix after a number of iterations completely.

3.4. Labeling.

In section 2 some examples of labeling were given for the Scarf method and with respect to Eaves' method such a labeling can also be formulated. What we need is a so-called proper labeling defined by Eaves as:

Def.3 A proper labeling of vectors in the original unit simplex S must be such that:

a) the set of vertices of S is completely labeled which means in our context that if we are dealing with vector labels, the associated vectors of the vertices of S form a feasible basis for $Ay = 1$.

b) no facet of S contains a completely labeled set

c) if a sequence of complete sets tends to $x \in S$, then x is a fixed point of f.

In our computer program in section 4 we used the following labels (= associated vectors), which in all problems we examined turned out to be sufficient: [1)]

$$a^j = f(x^j) - x^j + 1 \quad \text{if } x^j > 0$$

and if x^j_{k+1} is the first non-zero element $k+1 \leq n$ in x^j after the first zero element in x^j then;

a^j = the k-th unit vector or

a^j = the n-th unit vector if x^j_n appears to be still zero starting from the first zero element in x^j.

This labeling has the advantage that we may start the algorithm with I_n as associated matrix and that during the algorithm associated

1) A formal proof of the sufficiency of this labeling will not be given here.

vectors corresponding with side-vectors (slack vectors) can be introduced in the associated matrix avoiding the costly pivot operation; which may be illustrated by the description of the computer program in the next section.

4. A FORTRAN COMPUTER PROGRAM FOR EAVES' METHOD

LIST OF THE MAIN CHARACTERS

PRIMITIVE SET

ITMAT (I,J) = Matrix $T_\sigma(\tau)$ in Eaves or T in flow chart of fig. 6
NP 1 = Number of rows in ITMAT
NP 2 = Number of columns in ITMAT
JMOUT = Column index of the vector in ITMAT to be replaced
ITM(I) = The vector to be introduced in ITMAT after a replacement step (= TIN in fig. 6)
JIN = Column index of the vector ITM in ITMAT
ITNUM(I) = Vector of numbers joined to the primitive set vectors in order to visualize the correspondence with the label vectors in the associated matrix

ASSOCIATED MATRIX OF LABEL VECTORS

AINV(I,J) = The inverse of the associated matrix of label vectors
AIN(I) = The label vector to be introduced in the associated matrix after a replacement step
JAOUT = The index of the pivot column in the associated matrix
PHULP(I) = AINV.AIN
W(I) = Vector with unit elements (in Ay=ı, see section 2)
QHULP = Element of AINV.W
CRITER = Small real number introduced to correct for round-off errors
IANUM(I) = Vector of numbers joined to the label vectors in order to visualize the correspondence with ITNUM(I)
IAACT(I) = Vector of numbers joined to the unit vectors in the associated matrix

OTHER IMPORTANT CHARACTERS

IGAM(I) = γ according to Eaves and fig. 6
IBETA(I) = β according to Eaves
N = Order of IBETA
KSLEK = k standing for the k-th slack vector
ITRE = Number of iterations
INVE = Number of inversions on the associated matrix
KDIM = Reserved storage capacity with respect to the number of rows in ITMAT

The examples are carried out on a DEC-10 computer system.

```
C*************************************************************
C
C  MAIN PROGRAM WHICH USES EAVES' PROCEDURE FOR FINDING
C  THE FIXED POINT OF F(X)
C
C  IN THE EXAMPLE F(X) IS DEFINED AS:
C       F(X(I)) = X(I+1)  FOR  I=1,..,N-1
C       F(X(N)) = X(1)
C
C*************************************************************
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  AINV(40,40),AIN(40),IANUM(40),IAACT(40)
      DIMENSION  ITMAT(40,41),ITNUM(41),ITM(40)
      DIMENSION  IGAM(40),IBETA(39),PHULP(40),W(40)
      DATA  KDIM,NP1,CRITER,W/40,20,1.D-7,40*1.D0/
      CALL MINIT(ITMAT,KDIM,NP1,NP2,ITNUM,IGAM,AINV,IANUM,
     $      IAACT,ITRE,INVE,JMOUT,ITMNU)
      N = NP1-1
      MGRID = 2**17
      NGRID = 2
      WRITE(3,5000)
      WRITE(3,5001)  NP1
      WRITE(3,5002)
  350 MGRIT = 0
      DO 400  I = 1,NP1
      MGRIT = MGRIT+ITMAT(I,1)
  400 CONTINUE
      IF(MGRIT-NGRID)  600,450,600
  450 DO 455  I = 1,NP1
      ITM(I) = ITMAT(I,1)
  455 CONTINUE
      DMAX = DABS((ITM(1)-ITM(NP1))/DFLOAT(MGRIT))
      DO 460  I = 2,NP1
      FXX = DABS((ITM(I)-ITM(I-1))/DFLOAT(MGRIT))
      IF(FXX.GT.DMAX)  DMAX=FXX
  460 CONTINUE
      WRITE(3,4004)  MGRIT,ITRE,INVE,DMAX
      NGRID = NGRID*2
      IF(MGRIT-MGRID)  600,500,500
  500 WRITE(3,4001)  (ITMAT(J,1),J=1,NP1)
      CALL EXIT
  600 ITRE = ITRE+1
      CALL EAVES(ITMAT,KDIM,N,NP1,NP2,ITNUM,IGAM,
     *       IBETA,JMOUT,ITM,ITMNU)
      CALL LABELS(ITM,NP1,AIN,NACT)
      IF(NACT.NE.0)  GOTO 800
      CALL LABELF(ITM,NP1,AIN)
  800 CALL PIVOT(AINV,KDIM,AIN,W,PHULP,CRITER,NP1,NP2,IAACT,NACT,
     $      INVE,IANUM,ITNUM,JMOUT,ITMNU)
      GOTO  350
 4001 FORMAT (///' FIRST COLUMN OF FINAL MATRIX ''ITMAT'''/
     $    (1H0,10I7))
 4004 FORMAT (1H ,I7,9X,I8,12X,I8,D26.7)
 5000 FORMAT (1H1,'EXAMPLE OF THE USE OF EAVES'' PROCEDURE'/
     $  1H0,'FUNCTION  F(X(I)) = X(I+1)  FOR  I=1,..,N-1'/
     $  1H ,10X,'F(X(N)) = X(1)'/)
 5001 FORMAT (1H0,'N =',I3)
 5002 FORMAT (//' GRIDSIZE',5X,'# OF ITERATIONS'
     $  ,5X,'# OF INVERSIONS',7X,'MAX(ABS(F(X)-X))')
      END
```

```
      SUBROUTINE  MINIT(ITMAT,KDIM,NP1,NP2,ITNUM,IGAM,
     $    AINV,IANUM,IAACT,ITRE,INVE,JMOUT,ITMNU)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  ITMAT(KDIM,1),ITNUM(1),IGAM(1)
      DIMENSION  AINV(KDIM,1),IANUM(1),IAACT(1)
C
C     ***************************
C     * INITIALIZE ALL VARIABLES *
C     ***************************
C
      NP2 = NP1+1
      ITRE = -1
      INVE = 0
      DO 10  I = 1,NP2
      ITNUM(I) = I
 10   CONTINUE
      DO 20  J = 1,NP1
      IAACT(J) = J
      IANUM(J) = J+1
      IGAM(J) = J
 20   CONTINUE
      IANUM(NP1) = 1
      DO 40  I = 1,NP1
      DO 30  J = 1,NP2
      ITMAT(I,J) = 0
 30   CONTINUE
      ITMAT(I,I) = 1
 40   CONTINUE
      JMOUT = NP2
      ITMNU = NP2
      DO 50  I = 1,NP1
      DO 45  J = 1,NP1
      AINV(I,J) = 0.D0
 45   CONTINUE
      AINV(I,I) = 1.D0
 50   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE  EAVES(ITMAT,KDIM,N,NP1,NP2,ITNUM,IGAM,
     *          IBETA,JMOUT,ITM,ITMNU)
      DIMENSION  ITMAT(KDIM,1)
      DIMENSION  ITM(1),ITNUM(1),IGAM(1),IBETA(1)
C
C     ************************************
C     * REPLACEMENT STEP IN PRIMITIVE SET *
C     ************************************
C
      IF(JMOUT.NE.1)  GOTO 400
      IG = IGAM(1)
      IF(IG.NE.1)  GOTO 200
      DO 112  I = 1,NP1
      IF(ITMAT(I,1).NE.2*ITMAT(I,NP2))  GOTO 115
  112 CONTINUE
      GOTO  210
C
C     JMOUT=1
C     CASE(3), I=1,J=1
C
  115 IDUM = 1
      I = 1
  120 II = I+IDUM
      IGAM(I) = IGAM(II)-1
      IF(IGAM(I).NE.N)  GOTO 140
      IDUM = 0
      I = I+1
      IGAM(I) = NP1
      JIN = I
  140 I = I+1
      IF(I.GT.NP1)  GOTO 230
      GOTO 120
  200 IF(IG.EQ.NP1)  GOTO 300
C
C     JMOUT=1
C     CASE(1), I=1
C
  210 DO 220  I = 2,NP1
      IGAM(I-1) = IGAM(I)
  220 CONTINUE
      IGAM(NP1) = IG
      JIN = NP2
  230 DO 240  I = 1,NP1
      ITM(I) = ITMAT(I,2)+ITMAT(I,JIN)-ITMAT(I,1)
  240 CONTINUE
      GOTO  800
```

```
C
C       JMOUT=1
C       CASE(4)
C
  300   DO 320  J = 1,N
        DO 305  I = 1,NP1
        IF(ITMAT(I,J+1).GT.ITMAT(I,J+2))  GOTO 310
  305   CONTINUE
  310   IBETA(J) = I
  320   CONTINUE
        DO 325  I = 1,NP1
        ITM(I) = ITMAT(I,2)
  325   CONTINUE
        KK = 0
        DO 350  I = 1,N
        IF(ITM(I)/2*2.EQ.ITM(I))  GOTO 335
        ITM(I) = ITM(I)+1
        ITM(I+1) = ITM(I+1)-1
        KK = KK+1
        GOTO  350
  335   DO 340  J = 1,N
        IF(I.EQ.IBETA(J))  GOTO 345
  340   CONTINUE
  345   IBETA(J) = -IBETA(J)
  350   CONTINUE
        IH = 0
        IHK = KK
        DO 370  I = 1,N
        IF(IBETA(I).GT.0)  GOTO 365
        IHK = IHK+1
        IGAM(I) = IHK
        GOTO  370
  365   IH = IH+1
        IGAM(I) = IH
  370   CONTINUE
        IGAM(NP1) = NP1
        DO 375  I = 1,NP1
        ITM(I) = ITMAT(I,2)-ITM(I)/2
  375   CONTINUE
        JIN = NP2
        GOTO  800
```

```
  400   IF(JMOUT.LT.NP2)  GOTO 600
        IG = IGAM(NP1)
        IF(IG.EQ.NP1)  GOTO 500
C
C       JMOUT=NP2
C       CASE(1), I=NP1
C
        DO 430  I = 2,NP1
        K = NP1-I+1
        IGAM(K+1) = IGAM(K)
  430   CONTINUE
        IGAM(1) = IG
  440   JIN = 1
        DO 450  I = 1,NP1
        ITM(I) = ITMAT(I,1)+ITMAT(I,JMOUT-1)-ITMAT(I,JMOUT)
  450   CONTINUE
        GOTO  850
C
C       JMOUT=NP2
C       CASE(2)
C
  500   DO 510  I = 1,NP1
        IGAM(I) = I-1
  510   CONTINUE
        IGAM(1) = NP1
        DO 520  I = 1,NP1
        ITM(I) = ITMAT(I,1)+ITMAT(I,NP1)
  520   CONTINUE
        JIN = 1
        GOTO  850
  600   IF(IGAM(JMOUT-1).NE.IGAM(JMOUT)-1)  GOTO 642
        DO 620  I = 1,NP1
        IF(ITMAT(I,JMOUT).NE.2*ITMAT(I,NP2))  GOTO 700
  620   CONTINUE
C
C       1 < JMOUT < NP2
C       CASE(1), I=JMOUT
C
  642   IDUM = IGAM(JMOUT-1)
        IGAM(JMOUT-1) = IGAM(JMOUT)
        IGAM(JMOUT) = IDUM
  660   DO 665  I = 1,NP1
        ITM(I) = ITMAT(I,JMOUT-1)+ITMAT(I,JMOUT+1)-ITMAT(I,JMOUT)
  665   CONTINUE
        JIN = JMOUT
        GOTO  900
```

```
 700  IF(IGAM(JMOUT),NE,NP1)  GOTO 660
C
C     1 < JMOUT < NP2
C     CASE(3), I=JMOUT,J=NP1
C
      DO 730  I = 1,NP1
      IGAM(I) = IGAM(I)+1
 730  CONTINUE
      DO 740  I = 2,JMOUT
      K = JMOUT-I+1
      IGAM(K+1) = IGAM(K)
 740  CONTINUE
      IGAM(1) = 1
      GOTO  440

 800  DO 820  J = 2,JIN
      ITNUM(J-1) = ITNUM(J)
      DO 820  I = 1,NP1
      ITMAT(I,J-1) = ITMAT(I,J)
 820  CONTINUE
      GOTO  900
 850  DO 870  J = 2,JMOUT
      JJ = JMOUT-J+1
      ITNUM(JJ+1) = ITNUM(JJ)
      DO 870  I = 1,NP1
      ITMAT(I,JJ+1) = ITMAT(I,JJ)
 870  CONTINUE

 900  DO 910  I = 1,NP1
      ITMAT(I,JIN) = ITM(I)
 910  CONTINUE
      ITNUM(JIN) = ITMNU
      RETURN
      END
```

```
      SUBROUTINE  LABELS(ITM,NP1,AIN,NACT)
      IMPLICIT  REAL*8  (A-H,O-Z)
      DIMENSION  ITM(1),AIN(1)
C
C     ***********************************
C     * ASSOCIATE LABEL TO SLACK-VECTOR *
C     ***********************************
C
      DO 10  I = 1,NP1
      II = I
      IF(ITM(II).EQ.0)  GOTO 15
 10   CONTINUE
      NACT = 0
      RETURN
 15   DO 20  K = 1,NP1
      IK = II+K
      IF(IK.GT.NP1)  IK=IK-NP1
      IF(ITM(IK).GT.0)  GOTO 25
 20   CONTINUE
 25   KSLEK = IK-1
      IF(KSLEK.EQ.0)  KSLEK=NP1
      DO 30  I = 1,NP1
      AIN(I) = 0.D0
 30   CONTINUE
      AIN(KSLEK) = 1.D0
      NACT = KSLEK
      RETURN
      END
```

```
      SUBROUTINE  LABELF(ITM,NP1,AIN)
      IMPLICIT  REAL*8  (A-H,O-Z)
      DIMENSION  ITM(1),AIN(1)
C
C     ***************************************
C     * ASSOCIATE LABEL TO INTERIOR VECTOR *
C     ***************************************
C
      NGRID = 0
      DO 10  I = 1,NP1
      NGRID = NGRID+ITM(I)
 10   CONTINUE
      GRID = NGRID
      DO 30  I = 2,NP1
      AIN(I-1) = (ITM(I)-ITM(I-1))/GRID+1.D0
 30   CONTINUE
      AIN(NP1) = (ITM(1)-ITM(NP1))/GRID+1.D0
      RETURN
      END
```

```
      SUBROUTINE  PIVOT(AINV,KDIM,AIN,W,PHULP,CRITER,NP1,NP2,IAACT,NACT,
     $     INVE,IANUM,ITNUM,JMOUT,ITMNU)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  AINV(KDIM,KDIM),AIN(1),W(1),PHULP(1)
      DIMENSION  IAACT(1),IANUM(1),ITNUM(1)
C
C     *************
C     * PIVOT STEP *
C     *************
C
      IF(NACT.EQ.0)  GOTO 215
      DO 205  I = 1,NP1
      JAOUT = I
      IF(IAACT(I).EQ.NACT)  GOTO 235
  205 CONTINUE
  215 XMAX = 0.DO
      DO 218  I = 1,NP1
      PHULP(I) = 0.DO
      DO 218  J = 1,NP1
      PHULP(I) = PHULP(I)+AINV(I,J)*AIN(J)
  218 CONTINUE
      DO 230  I = 1,NP1
      IF(PHULP(I).LE.CRITER)  GOTO 230
      QHULP = 0.DO
      DO 222  J = 1,NP1
      QHULP = QHULP+AINV(I,J)*W(J)
  222 CONTINUE
      IF(QHULP.GT.CRITER)  GOTO 226
      JAOUT = I
      GOTO  232
  226 QUOT = PHULP(I)/QHULP
      IF(QUOT.LE.XMAX)  GOTO 230
      XMAX = QUOT
      JAOUT = I
  230 CONTINUE
  232 INVE = INVE+1
C
C     BARTELS' PROCEDURE
C
      DO 234  IC = 1,NP1
      CONST = AINV(JAOUT,IC)/PHULP(JAOUT)
      DO 233  JR = 1,NP1
      AINV(JR,IC) = AINV(JR,IC)-PHULP(JR)*CONST
  233 CONTINUE
      AINV(JAOUT,IC) = CONST
  234 CONTINUE

  235 IAOUD = IANUM(JAOUT)
      IANUM(JAOUT) = ITMNU
      IAACT(JAOUT) = NACT
      DO 270  I = 1,NP2
      IF(ITNUM(I).EQ.IAOUD)  GOTO 300
  270 CONTINUE
  300 JMOUT = I
      ITMNU = ITNUM(JMOUT)
      RETURN
      END
```

EXAMPLE OF THE USE OF EAVES' PROCEDURE

FUNCTION F(X(I)) = X(I+1) FOR I=1,..,N-1
          F(X(N)) = X(1)

N = 10

GRIDSIZE	# OF ITERATIONS	# OF INVERSIONS	MAX(ABS(F(X)-X))
2	0	0	0.5000000D+00
4	10	0	0.5000000D+00
8	38	0	0.2500000D+00
16	117	0	0.1250000D+00
32	188	47	0.6250000D-01
64	206	65	0.3125000D-01
128	224	83	0.1562500D-01
256	242	101	0.7812500D-02
512	260	119	0.3906250D-02
1024	278	137	0.1953125D-02
2048	296	155	0.9765625D-03
4096	314	173	0.4882813D-03
8192	332	191	0.2441406D-03
16384	350	209	0.1220703D-03
32768	368	227	0.6103516D-04
65536	386	245	0.3051758D-04
131072	404	263	0.1525879D-04

FIRST COLUMN OF FINAL MATRIX 'ITMAT'

 13107 13108 13106 13108 13108 13106 13108 13106 13108 13107

END OF EXECUTION
CPU TIME: 1.94 ELAPSED TIME: 1:2.78
EXIT

EXAMPLE OF THE USE OF EAVES' PROCEDURE

FUNCTION F(X(I)) = X(I+1) FOR I=1,..,N-1
F(X(N)) = X(1)

N = 20

GRIDSIZE	# OF ITERATIONS	# OF INVERSIONS	MAX(ABS(F(X)-X))
2	0	0	0.5000000D+00
4	20	0	0.5000000D+00
8	78	0	0.2500000D+00
16	287	0	0.1250000D+00
32	846	0	0.6250000D-01
64	1558	423	0.3125000D-01
128	1620	485	0.1562500D-01
256	1682	547	0.7812500D-02
512	1744	609	0.3906250D-02
1024	1806	671	0.1953125D-02
2048	1868	733	0.9765625D-03
4096	1930	795	0.4882813D-03
8192	1992	857	0.2441406D-03
16384	2054	919	0.1220703D-03
32768	2116	981	0.6103516D-04
65536	2178	1043	0.3051758D-04
131072	2240	1105	0.1525879D-04

FIRST COLUMN OF FINAL MATRIX 'ITMAT'

6553 6554 6554 6554 6552 6554 6554 6554 6554 6554

6552 6554 6554 6554 6554 6552 6554 6554 6554 6553

END OF EXECUTION
CPU TIME: 25.76 ELAPSED TIME: 1:51.16
EXIT

.

REFERENCES

[1] Bartels, R.H., A Stabilization of the simplex method, Num. Math 16, 414-434 (1971).

[2] Dohmen, J. and J. Schoeber, Approximated fixed points, Research memorandum EIT/55, Dep. of Econometrics; Tilburg University; Tilburg (1975).

[3] Eaves, B.C., Homotopies for Computation of Fixed Points, Mathematical Programming 3, 1-2 (1972)

[4] Scarf, H., The Approximation of Fixed Points of a continuous Mapping, SIAM Journal of Applied Mathematics 15, 1328-1343 (1967).

[5] Scarf, H., The computation of economic equilibria, Yale University Press, London (1973).

APPENDIX A PROOF OF THE EQUIVALENCE OF THE REFORMULATED AND EAVES' ORIGINAL REPLACEMENT STEP

In our proofs of the equivalence we will follow the chart flow of fig. 6.

1. JMOUT = 1, $\gamma_1 = 1$ and $T^1 = 2xT^{n+1}$

$T^1 = 2xT^{n+1}$ implies $T_\sigma(v^1) = 2xT_\sigma(v^{n+1})$ and since T_σ is a linear transformation on σ having full rank we may conclude that

$$v^1 = 2v^{n+1}$$
$$= 2(v^1 + \sum_{i=1}^{n} q(\gamma_i)) = 2(v^1 + \begin{pmatrix} -1 \\ 0 \\ \vdots \\ 0 \end{pmatrix})$$

and therefore $v^1 = \begin{pmatrix} 2 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ and $v^{n+1} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$. Moreover $v^2 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$

Computing v we find:

$$v = v^{n+1} + v^2 - v^1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} - \begin{pmatrix} 2 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \text{ yielding}$$

case (1), i = 1.

Case (1) shows that $\sigma(u^1,\beta)$ does not change and so

* $T_{\sigma'}(v) = T_\sigma(v) = T_\sigma(v^{n+1}) + T_\sigma(v^2) - T_\sigma(v^1)$ or in our notation:

$$TIN = T^{n+1} + T^2 - T^1$$

Table 1 shows furthermore that the new τ, denoted by τ' is built up in the following way:

$\tau' = \{v^2, v^3, \ldots, v^{n+1}, v\}$ and so TIN must be introduced in T on the (n+1)-th place. (JIN=n+1)

* All expressions with a prime will refer throughout this appendix to situations after the replacement step has been carried out.

2. JMOUT = 1, $\gamma_1 = 1$ and $T^1 \neq 2 \times T^{n+1}$

$T^1 \neq 2 \times T^{n+1}$ implies $v^1 \neq \begin{pmatrix} 2 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ and since $\gamma_1 = 1$, the first element of v^1, denoted by v_1^1 must be 1 and $v_1^2 = 0$.

From 1 we saw that $v^1 - v^{n+1} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ and so $v_1^{n+1} = 0$

Computing $v_1 = v_1^{n+1} + v_1^2 - v_1^1 = -1$ we arrive at case (3), with i = 1 and j = 1.

Table 2 shows that in this case, $\sigma' = \{u^2,\ldots,u^n,u^n+q(\beta_1)\}$ and using the changes in v^1 and γ as stated in table 3, we find:

$$T_{\sigma'}(v^{1'}) \overset{\text{def}}{=} \sum_{i=1}^{n} u^{i'}, v_i^{1'} = \sum_{i=1}^{n-1} u^{i+1} \cdot v_{i+1}^2, \text{ since } v_n^{1'} = 0$$

$$= \sum_{i=1}^{n} u^i . v_i^2 \text{ , since } \gamma_1 = 1 \text{ and so } v_1^2 = 0$$

$$= T_\sigma(v^2)$$

$$T_{\sigma'}(v^{2'}) \overset{\text{def}}{=} \sum_{i=1}^{n} u^{i'} . v_i^{2'} = \sum_{i=1}^{n-1} u^{i+1} . v_{i+1}^3, \text{ since } v_n^{2'} = 0 \text{ and } \gamma_1' = \gamma_2 - 1$$

$$= \sum_{i=1}^{n} u^i . v_i^3 = T_\sigma(v^3), \text{ since } v_1^3 = 0.$$

This procedure may be continued untill $T_{\sigma'}(v^{k'})$, $\gamma_k = n$.
Table 3 learns that:

$$T_{\sigma'}(v^{k'}) \overset{\text{def}}{=} \sum_{i=1}^{n} u^{i'} . v_i^{k'} = \sum_{i=1}^{n-1} u^{i+1} . v_{i+1}^k - u^n . v_n^k +$$

$$+ u^n . (v_n^k - 1) + u^n + q(\beta_1) = T_\sigma(v^k) + q(\beta_1)$$

It is immediate that this is the vector to be introduced in T, since $T_{\sigma'}(v^{i'}) = T_\sigma(v^i)$, i = k+1, n+1, so JIN = k, and since $\gamma_k = n$, k = NRLH.

It is straightforward to prove the calculation rule:

$$TIN = T^{NRLH} + T^2 - T^1$$

$$= T_\sigma(v^k) + \sum_i u^i_\cdot y^2_i - \sum_i u^i_\cdot v^1_i \text{ , and since } \gamma_1 = 1 \text{ we may write:}$$

$$= T_\sigma(v^k) + \sum_i u^i_\cdot v^1_i - u^1 + u^2 - \sum_i u^i_\cdot v^1_i = T_\sigma(v^k) + q(\beta_1)$$

$$= T_{\sigma'}(v^{k'})$$

3. JMOUT = 1, $\gamma_1 = n$

From $v^1 - v^{n+1} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ and $\gamma_1 = n$ it is seen that v^1_1 and v^1_n cannot be zero.

Therefore $v^1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 1 \end{pmatrix}$, $v^2 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ and $v^{n+1} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$

Computing $v = v^2 + v^{n+1} - v^1 = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$, we arrive at case (4).

4. JMOUT = 1 , $\gamma_1 \neq 1, n$

From $v = v^2 + v^{n+1} - v^1$ or otherwise $v = v^2 - \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ it is seen that

case (1) only may appear if a. $v^2_1 \neq 0$ and b. $0 < \sum_i v_i \leq 2$.

ad.a. Since $\gamma_1 \neq 1$ it follows that $v^2_1 = v^1_1 \neq 0$.

ad.b. Since $\gamma_1 \neq n$, $\sum_i v^2_i = \sum v^1_i = 2$, and so $0 < \sum_i v_i \leq 2$.

The proof that $TIN = T^{n+1} + T^2 - T^1$ and $JIN = n + 1$ is completely analogue with 1.

5. JMOUT = n + 1, $\gamma_n = n$

From $\gamma_n = n$ it follows that $\sum_i v_i^n = 2$. So if we compute

$v = v^1 + v^n - v^{n+1} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + v^n$, we arrive at case (2) since

$\sum_i v_i = 3$.

We will prove now that the new vector to be introduced in T may be computed as TIN = $T^1 + T^n$ and that JIN = 1.

The scheme of case (2) tells us that $T^{1'} = u'_1 + u'_n$, since $\gamma' = (1,0,\ldots,0,1)$. This may be rewritten as:

$$
\begin{aligned}
T^{1'} &= 2u'_1 + \sum_{i=1}^{n-1} q(\beta_i) \\
&= 2\{u^1\theta_1 + \sum_{i=1}^{n-1} \theta_{i+1} q(\beta_i)\} + \sum_i q(\beta_i) \\
&= 2\{\sum_{i=2}^{n} \theta_i (u^i - u^{i-1}) + u^1\theta_1\} + \sum_i q(\beta_i) \\
&= 2\{\sum_{i=1}^{n-1} u^i(\theta_i - \theta_{i+1}) + u^n\theta_n\} + \sum_i q(\beta_i) \\
&= 2\sum_{i=1}^{n} u^i v_i^1 + \Sigma\, q(\beta_i) = 2\,\sigma.v^1 + \Sigma\, q(\beta_i) \\
&= 2\sigma v^1 + \sum_{i=1}^{n-1} q(\gamma_i), \text{ since } \gamma_n = n \\
&= 2\sigma v^1 + \sigma \sum_{i=1}^{n-1} q(\gamma_i) \\
&= \sigma(v^1 + v^n) = T^1 + T^n
\end{aligned}
$$

q.e.d.

6. JMOUT = n+1, $\gamma_n \neq n$

Since $\gamma_n \neq n$, $\sum_i v_i^n = 1$. Computing $v = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + v^n$ it is immediate that $0 < \sum_i v_i \leq 2$ and $v_i \geq 0$, for all i, which correspond to case (1), i = n + 1

The proof that TIN = $T^1 + T^n - T^{n+1}$ and JIN = 1 is analogue with 1.

7. JMOUT $\neq$ 1, n + 1, $\gamma_{JMOUT-1} = \gamma_{JMOUT} - 1$, $T^{JMOUT} = 2 \times T^{n+1}$

In 1 we already stated that $T^{JMOUT} = 2 \times T^{n+1}$ implies $v^{JMOUT} = 2 \times v^{n+1}$ and moreover that v^{JMOUT} can only have one entry differing from zero equalling 2 and since JMOUT $\neq$ 1 this entry cannot be the first element of v^{JMOUT}.

Suppose the k-th element of v^{JMOUT} is 2. It is seen immediately that $v_k^{JMOUT-1} = 1$ and $v_{k-1}^{JMOUT-1} = 1$ and

$$v_k^{JMOUT+1} = 1 \text{ and } v_{k+1}^{JMOUT+1} = 1$$

From $v = v^{JMOUT-1} + v^{JMOUT+1} - v^{JMOUT}$ it follows then that only $v_{k-1} = v_{k+1} = 1$, while all other elements of v are zero.

So we are in case(1), i = JMOUT.

The new vector TIN may be computed as $TIN = T^{JMOUT-1} + T^{JMOUT+1} - T^{JMOUT}$ and JIN = JMOUT, which is easily proved according to 1.

8. JMOUT $\neq$ 1,n+1, $T^{JMOUT} \neq 2 \times T^{n+1}$, $\gamma_{JMOUT-1} = \gamma_{JMOUT} - 1$

Suppose $\gamma_{JMOUT} = k$ and $\gamma_{JMOUT-1} = k - 1$.

Since $v_k^{JMOUT} \neq 2$ and $\gamma_{JMOUT} = k$, v_k^{JMOUT} must equalling 1 and $v_k^{JMOUT+1} = 0$ and moreover $v_k^{JMOUT-1} = 0$, since $\gamma_{JMOUT-1} = k - 1$.

So $v_k = v_k^{JMOUT-1} + v_k^{JMOUT+1} - v_k^{JMOUT} = -1$, corresponding to case(3).

a. γJMOUT $\neq$ n.

Since $\gamma_{JMOUT} \neq 1,n$ it is noticed that $v_1 \neq -1$ and $v_n \neq -1$.

Therefore we are dealing with $1 < j < n$ in table 2 and $1 < j < n$, $1 \leq i \leq n+1$ in table 3.

From table 2 it is seen that σ is unchanged, except for u^k, $u^{k'} = u^{k-1} + u^{k+1} - u^k$. So in general this may affect $T_\sigma(\tau)$ for all v^i, having a non-zero element on the k-th place. But from our previous discussion it appears that v^{JMOUT} can only have a non-zero element on the k-th place, implying that in $T_\sigma(\tau)$ only $T_\sigma(v^{JMOUT})$ can change; so JIN = JMOUT.

We compute TIN according to:

$$TIN = T^{JMOUT-1} + T^{JMOUT+1} - T^{JMOUT}$$

$$\stackrel{def}{=} \sigma_{\sim}.v^{JMOUT-1} + \sigma.v^{JMOUT+1} - \sigma.v^{JMOUT}$$

$$= \sigma(v^{JMOUT}-q(k-1)+ \sigma(v^{JMOUT}+q(k)) - \sigma.v^{JMOUT}$$

$$= \sigma.v^{JMOUT} + \sigma(q(k)-q(k-1))$$

$$= \sigma.v^{JMOUT} + u^{k-1} - 2u^{k} + u^{k+1}$$

$$= \sigma.v^{JMOUT} + u^{k'} - u^{k} = \sigma'v^{JMOUT}, \text{ since } v_k^{JMOUT} = 1$$

$$= T_{\sigma'}(v^{JMOUT}) = T_{\sigma'}(v^{JMOUT'}), \text{ since } \tau(v^{1},\gamma)$$

does not change according to table 3.

b. $\gamma_{JMOUT} = n$

In this case we are dealing with j = n in table 2 and j = n, $1 \leq i \leq n$ in table 3.

If we are able to prove that $TIN = T^{1} + T^{JMOUT-1} + T^{JMOUT} = T_{\sigma'}(v^{1'})$ then it is clear that $T_{\sigma'}(v^{1'})$ is the new vector to be introduced, since $\{T^{1},\ldots,T^{JMOUT}\}$ consisting of vectors in the same level must form a linear independent set. So in that case JIN = 1.

We will compute $T_{\sigma'}(v^{1'})$ in the following way.

$$T_{\sigma'}(v^{1'}) \stackrel{def}{=} \sigma'.v^{1'}$$

Using the definition of $v^{1'}$ in table 3, we may write:

$$T_{\sigma'}(v^{1'}) = \sigma'.v^{1'} = u^{1'} - u^{2'} + \sum_{i=2}^{n} u^{i'}.v_{i-1}^{1} \text{ and from}$$

table 2, this may be reformulated in terms of the original $\sigma(u^{1},\beta)$ by:

$$= u^{1} - q(\beta_{n-1}) - u^{1} + \sum_{i=1}^{n-1} u^{i}.v_{i}^{1},$$

$$= u^{n-1} - u^{n} + \sum_{i=1}^{n} u^{i}.v_{i}^{1}, \text{ since } v_n^{1} \text{ must}$$

be zero; an impact of the fact that $\gamma_{JMOUT} = n$ and $\gamma_{JMOUT-1} = n-1$.

$$= - \sigma.q(n-1) + T_1$$

$$= T_1 + \sigma(v^{JMOUT} - q(n-1)) - \sigma.v^{JMOUT}$$

$$= T_1 + T^{JMOUT-1} - T^{JMOUT}, \qquad \text{q.e.d.}$$

9. JMOUT $\neq$ 1,n+1 $\qquad \gamma_{JMOUT-1} \neq \gamma_{JMOUT} - 1$

$\gamma_{JMOUT-1} \neq \gamma_{JMOUT} - 1$ implies that if $\gamma_{JMOUT} = k$, $v_k^{JMOUT} = 1$, $v_k^{JMOUT+1} = 0$ and $v_k^{JMOUT-1} = 1$ and supposing further that $\gamma_{JMOUT-1} = \ell \neq k-1$;we are in the following position

$$v_\ell^{JMOUT-1} = 1 \qquad v_\ell^{JMOUT} = 0$$

$$v_{\ell+1}^{JMOUT-1} = 0 \qquad v_{\ell+1}^{JMOUT} = 1 \qquad v_{\ell+1}^{JMOUT+1} = 1$$

$$v_k^{JMOUT-1} = 1 \qquad v_k^{JMOUT} = 1 \qquad v_k^{JMOUT+1} = 0$$

Computing v it follows then that $v_i \geq 0$, for all i and $\sum_i v_i \leq 2$, corresponding to case (1), i = JMOUT, which on its turn come up for the same implementations as in 7.

REPORTS 1977

7700 List of Reprints, nos. 179-194; List of Reports, 1976

7701/M "Triangular - Square -Pentagonal Numbers", by R.J. Stroeker.

7702/ES "The Exact MSE-Efficiency of the General Ridge Estimator relative to OLS", by R. Teekens and P.M.C. de Boer.

7703/ES "A Note on The Estimation of the Parameters of a Multiplicative Allocation Model", by R. Teekens and R. Jansen.

7704/ES "On the Notion of Probability: A Survey",by E. de Leede and J. Koerts.

7705/ES "A Mathematical Theory of Store Operation", by B. Nooteboom.

7706/ES "An Analysis of Efficiency in Retailing", by B. Nooteboom.

7707/S "A Note on Theil's Device for choosing a Blus Base", by C. Dubbelman.

7708/E "A General Market Model of Labour Income Distribution: An Outline, by W.H. Somermeyer.

7709/E "Further Results on Efficient Estimation of Income Distribution Parameters", by T. Kloek and H.K. van Dijk.

7710 "List of Reprints, nos 195-199; Abstracts of Reports First Half 1977".

7711/M "Degenerating Families of Linear Dynamical SystemsI", by M. Hazewinkel.

7712/M "Twisted Lubin-Tate Formel Group Laws, Ramified Witt Vectors and (Ramified) Artin-Hasse Exponential Mappings", by M. Hazewinkel.

7713/EM "An Efficient Way of Programming 'Eaves' Fixed Point Algorithm" by R. Jansen and A.S. Louter.