Approximated fixed points

Dohmen, J.; Schoeber, J.

Publication date:
1975

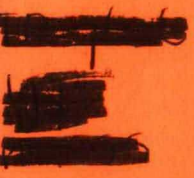
Document Version
Publisher's PDF, also known as Version of record

Jo Dohmen
Jan Schoeber

APPROXIMATED FIXED POINTS

Research memorandum

TILBURG UNIVERSITY
DEPARTMENT OF ECONOMETRICS
Hogeschoollaan 225 Tilburg (Holland)



APPROXIMATED FIXED POINTS

Jo Dohmen
Jan Schoeber

PREFACE

In recent years, economists build more and more complex models. Finding an equilibrium in these models involves lots of difficulties. Hence there is an increasing need for a general method to compute equilibria in mathematical models for complex systems. In this research memorandum we describe an efficient algorithm to compute approximated fixed points of vector functions, based on algebraic topological methods. Our work has been particularly inspired by a paper of B.C. Eaves [5]. Besides for the equilibria problem mentioned above the algorithm can be used in other branches of applied mathematics dealing with highly non-linear equation systems.
The E.I.T., the Katholieke Hogeschool Tilburg and especially the Computercentre of the K.H.T. made this project possible by supplying the means.
In addition to these institutes we are much indebted to Dr. M.H.C. Paardekoper whose help has been indispensable.
The E.I.T. provided secreterial assistence for typing the manuscript.
Finally we remark that the main lines of this report can be understood without reading the proofs and the lemmas.

Tilburg, June 1975.

Jo Dohmen
Jan Schoeber

CONTENTS

LIST OF SYMBOLS

α	Vector with $\Sigma_{i=0}^{n} \alpha_i = 1$, defined in 2.3.2.
β_{ij}	Elements of B^{-1}
ε	Small positive real number
$[\varepsilon]$	$(-\Sigma_{i=1}^{n} \varepsilon^i, \varepsilon, \varepsilon^2, \ldots, \varepsilon^n)^T$
ε'	$(1, \varepsilon, \varepsilon^2, \ldots, \varepsilon^n)^T$
λ	Vector with $\Sigma\lambda_i = 1$ and $\lambda_i \geq 0$
ρ	Scalar
σ, σ_*	Simplexes (in chapter 3 and 4: simplex of some K_d). σ^i, σ' are faces of σ
τ, τ_*	Simplexes of an (n+1)-pseudomanifold. (In chapter 3 of $K_d * \{0\}$; in chapter 4 of T). τ^i, τ' are faces of τ
ϕ	Simplex of K_*. ϕ^i, ϕ' are faces of ϕ
B	(n+1)×(n+1)-matrix defined in 2.3.2.
C(.)	Convex hull
C	(n+1)×n-matrix defined in 3.2.1.
C'	(n+1)×(n+1)-matrix $[C, -e^n]$
D	The set $\{d \mid d = 2^n, n \in N\}$
F	Point to set mapping from S into S^*
K	Complex, pseudomanifold
K^0	The set of vertices of K
K_d	Standard subdivision of d·S

$K_d * \{0\}$	(n+1)-pseudomanifold defined in 3.3.1.
K_*	(n+1)-pseudomanifold defined in 4.4.1.
N	The set of positive integers
R^n	n-dimensional real space
S	Standard simplex $\{x \mid ex = 1,\ x_i \geq 0\} \subset R^{n+1}$ $d \cdot S = \{u \mid u = dx,\ x \in S\}$
S^*	The collection of convex subsets of S
T	(n+1)-pseudomanifold defined in 4.2.1.
Z	number of iterations
$\bar{Z}(n,o)$	standard for the expected number of iterations (5.6.1.)
b_i	for i = 0,...n: i^{th} column of B; for i = n+1: column defined in 2.3.2.
c	$Z/\bar{Z}(n,o)$
c(i)	i^{th} column of C and C'
d	positive integer. (In chapter 4 d is an element of D)
e	sum vector $(1,1,\ldots,1) \in R^{n+1}$
e^i	i^{th} unit vector of R^{n+1}
f	continuous function from S into itself
f_d	perturbed piecewise linear approximation of f
g	perturbation of f
h_σ	bijection from T into $\sigma \cup (\sigma+\sigma)$
k	index of the vertex to be deleted from τ or ϕ
ℓ	labelling

m	array containing integer labels (chapter 3) index indicating that $q_m = n+1$ (chapter 4)
n	dimension of S
o	$^2\log d$
p	permutation of (1,...,n)
q	permutation of (1,...,n+1)
r	permutation of (1,...,n)
s	permutation of (1,...,n+1)
u, u^i, u^*	vertex of some σ; u^* the vertex added
v, v^i, v^*	vertex of some τ; v^* the vertex added
w	vertex of K^0_*
x	point of S
y^i	vertex of some σ
z^i	vertex of some τ.

Chapter 1. INTRODUCTION

According to Brouwer's fixed point theorem a continuous function $F: C \to C$ from a compact convex subset C of R^n into itself has a point $\hat{x} \in C$ with $f(\hat{x}) = \hat{x}$, known as a fixed point. A general method to compute such a fixed point is not indicated by Brouwer's proof nor by the one based on Sperner's lemma (see [9], page 127). Shortly however various algorithms to approximate fixed points have been constructed. These algorithms generally take place in a simplex. Such a simplification is allowed since a compact convex set and a simplex of the same dimension are homeomorphic[1].

Sperner's lemma states that if a simplex is subdivided into subsimplexes and, according to certain demands, labels are assigned to the vertices of these subsimplexes then a subsimplex exists with a specific label constellation. Further on we shall explain the specific properties of such a subsimplex and call it complete. By choosing a proper labelling we can manage this subsimplex to serve as a basis for an approximation of a fixed point. Furthermore it is possible to construct a sequence of such simplicial subdivisions with simplexes of decreasing size. The corresponding sequence of complete subsimplexes will lead to continually better approximations of a fixed point and indeed in the limiting sense yields a fixed point.

From the algorithmic point of view first the problem arises to find a complete subsimplex, whose existence is stated by Sperner's lemma. Scarf [6] was the first to describe an algorithm to approximate fixed points by solving an analoguous problem. His idea served as a basis for the other procedures as well. One of the algorithms to find a complete subsimplex in a fixed simplicial subdivision K_d is outlined in chapter 3.

---

1) The condition of convexity is not necessary but can be replaced by a weakened condition.

The second problem is the construction of a sequence of simplicial subdivisions in such a way that the successive complete subsimplexes can be found recursively. This was solved by Eaves [5], who united simplicial subdivisions K_d of the n-dimensional standard simplex into a (n+1)-dimensional complex K_* and pointed out a pivoting routine to find a sequence of complete subsimplexes.
To this procedure, that might be called a generalized bisection algorithm, we shall pay extensive attention in chapter 4. In chapter 5 this procedure is implemented in a computer program. Although the search algorithms in a fixed subdivision K_d have appeared to be inferior to the latter, we have decided to describe one of them. Not only for historical reasons, but firm knowledge of K_d is needed anyway for the description of K_*. To obtain a narrow correspondence with chapter 4 the algorithm of chapter 3 is embedded in a very general theoretical environment so that both can be based on the same fundamental theorems, treated in chapter 2.

Without loss of generality we can confine ourselves to the *standard n-simplex* $S := C(e^0,\ldots,e^n)$, the convex hull of the unit vectors of R^{n+1}. So from now we are interested in the fixed points of the function $f: S \to S$. Except for this S the word simplex will always indicate a set of vertices and not the convex hull of these vertices. Further we shall use e for the row vector $(1,\ldots,1) \in R^{n+1}$ and for any map $F: V \to W$ and any set $U \subset V$ we shall denote with $F(U)$ the set $\{F(u) \mid u \in U\} \subset W$.
Calling $\tilde{x}$, obtained by an algorithm, an approximation of a fixed point $\hat{x}$ suggests that $\hat{x}$ is near to $\tilde{x}$. But this can not be quaranteed, though it is mostly true. Therefore we shall speak of an ε-approximated fixed point $\tilde{x}$ if $|\tilde{x}-f(\tilde{x})| \leq \varepsilon$ in some norm. In fact such an $\tilde{x}$ can be considered as a fixed point of a function $\tilde{f}$ which approximates f.
A function $\ell: V \to (S-S)$ is called a *labelling*. A subset U of V is said to be *complete* if $0 \in C(\ell(U))$. An example of a

proper labelling on S is $\ell(x) = x-f(x)$ for $x \in S$. For a given ε this labelling enables a sufficiently small complete subsimplex σ of S to provide an ε-approximated fixed point of f, as is proved in appendix 1. In other words the linear function $\tilde{f}: C(\sigma) \to S$ determined by the function values of f on σ has a fixed point $\tilde{x} \in C(\sigma)$.
As will be demonstrated at the end of chapter 3, the usual integer labelling in the proof of Brouwer's theorem by means of the Sperner lemma can be considered as a short notation of a special kind of labelling in the sense just defined. Finally, also Kakutani fixed points can be approximated by the procedures to be described. For, if $F: S \to S^*$ is a point to set mapping from S into S^*, the collection of convex subsets of S, this can be done by using the algorithm for the function $f: S \to S$ with $f(x) :=$ any element of $F(x)$ for $x \in S$. This simple device may lead to some practical problems, to which we return briefly in section 5.4.

Chapter 2. BASIC THEOREMS

2.1. INTRODUCTION

In this chapter we shall treat basic principles of the algorithms in this report. In addition to the term complete in connection with labellings we introduce the term very complete in order to avoid degeneracy just as in linear programming (see [3]). Theorem 2.3.3. then states the existence of a sequence of very complete simplexes in a pseudomanifold (see figure 1). For the computation of this sequence we need an iterative procedure, based on theorem 2.3.1.

2.2. DEFINITIONS

Let K^0 be a set of objects called *vertices* and let K be a collection of nonempty finite subsets of K^0 called *simplexes* such that

(1) if $u \in K^0$ then $\{u\} \in K$,

(2) if $\phi \neq \sigma' \subset \sigma \in K$ then $\sigma' \in K$.

We define such a K to be a *complex*. A simplex $\sigma \in K$ containing exactly q+1 vertices is called a *q-simplex* and is said to be q-dimensional. If $\sigma' \subset \sigma$ then σ' is called a *face* of σ, and more precisely a *p-face* if σ' is a p-simplex. Two distinct p-simplexes of K are said to be *adjacent* if they are faces of a common simplex. We further say that K is q-dimensional and call it a *q-complex* if K contains a q-simplex but no (q+1)-simplex.

A complex K is said to be a q-dimensional psuedomanifold, or a *q-psuedomanifold*, if

(1) each simplex of K is a face of a q-simplex,

(2) each (q-1)-simplex is a face of one or two q-simplexes.

Those (q-1)-simplexes that are in only one q-simplex are termed *boundary*. The others are termed *interior* [8].

Let K be an (n+1)-psuedomanifold and $\ell: K^0 \to S\text{-}S$. A simplex $\sigma = \{u^0,\ldots,u^k\} \in K$ is defined to be *very* (or nondegenerate) *complete* if there is an $\varepsilon_\sigma \in R_+$ such that for each $\varepsilon \in [0,\varepsilon_\sigma]$ there is a $\lambda = (\lambda_0,\ldots,\lambda_k) \in R^{k+1}$ with $e\lambda = 1$ and $\lambda \geq 0$ for which

$$\sum_{i=0}^{k} \lambda_i \ell(u^i) = [\varepsilon] := \begin{bmatrix} -\varepsilon-\varepsilon^2-\ldots-\varepsilon^n \\ \varepsilon \\ \varepsilon^2 \\ \vdots \\ \varepsilon^n \end{bmatrix},$$

where ε^i is the ith power of ε.

We can consider $[\varepsilon]$ as a point on a curve parametrized with ε. Then this last definition implies that $\sigma \in K$ is very complete if there exists a connected piece, containing 0, of the non-negative part of this curve, that is a subset of $C(\ell(\sigma))$. Because this curve can only be contained in spaces of dimension n or higher, only n- and (n+1)-simplexes in K can be very complete. It follows immediately that an (n+1)-simplex of K that contains a very complete n-face is very complete too.

2.3. THE MAIN THEOREMS

THEOREM 2.3.1.

Given an (n+1)-psuedomanifold K and $\ell: K^0 \to S\text{-}S$, then a very complete boundary n-simplex of K is adjacent to exactly one very complete n-simplex of K and a very complete interior n-simplex of K is adjacent to exactly two very complete n-simplexes of K.

PROOF: Let $\tau = \{v^0,\ldots,v^{n+1}\} \in K$, $\tau^i := \tau\backslash\{u^i\}$ and $\ell^i := \ell(v^i) = (\ell_0(v^i),\ldots,\ell_n(v^i))^T$. Suppose $\tau^{n+1} = \{v^0,\ldots,v^n\}$ is a very complete n-face of τ, i.e. for all sufficiently small $\varepsilon \geq 0$ there exists a $\lambda = (\lambda_0,\ldots,\lambda_n)^T \in S$ such that

$$\sum_{i=0}^{n} \lambda_i \ell^i = [\varepsilon]$$

or, by defining $b^i := (1,\ell^i,\ldots,\ell^i_n)^T$, $\varepsilon' := (1,\varepsilon,\varepsilon^2,\ldots,\varepsilon^n)^T$ and $B := (b^0,\ldots,b^n) \in R^{(n+1)\times(n+1)}$:
for all sufficiently small $\varepsilon \geq 0$ the system $B\lambda = \varepsilon'$ has a solution $\lambda \geq 0$.
It follows that B has an inverse and that λ depends on ε, so we can write $\lambda(\varepsilon) = B^{-1}\varepsilon'$. Let us define $[\beta_{ij}] := B^{-1}$, $\beta_{i\cdot}$ as the i^{th} row of B^{-1} and $\beta_{\cdot j}$ as the j^{th} column of B^{-1} $(i,j \in \{0,\ldots,n\})$. Then

$$\lambda_i(\varepsilon) = \beta_{i\cdot}\varepsilon' = \sum_{j=0}^{n} \beta_{ik}\varepsilon^j \,. \qquad (2.3.1.1.)$$

$\lambda_i(\varepsilon)$ is a polynomial of degree n of ε. So it has at most n zeroes. Let ε_i be the smallest positive zero of $\lambda_i(\varepsilon)$. Then for all $\varepsilon \in (0, \min\{\varepsilon_i \mid 0 \leq i \leq n\})$ all $\lambda_i(\varepsilon)$ are positive, so for all sufficiently small $\varepsilon > 0$, $\lambda(\varepsilon) = B^{-1}\varepsilon' > 0$. Note that $\lambda(0) = \beta_{\cdot 0}$.
From $\ell^i \in S\text{-}S$ and τ^{n+1} very complete we derive: τ is very complete too and ℓ^{n+1} can be written as

$$\ell^{n+1} = \sum_{i=0}^{n} \alpha_i \ell^i \quad \text{with} \quad \sum_{i=0}^{n} \alpha_i = 1, \qquad (2.3.1.2.)$$

which implies that $b^{n+1} = B\alpha$, with $\alpha := (\alpha_0,\ldots,\alpha_n)^T$, so $\alpha = B^{-1}b^{n+1}$.
Now let us examine τ^j for $j \in \{0,\ldots,n\}$. Then there are two possibilities: (i)$\alpha_j \neq 0$ and (ii)$\alpha_j = 0$.

(i). If $\alpha_j \neq 0$ then $\ell^j = \ell^{n+1}/\alpha_j - \Sigma^n_{i=0,i\neq j}(\alpha_i/\alpha_j)\ell^i$.
The very completeness of τ^{n+1} implies that $\Sigma^n_{i=0}\lambda_i(\varepsilon)\ell^i =$
$= (\lambda_j(\varepsilon)/\alpha_j)\ell^{n+1} + \Sigma^n_{i=0,i\neq j}(\lambda_i(\varepsilon)-(\alpha_i/\alpha_j)\lambda_j(\varepsilon))\ell^i = [\varepsilon]$.
Hence τ^j is very complete if and only if the following holds for all sufficiently small $\varepsilon \geq 0$:

$$\frac{\lambda_j(\varepsilon)}{\alpha_j} \geq 0 \qquad (2.3.1.3.)$$

$$\text{and } \lambda_i(\varepsilon) - \frac{\alpha_i}{\alpha_j}\lambda_j(\varepsilon) \geq 0 \text{ for every } i \neq j\,. \qquad (2.3.1.4.)$$

For (2.3.1.3.) being true α_j needs to be positive, because $\lambda_j(\varepsilon) \geq 0$. Then (2.3.1.4.) holds for all i with $\alpha_i \leq 0$ because for these i $\lambda_i(\varepsilon)-(\alpha_i/\alpha_j)\lambda_j(\varepsilon) \geq$
$\geq \lambda_i(\varepsilon) \geq 0$. In the other cases, if $\alpha_i > 0$ the condition (2.3.1.4.) can be rewritten as $\lambda_i(\varepsilon)/\alpha_i - \lambda_j(\varepsilon)/\alpha_j \geq 0$.
But this occurs for every i with $\alpha_i > 0$ iff for all sufficiently small $\varepsilon \geq 0$

$$\frac{\lambda_j(\varepsilon)}{\alpha_j} = \min\left\{\frac{\lambda_i(\varepsilon)}{\alpha_i}\,\middle|\,\alpha_i > 0,\ 0 \leq i \leq n\right\}. \qquad (2.3.1.5.)$$

So if $\alpha_j \neq 0$, τ^j is very complete if and only if (2.3.1.5.) is true. Since at least one α_i is positive an index j satisfying (2.3.1.5.) exists.
Now let us demonstrate the uniqueness of this index. From (2.3.1.1.) we know that $\lambda_j(\varepsilon) = \Sigma^n_{i=0}\beta_{ji}\varepsilon^i$. Now let J^0 be the collection defined by

$$J^0 := \left\{j\,\middle|\,\frac{\beta_{j0}}{\alpha_j} = \min\left\{\frac{\beta_{i0}}{\alpha_i}\,\middle|\,\alpha_i > 0,\ 0 \leq i \leq n\right\}\right\}.$$

If J^0 contains a single element j, then by taking ε sufficiently small this j is the unique index satisfying (2.3.1.5.). If J^0 contains more than one element

then we perform the collections J^1, J^2,..., defined recursively by

$$J^m := \left\{ j \mid \beta_{jk}/\alpha_j = \min \{ \beta_{ik}/\alpha_i \mid i \in J^{m-1} \} \right\}, \quad m = 1,\ldots,n$$

until we find a J^m containing one single element, say $J^m = \{j\}$. Then by taking ε sufficiently small, this j is the unique one satisfying (2.3.1.5.). The existence of such a J^m is easily proved, for suppose J^n contains more than one element, say $g \in J^n$ and $h \in J^n$, then $\beta_{gm}/\alpha_g = \beta_{hm}/\alpha_h$ for every $m \in \{0,\ldots,n\}$. But this means $\beta_{g\cdot} = (\alpha_g/\alpha_h)\beta_{h\cdot}$, so two rows of B^{-1} would be linearly dependent, a contradiction. Hence there is a unique index k satisfying (2.3.1.5.) and consequently among the τ^j with $\alpha_j \neq 0$ there is a unique very complete n-simplex τ^k. Furthermore for this simplex α_k is positive.

(ii). If $\alpha_j = 0$ then τ^j is not very complete. To prove this we assume that τ^j is very complete, i.e. for all sufficiently small ε there exists a $\mu(\varepsilon) \in S$ such that $\Sigma_{i=0,i\neq j}^{n+1} \mu_i(\varepsilon) \ell^i = [\varepsilon]$. From (2.3.1.2.) and the fact that $\alpha_j = 0$, we also know that $\ell^{n+1} = \Sigma_{i=0,i\neq j}^{n} \alpha_i \ell^i$ with $\Sigma_{i=0,i\neq j}^{n} \alpha_i = 1$. So from part (i) of this proof it follows that τ^j has a very complete face $\tau^{ji} := \tau^j \setminus \{v^i\}$ (with $\alpha_i > 0$). But this τ^{ji} would be a very complete $(n-1)$-simplex, which is impossible.

From the parts (i) and (ii) (note: at least one positive α_j exists) follows the existence of a unique very complete n-face of τ differing from τ^{n+1}. So a very complete n-face of an $(n+1)$-simplex is adjacent to just one other very complete n-face of this $(n+1)$-simplex. This exactly states our theorem, regarding the definitions of boundary and interior n-simplexes in an $(n+1)$-psuedomanifold and the triviality: two $(n+1)$-simplexes have at most one n-face in common. □

2.3.2.

For a given ℓ: $K^0 \to$ S-S and a given $\tau = \{v^0,\ldots,v^{n+1}\} \in K$ with a very complete face $\tau^{n+1} = \tau\backslash\{v^{n+1}\}$ this theorem indicates the computation of the other very complete face of τ, namely:

Compute $B := \begin{bmatrix} 1 & \ldots & 1 \\ \ell_1(v^0) & & \ell_1(v^n) \\ \vdots & & \vdots \\ \ell_n(v^0) & \ldots & \ell_n(v^n) \end{bmatrix}$ and $b^{n+1} := \begin{bmatrix} 1 \\ \ell_1(v^{n+1}) \\ \vdots \\ \ell_n(v^{n+1}) \end{bmatrix}$.

Compute $[\beta_{ij}] := B^{-1}$ and $\alpha := B^{-1}b^{n+1}$.

Compute for those i with $\alpha_i > 0$ the rows $\beta_{i\cdot}/\alpha_i$ and determine which row $\beta_{k\cdot}/\alpha_k$ is their lexicographic minimum.
Result $\tau^k := \tau\backslash\{v^k\}$ is the other very complete face of σ.

THEOREM 2.3.3.

Given an (n+1)-psuedomanifold K and a labelling ℓ: $K^0 \to$ S-S, let τ_0 be a very complete boundary n-simplex of K. Then there is a unique sequence $\tau_0,\tau_1,\ldots$ of distinct very complete consecutively adjacent n-simplexes in K. This sequence terminates with a very complete boundary n-simplex or is infinite. (Of course the latter can only occur if K is infinite.)

PROOF: According to theorem 2.3.1. the boundary n-simplex τ_0 is adjacent to a unique very complete n-simplex, to be called τ_1. Now let $\tau_0,\ldots,\tau_k$ be the unique sequence of k+1 dinstinct very complete adjacent n-simplexes in K. If τ_{k-1} is the only very complete n-simplex adjacent to τ_k, then τ_k is a very complete boundary n-simplex,

terminating our sequence. Else, so τ_k being an interior simplex, except for τ_{k-1} there is one other very complete n-simplex adjacent to τ_k, say σ. If $\sigma = \tau_0$, then $\tau_k = \tau_1$, a contradiction. If $\sigma = \tau_i$ for some $i \in \{1,\ldots,k-2\}$ then $\tau_k = \tau_{i-1}$ or $\tau_k = \tau_{i+1}$, again a contradiction. So $\tau_0,\ldots,\tau_k,\tau_{k+1}$ $(:= \sigma)$ is again a sequence of distinct very complete consecutively adjacent n-simplexes and is the unique sequence consisting of k+2 elements. □

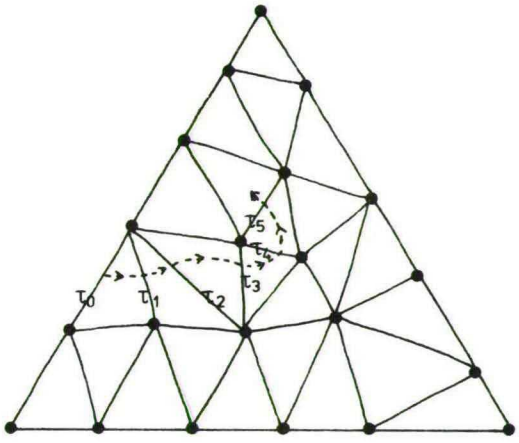


figure 1

Chapter 3. AN ALGORITHM FOR A FIXED SUBDIVISION

3.1. INTRODUCTION

In this chapter we shall describe the psuedomanifold K_d, that subdivides the set $d \cdot S := \{u | u = d \cdot x,\ x \in S\}$, where d is a given positive integer. Our purpose will be to give an algorithm to find a very complete simplex in K_d. Of course this algorithm is based on theorem 2.3.3., indicating an iterative procedure. Each step of this exchange procedure is built up of two parts: deleting a vertex and adding a vertex. How to delete a vertex has been pointed out in 2.3.2. Adding a vertex depends of course on the characteristics of the pseudomanifold and so will be discussed in this chapter.

3.2. THE STANDARD SUBDISIVION

DEFINITION 3.2.1.

Let d be a positive integer and

$$K_d^0 := \{u \in R^{n+1} | u_i \in \{0,1,\ldots,d\},\ eu = d\}$$

Then the *standard subdivision* K_d of $d \cdot S$ is defined to be the n-complex containing the faces of all simplexes $\sigma := \{u^0,\ldots,u^n\} \subset K_d^0$ that can be described by means of

a vertex u^0 and a permutation $p := (p_1,\ldots,p_n)$

of $(1,\ldots,n)$ in such a way that the other vertices of σ can be computed recursively from u^0 by

$$u^i = u^{i-1} + c(p_i) \qquad i = 1,\ldots,n\ ,$$

where $c(p_i) = (c_0(p_i),\ldots,c_n(p_i))^T$ is the p_ith column of the $(n+1)\times n$-matrix

$$C := \begin{bmatrix} -1 & 0 & . & . & . & 0 & 0 \\ +1 & -1 & . & & & . & . \\ 0 & +1 & & . & & . & . \\ . & 0 & & & . & . & . \\ . & . & . & & & 0 & . \\ . & . & & . & & -1 & 0 \\ . & . & & & . & +1 & -1 \\ 0 & 0 & . & . & . & 0 & +1 \end{bmatrix} .$$

We shall write $\sigma \nsim (u^0,p)$. Note that $u^0,u^1,\ldots,u^n$ is a lexicographic decreasing sequence of vertices, and that it follows that σ is uniquely described by (u^0,p).

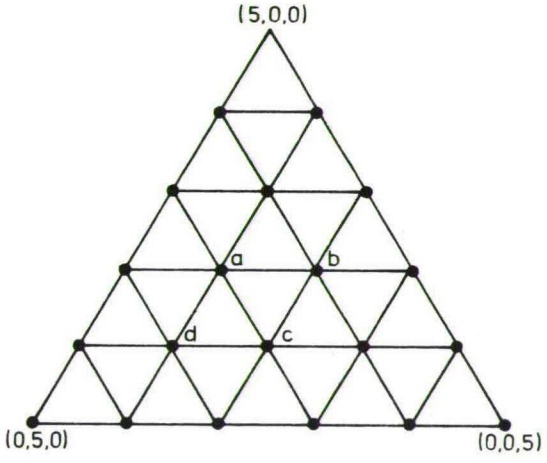


figure 2. K_5 $\{a,b,c\} \nsim ((2,2,1),(2,1))$
$\{a,d,c\} \nsim ((2,2,1),(1,2))$

Before we prove that this K_d is a pseudomanifold we shall state first of all

LEMMA 3.2.2.

Let $\sigma \approx (u^0,p)$ be an n-simplex of K_d, then in each other n-simplex (y^0,r) that contains two successive vertices u^{i-1} and u^i of σ, these vertices are successive too.

PROOF: Suppose $u^{i-1} = y^j$ and $u^i = y^k$, then $k > j$ because both $u^0,\ldots,u^n$ and $y^0,\ldots,y^n$ are lexicographic decreasing sequences. From the definition of simplexes in K_d it follows that $u^i - u^{i-1} = c(p_i)$ and $y^k - y^j = \Sigma_{i=j+1}^{k} c(r_i)$. So $c(p_i) = \Sigma_{i=j+1}^{k} c(r_i)$. Then from the linear independency of the columns of C follows $k = j+1$ and $p_i = r_k$. □

THEOREM 3.2.3.

K_d is an n-psuedomanifold.

PROOF: From the definition of K_d it follows that every simplex in K_d is a face of an n-simplex in K_d. So the first requirement for a pseudomanifold has been fulfilled and hence for each (n-1)-simplex $\sigma' \in K_d$ an n-simplex $\sigma \approx (u^0,p) \in K_d$ exists such that σ' is a face of σ. To demonstrate that K_d is a pseudomanifold it suffices to show for given σ' the existence of unique u^*, y^0 and r such that $\sigma_* := \sigma' \cup \{u^*\} \approx (y^0,r)$ is different from σ. Let $\{u_j\}$ be $\sigma \setminus \sigma'$. Then we distinguish three cases.

(1). $j = 0$. Then $\sigma' = \{u^1,\ldots,u^n\}$ and lemma 3.2.2. implies two possibilities with respect to σ'. Firstly $\sigma' = \{y^1,\ldots,y^n\}$ and $(r_2,\ldots,r_n) = (p_2,\ldots,p_n)$ implying that σ_* is not different from σ. Secondly $\sigma' =$

$\{y^0,\ldots,y^{n-1}\}$ and $(r_1,\ldots,r_{n-1}) = (p_2,\ldots,p_n)$. Then $r_n = p_1$ and $u^* = y^n = u^n + c(p_1)$.

(2). $0 < j < n$. Then $\sigma' = \{u^0,\ldots,u^{j-1}, u^{j+1},\ldots,u^n\}$ and equals $\{y^0,\ldots,y^{j-1},y^{j+1},\ldots,y^n\}$, since every other configuration clashes with lemma 3.2.2. Then $(r_1,\ldots,r_{j-1},r_{j+2},\ldots,r_n) = (p_1,\ldots,p_{j-1},p_{j+2},\ldots,p_n)$ and either $r_j = p_j$ and $r_{j+1} = p_{j+1}$, which implies that $u^* = y^j = u^j$ and σ^* is not different from σ, or $r_j = p_{j+1}$ and $r_{j+1} = p_j$. Then $u^* = y^j = u^{j-1} + c(p_{j+1}) = u^{j+1} - c(p_j)$.

(3). $j = n$. For reasons of symmetry with case (1) $\sigma' = \{u^0,\ldots,u^{n-1}\} = \{y^1,\ldots,y^n\}$, $(r_1,\ldots,r_n) = (p_n,p_1,\ldots,p_{n-1})$ and $u^* = y^0 = u^0 - c(p_n)$.

So in all three cases there is a unique $\sigma_* = \sigma' \cup\{u^*\} \neq \sigma$. □

This proof describes the second part of the exchange procedure, namely adding a vertex u^* to a given (n-1)-face $\sigma' = \sigma\backslash\{u^j\}$ of $\sigma \nsim (u^0,p) \in K_d$, and supplies us with the rules for the computation of $(y^0,r) \nsim \sigma_* := \sigma' \cup\{u^*\} \neq \sigma$ as comprimed in

Table 1.

	j	y^0	r	u^*
case 1	j=0	$u^0+c(p_1)$	$(p_2,..,p_n,p_1)$	$y^n=u^n+c(p_1)$
case 2	0<j<n	u^0	$(p_1,..,p_{j-1},p_{j+1},p_j,..,p_n)$	$y^j=u^{j-1}+c(p_{j+1})$
case 3	j=n	$u^0-c(p_n)$	$(p_n,p_1,..,p_{n-1})$	$y^0=u^0-c(p_n)$

Obviously σ' is an interior simplex if $u^* \in K_d^0$, otherwise σ' is a boundary simplex. Observe that σ' is a boundary simplex if and only if $u_i^* < 0$ for some i, so, in consequence of the structure of the matrix C, iff an index i exists such that $u_i = 0$ for every $u \in \sigma'$.

3.3. ON THE START OF THE ALGORITHM.

In order to obtain a very complete n-simplex in a labelled K_d we extend K_d to another pseudomanifold in such a way that a very complete boundary n-simplex of this extended pseudomanifold can easily be found. This boundary simplex can serve as the first element of the sequence of very complete n-simplexes mentioned in basic theorem 2.3.3.

DEFINITION 3.3.1.

$K_d * \{0\}$ denotes the (n+1)-complex defined by

$$K_d * \{0\} := \{\tau \mid \tau \in K_d,\ \tau \in \{0\} \text{ or } \tau = \sigma \cup \{0\} \text{ with } \sigma \in K_d\}$$

THEOREM 3.2.2.

$K_d * \{0\}$ is an (n+1)-pseudomanifold.

PROOF: Let $\tau' \in K_d * \{0\}$, then *either* $\tau' \in K_d$ and therefore a face of an n-simplex $\sigma \in K_d$ and so also a face of the (n+1)-simplex $\sigma \cup \{0\} \in K_d * \{0\}$, *or* $\tau' = \{0\}$, so a face

of every (n+1)-simplex of $K_d * \{0\}$, *or* there is a $\sigma' \in K_d$ such that $\tau' = \sigma' \cup \{0\}$. In the last case an n-simplex $\sigma \in K_d$ exists with σ' as a face, and therefore τ' is a face of the (n+1)-simplex $\sigma \cup \{0\} \in K_d * \{0\}$. So in all possible cases τ' is a face of an (n+1)-simplex $\tau \in K_d * \{0\}$.

Now let τ' be an n-simplex of $K_d * \{0\}$. Then *either* $\tau' \in K_d$ *or* there is an (n-1)-simplex $\sigma' \in K_d$ such that $\tau' = \sigma' \cup \{0\}$. If $\tau' \in K_d$ then $\tau' \cup \{0\}$ is the only (n+1)-simplex of $K_d * \{0\}$ containing τ' and so τ' is a boundary simplex of $K_d * \{0\}$. In the second case, $\tau' = \sigma' \cup \{0\}$, this σ' is *either* a boundary simplex of K_d and a face of only one n-simplex $\sigma \in K_d$, implying that τ' is a boundary simplex of $K_d * \{0\}$ and a face of $\sigma \cup \{0\}$, *or* σ' is an interior simplex and a face of two n-simplexes σ_1 and $\sigma_2 \in K_d$. In this case τ' is an interior n-simplex of $K_d * \{0\}$ and a face of both $\sigma_1 \cup \{0\}$ and $\sigma_2 \cup \{0\}$. □

THEOREM 3.3.3.

Given $K_d * \{0\}$ and $\ell: K_d^0 \cup \{0\} \to$ S-S, let τ_0' be the only very complete boundary n-simplex of $K_d * \{0\}$ that contains 0, then there is a unique sequence $\tau_0', \tau_1', \ldots, \tau_k'$ of distinct consecutively adjacent very complete n-simplexes in $K_d * \{0\}$. This sequence terminates with a very complete n-simplex of K_d.

PROOF: Since $K_d * \{0\}$ is a finite pseudomanifold and τ_0' the only very complete boundary n-simplex containing 0, this follows immediately from theorem 2.3.3. □

We adept from Eaves [4] an example of a labelling that fullfils the conditions of theorem 3.3.3.
Given a boundary (n-1)-simplex σ_0' of K_d and some interior

point x of S the labelling

$$\ell(u) = \begin{cases} \frac{u}{d} - f(\frac{u}{d}) & \text{if } u > 0 \\ \frac{u}{d} - x & \text{if } u \not> 0 \text{ but } u \neq 0 \\ -\rho \sum_{u \in \sigma_0'} (\frac{u}{d} - x) & \text{if } u = 0, \text{ where } \rho \text{ is the largest scalar such that } \ell(0) \in C(\{e^i - x \mid i = 0, \ldots n\}) \end{cases}$$

effectuates $\tau_0' := \sigma_0' \cup \{0\}$ to be the only very complete boundary simplex of $K_d * \{0\}$ containing 0.

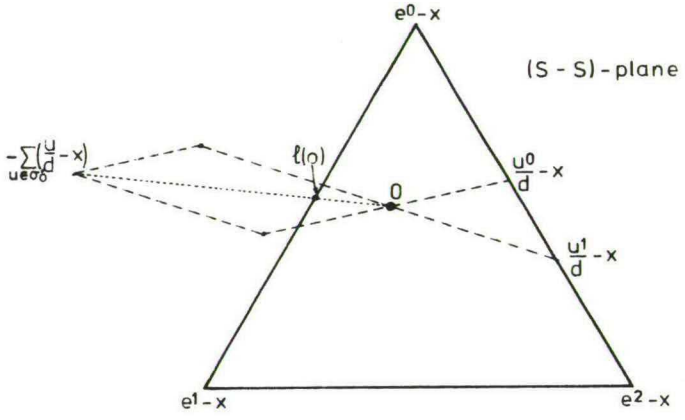


figure 3. $\sigma_0' = \{u^0, u^1\} \in K_d$

Anyway, once given a labelling and such a unique τ_0', face of τ_0, we are able to compute the sequence $\tau_0', \tau_1', \ldots, \tau_k'$ in the sense of theorem 3.3.3. and the corresponding sequence $\tau_0, \tau_1, \ldots, \tau_{k-1}$, where τ_i is the (n+1)-simplex of $K_d * \{0\}$ containing τ_i' and τ_{i+1}'. In fact by computing in turn τ_i' from τ_{i-1}' and τ_{i-1} by means of the method described in 2.3.2. (deleting a vertex) and τ_i from τ_{i-1} and τ_i' according to table 1 (adding a vertex). When in the k^{th} iteration of this procedure τ_k' does not contain 0, our algorithm terminates and

τ_k' is a very complete n-simplex of K_d.

3.4. AN ALGORITHM BASED ON AN INTEGER LABELLING

In this section we shall pay attention to a special labelling ℓ leading to a great simplification in actual computations. A computer program for the determination of a very complete simplex in K_d with this labelling will been given in the next section.

Let for i = 0,...,n ℓ^i be the given i^{th} column of the (n+1)×(n+1)-matrix

$$L: = \begin{bmatrix} +1 & -1 & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & +1 & -1 & & & & \cdot \\ \cdot & 0 & +1 & \cdot & & & \cdot \\ \cdot & \cdot & 0 & & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & & \cdot & 0 \\ 0 & \cdot & \cdot & & & & -1 \\ -1 & 0 & 0 & \cdot & \cdot & \cdot & +1 \end{bmatrix} .$$

Then $\ell: K_d^0 \cup\{0\} \to$ S-S is defined by $\ell(u) = \ell^i$, where i is the smallest integer such that

$$\begin{cases} \frac{u_i}{d} \leq f_i(\frac{u}{d}) & \text{if } u > 0 \\ u_i = 0 & \text{if } u \not> 0 \end{cases}$$

Evidently $\ell(0) = \ell^0$ and a simplex $\tau \in K_d * \{0\}$ is very complete if and only if $\ell(\tau) = \{\ell^0,\ldots,\ell^n\}$. Since $\ell(K_d * \{0\})$ contains only n+1 elements it follows immediately that an (n+1)-simplex τ is very complete if and only if there are unique u and v $\in \tau$ such that $\ell(u) = \ell(v)$. Its very complete

faces are $\tau\backslash\{u\}$ and $\tau\backslash\{v\}$.
The computation of τ'_i from τ'_{i-1} and τ_{i-1} can now be simplified. Instead of applying the method described in 2.3.2. we now only need to determine $u \in \tau'_{i-1}$ with $\ell(u) = \ell(v)$, where $\{v\} = \tau_{i-1}\backslash\tau'_{i-1}$. Then $\tau'_i = \tau_{i-1}\backslash\{u\}$ is the other very complete face of τ_{i-1}. This leads to two more simplifications. Firstly, since for every $i \in \{0,1,\ldots,k-1\}$ $0 \in \tau'_i$ and $0 \in \tau_i$, we only need to compute the sequences $\sigma'_0,\sigma'_1,\ldots,\sigma'_{k-1}$ and $\sigma_0,\sigma_1,\ldots,\sigma_{k-1}$ in K_d, where $\sigma'_i := \tau'_i\backslash\{0\}$ and $\sigma_i := \tau_i\backslash\{0\}$. Then σ_{k-1} appears to be τ'_k, the very complete simplex of K_d we look for. (See figure 4). Secondly we don't need the use of $\ell(u)$ but can confine ourselves to the function $m: K_d^0 \to \{0,\ldots,n\}$ defined by $m(u) = i$ if $\ell(u) = \ell^i$, or more directly by $m(u) = i$, where i is the smallest integer such that

$$\begin{cases} u_i = 0 & \text{if } u \neq 0 \\ \dfrac{u_i}{d} \leq f_i\left(\dfrac{u}{d}\right) & \text{if } u > 0 \end{cases} . \qquad (3.4.1.1.)$$

In fact m is the well-known integer labelling that has hitherto played an important role in proofs of Brouwer's theorem and in most fixed point algorithms.
These considerations suggest our simplified algorithm:

1. Start with τ', u^* and the boundary simplex $\sigma' = \tau'\backslash\{u^*\}$ as given below. $m(u^*)$ is known.
2. Determine j such that $u^j \in \sigma'$ and $m(u^j) = m(u^*)$.
3. Compute new τ' according to the rules of table 1.
4. Compute new u^* according to table 1 and let σ' be $\tau'\backslash\{u^*\}$.
5. Compute $m(u^*)$ according to (3.4.1.1.). If $m(u^*) = 0$ then τ' is a very complete simplex of K_d. If $m(u^*) \neq 0$ then continue with 2.

Clearly we need to specify the initial data. Let $u^0 := (d-n+2,1,1,\ldots,1,0,0)$ and $p := (n-1,n-2,\ldots,1,n)$, $n \geq 2$. Then $\tau' \overset{\sim}{\sim} (u^0,P)$, $u^* = u^n$ and $m(u^*) = n-1$. In the first iteration $j = 0$.

It is easy to see that σ' is the only boundary simplex of K_d with $m(\sigma') = \{1,\ldots,n\}$ and therefore $\sigma' \cup \{0\}$ has the required properties of the starting simplex σ_0 of theorem 3.3.3.

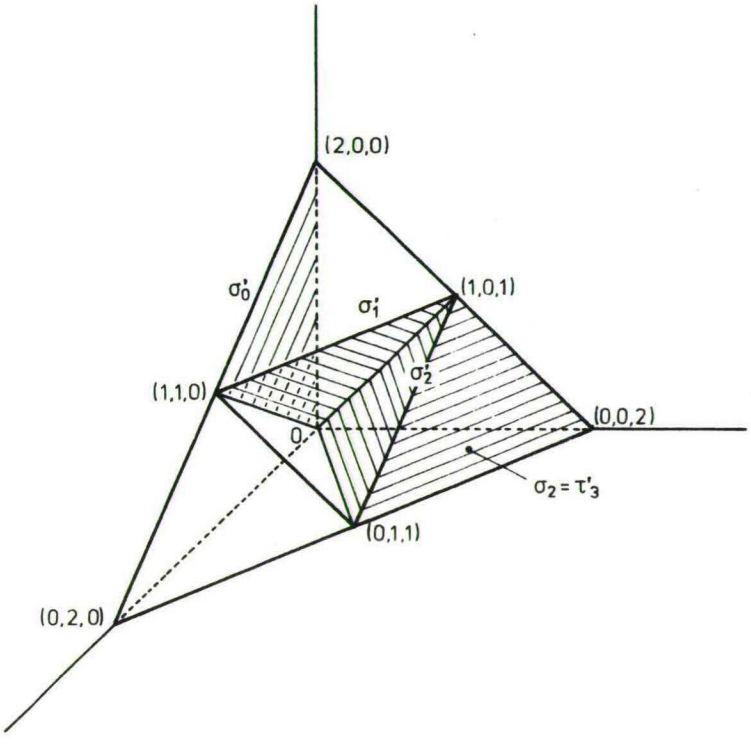


figure 4

$\tau_0' = \sigma_0' \cup \{0\}$; $\tau_1' = \sigma_1' \cup \{0\}$; $\tau_2' = \sigma_2' \cup \{0\}$;

$\sigma_0 = \sigma_0' \cup \sigma_1' = ((2,0,0),(1,2))$; $\sigma_1 = \sigma_1' \cup \sigma_2' = ((1,1,0),(2,1))$;

$\tau_3' = \sigma_2 = ((1,0,1),(1,2))$; $\tau_0 = \sigma_0 \cup \{0\}$; $\tau_1 = \sigma_1 \cup \{0\}$; $\tau_2 = \sigma_2 \cup \{0\}$.

3.5. AN ALGOL-60 PROGRAM

```
procedure Apprfp(n,d)Function:(f)
          The output parameters are a vertex:(u) and
          a permutation:(p);
value n,d;integer n,d;integer array u,p;real procedure f;
begin integer i,n1,old,new,pi;real h;
      integer array m[0:n];real array x[0:n];
      integer procedure label;
      begin for i:=0 step 1 until n do
            if x[i]=0 then go to ready;
            for i:=0 step 1 until n do
            if x[i]≤f(n,i,x) then go to ready;
    ready: label:=i
      end of label;
      comment initialisation;
      n1:=n-1; h:=1/d; old:=0;
      for i:=1 step 1 until n1 do
      begin u[i]:=1, m[i-1]:=p[i]:=n-i end;
      u[0]:=d-n+2; u[n1]:=u[n]:=0; m[n1]:=p[n]:=n; m[n]:=n1;
start:if old=0 then
      begin comment case 1; new:=n; pi:=p[1];
            u[pi-1]:=u[pi-1]-1; u[pi]:=u[pi]+1;
            for i:=1 step 1 until n1 do
            begin p[i]:=p[i+1]; m[i-1]:=m[i] end;
            p[n]:=pi; m[n1]:=m[n]
      end case 1 else
      if old≠n then
      begin comment case 2; new:=old; pi:=p[old];
            p[new]:=p[old+1]; p[new+1]:=pi
      end case 2 else
      begin comment case 3; new:=0; pi:=p[n];
            u[pi-1]:=u[pi-1]+1; u[pi]:=u[pi]-1;
            for i:=n step -1 until 2 do
            begin p[i]:=p[i-1]; m[i]:=m[i-1] end;
            p[1]:=pi; m[1]:=m[0]
      end case 3;
      comment computation of new x in S;
      for i:=0 step 1 until n do x[i]:=u[i]×h;
      for i:=1 step 1 until new do
      begin pi:=p[i]; x[pi-1]:=x[pi-1]-h; x[pi]:=x[pi]+h end;
      comment computation and test of the new integer label;
      m[new]:=label;
      if m[new]=0 then go to complete;
      for old:=0 step 1 until n do
      if m[new]=m[old]∧new≠old then go to start;
complete:
end of Apprfp;
```

Chapter 4 AN ALGORITHM FOR A SEQUENCE OF SUBDIVISIONS

4.1. INTRODUCTION

When the algorithm of the preceeding chapter has provided us with a very complete simplex in a labelled K_d it enables us to compute an approximated fixed point, for instance by means of a linear extension. However the approximation may be less accurate than wanted. Then we could take a larger d to obtain a smaller very complete simplex. But there are two disadvantages. Firstly, we would have to start the algorithm again at the very beginning, and secondly, the number of iterations would be considerably larger, especially for large n, since the number of n-simplexes in K_d is equal to d^n. Fortunately a method exists to use the computed very complete simplex in K_d in order to obtain another one in K_{2d}. Furthermore this generally leads to quite a reduction in the number of iterations to obtain such a simplex, especially for large n. This method only uses those K_d's with $d \in D := \{2^n \mid n \in N \cup \{0\}\}$. In order to link these pseudomanifolds $K_1, K_2, K_4, \ldots$ we make use of the pseudomanifold T.

4.2. CONNECTING TWO SUBDIVISIONS INTO A PSEUDOMANIFOLD T

DEFINITION 4.2.1.

We call T the (n+1)-complex containing the faces of all simplexes

$$\tau := \{v^0, v^1, \ldots, v^{n+1}\} \subset T^0 := K_1^0 \cup K_2^0$$

that can be described by means of

a vertex v^0 and a permutation $q = (q_1, \ldots, q_n, q_{n+1})$

of $(1,\ldots,n,n+1)$ in such a way that the other vertices of τ can be computed recursively from v^0 by

$$v^i := v^{i-1} + c(q_i) \qquad i = 1,\ldots,n+1,$$

where $c(q_i)$ is the q_i^{th} column of the $(n+1)\times(n+1)$-matrix

$$C' := [C,-e^n] = \begin{bmatrix} -1 & 0 & . & . & . & 0 & 0 \\ +1 & -1 & & & & . & . \\ 0 & +1 & . & & & . & . \\ . & 0 & & . & & . & . \\ . & . & & & . & 0 & . \\ . & . & & & & -1 & 0 \\ 0 & 0 & . & . & . & +1 & -1 \end{bmatrix}.$$

Again we write $\tau \underset{\sim}{\sim} (v^0,q)$. Note that if $q_m = n+1$ then $v^i \in K_2^0$ for $i = 0,\ldots,m-1$ and $v^i \in K_1^0$ for $i = m,\ldots,n+1$.

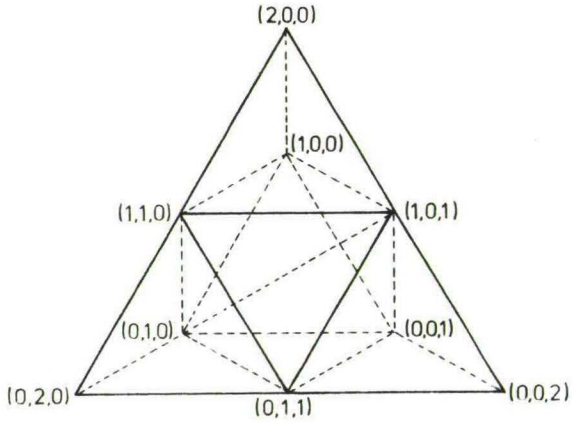


figure 5. T for n = 2.

e.g. $\{(1,1,0),(1,0,1),(0,1,1),(0,1,0)\} \underset{\sim}{\sim} ((1,1,0),(2,1,3))$

THEOREM 4.2.2.

T is an (n+1)-pseudomanifold.

The proof of this theorem is in analogy with the proof that each K_d is a pseudomanifold. (Theorem 3.2.3.). If we take an n-simplex $\tau' \in T$, being a face of an (n+1)-simplex say $\tau \approx (v^0,q) \in T$, there are also considered three cases for $\{u^k\} := \tau \setminus \tau'$. In each of the three cases the existence of unique v^*, z^0 and s such that $\tau_* := \tau' \cup \{v^*\} \approx (z^0,s)$ is different from τ can be shown in a similar way. If $v^* \in T^0$ then τ' is an interior simplex, else a boundary simplex of T.

The rules for the computation of $\tau_* \approx (z^0,s)$ and v^* are comprimed in

Table 2.

k	z^0	s	v^*
k=0	$v^0+c(q_1)$	$(q_2,..,q_{n+1},1)$	$z^{n+1}=v^{n+1}+c(q_1)$
$1 \leq k \leq n$	v^0	$(q_1,..,q_{k-1},q_{k+1},q_k,..,q_{n+1})$	$z^k=v^{k-1}+c(q_{k+1})$
k=n+1	$v^0-c(q_{n+1})$	$(q_{n+1},q_1,..,q_n)$	$z^0=v^0-c(q_{n+1})$

Remembering that K_d is n-dimensional and T is (n+1)-dimensional, it is clear that this table 2 is exactly the same as table 1.

LEMMA 4.2.3.

Let $\tau \approx (v^0,q)$ be an (n+1)-simplex of T and $\tau^k := \tau \{v^k\}$. Then τ^k is a boundary simplex of T if and only if one of the following statements is true:

(1) $\tau^k \in K_1$.

(2) $\tau^k \in K_2$.

(3) There is a coordinate j such that $v_j = 0$ for all $v \in \tau^k$.

PROOF: First let τ^k be a boundary simplex of T. Then v^* computed according to table 2 is not in T^0. This means that either $ev^* < 1$ or $ev^* > 2$ or $1 \leq ev^* \leq 2$ and $v_j^* < 0$ for some j. A short look at C' tells that $|ev^*-ev| \in \{0,1\}$ for all $v \in \tau^k$. It follows that if $ev^* < 1$ then $\tau^k \in K_1$ and if $ev^* > 2$ then $\tau^k \in K_2$. In case $1 \leq ev^* \leq 2$ and $v_j^* < 0$ for some j another look at C' shows that $|v_j^*-v_j| \in \{0,1\}$ for all $v \in \tau^k$. So $v_j = 0$ for all $v \in \tau^k$.

Now let (1), (2) or (3) be true.

If (1) holds then $\tau^k = \{e^0,\ldots,e^n\}$, the only n-simplex of K_1. By simple reasoning one finds $v^0 = e^0 + e^n$, $q = (n+1,1,\ldots,n)$ and $k = 0$ to be the only possibility for $\tau^k \subset \tau \approx (v^0,q) \in T$, since computing v^* according to table 2 leads to $v^* = v^{n+1} + c(n+1) = e^n-e^n = 0 \notin T^0$.

If (2) holds then τ^k can be represented as (u^0,p), where $u^0 \in K_2^0$ and $p = (p_1,\ldots,p_n)$ a permutation of $(1,\ldots,n)$. Furthermore $ev = 2$ for every $v \in \tau^k$. $v^0 = u^0$, $q = (p_1,\ldots,p_n,n+1)$ and $k = n+1$ fullfil $\tau^k \subset \tau \approx (v^0,q) \in T$ and computing v^* according to table 2 leads to $v^* = v^0-c(n+1) \notin T^0$ since $ev^* = ev^0-e.c(n+1) = 3$.

If (3) holds then $v_j^k > 0$, for from C' we see that no coordinate has the same value for all vertices of an (n+1)-simplex of T. From the fact that $v_j^i = 0$ for $i \neq k$ and $v_j^k > 0$ we conclude:

if $k = 0$, so $v^k = v^0 = v^1-c(q_1)$, then $q_1 = j+1$ and computation of v^* leads to $v^{*1} = v^n+c(q_1) = v^n+c(j+1) \notin T^0$ since $v_j^* = v_j^n - 1 < 0$.

if $1 \leq k \leq n$, so $v^k = v^{k+1}-c(q_{k+1})$, then $q_{k+1} = j+1$ and computation of v^* leads to $v^* = v^{k-1}+c(q_{k+1}) = v^{k-1}+c(j+1) \notin T^0$ since $v_j^* = v_j^{k-1} - 1 < 0$.

if $k = n+1$, so $v^k = v^{n+1} = v^n + c(q_{n+1})$, then $q_{n+1} = j$ and computation of v^* leads to $v^* = v^0 - c(q_{n+1}) = v^0 - c(j) \notin T^0$ since $v^*_j = v^0_j - 1 < 0$.

Hence if (1), (2) or (3) hold τ is the only $(n+1)$-simplex of T containing τ^k and so τ^k is a boundary simplex of T. □

From this proof we derive:

$$\begin{cases} v^* = 0 \Leftrightarrow \tau^k \in K_1 \\ ev^* = 3 \Leftrightarrow \tau^k \in K_2 \\ v^*_j = -1 \Leftrightarrow v_j = 0 \quad \text{for all } v \in \tau^k \end{cases} \qquad (4.2.3.1.)$$

Furthermore it is easily seen that no other case occurs with respect to boundary simplexes and that the three cases in (4.2.3.1.) exclude each other.

4.3. THE CONSTRUCTION OF NEW PSEUDOMANIFOLDS BY MEANS OF A MAP ON THE VERTICES OF T.

In the preceeding section we have introduced the pseudomanifold T in order to link the pseudomanifolds K_d ($d \in D$). To use this T to combine the K_d's of this sequence $K_1, K_2, K_4, \ldots$ into a new pseudomanifold K_* we need the notion of a map h_σ for a given σ of some K_d.

DEFINITION 4.3.1.

Let $d \in D$ and $\sigma \approx (u^0, p) \in K_d$ then we define the map

$$h_\sigma : T^0 \to \sigma \cup (\sigma + \sigma) \qquad \text{by} \qquad h_\sigma(v) = \sum_{i=0}^{n} v_i u^i .$$

THEOREM 4.3.2.

$h_\sigma: T^0 \to \sigma \cup (\sigma+\sigma)$ is a bijection.

PROOF: Assume v^1 and $v^2 \in T^0$ such that $h_\sigma(v^1) = h_\sigma(v^2)$. Then $\sum_{i=0}^{n} (v^1_j - v^2_i)u^i = 0$. In consequence of the structure of the matrix C the vertices $u^0,\ldots,u^n$ are linearly independent; hence $v^1 = v^2$. So h_σ is injective. To prove that h_σ is also surjective, let $w \in \sigma \cup (\sigma+\sigma)$. If $w \in \sigma$, say $w = u^i$, then $w = \sum_{i=0}^{n} v_i u^i$ with $v = e^i \in K_1 \subset T^0$. If $w \in \sigma+\sigma$, say $w = u^i+u^j$, then $w = \sum_{i=0}^{n} v_i u^i$ with $v = e^i+e^j \in K_2 \subset T^0$. □

DEFINITION 4.3.3.

$h_\sigma(T)$ denotes the (n+1)-complex defined by
$h_\sigma(T) := \{h_\sigma(\tau) \mid \tau \in T\}$.

THEOREM 4.3.4.

$h_\sigma(T)$ is an (n+1)-pseudomanifold.

PROOF: Follows immediately from theorems 4.2.2. and 4.3.2. □

Since h_σ is a bijection the properties like dimension, face, adjacent, boundary or interior, are lifted from T to $h_\sigma(T)$.

LEMMA 4.3.5.

Let $\tau \in T$ and let J be the collection of all indexes j such that there is a $v \in \tau$ with $v_j > 0$. Further let $d \in D$, $\sigma \approx (u^0,p) \in K_d$ and $\sigma' := \{u^j \mid j \in J\}$.
Then

$$h_\sigma(\tau) \subset \sigma' \cup (\sigma'+\sigma')$$

and

$$h_\sigma(\tau) \subset \sigma'' \cup (\sigma''+\sigma'') \Rightarrow \sigma'' \supset \sigma' (\sigma'' \in K_d).$$

(Verbally: in K_d σ' is the minimal set with this inclusion property).

PROOF: Let $w \in h_\sigma(\tau)$. Then there is a $v \in \tau$ such that $w = \sum_{j=0}^{n} v_j u^j$. If $j \notin J$ then $v_j = 0$, so $w = \sum_{j \in J} v_j u^j \in \sigma' \cup (\sigma'+\sigma')$.
Now assume $\sigma'' \in K_d$ is such that $h_\sigma(\tau) \subset \sigma'' \cup (\sigma''+\sigma'')$ and $u^k \in \sigma'$, then $k \in J$ and there is a $v \in \tau$ with $v_k > 0$. Then either (if ev = 1) $h_\sigma(v) = u^k$ and, since $h_\sigma(v) \in \sigma'' \cup (\sigma''+\sigma'')$, $u^k \in \sigma''$, or (if ev = 2) a $u^l \in \sigma'$ exists such that $h_\sigma(v) = u^k+u^l$. But $h_\sigma(v) \in \sigma'' \cup (\sigma''+\sigma'')$, so $u^k+u^l \in (\sigma''+\sigma'')$. This means that there are w^a and $w^b \in \sigma''$ such that $u^k+u^l = w^a+w^b$. Without loss of generality we may assume that w^a is lexicographically smaller than w^b and that $k < l$. Then there are P and Q $\subset\{0,\ldots,n\}$ such that $u^l = u^k \sum_{i \in P} c(i)$ and $w^b = w^a + \sum_{i \in Q} c(i)$. So $2u^k + \sum_{i \in P} c(i) = 2w^a + \sum_{i \in Q} c(i)$. Then $g := \sum_{i \in P} c(i) - \sum_{i \in Q} c(i) = 2(w^a - u^k)$ has only even coordinates. Suppose $g \neq 0$ then $R := (P \cup Q) \setminus (P \cap Q) \neq \phi$. Let $i_* := \min R$, then the $(i_*-1)^{th}$ coordinate of all $c(i)$ with $i \in R$ except $c(i^*)$ is equal to zero. Therefore $g_{i_*-1} = -1$ or $+1$. A contradiction, since g_i is even for all i. So $g = 0$ and $u^k = w^a \in \sigma''$. Conclusion: $\sigma'' \supset \sigma'$. □

COROLLARY 4.3.6.

If $J = \{0,\ldots,n\}$ only one $\xi \in K_d$ exists for which $h_\sigma(\tau) \in h_\xi(T)$, namely σ itself. It can easily be verified that

$J = \{0,\ldots,n\}$ if and only if τ is either an $(n+1)$-simplex or an interior n-simplex or an n-simplex of $K_1 \cup K_2$.

4.4. CONNECTING A SEQUENCE OF SUBDIVISIONS

DEFINITION 4.4.1.

K_* is defined to be the $(n+1)$-complex containing all simplexes $\phi = h_\sigma(\tau)$, where σ is an n-simplex in $\bigcup_{d\in D} K_d$ and $\tau \in T$. Its set of vertices is $K^0_* := \bigcup_{d\in D} K^0_d$.

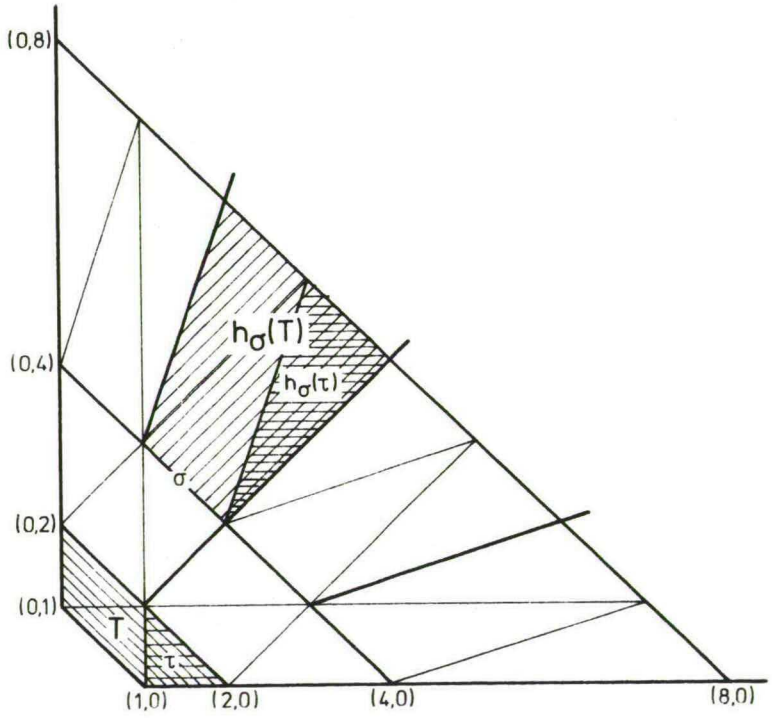


figure 6. K_* for $n = 1$.

e.g. $\tau \approx ((2,0),(1,2))$, $\sigma \approx ((2,2),(1))$ then
$h_\sigma(\tau) = ((4,4),(3,5),(2,2))$.

Lemma 4.3.5. implies that for an (n+1)-simplex $\phi \in K_*$ its representation $h_\sigma(\tau)$ is unique. For firstly τ is an (n+1)-simplex too and secondly the (n+1)-simplex ϕ has vertices at two levels, say K_d and K_{2d}, so only K_d contains such a unique σ with $\phi \subset \sigma \cup (\sigma+\sigma)$. But then τ is also unique because h_σ is a bijection.

THEOREM 4.4.2.

K_* is a pseudomanifold.

PROOF: Let $\phi' = h_\sigma(\tau')$ be a simplex of K_*, then τ' is a face of an (n+1)-simplex $\tau \in T$ and so ϕ' is a face of the (n+1)-simplex $h_\sigma(\tau) \in K_*$.
Now let ϕ be an (n+1)-simplex of K_*. Then there are unique $d \in D$, $\tau \approx (v^0,q) \in T$ and $\sigma \approx (u^0,p) \in K_d$ such that $\phi = h_\sigma(\tau)$. Let $\tau^k := \tau \setminus \{v^k\}$, $\phi^k := h_\sigma(\tau^k)$ and $\sigma^j := \sigma \setminus \{u^j\}$ for $j = 0,\ldots,n$. We will demonstrate that, except of ϕ, ϕ^k is a face of at most one other (n+1)-simplex of K_*.

A. First let ϕ^k be an interior simplex of $h_\sigma(T)$, and so τ^k an interior simplex of T. Then, since some vertices of τ^k are in K_1^0 and others in K_2^0, there is only one K_i that contains simplexes ξ such that $\phi^k \subset \xi \cup (\xi+\xi) = h_\xi(T^0)$. According to corollary 4.3.6. this ξ is unique and evidently equal to σ. So every (n+1)-simplex of K_* with ϕ_k as a face is in $h_\sigma(T)$. In $h_\sigma(T)$ there are two of them, ϕ and one other, say $h_\sigma(T_*)$. So ϕ^k is an interior simplex of K_*. Needless to say that τ_* can be computed according to table 2.

B. Secondly ϕ^k can be a boundary simplex of $h_\sigma(T)$. Then in $h_\sigma(T)$ it is a face of only ϕ. But there may be (n+1)-simplexes $\phi_* = h_{\sigma_*}(\tau_*)$ with ϕ^k as a face in other pseudomanifolds $h_{\sigma_*}^*(T)$. In that case a face τ_*' of τ_* exists such that $\phi^k = h_{\sigma_*}^*(\tau_*')$. If ϕ^k is a boundary simplex of $h_\sigma(T)$ then τ^k is one of T and in consequence of lemma 4.2.3. we consider three cases:

B.1. $\tau^k \in K_1$, so $\phi^k \in K_d$. Since eu = d for every $u \in K_d$, for ϕ^k being $h_{\sigma_*}(\tau'_*)$ there are only two possibilities: *either* $\tau'_* \in K_1$ and $\sigma_* \in K_d$, in which case $\sigma_* = \sigma$ in view of corollary 4.3.6., *or* $\tau'_* \in K_2$ and $\sigma_* \in K_{\frac{1}{2}d}$. We see that if d = 1 ϕ^k is a boundary simplex of K_*. If d > 1 then corollary 4.3.6. implies that there is only one $\sigma_* \in K_{\frac{1}{2}d}$ such that $\phi^k \in h_{\sigma_*}(T)$. Further $\tau'_* \in K_2$ is a face of only one (n+1)-simplex of T, say τ_*. So except ϕ there is only one (n+1)-simplex $h_{\sigma_*}(\tau_*) \in K_*$ with ϕ^k as a face. The existence of such a simplex and even more how to compute this simplex is shown after the proof of this theorem. So, if d > 1 then ϕ^k is an interior simplex of K_*.

B.2. $\tau^k \in K_2$, so $\phi^k \in K_{2d}$. Now *either* $\tau'_* \in K_2$ and $\sigma_* \in K_d$, in which case $\sigma_* = \sigma$, *or* $\tau'_* \in K_1$ and $\sigma_* \in K_{2d}$. As in case 1, σ_* and τ_* are unique. Their existence and how to compute them is also shown after the proof of this theorem. So again ϕ_k is an interior simplex of K_*.

B.3. There is an index j with $v_j = 0$ for every $v \in \tau^k$. It can be proved that j is unique, for suppose $1 \neq j$ and $v_1 = v_j = 0$ for all $v \in \tau^k$. Then from the structure of the matrix C' it follows that no coordinate has the same value for all vertices of an (n+1)-simplex of T, so $v^k_j > v_j = 0$ and $v^k_1 > v_1 = 0$ for all $v \in \tau^k$. But, if k = 0 then $v^k = v^1 - c(q_1)$ in contradiction with $v^k_j > v^1_j$ and $v^k_1 > v^1_1$, and if k > 0 then $v^k = v^{k-1} + c(q_k)$ in contradiction with $v^k_j > v^{k-1}_j$ and $v^k_1 > v^{k-1}_1$. So indeed j is unique.
Further we note that, since some vertices of τ^k are in K^0_1 and others in K^0_2, there is no other K_i than K_d containing simplexes ξ such that $\phi^k \subset \xi \cup (\xi+\xi)$. Then in consequence of lemma 4.3.5. $\phi^k \subset \sigma^j \cup (\sigma^j+\sigma^j)$ and all other simplexes ξ for which $\phi^k \subset \xi \cup (\xi+\xi)$ contain σ^j as a face.
If σ^j is a boundary simplex of K_d then σ is the only n-simplex of K_d containing σ^j, which implies that no other $\sigma_* \neq \sigma$ exists such that $\phi^k \in h_{\sigma_*}(T)$. So, if σ^j is a boundary simplex of K_d, ϕ^k is one of K_*. If otherwise σ^j

is an interior simplex of K_d, then it is a face of one other n-simplex $\sigma_* \neq \sigma$. So $\phi_k \in h_{\sigma_*}(T)$, say $\phi_k = h_{\sigma_*}(\tau'_*)$. τ'_* is not an interior simplex of T, for in analogy with part A of this of this proof this assumption would imply $\sigma_* = \sigma$, a contradiction. So τ'_* is a boundary n-simplex of T and a face of only one (n+1)-simplex $\tau_* \in T$. Therefore ϕ^k is a face of $h_{\sigma_*}(\tau_*) \neq \phi$ and is an interior simplex of K_*. The computation of σ_* and τ_* is given below. □

4.4.3.

In order to construct a sequence of n-simplexes in K_* we make use of an exchange procedure. The first part of each iteration in this procedure is the computation of the index k of the vertex to be deleted from the (n+1)-simplex $\phi = h_\sigma(\tau)(\sigma \approx (u^0,p) \in K_d,\ \tau \approx (v^0,q) \in T,\ \tau^k := \tau\setminus\{v^k\},\ \phi^k := h_\sigma(\tau^k) \subset \phi)$. This is described in 2.3.2. The second part of each iteration, i.e. the construction of the other (n+1)-simplex $\phi_* = h_{\sigma_*}(\tau_*)(\sigma_* \approx (y_0,r),\ \tau_* \approx (z^0,s))$ that contains ϕ^k, will be described below.

First compute v^* according to table 2 (page 28).

A. If $v^* \in T^0$ then ϕ^k is an interior simplex of $h_\sigma(T)$, and $\phi_* = h_{\sigma_*}(\tau_*)$ where $\sigma_* = \sigma$ and τ_* is as given in table 2.

B. If $v^* \notin T^0$ then ϕ^k is a boundary simplex of $h_\sigma(T)$ and there are three cases as seen in the proof of theorem 4.4.2. Which of them is actual tells v^* (see (4.2.3.1.)).

B.1. $v^* = 0$. Then $\tau^k \in K_1$ so $\phi^k = \sigma$, $k = 0$, and $q_1 = n+1$. If $d = 1$ ϕ_* does not exist. If $d > 1$ compute $\sigma_* \approx (y^0,r)$ and $\tau_* \approx (z^0,s)$ according to the following scheme:
Define a sequence $\{w^0,\ldots,w^n\}$ by

$$w^0 := u^0$$

$$w^i := \begin{cases} w^{i-1} & \text{if } w_{i-1}^{i-1} \text{ is even} \\ w^{i-1}-c(i) & \text{if } w_{i-1}^{i-1} \text{ is odd} \end{cases} \qquad i = 1,\ldots,n.$$

Then w^n has only even coordinates, so $y^0 := w^n/2$ is in $K_{\frac{1}{2}d}$. Define R_1 and R_2 by

$$i \in \begin{cases} R_1 & \text{if } w^i = w^{i-1}-c(i) \\ R_2 & \text{if } w^i = w^{i-1} \end{cases} \qquad i = 1,\ldots,n.$$

Let j be the number of elements in R_1, so n-j the number of elements in R_2. Now define the permutation r: = $= (r_1,\ldots,r_j,r_{j+1},\ldots,r_n)$ with $\{r_1,\ldots,r_j\} = R_1$ and $\{r_{j+1},\ldots,r_n\} = R_2$ both having the same order as in p. Define $z^0 := e^0+e^j$ and s: $(s_1,\ldots,s_{n+1})$ where $s_i = m$ if $p_i = r_m$ for $i = 1,\ldots,n$ and $s_{n+1} = n+1$. Then $\phi_* = h_{\sigma_*}(\tau_*)$, with $\sigma_* \approx (y^0,r)$ and $\tau_* \approx (z^0,s)$, is the unique (n+1)-simplex in K_* containing ϕ^k and differing from ϕ.

PROOF: $h_{\sigma_*}(z^0) = \sum_{i=0}^{n} z_i^0 y^i = y^0+y^j = \frac{w^n}{2} + \frac{w^n}{2} + \sum_{i=1}^{j} c(r_i) =$

$$= w^n + \sum_{i\in R_1} c(i) = w^0 = u^0.$$

By induction for $m = 1,\ldots,n$:

$$h_{\sigma_*}(z^m) = \sum_{i=0}^{n} z_i^m y^i = \sum_{i=0}^{n} z_i^{m-1} y^i + \sum_{i=0}^{n} c_i(s_m).y^i =$$

$$= h_{\sigma_*}(z^{m-1}) + y^{s_m - s_{m-1}} = u^{m-1}+c(r_{s_m}) =$$

$$= u^{m-1}+c(p_m) = u^m.$$

$$h_{\sigma_*}(z^{n+1}) = \sum_{i=0}^{n} z_i^{n+1} y^i = \sum_{i=0}^{n} z_i^n y^i + \sum_{i=0}^{n} c_i(s_{n+1}).y^i =$$

$$= h_{\sigma_*}(z^n) - y^{n+1} = u^n - y^{n+1} = u^0 + \sum_{i=0}^{n} c(p_i) - y^n =$$

$$= u^0 + \sum_{i=0}^{n} c(r_i) - y^n = u^0 - y^0 = y^j.$$

This y^j is an element of K_*^0 since from the computation of the sequence $\{w^0,\ldots,w^n\}$ it can be seen that $y_i^0 \leq u_i^0$ for every coordinate i.
So $\phi_* = \{h_{\sigma_*}(z^0),\ldots,h_{\sigma_*}(z^{n+1})\} = \{u^0,\ldots,u^n,y^j\} =$
$= \phi^k \cup \{y^j\} \neq \phi$ and $\phi_* \in K_*$. □

B.2. $ev^* = 3$. Then $\tau^k \in K_2$, $\phi^k \in K_{2d}$, $K + q_{n+1} = n+1$.
Define
$y^0 := h_\sigma(v^0)$, $r := (P_{q_1},\ldots,P_{q_n})$,
and
$z^0 := (1,0,\ldots,0,1)$, $s := (n+1,1,\ldots,n)$.
Then $\phi_* = h_{\sigma_*}(\tau_*)$, with $\sigma_* \approx (y^0,r)$ and $\tau_* \approx (z^0,s)$, is the unique (n+1)-simplex in K_* containing ϕ^k and differing from ϕ.

PROOF: $h_{\sigma_*}(z^1) = \sum_{i=0}^{n} z_i^1 y^i = y^0 = h_\sigma(v^0)$

and by induction for m = 1,...,n:

$$h_{\sigma_*}(z^{m+1}) = \sum_{i=0}^{n} z_i^{m+1} y^i = y^m = y^{m-1} + c(r_m) =$$

$$= y^{m-1} + c(p_{q_m}) = h_\sigma(v^{m-1}) + u^{q_m} - u^{q_m-1} =$$

$$= \sum_{i=0}^{n} v_i^{m-1} u^i + \sum_{i=0}^{n} c_i(q_m).u^i = \sum_{i=0}^{n} v_i^m u^i = h_\sigma(v^m).$$

$$h_{\sigma_*}(z^0) = \sum_{i=0}^{n} z_i^0 y^i = y^0 + y^n \in K^0_{4d} \subset K^0_*.$$

So ϕ_*: $= h_{\sigma_*}(\tau_*) = \{y^0+y^n, y^0, \ldots, y^n\} = \{y^0+y^n\} \cup \phi^k \neq \phi$ and $\phi_* \in K_*$.□

B.3. $v^*_j = -1$ for a unique index j. Then $v_j = 0$ for all $v \in \tau^k$. First compute $\sigma_* \approx (y^0, r)$ according to table 1 (pag. 18). Then compute $\tau_* \approx (z^0, s)$ according to table 3, where m is such that $q_m = n+1$.

TABLE 3.

j	z^0	s
j=0	$(v^1_1, .. v^1_n, 0)$	$(q_2-1, .., q_{m-1}-1, n, q_m, q_{m+1}-1, .., q_{n+1}-1)$
0<j<n	v^0	q
j=n	$(1, v^0_0-1, v^0_1, .., v^0_{n-1})$	$(1, q_1+1, .., q_{k-1}+1, n+1, q_{k+2}+1, .., q_{n+1}-1)$

Observe that if j = 0 then $q_1 = 1$ and k = 0. If j = n then $q_k = n$, $q_{k+1} = n+1$, $k \neq 0$ and $k \neq n+1$. Some computations will show that ϕ_*: $= h_{\sigma_*}(\tau_*) = \phi^k \cup \{w^*\} \neq \phi$, where

$$w^* = \begin{cases} h_{\sigma_*}(z^{m-1}) & \text{if} \quad j = 0 \\ h_{\sigma_*}(z^k) & \text{if} \quad 0 < j < n, \\ h_{\sigma_*}(z^0) & \text{if} \quad j = n \end{cases}$$

and that $\phi_* \in K_*$ if ϕ^k is an interior simplex of K_*.

As is seen in the proof of theorem 4.4.2. there are two cases

in which an n-simplex $\phi^k = h_\sigma(\tau^k)$ may be a boundary simplex of K_*. Firstly in case B.1., namely if d = 1. Then $\phi^k = \{e^0,\ldots,e^n\}$. Secondly in case B.3., namely if σ^j is a boundary simplex of K_d. Then an index i exists such that $u_i = 0$ for every $u \in \sigma^j$, implying that $w_i = 0$ for every $w \in \phi^k$.

4.5. THE ALGORITHM

Since we now have at our disposal a pseudomanifold, K_*, again we apply theorem 2.3.3. to state

THEOREM 4.5.1.

Given the (n+1)-pseudomanifold K_* and a labelling $\ell: K_*^0 \to$ S-S, let ϕ_0' be the only very complete boundary n-simplex of K_*. Then there is a unqiue sequence $\phi_0',\phi_1',\ldots$ of distinct very complete consecutively adjacent n-simplexes in K_*. This sequence is infinite.

PROOF: follows immediately from theorem 2.3.3. □

Now, if we have a labelling and know the unique ϕ_0', a face of ϕ_0, then we can compute the sequences $\phi_0',\phi_1',\ldots$ and $\phi_0,\phi_1,\ldots$, where $\phi_i = \phi_i' \cup \phi_{i+1}'$, as far as we want to. This indeed can be done by computing in turn ϕ_i' from ϕ_{i-1}' and ϕ_{i-1} according to 2.3.2., and ϕ_i from ϕ_{i-1} and ϕ_i' by the rules given in 4.4.3. It is obvious that we would like to start the algorithm with $\phi_0' = \{e^0,\ldots,e^n\}$. Then $\phi_0 = h_{\sigma_0}(\tau_0)$, where $\sigma_0 \approx (u^0,p) :=$ $= ((1,0,\ldots,0),(1,\ldots,n))$ and $\tau_0 \approx (v^0,q) :=$ $= ((1,0,\ldots,0,1),(n+1,1,\ldots,n))$. Hence we need a labelling such that this ϕ_0' is the only very complete boundary simplex of K_*. Such a labelling $\ell: K_*^0 \to$ S-S is given by

$$\ell(w) := \begin{cases} x-f(x) & \text{if } x_0 \neq 0 \text{ or } f_0(x) \neq 0 \\ \\ x-g(x,d) & \text{if } x_0 = f_0(x) = 0 \end{cases}, \quad (4.5.2.1.)$$

where $d = ew$, $x = \frac{w}{d} \in S$ and $g(x,d) = f(x) + \frac{\rho}{d}(e^0-f(x))$, a perturbation of $f(x)$. (ρ is a small positive number).
Given this labelling no boundary n-simplex $\phi' \in K_*$ with $w_j = 0$ for all $w \in \phi'$ is very complete. For if $j = 0$ then $\ell_0(w) < 0$ for all $w \in \phi'$ and if $j \neq 0$ then $\ell_j(w) \leq 0 < \varepsilon^j$ for all $w \in \phi'$ and all $\varepsilon > 0$. Contrary $\{e^0,\ldots,e^{n_j}\}$ is very complete, as is proved in appendix 2. There are no other types of boundary simplexes in K_*.
Hence in view of theorem 4.5.1. we conclude to the existence of a unique infinite sequence $\phi'_0,\phi'_1,\ldots$ of distinct very complete consecutively adjacent n-simplexes in K_*. From this sequence we now select a sub-sequence $\{\psi_d\} = \psi_1,\psi_2,\psi_4,\psi_8,\ldots$ with $\psi_d \in K_d$ and define a corresponding sequence $\{\xi_d\} = \xi_1,\xi_2,\xi_4,\ldots$ by $\xi_d := \{x^i | x^i = u^i/d \text{ and } u^i \in \psi_d\}$. These sequences are also infinite since every K_d is a finite pseudomanifold. Now for every $d \in D$ there is a $\lambda^d \in S$ such that $\Sigma_{i=0}^n \lambda_i^d \ell(u^i) = 0$, where $\{u^0,\ldots,u^n\} = \psi_d$. We even might infer that $\hat{x}^d := \Sigma_{i=0}^n \lambda_i^d x^i(\{x^0,\ldots,x^n\} = \xi_d)$ in fact is a fixed point of the continuous piecewise linear function f_d: S-S, where

$$f_d(x) = \begin{cases} x-\ell(dx) & \text{if } dx \in K_d^0 \\ \\ \sum_{i=0}^n \lambda_i f_d(\frac{u_i}{d}) & \text{if } dx \notin K_d^0, \text{ where } (u^0,p) \in K_d \text{ and} \\ & \lambda \in S \text{ such that } dx = \\ & = \sum_{i=0}^n \lambda_i u^i. \text{ (see [7], page 179).} \end{cases}$$

Since $\lim_{d\to\infty} f_d = f$, uniform on S, the cluster points of the sequence $\hat{x}^1,\hat{x}^2,x^4,\ldots$ are fixed points of f.

Chapter 5. AN ALGOL PROGRAM

5.1. INTRODUCTION

In this chapter a computer implementation of the algorithm of the preceeding chapter will be given. Since it is based particularly on 2.3.2. and 4.4.3. we only have to explain the symbols used in this program. However beforehand we should remark that numerical perfection has not been our prime intention. Suggestions for possible improvements will be treated in section 5.4.

5.2. THE SYMBOLS

Input parameters:

n : dimension of $S = \{x | x_i \geq 0, \Sigma_{i=0}^{n} x_i = 1\} \subset R^{n+1}$.

LABEL : procedure which maps a point x of S into a point $\ell := x - f(x)$ of S-S. In the program this procedure is called by LABEL (x,ℓ) where x and ℓ are real arrays with n+1 components.

INVERT: a procedure, called by INVERT (B,INVB), which maps the $(n+1)\times(n+1)$-matrix B into its inverse INVB.

eps : the ε-criterion; the program stops if a point of S with label ℓ is computed such that $\ell_i \leq \varepsilon$ for $i = 0,\ldots,n$. Such a point is called an ε-approximated fixed point.

limit : iteration limit; the program stops if after this number of iterations an ε-approximated fixed point is not found yet. This stopcriterion is necessary since by inaccurate computation of INVB we may encounter an n-simplex that is not very complete, in which case there is a theoretical chance of cycling.

Output parameters:

approx : a boolean variable which is set to true if an ε-approximated fixed point is computed or is set to false if the iteration limit is reached.

app : if approx is true then app is an ε-approximated fixed point in S, else app is the last computed approximation.

d : indicates that app has been computed by means of an n-simplex in k_d.

iter : the number of iterations executed. In each iteration one vertex is exchanged.

The correspondence between the most important symbols used in the preceeding chapter and those in the program is given in table 4.

TABLE 4

single variables		vectors		matrices	
k	old	u^0, y^0	u	B	B
α_i	alph, alphi	v^0, z^0	v	B^{-1}	INVB
m	m	v^*	vx		
d	d	p,r	p,r		
ρ/d	pert	q,s	q		
		x	x		
		$\ell(x)$	ℓ		
		$\hat{x}^d$	app		

The symbol new denotes the number of the vertex (the vertices of a simplex taken in a lexicographical order) to be deleted. ρ is given the value 0.01.

Finally we explain the behavior of the process in front of a pathological situation. Caused by rounding errors, par example

in the computation of INVB, a boundary simplex of K_* may be reached in contrast with the theoretical result about the uniqueness of a very complete boundary simplex. This may occur only in case B3. In this pathological situation the procedure FLIP is used in order to perform the computation of this iteration once again for an other value of old. Once again we mention the possibility of cycling caused by rounding errors.

5.3. THE PROGRAM

```
procedure FIXED POINT(n,LABEL,INVERT,eps,limit)
                                    output:(approx,app,d,iter);
value n,eps,limit;
integer n,limit,d,iter; real eps; array app; boolean approx;
procedure invert,label;
begin integer i,j,k,m,n1,pi,qi,new,old,jold;
      real appi,pert,min,alph,alphi,ratio,d2;
      integer array u,v,p,aid,pos[0:n],vx,q[0:n+1];
      real array x,l[0:n],B,BX,INVB[0:n,0:n]; boolean admin;

      procedure FLIP;
      begin pos[jold]:=old;INVB[jold,0]:=INVB[jold,0]-10;
            go to search
      end FLIP;

      procedure RECUR(flb,fub,slb,sub,sn);
      comment recursive computation of vx,see table 2;
      value flb,fub,slb,sub,sn; integer flb,fub,slb,sub,sn;
      begin for i:=0 step 1 until n do vx[i]:=v[i]; vx[n1]:=0;
            for i:=flb step 1 until fub,slb step 1 until sub do
            begin qi:=q[i]; vx[qi-1]:=vx[qi-1]-sn;
                  vx[qi]:=vx[qi]+sn
            end
      end RECUR;

      procedure WHICH CASE;
      begin if old=0 then
            begin if m=1 then begin CASE B1; go to ready end;
                  RECUR(1,1,1,0,1); vx[0]:=vx[0]-1
            end       else
            if old≠n1 then RECUR(1,old-1,old+1,old+1,1)
                      else
            begin comment old=n+1;
                  if m=n1 then begin CASE B2; go to ready end;
                  RECUR(1,0,n1,n1,-1)
            end of test on old;
            for j:=0 step 1 until n do if vx[j]<0 then
                              begin CASE B3; go to ready end;
```

```
      CASE A;
ready:
end WHICH CASE;

procedure CASE A;
begin if old=0 then
      begin new:=n1; qi:=q[1];
            v[qi-1]:=v[qi-1]-1; v[qi]:=v[qi]+1;
            for i:=1 step 1 until n do q[i]:=q[i+1];
            q[n1]:=qi; m:=m-1;
            for i:=0 step 1 until n do pos[i]:=pos[i]-1
      end      else
      if old≠n1 then
      begin new:=old; qi:=q[old];
            q[new]:=q[old+1]; q[new+1]:=qi;
            if m=old then m:=old+1 else
            if m=old+1 then m:=old
      end      else
      begin comment old=n1; new:=0; qi:=q[n1];
            v[qi-1]:=v[qi-1]+1; v[qi]:=v[qi]-1;
            for i:=n step -1 until 1 do q[i+1]:=q[i];
            q[1]:=qi; m:=m+1;
            for i:=0 step 1 until n do pos[i]:=pos[i]+1
      end
end CASE A;

procedure CASE B1;
begin integer iu,ir1,ir2; integer array r[1:n];
      d:=d÷2; j:=0;
      for i:=0 step 1 until n do if u[i]≠u[i]÷2×2 then
      begin j:=j+1; r[j]:=i+1; u[i+1]:=u[i+1]-1;
            u[i]:=u[i]÷2+1
      end   else u[i]:=u[i]÷2;
      iu:=1; ir2:=j+1;
      for i:=1 step 1 until n do
      begin for ir1:=1 step 1 until j do
            begin if p[i]=r[ir1] then
                  begin p[iu]:=r[ir1]; q[i]:=iu;
                        iu:=iu+1; go to go
                  end
            end;
            r[ir2]:=p[i]; q[i]:=ir2; ir2:=ir2+1;
      go:
      end;
      for i:=j+1 step 1 until n do p[i]:=r[i];
      for i:=0 step 1 until n do
      begin vx[i]:=v[i]:=0; pos[i]:=pos[i]-1 end;
      vx[j]:=v[j]:=v[0]:=1; q[n1]:=m:=new:=n1
end CASE B1;

procedure CASE B2;
begin approx:=true; d:=d+d; new:=0;
      if v[0]=1 then for i:=1,i+1 while v[i-1]=0 do
      begin pi:=p[i]; aid[pi-1]:=aid[pi-1]-1;
            aid[pi]:=aid[pi]+1
```

```
        end;
        for i:=0 step 1 until n do
        begin u[i]:=2×u[i]+aid[i]; aid[i]:=p[q[i]]; v[i]:=0
        end;
        v[0]:=v[n]:=m:=1;
        for i:=0 step 1 until n do
        begin p[i]:=aid[i]; aid[i]:=0; pos[i]:=pos[i]+1;
              q[i+1]:=i; vx[i]:=v[i]
        end;
        q[1]:=n1
end CASE B2;

procedure CASE B3;
begin comment if the computed n-simplex is a boundary
              simplex then the procedure FLIP is called;
        if j=0 then
        begin if u[p[1]-1]=0 ∨ (u[0]=1 ∧ p[1]=1) then FLIP;
              new:=m-1; v[0]:=v[1]+1;
              pi:=p[1]; u[pi-1]:=u[pi-1]-1; u[pi]:=u[pi]+1;
              for i:=1 step 1 until n-1 do
              begin v[i]:=v[i+1]; pi:=p[i+1] end;
              v[n]:=0; p[n]:=pi;
              for i:=1 step 1 until m-2 do q[i]:=q[i+1]-1;
              q[m-1]:=n;
              for i:=m+1 step 1 until n1 do q[i]:=q[i]-1;
              for i:=0 step 1 until n do
              if pos[i] ≤ new then pos[i]:=pos[i]-1;
              RECUR(1,new,1,0,1)
        end    else
        if j≠n then
        begin if u[p[j]]=0 ∧ p[j+1]-p[j]=1 then FLIP;
              new:=old; pi:=p[j]; p[j]:=p[j+1]; p[j+1]:=pi;
              vx[j]:=1; vx[j-1]:=vx[j-1]-1;vx[j+1]:=vx[j+1]-1
        end    else
        begin comment j=n; if u[p[n]]=0 then FLIP;
              new:=0;
              pi:=p[n]; u[pi-1]:=u[pi-1]+1; u[pi]:=u[pi]-1;
              for i:=n step -1 until 2 do
              begin vx[i]:=v[i]:=v[i-1]; p[i]:=p[i-1] end;
              vx[1]:=v[1]:=v[0]-1; vx[0]:=v[0]:=1; p[1]:=pi;
              for i:=old step -1 until 1 do q[i+1]:=q[i]+1;
              q[1]:=1; m:=old+1;
              for i:=old+2 step 1 until n1 do q[i]:=q[i]+1;
              for i:=0 step 1 until n do
              if pos[i]<old then pos[i]:=pos[i+1]
        end
end CASE B3;


comment initialisation, see Part I, page 40;
for i:=0 step 1 until n do
begin aid[i]:=u[i]:=v[i]:=0; x[i]:=0;
      p[i]:=q[i+1]:=i; pos[i]:=i+1
end;
```

```
      for j:=0 step 1 until n do
      begin x[j]:=1; LABEL(x,ℓ); BX[0,j]:=0; B[0,j]:=1;
            for i:=1 step 1 until n do
            begin B[i,j]:=ℓ[i]; BX[i,j]:=0 end;
            x[j]:=0; BX[j,j]:=1
      end;
      new:=0; u[0]:=v[0]:=v[n]:=m:=d:=1; q[1]:=n1:=n+1;
      x[0]:=x[n]:=.5; approx:=admin:=true; iter:=-1;
start:iter:=iter+1;
      if iter>limit then begin approx:=false; go to finish end;
      comment first part of the exchange procedure: determination
              of the index old of the vertex to be deleted;
      INVERT(B,INVB);
      if approx then
      begin comment if the new simplex is in K_d an approximation
                    app is computed and tested on the
                    eps-criterion;
            for i:=0 step 1 until n do
            begin appi:=0; for j:=0 step 1 until n do
                  appi:=appi+BX[i,j]×INVB[j,0];
                  app[i]:=appi
            end;
            LABEL(app,ℓ);
            for i:=0,i+1 while approx ∧ i≠n1 do
            if ABS(ℓ[i])>eps then approx:=false;
            if approx then go to finish
      end of approximation;
      LABEL(x,ℓ);
      if x[0]=0 ∧ ℓ[0]=0 then
      begin pert:=1/(100.×d); ℓ[0]:=-pert;
            for i:=1 step 1 until n do
            ℓ[i]:=ℓ[i]+pert×(ℓ[i]-x[i])
      end of perturbation;
search:
      jold:=-1;
      for j:=0 step 1 until n do
      begin alph:=INVB[j,0];
            for i:=1 step 1 until n do alph:=alph+INVB[i,j]×ℓ[i];
            if alph>0 then
            begin if jold<0 then begin jold:=j; alphi:=alph end
                            else
                  for i:=0,i+1 while ratio=min do
                  begin ratio:=INVB[j,i]/alph;
                        min:=INVB[jold,i]/alphi;
                        if ratio<min then
                        begin jold:=j; alphi:=alph end
                  end
            end
      end;
      old:=pos[jold]; pos[jold]:=new;
      comment second part of the exchange procedure:
              determination of the vertex to be added;
jump: WHICH CASE;
      if admin then
      begin comment updating B and BX; BX[0,jold]:=x[o];
```

```
            for i:=1 step 1 until n do
            begin B[i,jold]:=ℓ[i]; BX[i,jold]:=x[i] end
        end     else admin:=true;
        for j:=0 step 1 until n do if vx[j]≠0 then
        begin if vx[j]=2 then
            begin comment the new vertex and its label
                        are already known;
                old:=n1;
                for jold:=0 step 1 until n do
                if pos[jold]=old then
                begin pos[jold]:=new; admin:=false;
                    iter:=iter+1; go to jump
                end
            end;
            for i:=1 step 1 until j do
            begin pi:=p[i]; aid[pi-1]:=aid[pi-1]-2;
                aid[pi]:=aid[pi]+2
            end;
            if new<m then
            for i:=j+1,i+1 while i≠n1 ∧ vx[i-1]=0 do
            begin pi:=p[i]; aid[pi-1]:=aid[pi-1]-1;
                aid[pi]:=aid[pi]+1
            end;
            d2=d+d;
            for k:=0 step 1 until n do
            begin x[k]:=(2×u[k]+aid[k])/d2; aid[k]:=0 end;
            go to start
        end;
finish:
end FIXED POINT;
```

5.4. COMMENTARY ON THE PROGRAM

5.4.1. For most problems the program will yield a satisfying result with a relatively small d. But for some problems d, and therefore also u, whose components sum to d, may become larger than the integer capacity of the computer permits.

5.4.2. An other difficulty can be the inversion of the matrix B. In many inversion procedures information is given about the possible smallness of pivots in a decomposition.

5.4.3. Inaccurracies made in the procedure INVERT may put the program on the wrong track. However the computer program has shown that this procedure is in practice selfcorrecting: dealing with simplexes that are not very complete it still tends to regain a very complete one. Also from practical experience we know that the chance of cycling can be neglected since the procedure tends to enlarge d continually without ever returning in the neighbourhood of an "old" simplex.

5.4.4. In the program the user is assumed to invert the matrix B anew at each iteration of the algorithm. However, since at each iteration only one column of B is changed, the inversion can also be carried out recursively. (See [1], [2]). On the other hand we do not yet oversee the consequences of inaccuracies that undoubtedly will be made then.

5.4.5. Since generally there is only one index k such that $\beta_{k_0}/\alpha_k = \min\{\beta_{i_0}/\alpha_i \mid 0 \leq i \leq n\}$ (see 2.3.1.) it will be sufficient to compute the first column $\beta_{\cdot_0}$ of INVB and the array α immediately from the linear equation systems $B\beta_{\cdot_0} = e^0$ and $B\alpha = b^{n+1}$ (see 2.3.1.). Only if there are two or more such indexes k then the system $B\beta_{\cdot_1} = e^1$ should be solved too. Etc....

5.4.6. By introduction of some small changes several facilities can be provided by the program:

5.4.6.1. There is a possibility to continue the procedure if after all the ε-approximated fixed point does not satisfy other criteria we may have. For this purpose one should conserve the variables v, u, p, q, and new by making them global or own parameters.
5.4.6.2. Each time d is increased a linear approximation is computed. If the user is interested in the sequence of approximations, the heading of the procedure LABEL can be extended with a Boolean variable indicating whether the supplied point of S is an approximation or not. In that case the procedure LABEL should be called for instance by LABEL(x, ℓ, approx).
5.4.6.3. The program can also be used to compute Kakutani fixed points of a point to set mapping $F: S \to S^*$ by means of the labelling $\ell(u) = x-y$ where $x = \frac{1}{e.u}.u$ and y is some element of F(x). However the ε-criterion may not work effectively, so that a different stopcriterion has to be used. (see [7], page 85-93).

5.4.7. All the problems arising from the inversion of the matrix B can just as in 3.4. be avoided by defining the labelling $\ell(u) := \ell^i$ where ℓ^i is the i^{th} column of the matrix L (see part I, p. 22) and i is the smallest integer such that

$$x_i > 0 \text{ and } x_i - f_i(x) = \max_{0 \le j \le n} (x_j - f_j(x)), \quad \left(x = \frac{u}{eu}\right).$$

Then B will remain constant during the whole program, differing from L only in the first row. (Of course the procedure can also be rewritten for integer labels as in 3.4.). Since the ε-criterion will fail to work with this labelling a different test is necessary.
Though this method has yielded rather nice results it still has appeared to be inferior to the one implemented in the program of 5.3., because we observed that the computertime saved per iteration is swept by a considerable increase in the number of iterations.

5.5. EXAMPLES

In this section we want to illustrate the algorithm with two examples, both concerning continuous not differentiable functions on S. (As in example 5.5.1. even discontinuities of f may occur at the boundary of S). In both examples the function has a fixed point $\hat{x}$ in which at least one of the derivatives $\partial f_i(\hat{x})/\partial \hat{x}_j$ does not exist.

EXAMPLE 5.5.1. This example shows the application of the algorithm to compute equilibrium prices in a Walrasian model of exchange. It has also been used by Scarf to illustrate his algorithm (see [6], example 3 and [7], section 3.2.). A detailed description of its economical foundation is also given by Scarf (see [6], section 6 and [7], sections 1.2., 2.1. and 3.2.) and will just briefly be indicated here.
Let m and n+1 respectively be the number of agents and the number of commodities in an economy. Agent j(j = 1,...,m) owns an initial stock of commodity i(i = 0,...,n) given by w_{ji}. For every vector of prices $x = (x_0,\ldots,x_n)^T$ he can sell his whole initial stock and take in the total amount of $\Sigma_{i=0}^{n} w_{ji}x_i$ to be spend to buy commodities according to his demand functions:

$$h_i^j(x) := \frac{a_{ji}\,\Sigma_{k=0}^{n} w_{jk}x_k}{x_i^{b_j}\,\Sigma_{k=0}^{n} a_{jk}x_k^{1-b_j}} \qquad (5.5.1.1.),$$

a_{ji} and b_j being utility parameters. At prices $x = (x_0,\ldots,x_n)^T$ the total *excess* demand in the market for commodity i is then given by

$$g_i(x) := \Sigma_{j=1}^{m} g_i^j(x) := \Sigma_{j=1}^{m} (h_i^j(x) - w_{ji}) \qquad (5.5.1.2.).$$

The economy is in equilibrium at a price vector $\hat{x}$ if at these prices the market excess demand $g_i(\hat{x})$ is zero for commodities with a positive price and non-positive for commodities with a price equal to zero.
Since the demand functions are homogeneous of degree 0,

implying that demand reacts on relative prices rather than on its absolute values, we may restrict our attention to price vectors x on S. The mapping

$$f_i(x) = \frac{x_i + \lambda.\max\{0,\ g_i(x)\}}{1 + \lambda.\Sigma_{k=0}^{n} \max\{0,\ g_k(x)\}} \qquad (i = 0,\ldots,n) \quad (5.5.1.3.),$$

where λ is a small positive number, can serve as a continuous function on S (- at least at the interior of S; for complications at the boundary of S see below -), taking S into itself. It can be proved that this function has a fixed point and that a fixed point $\hat{x}$ of f is indeed an equilibrium price vector. In our program the problems of possible discontinuities at the boundary of S are avoided by the following procedure: if x is a boundary vector of S the function value of a vector y very near to x (the distance between y and x being much smaller than the smallest possible grid of K_d on the computer) is associated to x. Note that, as far as a discontinuity does not, in contradiction with the facts, suggest a fixed point at this discontinuity, a discontinuity yields no problem for the algorithm. The only problem is, as in the present case, the possible non-existence of the function value at such a point of discontinuity.
In the present example n = g, m = 5 and λ = 1. The parameters W, A and b are as given in appendix 3, table A1. Thus this example is exactly the same as the one used by Scarf ([6], example 3 and [7], section 3.2.). Succesively ε-approximated fixed points were computed for $\varepsilon = 10^{-2}$, 10^{-3}, 10^{-4} and 10^{-5}. The total number of iterations was respectively 471, 598, 763 and 892. Detailed results are given in appendix 3, table A2.

EXAMPLE 5.5.2. This second example concerns the function, whose only fixed point is the barycenter of S, described by

$$f_i(x) = \frac{\max\{x_{i+1},\ x_{i+2}/(i+1)\}}{\Sigma_{k=0}^{n} \max\{x_{k+1},\ x_{k+2}/(k+1)\}} \qquad (i = 0,\ldots,n) \quad (5.5.2.1.),$$

where $x_{n+1} := x_0$ and $x_{n+2} := x_1$.
For n = 6, 12 and 18 and $\varepsilon = 10^{-2}, 10^{-3}, \ldots, 10^{-8}$ successively ε-approximated fixed points were computed. The results are comprimed in appendix 3, table A3, where b is the barycenter $(1/(n+1), \ldots, 1/(n+1))^T$, and the norm $|y|$ is taken to be $\max\{|y_0|, \ldots, |y_n|\}$. After respectively 260, 888 and 2112 iterations for n = 6, 12 and 18 both $|\tilde{x}-f(\tilde{x})|$ and $|\tilde{x}-b|$ were less than 10^{-8}.

5.6. NUMERICAL RESULTS.

In this last section we intend to provide some information about the practical experience with the program and to give some indication of its efficiency. This could be done by estimating the number of iterations or the computing time required to reach an ε-approximated fixed point in terms of n, the dimension of the problem, and ε, the desired accuracy. But, since the computing time depends highly on the function f and the type of computer used, we chose to give an estimation of the number of iterations. Such an estimation in terms of ε, however, would also depend on the function f. To illustrate this: if in example 5.5.1. λ equals .001 in stead of 1, the algorithm goes along the same path through the complex K_*, while in exactly the same number of iterations a thousand times better accuracy would be achieved. Since such changes in the function f don't influence the number of iterations required to reach an approximated fixed point in a certain K_d, the presumption is justified that a reliable estimation of the number of iterations in terms of n and d is possible.
But beforehand we shall discuss some prior information in order to comparise the results of such an estimation and some standard indicating to what extend the algorithm can be called efficient. Here we should take into account that it is impossible to compute the minimum number of iterations required to reach an arbitrary n-simplex in a given K_d since it depends on which simplex is reached. This is why we use the

standard denoted as $\bar{Z}(n, o)$ which is obtained as follows: for each of the 2^n n-simplexes in the n-complex K_2 the shortest path to reach it starting from K_1 is computed and the acquired 2^n minima of numbers of iterations are averaged. This averaged minimum turns out to be $\frac{1}{8}n^2 + \frac{7}{8}n + 1 =: \bar{Z}(n, 1)$. Since for every σ of every K_d $h_\sigma(T)$ is homeomorf with T, $\bar{Z}(n, 1)$ also is the (averaged) minimum number of iterations to reach a simplex of K_{2d} starting from σ in K_d and staying within $h_\sigma(T)$. Consequently the averaged minimum number of iterations to reach a simplex in a certain K_d is then given by

$$\bar{Z}(n, o) = o.\bar{Z}(n, 1) = {}^2\log d(\tfrac{1}{8}n^2+\tfrac{7}{8}n+1), \qquad (5.6.1.)$$

where o is the number of times d is doubled. (This formula neglects the fact that sometimes a shorter path can be obtained by leaving $h_\sigma(T)$ not on the upper side (CASE B2), but going sideways (CASE B3) to another $h_{\sigma'}(T)$ and from there going upwards and sooner or later sideways again, which at first sight seems to be a roundabout way).
In order to illustrate the behavior of the algorithm and to give some evidence for the relationship between the number of iterations and the parameters n and d the algorithm has been applied to a set of functions given by

$$f_i(x) = ((1+\Sigma_{j=0}^{n} a_{kj}x_j) \cdot \Sigma_{k=0}^{n}(1+\Sigma_{j=0}^{n} a_{kj}x_j)^{-1})^{-1}, \qquad (5.6.2.)$$

where a_{kj} are the elements of a square matrix A. For four different matrices A of size 19 × 19 and for n = 3, 6, 9, 12, 15 and 18 the first n+1 columns and rows of these matrices were used to obtain 4 problems for each size of n. The results are comprimed in appendix 3, table A4, where $o = {}^2\log d$, Z is the number of iterations required to reach the corresponding ε-approximated fixed point in K_d and c: = $Z/\bar{Z}(n, o)$, the number of iterations related to the standard. For this set of functions and an accuracy of 10^{-8} c varies from 0.89 to 1.16, so the required number of iterations is about the same as the standard $\bar{Z}(n, o)$. There is substantial evidence that these

results are fairly normal for rather uncomplicated functions. For more complicated functions c can be higher. For the three examples of 5.5.2. c was respectively 1.01, 1.25 and 1.42, and c turned out to be 2.04 for example 5.5.1. Our experience is that in virtually all cases no more than two or three times the (averaged) minimum number of iterations is required.
The second, perhaps most important conclusion of these examples examples, which also holds for most other examples, is that c is very stable during the course of the algorithm. This implies that a linear increase in the number of iterations corresponds with a linear increase of o, and so with an exponential increase of d, mostly causing an exponential increase of the accuracy (at least as soon as a minimal accuracy, in these examples of about 10^{-2}, is reached).
Besides the fact that the algorithn continuous until a *a priori* accuracy is reached, it is this result, following from the structure of the complex K_*, which is the great advantage of the present algorithm in comparison with all algorithms for a fixed K_d. For a rough comparison of the algorithm of chapter 3 for a fixed K_d and the one of chapter 4 for K_* the standard $\bar{Z}$ also applies to K_d. If each n-simplex of K_d is given an equal chance to be reached as a very complete one terminating the algorithm, the "averaged minimum number of iterations" is about $\frac{n}{n+1}\,\bar{Z}(n-1,\ d-1)\ (\approx d.(\frac{1}{8}n^2+\frac{1}{2}n-\frac{1}{4}))$. Although this formula again neglects some complications, it is nevertheless useful for a rough comparison.
This standard also fits for the algorithms used in Scarf's book "The computation of economic equilibria" [7]. These algorithms also proceed in a fixed K_d and of the 27 numerical examples given in his book one needed 5.41 times the standard, another one took 0.71 times, again another one (the same as in 5.5.1.) terminated after only 0.12 times as much iterations, and all the other 24 examples needed a number of iterations between 0.98 and 1.98 times this standard.
Since $\frac{n}{n+1}\,\bar{Z}(n-1,\ d-1)$ is linearly proportional to d, a linear increase of the number of iterations leads to a linear increase of the accuracy, while in case of the algorithm for K_* this

would lead to an exponential increase of the accuracy. A comparison of the two standards makes clear that the expected numbers of iterations for the two algorithms to obtain the same accuracy are roughly in the proportion of 2^{o}: o. This shows the superiority of the algorithm for K_* compared to those for a fixed K_d at least if a high accuracy is wanted. For instance, using the algorithm for a fixed K_d, the expected number of iterations to reach an accuracy less than 10^{-5} for the average of the four examples of this section would be about 130, 600, 1800, 7600 and 11000 for n = 3 up to 18, and an accuracy less than 10^{-8} would take about 6,000, 20,000, 100,000, 140,000, 290,000 and 240,000 iterations respectively. An extreme example like 5.5.2., where for n = 18 an accuracy of 10^{-8} was reached in 2112 iterations at a d of 2^{26}, shows the impossibilities of the algorithm for a fixed K_d, since this would take some 3 billions of iterations.

However, the weak spot in the algorithm is the first part of the exchange procedure, especially the operations with the matrix B and its inverse. Of course the possibility exists that round-off errors cause an incorrect exchange of vertices, but, which counts more, is the fact that the inversion of the matrix B requires an order of $(n+1)^3$ arithmetical operations and is therefore, besides the function computations, the most time-spending procedure in the program. Although for instance in example 5.3.1., where n = 9, about $\frac{5}{6}$ of the total computation time was consumed by the function evaluations and only $\frac{1}{6}$ was used by the rest of the procedure, including a matrix inversion at each iteration, for large n the time needed for the inversion of an $(n+1)\times(n+1)$ matrix becomes prohibitively large. We think this to be one of the main subjects for improving the algorithm.

First, strictly speaking the computation of the inverse of the matrix B is not necessary; at nearly all iterations only two systems of equations need to be solved: $B.\alpha = b^{n+1}$ and $B.\beta_{.0} = e^0$. Very seldom $B.\beta_{.i} = e^i$ for $i = 1,\ldots,j$ $(j \leq n)$ should be solved too. Moreover, if these equations are solved

by LU-decomposition computer time can be saved by reversing the order of the rows of B. This reduces the number of operations since in that case the system $L.U.\beta_{.0} = e^n$ instead of $L.U.\beta_{.0} = e^0$ has to be solved. But computing the LU-decomposition of B still takes an order of $\frac{1}{3}(n+1)^3$ arithmetical operations at each iteration. Since B changes in only one column, however, this much work is not needed. As shown by Bartels [2], a stable procedure exists to compute at each iteration the changes in the LU-decomposition, which only takes an order of $2(n+1)^2 + k(n+1)$ operations (where k is the number of iterations). Although this procedure implies that the LU-decomposition has to be computed anew from time to time, and so in the context of the fixed point algorithm then also the present simplex should be tested on its very completeness, we think Bartels' results justify further research on the utility of his procedure for the fixed point algorithm.

Another possible improvement is the computation of B^{-1} at each iteration from B^{-1} of the proceeding iteration and the changed column of B. This takes only an order of $2(n+1)^2$ arithmetic operations. Although, as shown by Bartels [2], in general the round-off errors of this procedure can not be bounded a priori, from experiments with the examples of this section we obtained hopeful results. During the whole course of the algorithm B^{-1} has been computed recursively in normal precision floating-point arithmetic without any reinversion and this we compared with B^{-1} computed by inverting B at each iteration. The relative differences between the elements of the two computed inverses turned out to remain stable during the course of the algorithm at a few orders of magnitude larger than the normal precision accuracy of the computing machine. We believe that further research on this subject will surely lead to improvements of the program.

APPENDIX 1.

Here we shall prove that a sufficiently small complete sub-simplex $\sigma = \{u^0,\ldots,u^n\}$ of S enables us to compute an approximated fixed point with predicted accuracy.
The completeness of σ states that there is an $\alpha \in S$ such that $\Sigma_{i=0}^{n}\alpha_i(u^i-f(u^i)) = 0$. Further, since f is continuous on S, for each $\varepsilon > 0$ there will be a $\delta > 0$ such that $|f(x)-f(y)| \leq \varepsilon$ whenever $|x-y| \leq \delta$. Now let us suppose that indeed every two points of σ have a distance of at most δ. Then $|\Sigma_{i=0}^{n}\alpha_i(u^j-f(u^i))| = |\Sigma_{i=0}^{n}\alpha_i(u^i-f(u^i))+u^j-u^i| =$ $\Sigma_{i=0}^{n}\alpha_i(u^i-f(u^i))+|u^j-u^i| \leq \delta$ and $|\Sigma_{i=0}^{n}\alpha_i(f(u^i)-f(u^j))| \leq$ $\leq \Sigma_{i=0}^{n}\alpha_i|f(u^i)-f(u^j)| \leq \varepsilon$. So $|u^j-f(u^j)| = |\Sigma_{i=0}^{n}\alpha_i(u^j-f(u^j))| \leq$ $\leq |\Sigma_{i=0}^{n}\alpha_i(u^j-f(u^i))| + |\Sigma_{i=0}^{n}\alpha_i(f(u^i)-f(u^j))| \leq \varepsilon + \delta$ for every j. And it follows that for each point $x \in C(\sigma)$ $|x-f(x)| \leq$ $\leq |x-u^j| + |u^j-f(u^j)| + |f(u^j)-f(x)| \leq 2(\varepsilon+\delta)$. So in fact every point x of $C(\sigma)$ is a $2(\varepsilon+\delta)$-approximated fixed point.□

APPENDIX 2.

In K_* with labelling $\ell: K_*^0 \to$ S-S as given in (4.5.2.1.) the boundary simplex $\{e^0,\ldots,e^n\}$ is very complete.

PROOF: Let L be the $(n+1)\times(n+1)$-matrix containing the elements $\ell_{ij} := \ell_i(e^j)$ and B: $= [b_{ij}]$ with $b_{ij} := \ell_{ij}$ for all j and all $i \neq 0$, and b_{0j} is 1 for all j. Then from (4.5.2.1.) it follows that $\ell_{00} > 0$, $\ell_{0j} < 0$ for $j \neq 0$, $\ell_{ii} > 0$ for $i \neq 0$ and $\ell_{ij} \leq 0$ for $i \neq 0$ and $j \neq i$. So the matrix L has the following structure, the unequalities refferring to zero:

$$\begin{bmatrix} \geq & < & < & . & . & . & < \\ & > & & & & & \\ & & > & & & \leq & \\ & & & . & & & \\ & \leq & & & . & & \\ & & & & & . & \\ & & & & & & > \end{bmatrix}.$$

Remind that $\Sigma_{i=0}^n \ell_{ij} = 0$, so $-\Sigma_{i=1, i\neq j}^n \ell_{ij} < \ell_{jj}$ for $j \neq 0$.
We can prove that B^{-1} exists by showing that the rows of B are linearly independent. To prove this let the vector α be a root of the equality $B^T\alpha = 0$, and let $\alpha_p := \min\{\alpha_i \mid 1 \leq i \leq n\}$ and $\alpha_q := \max\{\alpha_i \mid 1 \leq i \leq n\}$.
Since $\Sigma_{i=0}^n \alpha_i b_{ij} = 0$, $\Sigma_{i=1}^n \alpha_i b_{ij} = -\alpha_0 b_{0j} = -\alpha_0$ for all j, and since $b_{ij} \leq 0$ for $i \neq j$ it follows that $\alpha_q \Sigma_{i=1}^n b_{iq} \leq$
$\leq \Sigma_{i=1}^n \alpha_i b_{iq} = -\alpha_0 = \Sigma_{i=1}^n \alpha_i b_{ip} \leq \alpha_p \Sigma_{i=1}^n b_{ip}$. So, since both $\Sigma_{i=1}^n b_{iq}$ and $\Sigma_{i=1}^n b_{ip}$ are positive, α_p and α_q and with them all α_i $(i \neq 0)$ have the same sign.
Now suppose that $\alpha_p < 0$, then $\Sigma_{i=1}^n \alpha_i b_{ip} < 0$, so $\alpha_0 > 0$ and therefore $\alpha_0 + \Sigma_{i=1}^n \alpha_i b_{i0} > 0$, a contradiction. In the same way the supposition $\alpha_q > 0$ leads to a contradiction with $\alpha_0 + \Sigma_{i=1}^n \alpha_i b_{i0} = 0$. So $\alpha_p = \min\{\alpha_i \mid 1 \leq i \leq n\} \geq 0$ and

$\alpha_q = \max\{\alpha_i \mid 1 \leq i \leq n\} \leq 0$. Clearly $\alpha_i = 0$ for all $i \neq 0$. Then also $\alpha_0 = -\Sigma_{i=1}^{n} \alpha_i b_{ij} = 0$. So $\alpha = 0$, the rows of B are linearly independent, and B^{-1} exists.

Now observe the function g defined by $g(x) = (I-L)x$ for $x \in S$. g is a continuous map from S into itself and therefore has a fixed point in S. This means that there is a $\lambda(0) \in S$ such that $(I-L)\lambda(0) = \lambda(0)$, and so $L\lambda(0) = 0$. Note that $\lambda_0(0) > 0$, since $\Sigma_{j=0}^{n} \lambda_j(0).\ell_{0j} = 0$, $\lambda_j(0) \geq 0$ for all j and $\ell_{0j} < 0$ for all $j \neq 0$. Further from $L\lambda(0) = 0$ and $\Sigma_{j=0}^{n} \lambda_j(0) = 1$ it follows that $B\lambda(0) = e^0$.

Next observe the system $B\lambda(\varepsilon) = \varepsilon'$ with solution $\lambda(\varepsilon) = B^{-1}\varepsilon'$, where $\varepsilon' = (1,\varepsilon,\varepsilon^2,\ldots,\varepsilon^n)^T$ and $\varepsilon \geq 0$. Then, since $\lambda_0(0) > 0$, $\lambda_0(\varepsilon) \geq 0$ for sufficiently small ε. Now suppose that $J(\varepsilon) := \{j \mid \lambda_j(\varepsilon) < 0\} \neq \phi$ for certain ε. Since $B\lambda(\varepsilon) = \varepsilon'$, $\Sigma_{j=0}^{n} \lambda_j(\varepsilon).b_{ij} = \varepsilon^i \geq 0$ for all i, so the more for all $i \in J(\varepsilon)$. But, since for all $i \in J(\varepsilon)$ and all $j \notin J(\varepsilon)$ it holds that $b_{ij} \leq 0$ and $\lambda_j \geq 0$, $\Sigma_{j \notin J(\varepsilon)} \lambda_j(\varepsilon).b_{ij} \leq 0$ and therefore also $\Sigma_{j \in J(\varepsilon)} \lambda_j(\varepsilon).b_{ij} \geq 0$ for all $i \in J(\varepsilon)$. So $\Sigma_{j \in J(\varepsilon)} (\lambda_j(\varepsilon) \Sigma_{i \in J(\varepsilon)} b_{ij}) = \Sigma_{i \in J(\varepsilon)} \Sigma_{j \in J(\varepsilon)} \lambda_j(\varepsilon).b_{ij} \geq 0$ in contradiction with the fact that $\lambda_j(\varepsilon) < 0$ and $\Sigma_{i \in J(\varepsilon)} b_{ij} \geq \Sigma_{i=1}^{n} b_{ij} > 0$ for all $j \in J(\varepsilon)$. So $J(\varepsilon) = \phi$ for all ε for which $\lambda_0(\varepsilon) \geq 0$!

So we conclude that the system $B\lambda(\varepsilon) = \varepsilon'$ has a solution $\lambda(\varepsilon) \geq 0$ for all sufficiently small $\varepsilon \geq 0$, which in other words means that $\{e^0,\ldots,e^n\}$ is very complete. □

APPENDIX 3.

TABLE A1. Data for example 5.5.1.

n+1, number of commodities: 10

m , number of agents in the economy: 5

i , commodity index: 0,...,9

j , agent index: 1,...,5

w_{ji}, initial stock of commodities:

0.6	0.2	0.2	20.0	0.1	2.0	9.0	5.0	5.0	15.0
0.2	11.0	12.0	13.0	14.0	15.0	16.0	5.0	5.0	9.0
0.4	9.0	8.0	7.0	6.0	5.0	4.0	5.0	7.0	12.0
1.0	5.0	5.0	5.0	5.0	5.0	5.0	8.0	3.0	17.0
8.0	1.0	22.0	10.0	0.3	0.9	5.1	0.1	6.2	11.0

a_{ji}, utility parameters:

1.0	1.0	3.0	0.1	0.1	1.2	2.0	1.0	1.0	0.7
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
9.9	0.1	5.0	0.2	6.0	0.2	8.0	1.0	1.0	0.2
1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0
1.0	13.0	11.0	9.0	4.0	0.9	8.0	1.0	2.0	10.0

b_j , utility parameters: 2.0 1.3 3.0 0.2 0.6

TABLE A2. Results of example 5.5.1.

ε: accuracy
Z: total number of iterations
g: excess demand function (5.5.1.2.)
f: continuous function from S into S (5.5.1.3.)
$\tilde{x}$: ε-approximated fixed point of f.

ε, z	i	$\tilde{x}_i$	$f_i(\tilde{x})$	$\tilde{x}_i - f_i(\tilde{x})$	$g_i(\tilde{x})$
	0	.187170	.191928	-.004758	.011335
	1	.109363	.112132	-.002770	.006612
	2	.098920	.095642	.003277	-.009081
.01	3	.043210	.041779	.001432	-.021972
	4	.116861	.118577	-.001716	.005779
	5	.077000	.074448	.002551	-.005668
471	6	.116976	.114366	.002611	.001308
	7	.102384	.102181	.000203	.003298
	8	.098686	.101153	-.002467	.005934
	9	.049431	.047793	.001638	-.046776
	0	.187252	.187747	-.000495	.001306
	1	.109378	.109418	-.000040	.000512
	2	.098898	.098473	.000425	-.000797
.001	3	.043197	.043011	.000186	-.007588
	4	.116865	.117164	-.000298	.000804
	5	.076977	.076646	.000331	-.000649
598	6	.116965	.117275	-.000310	.000816
	7	.102381	.102442	-.000061	.000503
	8	.098691	.098642	.000049	.000377
	9	.049395	.049182	.000212	-.002476

ε, z	i	$\tilde{x}_i$	$f_i(\tilde{x})$	$\tilde{x}_i - f_i(\tilde{x})$	$g_i(\tilde{x})$
	0	.187261	.187349	-.000088	.000179
	1	.109379	.109393	-.000014	.000067
	2	.098897	.098848	.000048	-.000179
.0001	3	.043192	.043171	.000021	-.000635
	4	.116867	.116859	.000008	.000050
	5	.076975	.076937	.000038	-.000061
763	6	.116966	.116909	.000057	-.000000
	7	.102381	.102429	-.000048	.000098
	8	.098691	.098737	-.000046	.000095
	9	.049393	.049369	.000024	-.000329
	0	.187262	.187268	-.000006	.000015
	1	.109379	.109381	-.000002	.000007
	2	.098896	.098891	.000005	-.000028
.00001	3	.043191	.043189	.000002	-.000083
	4	.116867	.116861	.000006	-.000001
	5	.076974	.076977	-.000002	.000006
892	6	.116966	.116969	-.000003	.000009
	7	.102381	.102380	.000001	.000004
	8	.098691	.098694	-.000003	.000008
	9	.049392	.049390	.000002	-.000001

TABLE A3. Results of example 5.5.2.

n : dimension of S

ε : accuracy

Z : total number of iterations

$\tilde{x}$: ε-approximated fixed point of f

f : continuous function from S into S (5.5.2.1.)

b : barycenter of S

$|y|$: $\max\{|y_0|,\ldots,|y_n|\}$

n	ε	z	$\tilde{x}$	$f(\tilde{x})$	$\lvert\tilde{x}-f(\tilde{x})\rvert$	$\lvert\tilde{x}-b\rvert$
6	10^{-2}	36	.137-.148	.139-.148	.00868205	.00634398
	10^{-3}	100	.143	.143	.00015905	.00009310
	10^{-4}	132	.143	.143	.00001992	.00001163
	10^{-5}	164	.14286	.14286	.00000249	.00000146
	10^{-6}	196	.142857	.142857	.00000031	.00000018
	10^{-7}	228	.142857	.142857	.00000004	.00000003
	10^{-8}	260	.14285714	.14285714	.00000000	.00000000
12	10^{-2}	150	.072-.082	.073-.081	.00835578	.00488917
	10^{-3}	332	.077	.077	.00046169	.00025041
	10^{-4}	453	.0769	.0769	.00001155	.00000726
	10^{-5}	585	.07692	.07692	.00000217	.00000118
	10^{-6}	651	.076923	.076923	.00000046	.00000025
	10^{-7}	798	.076923	.076923	.00000009	.00000005
	10^{-8}	888	.07692308	.07692308	.00000000	.00000000
18	10^{-2}	283	.049-.060	.049-.060	.00423262	.00786243
	10^{-3}	757	.053	.053	.00017529	.00009252
	10^{-4}	986	.053	.053	.00008927	.00004713
	10^{-5}	1357	.05263	.05263	.00000254	.00000134
	10^{-6}	1522	.052632	.052632	.00000013	.00000007
	10^{-7}	1761	.0526316	.0526316	.00000006	.00000003
	10^{-8}	2112	.05263158	.05263158	.00000000	.00000000

TABLE A4. Results of four examples of section 5.6.

n: dimension of S

ε: accuracy

o: $^2\log d$, number of times d is doubled

Z: total number of iterations

c: $Z/\bar{Z}(n,o)$

		example 1			example 2			example 3			example 4			average 4 examples		
	ε	o	z	c	o	z	c	o	z	c	o	z	c	o	z	c
$n = 3$ $\bar{z}(3,1) = 4.75$	10^{-2}	2	10	1.05	0	0	1.00	1	6	1.26	2	8	0.84	1.25	6.00	1.01
	10^{-3}	4	20	1.05	1	6	1.26	3	15	1.05	3	12	0.84	2.75	13.25	1.01
	10^{-4}	4	20	1.05	4	22	1.16	4	19	1.00	5	20	0.84	4.25	20.25	1.00
	10^{-5}	6	28	0.98	5	28	1.18	6	27	0.95	6	24	0.84	5.75	26.75	0.98
	10^{-6}	9	44	1.03	7	37	1.11	8	38	1.00	8	34	0.89	8.00	38.25	1.01
	10^{-7}	10	50	1.05	9	45	1.05	10	46	0.97	9	39	0.91	9.50	45.00	1.00
	10^{-8}	12	59	1.04	10	49	1.03	12	56	0.98	11	48	0.92	11.25	53.00	0.99
$n = 6$ $\bar{z}(6,1) = 10.75$	10^{-2}	3	38	1.18	1	15	1.40	3	34	1.05	1	10	0.93	2.00	24.25	1.13
	10^{-3}	4	47	1.09	3	45	1.40	4	44	1.02	3	31	0.96	3.50	41.75	1.11
	10^{-4}	6	67	1.04	4	52	1.21	6	68	1.05	4	38	0.88	5.00	56.25	1.05
	10^{-5}	7	76	1.01	6	77	1.19	7	80	1.06	6	61	0.95	6.50	73.50	1.05
	10^{-6}	9	98	1.01	8	97	1.13	9	107	1.11	7	75	1.00	8.25	94.25	1.06
	10^{-7}	10	106	0.99	10	122	1.13	10	119	1.11	9	99	1.02	9.75	111.50	1.06
	10^{-8}	12	129	1.00	11	137	1.16	12	145	1.12	11	122	1.03	11.50	133.25	1.08
$n = 9$ $\bar{z}(9,1) = 19$	10^{-2}	3	67	1.18	2	40	1.05	4	73	0.96	2	35	0.92	2.75	53.75	1.03
	10^{-3}	4	83	1.09	4	73	0.96	5	95	1.00	4	71	0.93	4.25	80.50	1.00
	10^{-4}	6	115	1.01	6	116	1.02	7	121	0.91	4	71	0.93	5.75	105.75	0.97
	10^{-5}	7	138	1.04	7	132	0.99	7	121	0.91	7	125	0.94	7.00	129.00	0.97
	10^{-6}	9	179	1.05	9	177	1.04	9	170	0.99	8	137	0.90	8.75	165.75	1.00
	10^{-7}	11	223	1.07	11	214	1.02	12	244	1.07	10	173	0.91	11.00	213.50	1.02
	10^{-8}	13	269	1.09	12	233	1.02	14	272	1.02	12	207	0.91	12.75	245.25	1.01

$n = 18$ $\bar{z}(18,1) = 57.25$ $n = 15$ $\bar{z}(15,1) = 42.25$ $n = 12$ $\bar{z}(12,1) = 29.5$

	example 1			example 2			example 3			example 4			average 4 examples		
ε	o	z	c	o	z	c	o	z	c	o	z	c	o	z	c
10^{-2}	4	146	1.24	2	60	1.02	3	89	1.01	2	55	0.93	2.75	87.50	1.08
10^{-3}	5	171	1.16	4	140	1.19	4	119	1.01	4	119	1.01	4.25	137.25	1.09
10^{-4}	6	195	1.10	6	197	1.11	6	178	1.01	5	142	0.96	5.75	178.00	1.05
10^{-5}	8	262	1.11	7	234	1.13	8	246	1.04	7	207	1.00	7.50	237.25	1.07
10^{-6}	9	301	1.13	8	267	1.13	9	282	1.06	8	235	1.00	8.50	271.25	1.08
10^{-7}	11	358	1.10	10	335	1.14	11	352	1.08	10	285	0.97	10.50	332.50	1.07
10^{-8}	13	423	1.10	12	403	1.14	13	407	1.06	12	316	0.89	12.50	387.25	1.05
10^{-2}	2	107	1.27	2	83	0.98	2	91	1.08	2	79	0.93	2.00	90.00	1.07
10^{-3}	5	236	1.12	5	211	1.00	4	183	1.08	4	173	1.02	4.50	200.75	1.06
10^{-4}	6	276	1.09	6	250	0.99	6	269	1.06	6	266	1.05	6.00	265.25	1.05
10^{-5}	8	373	1.10	8	335	0.99	7	317	1.07	8	342	1.01	7.75	341.75	1.04
10^{-6}	9	413	1.09	9	374	0.98	9	410	1.08	9	380	1.00	9.00	394.25	1.04
10^{-7}	11	502	1.08	11	433	0.93	11	489	1.05	11	473	1.02	11.00	474.25	1.02
10^{-8}	13	581	1.06	13	501	0.91	13	600	1.09	13	570	1.04	13.00	563.00	1.03
10^{-2}	2	125	1.09	3	154	0.90	2	127	1.11	3	189	1.10	2.50	148.75	1.04
10^{-3}	4	248	1.08	4	222	0.97	4	257	1.12	4	257	1.12	4.00	246.00	1.07
10^{-4}	6	369	1.07	6	333	0.97	6	385	1.12	5	332	1.16	5.75	354.75	1.08
10^{-5}	8	499	1.09	8	482	1.05	7	439	1.10	8	513	1.12	7.75	483.25	1.09
10^{-6}	9	525	1.02	10	600	1.05	9	516	1.00	10	629	1.10	9.50	567.50	1.04
10^{-7}	11	675	1.07	11	666	1.06	11	622	0.99	11	688	1.09	11.00	662.75	1.05
10^{-8}	12	741	1.08	13	773	1.04	12	673	0.98	12	732	1.07	12.25	729.75	1.04

REFERENCES

[1] Bartels, R.H., Golub, G.H., Saunders, M.A., Numerical techniques in mathematical programming, Report STAN-CS-70-162, Stanford University, 1970.

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

[3] Dantzig, G.B., *Linear programming and extensions*, Princeton University Press, Princeton, N.J., 1963.

[4] Eaves, B.C., Computing Kakutani fixed points, *SIAM Journal of Applied Mathematics* 21, 236-244 (1971).

[5] Eaves, B.C., Homotopies for computation of fixed points, *Mathematical Programming* 3, 1-2 (1972).

[6] Scarf, H., The approximation of fixed points of a continuous mapping, *SIAM Journal of Applied Mathematics* 15, 1328-1343 (1967).

[7] Scarf, H., *The computation of economic equilibria*, Yale University Press, London, 1973.

[8] Spanier, E.H., *Algebraic topology*, McGraw-Hill, New York, 1966.

[9] Stoer, J., Witzgall, Chr., *Convexity and optimization in finite dimensions I*, Springer-Verlag, Berlin, 1970.

PREVIOUS NUMBERS:

EIT 40	F.A. Engering	The monetary multiplier and the monetary model.
EIT 41	J.M.A. van Kraay	The international product life cycle concept.
EIT 42	W.M. van der Goorbergh	Productionstructures and external diseconomies.
EIT 43	H.N. Weddepohl	An application of game theory to a problem of choice between privat en public transport.
EIT 44	B.B. van der Genugten	A statistical view to the problem of the economic lot size.
EIT 45	J.J.M. Evers*	Linear infinite horizon programming.
EIT 46	Th. van de Klundert A. van Schaik	On shift and share of durable capital.
EIT 47	G.R. Mustert	The development of income distribution in the Netherlands after the second world war.
EIT 48	H. Peer	The growth of labor-management in a private economy.
EIT 49	J.J.M. Evers	On the initial state vector in linear infinite horizon programming.
EIT 50	J.J.M. Evers	Optimization in normed vector spaces with applications to optimal economic growth theory.
EIT 51	J.J.M. Evers	On the existence of balanced solutions in optimal economic growth and investment problems.
EIT 52	B.B. van der Genugten	An (s,S)-inventory system with exponentially distributed lead times.
EIT 53	H.N. Weddepohl	Partial equilibrium in a market in the case of increasing returns and selling costs.
EIT 54	J.J.M. Evers	A duality theory for convex ∞-horizon programming.

EIT 1975
*) not available