METHODS OF FEASIBLE DIRECTIONS

A STUDY IN LINEAR AND NON-LINEAR PROGRAMMING

G. ZOUTENDIJK

METHODS OF FEASIBLE DIRECTIONS

METHODS OF FEASIBLE DIRECTIONS

A STUDY IN LINEAR AND NON-LINEAR PROGRAMMING

ACADEMISCH PROEFSCHRIFT
TER VERKRIJGING VAN DE GRAAD VAN
DOCTOR IN DE WIS- EN NATUURKUNDE
AAN DE UNIVERSITEIT VAN AMSTERDAM
OP GEZAG VAN DE RECTOR MAGNIFICUS
DR. J. KOK
HOOGLERAAR IN DE FACULTEIT DER WIS- EN NATUURKUNDE
IN HET OPENBAAR TE VERDEDIGEN
IN DE AULA DER UNIVERSITEIT
OP WOENSDAG 22 JUNI 1960
DES NAMIDDAGS TE 3 UUR PRECIES

DOOR

GUUS ZOUTENDIJK

GEBOREN TE 'S-GRAVENHAGE

ELSEVIER PUBLISHING COMPANY
AMSTERDAM – LONDON – NEW YORK – PRINCETON

Promotor: Prof. Dr. Ir. A. van Wijngaarden

Voor mijn ouders
Voor Miekie

VOORWOORD

De studie die het onderwerp is van dit proefschrift is in de jaren 1958 en 1959 gemaakt op het Koninklijke/Shell-Laboratorium, Amsterdam (Shell Internationale Research Maatschappij, N.V.). Bijzondere dank ben ik verschuldigd aan de directie van dit laboratorium voor de toestemming tot publikatie en de vele administratieve hulp welke mij verleend werd.

Van de velen die tot mijn wiskundige vorming bijdroegen wil ik in het bijzonder vermelden Prof.Dr. J.Droste van de Rijksuniversiteit te Leiden wiens assistent ik gedurende 3 jaar mocht zijn en wijlen Prof.Dr.D. van Dantzig, onder wiens leiding ik gedurende ruim 1 jaar op de statistische afdeling van het Mathematisch Centrum gewerkt heb en die aanvankelijk als promotor zou optreden. Mijn huidige promotor, Prof.Dr.Ir. A. van Wijngaarden ben ik zeer verplicht voor zijn bereidheid als promotor op te treden toen de studie reeds in een vergevorderd stadium was.

De voortdurende belangstelling en de vele stimulerende opmerkingen van Dr.R.J. Lunbeck en J.F. Benders worden met dank vermeld. De nieuwe numerieke versie van de simplex methode welke in dit proefschrift wordt gerapporteerd is gegroeid uit discussies met D.M. Carstens, D.R. Horner en H.J. Krajenbrink. Het waardevolle commentaar van E.M.L. Beale met betrekking tot een eerdere versie van de methoden der toelaatbare richtingen droeg veel bij tot de uiteindelijke ontwikkeling van deze methoden.

CONTENTS

1. INTRODUCTION

1.1 *History*

Mathematical programming is concerned with the problem of maximizing or minimizing a function of variables that are restricted by a number of constraints. Interest in this problem arose in economics and management sciences, where it was realized that many problems of optimum allocation of scarce resources could be formulated mathematically as programming problems. The introduction of large, high-speed electronic computers, moreover, made it possible in principle to obtain numerical solutions, provided efficient mathematical methods and computational techniques could be developed. These methods cannot immediately be derived from classical tools, such as the method of Lagrange multipliers. The latter has effectively been applied to extremum problems in which the variables were only restricted by equality constraints but it is hardly, if at all, possible to extend such a method to inequality-constrained extremum problems. Moreover, mathematical programming problems nearly always consist of many variables and constraints - there may be several hundreds or even more - and to a large extent it is this property which distinguishes them from classical extremum problems and which makes the development of efficient computational techniques necessary.

A great stimulus was the development by G. B. Dantzig of the simplex method for the linear programming problem, i.e. the problem of optimizing a linear function of variables restricted by a number of linear inequality and equality constraints. A surprisingly large class of industrial and business problems could be formulated as a linear programming problem, sometimes after linearization of the objective function or part of the constraints. The bibliography, composed by Riley and Gass [33][1], comprises hundreds of case study references. In many cases linear programming has contributed much to better decision making and a better understanding of the problems in business and industry. Computer codes have been written for various computers and are used extensively all over the world. Attention has also been paid to linear programming problems with a special, simple structure like the transportation problem or more general network problems. Some textbooks on linear programming have been written. The mathematical and computational theory can for instance be found in [22] and [36].

Many practical problems, however, could not or hardly be represented by a linear programming model. Therefore, attempts were and are made to develop more general mathematical pro-

1) Figures between square brackets refer to the references at the end of the book.

gramming methods. A survey of existing methods which does not intend to be complete will be given in the next section.

1.2 *Current Status of Mathematical Programming*

The following three aspects can be distinguished in mathematical programming:

1. The applicational or technological problem, i.e. the formulation of the model, the gathering of the data, the interpretation and study of the results, etc.
2. The mathematical problem, i.e. the development of mathematical techniques for a certain class of models.
3. The computational problem, i.e. the study of the computational aspects of a mathematical method and the development of computer codes for it.

These three problems are of course not independent. The mathematical models, for instance, will be adapted as far as possible to the methods available. The linearizing of non-linear constraints or of the objective function which has been done with a view to applying an existing linear programming method is an example of this. A study of the computational aspects of the method and computational experience with it will often lead to improvements in the method itself. Some questions like the accrual of rounding-off error and the speed of convergence can hardly be studied in another way than computationally. Finally, the computer, more than the theory will show whether a method is good in practice or not. The facts which make the simplex method so useful are not its theoretical finiteness or anti-degeneracy precautions but the computational experience that the number of iterations needed is surprisingly small, even in larger problems, that the rounding-off error is much less serious in practical problems than theoretically estimated and that an anti-degeneracy precaution is not necessary in practice.

Mathematical programming problems can be divided into four classes:

1. Deterministic, continuous models: the set of points, satisfying all constraints, - to be called feasible region - is connected; the objective function, i.e. the function to be optimized is continuous.
2. Deterministic, discontinuous models: the feasible region is not connected or (and) the objective function is not continuous.
2. Stochastic models: the coefficients in the constraints or (and) in the objective function are random variables.
4. Dynamic models: the coefficients in the constraints or (and) in the objective function are dependent on a parameter (e.g. the time). For each value of this parameter we now want to solve the problem.

In the first class we can find:

a. Linear programming.

Most of the work on this subject has already been completed, as

follows from section 1.1. Attention is still being paid to the computational aspects of the simplex method as follows from part II of this monograph. Quite recently special methods have been proposed by Dantzig and Wolfe [16], and independently by Benders [6], for the solution of very large linear programming problems which use the simplex method in a number of smaller sub-problems and may be more efficient if the problem has a special, repetitive structure.

b. Quadratic programming, i.e. the problem of minimizing a convex quadratic function, subject to linear constraints. Several methods have been devised for this problem by various authors. Those of Beale [3] and Wolfe [39] are direct extensions of the simplex method, those of Frank and Wolfe [20] and Zoutendijk [40], chapter 10 of this monograph, use the simplex method for the solution of a number of sub-problems.

c. The problem of minimizing a general convex function, subject to linear constraints. Most of the methods developed for this problem can be considered as large-step gradient methods. We mention those of Frank and Wolfe [20], Rosen [34] and Zoutendijk [40], chapter 11 of this monograph. If the convex function to be minimized is separable, then a linearization of the objective function can easily be accomplished as has been pointed out by Charnes and Lemke [9]. An interesting analogy between mathematical programming problems of this kind and electrical networks has been shown by Dennis [17].

d. Convex programming, i.e. the problem of minimizing a convex function (or maximizing a concave function) in a convex region. Methods for this problem have been developed by Arrow and Hurwicz [1], whose method is a small-step gradient method, by Zoutendijk [40], chapter 11 of this monograph, by Kelley, and by Cheney and Goldstein [10]. The convexity assumption plays an important role in these problems; without such an assumption the existing methods either do not work or they lead at best to a local optimum, so that one is never sure whether or not the global optimum has been found.

In the second class we have for instance:

a. Integer linear programming. The solution has to satisfy the additional requirement that it consists of integers. Methods for the case that all variables have to be integers have been developed by Gomory and by Benders, Catchpole and Kuiken.

b. Mixed discrete/continuous programming. Only part of the variables in the optimum solution must be integer-valued. Many well-known case studies can be formulated as a mixed programming problem, e.g. the travelling-salesman problem and the fixed-charge problem. Methods for solving this type of problem have been proposed by Beale and by Benders.

In the third class we have the chance-constrained programming problems. A simple example is a linear programming problem with a stochastic requirements or objective vector. Although some special problems have been studied, there are no general methods available at this time, which can cope with large problems of this type.

Finally the dynamic models can often be solved by using Bellmanns dynamic programming [5]. In many cases the problem can also be formulated in a static way which may then give rise to a large linear programming problem.

1.3 *Scope of the Monograph*

The study which is the subject of this monograph is concerned with the mathematical and computational aspects of the linear, quadratic and convex programming problem. The most important results are:

1. The development of a new computational algorithm for the simplex method, to be called revised product-form algorithm (chapter 4). It is shown in chapter 5 that this new algorithm can be expected to lead to less computing time per iteration than any existing algorithm.
2. The development of a number of methods for the convex programming problem, to be called methods of feasible directions (chapter 7). These methods are finite in the linear and quadratic case (chapters 9 and 10).
3. The discovery of an equivalence between many existing methods for the linear, quadratic or convex programming problem and the methods of feasible directions.

Apart from an introductory part I, consisting of this chapter and a chapter in which the mathematical theory of linear and convex programming is studied, the monograph can be divided into two parts. In part II (chapters 3-6) the linear programming problem will be considered. After a brief description of the simplex method, dual simplex method and primal-dual method (chapter 3), the computational aspects of the first two methods are dealt with (chapter 4). To existing algorithms as the straightforward, the explicit inverse and the product-form algorithm, the revised product-form algorithm will be added. In this new algorithm a better use is made of the fact that practical linear programming problems only have a few non-zero matrix elements. Especially the calculation of the shadow prices and the re-inversions are performed more efficiently. Another improvement is that a computational advantage is obtained from variables going into and out of the basis again and again. A theoretical comparison of the different algorithms in chapter 5 will show that the new algorithm leads to by far the fastest iterations. The number of iterations is not reduced by applying the new algorithm. The development of tricks which will reduce this number may be an important subject for future research. Finally, chapter 6 will link the two parts of the monograph since in this chapter some special linear programming problems will be considered which will occur as sub-problem in the non-linear programming procedures of part III. It will be studied how the revised product-form algorithm can be applied to their solution.

In the third part of the book the methods of feasible directions

are studied. They are iterative procedures for the problem of maximizing a concave function in a convex region and satisfy the following rules:

1. The initial solution to start the iterative procedure with must be feasible, i.e. satisfy all constraints,

2. From a feasible non-maximum solution a new feasible one is constructed with a higher value for the objective function. This process is repeated until to further increase in value can be observed.

3. The sequence of values for the objective function converges to the maximum value of the function restricted by the constraints provided such a maximum value exists. Otherwise this sequence tends to infinity.

The second rule can be subdivided into:

2a. the determination of a usable feasible direction, i.e. a direction in which we can make a finite step such that we are still within the feasible region and have obtained a higher value for the objective function;

2b. the determination of the length of the step to be taken in that direction.

Methods of feasible directions can be regarded as long-step gradient methods and should be distinguished from the small-step gradient methods which have also been applied successfully to the solution of convex programming problems [1]. It is clear that many existing methods are actually methods of feasible directions.

Chapter 7 will be concerned with the general theory of the methods of feasible directions. It will appear there that, dependent on the normalization used in the determination of the directions, several different methods can be devised. There is thus an interesting analogy with the results of Hestenes and Stiefel [25], who showed that several existing methods for solving systems of linear equations are actually methods of conjugated directions and only differ in the way they fix the directions by additional requirements. Five different normalizations are mentioned in chapter 7. It is not difficult to find other ones. With each normalization we obtain a direction-finding problem which can always be solved by means of the simplex or dual simplex method. The remainder of chapter 7 is devoted to methods for finding an initial feasible point, to the determination of the step length, to proofs of convergence and to the consequences of a relaxation of the convexity assumptions.

The computational consequences of the different normalizations will be studied in chapter 8. Here it will be shown how the revised product-form algorithm can effectively be applied to the solution of a direction-finding problem and how the final information of a direction-finding problem can be used to speed up the solution of the next one.

Chapters 9, 10 and 11 will be concerned with the application of the methods of feasible directions to the linear, quadratic and general convex programming problems, respectively. In each chapter the procedures will be schematically outlined. Some further results are:

1. By also considering the idea of conjugated directions, we obtain a number of finite methods in the case of quadratic programming.

2. One of the normalizations used appears to lead to a method equivalent to the simplex method itself in the linear programming case and to Beale's method in the quadratic case. The procedure based on this normalization can therefore be considered as an extension of the simplex method to non-linear programming.

3. Another normalization appears to lead to a method equivalent to the primal-dual method, so that the corresponding procedure can be considered as an extension of the primal-dual method to non-linear programming.

4. Methods of feasible directions, in the form as proposed in this monograph, are capable of solving very large linear programming problems by means of a sequence of smaller sub-problems, so that they contribute to a solution of the important problem of solving very large linear programming problems of a special structure.

1.4 *Notation*

Throughout the book use is made of matrix notation. Matrices are denoted by capitals, column vectors by roman characters and row vectors by using the transposition symbol T. The only exceptions to this rule occur in the computational theory of the simplex method, where, in accordance with the literature on this subject, some special vectors are denoted by greek characters. In all other cases greek characters will denote scalars. A vector inequality $x \leqq y$ means that every component of the vector x is less than or equal to the corresponding component of y. In $x \geqq 0$, the symbol 0 denotes a vector consisting of zero's only. The components of a vector x are denoted by x_j, whereas x^ν is the ν-th vector of a sequence (having components x_j^ν). A set can be denoted, either by mentioning all elements within braces, e.g. $I = \{1, \ldots, m\}$, or by giving the properties which define the set, e.g. $R = \{x \mid Ax \leqq b,\ x \geqq 0\}$.

If F(x) is a function of the vector x and if R is a set of vectors (points) x, then the notation $\text{Max}\{F(x) \mid x \in R\}$ stands for the limes superior of F(x) under the condition $x \in R$. Strictly speaking the term "Max" can only be used if it is known that there is a vector $x \in R$ for which F(x) is maximum. We also use it if this is not known, in accordance with the literature on the subject. We shall also speak of the mathematical programming problem $\text{Max}\{F(x) \mid x \in R\}$ and then we are concerned with the following questions:

1. Is there a vector x belonging to R?

2. If R is non-empty, does F(x) have a finite maximum on R?

3. If the maximum of F(x) on R is finite, what is the value of this maximum and for which vector x is it attained? If there are more vectors $x \in R$ for which this value is attained, we shall be content with one of these vectors.

At some places we shall use the operators $\forall$ and $\exists$.
"$\forall_x$ ($x \in R$) $F(x) \leqq m$" stands for "for all x belonging to R we have that $F(x) \leqq m$ holds";
"$\exists_x$ $x \in R$, $F(x) \leqq m$" stands for "there is a vector x belonging to R for which $F(x) \leqq m$ holds".

2. THEORY OF CONVEX PROGRAMMING

2.1 *Introduction*

As already pointed out in chapter 1 the convex programming problem deals with the minimization of a convex function on a convex set (or the maximization of a concave function on a convex set). In this chapter the mathematics of this problem will be studied. In the first three sections some attention will be paid to the theory of linear inequalities, to semi-definite matrices and to convex functions. No attempt will be made to give a complete survey of these fields. Sections 5 and 6 will then be concerned with the linear and the convex programming problem respectively.

2.2 *Theory of Linear Inequalities*

Let A be an m by n matrix, p and x n-component vectors; the elements of A, x, p as well as of all other matrices and vectors to be defined later on will be real numbers. Hence the rows of A and the vectors p and x can be considered as points in an n-dimensional Euclidean space E_n. The suffix T will denote transposition.

The main theorem in the theory of linear inequalities is Farkas' theorem [18]:

Theorem 1: If for all vectors x satisfying a system of linear inequalities $Ax \geqq 0$ ($Ax \leqq 0$) we have that $p^T x \geqq 0$ ($p^T x \leqq 0$), then p is a non-negative linear combination of the rows of the matrix A: $p = A^T u$, $u \geqq 0$ being an m-component vector.

Proof: Let $C = \{ y \in E_n \mid \exists_u \, u \geqq 0, \; y = A^T u \}$. The set C is a convex polyhedral cone in n-space. Hence if $y \in C$, then $\lambda y \in C$ for all $\lambda \geqq 0$ and if y_1 and $y_2 \in C$, then $y_1 + y_2 \in C$. We shall shall that if p is a vector not belonging to C, a vector x exists satisfying $Ax \geqq 0$ and $p^T x < 0$. This will prove the theorem. Hence assume $p \notin C$. Let q be the projection of p on C and $r = q - p$. It follows:

1. For all $y \in C$, $(q-p)^T(y-q) \geqq 0$; for, if <0 there would have been a point on the segment joining y and q closer to p than q is. As this point belongs to C, it follows that q could not have been the projection of p on C.

2. $r^T q = 0$; for, if $\neq 0$ there would have been a point on the line through the origin and q closer to p than q is. As this point belongs to C, it follows that q could not have been the projection of p on C.

From 1 and 2 we obtain $r^T y \geqq 0$ for all $y \in C$. If $a_{i.}^T$ is the i-th row of the matrix A, so that $a_{i.}^T = e_i^T A$ (e_i being the i-th unit vector), it follows that for the corresponding point in n-space holds $a_{i.} \in C$; hence $r^T a_{i.} \geqq 0$ or $a_{i.}^T r \geqq 0$, so that $Ar \geqq 0$ holds.

Since moreover $p^T r = q^T r - r^T r = - r^T r < 0$ we see that r is the vector looked for.

Theorem 2: The sets of linear inequalities $A\,x \geqq 0$, $A^T u = 0$, $u \geqq 0$ always have a solution x', u', satisfying $Ax' + u' > 0$ and $u_i' \, a_{i.}^T \, x' = 0$ for all i.

Proof: If $a_{i.}^T \, x \leqq 0$ for all x satisfying $Ax \geqq 0$, then by Farkas' theorem $-a_{i.} = A^T u$ for some $u \geqq 0$, so that $A^T u^i = 0$ would hold for some vector $u^i \geqq 0$ with $u_i^i > 0$. Hence either $a_{i.}^T \, x^i > 0$ for some x^i satisfying $Ax^i \geqq 0$, or $A^T u^i = 0$ for some $u^i \geqq 0$, $u_i^i > 0$. Define $u^i = 0$ in the former case and $x^i = 0$ in the latter, so that $a_{i.}^T \, x^i + u_i^i > 0$ will hold for all i. Let $x' = \sum_{i=1}^{m} x^i$ and $u' = \sum_{i=1}^{m} u^i$, then:

1. $Ax' \geqq 0$ follows from $Ax^i \geqq 0$ for all i,
2. $A^T u' = 0$ follows from $A^T u^i = 0$ for all i,
3. $u' \geqq 0$ follows from $u^i \geqq 0$ for all i,
4. $Ax' + u' > 0$ follows from $a_{i.}^T \, x^r + u_i^r \geqq 0$ for all r and > 0 for r=i.
5. $u_i' \, a_{i.}^T \, x' = 0$ follows from $u_i^h \, a_{i.}^T \, x' = 0$ for all h.

This proves the theorem.

Theorem 2 was first formulated by Tucker [35] and is called the "key theorem". A direct elementary proof can be found in a paper by Good [24]. From the key theorem it is not difficult to derive Farkas' theorem as well as many other theorems concerning linear inequalities, for instance:

Theorem 3: The system of linear inequalities $Ax < 0$ is inconsistent if and only if $A^T u = 0$ for some $u \geqq 0$ and $u_i > 0$ for at least one i, hence if and only if one of the rows of the matrix A is linearly dependent with non-positive coefficients on the other rows of A.

Proof: From the key theorem we know that vectors x', u' exist satisfying: $-Ax' \geqq 0$, $A^T u' = 0$, $u' \geqq 0$, and $-a_{i.}^T \, x' + u_i' > 0$. If $u_i' = 0$ for all i the system $a_{i.}^T \, x < 0$ would have a solution, namely x', so that if this system is inconsistent we necessarily have $A^T u = 0$, $u \geqq 0$ and $u_i > 0$ for some u and i. On the other hand if $A^T u = 0$, $u \geqq 0$ and $u_i > 0$ for some u and i, we shall have for all x that $x^T A^T u = 0$ or $u^T Ax = 0$ and for all x satisfying $Ax \leqq 0$ that $a_{i.}^T \, x = 0$ if $u_i > 0$. Hence no x can satisfy $a_{i.}^T \, x < 0$ for all i, so that this system is inconsistent.

Corollary: From the key theorem it also follows immediately that u can be chosen such that $u_i > 0$ holds for all i satisfying (for all x) $a_{i.}^T \, x = 0$ if $Ax \leqq 0$.

2.3 *Semi-definite Matrices*

A square and symmetric matrix C will be called: positive definite if $s^T C s \geq 0$ holds for any vector $s \neq 0$; positive semi-definite if $s^T C s \geqq 0$ holds for any vector s; negative (semi-)definite if -C is positive (semi-)definite.

Theorem 1: Any principal minor of a positive (semi-)definite matrix is again positive (semi-)definite.
Proof: Let the principal minor P of C consist of those rows and columns of C for which $i \in I_1 \subset I = \{1, \ldots, m\}$. Let s_i be arbitrary if $i \in I_1$ and = 0 if $i \notin I_1$. Then

$$0 \leqq s^T C s = \sum_{i=1}^{m} \sum_{j=1}^{m} c_{ij}\, s_i\, s_j = \sum_{i \in I_1} \sum_{j \in I_1} p_{ij}\, s_i\, s_j ,$$

which proves the theorem. Strict inequality will hold if C is positive definite and $s \neq 0$.
Corollary: The diagonal elements of a positive (semi-)definite matrix are always positive (non-negative).
Theorem 2: If C is positive semi-definite and $s^T C s = 0$, then $Cs = 0$.
Proof: $0 \leqq (t + \lambda s)^T C (t + \lambda s) = t^T C\, t + 2\lambda t^T C s$ for all n-component vectors t and scalars λ. Hence $t^T C s = 0$ for all t from which we obtain $Cs = 0$.
Theorem 3: If C is positive definite, then C^{-1} exists and is also positive definite.
Proof: Let C be positive definite. Suppose C has a vanishing determinant, so that the system of linear inequalities $Cs = 0$ would have a non-trivial solution s', then $(s')^T C s' = 0$, $s' \neq 0$ would hold, contrary to the assumption that C is positive definite. Hence C^{-1} exists. Take s arbitrary $\neq 0$ and call $t = C^{-1} s$. We then have $t \neq 0$ and $s^T C^{-1} s = s^T C^{-1} C C^{-1} s = t^T C\, t > 0$. Hence C^{-1} is positive definite.
Theorem 4: If A is an arbitrary (possibly rectangular) matrix, then AA^T and $A^T A$ are positive semi-definite.
Proof: $s^T A A^T s = (s^T A)(A^T s) = t^T t \geqq 0$ if we define $t = A^T s$.
Theorem 5: A symmetric and idempotent matrix is positive semi-definite.
Proof: Let A be symmetric and idempotent; hence $A^T = A$ and $A^2 = A$, so that $A = A^T A$. From this and theorem 4 the semi-definiteness follows.
Theorem 6: Let the positive semi-definite matrix C be decomposed as $C = \begin{pmatrix} P & Q \\ Q^T & S \end{pmatrix}$, where P is n_1 by n_1 and positive definite, S is n_2 by n_2 and positive semi-definite ($n_1 + n_2 = n$) and Q is an n_1 by n_2 matrix. Then the n_2 by n_2 matrix $-Q^T P^{-1} Q + S$ is also positive semi-definite.
Proof: Let $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ where x_1 has n_1 components and x_2 has n_2 components.
We have

$$0 \leqq (x_1^T,\ x_2^T) \begin{pmatrix} P & Q \\ Q^T & S \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = x_1^T\, P x_1 + 2\, x_1^T Q x_2 + x_2^T S x_2 .$$

Take $x_1 = -P^{-1} Q x_2$. Hence

$$x_2^T Q^T P^{-1} Q x_2 - 2\, x_2^T Q^T P^{-1} Q x_2 + x_2^T S x_2 \geqq 0$$

for any vector x_2, or $x_2^T(-Q^T P^{-1}Q + S)x_2 \geqq 0$ for any vector x_2, which proves the theorem.

Let C be a positive definite matrix and p an n-component vector. We shall consider the system of linear equations $Cx = p$ and perform the following procedure:
1. Take x^0 arbitrary, call $g^0 = p - Cx^0$, take a vector $s^0 \neq 0$ and define.

$$\lambda_0 = \frac{(g^0)^T s^0}{(s^0)^T C\, s^0} \text{ and } x^1 = x^0 + \lambda_0 s^0. \tag{2.3.1}$$

2. Having already found the points x^h, vectors s^h and scalars λ_h for $h = 0, 1, \ldots, k-1$ we define

$$x^k = x^{k-1} + \lambda_{k-1}\, s^{k-1}, \tag{2.3.2}$$

and

$$g^k = p - C\, x^k. \tag{2.3.3}$$

For s^k we now take a vector, satisfying

$$(s^h)^T C s^k = 0, \quad h = 0, 1, \ldots, k-1 \text{ and } s^k \neq 0, \tag{2.3.4}$$

provided such a vector exists,
and define:

$$\lambda_k = \frac{(g^k)^T\, s^k}{(s^k)^T C\, s^k}; \tag{2.3.5}$$

We stop if such a vector does no longer exist.
We now have:
Theorem 7: The vectors s^h generated by the above mentioned procedure are linearly independent.

Proof: Suppose $\sum_0^p u_h\, s^h = 0$ and $\sum_0^{p-1} u_h\, s^h \neq 0$, so that $u_p \neq 0$.

Hence $\sum_0^p u_h\, C\, s^h = 0$. Since $(s^p)^T C\, s^h = 0$ for $h = 0, 1, \ldots, p-1$, we have $u_p (s^p)^T C\, s^p = 0$. Since $u_p \neq 0$ it follows that $(s^p)^T C s^p = 0$, hence $s^p = 0$. From this it follows that the vectors s^h are linearly independent.
Corollary: The procedure ends after a finite number $k\ (\leqq n)$ of steps.

Theorem 8: Let $\bar{x} = x^0 + \sum_{h=0}^{k-1} \lambda_h\, s^h$, hence $\bar{x} = x^k$.

Then $\bar{x}$ solves the set of linear equations $Cx = p$.
Proof: Let $\bar{g} = p - C\bar{x}$ and suppose $\bar{g} \neq 0$, so that a vector s exists with $\bar{g}^T s \neq 0$ (we can take $s = \bar{g}$).

Let $s = \sum_{h=0}^{k-1} \mu_h s^h + t$ with $\mu_h = \frac{(s^h)^T C\, s}{((s^h)^T C\, s^h}$

Hence

$$(s^i)^T C\, t = (s^i)^T C\, s - \sum_{h=0}^{k-1} \mu_h (s^i)^T C\, s^h =$$

$$= (s^i)^T C\, s - \mu_i (s^i)^T C\, s^i = 0 \text{ for } i = 0, 1, \ldots, k-1.$$

Now for all i the relation $\bar{g} = g^i - \sum_{h=i}^{k-1} \lambda_h C\, s^h$ holds, so that

$$\bar{g}^T s^i = (g^i)^T s^i - \lambda_i (s^i)^T C\, s^i = 0 \text{ for } i = 0, 1, \ldots, k-1.$$

Consequently $\bar{g}^T s = \bar{g}^T t \neq 0$, so that t would be a vector satisfying $t \neq 0$ and $(s^i)^T C\, t = 0$ for $i = 0, 1, \ldots, k-1$, contrary to the assumption that such a vector does not exist in x^k. Hence $\bar{g} = g^k \neq 0$ cannot hold, which proves the theorem.

The directions s^h are called conjugated directions. They are not uniquely determined. Many methods for solving systems of linear equations are actually working with conjugated directions and differ only in the way they fix the vectors s^h by means of additional requirements as has been pointed out by Hestenes and Stiefel [25]. Among them are the Gauss elimination method and the conjugate gradients method.

2.4 *Convex Functions and Regions*

Let $F(x) = F(x_1, \ldots, x_n)$ be a function of the n real variables $x_1, \ldots, x_n$, hence of the vector $x \in E_n$. We shall suppose that F(x) is differentiable with continuous partial derivatives. Hence, if the gradient vector in x is denoted by g(x), i.e.

$$\{g(x)\}^T = \{\frac{\partial F}{\partial x_1}, \ldots, \frac{\partial F}{\partial x_n}\}, \tag{2.4.1}$$

the components $g_j(x)$ are continuous functions of x.

A function F(x) will be called convex if linear interpolation between values of the function never underestimates the real value at the interpolated point; F(x) will be concave if -F(x) is convex. Hence, for any two vectors x and y and any scalar λ, $0 < \lambda < 1$, F(x) is convex if

$$F\{(1-\lambda)x + \lambda y\} \leqq (1-\lambda) F(x) + \lambda F(y), \tag{2.4.2}$$

and concave if

$$F\{(1-\lambda)x + \lambda y\} \geqq (1-\lambda) F(x) + \lambda F(y). \tag{2.4.3}$$

A function F(x) is strictly convex (concave) if strict inequality

holds in (2.4.2) ((2.4.3)) for $x \neq y$. It is clear that a linear function is convex as well as concave but neither strictly convex nor strictly concave. We have the following theorems:

Theorem 1: $F(x)$ is convex if and only if the one dimensional function $\varphi(\lambda) = F(x + \lambda s)$ is convex for any two vectors x and s.

Proof: The "only if" part is trivial. Now suppose $\varphi(\lambda)$ is convex for any two vectors x and s. Take x and $y \in E_n$ arbitrarily, call $y - x = s$, then

$$F\{(1-\lambda)x + \lambda y\} = F(x + \lambda s) = \varphi(\lambda) = \varphi\{(1-\lambda).0 + \lambda.1\} \leqq (1-\lambda)\varphi(0) +$$

$$+ \lambda\varphi(1) = (1-\lambda)F(x) + \lambda F(y), \text{ q.e.d.}$$

Theorem 2: If $x^h \in E_n$ for $h = 1, \ldots, m$ and $F(x)$ convex, then

$$F\{\frac{1}{m} \sum_{h=1}^{m} x^h\} \leqq \frac{1}{m} \sum_{h=1}^{m} F(x^h).$$

Proof: The theorem holds for $m = 1$ or 2. Suppose it holds for m. We shall prove that it then also holds for $m+1$:

$$F\{\frac{1}{m+1} \sum_{h=1}^{m+1} x^h\} = F\{\frac{m}{m+1} \cdot \frac{1}{m} \sum_{h=1}^{m} x^h + \frac{1}{m+1} x^{m+1}\} \leqq$$

$$\leqq \frac{m}{m+1} F\{\frac{1}{m} \sum_{h=1}^{m} x^h\} + \frac{1}{m+1} F(x^{m+1}) \leqq \frac{1}{m+1} \sum_{h=1}^{m+1} F(x^h), \text{ q.e.d.}$$

Theorem 3: The following statements are equivalent:

1. $F(x)$ is convex;
2. $F(x_2) - F(x_1) \geqq g(x_1)^T(x_2 - x_1)$ for any two vectors x_1 and $x_2 \in E_n$;
3. $g(x + \lambda s)^T s$ is monotonously non-decreasing as a function of λ for any x and s;

and when all second partial derivatives exist and form a matrix $C(x)$:

4. $C(x)$ is positive semi-definite for any $x \in E_n$.

Proofs: 1 $\longrightarrow$ 2. Let $F(x)$ be convex. Hence, for any two vectors x_1 and $x_2 \in E_n$:

$$F\{(1-\lambda)x_1 + \lambda x_2\} \leqq (1-\lambda) F(x_1) + \lambda F(x_2) \quad , \quad 0 < \lambda < 1, \text{ or}$$

$$F\{(x_1 + \lambda (x_2 - x_1)\} \leqq F(x_1) + \lambda\{F(x_2) - F(x_1)\}, \text{ or}$$

$$\frac{F\{x_1 + \lambda(x_2 - x_1)\} - F(x_1)}{\lambda} \leqq F(x_2) - F(x_1). \qquad (2.4.4)$$

Taking the limit for $\lambda \longrightarrow 0$ we obtain

$$g(x_1)^T(x_2 - x_1) \leqq F(x_2) - F(x_1). \qquad (2.4.5)$$

$2 \longrightarrow 3$. $\varphi(\lambda) = F(x + \lambda s)$ is convex, so that by (2.4.5) we have:

$$\varphi'(\lambda_1)\,(\lambda_2-\lambda_1) \leqq \varphi(\lambda_2) - \varphi(\lambda_1),$$
$$\varphi'(\lambda_2)\,(\lambda_1-\lambda_2) \leqq \varphi(\lambda_1) - \varphi(\lambda_2).$$

Hence if $\lambda_2 > \lambda_1$ we obtain:

$$\varphi'(\lambda_1) \leqq \frac{\varphi(\lambda_2) - \varphi(\lambda_1)}{\lambda_2-\lambda_1} \leqq \varphi'(\lambda_2), \qquad (2.4.6)$$

so that $\varphi'(\lambda_1) \leqq \varphi'(\lambda_2)$ or $g(x+\lambda_1 s)^T s \leqq g(x+\lambda_2 s)^T s$ for $\lambda_1 < \lambda_2$, q.e.d.

$3 \longrightarrow 1$. Suppose $g(x+\lambda s)^T s$ is monotonously non-decreasing as a function of λ for all x and s. Hence $\varphi'(\lambda_1) \leqq \varphi'(\lambda_2)$ if $\lambda_1 < \lambda_2$. We have, if $0 < \mu < 1$ and $\lambda_2 > \lambda_1$:

$$0 \leqq \mu(\lambda_2-\lambda_1) \int_0^1 d\tau \left[\varphi'\{\lambda_1+\tau(\lambda_2-\lambda_1)\} - \varphi'\{\lambda_1 + \mu\,\tau\,(\lambda_2-\lambda_1)\}\right] =$$
$$= (1-\mu)\,\varphi(\lambda_1) + \mu\,\varphi(\lambda_2) - \varphi\{(1-\mu)\,\lambda_1+\mu\,\lambda_2\},$$

from which the convexity of $\varphi(\lambda)$, hence of F(x), follows.

$3 \longrightarrow 4$. From (2.4.6) we obtain $\varphi''(\lambda) \geqq 0$ for all λ, hence $s^T C(x+\lambda s)s \geqq 0$ for all x, s and λ, so that $s^T C(x)s \geqq 0$ will hold for all x and s, from which the positive semi-definiteness of C(x) for all x follows.

$4 \longrightarrow 3$. If C(x) is positive semi-definite, then $s^T C(x + \lambda s)s \geqq 0$ for all x, s and λ; hence $\varphi''(\lambda) \geqq 0$ for all λ, from which it trivially follows that $\varphi'(\lambda)$ is monotonously non-decreasing.

Corollaries:

1. If $F(x_2) < F(x_1)$ and F(x) convex, then $g(x_1)^T(x_2 - x_1) < 0$ (follows from (2.4.5)).
2. If $g(x_1)^T(x - x_1) \geqq 0$ for all x and F(x) convex, then F(x) has a minimum in x_1.
3. The quadratic function $F(x) = p^T x + \frac{1}{2}x^T C x$ is convex if and only if the n by n symmetric matrix C is positive semi-definite.

Theorem 4: A positive linear combination of convex functions is again convex and strictly convex if at least one of the functions is strictly convex.

Proof: Follows immediately from the definition of convexity.

Let I be a finite or infinite set of indices, let a_i be a vector and b_i a scalar for all $i \in I$. Then we have:

Theorem 5: If $F(x) = \sup_i \{a_i^T x - b_i \mid i \in I\}$, then F(x) is convex.

Proof: Take x_1 and $x_2 \in E_n$ and $0 < \lambda < 1$:

$$F\{(1-\lambda)x_1+\lambda x_2\} = \sup_i [a_i^T\{(1-\lambda)x_1 + \lambda x_2\} - b_i] \leqq$$
$$\leqq (1-\lambda) \sup_i \{a_i^T x_1 - b_i\} + \lambda \sup_i \{a_i^T x_2 - b_i\} = (1-\lambda)\,F(x_1) + \lambda\,F(x_2).$$

Definition: A set R in n-space is convex if it contains with any two points $x_1 \in R$ and $x_2 \in R$ the segment joining x_1 and x_2, i.e. if:

$$\forall_{x_1,\ x_2}\ (x_1, x_2 \in R)\ \{x \mid \exists_\lambda,\ 0 \leqq \lambda \leqq 1,\ x = (1-\lambda)x_1 + \lambda x_2\} \subset R.$$

The following theorems will hold:

Theorem 6: The intersection of a finite or infinite number of convex sets is convex.

Proof: Let R_i be convex for $i \in I$ and let x_1 and $x_2 \in \cap R_i$ and $0 \leqq \lambda \leqq 1$. x_1 and x_2, hence $(1-\lambda)x_1 + \lambda x_2 \in R_i$ for all i, hence $\in \cap R_i$, q.e.d.

Theorem 7: If $f_i(x)$ are convex functions of $x \in E_n$, $i \in I$ and if b_i are scalars, then

$R = \{x \mid \forall_i\ (i \in I)\ f_i(x) \leqq b_i\}$ is convex and closed.

Proof: Let $R_i = \{x \mid f_i(x) \leqq b_i\}$, then $R = \cap R_i$ and it suffices to prove that R_i is convex and closed. Let x_1 and x_2 be arbitrary points of R_i and $0 \leqq \lambda \leqq 1$, then $f_i\{(1-\lambda)x_1 + \lambda x_2\} \leqq (1-\lambda)\ f_i(x_1) + \lambda f_i(x_2) \leqq b_i$, hence $(1-\lambda)x_1 + \lambda x_2 \in R_i$. If $x_j \in R_i$, $j = 1, 2, \ldots$, and x is a point of accumulation of this sequence, then $f_i(x) \leqq b_i$ follows from $f_i(x_j) \leqq b_i$ for all j, hence $x \in R_i$, so that R_i is closed.

Theorem 8: If R and F(x) are convex, then $\{x \in R \mid F(x) \leqq m\}$ is a convex (possibly empty) subset of R for each number m.

Proof: $R_1 = \{x \mid F(x) \leqq m\}$ is convex for each m. We further have: $\{x \in R \mid F(x) \leqq m\} = R \cap R_1$.

Theorem 9: Any local minimum of a convex function attained on a convex set is a global minimum. The set of all these minima is convex.

Proof: Suppose that $x_1 \in R$ and $x_2 \in R$ are local minima and that $F(x_2) < F(x_1)$ would hold. From theorem 3, corollary 1, it then follows that $g(x_1)^T(x_2 - x_1) < 0$ holds, so that F(x) would decrease if we go from x_1 in the direction of x_2. Since the segment joining x_1 and x_2 belongs to R, x_1 could not be a local minimum. Hence $F(x_2) = F(x_1)$ will hold. The second assertion follows immediately from theorem 8.

Theorem 10: A strictly convex function has only one minimum on a convex set or is unbounded below.

Proof: Let F(x) be strictly convex and let $F(x_1) = F(x_2)$ for $x_1 \in R$ and $x_2 \in R$. From the strict convexity assumption it follows that any point on the segment joining x_1 and x_2 will have a lower function value, so that there cannot be two minima.

Theorem 11: Let $f_i(x)$ be convex with continuous gradient vector $q_i(x)$ for $i = 1, \ldots, m$ and let $R = \{x \mid f_i(x) \leqq b_i,\ i = 1, \ldots, m\}$ be non-empty. Then the system of inequalities $f_i(x) < b_i$ will be inconsistent if and only if non-negative numbers u_i exist, $\Sigma u_i = 1$, such that $\Sigma u_i q_i(x) = 0$ and $\Sigma u_i f_i(x) = \Sigma u_i b_i$ for all $x \in R$.

Proof: If non-negative numbers u_i exist, $\Sigma u_i = 1$, such that $\Sigma u_i f_i(x) = \Sigma u_i b_i$ for all $x \in R$, then the system $f_i(x) < b_i$, $i = 1, \ldots, m$ is necessarily inconsistent. For, if, for some $\bar{x}$, $f_i(\bar{x}) < b_i$ should hold for all i and if $u_r > 0$, then $u_r f_r(\bar{x}) < u_r b_r$

and for $i \neq r$, $u_i f_i(\bar{x}) \leqq u_i b_i$, so that $\Sigma u_i f_i(\bar{x}) < \Sigma u_i b_i$ would hold.

Now suppose the system $f_i(x) < b_i$, $i = 1, \ldots, m$ is inconsistent. Take a point $y \in R$, let $f_i(y) = b_i$ for $i \in I_1$ and $f_i(y) < b_i$ for $i \in I_2$, $I_1 + I_2 = \{1, \ldots m\}$, and suppose the system $q_i(y)^T s < 0$, $i \in I_1$, to be consistent. It is clear that in that case $f_i(y + \lambda s) < f_i(y) \leqq b_i$ would hold for some $\lambda > 0$ and all i, contrary to our assumption. Hence the system of linear inequalities $q_i(y)^T s < 0$, $i \in I_1$, will be inconsistent, so that by theorem 3 of section 2.2 non-negative numbers u_i can be found, $u_i > 0$ for at least one i or equivalently $\sum u_i = 1$, such that $\sum_{i \in I_1} u_i q_i(y) = 0$ or $\sum_{i=1}^{m} u_i q_i(y) = 0$, $u_i = 0$ if $f_i(y) < b_i$. Hence $\sum_{i=1}^{m} u_i f_i(y) = \sum_{i=1}^{m} u_i b_i$ will hold. By theorem 3 the function $\Sigma u_i f_i(x)$ is convex; its gradient vector equals zero in the point $y \in R$, so that the function assumes a minimum there (corollary 2 of theorem 3). Hence, for all x, $\Sigma u_i f_i(x) \geqq \Sigma u_i f_i(y) = \Sigma u_i b_i$ but, for $x \in R$, $\Sigma u_i f_i(x) \leqq \Sigma u_i b_i$. Hence $\Sigma u_i f_i(x) = \Sigma u_i b_i$ will hold for all $x \in R$, $\Sigma u_i = 1$. Moreover any $x \in R$ will be an unrestricted minimum of $\Sigma u_i f_i(x)$, so that $\Sigma u_i q_i(x) = 0$ will hold for all $x \in R$. This completes the proof.

Corollary: If the system $f_i(x) < b_i$ is inconsistent and if $f_r(x) = b_r$ holds for all $x \in R$, then the quantities u_i of theorem 11 can be chosen such that $u_r > 0$ holds. This follows from the corollary of theorem 3, section 2.2.

We have thus proved that, if for no x the system $f_i(x) < b_i$ can be satisfied for all i, the outward pointing normals $q_i(x)$ in any x satisfying $f_i(x) \leqq b_i$ for all i will be linearly dependent with non-negative coefficients which do not depend on x.

An example of such a system is $x^2 + y < 0$, $x^2 - y < 0$, $z < 1$. For no x, y and z can this system be satisfied. The set R is the line segment $x = 0$, $y = 0$, $z \leqq 1$. We have $q_1^T = (2x, 1, 0)$, $q_2^T = (2x, -1, 0)$ and $q_3^T = (0, 0, 1)$, so that $q_1 + q_2 = 0$ holds for all $x \in R$.

Theorems 1-5 and 7-11 can also be formulated for concave functions.

2.5 *The Linear Programming Problem*

Let A be an m by n matrix with rows $a_{i.}^T$, $i \in I = \{1, \ldots, m\}$, columns $a_{.j}$, $j \in J = \{1, \ldots, n\}$ and elements a_{ij}; let x and p be n-component vectors, y and b m-component vectors. The linear programming problem will now be defined as

$$\text{Max}\{p^T x \mid Ax \leqq b,\ x \geqq 0\}, \tag{2.5.1}$$

i.e. the vector(s) x must be determined among those satisfying the system of linear inequalities $Ax \leqq b$, $x \geqq 0$, for which the linear function $p^T x$, to be called objective function, assumes its maximum, or it must be concluded either that the system $Ax \leqq b$,

$x \geqq 0$ will be satisfied for no x, in which case the problem will not be feasible, or that for no x', $Ax' \leqq b$, $x' \geqq 0$, we have $p^T x \leqq p^T x'$ for all x satisfying $Ax \leqq b$, $x \geqq 0$ in which case the maximum will be said to be infinite.

It is clear that any linear programming problem - also those with equality constraints or unrestricted variables - can be written in the form (2.5.1). We shall sometimes write $Ax + y = b$, $y \geqq 0$ or $\bar{A}\bar{x} = b$, where

$$\bar{A} = (A, E) \; ; \; \bar{x} = \begin{pmatrix} x \\ y \end{pmatrix}, \tag{2.5.2}$$

(E being an m by m unit matrix).

We define:

a. The constraint set or feasible region R: the convex polyhedron in n space formed by the constraints $Ax \leqq b$, $x \geqq 0$.

b. A feasible point or vector: any $x \in R$ or any x satisfying the constraints. If such an x exists the problem will be called feasible.

c. A basic feasible point or vertex: any $x \in R$ which cannot be written as a convex linear combination of two other feasible points.

d. An optimum feasible point or optimum solution: a vector x which solves (2.5.1). It can easily be proved that if there is an optimum solution, there will also be an optimum basic solution.

For each $x \in R$ we can define sets $I(x) \subset I$ and $J(x) \subset J$ such that $a_{i.}^T x = b_i$ if $i \in I(x)$ and $a_{i.}^T x < b_i$ if $i \in I - I(x)$; $x_j = 0$ if $j \in J(x)$ and $x_j > 0$ if $j \in J - J(x)$. (I(x) may be empty or = I; J(x) may be empty or = J).

A direction s in x will be called feasible if none of the constraints will be violated by making a sufficiently small step in the direction s, i.e. s is feasible in x if

$$\exists_\lambda \lambda > 0, \; x + \lambda s \in R. \tag{2.5.3}$$

For a direction s to be feasible in x it is necessary and sufficient that

$$\begin{aligned} a_{i.}^T s &\leqq 0 \text{ if } i \in I(x), \\ s_j &\geqq 0 \text{ if } j \in J(x). \end{aligned} \tag{2.5.4}$$

A feasible direction in x is usable if in addition:

$$p^T s > 0. \tag{2.5.5}$$

A point $x \in R$ will be an optimum point if no usable vector exists in x, hence if $p^T s \leqq 0$ for all vectors s satisfying (2.5.4). We can now state:

Theorem 1: A point $x \in R$ is optimum if and only if the objective vector p can be written as a non-negative linear combination of the outward-pointing normals in x, i.e. if non-negative numbers

u_i and v_j ($i \in I(x)$, $j \in J(x)$) can be found such that:

$$p_j = \sum_{i \in I(x)} u_i \, a_{ij} - \sum_{j \in J(x)} v_j \, e_j ,$$

where e_j is the unit vector with 1 at place j.

Proof: Let x be an optimum point, so that $p^T s \leqq 0$ holds for any s, satisfying $a_i^T s \leqq 0$ if $i \in I(x)$, and $-s_j \leqq 0$ if $j \in J(x)$. We can therefore apply Farkas' theorem (section 2.2, theorem 1) which gives us the result immediately. The reverse is trivial.

Hence in an optimum point x, y we have:

$$\begin{aligned} &p = A^T u - v, \; u \geqq 0, \; v \geqq 0, \\ &u^T y = 0, \; v^T x = 0, \end{aligned} \tag{2.5.6}$$

which we obtain by introducing $u_i = 0$ if $i \in I - I(x)$, hence if $y_i > 0$, and $v_j = 0$ if $j \in J - J(x)$, i.e. if $x_j > 0$.

Let us now consider the linear programming problem:

$$\text{Min} \{ b^T u \mid A^T u \geqq p, \; u \geqq 0 \}, \tag{2.5.7}$$

which we obtain from (2.5.1) by:
1. interchanging the vectors p and b (i.e. objective vector and requirements vector);
2. transposing the matrix A;
3. reversing the inequality signs;
4. interchanging maximum and minimum.

Problem (2.5.7) will be called the dual problem of the primal problem (2.5.1). Instead of $A^T u \geqq p$, we can also write $A^T u - v = p$, $v \geqq 0$.

We have the following theorems:

Theorem 2: The dual problem of the dual problem is the primal problem.

Proof: Follows immediately from the rules 1-4.

Theorem 3: If $\begin{pmatrix} x \\ y \end{pmatrix}$ and $\begin{pmatrix} u \\ v \end{pmatrix}$ are feasible solutions of the primal and dual problem respectively, then $p^T x \leqq b^T u$.

Proof: $p^T x = u^T A x - v^T x = u^T b - u^T y - v^T x \leqq b^T u$.

Theorem 4: If $\begin{pmatrix} x' \\ y' \end{pmatrix}$ is an optimum feasible solution of the primal problem, then the dual problem is also feasible and for any optimum solution $\begin{pmatrix} u' \\ v' \end{pmatrix}$ of it we have:

$$p^T x' = b^T u', \; u'^T y' = 0, \; v'^T x' = 0.$$

Proof: Let $\begin{pmatrix} x' \\ y' \end{pmatrix}$ be an optimum solution of (2.5.1). It then follows from (2.5.6) that there exists a feasible solution $\begin{pmatrix} u' \\ v' \end{pmatrix}$ of (2.5.7)

with $u'^T y' = 0$ and $v'^T x' = 0$. The proof of theorem 3 shows that $p^T x' = b^T u'$ will hold, so that $\begin{pmatrix} u' \\ v' \end{pmatrix}$ will be an optimum solution. This proof also shows that the relations $u^T y' = 0$, $v^T x' = 0$ will hold for any other optimum solution of the dual problem.

Theorem 5: If $\begin{pmatrix} x \\ y \end{pmatrix}$ and $\begin{pmatrix} u \\ v \end{pmatrix}$ are feasible solutions of the primal and dual problem respectively and if $u^T y = 0$ and $v^T x = 0$, then they are optimum solutions.

Proof: Follows immediately from theorem 3 and its proof.

Bearing in mind that each equation can be written as two inequalities and each unrestricted variable as the difference of two restricted variables, one can easily derive the following theorems (primes denote optimum solutions, if they exist).

Theorem 6: If the primal problem is: $\text{Max}\{p^T x \mid Ax = b,\ x \geqq 0\}$, then the dual problem is: $\text{Min}\{b^T u \mid A^T u - v = p,\ v \geqq 0\}$ and the duality relations are: $p^T x' = b^T u'$; $v'^T x' = 0$.

Theorem 7: If the primal problem is:

$$\text{Max}\{p^T x + q^T y \mid Ax + y = b,\ x \geqq 0,\ y \geqq 0\},$$

then the dual problem is:

$$\text{Min}\{b^T u \mid A^T u - v = p,\ u \geqq q,\ v \geqq 0\},$$

and the duality relations are:

$$p^T x' + q^T y' = b^T u',$$
$$v'^T x' = 0,\ u'^T y' = q^T y'.$$

Theorem 8: If the primal problem is:

$$\text{Max}\{p_1^T x_1 + p_2^T x_2 \mid A_{11} x_1 + A_{12} x_2 + y_1 = b_1;\ y_1 \geqq 0;\ A_{21} x_1 + A_{22} x_2 = b_2;\ x_1 \geqq 0\},$$

then the dual problem is:

$$\text{Min}\{b_1^T u_1 + b_2^T u_2 \mid A_{11}^T u_1 + A_{21}^T u_2 - v_1 = p_1;\ v_1 \geqq 0;\ A_{12}^T u_1 + A_{22}^T u_2 = p_2;\ u_1 \geqq 0\}$$

and the duality relations are $p_1^T x_1' + p_2^T x_2' = b_1^T u_1' + b_2^T u_2'$;
$v_1'^T x_1' = 0$; $u_1'^T y_1' = 0$.

Hence the dual variable corresponding to a primal equality will not be restricted and a dual equality will correspond to an unrestricted primal variable.

We always have the following rules:
1. the primal and dual solution have the same value;
2. the restricted variables of the two optimum solutions have the so-called complementary slackness property.

2.6 *The Convex Programming Problem*

Let $f_i(x)$ be non-linear convex differentiable functions of $x \in E_n$ and $i \in I_1 = \{1, \ldots, m_1\}$ with continuous partial derivatives. Let a_i be n-component vectors for $i \in I_2 = \{m_1 + 1, \ldots, m\}$ and let b_i be real numbers for $i \in I = I_1 + I_2$. Further $J = \{1, \ldots, n\}$. Then the convex region R will be defined by:

$$R = \{ x \mid f_i(x) \leqq b_i,\ i \in I_1;\ a_i^T x \leqq b_i,\ i \in I_2;\ x \geqq 0 \}. \tag{2.6.1}$$

R will be called the feasible region or constraint set; any $x \in R$ will be a feasible point or vector. We shall sometimes write:

$$\begin{aligned} y_i &= b_i - f_i(x) \geqq 0,\ i \in I_1, \\ &= b_i - a_i^T x \geqq 0,\ i \in I_2. \end{aligned} \tag{2.6.2}$$

The gradient vector of $f_i(x)$ will be denoted by $q_i(x)$:

$$q_i(x) = \{ \frac{\partial f_i}{\partial x_j},\ j = 1, \ldots, n \}. \tag{2.6.3}$$

If $f_i(x) = b_i$ for some x and i, then $q_i(x)$ is the outward-pointing normal of $f_i(x)$ in x.

We shall impose the following regularity condition on R:

C1: $\forall_i (i \in I_1) \exists_x\ x \in R,\ f_i(x) < b_i$.

Theorem 1: If C1 is satisfied then $f_i(x) < b_i$ holds for some $x \in R$ and all i, i.e.

$$\exists_x\, x \in R,\ \forall_i\, (i \in I_1)\ f_i(x) < b_i.$$

Proof: Let, for all $i \in I_1$, $x^i \in R$ be such that $f_i(x^i) < b_i$. Take $x = \frac{1}{m_1} \sum_{h=1}^{m_1} x^h$, then by theorem 2, section 2.4 we have

$$f_i \{ \frac{1}{m_1} \sum_{h=1}^{m_1} x^h \} \leqq \frac{1}{m_1} \sum_{h=1}^{m_1} f_i(x^h) < b_i$$

since $f_i(x^h) \leqq b_i$ for all h and $< b_i$ for $h = i$.

Let $x \in R$, let $I_1(x) = \{ i \mid f_i(x) = b_i \}$, $I_2(x) = \{ i \mid a_i^T x = b_i \}$ and $J(x) = \{ j \mid x_j = 0 \}$.
We now have:
Theorem 2: If C1 is satisfied, $x \in R$ and

$$\sum_{i \in I_1(x)} u_i \, q_i(x) + \sum_{i \in I_2(x)} u_i \, a_i - \sum_{j \in J(x)} v_j \, e_j = 0; \; u_i \geqq 0; \; v_j \geqq 0,$$

then $u_i = 0$ if $i \in I_1(x)$, i.e. in no $x \in R$ will one of the outward-pointing normals of the non-linear hypersurfaces $f_i(x) = b_i$ be linearly dependent with non-positive coefficients on the other outward-pointing normals in that point.
Proof: Suppose $u_i > 0$ for some $i \in I_1(x)$, say $u_r > 0$. The convex function

$$\sum_{i \in I_1(x)} u_i \, f_i(x) + \sum_{i \in I_2(x)} u_i \, a_i^T x - \sum_{j \in J(x)} v_j \, x_j \text{ will equal } \sum_{i \in I(x)} u_i b_i \text{ for all}$$

$x \in R$ ($I(x) = I_1(x) + I_2(x)$). Hence, for no $x \in R$, $f_r(x) < b_r$ could hold, contrary to condition C1.
Corollary: if C1 is satisfied and $x \in R$, $i \in I_1(x)$, then $q_i(x) \neq 0$.

Let $x \in R$. A direction s in x will be called feasible if we do not immediately leave the region R when making a sufficiently small move in the direction s, hence if $\exists_\lambda \lambda > 0$, $x + \lambda s \in R$.
Let $S(x)$ be the convex polyhedral cone of all vectors s satisfying $q_i(x)^T s \leqq 0$ if $i \in I_1(x)$; $a_i^T s \leqq 0$ if $i \in I_2(x)$ and $s_j \geqq 0$ if $j \in J(x)$. A necessary condition for a vector to be feasible in R is that $s \in S(x)$. If $I_1(x)$ is non-empty this condition will in general not be sufficient.

In literature instead of our condition C1 we usually find the condition C2 imposed on R: For all $x \in R$ any vector $s \in S(x)$ is tangent to an arc which is completely contained in the constraint set. (See Kuhn and Tucker [28] and Dennis [17], p. 133).

This condition can easily be shown to be equivalent to the condition that for all $x \in R$ the closure of the cone of feasible directions $S'(x)$ equals $S(x)$. It is slightly weaker than C1, which follows from:
Theorem 3: Condtion C1 implies C2.
Proof: Suppose C1 is satisfied, so that, for some $\bar{x} \in R$, $f_i(\bar{x}) < b_i$ will hold for all $i \in I_1$. Let $x \in R$ be arbitrary and consider the feasible vector $\bar{s} = \bar{x} - x$. We have $q_i(x)^T \bar{s} < 0$ for all $i \in I_1(x)$ (theorem 3, corollary 1, section 2.4). Let $s \in S(x)$. For all $\lambda > 0$ it follows that $q_i(x)^T (s + \lambda \bar{s}) < 0$ holds for all $i \in I_1(x)$, so that $s + \lambda \bar{s} \in S'(x)$ for all $\lambda > 0$. This implies C2.

Hence condition C2 includes a larger class of constraints than C1 does but examples where C2 is satisfied but C1 is not are rather artificial. In such examples $f_i(x) = b_i$ will hold for all $x \in R$ and some i, say $i \in I_1' \subset I_1$. Neither C1 nor C2 seems to be a serious restriction on R. It is perhaps worth mentioning that if C1 is not satisfied, replacement of all constraints $f_i(x) \leqq b_i$ by

$f_i(x) \leqq b_i + \varepsilon$, ε being a small positive number, will lead to a feasible region $R(\varepsilon) \supset R$ for which C1 is satisfied (assuming that R is not empty). The adding of ε can be restricted to those constraints $f_i(x) \leqq b_i$ for which $f_i(x) < b_i$ is impossible for any $x \in R$.

Let F(x) be a concave differentiable function of $x \in E_n$ with continuous gradient vector g(x), let R, defined by (2.6.1), satisfy condition C1, then the convex programming problem is defined by

$$\text{Max } \{F(x) \mid x \in R\}. \tag{2.6.4}$$

The function F(x) will be called the objective function.
A feasible direction s in $x \in R$ will be called usable if:

$$\left(\frac{d\,F(x + \lambda s)}{d\,\lambda}\right)_{\lambda=0} = g(x)^T s > 0 \text{ holds.}$$

Theorem 4: If no feasible direction in $x \in R$ is usable, then x will be a maximum of F(x) on R.
Proof: Suppose $y \in R$ can be found such that $F(y) > F(x)$. Corollary 1 of theorem 3, section 2.4 states that in that case $g(x)^T(y-x) > 0$ holds (note that $-F(x)$ is convex). Since y - x is a feasible direction it follows that it is also a usable direction. Hence, if no usable direction exists, x will be a maximum of F(x) on R.
Theorem 5: A feasible point $x \in R$ will be a maximum of F(x) on R if and only if $g(x)^T s \leqq 0$ for all s satisfying $q_i(x)^T s \leqq 0$ if $i \in I_1(x)$; $a_i^T s \leqq 0$ if $i \in I_2(x)$ and $s_j \geqq 0$ if $j \in J(x)$, i.e. for all $s \in S(x)$.
Proof: The "if" part is a trivial consequence of the definition of a usable direction and theorem 4. To prove the "only if" part we suppose that an $s \in S(x)$ exists, satisfying $g(x)^T s > 0$. From theorem 1 we know that, for some $x' \in R$, $f_i(x') < b_i$ will hold for all $i \in I_1$. Let s' be x' - x, then $q_i(x)^T s' < 0$ for all $i \in I_1(x)$. Let $s(\lambda) = \lambda s + s'$. Clearly $s(\lambda)$ is feasible for all $\lambda \geqq 0$ since $q_i(x)^T s(\lambda) < 0$, $i \in I_1(x)$; $a_i^T s(\lambda) \leqq 0$, $i \in I_2(x)$, and $\{s(\lambda)\}_j \geqq 0$, $j \in J(x)$ will hold for all $\lambda \geqq 0$. Moreover $g(x)^T s$ being > 0, λ can be chosen so large that $g(x)^T s(\lambda) > 0$ will hold. Consequently x cannot be a maximum point.
Remark: If condition C2 is imposed on R instead of C1, the theorem is also correct. A proof can be found in [17], p. 133-135.
Theorem 6: A point $x \in R$ is a maximum point of F(x) on R if and only if the gradient vector in x is a non-negative linear combination of the outward-pointing normals in x, i.e.:

$$x \text{ maximum} \Longleftrightarrow g(x) = \sum_{i \in I_1(x)} u_i\, q_i(x) + \sum_{i \in I_2(x)} u_i\, a_i - \sum_{j \in J(x)} v_j\, e_j; \tag{2.6.5}$$

$$u_i \geqq 0,\ v_j \geqq 0.$$

Proof: If g(x) can be written as in (2.6.5), right-hand member, we have that $g(x)^T s \leqq 0$ for any $s \in S(x)$, so that, by theorem 5, x is a maximum of F(x) on R. On the other hand if x is a maximum of F(x) on R, then theorem 5 shows, that $g(x)^T s \leqq 0$ for any $s \in S(x)$,

so that the right-hand member of (2.6.5) follows from Farkas' theorem (section 2.2, theorem 1).

Instead of (2.6.5) we can also write:

$$x \text{ maximum} \Longleftrightarrow g(x) = \sum_{i \in I_1} u_i\, q_i(x) + \sum_{i \in I_2} u_i\, a_i - v; \qquad (2.6.6)$$

$$u \geqq 0,\ v \geqq 0;\ u^T y = 0,\ v^T x = 0,$$

with y defined by (2.6.2).

Kuhn and Tucker [28] have shown that the maximization of a concave differentiable function F(x) in a convex region R defined by $x \geqq 0$ and $f_i(x) \leqq b_i$, $i \in I$, $f_i(x)$ convex and differentiable, satisfying condition C2, is equivalent to the determination of a saddle-point of the function $\varphi(x,u) = F(x) + u^T y$ under the conditions $u \geqq 0$, $x \geqq 0$ and $y_i = b_i - f_i(x) \geqq 0$, and that in the saddle-point x', u' the following conditions are satisfied:

$$1.\left(\frac{\partial \varphi(x,u)}{\partial x}\right)_{x',u'} \leqq 0, \text{ i.e. } g(x') - \Sigma\, u_i'\ q_i(x') \leqq 0;$$

$$x'^T\left(\frac{\partial \varphi(x,u)}{\partial x}\right)_{x',u'} = 0, \text{ hence if } x_j' > 0, \text{ then } g_j(x') - \Sigma\, u_i' \{q_i(x')\}_j = 0;$$

$$2.\left(\frac{\partial \varphi(x,u)}{\partial u}\right)_{x',u'} \geqq 0, \text{ i.e. } f_i(x') - b_i \leqq 0;$$

$$u'^T\left(\frac{\partial \varphi(x,u)}{\partial u}\right)_{x',u'} = 0, \text{ hence if } f_i(x') < b_i, \text{ then } u_i' = 0.$$

These conditions are called the Kuhn-Tucker optimality conditions. They are the same as the conditions (2.6.6).

Theorem 7: Let x' be a maximum of F(x) on R. Then x' also solves the problems

1. $\text{Max}\{g(x')^T x \mid x \in R\}$;
2. $\text{Max}\{g(x')^T x \mid q_i(x')^T x \leqq q_i(x')^T x',\ i \in I_1(x');$
$a_i^T x \leqq b_i,\ i \in I_2(x') \text{ and } x_j \geqq 0,\ j \in J(x')\}$.

Proof: If x' is a maximum of F(x) on R the conditions (2.6.6) hold. Since x' is a feasible point of problems 1 and 2 and satisfies the optimality conditions for these problems (which are exactly the conditions (2.6.6)) it follows that x' is a maximum.

Note that problem 2 is a linear programming problem.

Theorem 8: For any $\bar{x} \in R$ we have:

$$\text{Max}\{F(x) \mid x \in R\} - F(\bar{x}) \leqq \text{Max}\{g(\bar{x})^T x \mid x \in R\} - g(\bar{x})^T \bar{x} \leqq$$
$$\leqq \text{Max}\{g(\bar{x})^T x \mid q_i(\bar{x})^T x \leqq q_i(\bar{x})^T \bar{x} + b_i - f_i(\bar{x}),\ i \in I_1;$$
$$a_i^T x \leqq b_i,\ i \in I_2;\ x \geqq 0\} - g(\bar{x})^T \bar{x}. \qquad (2.6.7)$$

Proof: $F(x)$ is concave, hence by theorem 3.2, section 2.4 we have ($-F(x)$ is convex):

$$F(x) - F(\bar{x}) \leqq g(\bar{x})^T(x-\bar{x}) \leqq \text{Max}\{g(\bar{x})^T x \mid x \in R\} - g(\bar{x})^T\bar{x}$$

for any $x \in R$ from which the first inequality follows. Let $x \in R$, so that $f_i(x) \leqq b_i$ for all i. Since $f_i(x)$ is convex, we have by theorem 3.2, section 2.4 that

$$f_i(x) \geqq f_i(\bar{x}) + q_i(\bar{x})^T(x-\bar{x}).$$

It follows that

$$q_i(\bar{x})^T x \leqq q_i(\bar{x})^T\bar{x} + b_i - f_i(\bar{x}),$$

so that the convex polyhedron defined by this set of inequalities contains R. From this the second inequality follows.

Theorem 8 can be considered as an approximation theorem. The maximum increase in value of the objective function if we start an iterative procedure in a feasible point $\bar{x}$ can never exceed the maximum increase in value of the linearized problem. The linearizing is obtained by maximizing the linear term in the Taylor expansion of the function $F(x)$ and by replacing the inequality $y_i = b_i - f_i(x) \geqq 0$ by a linear inequality obtained by expanding $b_i - f_i(x)$ in a Taylor series and taking the linear terms. The approximation is consistent since the inequality signs will be equality signs if $\bar{x}$ is already the real maximum.

3. METHODS OF SOLUTION FOR THE LINEAR PROGRAMMING PROBLEM

3.1 *Introduction*

Many methods have been developed for the solution of the linear programming problem. They can be distinguished into finite and infinite iterative methods. The first group of methods is mostly based on the principle of the Gauss elimination of variables. We have in this group the simplex method, the dual simplex method, Beale's method of leading variables [2], the primal-dual algorithm and the methods of feasible directions. The first two methods and the fourth one will be described in this chapter; the last one will be dealt with in chapter 9. The group of infinite methods includes all kinds of relaxation methods. In spite of the small amount of work involved in an iteration they mostly require so many iterations, i.e. converge so badly that it must be doubted whether they can compete with the finite methods for larger problems.

3.2 *The Simplex Method*

The simplex method is the most widely known numerical procedure for solving the linear programming problem. It is a finite iterative method and has been described in many papers and books (e.g. Dantzig, Orden and Wolfe [15], Vajda [36] and Gass [22]).

For explaining the simplex method we shall first assume that the problem is of the form (see section 2.5 for the notation):

$$\mathrm{Max}\{\, p^T x \mid Ax + y = b;\ x \geqq 0,\ y \geqq 0,\ b \geqq 0 \,\}. \tag{3.2.1}$$

The y_i will be called elementary variables.
Defining

$$\bar{A} = (A, E),\quad \bar{x} = \begin{pmatrix} x \\ y \end{pmatrix} \text{and } \bar{p}^T = (p^T,\ 0),$$

we obtain the equivalent problem

$$\mathrm{Max}\{\, \bar{p}^T \bar{x} \mid \bar{A}\bar{x} = b \geqq 0,\ \bar{x} \geqq 0 \,\}. \tag{3.2.2}$$

For the time being we shall also impose the non-degeneracy assumption:
For any non-singular m by m submatrix B of (A, E) we have that, if $B^{-1} b \geqq 0$, then $B^{-1} b > 0$ (no component of $B^{-1} b$ shall be zero).

It can be shown that this condition is equivalent to the geometrical condition that each vertex of the feasible region will be the intersection of exactly n linearly independent bounding hyperplanes but shall not lie in any of the other bounding hyperplanes. Still another formulation is that any feasible solution shall have at least m positive components. Clearly $b > 0$ will hold in (3.2.1) if this condition is satisfied.

Let us suppose that we know a non-singular m by m submatrix $B = (\bar{a}_{.j_1}, \ldots, \bar{a}_{.j_m})$, satisfying $B^{-1} b > 0$ ($\bar{a}_{.j}$ denotes the j-th column of the matrix $\bar{A}$, j here runs from 1 to n + m, hence $\bar{a}_{.n+r} = e_r$, the r-th unit vector). Writing $\bar{A} = (B, D)$, $\bar{p} = \begin{pmatrix} \bar{p}_B \\ \bar{p}_D \end{pmatrix}$ and $\bar{x} = \begin{pmatrix} \bar{x}_B \\ \bar{x}_D \end{pmatrix}$, we have $b = B\bar{x}_B + D\bar{x}_D$, hence $\bar{x}_B = B^{-1} b - B^{-1} D \bar{x}_D$.
It follows that $\bar{x}_B = B^{-1} b > 0$ and $\bar{x}_D = 0$ is a basic feasible solution with value $\bar{p}_B{}^T B^{-1} b$.
The matrix B will be called the basis; the variables $\bar{x}_{Bi} = \bar{x}_{j_i}$, $i = 1, \ldots, m$ will be said to be in the current basis, the variables $\bar{x}_{Dj}$ are in the current non-basis. Substituting for $\bar{x}_B$ in the objective function we obtain:

$$p^T x = \bar{p}_B{}^T \bar{x}_B + \bar{p}_D{}^T \bar{x}_D = \bar{p}_B{}^T B^{-1} b - (\bar{p}_B{}^T B^{-1} D - \bar{p}_D{}^T)\bar{x}_D.$$

Let us now define a vector $\bar{d}$ by means of its components:

$$\bar{d}_j = \bar{p}_B{}^T B^{-1} \bar{a}_{.j} - \bar{p}_j, \quad j = 1, \ldots, n+m, \tag{3.2.3}$$

so that $\bar{d}_j = 0$ if $\bar{x}_j$ belongs to the $\bar{x}_B$ part of $\bar{x}$, i.e. if $\bar{x}_j$ is in the current basis. Moreover writing $\bar{d}^T = (d_1{}^T, d_2{}^T)$ where d_1 is an n-component vector corresponding with the variables x_j and d_2 is an m-component vector corresponding with the variables y_i, we have

$$d_2{}^T = \bar{p}_B{}^T B^{-1}; \; d_1{}^T = \bar{p}_B{}^T B^{-1} A - p^T. \tag{3.2.4}$$

The $\bar{d}_j$ variables are often called shadow prices. The row of shadow prices will be called d row.

From the foregoing it follows that

$$\bar{p}^T \bar{x} = \bar{p}_B{}^T B^{-1} b - \bar{d}^T \bar{x}.$$

We now have two possibilities:

1. $\bar{d}_j \geqq 0$ for all j. It then follows that $\bar{x} = \begin{pmatrix} \bar{x}_B \\ \bar{x}_D \end{pmatrix} = \begin{pmatrix} B^{-1} b \\ 0 \end{pmatrix}$ is an optimum solution since $u^T = d_2{}^T$ and $v^T = d_1{}^T$ satisfy the dual constraints of (3.2.1) whereas $u^T y = 0$ and $v^T x = 0$, so that the optimality follows from theorem 5, section 2.5.
2. $\exists_s \; \bar{d}_s < 0$. We can now try to increase $\bar{x}_s$ from zero until one of the non-zero (basic) variables $\bar{x}_{Bi}$ becomes zero, i.e. we shall have to determine a scalar $\lambda_r > 0$ such that

$$\lambda_r = \max \{\lambda \mid B^{-1} b - \lambda B^{-1} \bar{a}_{.s} \geqq 0\}. \tag{3.2.5}$$

If $\lambda_r = +\infty$, hence if $B^{-1}\bar{a}_{.s} \leqq 0$, the problem will have an infinite solution, since $\bar{x}_s = \lambda$, $\bar{x}_B = B^{-1}b - \lambda B^{-1}\bar{a}_{.s}$, and $\bar{x}_{Dj} = 0$ if $\bar{x}_{Dj} \neq \bar{x}_s$, will be feasible for any $\lambda > 0$, whereas, if λ goes to infinity, $\bar{p}^T\bar{x} = \bar{p}_B^T B^{-1} b - \lambda \bar{d}_s$ will grow beyond all bounds.
If $\lambda_r < \infty$, the choice of r is unique since otherwise $\bar{x}_s = \lambda_r$, $\bar{x}_B = B^{-1} b - \lambda_r B^{-1} \bar{a}_{.s}$, and $\bar{x}_{Dj} = 0$ if $\bar{x}_{Dj} \neq \bar{x}_s$, would be a feasible solution with less than m positive components, contrary to our non-degeneracy assumption. We can now solve for $\bar{x}_s$ in the r-th equation (obviously $(B^{-1}\bar{a}_{.s})_r > 0$) and eliminate $\bar{x}_s$ from all other equations and the objective vector, so that we obtain a new set of basic variables and a new basis B*.
Introducing the "η-vector" η_r by

$$(\eta_r)_i = -\frac{(B^{-1}\bar{a}_{.s})_i}{(B^{-1}\bar{a}_{.s})_r} \text{ if } i \neq r, \tag{3.2.6}$$

$$(\eta_r)_r = \frac{1}{(B^{-1}\bar{a}_{.s})_r}, \tag{3.2.7}$$

and the elementary matrix E_r by

$$E_r = (e_1, \ldots, e_{r-1}, \eta_r, e_{r+1}, \ldots, e_m), \tag{3.2.8}$$

we have:

$$(B^*)^{-1} = E_r B^{-1}, \tag{3.2.9}$$

and for any column of the matrix $\bar{A}$:

$$(B^*)^{-1}\bar{a}_{.j} = E_r(B^{-1}\bar{a}_{.j}). \tag{3.2.10}$$

Suppose that we had a tableau with columns $B^{-1}\bar{a}_{.j}$, then we get a new tableau, corresponding to the new basis, by premultiplying every column of this old tableau by the elementary matrix E_r. We shall then have that the unit column $B^{-1}\bar{a}_{.j_r}$ transforms to the column η_r and that the column $B^{-1}\bar{a}_{.s}$ transforms to a unit column. The premultiplication by E_r leads to the well-known transformation formulae which are the same as those used in the Jordan method for matrix inversion.

The column to be introduced in the basis in a certain iteration will be called the main column or pivot column. The index r given by (3.2.5) determines the main row or pivot row. The common element of pivot column and pivot row will be called the pivot element, i.e. $(B^{-1}\bar{a}_{.s})_r$.

The increase in value of the objective function corresponding with this change of basis equals $-\lambda_r \bar{d}_s > 0$. Hence, by proceeding in this way we obtain a sequence of bases and corresponding basic solutions with ever-increasing value for the objective function. Since the number of possible bases is finite and no basis can ever repeat, the procedure is obviously finite under the non-degeneracy assumption.

From the foregoing it follows that the problem will be solved as soon as a square submatrix B' of $\bar{A}$ has been found with the properties:

$$1.\ \{B'\}^{-1} b \geqq 0 \tag{3.2.11}$$

(hence > 0 under the non-degeneracy assumption),

$$2.\ \bar{d}_j' = \bar{p}_{B'}^T \{B'\}^{-1} \bar{a}_{.j} - \bar{p}_j \geqq 0 \text{ for all } j = 1, \ldots, n+m. \tag{3.2.12}$$

The optimum solution will then be:

$$\bar{x}_{B'} = \{B'\}^{-1} b,\ \bar{x}_{D'} = 0. \tag{3.2.13}$$

The solution of the dual problem will be:

$$u'^T = \bar{p}_{B'}^T \{B'\}^{-1} = d_2'^T,\ v'^T = \bar{p}_{B'}^T \{B'\}^{-1} A - p^T = d_1'^T, \tag{3.2.14}$$

as can be verified by direct substitution in (2.5.7).
The solution is obtained at the same time by means of the final d row.

Summarizing we can say that the ν-th iteration in the simplex method proceeds along the following lines:

1. The pivot column, if any, is determined by $\bar{d}_{s_\nu}^{\nu-1} < 0$ (usually the most negative is chosen);

$$(\bar{d}^{\nu-1})^T = \begin{pmatrix} d_1^{\nu-1} \\ d_2^{\nu-1} \end{pmatrix}^T = (\bar{p}_{B\nu-1}^T \{B^{\nu-1}\}^{-1} A - p^T,\ \bar{p}_{B\nu-1}^T \{B^{\nu-1}\}^{-1}),$$

where $B^{\nu-1}$ stands for the m by m matrix derived from the matrix (A, E) by taking only those columns the variables of which are in the basis at iteration $\nu-1$.

2. In the chosen pivot column the row number r_ν of the pivot element is determined by

$$\frac{b_{r_\nu}^{\nu-1}}{\bar{a}_{r_\nu s_\nu}^{\nu-1}} = \min_i \left\{ \frac{b_i^{\nu-1}}{\bar{a}_{is_\nu}^{\nu-1}} \;\middle|\; \bar{a}_{is_\nu}^{\nu-1} > 0 \right\}, \tag{3.2.15}$$

where $b^{\nu-1} = \{B^{\nu-1}\}^{-1} b$ and $\bar{a}_{.s_\nu}^{\nu-1} = \{B^{\nu-1}\}^{-1} \bar{a}_{.s_\nu}$.

3. The transformation itself is carried out by making the new "η-vector":

$$\{\eta_{r_\nu}^{\nu}\}_i = -\frac{\bar{a}_{is_\nu}^{\nu-1}}{\bar{a}_{r_\nu s_\nu}^{\nu-1}} \qquad \text{if } i \neq r_\nu, \tag{3.2.16}$$

$$\{\eta_{r_\nu}^{\nu}\}_{r_\nu} = \frac{1}{\bar{a}_{r_\nu s_\nu}^{\nu-1}}; \tag{3.2.17}$$

and performing the transformation:

$$b^{\nu} = E_{r_\nu}{}^{\nu}\, b^{\nu-1}, \tag{3.2.18}$$

$$\bar{a}_{.j}{}^{\nu} = E_{r_\nu}{}^{\nu}\, \bar{a}_{.j}{}^{\nu-1}, \quad j=1,\ldots,n+m. \tag{3.2.19}$$

It is not necessary to do all these calculations in an iteration. In chapters 4 and 5 these computational aspects will be investigated.

Up to now we have assumed that the problem is of the form (3.2.1). This is not always the case. We shall distinguish the following cases:

1. Some of the y_i variables have coefficients in the objective function, so that $\bar{p}^T = (p^T, q^T)$. In that case we must calculate the initial d row by

$$d_1{}^0 = A^T q - p, \quad d_2{}^0 = 0,$$

and use $-\bar{d}^0$ as objective function (hence define a new $\bar{p} = -\bar{d}^0$). The solution of the dual problem in this case will be

$$u' = d_2^1 + q, \quad v' = d_1^1, \tag{3.2.20}$$

as can easily be verified (see 2.5, theorem 7).

2. $b \geqq 0$ but not all the y_i exist (hence $\bar{A}$ does not contain a complete set of unit vectors). In this case we could for instance add artificial unit vectors and artificial variables z_i for those values of i for which $\bar{A}$ does not contain the unit vector e_i, say for $i \in I_z \subset I$.

We now first solve the artificial problem (phase I):

$$\text{Max}\Big\{ - \sum_{i \in I_2} z_i \;\Big|\; a_{i.}{}^T x + y_i = b_i \text{ if } i \in I - I_z;\ a_{i.}{}^T x + z_i = b_i \text{ if } i \in I_z;\ x \geqq 0,\ y_i \geqq 0,\ z_i \geqq 0,\ b \geqq 0 \Big\}, \tag{3.2.21}$$

where the row vectors $a_{i.}{}^T$ are the rows of the matrix A.

This is again a linear programming problem but now of the type (3.2.1). It will be solved in a finite number of steps. In the optimum solution we shall have:

either $\Sigma z_i > 0$ in which case we conclude that the original system of equations is inconsistent, so that the problem is not feasible,

or $\Sigma z_i = 0$ in which case we have obtained a basic feasible solution with, owing to the non-degeneracy assumption, none of the z_i variables in the basis, so that we can completely forget these artificial variables and proceed to the second phase, the solution of the actual problem for which we now know an initial basis B.

3. $b_i < 0$ for some i. If such a row does not contain a y_i variable we can multiply it by -1 after which the artificial z_i is introduced. If it contains a y_i variable the multiplication by -1 will make that y_i can no longer be used as initial basic variable, so

that here too an artificial z_i must be introduced after the multiplication. In both cases we shall then first have to solve a problem of type (3.2.21).

4. Some of the variables x_j are not restricted by a non-negativity requirement. We can always write $x_j = x_j^+ - x_j^-$, $x_j^+ \geqq 0$, $x_j^- \geqq 0$ for such a free variable. If $d(x_j^+) \leqq 0$ we introduce x_j^+ in the basis, otherwise x_j^- and as soon as this has been done the row to which it then belongs in the tableau can be disregarded in choosing a pivot row.

It should be well understood that the procedures described are not always the most efficient ones from a computational point of view.

The proof of finiteness of the simplex method breaks down in the likely case of degeneracy. Several proposals have been made which cope with this difficulty. Degeneracy may lead to a situation where the choice of λ_r is no longer unique. Intuitively it is clear that a perturbation of the right-hand member b, if introduced correctly, will lead to a unique choice. On this idea the first of the following two anti-degeneracy proposals is based:

Anti-degeneracy proposal 1 (see Charnes, [7]):

a. Find $I_0 = \left\{ r \in I \,\middle|\, \lambda_r^* = \frac{b_r^*}{\bar{a}_{rs}^*} = \min_i \frac{b_i^*}{\bar{a}_{is}^*},\ \bar{a}_{is}^* > 0 \right\}$, (the asterisk denotes the transformed figures, hence $b_i^* = (B^{-1}b)_i$ if B^{-1} is the inverse of the current basis).

If I_0 is empty, there will be an infinite solution; if it consists of one element the choice of λ_r^* will be unique.

b. Suppose sets $I_0, I_1, I_2, \ldots, I_{k-1}$, have already been determined and I_{k-1} still consists of more than one element. We then define:

$$I_k = \left\{ r \in I_{k-1} \,\middle|\, \frac{\bar{a}_{rk}^*}{\bar{a}_{rs}^*} = \min_i \frac{\bar{a}_{ik}^*}{\bar{a}_{is}^*},\ i \in I_{k-1} \right\}.$$

Since no two rows of the matrix are dependent (the matrix contains the unit matrix) we shall arrive at a unique r after a finite number of steps of this kind which is less than the number of columns in the (perhaps artificially extended) matrix $\bar{A}$.

Anti-degeneracy proposal 2 (see Dantzig, Orden and Wolfe, [15]):

a. Find $I_0 \subset I$ as in the first proposal.

b. Suppose sets $I_0, I_1, \ldots, I_{k-1}$ have already been chosen.

We now define:

$$I_k = \left\{ r \in I_{k-1} \,\middle|\, \frac{(B^{-1}e_k)_r}{\bar{a}_{rs}^*} = \min_i \frac{(B^{-1}e_k)_i}{\bar{a}_{is}^*},\ i \in I_{k-1} \right\}.$$

Since no two rows of the inverse are dependent this procedure will lead to a unique choice of r after a number of steps which is $\leqq m$.

It can be proved that both methods for breaking ties will guarantee an optimum solution of a linear programming problem after a finite number of iterations. Hence (3.2.21) will be solved and phase I finished after a finite number of iterations. But, in the case of degeneracy, it will be possible that some of the z_i are still in the basis (with zero values, otherwise the problem would not be feasible). We can still proceed to phase II provided we add

an extra equation $\Sigma\, z_i \leqq 0$ to the problem, the summation being over those values of $i \in I_z$ for which z_i is still in the basis at the end of phase I. This extra equation will guarantee that none of the z_i, still in the basis, will ever become positive.

3.3 *The Dual Simplex Method*

Let a linear programming problem be of the form (2.5.1) with $p_j \leqq 0$ for all j and $b_i < 0$ for some i. From (2.5.7) it then follows that $u = 0$, $v = -p$ is an initial basic feasible solution for the dual problem, so that this dual problem can be solved immediately without using some artifice. When using the simplex method for the solution of this dual problem we shall also obtain the solution of the primal problem (see theorem 2, section 2.5 and (3.2.14)).

Lemke [29] remarked that instead of dualizing first and applying then the simplex method to the dual problem, we can also work within the organization scheme of the primal problem and there use a set of criteria which would lead to the choice of those columns and rows as main columns and main rows that would have been chosen by applying the simplex method to the dual problem.

We then obtain the dual simplex method which proceeds along the following lines:

1. In iteration ν the pivot row r_ν, if any, is determined by $b_{r_\nu}^{\nu-1} < 0$ (usually the most negative is chosen); $b^{\nu-1} = \{B^{\nu-1}\}^{-1} b$, where $B^{\nu-1}$ stands for the m by m matrix derived from the matrix (A,E) by taking only those columns the variables of which are in the (primal) basis at iteration $\nu-1$.

2. In the chosen pivot row the column number s_ν of the pivot element is determined by:

$$\frac{\bar{d}_{s_\nu}^{\nu-1}}{-\bar{a}_{r_\nu s_\nu}^{\nu-1}} = \min_j \left\{ \frac{\bar{d}_j^{\nu-1}}{-\bar{a}_{r_\nu j}^{\nu-1}} \;\middle|\; \bar{a}_{r_\nu j}^{\nu-1} < 0 \right\}, \qquad (3.3.1)$$

provided $\bar{a}_{r_\nu j}^{\nu-1} < 0$ holds for some j. Otherwise the problem is not feasible.

3. The transformation formulae are the same as in the simplex method.

4. The problem will be solved at iteration k if $b_i^k \geqq 0$ for all i (the condition (3.3.1) automatically entails that $\bar{d}_j^\nu \geqq 0$ will hold for all j and ν). The solution will always be reached in a finite number of iterations if an anti-degeneracy precaution is applied.

A geometrical interpretation of the dual simplex method within the geometry of the primal problem can easily be given. For, any non-optimal dual basic feasible point (i.e. any basic feasible point of the dual constraint set which is not the optimum) corresponds with a non-feasible vertex in the primal problem (i.e. intersection of n linearly independent constraining hyperplanes) which is optimum with regard to a restricted primal problem obtained from the primal problem by omitting those constraints which are not satisfied. If such a point is called a non-feasible

basic optimal point, then the dual simplex method will go through a vertex-to-vertex path of non-feasible basic optimal points until a feasible basic optimal point is reached.

It should be remarked that many vertices are neither feasible nor dual feasible. If the initial vertex is such a point we shall have to use an artifice in both methods. In the simplex method we can introduce artificial variables for all constraining equations and for the constraining inequalities with negative right-hand member figure and use the phase I approach of section 3.2, formula (3.2.21); in the dual simplex method we can introduce artificial variables for the constraining equations and an extra constraint $\Sigma\, x_j \leqq M$, where the summation is over all values of j for which $p_j > 0$, say $j \in J_+$, and M is so large a number that the artificial constraint will not determine the optimal solution. We then pivot in this row and in the column s determined by $p_s = \max p_j$, so that $b_i' = b_i - Ma_{is}$, $i = 1, \ldots, m$, $b_{m+1}' = M$, will be the new right-hand member, whereas $p_j' = p_j - p_s$ if $j \in J_+$ and $= p_j$ if $j \notin J_+$ so that $p_j' \leqq 0$ will now hold for all j. Instead of fixing M we can also use a two-phase approach by taking first $-a_{.s}$ as criterion column for the selection of main rows until the elements of this column have all become non-negative, after which we can proceed to the real criterion column, i.e. the b column.

The dual simplex method has the advantage that artificial variables, due to equations, can be eliminated from the basis one after the other in successive iterations. Its disadvantage is that the solution will remain non-feasible until the end of the calculation, so that the problem has to be completely solved before the results will have a practical meaning. It has found its major application in a number of post-optimal problems (e.g. amendments problems, parametric programming, see Gass and Saaty, [23] and Gass, [22]).

3.4 *The Primal-Dual Method*

For a description of this method we suppose that the linear programming problem is of the form

$$\text{Max}\left\{ p^T x \;\middle|\; Ax = b,\ x \geqq 0,\ b \geqq 0 \right\}, \tag{3.4.1}$$

so that its dual is given by (theorem 6, section 2.5)

$$\text{Min}\left\{ b^T u \;\middle|\; A^T u - v = p,\ v \geqq 0 \right\}. \tag{3.4.2}$$

The assumption $b \geqq 0$ is no restriction since this can always be obtained by multiplying some rows by -1.

We shall show how we can obtain a feasible solution $u^{\nu+1}$ of (3.4.2) from another feasible solution u^ν in such a way that $b^T u^{\nu+1} < b^T u^\nu$. The procedure must thus be started with an initial dual feasible solution u^0 (u^0 may be arbitrary and may or may not be a basic solution).

Let $J_\nu \subset J$ be defined by

$$J_\nu = \{ j \in J \mid v_j^\nu = 0 \}. \tag{3.4.3}$$

We now solve the restricted primal problem:

$$\text{Max} \{ - \sum_{i=1}^{m} z_i \mid \sum_{j \in J_\nu} a_{ij} x_j + z_i = b_i,\ x_j \geqq 0,\ z_i \geqq 0 \}, \tag{3.4.4}$$

so that the optimum solution z_i', x_j' is obtained as well as the final d figures $d'(z_i)$ and $d'(x_j)$.
There are two possibilities:

$$1. \quad - \sum_{i=1}^{m} z_i' = 0.$$

The vector x' will then satisfy: $Ax' = b$, $x' \geqq 0$ and $x'^T v^\nu = 0$, so that, by theorems 5 and 6 of section 2.5, x' will solve (3.4.1).

$$2. \quad - \sum_{i=1}^{m} z_i' < 0.$$

By (3.2.20) the solution of the problem dual to (3.4.4) will be $u_i' = d'(z_i) - 1$, $v_j' = d'(x_j)$ and, by theorem 7 of 2.5, we have

$$- \sum_{i=1}^{m} z_i' = \sum_{i=1}^{m} b_i (d'(z_i) - 1) < 0. \tag{3.4.5}$$

Moreover, if $j \in J_\nu$, hence $v_j = 0$:

$$0 \leqq d'(x_j) = v_j' = \sum_{i=1}^{m} u_i' a_{ij} = \sum_{i=1}^{m} \{ d'(z_i) - 1 \} a_{ij}. \tag{3.4.6}$$

Now let

$$u_i = u_i^\nu + \lambda \{ d'(z_i) - 1 \}, \tag{3.4.7}$$

and

$$\lambda_\nu = \text{Max} \{ \lambda \mid \sum_{i=1}^{m} a_{ij} u_i \geqq p_j \}, \tag{3.4.8}$$

then $\lambda_\nu > 0$ by (3.4.6). If $\lambda_\nu = \infty$, (3.4.2) will have an infinite solution since (3.4.5) holds, so that (3.4.1) will not be feasible. Let

$$u_i^{\nu+1} = u_i^{\nu} + \lambda_\nu \{ d'(z_i) - 1 \}, \tag{3.4.9}$$

then $u^{\nu+1}$ will be a feasible solution of (3.4.2) and $b^T u^{\nu+1} < b^T u^{\nu}$ will hold. Moreover it follows from (3.4.8) and $\lambda_\nu < \infty$ that, for at least one j, $v_j^{\nu+1} = 0$ and $v_j^{\nu} > 0$.

We now have a new set $J_{\nu+1}$, so that we can make a new restricted primal problem of type (3.4.4). Under the non-degeneracy assumption we shall necessarily have $\{ \sum_{i=1}^{m} z_i' \}^{\nu+1} < \{ \sum_{i=1}^{m} z_i' \}^{\nu}$

Since any basis in a restricted problem (3.4.4) is an m by m submatrix of the matrix (A, E) and a certain value Σz_i corresponds to each basis, it follows that no basis can ever repeat. Hence after a finite number of steps we shall obtain $\Sigma z_i = 0$ as optimum solution of (3.4.4).

The primal-dual method (Dantzig, Ford and Fulkerson, [13]) has been applied to linear programming problems of a very special so-called network structure (see Ford and Fulkerson, [19]; Jewell, [27]), where it was possible to solve the restricted problems by means of a special technique, the maximum-flow technique. In problems of this type it is always possible to find an initial feasible solution in a simple way. In the general problem this is mostly impossible so that we then have to use an artifice. This will be further described in section 9.4 where we shall show that the primal-dual method is a version of the more general method of feasible directions.

4. COMPUTATIONAL ALGORITHMS FOR THE SIMPLEX METHOD AND DUAL SIMPLEX METHOD

4.1 *Introduction*

In literature at least three different computational algorithms have been described for the simplex method. They will be called straightforward algorithm, explicit inverse algorithm and product-from algorithm. To these three we shall add a fourth, to be called revised product-form algorithm. All algorithms will be described briefly in this chapter. A numerical comparison between them will be made in chapter 5. In all algorithms the choice of main columns and main rows will be the same, so that there will not be a difference in the total number of iterations. Main columns are usually determined by the most negative d_j figure, if any. In a chosen pivot column the pivot element is determined by using (3.2.15). In case of ties there will be a unique device for breaking the tie.

We shall assume that the problem is of the form (3.2.1):

$$\text{Max } \{ p^T x \mid Ax + y = b \geqq 0,\ x \geqq 0,\ y \geqq 0 \}.$$

Artifices to be used if the problem is not of this form can be the same in either algorithm. We shall write $\overline{A} = (A, E)$, the columns of which will be denoted by $\overline{a}_{.j}$. Instead of b we also write $\overline{a}_{.0}$ and instead of $\overline{d}^\nu$ also $\overline{a}_{0.}{}^\nu$. Hence $\overline{a}_{0j} = -p_j$. Extending the matrix $\overline{A}$ with the column $\overline{a}_{.0}$ and the row $\overline{a}_{0.}{}^T$ we shall obtain the m+1 by m+n+1 matrix ${}_0\overline{A}$. In the same way we shall write ${}_0B$ for an m+1 by m+1 matrix, obtained by taking an m by m submatrix B from $\overline{A}$ and by adding the $\overline{a}_{0.}$ figures to it as an extra row and e_0 as an extra column (e_0 is a unit vector with the 1 at the place corresponding with this extra row). The rows of $\{{}_0B^\nu\}^{-1}$ will be denoted by $\beta_i{}^\nu$, $i = 0, 1, \ldots, m$. Note that $\overline{a}_{.0}{}^\nu$ will never be in a basis ${}_0B^\nu$ and that $\overline{a}_{00}{}^\nu$ is the value of the objective function at iteration ν (initially 0). The η-vectors will also be extended, $\{\eta_{r_\nu}{}^\nu\}_0 = \dfrac{-\overline{d}_{s_\nu}{}^{\nu-1}}{\overline{a}_{r_\nu s_\nu}{}^{\nu-1}}$, and so will the elementary matrices ${}_0E_{r_\nu}{}^\nu$.

We shall write $J = \{1, \ldots, n\}$, $I = \{1, \ldots, m\}$. If $j \in J^\nu$, x_j will be in the basis at iteration ν. So will y_i if $i \in I^\nu$.

In the sections 4.2, 4.3 and 4.4 the straightforward algorithm, the explicit inverse algorithm and the product-form algorithm are briefly described and the computational aspects of these algorithms are discussed. It is assumed that a high-speed electronic computer is available and that magnetic tapes are used for permanent storage of the data.

In section 4.5 the new revised product-form algorithm will be described. The computational versions of the dual simplex method will be considered in section 4.6.

4.2 *The Straightforward Algorithm*

In this algorithm the complete matrix ${}_0\bar{A}$ is transformed in each iteration:

$$ {}_0\bar{A}^{\nu} = {}_0E^{\nu}_{r_\nu} \quad {}_0\bar{A}^{\nu-1}. \tag{4.2.1} $$

Hence the operations are:
1. Find pivot element in pivot column by means of (3.2.15),
2. Make new η-vector (using (3.2.16) and (3.2.17)),
3. Transform all columns using (4.2.1) and select next main column.

Note that m of the m+n columns of $\bar{A}$ are always unit vectors and that m-1 of them are not changed in an iteration. We can therefore write:

$$ \bar{A}^{\nu} = \{A^{\nu}, E\}, \tag{4.2.2} $$

where A^{ν} is the non-basis matrix at iteration ν.

The actual transformation formulae are:

$$ \bar{a}_{ij}{}^{\nu} = \bar{a}_{ij}{}^{\nu-1} + \bar{a}_{r_\nu j}{}^{\nu-1} (\eta_{r_\nu}{}^{\nu})_i, \qquad i \neq r_\nu, \tag{4.2.3} $$

$$ \bar{a}_{r_\nu j}{}^{\nu} = \bar{a}_{r_\nu j}{}^{\nu-1} (\eta_{r_\nu}{}^{\nu})_{r_\nu}. \tag{4.2.4} $$

Hence a column will not be transformed if its main row element $\bar{a}_{r_\nu j}{}^{\nu-1} = 0$. If, in a computer, the columns of the matrix are stored in compressed form on magnetic tape, i.e. if only the non-zero elements and their row numbers are stored, we shall have to read the matrix ${}_0\bar{A}^{\nu-1}$ in iteration ν and to write the matrix ${}_0\bar{A}^{\nu}$. The transformation of each column will then entail:
a. a search for the main row element in that column,
b. a relocation of the column if this main row element is zero,
c. a transformation and relocation if it is non-zero.

During the transformation of the columns the best d_j^{ν} and the corresponding main column for the next iteration, if any, will be chosen by successive replacements. The b^{ν} column can best be stored at the end of the matrix since it will then be immediately available for operation 1.

4.3 *The Explicit Inverse Algorithm*

In this algorithm only the extended inverse of the basis and the b column will be transformed in each iteration. As a consequence the main column $\bar{a}_{.s_\nu}{}^{\nu-1}$ is not immediately available but has to

be calculated by a separate subroutine from the column $\vec{a}_{.s_\nu}$ of the original matrix. The term "update main column" will be used for this operation. Also the d_j^ν figures have to be recalculated in each iteration. We can therefore distinguish the following operations:

1. Update main column:

$$a_{.s_\nu}^{\nu-1} = \{B^{\nu-1}\}^{-1} a_{.s_\nu} \quad \text{if } 1 \leqq s_\nu \leqq n, \tag{4.3.1}$$

$$= \{B^{\nu-1}\}^{-1} e_\ell \quad \text{if } s_\nu = n + \ell. \tag{4.3.2}$$

2. Find pivot element in main column by means of (3.2.15).
3. Make new η-vector (using (3.2.16) and (3.2.17)).
4. Transform right-hand member:

$${}_0b^\nu = {}_0E_{r_\nu}{}^\nu \cdot {}_0b^{\nu-1}. \tag{4.3.3}$$

5. Transform the extended inverse:

$$\{{}_0B^\nu\}^{-1} = {}_0E_{r_\nu}{}^\nu \{{}_0B^{\nu-1}\}^{-1}; \quad {}_0B^0 = E. \tag{4.3.4}$$

6. Perform pricing operation, i.e. calculate the d_j figures and select next main column from original matrix, if any, by means of:

$$d_{1j}^\nu = \beta_0{}^\nu \, {}_0a_{.j}, \quad j \notin J^\nu, \tag{4.3.5}$$

$$d_{2i}^\nu = (\beta_0{}^\nu)_i, \quad i \notin I^\nu. \tag{4.3.6}$$

(These formulae follow immediately from (3.2.3), (3.2.4) and the definitions of β_0 and ${}_0a_{.j}$).

Operations 1, 5 and 6 are called the major operations. The pricing operation need not be done for variables in the current basis since $d_j^\nu = 0$ if $j \in J^\nu$. The inverse $\{B^\nu\}^{-1}$ will have as many unit columns as there are elementary variables left in the basis. If an elementary variable wants to re-enter the basis operation 1 shows that we simply have to take a column of the inverse which is then the updated main column. If many elementary variables stay in the basis, the inverse will contain many unit vectors, so that the operations 1 and 5 will be relatively cheap. The pricing operation itself will entail fewer calculations per column if there are more elementary variables left in the basis since $\beta_0{}^\nu$ will have a zero at places corresponding with these variables. The number of columns to which $\beta_0{}^\nu$ has to be applied will be larger, however.

For operation 1 we have to read the inverse, in operation 5 we have to read and write it and in operation 6 we have to read the original matrix A, so that it seems that the inverse has to be read twice. By row-wise storing of the inverse it is possible, however, to read it once provided we slightly rearrange the operations. We then obtain the following scheme:

1. Suppose $\eta_{r_\nu}{}^\nu$ and b^ν are available; the matrix $\{B^{\nu-1}\}^{-1}$ is stored row-wise on tape. Suppose further that the main row of the inverse, $\beta_{r_\nu}{}^{\nu-1}$, is available.

2. Read and transform $\beta_0{}^{\nu-1}$:

$$\beta_0{}^\nu = \beta_0{}^{\nu-1} + \beta_{r_\nu}{}^{\nu-1} \cdot (\eta_{r_\nu}{}^\nu)_0. \tag{4.3.7}$$

3. Read original matrix and perform pricing operation. Find $s_{\nu+1}$ and $\bar{d}_{s_{\nu+1}}{}^\nu$, if $\bar{d}_j{}^\nu < 0$ for some j, and $a_{\cdot s_{\nu+1}}$ if a non-elementary variable will enter the basis. Write $\beta_0{}^\nu$

4. Read and transform the rows $\beta_i{}^{\nu-1}$, $i > 0$:

$$\beta_i{}^\nu = \beta_i{}^{\nu-1} + \beta_{r_\nu}{}^{\nu-1} (\eta_{r_\nu}{}^\nu)_i, \quad i \neq r_\nu, \tag{4.3.8}$$

$$\beta_{r_\nu}{}^\nu = \beta_{r_\nu}{}^{\nu-1} (\eta_{r_\nu}{}^\nu)_{r_\nu}. \tag{4.3.9}$$

5. Calculate $\bar{a}_{is_{\nu+1}}{}^\nu = \beta_i{}^\nu \cdot a_{\cdot s_{\nu+1}}$ if $1 \leqq s_{\nu+1} \leqq n$, (4.3.10)

$$= (\beta_i{}^\nu)_\ell \text{ if } s_{\nu+1} = n + \ell. \tag{4.3.11}$$

6. If $\bar{a}_{is_{\nu+1}}{}^\nu > 0$, calculate $\lambda_i{}^{\nu+1} = \dfrac{b_i{}^\nu}{\bar{a}_{is_{\nu+1}}{}^\nu}$; select at the same time $\lambda_{r_{\nu+1}}{}^{\nu+1} = \min_i \lambda_i{}^{\nu+1}$ and $\beta_{r_{\nu+1}}{}^\nu$; write $\beta_i{}^\nu$.

Repeat 4, 5 and 6 until all rows of the inverse have been transformed.

7. Make new η-vector $\eta_{r_{\nu+1}}{}^{\nu+1}$.

8. Transform b^ν. We are now again in the situation 1, so that the iteration has been completed.

Note in operation 4 that, if $(\eta_{r_\nu}{}^\nu)_i = 0$, there will not be any transformation. There will also be no search operation. If compressed row storage is used there will only be a search operation in operation 5 if (4.3.11) has to be used (hence if an elementary variable will re-enter the basis).

In a remark at the end of section 4.5 it will be shown how operation 6 (the pricing operation) of the explicit inverse algorithm can be improved.

The explicit inverse algorithm has first been devised by Charnes and is usually called revised simplex method. It is described in [11] , and [22] , chapter 6.

4.4 *The Product-Form Algorithm*

In this algorithm the inverse $\{{}_0B^\nu\}^{-1}$ is not used explicitly but as a product of elementary matrices ${}_0E_{r_\rho}{}^\rho$, $\rho = 1, \ldots, \nu$, so that we only have to store the η-vectors. For, from (4.3.4) it follows that

$$\{{}_0B^\nu\}^{-1} = {}_0E_{r_\nu}{}^\nu \cdot {}_0E_{r_{\nu-1}}{}^{\nu-1} \cdot \ldots \cdot {}_0E_{r_1}{}^1. \tag{4.4.1}$$

It is clear that the vector $\beta_0{}^{\nu}$ is not immediately available now but has to be calculated. On the other hand the operation "transform the inverse" of the explicit inverse algorithm will not exist. The following operations can be distinguished in this algorithm:

1. Update main column:

$$\bar{a}_{.s_\nu}{}^{\nu-1} = E_{r_{\nu-1}}{}^{\nu-1} \cdot E_{r_{\nu-2}}{}^{\nu-2} \cdot \ldots \cdot E_{r_1}{}^{1}\, \bar{a}_{.s_\nu}. \tag{4.4.2}$$

2. Find pivot element in main column by means of (3.2.15).
3. Make new η-vector (using (3.2.16) and (3.2.17)).
4. Transform right-hand member (using (4.3.3)).
5. Calculate $\beta_0{}^{\nu}$:

$$\beta_0{}^{\nu} = e_0{}^{T}\, {}_0E_{r_\nu}{}^{\nu} \cdot {}_0E_{r_{\nu-1}}{}^{\nu-1} \cdot \ldots \cdot {}_0E_{r_1}{}^{1}. \tag{4.4.3}$$

6. Perform pricing operation and select next main column, if any, using (4.3.5) and (4.3.6).

Operations 1, 5 and 6 are the major operations. In operation 1 we are gradually calculating the columns $\bar{a}_{.s_\nu}{}^{\rho}$ $(0 \leqq \rho \leqq \nu-1)$:

$$\bar{a}_{.s_\nu}{}^{\rho} = E_{r_\rho}{}^{\rho} \cdot \bar{a}_{.s_\nu}{}^{\rho-1}. \tag{4.4.4}$$

Consequently, if $\bar{a}_{r_\rho s_\nu}{}^{\rho-1} = 0$ we shall have $\bar{a}_{.s_\nu}{}^{\rho} = \bar{a}_{.s_\nu}{}^{\rho-1}$. From this it is clear that it is better to start by expanding the s_ν-th column, so that there will be no searching or relocating during the whole operation. We start the calculation with the first η-vector and gradually proceed to the last one.

Let us define the row vector $\beta_0{}^{\nu,\rho}$ for $0 \leqq \rho \leqq \nu$ by

$$\beta_0{}^{\nu,0} = e_0{}^{T};\ \beta_0{}^{\nu,\rho+1} = \beta_0{}^{\nu,\rho} \cdot {}_0E_{r_{\nu-\rho}}{}^{\nu-\rho}, \tag{4.4.5}$$

so that $\beta_0{}^{\nu,\nu} = \beta_0{}^{\nu}$ and:

$$\{\beta_0{}^{\nu,\rho+1}\}_i = \{\beta_0{}^{\nu,\rho}\}_i \quad \text{if } i \neq r_{\nu-\rho}, \tag{4.4.6}$$

$$\{\beta_0{}^{\nu,\rho+1}\}_{r_{\nu-\rho}} = \sum_{\ell=0}^{m} \{\beta_0{}^{\nu,\rho}\}_\ell \cdot \{{}_0\eta_{r_{\nu-\rho}}{}^{\nu-\rho}\}_\ell. \tag{4.4.7}$$

We see that only one new figure has to be calculated in each step of operation 5. This new figure can immediately be put in its place in the vector $\beta_0{}^{\nu,\rho}$ if we use this vector in expanded form (hence start with an expanded unit vector $e_0{}^{T}$). For the operation we need the last η-vector first and finish with the first one.

The third major operation, the pricing operation, is the same as in the explicit inverse algorithm. Also in the product-form algorithm the behaviour of elementary variables is of importance. The more there are in the basis, the fewer calculations have to be done in operation 5. In operation 1 there will also be an advantage if $\bar{a}_{.s_\nu}$ is a unit vector, say $= e_\ell$. Up to the iteration where y_ℓ left the basis, $\bar{a}_{.s_\nu}{}^{\rho} = e_\ell$ will hold, so that there will not be

any calculation during that part of operation 1. The same advantage can be obtained if a variable which entered the basis in iteration ν_1, but left it again in iteration $\nu_2 > \nu_1$ wants to re-enter it in iteration $\nu > \nu_2$, so that, for $\nu_1 \leqq \rho < \nu_2$, $\bar{a}_{.s_\nu}{}^{\rho}$ is a unit vector (as a consequence of rounding-off error this may not hold exactly in practice, so that the computational advantage may get lost).

As the computation proceeds we obtain more and more η-vectors, so that the iterations will take more and more time. Since the way in which basic and non-basic variables are interchanged is mostly rather inefficient, it finally becomes advantageous to restart the problem with a so-called re-inversion. In this operation we start afresh but select our main columns and main rows in such a way that the variables which were in the basis just before the re-inversion started are put in it again in the least possible number of iterations. The iterations in such a re-inversion are very simple since two of the three major operations need not be performed in it (operations 5 and 6). The columns that have to be introduced in the basis are selected one after the other. A chosen column is then updated and the pivot element is always chosen at a free place (e.g. the absolute largest among all those in free rows, i.e. in those rows which have not yet been but shall have to be selected as pivot row). The b column will also be recalculated by gradually transforming it during the re-inversion. In this way we obtain a considerably smaller number of new η-vectors, so that future operations will also be speeded up. It is clear that a re-inversion is also useful from the point of view of accuracy.

The product-form algorithm has been described for instance in [14] and [32].

4.5 *The Revised Product-Form Algorithm*

The calculation of the d row, necessary for the selection of the next main column, is a very expensive operation in the product-form algorithm since it involves two of the three major operations. Large linear programming problems always have the property that the non-basic matrices A^ν contain a large number of zero elements. Not only does this hold at the beginning but also at the end of the computation, when the number of non-zero elements in A^ν usually will not be more than 30 to 50% of the total number of elements, mn. Since in an iteration a $\bar{d}_j$ is only changed if the main row element in that column is non-zero it is clear that many of the non-basic $\bar{d}_j$ figures, say 75% on the average, will not change at all in the course of an iteration, so that their recalculation is quite superfluous. The d row could also be transformed in each iteration as is done in the straightforward algorithm. It is then necessary to have $\bar{d}^{\nu-1}$ and $\bar{a}_{r_\nu}{}^{\nu}$ available. The row $\bar{d}^{\nu-1}$ can be kept from the previous iteration but $\bar{a}_{r_\nu}{}^{\nu}$ has then to be calculated in iteration ν. The updating of the main row is of the same type as the recalculation of the d row, hence consists

of two operations: the calculation of $\beta_{r_\nu}{}^{\nu}$ and the calculation of $a_{r_\nu j}{}^{\nu} = \beta_{r_\nu}{}^{\nu} . a_{.j}$ for all j. But where $\beta_0{}^{\nu}$ can be expected to have a non-zero component at all non-elementary places (corresponding to rows where an x_j variable has replaced a y_i variable in the basis), the non-elementary part of $\beta_{r_\nu}{}^{\nu}$ will have the average fullness of the non-basis matrix, fullness to be defined as the ratio between the number of non-zero elements and the total number of elements. Hence in the revised product-form algorithm we shall update the main row in each iteration and use this row for the transformation of the d row. This is the major modification but many other revisions have also been incorporated in the new algorithm. They lead to a considerable reduction in the number of arithmetic operations and also to a reduction of the number of tape movements to be expected. A survey of all the revisions will now follow:

1. In iteration ν we calculate a row vector $\gamma_{r_\nu}{}^{\nu}$ which is $-\bar{d}_{s_\nu}{}^{\nu-1}$ times the main row of the inverse:

$$\gamma_{r_\nu}{}^{\nu} = -\bar{d}_{s_\nu}{}^{\nu-1} \beta_{r_\nu}{}^{\nu} = -\bar{d}_{s_\nu}{}^{\nu-1} e_{r_\nu}{}^{T} E_{r_\nu}{}^{\nu} E_{r_{\nu-1}}{}^{\nu-1} \ldots E_{r_I}{}^{1}, \quad (4.5.1)$$

(the multiplication by the scalar $-\bar{d}_{s_\nu}{}^{\nu-1}$ will be made clear later on). We define $\gamma_{r_\nu}{}^{\nu,\rho}$ for $0 \leqq \rho \leqq \nu$ by

$$\gamma_{r_\nu}{}^{\nu,0} = -\bar{d}_{s_\nu}{}^{\nu-1} e_{r_\nu}{}^{T} ; \gamma_{r_\nu}{}^{\nu,\rho+1} = \gamma_{r_\nu}{}^{\nu,\rho} . E_{r_{\nu-\rho}}{}^{\nu-\rho}, \quad (4.5.2)$$

so that $\gamma_{r_\nu}{}^{\nu,\nu} = \gamma_{r_\nu}{}^{\nu}$ and

$$\{\gamma_{r_\nu}{}^{\nu,\rho+1}\}_i = \{\gamma_{r_\nu}{}^{\nu,\rho}\}_i \quad \text{if } i \neq r_{\nu-\rho}, \quad (4.5.3)$$

$$\{\gamma_{r_\nu}{}^{\nu,\rho+1}\}_{r_{\nu-\rho}} = \sum_{\ell=1}^{m} \{\gamma_{r_\nu}{}^{\nu,\rho}\}_\ell . \{\eta_{r_{\nu-\rho}}{}^{\nu-\rho}\}_\ell . \quad (4.5.4)$$

We see that only one new figure has to be calculated in each step of this operation. We also see that we can work here with $E_{r_\nu}{}^{\nu}$ matrices instead of ${}_0E_{r_\nu}{}^{\nu}$ matrices, so that the η-vectors will have one element less.

2. The $\bar{d}_j$ will now be calculated during the "γ-operation" and the "pricing operation" by means of the following formulae:

$$d_{2i}{}^{\nu} = d_{2i}{}^{\nu-1} - \bar{d}_{s_\nu}{}^{\nu-1} \{\beta_{r_\nu}{}^{\nu}\}_i = d_{2i}{}^{\nu-1} + \{\gamma_{r_\nu}{}^{\nu}\}_i , \quad (4.5.5)$$

$$d_{1j}{}^{\nu} = d_{1j}{}^{\nu-1} - \bar{d}_{s_\nu}{}^{\nu-1} \{\beta_{r_\nu}{}^{\nu} . a_{.j}\} = d_{1j}{}^{\nu-1} + \{\gamma_{r_\nu}{}^{\nu} . a_{.j}\} . \quad (4.5.6)$$

These formulae show why it is better to calculate $\gamma_{r_\nu}{}^{\nu}$ instead of $\beta_{r_\nu}{}^{\nu}$.

3. Let us define: a virgin column (variable) is a column (variable) which has never been in the basis since the beginning of the last re-inversion; the other columns are called non-virgin. Let us suppose that in iteration $\nu-\rho$ ($1 \leqq \rho \leqq \nu$) variable $\bar{x}_{j_{\nu-\rho}}$ left the basis, so that, in tableau $\nu-\rho-1$, $\bar{x}_{j_{\nu-\rho}}$ was in the basis at place

$r_{\nu-\rho}$, $\bar{a}_{.j_{\nu-\rho}}^{\nu-\rho-1} = e_{r_{\nu-\rho}}$ and $\bar{a}_{.j_{\nu-\rho}}^{\nu-\rho} = \eta_{r_{\nu-\rho}}^{\nu-\rho}$.

Hence $\{\beta_{r_\nu}^{\nu,\rho+1}\}_{r_{\nu-\rho}} = \beta_{r_\nu}^{\nu,\rho+1}$ $e_{r_{\nu-\rho}} = \beta_{r_\nu}^{\nu,\rho+1} \cdot \bar{a}_{.j_{\nu-\rho}}^{\nu-\rho-1} =$

$= \beta_{r_\nu}^{\nu,\rho+1} \cdot E_{r_{\nu-\rho-1}}^{\nu-\rho-1} \ldots E_{r_1}^{1} \bar{a}_{.j_{\nu-\rho}} = \beta_{r_\nu}^{\nu} \ldots \bar{a}_{.j_{\nu-\rho}} = \bar{a}_{r_\nu, j_{\nu-\rho}}^{\nu}$.

Hence each new figure $\{\gamma_{r_\nu}^{\nu,\rho+1}\}_{r_{\nu-\rho}}$, $0 \leq \rho \leq \nu-1$, will be an element $\gamma_{r_\nu}^{\nu}\ \bar{a}_{.j}$ (will be $-\bar{d}_{s_\nu}^{\nu-1}$ times an element of the updated main row of iteration ν), so that the $\bar{d}_j^{\nu}$ values of the non-virgin variables $\bar{x}_{j_1}, \ldots, \bar{x}_{j_\nu}$ can be calculated during the "γ-operation":

$$\bar{d}_{j_{\nu-\rho}}^{\nu} = \bar{d}_{j_{\nu-\rho}}^{\nu-1} + \{\gamma_{r_\nu}^{\nu,\rho+1}\}_{r_{\nu-\rho}}, \quad \rho = 0, 1, \ldots, \nu-1, \qquad (4.5.7)$$

If a variable left the basis twice in the period before the ν-th iteration we can prevent by careful but simple bookkeeping that the $\bar{d}_j$ value is transformed twice.

4. For many values of ρ it is not necessary at all to calculate the new figure $\{\gamma_{r_\nu}^{\nu,\rho+1}\}_{r_{\nu-\rho}}$:

a. If $\bar{x}_{j_{\nu-\rho}}$ is in the ν-th basis we shall have

$\{\gamma_{r_\nu}^{\nu,\rho+1}\}_{r_{\nu-\rho}} = -\bar{d}_{s_\nu}^{\nu-1}$ if $\bar{x}_{j_{\nu-\rho}} = \bar{x}_{s_\nu}$ (hence if $\bar{x}_{j_{\nu-\rho}}$ is at place r_ν in the ν-th basis);

$= 0$ otherwise.

b. If $\bar{x}_{j_{\nu-\rho}}$ left the basis also in some iteration between the $(\nu-\rho)$-th and the ν-th one we did already calculate $\{\gamma_{r_\nu}^{\nu,\rho+1}\}_{r_{\nu-\rho}}$, so that we need not recalculate it provided we also store the figure in a separate list in all cases where it can be expected that it will have to be calculated again. This of course requires additional bookkeeping which, however, can be organized in such a way that it does not lead to undue complications.

5. It is clear that more and more virgin columns will become non-virgin when the calculation proceeds so that their $\bar{d}_j^{\nu}$ values can be found during the calculation of $\gamma_{r_\nu}^{\nu}$. In the pricing operation we use the non-basic columns of the original matrix and such a column will no longer be used if it becomes a non-virgin column. Usually the original matrix columns will be stored on tape. Therefore there is sense in removing the non-virgin columns from this tape at regular times, i.e. in making a new virgin-columns tape at regular times. This will reduce tape time in the pricing operation.

6. If a non-virgin variable which left the basis in iteration $\nu-\rho$ wants to re-enter it in iteration ν, we shall have

$$\bar{a}_{.s_\nu}^{\nu-1} = E_{r_{\nu-1}}^{\nu-1} \ldots E_{r_{\nu-\rho+1}}^{\nu-\rho+1}\ \bar{a}_{.s_\nu}^{\nu-\rho} = \\ = E_{r_{\nu-1}}^{\nu-1} \ldots E_{r_{\nu-\rho+1}}^{\nu-\rho+1}\ \eta_{r_{\nu-\rho}}^{\nu-\rho}, \qquad (4.5.8)$$

so that the operation "update main column" can be started with the $(\nu-\rho)$-th η-vector. The number $\nu-\rho$ to be known for this can immediately be obtained from the "γ-operation", since we can,

during the gradual transformation of the $\bar{d}_j$ figures of the non-virgin columns by means of (4.5.7), store the number $\nu-\rho$ in all those cases where $\bar{d}_{j\nu-\rho}{}^{\nu}$ appears to be the most negative one of those already calculated. At the end of the γ- and pricing operation we then know whether a non-virgin or a virgin column wants to re-enter the basis and in the former case we also have the iteration number $\nu-\rho$.

7. When performing a re-inversion we know which columns of the original matrix we have to select as main columns. Instead of taking them one after the other it would be better to compare them and take that one which will have a minimum number of non-zero elements at that stage of the calculation since such a procedure would generally lead to less full η-vectors. But in order to be able to compare them in iteration ρ of the re-inversion we must have them all in updated form: we must compare the columns $a_j{}^{\rho}$ instead of the columns a_j. Our proposal now is to perform the re-inversion in a straightforward way: in each iteration we transform all columns which have been selected for the re-inversion and have not yet been chosen as main column. Our criterion for the choice of the main column will now be: the column with a minimum number of non-zero elements (in case of ties the first one among the tied ones); the pivot element in that column will be chosen such that a. it is in a free row, b. it is not too small and c. it is in that row of the remaining ones which has a minimum number of elements. Each new transformed main column can then be considered as a new η-vector. It will no longer be transformed in the re-inversion. It can be expected that this way of choosing pivot columns and pivot rows will lead to considerably less full η-vectors and hence to fewer arithmetic and tape operations in the iterations following the re-inversion. It seems to be as good as or better than Markowitz' elimination form [31] but less complicated. During the re-inversion the b column will be gradually transformed. After the re-inversion the d row has to be recalculated and a new virgin-columns tape has to be made.

If a good initial basis is known, we can also start the calculation with a re-inversion which should then be called pre-inversion. Such a pre-inversion routine can be made such that a column is rejected as main column if no reasonably large pivot element at a free place can be found in it. A pre-inversion routine will be particularly useful if a problem has to be solved whose matrix differs only slightly from the matrix of a previous problem. We can then give the final basis of the previous problem as an initial basis for the new problem. If in this new problem this predesigned basis appears to be a singular matrix, then the only consequence is that one or more columns will be rejected in the pre-inversion. We may obtain a b column with some negative figures in it, however, so that an efficient means for dealing with such a situation must be available. For a special problem we shall describe such a means in section 6.3, last paragraph but one.

From the foregoing it follows that the following operations can be distinguished in the main part of the revised product-form algorithm:

1. Update main column by using (4.4.2) in case of a virgin variable entering the basis and (4.5.8) in case of a non-virgin variable.
2. Find pivot element in main column by means of (3.2.15).
3. Make new η-vector (using (3.2.16) and (3.2.17)).
4. Transform right-hand member (using (4.3.3).
5. Perform "γ-operation" with the aid of (4.5.3) and (4.5.4) and use (4.5.7) to transform the $\bar{d}_j$ of the non-virgin variables and choose the most negative one, if any, and the iteration number of the corresponding η-vector.
6. Perform pricing operation by applying (4.5.6) to the virgin columns of the original matrix. Replace most negative $\bar{d}_j$ of operation 5, if any, by a more negative one, if any, and keep its column in that case.

Operations 1, 5 and 6 are again the major operations.

Also in the revised product-form algorithm it is advantageous if many elementary variables stay in the basis. But moreover use is made of the behaviour of variables which go in and out the basis again and again. These variables will lead to fewer calculations in the operations 1 and 5. The algorithm has primarily been designed to reduce the number of arithmetic operations as much as possible since this was thought to be of much importance now that computers are available with very fast tapes and simultaneous computing and tape operations. In chapter 5 it will appear, moreover, that the revised product-form algorithm will also take up less tape time than any other algorithm.

Remark: It is clear that the number of arithmetic operations in the explicit inverse algorithm can also be reduced considerably if instead of applying $\beta_0{}^\nu$ to the original matrix (operation 6, section 4.3) we apply $\beta_{r_\nu}{}^\nu$ to it and transform the d row in each iteration by means of (4.5.5) and (4.5.6). The computational scheme given at the end of section 4.3, in which it was only necessary to read the inverse once can easily be modified in this respect.

4.6 *Computational Versions for the Dual Simplex Method*

For the dual simplex method we can also develop several computational versions. We shall assume compressed column storage and shall then have:

a. Straightforward version. The next main row number is chosen during the transformation of the b column which must now precede the matrix; the quotients (3.3.1) can be calculated during the transformation of the columns of the extended matrix. Presumptive main columns can be kept in store. At the end of the transformation we can then immediately perform the divisions by the

pivot element in the main column after which the next iteration can be started with the transformation of the b column, etc.

b. Explicit inverse version. We shall give the scheme in such a way that the inverse only has to be read once in each iteration: Suppose that the rows $\beta_i^{\nu-1}$ of the inverse are on tape, the row $\beta_{r_\nu}^{\nu-1}$ is in store (r_ν is the main row number for the next iteration); the vectors $b^{\nu-1}$ and $\bar{d}^{\nu-1}=\begin{pmatrix} d_1^{\nu-1} \\ \beta_0^{\nu-1} \end{pmatrix}$ have already been calculated. We now proceed as follows:

1. Calculate the quotients $\frac{\{\beta_0^{\nu-1}\}_i}{-\{\beta_{r_\nu}^{\nu-1}\}_i}$ for those i for which the denominator is positive and select the smallest one and the corresponding column number ℓ.

2. Calculate the row vector $\beta_{r_\nu}^{\nu-1}.A$ and the quotients $\frac{d_{1j}^{\nu-1}}{-\beta_{r_\nu}^{\nu-1}.a_{.j}}$ for those j for which the denominator is positive. If such a quotient is smaller than the smallest one already calculated in 1 and 2, keep then the corresponding column $a_{.j}$ in store.

3. If no quotient has been calculated in 1 and 2 (no positive denominator), stop, the problem is not feasible; otherwise perform 4.

4. If $0 < s_\nu \leqq n$, calculate $\{\eta_{r_\nu}^{\nu}\}_{r_\nu} = \frac{1}{a_{r_\nu s_\nu}^{\nu-1}}$ ($a_{r_\nu s_\nu}^{\nu-1}$ is the s_ν-th element from the row $\beta_{r_\nu}^{\nu-1}.A$);
if $s_\nu = n+\ell$, find $\{\beta_{r_\nu}^{\nu-1}\}_\ell$ and calculate $\{\eta_{r_\nu}^{\nu}\}_{r_\nu} = \frac{1}{\{\beta_{r_\nu}^{\nu-1}\}_\ell}$.

5. Transform $\beta_0^{\nu-1}$ and $d_1^{\nu-1}$:

$$\beta_0^{\nu} = \beta_0^{\nu-1} + \beta_{r_\nu}^{\nu-1}\{\eta_{r_\nu}^{\nu}\}_0,$$
$$d_1^{\nu} = d_1^{\nu-1} + (\beta_{r_\nu}^{\nu-1}.A)\{\eta_{r_\nu}^{\nu}\}_0,$$

where $\{\eta_{r_\nu}^{\nu}\}_0 = \frac{\bar{d}_{s_\nu}^{\nu-1}}{-\bar{a}_{r_\nu s_\nu}^{\nu-1}}$ follows immediately from 1 or 2.

6. Read the rows $\beta_i^{\nu-1}$ of the inverse ($1 \leqq i \leqq m$); calculate $a_{is_\nu}^{\nu-1} = \beta_i^{\nu-1} a_{.s_\nu}$ (or find $\bar{a}_{in+\ell}^{\nu-1} = \{\beta_i^{\nu-1}\}_\ell$; calculate $\{\eta_{r_\nu}^{\nu}\}_i = -\bar{a}_{is_\nu}^{\nu-1}\{\eta_{r_\nu}^{\nu}\}_{r_\nu}$ and transform $\beta_i^{\nu-1}$:

$$\beta_i^{\nu} = \beta_i^{\nu-1} + \{\eta_{r_\nu}^{\nu}\}_i \beta_{r_\nu}^{\nu-1};$$

calculate $b_i^{\nu} = b_i^{\nu-1} + \{\eta_{r_\nu}^{\nu}\}_i b_{r_\nu}^{\nu-1}$; if $b_i^{\nu} < 0$ and smaller than any b figure already calculated keep then i and β_i^{ν} in store as presumptive main row number and main row of the inverse, respectively; write β_i^{ν} on tape.

7. When 6 has been completed for all i, we can return to 1.

c. Revised product-form version. The operations will be:

1. Calculate $\beta_{r_\nu}^{\nu-1}$, the elements $\beta_{r_\nu}^{\nu-1}\bar{a}_{.j}$ and the quotients (3.3.1) for the non-virgin variables. Keep the smallest quotient and the corresponding η-vector number in store.

2. Calculate $\beta_{r_\nu}^{\nu-1} a_{.j}$ for all virgin columns, calculate the quotients (3.3.1). If such a quotient is smaller than the smallest

one already calculated, keep then the corresponding column $a_{.j}$ in store.

3. If no quotient has been calculated in 1 and 2, stop, the problem is not feasible; otherwise perform 4.

4. See 5 of the dual explicit inverse algorithm.

5. Update main column using (4.4.2) or (4.5.8).

6. Make new η-vector.

7. Transform right-hand member and select next main row number.

Re-inversions can be made at regular times in the way described in section 4.5.

5. NUMERICAL COMPARISON OF THE DIFFERENT ALGORITHMS

5.1 *Introduction*

In this chapter an attempt will be made to compare the different computational versions of the simplex method for a high-speed electronic computer. Our concern will be with a large linear programming problem of, say, 300 or more constraints and 500 or more non-basic variables. The matrix will be supposed to have only a few non-zero elements (e.g. 2 or 3%) and the computer code to be used will make use of this fact, so that a multiplication or addition will only be carried out if neither of the two figures is zero. In section 5.2 we shall briefly discuss some measures of comparison which have to be considered. In section 5.3 we shall discuss the behaviour of linear programming matrices when being solved by means of the simplex method. Section 5.4 will be concerned with the assumptions we have to make for the comparison whereas some results of the comparison will be given in section 5.5. We shall use the following abbreviations in this chapter:

SFA : the straightforward algorithm,
EIA : the explicit inverse algorithm,
PFA : the product-form algorithm,
RPFA: the revised product-form algorithm.

5.2 *Measures of comparison*

As long as there were no computers with simultaneous tape/drum and computing operations and tape operations themselves were rather slow it was immediately clear that that algorithm would be the best which minimized the number of tape/drum transfers. Actually this was the main reason for the product-form algorithm to be developed some years ago (see Orchard-Hays, [32]). But for modern computers this argument will no longer hold and a comparison becomes more complicated since there are many other factors to be taken into account. We mention:

a. Number of arithmetic operations.

Since in all algorithms the greater part of the multiplications are followed by an addition and since the number of divisions is relatively small (they only occur in two minor operations which, moreover, are the same for all algorithms), total time needed for arithmetic operations will be approximately proportional to the total number of multiplications.

b. Number of additional bookkeeping operations.
Some of these additional operations may be the same in all algorithms, e.g. changes in a basis identification list in each iteration. Some may be directly proportional to the number of multiplications. For instance, in the operation p = a + b.c which is the usual type of operation in the simplex method we always have the test "a = 0?" preceding the addition and we also have a test whether p is not too small compared with a (if p is non-zero and $|p| \ll |a|$ it is assumed to be non-zero owing to rounding-off error and replaced by zero). Some, however, will be a consequence of the organization of the code. When using SFA and compressed column storage, we shall have many search and relocation operations (see section 4.2) which hardly occur in EIA and do not occur in PFA or RPFA.

c. Number of tape/drum transfers.
This number will of course be dependent on the computer to be used, e.g. on the size of its fast memory and on the size of the records to be transferred. Therefore we rather use here as criterion the quantity of numbers to be processed, which - if no fast memory is used for permanent storage of the information concerning the problem (matrix, inverse or η-vectors) - will be approximately proportional to the number of tape/drum transfers.

d. Accuracy.

e. Flexibility of the code, e.g. restart possibilities.

f. Complexity of the code itself.

5.3 *Behaviour of Practical Linear Programming Problems*

In order to be able to make any comparison we shall have to make several assumptions about the behaviour of a large linear programming problem that is being solved by means of the simplex method. Therefore we studied the behaviour of a number of large linear programming problems.

From chapter 4 it follows that the most important factors which determine the total computing time besides the algorithm itself and the speed of the computer are:

1. size of the problem,
2. number of iterations,
3. initial fullness of the matrix and the way fullness grows if the calculation proceeds (note that fullness is defined as the number of non-zero's in the non-basis matrix divided by the total number of elements, i.e. mn),
4. behaviour of elementary variables, especially the total number which will leave the basis (only of importance in EIA, PFA and RPFA),
5. the number of re-inversions in PFA or RPFA and the iteration numbers at which they are performed,
6. the question whether single or double-length arithmetic has to be used.

Since for a particular problem the size is fixed and the number of iterations is the same for all algorithms (usually between m and 2.5 m), we only have to consider the last four points.

The study of a number of large linear programming problems has taught us that in most cases fullness grows rather rapidly in the first iterations (about one third of the total number) after which it fluctuates around a rather constant level whereas in some cases it goes down in the last iterations. In the first iterations an elementary column will nearly always be replaced in the basis by a non-elementary column. These iterations are followed by a great number of iterations in which the number of elementary columns in the basis only slightly decreases until at the end many of them (30 to 40% is possible) are still in the basis. About the re-inversions we learned that computing time is highly influenced by their number but that they may be performed a few iterations earlier or later without affecting total time noticeably. There is no sense in doing re-inversions in the first part of the calculation when the elementary columns are still leaving the basis rather rapidly and when fullness in still growing rapidly. They should be performed at regular intervals. About accuracy we learned that surprisingly large problems can be solved in single precision (say with 25 to 30 bits word length, floating point), presumably owing to the fact that practical large matrices are very sparse and have a special structure. Double precision, only for the re-inversion, seems to be satisfactory for problems with up to, say, 400 or 500 constraints. On this experience are based the assumptions for a theoretical comparison of the algorithms which will be given in the next section.

5.4 *Theoretical Assumptions about the Behaviour of Linear Programming Problems*

Notation:

- m number of rows of the matrix,
- n number of columns,
- k total number of iterations,
- $\varphi(\nu)$ fullness of the non-basis matrix at iteration ν, $\varphi(0) = \pi$, $\varphi(k) = q$,
- τ rate of growth of fullness $= \frac{\varphi - \pi}{\alpha m}$ if fullness grows from π to φ in αm $(0 < \alpha)$ iterations,
- $\sigma(\nu)$ number of elementary columns in the basis at iteration ν.

Assumptions:

1. About elementary columns:

a. In EIA:

$\sigma(\nu) = \nu$ if $1 \leqq \nu \leqq \alpha m$, $0 < \alpha \leqq 1$,
$= \alpha m$ if $\nu \geqq \alpha m$;

in PFA or RPFA:

$\sigma(\nu) = \nu$ if $1 \leqq \nu \leqq \alpha_0 m$, $0 < \alpha_0 \leqq 1$,
$= \alpha_0 m$ if $\nu \geqq \alpha_0 m$ and we have not made a re-inversion,
$= \alpha_i m$ in stage i, i.e. after i re-inversions.

b. In EIA:

if $\nu \leqq \alpha m$ no elementary column will re-enter the basis;
if $\nu > \alpha m$ there will be a chance γ' in each iteration that an elementary column re-enters the basis;

in PFA or RPFA:

if $\nu \leqq \alpha_0 m$ no elementary column will re-enter the basis;
if $\nu > \alpha_0 m$ and we have already made i re-inversions there will be a chance γ_i' in each iteration that an elementary column will re-enter the basis.

2. About fullness:

a. In SFA and EIA: the non-basis matrix, the non-elementary part of the inverse and each column of the matrix will have a fullness which grows linearly in the first βm iterations ($\beta \geqq \alpha$ in EIA) after which it remains constant:

$$\varphi(\nu) = \pi + \nu\tau \quad \text{if } 1 \leqq \nu \leqq \beta m;$$
$$= \varphi \quad \text{if } \nu \geqq \beta m,$$

so that

$$\tau = \frac{\varphi - \pi}{\beta m};$$

in PFA and RPFA:
stage 0 (before the first re-inversion):

$$\varphi(\nu) = \pi + \nu\tau_0 \quad \text{if } 1 \leqq \nu \leqq \alpha_0 m;$$
$$= \varphi_0 \quad \text{if } \nu \geqq \alpha_0 m,$$

so that

$$\tau_0 = \frac{\varphi_0 - \pi}{\alpha_0 m};$$

stage i (after i re-inversions, $i > 0$):

$\varphi(\nu) = \varphi_i$ (constant);

re-inversion in PFA:

$\varphi(\nu_i) = \pi + \tau_i \nu_i$ (ν_i counts the iterations in the re-inversions),

so that $\tau_i = \frac{\varphi_i - \pi}{\alpha_i m}$ (note that $\alpha_i m$ is the number of iterations in the re-inversion);

re-inversion in RPFA:

$\varphi(\nu_i) = \pi + \frac{\tau_i}{\alpha_i m}\nu_i^2$ (here we assume a quadratic growth of the fullness; note that also here $\varphi(\alpha_i m) = \varphi_i$).

b. With a fullness $\varphi(\nu)$ every element of the matrix has the same chance $\varphi(\nu)$ of being non-zero. The chance that a multiplication bc must be carried out is $\{\varphi(\nu)\}^2$ unless we know beforehand that b or c will be non-zero.

c. Main columns and the non-elementary part of the main rows will have the average fullness of the non-basis matrix, i.e. have the fullness $\varphi(\nu)$.

3. About the number of non-zero's in $\beta_0^{\nu,\rho}$ or $\gamma_{r_\nu}^{\nu,\rho}$: This number will be:

if $\rho \leqq \alpha_i m$: ρ in $\beta_0^{\nu,\rho}$ and $\rho\varphi(\nu)$ in $\gamma_{r_\nu}^{\nu,\rho}$
if $\rho \geqq \alpha_i m$: $\alpha_i m$ in $\beta_0^{\nu,\rho}$ and $\alpha_i m\ \varphi(\nu)$ in $\gamma_{r_\nu}^{\nu,\rho}$.

4. About non-virgin variables re-entering the basis:

a. In stage i there is a chance γ_i in each iteration that a non-virgin variable re-enters the basis (if i = 0 only for $\nu > \alpha_0 m$).

b. With a chance $\gamma_i' < \gamma_i$ this variable will be an elementary variable which re-enters for the first time. It is then assumed to have left the basis in iteration $\alpha_i m$ (consequently the operation "update main column" will only entail multiplications if $\rho \geqq \alpha_i m$; in RPFA we even start with $\rho = \alpha_i m$).

c. With a chance $\gamma_i'' = \gamma_i - \gamma_i'$ this variable will enter the basis for the second time. We assume then that it went in for the first time in iteration $\alpha_i m$ and stayed in the basis during one third of the iterations until the next re-inversion.

5. About re-inversions:

a. The decision whether a re-inversion will be performed in PFA or RPFA or not will not be influenced by accuracy questions.

b. The first re-inversion will be made after at least $\alpha_0 m$ iterations.

6. About accuracy:

Either the whole calculation can be done in single precision or only the re-inversion part of the calculation has to be done in double precision. In the latter case we assume that a double-length multiplication can be compared with 3 single-length multiplications and that in EIA re-inversions are performed in double-length to improve accuracy.

7. Additional assumptions for RPFA:

a. We shall neglect the fact that $\{\gamma_{r_\nu}^{\nu,\rho}\}_{r_\nu - \rho}$ need not be calculated at all in some cases (see section 4.5, point 4);

b. A new virgin-columns tape will only be made after a re-inversion.

8. About the code to be used:

a. There will always be compressed storage, so that only non-zero elements are stored, together with a list of row numbers. This holds for the matrices A^ν in SFA, for $\{B^\nu\}^{-1}$ and A in EIA; for the η-vectors and A in PFA and RPFA.

b. There is no permanent place in the fast store for the data mentioned in a; they are stored on tapes or drums.

9. About the operations to be compared:

Only the major operations will be compared. The other operations only take a small part of the total time. Moreover they are the same or nearly the same for all algorithms. Hence we shall neglect: in all algorithms: operations "find pivot element", "make new η-vector", "transform b column",

in SFA : transformation of the d row,
in EIA : transformation of $\beta_0^{\nu-1}$,
in PFA : the transformation of the $\{\eta_{r_\nu}{}^{\nu}\}_0$ elements, i.e. of the d-row element of an η-vector,
in RPFA: the recalculation of the d row after a re-inversion.

It is realized that these assumptions are sometimes arbitrary or even unrealistic. In those cases, however, either their influence is rather small (e.g. 4 b and c) or they over-estimate the time required (e.g. 3), while they can hardly be replaced by more realistic assumptions. The results obtained with these assumptions appeared to be in good agreement with practical results as could be concluded from some experiments. Assumption 3 only is a strongly over-estimating one. Actually it assumes that, if $\rho \leqq \alpha m$, the place $r_{\nu-\rho}$ in $\{\beta_0{}^{\nu,\rho}\}_{r_{\nu-\rho}}$ or $\{\gamma_{r_\nu}{}^{\nu,\rho}\}_{r_{\nu-\rho}}$ is always a new one, hence that between iteration $\nu-\alpha m$ and iteration ν a pivot element is always chosen in a different row. This is certainly not true in practice, so that the actual number of non-zero's in $\beta_0{}^{\nu,\rho}$ or $\gamma_{r_\nu}{}^{\nu,\rho}$ is less than the theoretical number. The fullness $\varphi(\nu-\rho)$ of the η-vector $\eta_{r_{\nu-\rho}}{}^{\nu-\rho}$, however, is relatively large as long as ρ is small. Consequently the overestimation is even larger. Considering instead of assumption 3 a gradual growth of the number of non-zero's in $\beta_0{}^{\nu,\rho}$ would lead to an underestimation.

With the assumptions of this section it is possible to calculate the number of multiplications in all algorithms as well as the quantity of numbers to be processed. The formulae will not be given here but in the next section the results of a comparison based on these formulae will be presented.

5.5 *Comparison of the Algorithms*

1. Number of multiplications.

The revised product-form algorithm leads to by far the least number of multiplication operations. In a typical example ($k = 2m$, $n = 2m$, $\pi = .03$, $\varphi = .30$, $\beta = 1$, two re-inversions in PFA and RPFA after m and 1.5 m iterations, $\alpha_i = \alpha = .7$ for $i = 0, 1, 2$; $\varphi_i = .3$, $\gamma_i' = \gamma' = .15$ and $\gamma_i'' = .25$ for all i), we obtain if

SFA = 100:

EIA = 56.1

1.	3.2
2.	33.9
3.	19.0

PFA = 79.6

1.	21.2
2.	35.7
3.	19.0
R.	3.7

RPFA = 31.0 { 1. 16.1; 2. 8.5; 3. 4.8; R. 1.7 }

1, 2, and 3 stand for the major operations; operation 1 is the updating of the main column; 2 is the transformation of the inverse in EIA, the calculation of $\beta_0{}^{\nu}$ ($\gamma_{r_\nu}{}^{\nu}$) in PFA(RPFA) and 3 is the pricing operation. The letter R stands for re-inversion.

In general we can conclude:

a. As far as multiplications are concerned the revised product-form algorithm is the best with the explicit inverse algorithm second best.

b. The difference between the straightforward algorithm and the inverse algorithms becomes smaller if n decreases, if the ratio π/φ increases, or if α or β increases.

c. Re-inversions have a considerable influence on the total number of multiplications. For the above mentioned problem we have:

	PFA	RPFA
No re-inversion	136.6	62.6
1 re-inversion at $\frac{4}{3}$ m	91.2(1.83)	38.0(0.84)
2 re-inversions (at m and $\frac{3}{2}$ m)	79.6(3.66)	31.0(1.68)
3 re-inversions (at m, $\frac{4}{3}$ m and $\frac{5}{3}$ m)	73.1(5.50)	27.9(2.52)

The figure between brackets is the part of the total number which is due to the re-inversions.

d. If the explicit inverse algorithm is revised, so that the d-row is transformed in each iteration with the help of the calculated main row, the figure for the number of multiplications in EIA can be decreased to 41.9.

2. Additional bookkeeping operations.

We have already seen in section 5.2 that there will be a large number of additional bookkeeping operations in SFA which do not occur or do not occur to such an extent in the other algorithms. This makes the straightforward algorithm even less attractive.

3. Quantity of numbers to be processed.

Realizing that:
in SFA we have to read and write the matrix in the course of each iteration,
in EIA we have to read and write the inverse and to read the original matrix in each iteration,
in PFA and RPFA we have to read the η-vectors and the original matrix (virgin columns in RPFA) in each iteration whereas writing can be neglected,
we obtain the following figures for the quantity Q of numbers to be processed:
If
in SFA $Q_r = Q_w = 100$ (r stands for read, w for write),
then
in EIA $Q_r = 45.5$ and $Q_w = 32.6$,

in PFA	no re-inversion	: Q_r = 106.0
	1 re-inversion ($\frac{4}{3}$m)	: Q_r = 82.0
	2 re-inversions (m, $\frac{3}{2}$ m)	: Q_r = 76.9
	3 re-inversions (m, $\frac{4}{3}$m, $\frac{5}{3}$ m)	: Q_r = 74.7
in RPFA	no re-inversion	: Q_r = 95.0
	1 re-inversion ($\frac{4}{3}$m)	: Q_r = 66.9, Q_w = 2.0
	2 re-inversions (m, $\frac{3}{2}$ m)	: Q_r = 58.8, Q_w = 4.0
	3 re-inversions (m, $\frac{4}{3}$ m, $\frac{5}{3}$ m)	: Q_r = 55.8, Q_w = 5.9

Note that there will be writing in RPFA during a re-inversion since this will be done in a straightforward way.

Again we see that the revised product-form algorithm is the best but now with the product-form algorithm itself as second best. This result is not surprising since the product-form algorithm has been designed to reduce tape operations, especially the amount of writing. We also see that tape operations will be reduced by re-inversions. The same result has been obtained in a number of other examples.

4. Accuracy.

As soon as double-length multiplications have to be done the situation for the inverse algorithms will become even better since in SFA we have to do the complete calculation in double length. Assuming in EIA and PFA that we perform two double-precision re-inversions and that one double-length multiplication is equivalent to three single-length multiplications, we obtain the following picture for the number of multiplications in the example mentioned:

SFA	300	
EIA	88.5	1. 3.2 2. 33.9 3. 19.0 R. 32.4
PFA (2 re-inversions)	87.0	1. 21.2 2. 35.7 3. 19.0 R. 11.1
RPFA (2 re-inversions)	34.6	1. 16.1 2. 8.5 3. 4.8 R. 5.1

The advantages of the revised product-form algorithm are very clear but the product-form algorithm itself seems to become the second best. The quantity of numbers to be processed will also increase if information has to be stored in double precision, so that here we have another argument for the revised product-form algorithm. Note that, even if no double-length operations are

used, the re-inversion feature of PFA or RPFA will make these algorithms more accurate.

5. Flexibility of the code.

It is clear that the re-inversion feature of PFA or RPFA gives these algorithms excellent restart possibilities. If a list of variables in the current basis is taken out of the machine, one can always restart later on with a re-inversion. In RPFA the additional advantage is then obtained that a set of η-vectors with very few non-zero elements is generated in this way. Since re-inversion is so very cheap, amendment programs are hardly necessary.

The product-form algorithm has the disadvantage that afterwards changes in the objective function are hard to make since this would entail a change of the "zero"-element of all the η-vectors. In the revised product-form algorithm we can always recalculate the d row without having to alter anything in the η-vectors. This makes the new algorithm more flexible. Finally we have seen in section 4.6, point c that the revised product-form algorithm can also be used as computational version in the dual simplex method.

6. Complexity of the code.

It is clear that the straightforward algorithm will require the simplest code and that the complexity of the code will increase in the order SFA, EIA, PFA, RPFA. The latter algorithm may be too complicated for smaller machines. For those machines the explicit inverse algorithm seems to be preferable.

Conclusions

1. For large computers the revised product-form algorithm is by far the best one.
Reasons:
it leads to fewer multiplications than any other algorithm;
it leads to fewer tape/drum transfers;
it can work partly in double length (the re-inversion) without leading to a big increase in computing time;
it has excellent restart possibilities.

2. For medium-size computers for which the revised product-form algorithm is too complicated the explicit inverse algorithm seems to be preferable.
Reasons:
it generally leads to fewer multiplications than the product-form algorithm;
its code is simpler than that of the product-form algorithm;
since the size of the problems will be restricted, serious rounding-off errors in single-length operations can hardly be expected, so that the advantages of the product-form algorithm in this respect are of less importance.

3. For small computers, finally, the straightforward algorithm seems to be the only possible one since its code is the simplest.

6. SOME SPECIAL LINEAR PROGRAMMING PROBLEMS

6.1 *Introduction*

In this chapter some special linear programming problems will be considered which will occur as sub-problem in the non-linear programming procedures to be described in the chapters 7 - 11. No attempt will be made to explore more special structures such as network structures, triangularity, etc. Section 6.2 will be concerned with bounded-variables problems; in section 6.3 we shall consider the problem of minimizing the sum of the absolute values of the differences between linear functions and known figures. This problem will be called the absolute-value problem.

6.2 *Bounded-Variables Problems*

In some linear programming problems the variables are also restricted by an upper bound, so that the problem is of the type:

$$\text{Max } \{p^T x \mid Ax+y = b,\ 0 \leqq x \leqq c_1;\ 0 \leqq y \leqq c_2\}, \tag{6.2.1}$$

where c_1 and c_2 are non-negative n- and m-component vectors, respectively (some of the components may be infinite). This problem could be solved in the conventional way by adding the upper-bound constraints as extra equations to the tableau but this would lead to a much larger problem and consequently to longer computing times.

As already pointed out by Charnes and Lemke [8] and Dantzig [12] it is equally well possible to work with an m by n tableau in which we indicate for each non-basic variable whether it is at its upper bound or not (i.e. if $x_j + \bar{x}_j = c_{1j}$, we consider either x_j or $\bar{x}_j$; these two variables have the same upper bounds and, if $x_j = c_{1j}$, then the non-basic variable $\bar{x}_j$ will be considered). Pivot columns will be determined in the normal way. The determination of the pivot element will be more complicated, however. If for the iteration considered the basic variables, denoted by y_i*, have b values b_i* and upper bounds $c(y_i$*) and if the variable to be introduced in the basis is x_s* with column $\bar{a}_{.s}$* and upper bound $c(x_s$*), then we shall increase x_s* from 0 to

$$\lambda^* = \min (\lambda_1^*, \lambda_2^*, \lambda_3^*), \tag{6.2.2}$$

where

$$\lambda_1^* = \max \{\lambda \mid b^* - \lambda \bar{a}_{.s}^* \geqq 0\}, \tag{6.2.3}$$

$$\lambda_2^* = c(x_s^*), \tag{6.2.4}$$

$$\lambda_3^* = \max \{\lambda \mid b^* - \lambda \bar{a}_{.s}^* \leqq c(y^*)\} \tag{6.2.5}$$

($c(y^*)$ is the vector with components $c(y_i^*)$).

If, as will be assumed, none of the variables in the initial basis exceeds its upper bound, then the described way of choosing λ will guarantee that this will never happen. If $\lambda^* = \lambda_1^*$, the transformation formulae are the same as in the normal simplex method. If $\lambda^* = \lambda_2^*$, we shall put x_s^* at its upper bound, i.e. we replace b^* by $b^* - c(x_s^*)\bar{a}_{.s}^*$, $\bar{a}_{.s}^*$ and $\bar{d}_s^*$ by $-\bar{a}_{.s}^*$ and $-\bar{d}_s^*$ and consider $\bar{x}_s^*$ instead of x_s^* as non-basic variable. In this case there is no further transformation. If $\lambda^* = \lambda_3^*$ and row r is the chosen row, then y_r^* will leave the basis. After the normal transformation with a negative pivot element, we shall then put y_r^* at its upper bound.

Proceeding in this way we shall arrive at the optimum solution in a finite number of iterations. This number will not be reduced by the application of a special bounded-variables technique. Some of the transformations (if $\lambda^* = \lambda_2^*$) will be very simple, however, and a great computational advantage will arise if we select more negative d_j figures and their columns at the same time. This will avoid search operations on tape for the second-best main column. There will then be no transformation of the matrix or the inverse and no "calculate β_0^ν $(\gamma_{r_\nu}^\nu)$" or "pricing" operation while no new η-vector will be made.

Wagner [38] has pointed out that the dual simplex method is perhaps more suited to solving bounded-variables problems, the reason being that we can avoid a phase 1 if we start with putting all variables at their upper bounds the p_j-values of which are positive. In that case the first d row will be non-negative. The criterion for selecting a main row will now be: any row r with either $b_r^* < 0$ or $c(y_r^*) - b_r^* < 0$ (usually the row belonging to the smallest of these figures is taken). In the former case we can select a pivot element by the normal criterion and perform the transformation; in the latter case we first multiply row r by -1, replace b_r^* by $c(y_r^*) - b_r^*$ and replace y_r^* by $\bar{y}_r^*$. Next we can select the pivot element in the normal way and perform the transformation. It is worth mentioning, however, that, if the variable to be introduced in the basis would get a b value $> c(x_s^*)$ (so that it could be removed in the next iteration), we do better to reject $\bar{a}_{rs}^*$ as a pivot element, put x_s^* at its upper bound and choose a second pivot element in the same row, using the normal criterion but ignoring the s-th column. If the new candidate for introduction should again exceed its upper bound in the basis, we also reject this variable, put it at its upper bound and choose a third one in the same row, etc. The rejection of pivot elements in a row and

upper bounding of the corresponding variables will lead to d_j figures $\leqq 0$ but the final transformation with a pivot element in that row will make them all non-negative again as can be easily verified.

This procedure will again be finite, whereas some of the iterations will be very simple: the updating of the main column will be the only major operation that has to be done in the dual explicit inverse or dual revised product-form algorithm, whereas search operations on tape can be prevented to a large extent by the selection of more presumptive main columns.

As an example and for later use we shall consider the problem

$$\begin{aligned} \text{Max}\{p^T x \mid & A_1 x \leqq 0;\ A_2 x = 0;\ -1 \leqq x_j \leqq 1 \text{ if } j \in J_1; \\ & 0 \leqq x_j \leqq 1 \text{ if } j \in J_2;\ -1 \leqq x_j \leqq 0 \text{ if } j \in J_3; \\ & x_j = 0 \text{ if } j \in J_4\}, \end{aligned} \tag{6.2.6}$$

where $J_1 + J_2 + J_3 + J_4 = J = \{1, \ldots, n\}$, and A_1 and A_2 are m_1 by n and m_2 by n matrices, respectively.

In this problem we also have variables which are not restricted by a zero lower bound; it is profitable to increase such a variable if its d_j value is less than zero and to decrease it if this d_j value is greater than zero.

Writing $A_1 x + y = 0,\ y \geqq 0;\ A_2 x + z = 0, z = 0$, we have new variables y_i with upper bound ∞ and artificial variables z_i.
Although the primal bounded-variables technique can be applied to this problem we prefer a description by means of the dual bounded-variables technique. Here we proceed as follows:

1. If $j \in J_1$ and $p_j \geqq 0$, put x_j at its upper bound (i.e. substract $a_{.j}$ from b, replace $a_{.j}$ by $-a_{.j}$);
   if $j \in J_1$ and $p_j < 0$, put x_j at its lower bound (i.e. add $a_{.j}$ to b);
   if $j \in J_2$ and $p_j \geqq 0$, put x_j at its upper bound;
   if $j \in J_3$ and $p_j < 0$, put x_j at its lower bound;
neglect columns x_j, $j \in J_4$ throughout the calculation (they could be omitted completely).
From now on all variables are required to be non-negative. The upper bounds are $c(x_j) = 2$ if $j \in J_1$, $c(x_j) = 1$ if $j \in J_2 + J_3$.

2. Remove the artificials from the basis in successive iterations using the described method for selecting pivot elements in a row (with rejection, if possible) and excluding columns with $j \in J_4$.

3. Apply the dual bounded-variables technique.

6.3 *The Absolute-Value Problem*

Dualizing (6.2.6) we obtain (see also theorem 8, section 2.5):

$$\text{Min}\Big\{ \sum_{j \in J_1} v_j^+ + \sum_{j \in J_1} v_j^- + \sum_{j \in J_2} v_j^+ + \sum_{j \in J_3} v_j^- \Big|$$

$$\Big| A_1^T u_1 + A_2^T u_2 + v^+ - v^- = p,\ u_1 \geqq 0,\ v^+ \geqq 0,\ v^- \geqq 0 \Big\}, \qquad (6.3.1)$$

or, writing $v_j = v_j^+ - v_j^-$, so that the v_j will not be restricted:

$$\text{Max}\Big\{ - \sum_{j \in J_1} |v_j| - \sum_{j \in J_2} \max(v_j, 0) + \sum_{j \in J_3} \min(v_j, 0) \Big|$$

$$\Big| A_1^T u_1 + A_2^T u_2 + v = p,\ u_1 \geqq 0 \Big\}. \qquad (6.3.2)$$

The vector u_1 will consist of variables u_{1i}, $i \in I_1$; the vector u_2 of variables u_{2i}, $i \in I_2$. We shall write $u(I_1) = \{u_i \mid i \in I_1\}$, i.e. the set of all u_{1i} variables. In the same way $u(I_2)$, $v(J_1)$, $v(J_2)$, $v(J_3)$ and $v(J_4)$ are defined. Note that the rows are now denoted by the index j and the columns by the index i. The main column will obtain the index r, the main row the index s. The elements of the matrix (A_1^T, A_2^T, E) will be denoted by $\overline{a}_{ji}$.

If J_2, J_3 and J_4 are empty, so that $J_1 = J$, this problem reduces to a problem in which the sum of the absolute values of the "slack" variables of a set of linear inequalities must be minimized. Therefore we shall call it an absolute-value problem.

For this problem a technique can be developed which is closely related to the dual simplex bounded-variables technique with rejection of pivot elements. Since the variables v_j are free we immediately have an initial basis: $v = p$. We can further calculate an initial d row by using the following coefficients in the objective function for the v_j.

$j \in J_1 : q(v_j) = -1$ if $p_j \geqq 0$ and $+1$ if $p_j < 0$;
$j \in J_2 : q(v_j) = -1$ if $p_j \geqq 0$ and $+0$ if $p_j < 0$;
$j \in J_3 : q(v_j) = -0$ if $p_j \geqq 0$ and $+1$ if $p_j < 0$;
$j \in J_4 : q(v_j) = 0$.

Hence, if $p_j \geqq 0$, v_j^+ is in the basis and, if $p_j < 0$, $-v_j^-$ is in it in accordance with (6.3.1).

For later use we also put $q(u_{1i}) = -0$ for all $i \in I_1$ and $= 0$ if $i \in I_2$ (sign unimportant in the latter case). We see that, if $p_j \neq 0$ and $j \notin J_4$, p_j and $q(v_j)$ will always have opposite signs (we distinguish between $+0$ and -0).

We shall introduce:

$$\delta(v_j) = 2 \text{ if } j \in J_1,\ \delta(v_j) = 1 \text{ if } j \in J_2 + J_3. \qquad (6.3.3)$$

Suppose that we have performed a number of iterations. The data obtained will be denoted by an asterisk. The technique will be such that the following two conditions are always satisfied:

1. The $d^*(v_j)$ of a non-basic variable v_j will be the $d^*(v_j^+)$. We then know that

$$d^*(v_j^-) = \delta(v_j) - d^*(v_j^+). \qquad (6.3.4)$$

2. The v_j variables in the basis will always be:
if $p^*(v_j) > 0$, a v_j^+ variable (coefficient $q(v_j^+)$ in the objective function),
if $p^*(v_j) < 0$, a $-v_j^-$ variable (coefficient $q(-v_j^-) = q(v_j^+) + \delta(v_j)$ in the objective function),
if $p^*(v_j) = 0$, either a v_j^+ or a $-v_j^-$ variable.

As main column we can now choose:
a. any non-basic u_{1i} column with $d^*(u_{1i}) < 0$ or
b. any non-basic u_{2i} column with $d^*(u_{2i}) \neq 0$ or
c. any non-basic v_j column with $d^*(v_j) < 0$ or
d. any non-basic v_j column with $d^*(v_j) > \delta(v_j)$.

If no column exists with this property the problem will be solved. We shall usually start with putting the u_{2i} in the basis in successive iterations. If all the u_{2i} are in the basis we shall continue with choosing the "best" column among the admitted ones. Let r be the number of the chosen main column and u_r^* be the variable entering the basis (this may be a v_j variable). If u_r^* is a v_j variable and $d^*(v_j) > \delta(v_j)$ we replace $d^*(v_j)$ by $\delta(v_j) - d^*(v_j)$. Consequently $-v_j^-$ will be introduced instead of v_j^+ ($u_r^* = -v_j^-$).

The choice of the pivot element in the pivot column will be such that:

1. The property that $\operatorname{sgn} q(v_j) = -\operatorname{sgn} p^*(v_j)$ for all v_j in the basis with $j \notin J_4$ and $p^*(v_j) \neq 0$ will be retained.

2. If a v_j variable will leave the basis it will always get a d-value between 0 and $\delta(v_j)$. If a v_j variable corresponding to a chosen pivot row does not satisfy this requirement, the pivot element will be rejected.

These two requirements lead to the following procedure:

1. If $u_r^* \in u(I_1)$, $v^+(J_1)$, $v^+(J_2)$ or $v^+(J_3)$,
or if $u_r^* \in u(I_2)$ and $d(u_r^*) \leq 0$:

determine s by

$$\frac{p_s^*}{\bar{a}_{sr}^*} = \min_j \left\{ \frac{p_j^*}{\bar{a}_{jr}^*} \middle| \bar{a}_{jr}^* \neq 0,\ \operatorname{sgn} q(v_j^*) = -\operatorname{sgn} \bar{a}_{jr}^*;\ v_j^* \notin u(I_2) + v(J_4) \right\};$$

if $-u_r^* \in v^-(J_1)$, $v^-(J_2)$ or $v^-(J_3)$,
or if $u_r^* \in u(I_2)$ and $d(u_r^*) > 0$:

determine s by

$$\frac{p_s^*}{-\bar{a}_{sr}^*} = \min_j \left\{ \frac{p_j^*}{-\bar{a}_{jr}^*} \middle| \bar{a}_{jr}^* \neq 0,\ \operatorname{sgn} q(v_j^*) = +\operatorname{sgn} \bar{a}_{jr}^*;\ v_j^* \notin u(I_2) + v(J_4) \right\}.$$

2. If $v_s^* \in u(I_1)$ we shall perform the transformation;
if $v_s^* \in v(J_1) + v(J_2) + v(J_3)$ we calculate $d_r^* + \delta(v_s^*) \mid a_{sr}^* \mid$:
if $d_r^* + \delta(v_s^*) \mid \bar{a}_{sr}^* \mid \geq 0$ we perform the transformation,
if $d_r^* + \delta(v_s^*) \mid \bar{a}_{sr}^* \mid < 0$:

a. $\bar{a}_{sr}^*$ will be rejected as a pivot element;

b. we replace $q(v_s^*)$ by $q(v_s^*) - \{\delta(v_s^*) + 0\} \operatorname{sgn} q(v_s^*)$
(the term +0 ensures that the new $q(v_s^*)$ will have the desired sign);

c. we replace d_i^* by $d_i^* + \delta(v_s^*).\bar{a}_{si}^*.\operatorname{sgn}\bar{a}_{sr}^*$;
d_0^* by $d_0^* + \delta(v_s^*).p_s^*.\operatorname{sgn}\bar{a}_{sr}^*$
(d_0^* is the value of the objective function);

d. we repeat steps 1 and 2.

A rejection of a pivot element will finally lead to a transformation with another pivot element and this transformation may change the sign of the p^* figure in the rejected row. This will always happen in the non-degenerate case. In the degenerate case such a p^* figure may become zero after the transformation. In the former case condition 1 will only be satisfied after the transformation if we change the $q^*(v_j)$ figure of the variable in the rejected row. In the latter case such a change does not do any harm. Therefore we perform operation 2b. This will result in a change of the whole d^* row. After the change the new $d(u_r^*)$ figure is still negative, so that a second pivot element can be chosen in that column. Since, for the rejected pivot elements, the signs of the q and p^* figures do not match any longer we shall automatically leave these variables out of consideration when choosing a second pivot element. Moreover the procedure guarantees that a v_j^+ variable will enter the basis at a non-negative level and a $-v_j^-$ variable at a non-positive level. It is clear that a pivot element in the main column will be found after a finite number of rejections (note that this problem will always have a maximum solution $\leqq 0$, so that infinite solutions cannot occur). The transformation will then lead to a new situation where condition 1 is satisfied. If a $-v_j^-$ variable leaves the basis we replace its d-value $-d(v_j^-)$ by $d(v_j^+)$ after the transformation, i.e. we add $\delta(v_j)$ to $-d(v_j^-)$. Consequently condition 2 will also be satisfied.

Proceeding in this way we see that we can solve (6.3.1) by means of the special technique developed for (6.3.2). This special technique is essentially the simplex method with some very simple iterations. With an anti-degeneracy precaution we are sure that the problem will be solved after a finite number of iterations.

Since (6.3.1) is dual to (6.2.6) we shall immediately, by theorems 7 and 8, section 2.5, have the solution of (6.2.6):

$$x_j = -d'(v_j^+) - q(v_j^+), \tag{6.3.5}$$

$$y_i = d'(u_{1i}), \tag{6.3.6}$$

(the prime indicates the optimum solution),

so that

$x_j = +1$ if v_j^+ in the final basis and $j \in J_1 + J_2$;
$= 0$ if v_j^+ in the final basis and $j \in J_3$;
$= 0$ if v_j^- in the final basis and $j \in J_2$;
$= -1$ if v_j^- in the final basis and $j \in J_1 + J_3$;
$= 0$ if $j \in J_4$.

The revised product-form algorithm can easily be applied to the solution of the absolute-value problem:
The operation "find pivot element in main column" will be different and more complicated but this is only a minor operation.
The changes in the d row if a pivot element has been rejected can be postponed and combined with the next "γ-operation". We only have to start this operation with a vector $\gamma_{s_\nu}^{\nu,1}$, defined by:
$\{\gamma_{s_\nu}^{\nu,1}\}_{s_\nu} = d^\nu(v_{s_\nu}^{\nu-1})$, where $v_{s_\nu}^{\nu-1}$ is the variable which ultimately left the basis in iteration $\nu-1$; if this is a v_j^- variable, then we mean the $-d^\nu(v_j^-)$ value (hence before we change it in $d^\nu(v_j^+)$),
$\{\gamma_{s_\nu}^{\nu,1}\}_j = \delta(v_j^{\nu-1}) \operatorname{sgn} \bar{a}_{jr_\nu}^{\nu-1}$ if $v_j^{\nu-1}$ is a rejected variable,
$= 0$ otherwise.
During the γ-operation the $d^\nu(v_j^+)$ figures are calculated, together with the $d^\nu(u_{1i})$ of the non-virgin u_{1i} variables. Note here that if a v_j variable leaves the basis its η-vector will always belong to the v_j^+ variable. The $d^\nu(v_j^-)$ can easily be derived from the $d^\nu(v_j^+)$.
From the foregoing it follows that iterations consisting of the rejection of a pivot element are very cheap ones. Almost their only consequence is that the next complete iteration will start with an initial vector in the γ-operation having more non-zero's.

The absolute-value problem will occur as sub-problem in some of the non-linear programming techniques to be described in chapters 7 - 11. A sequence of problems of this kind must then be solved and two consecutive problems only differ in one or more of the following respects:

1. The sets J_1, J_2, J_3 and J_4 change to $\tilde{J}_1$, $\tilde{J}_2$, $\tilde{J}_3$ and $\tilde{J}_4$ but we still have $\tilde{J}_1 + \tilde{J}_2 + \tilde{J}_3 + \tilde{J}_4 = J$ (the sets are always mutually exclusive).
2. The sets I_1 and I_2 may change to $\tilde{I}_1$ and $\tilde{I}_2$. Here we only have the following possible changes:
   a. $i \in I_2$ and $\in \tilde{I}_1$,
   b. $i \in I_1$ and $\notin \tilde{I}_1$ (only if u_{1i} is not in the final basis),
   c. $i \notin I_1 + I_2$ and $\in \tilde{I}_1$ ($d(u_i) < 0$ will then always hold for the new d figure),
   d. $i \notin I_1 + I_2$ and $\in \tilde{I}_2$.
3. The vector p may change to $\tilde{p}$.
4. For some values of i the vector $a_{.i}$ may change to $\tilde{a}_{.i}$.

We shall now investigate how we can use the final solution and set of η-vectors of one problem to start solving the next one with the help of the revised product-form algorithm. A prime will denote the final figures of the first problem.

1. These changes are simple; we introduce the new $q(v_j)$ and $\delta(v_j)$ figures. The coefficients in the objective function can now be substituted using the final vector p' instead of p, provided p does not change itself. Otherwise we first perform point 3.
A recalculation of the d row is necessary.

2. a. Make the changes in $\tilde{I}_1$ and $\tilde{I}_2$. Variable u_i will no longer be neglected as possible candidate for leaving the basis. If it is negative in the basis, $p'(u_i) < 0$, we shall give it a high penalty +M

in the objective function, $q(u_i) = +M$ (high enough to be sure that it will not stay in the basis at a negative value), introduce $\delta(u_i) = M$ and treat u_i in the same way as the v_j variables in the basis when choosing a pivot element. It will then either leave the basis in one of the following iterations after which we must delete the penalty or it will stay in the basis, loose its penalty (pivot element rejected) and get a non-negative p value.
A recalculation of the d row is necessary in this case.

b. We only have to omit the column $a_{.i}$ from the original matrix, and the $d(u_i)$ figure from the d row.

c, d. We have to add column $a_{.i}$ to the original matrix, to update it with the help of the η-vectors and to calculate its d figure.

3. We calculate $\tilde{p}'$ with the help of the η-vectors. Two things may happen now:
(i) $\tilde{p}'(u_{1i}) < 0$ for some i. We then give u_{1i} a penalty +M in the objective function, $q(u_{1i}) = +M$, introduce $\delta(u_{1i}) = M$ and treat u_{1i} in the same way as the v_j variables in the basis when choosing a pivot element.
(ii) $\operatorname{sgn} q(v_j) \cdot \tilde{p}'(v_j) \leqq 0$ does no longer hold for some j. We must then replace $q(v_j)$ by $q(v_j) - \{\delta(v_j) + 0\} \operatorname{sgn} q(v_j)$.
In both cases a recalculation of the d row is necessary.

4. If $a_{.i}$ is not in the final basis we can easily replace it in the original matrix by $\tilde{a}_{.i}$ and recalculate its d figure.

If $a_{.i}$ is in the final basis we add $\tilde{a}_{.i}$ to the original matrix, remove $a_{.i}$ from it, introduce variable $\tilde{u}_i$ and make u_i artificial (by giving it a penalty -M if $\tilde{p}(u_i) \geqq 0$ and +M if $\tilde{p}(u_i) < 0$; such a variable must be removed from the basis as soon as possible so that there is no rejection possibility here).

It is clear that too many changes of type 3 or 4 will make any restart with a final solution of a previous problem a hardly, if at all, profitable operation. Another possibility is to start the new problem with a pre-inversion as described in section 4.5. The variables to be taken for the pre-inversion can then follow from the previous problem. If even this pre-inversion does not seem to be profitable, hence if the two problems have hardly any resemblance, we can of course start the second problem in the conventional way.

7. METHODS OF FEASIBLE DIRECTIONS

7.1 *Introduction*

Let $F(x)$ be a concave function of the n-component vector $x \in E_n$, let $f_i(x)$ be convex functions of $x \in E_n$ and for $i \in I_C$, $I_C \subset I = \{1, \ldots, m\}$ being a set of integers; let a_i be n-component vectors for $i \in I_L = I - I_C$; let further c and p be n-component vectors and b be an m-component vector (some or all components of c may be infinite). Then we shall consider the problem:

$$\text{Max } \{F(x) \mid f_i(x) \leqq b_i,\ i \in I_C;\ a_i^T x \leqq b_i,\ i \in I_L;\ 0 \leqq x \leqq c\}. \tag{7.1.1}$$

The region R defined by the constraints in (7.1.1) will be convex. Therefore a problem of type (7.1.1) will be called a convex programming problem. We shall suppose that the non-linear constraints $f_i(x) \leqq b_i$ satisfy the regularity condition C 1 of section 2.6. Moreover we assume the functions $F(x)$ and $f_i(x)$ to be differentiable with continuous partial derivatives and shall define:

$$g(x) = \left\{\frac{\partial F}{\partial x_j},\ j = 1, \ldots, n\right\}; \tag{7.1.2}$$

$$q_i(x) = \left\{\frac{\partial f_i}{\partial x_j},\ j = 1, \ldots, n\right\},\ i \in I_C. \tag{7.1.3}$$

We shall also introduce non-negative variables y_i defined by

$$\begin{aligned} y_i = y_i(x) &= b_i - f_i(x) \quad \text{if } i \in I_C, \\ &= b_i - a_i^T x \quad \text{if } i \in I_L. \end{aligned} \tag{7.1.4}$$

Let m be the maximum of $F(x)$ on R. We shall assume that the concave function $F(x)$ is either unbounded on R ($m = \infty$) or that

$R_m = \{x \in R \mid F(x) = m\}$ is bounded. (condition C 3).

Theorem 1: If $F(x)$ satisfies condition C 3 on R and $m < \infty$, then

$R_\alpha = \{x \in R \mid F(x) \geqq \alpha\}$ will be bounded for all α.

Proof: $R_\alpha = 0$ if $\alpha > m$; $R_\alpha \supset R_m$ and convex if $\alpha \leqq m$ (section 2.4, theorem 8). Suppose R_α is unbounded for some α, so that, for any $x \in R_m$, a vector s can be found, such that $x + \lambda s \in R_\alpha$ for all $\lambda > 0$. Take λ_1 so large that $x_1 = x + \lambda_1 s \notin R_m$. Hence $F(x_1) < m = F(x)$, so that $g(x_1)^T(x - x_1) > 0$ by corollary 1 of theorem 3, section 2.4. Consequently $g(x_1)^T s < 0$ holds. Now $x + \lambda s = x_1 + (\lambda - \lambda_1)\, s$, $F(x + \lambda s) \leqq F(x_1) + (\lambda - \lambda_1) g(x_1)^T s < \alpha$ if λ is large enough. Hence $x + \lambda s \in R_\alpha$ for all $\lambda > 0$ cannot hold.

Corollary: If $m < \infty$, then, for any $y \in R$, the set $\{x \in R \mid F(x) \geqq F(y)\}$ will be bounded.

A method of solution for the convex programming problem (7.1.1) will be called a method of feasible directions if it has the following properties:

1. The starting point will be feasible: $x^0 \in R$.

2. Having obtained points $x^0, x^1, \ldots, x^k \in R$ satisfying $F(x^h) > F(x^{h-1})$ for $h = 1, \ldots, k$, a point $x^{k+1} \in R$ is determined with $F(x^{k+1}) \geq F(x^k)$ or it is concluded either that no better point x^{k+1} exists, so that x^k is a maximum solution of (7.1.1) or that $F(x)$ will be unbounded on R. The determination of x^{k+1} will consist of two parts:

a. In x^k a usable feasible direction s^k is determined (see section 2.6 for the definition of feasible and usable feasible directions).

b. A step length λ_k is determined, so that $x^{k+1} = x^k + \lambda_k s^k \in R$ and $F(x^{k+1}) > F(x^k)$.

3. The choice of the directions s^k and scalars λ_k, hence of the points $x^k \in R$ is such that convergence to the maximum m of $F(x)$ on R is obtained:

$$\lim_{k \to \infty} F(x^k) = m, \tag{7.1.5}$$

or that $F(x^k)$ is unbounded. In the latter case (7.1.1) will have an infinite solution.

Methods of feasible directions can be considered to be large-step gradient methods and differ from the small-step gradient methods, for instance those considered by Arrow and Hurwitz [1], chapters 6, 7 and 8. Many methods for solving linear or non-linear programming problems will appear to be methods of feasible directions. They only differ in the extra requirements for fixing the starting point x^0, the directions s^k or the step lengths λ_k. Among them are the simplex method, the primal-dual method, Rosen's gradient projection method [34], Beale's quadratic programming method, [3] and [4], and Frank and Wolfe's method for non-linear programming with linear constraints [20]. In section 7.2 we shall discuss the problem of obtaining an initial feasible point x^0. Section 7.3 and chapter 8 will be devoted to the different ways of obtaining usable feasible directions. The determination of the step lengths will be considered in section 7.4. In section 7.5 the procedures will be summarized. Their convergence will be studied in section 7.6. Finally in section 7.7 we shall discuss what will happen if the concavity assumption for $F(x)$ or the convexity assumptions for $f_i(x)$ are not fulfilled. Chapter 9 will be devoted to the linear programming problem, chapter 10 to the quadratic programming problem and chapter 11 to the problem of maximizing a concave function subject to either a set of linear constraints or to a set of non-linear and linear constraints (which determine a convex region). It will there be studied how the convergence of the procedures can be improved.

7.2 *Finding an Initial Feasible Solution*

In many practical problems a point $x^0 \in R$ will immediately follow from the prior knowledge of the problem. If an initial feasible point $x^0 \in R$ is not available, however, we can take any point x^0 satisfying $0 \leqq x^0 \leqq c$. We then introduce an extra variable ξ_1, and non-negative numbers ρ_i, $i \in I_L$ with $\rho_i = 0$ if $y_i(x^0) \geqq 0$ and $\rho_i > 0$ if $y_i(x) < 0$.

We first solve the problem:

$$\text{Max}\{-\xi_1 \mid a_i^T x - \rho_i \xi_1 \leqq b_i,\ i \in I_L;\ 0 \leqq x \leqq c;\ \xi_1 \geqq 0\}. \quad (7.2.1)$$

This is a linear programming problem which can be solved by the simplex method but also by any other method of feasible directions. An initial feasible point is then $x = x^0$, $\xi_1 = \max\left\{\frac{-y_i(x^0)}{\rho_i} \,\middle|\, \rho_i > 0\right\}$. These methods can all be devised in such a way that they are finite in the linear case (see chapter 9). If the region R is non-empty we shall obtain $\xi_1 = 0$ after a finite number of steps. The point x_1^0 which solves (7.2.1) will satisfy all the linear constraints in (7.1.1). If $y_i(x_1^0) \geqq 0$ for all $i \in I_C$ it follows that $x_1^0 \in R$, so that we have obtained an initial feasible point. If $y_i(x_1^0) < 0$ for some $i \in I_C$, we introduce an extra variable ξ_2 and non-negative numbers ρ_i, $i \in I_C$, with $\rho_i = 0$ if $y_i(x_1^0) \geqq 0$ and $\rho_i > 0$ if $y_i(x_1^0) < 0$, and solve the problem

$$\text{Max}\{-\xi_2 \mid f_i(x) - \rho_i \xi_2 \leqq b_i,\ i \in I_C;\ a_i^T x \leqq b_i,\ i \in I_L;\ 0 \leqq x \leqq c;\ \xi_2 \geqq 0\}. \quad (7.2.2)$$

An initial feasible solution for the problem (7.2.2) is easily obtained: $x = x_1^0$, $\xi_2 = \max\left\{\frac{-y_i(x_1^0)}{\rho_i} \,\middle|\, \rho_i > 0\right\}$. Since (7.2.2) is also a convex programming problem of the type (7.1.1) we can apply the same method to its solution.

Theorem 1: If R is non-empty and satisfies condition C 1, then any method of feasible directions applied to the solution of (7.2.2) will lead to $\xi_2 = 0$ after a finite number of iterations.

Proof: Condition C 1 will guarantee the existence of a vector x satisfying $f_i(x) < b_i$, $i \in I_C$; $a_i^T x \leqq b_i$, $i \in I_L$ and $0 \leqq x \leqq c$. Hence, if the condition $\xi_2 \geqq 0$ was not imposed in (7.2.2) we should be sure of convergence to a $\xi_2 < 0$, so that ξ_2 would pass zero after a finite number of iterations (property 3, section 7.1, of a method of feasible directions). Consequently with the condition $\xi_2 \geqq 0$ we shall obtain $\xi_2 = 0$ after a finite number of iterations. This proves the theorem.

Instead of solving (7.2.1) first and then (7.2.2) we could also solve the problem

$$\text{Max}\{-\xi \mid f_i(x) - \rho_i \xi \leqq b_i,\ i \in I_C;\ a_i^T x - \rho_i \xi \leqq b_i,\ i \in I_L;\ 0 \leqq x \leqq c;\ \xi \geqq 0\}, \quad (7.2.3)$$

where $\rho_i \geqq 0$ and > 0 if and only if $y_i(x^0) < 0$. For this problem we should have the initial solution $x = x^0$, $\xi = \max_i \left\{ \frac{-y_i(x^0)}{\rho_i} \middle| \rho_i > 0 \right\}$, so that we could immediately apply the method of feasible directions we have chosen. But, whereas (7.2.1) and (7.2.2) guarantee finiteness, this may not always be the case in (7.2.3).

It is also possible to solve the following problem:

$$\text{Max } \{F(x) - M\xi \mid f_i(x) - \rho_i \xi \leqq b_i,\ i \in I_C;\ a_i^T x - \rho_i \xi \leqq b_i,\ i \in I_L;\ 0 \leqq x \leqq c;\ \xi \geqq 0\}, \qquad (7.2.4)$$

where M is a large positive number. We have

Theorem 2: If (7.1.1) has a finite maximum x', $F(x') < \infty$, it follows that a number M_1 exists, so that for $M \geqq M_1$ problem (7.2.4) is also solved by x'.

Proof: Since x' is a maximum of (7.1.1) we know from section (2.6), theorem 6 and (2.6.6), that non-negative numbers u_i', v_{1j}' and v_{2j}' exist such that

$$g(x') = \Sigma u_i' q_i(x') + \Sigma u_i' a_i - \Sigma v_{1j}' e_j + \Sigma v_{2j}' e_j,$$
$$u'^T y' = 0,\ v_1'^T x' = 0,\ v_2'^T (c - x') = 0.$$

On the other hand problem (7.2.4) will be solved by x, y, ξ if this solution is feasible and the system

$$g(x) = \Sigma u_i q_i(x) + \Sigma u_i a_i - \Sigma v_{1j} e_j + \Sigma v_{2j} e_j,$$

$$-M = -\Sigma u_i \rho_i - v_\xi,$$

$$u^T y = 0,\ v_1^T x = 0,\ v_2^T (c - x) = 0,\ \xi v_\xi = 0,$$

$$u \geqq 0,\ v_1 \geqq 0,\ v_2 \geqq 0,\ v_\xi \geqq 0,$$

is consistent. This system will be solved by $x = x'$, $y = y'$, $\xi = 0$, $u = u'$, $v_1 = v_1'$, $v_2 = v_2'$ and $v_\xi = M - \Sigma u_i' \rho_i$, provided $M \geqq \Sigma u_i' \rho_i$ holds. Hence taking $M_1 = \Sigma u_i' \rho_i$ we have proved the theorem.

The two-phases approach will be preferred if $F(x)$ is a non-linear function since in the first phase we shall then have the simpler linear objective function, hence no recalculation of the gradient vector. The second procedure can for instance be applied if, as a consequence of rounding-off error, we have obtained a point x^k somewhere in the calculation which is slightly non-feasible.

Up to now we have said nothing about the choice of the ρ_i. A possible choice would be $\rho_i = 0$ if $y_i(x^0) \geqq 0$ and $\rho_i = 1$ if $y_i(x^0) < 0$. Another possibility is $\rho_i = 0$ if $y_i(x^0) \geqq 0$ and $\rho_i = -y_i(x^0)$ if $y_i(x^0) < 0$. In the latter case $x = x^0$, $\xi = 1$ will be a feasible solution of the modified problem and for this initial solution equality holds in all constraints violated by x^0 in the original problem. This may be of use when the ξ-procedure is used to correct rounding-off

errors. (With the choice $\rho_i = 1$ if $y_i(x^0) < 0$, the trial solution would no longer lie in some of the constraints in which it would presumably have been if there had not been rounding-off errors).

7.3 *Determination of Usable Feasible Directions*

Let for any $x \in R$ $I_C(x) \subset I_C$, $I_L(x) \subset I_L$, $J^-(x) \subset J$ and $J^+(x) \subset J$ be defined in such a way that

$$i \in I_C(x) \text{ if and only if } f_i(x) = b_i, \tag{7.3.1}$$

$$i \in I_L(x) \text{ if and only if } a_i^T x = b_i, \tag{7.3.2}$$

$$j \in J^-(x) \text{ if and only if } x_j = 0 \text{ and} \tag{7.3.3}$$

$$j \in J^+(x) \text{ if and only if } x_j = c_j. \tag{7.3.4}$$

Let further

$$S(x) = \{s \mid q_i(x)^T s \leqq 0,\ i \in I_C(x);\ a_i^T s \leqq 0,\ i \in I_L(x);\ s_j \geqq 0,\ j \in J^-(x);\ s_j \leqq 0,\ j \in J^+(x)\}. \tag{7.3.5}$$

Let σ be an extra variable and let Θ_i be positive numbers, then we define:

$$S'(x) = \{(s, \sigma) \mid q_i(x)^T s + \Theta_i \sigma \leqq 0,\ i \in I_C(x);\ a_i^T s \leqq 0,\ i \in I_L(x);\ s_j \geqq 0,\ j \in J^-(x);\ s_j \leqq 0,\ j \in J^+(x)\}. \tag{7.3.6}$$

We shall now look for a vector (s, σ) which satisfies the requirements:

1. $(s, \sigma) \in S'(x)$, (7.3.7)

2. $-g(x)^T s + \sigma \leqq 0$, (7.3.8)

3. normalization requirement, (7.3.9)

4. σ to be maximized. (7.3.10)

The problem defined by these four requirements will be called a direction-finding problem. Any (s, σ), satisfying (7.3.7) with $\sigma > 0$ will be a usable feasible direction since, if $\sigma > 0$, we shall have $q_i(x)^T s < 0$ if $i \in I_C(x)$, and $g(x)^T s > 0$. By means of the normalization requirement we can prevent a method for solving the direction-finding problem from producing an infinite solution. The normalization is such that if (s, σ) satisfies (7.3.7) and (7.3.8), then there is a $\bar{\beta} > 0$, so that for $0 \leqq \beta \leqq \bar{\beta}$ the vector $(\beta s, \beta \sigma)$ will satisfy (7.3.7 - 9) and that, if s_1 and s_2 are normalized vectors, $\lambda s_1 + (1 - \lambda) s_2$ will also satisfy the normalization requirement for all λ, $0 \leqq \lambda \leqq 1$.

If $\sigma' = 0$ for the optimum solution s', σ' of (7.3.7 - 10), it follows that the system of linear inequalities, given by (7.3.7), (7.3.8) and $\sigma > 0$ will be inconsistent, so that, by theorem 3, section 2.2, there will exist non-negative quantities u_i, v_j^-, v_j^+ and u_0, such that:

$$\sum_{i \in I_C(x)} u_i q_i(x) + \sum_{i \in I_L(x)} u_i a_i - \sum_{j \in J^-(x)} v_j^- e_j + \sum_{j \in J^+(x)} v_j^+ e_j +$$

$$- u_0 g(x) = 0, \qquad (7.3.11)$$

$$\sum_{i \in I_C(x)} u_i \Theta_i + u_0 = 1. \qquad (7.3.12)$$

If $u_0 > 0$ it follows from (7.3.11) and theorem 6, section (2.6) that x is a maximum of F(x) on R.

If $u_0 = 0$, (7.3.12) tells us that $u_i > 0$ will hold for at least one $i \in I_C(x)$, so that from (7.3.11) it follows that condition C 1 would not be satisfied. Hence with condition C 1 we know that we have arrived at the optimum of F(x) on R as soon as $\sigma \leqq 0$ holds for all (s, σ) satisfying (7.3.7) and (7.3.8).

Hence we have proved:

Theorem 1: Under condition C 1 a point $x \in R$ will be a maximum of F(x) on R if and only if $\sigma \leqq 0$ holds for all $(s, \sigma) \in S'(x)$ satisfying $-g(x)^T s + \sigma \leqq 0$.

The choice of the Θ_i may be arbitrary. We can for instance take $\Theta_i = 1$ for all $i \in I_C(x)$ but we can also increase Θ_i when we arrive in a certain hypersurface again and again.

If $I_C = 0$, hence if there are no non-linear constraints, then the direction-finding problem reduces to:

1. $s \in S(x)$, (7.3.13)

2. normalization requirement, (7.3.14)

3. $g(x)^T s$ to be maximized. (7.3.15)

Any s satisfying (7.3.13) with $g(x)^T s > 0$ will be a usable feasible direction. If, for the maximum vector s', we have $g(x)^T s' = 0$, x will be a maximum of F(x) on R.

The requirement (7.3.10) or (7.3.15) will lead to the best usable feasible direction among those which satisfy the normalization requirement. It is an astonishing fact that many methods for solving the linear, quadratic or convex programming problem, although they seem to be quite different at first sight, only differ in the way they fix the directions s by a normalization requirement and in the computational consequences of such a fixation.

Examples of possible normalizations are:

$$
\begin{aligned}
&\text{N 1 :} && s^T s \leqq 1; \\
&\text{N 2 :} && -1 \leqq s_j \leqq 1 \text{ for all } j; \\
&\text{N 3 :} && s_j \leqq 1 && \text{if } g_j(x) > 0, \\
& && s_j \geqq -1 && \text{if } g_j(x) < 0; \\
&\text{N 4 :} && \sigma \leqq 1 && \text{in (7.3.7 - 10)}, \\
& && g(x)^T s \leqq 1 && \text{in (7.3.13 - 15)}; \\
&\text{N 5 :} && q_i(x)^T s + \Theta_i \sigma \leqq y_i(x) && \text{if } i \in I_C, \\
& && a_i^T s \leqq y_i(x) && \text{if } i \in I_L, \\
& && -x_j \leqq s_j \leqq c_j - x_j && \text{if } j \in J.
\end{aligned}
$$

For the convergence of the procedures to be studied in section 7.6 it will be necessary that the directions s^k are bounded. This will always be the case if procedure N 1 or N 2 is used. In the other cases we can for instance require that the absolute largest component of s^k may not exceed a given number and, if it does, we can consider a vector αs^k, where $\alpha < 1$ is such that the absolute largest component is at the required bound. The normalizations N 1 - N 5 will be studied in chapter 8.

Normalization N 5 differs from the other ones in that it also takes into account the constraints in which the trial solution x does not lie. Let us fix the point $x \in R$ by putting a bar over it and let us write $s = x - \bar{x}$ ($\bar{x}$ fixed and x variable) and $\sigma = x_0$. Then (7.3.7), (7.3.8), N 5 and (7.3.10) lead to the following problem:

$$
\begin{aligned}
\text{Max} \{ x_0 \mid & \bar{q}_i^T x + \Theta_i x_0 \leqq \bar{q}_i^T \bar{x} + b_i - f_i(\bar{x}),\ i \in I_C;\ a_i^T x \leqq b_i,\ i \in I_L\ ; \\
& 0 \leqq x_j \leqq c_j,\ j \in J;\ -\bar{g}^T x + x_0 \leqq -\bar{g}^T \bar{x} \} ,
\end{aligned}
\qquad (7.3.16)
$$

where we have written $\bar{q}_i$ and $\bar{g}$ for $q_i(\bar{x})$ and $g(\bar{x})$. Since (7.3.16) can still have an infinite solution, normalization N 5 is not a normalization in the proper sense of the word. In the case of an infinite solution of (7.3.16) we shall derive the usable vector s from the extreme ray of the constraint set which led to the infinite solution and which can easily be obtained from the final data of (7.3.16). See section 3.2, the paragraph following formula (3.2.5).

7.4 *Determination of the Length of the Steps*

Having found a usable feasible direction s in x, we want to improve the trial solution x by making a move in the direction s. Defining

$$
\lambda' = \max \{ \lambda \mid x + \lambda s \in R \}, \qquad (7.4.1)
$$

we are sure that the length λ of the move must be $\leqq \lambda'$. ($\lambda' = \infty$ is possible if some of the c_j are ∞). Let λ'' be the smallest λ such that

$$F(x + \lambda'' s) = \max_{\lambda} F(x + \lambda s), \tag{7.4.2}$$

hence

$$\lambda'' = \max \{ \lambda \mid F(x + \mu s) < F(x + \lambda s) \text{ for all } \mu < \lambda \},$$

then $0 < \lambda'' \leqq \infty$ will hold and we shall choose

$$\lambda = \min (\lambda', \lambda''), \tag{7.4.3}$$

so that

$$F(x + \lambda s) = \text{Max} \{ F(x + \mu s) \mid \mu \leqq \lambda' \}. \tag{7.4.4}$$

If $\lambda = \infty$, then regularity condition C 3 for $F(x)$ ensures that $m = \infty$ will hold, so that there is no finite maximum. For, if $m < \infty$, then $\{ y \in R \mid F(y) \geqq F(x) \}$ is bounded, so that $\lambda = \infty$ cannot hold.

Problem (7.4.4) is one-dimensional and can easily be solved if all the constraints are linear and if the objective function is linear or quadratic. In the general case the problems of finding λ' and λ'' are of the same type: In (7.4.1) we must, for each i, find the largest root of the equation $f_i(x + \lambda s) = b_i$, and take then the smallest of the figures obtained. For each $i \in I_C$ this can for instance be done by Newton's method (note that λ is the only variable in the equation; x and s are known vectors). For the linear constraints we simply have

$$\lambda' = \min (\lambda_1', \lambda_2', \lambda_3'), \tag{7.4.5}$$

with

$$\lambda_1' = \min_i \left\{ \frac{y_i(x)}{-t_i} \;\middle|\; t_i < 0,\ i \in I_L - I_L(x) \right\}, \tag{7.4.6}$$

where

$$t_i = - a_i^T s ; \tag{7.4.7}$$

$$\lambda_2' = \min_j \left\{ \frac{x_j}{-s_j} \;\middle|\; s_j < 0,\ j \in J - J^-(x) \right\}; \tag{7.4.8}$$

$$\lambda_3' = \min_j \left\{ \frac{c_j - x_j}{s_j} \;\middle|\; s_j > 0,\ j \in J - J^+(x) \right\}. \tag{7.4.9}$$

To find λ we can proceed as follows:
if $g(x + \lambda' s)^T s \geqq 0$, $\lambda = \lambda'$ will hold ($F(x)$ is concave);
if $g(x + \lambda' s)^T s < 0$, then we can apply the regula falsi to find the point λ'' where $g(x + \lambda'' s)^T s = 0$ will hold (λ is the only variable, x and s are known vectors).

7.5 *Procedures*

In this section we shall summarize the procedures which are based on the different normalizations. Procedure P 1 will use one

of the normalizations N 1 - N 4; procedure P 2 will be based on normalization N 5. In order to ensure convergence of $F(x^k)$ to $m = \max\{F(x) \mid x \in R\}$ it is necessary in procedure P 1 to apply a so-called anti-zigzagging precaution. Without such a precaution examples can be constructed in which $\lim F(x^k) < m$ would hold. Examples of such precautions are:

AZ 1:

a. Let $\varepsilon > 0$ be an arbitrary number.

b. Define for $x \in R$:

$$I_C(x, \varepsilon) = \{i \in I_C \mid b_i - \varepsilon \leqq f_i(x) \leqq b_i\}, \tag{7.5.1}$$

$$I_L(x, \varepsilon) = \{i \in I_L \mid b_i - \varepsilon \leqq a_i^T x \leqq b_i\}, \tag{7.5.2}$$

$$J^-(x, \varepsilon) = \{j \in J \mid 0 \leqq x_j \leqq \varepsilon\}, \tag{7.5.3}$$

$$J^+(x, \varepsilon) = \{j \in J \mid c_j - \varepsilon \leqq x_j \leqq c_j\}, \tag{7.5.4}$$

$$H(x, \varepsilon) = I_C(x, \varepsilon) + I_L(x, \varepsilon) + J^-(x, \varepsilon) + J^+(x, \varepsilon), \tag{7.5.5}$$

$$H(x) = H(x, 0). \tag{7.5.6}$$

We see that for $\varepsilon = 0$ we have
$I_C(x, 0) = I_C(x)$, $I_L(x, 0) = I_L(x)$, $J^-(x, 0) = J^-(x)$ and $J^+(x, 0) = J^+(x)$.

c. Define

$$S'(x, \varepsilon) = \{(s, \sigma) \mid q_i(x)^T s + \Theta_i \sigma \leqq 0,\ i \in I_C(x, \varepsilon);$$
$$a_i^T s \leqq 0,\ i \in I_L(x, \varepsilon);\ s_j \geqq 0,\ j \in J^-(x, \varepsilon);$$
$$s_j \leqq 0,\ j \in J^+(x, \varepsilon)\}. \tag{7.5.7}$$

d. Replace $S'(x)$ in (7.3.7), hence in the direction-finding problems by $S'(x, \varepsilon)$.

e. If for the final bounded vector s_{opt} of a direction-finding problem σ_{opt} is less than ε, then we replace ε by $\frac{\varepsilon}{2}$, hence $S'(x, \varepsilon)$ by $S'(x, \frac{\varepsilon}{2})$ and solve the direction-finding problem again. (If $\sigma_{opt} = 0$ and $S'(x, \varepsilon) = S'(x)$, then we know already that x is a maximum of $F(x)$ on R).

AZ 2:

a. Let $\varepsilon > 0$ be an arbitrary number.

b. Let x^k have just been determined, so that we have to solve the k-th direction-finding problem; let $h \in H(x^k)$ be such that $h \in H(x^k)$, $h \notin H(x^{k-1})$ and $h \in H(x^\ell)$ for some $\ell \leqq k-2$. For such values of h we shall retain the requirement $n_h^T s + \Theta_h \sigma \leqq 0$ in all direction-finding problems (7.3.7-10) or (7.3.13-15) following the k-th one until condition c. is met. Hence even if $h \notin H(x^{k+1})$ we nevertheless require $n_h^T s + \Theta_h \sigma \leqq 0$ in the (k+1) -th direction-finding problem (n_h is the outward pointing normal in x^k of the h-th hypersurface, so that $n_h = q_i(x^k)$, a_i, $-e_j$ or e_j; $\Theta_h = 0$ if $h \notin I_C$).

c. These extra requirements will be omitted in the next direction-finding problem if $\lambda_k < \lambda_k'$ has to be chosen; they will be omitted in the direction-finding problem itself if $\sigma_{opt} < \varepsilon$ for the final bounded vector s_{opt} of it, so that we then solve the problem again without extra requirements in it. Moreover we replace ε by $\frac{\varepsilon}{2}$ for the new direction-finding problems.

In AZ 1 we also take into account hypersurfaces for which the variable $y_i(x)$, x_j or $c_j - x_j$, though non-zero, is less than ε, so that we try to prevent making very small steps by the extra requirements (ε may be considered as a small number which is moreover gradually reduced, so that it will not hinder us in the neighbourhood of the true optimum). In AZ 2 we add an extra requirement if we arrive in a hypersurface for the second time, so that we try to avoid a third arrival in it. In AZ 2 we may have an extra requirement $n_h^T s + \Theta_h \sigma \leqq 0$ even though we are rather far from the corresponding hypersurface. This danger will be reduced by the rule that the extra requirement will be omitted if $\lambda_k < \lambda_k'$ has to be chosen. Precaution AZ 1 is the simplest one but AZ 2 seems to be somewhat more effective since we shall sooner be forced to leave a certain combination of hypersurfaces if it is not the correct one.

For future use we moreover formulate in the case of only linear constraints:

AZ 3:

a. Require $n_h^T s = 0$ instead of $\leqq 0$ if we arrive for the second time in a certain constraining hyperplane. Retain this requirement until

b. $g(x^k)^T s_{opt} = 0$ holds in some direction-finding problem. We then replace at least one of the requirements $n_h^T s = 0$ by $n_h^T s \leqq 0$.

Precaution AZ 3 will be used in the quadratic programming procedures to be discussed in chapter 10; its main function is to make the method finite. In chapter 11 we shall show how it can also be used to speed up convergence in the case of a general convex programming problem. It can then be combined with AZ 1.

Summarizing we have the following two convex programming procedures:

Procedure P 1:

1. Start with $x^0 \in R$. See section 7.2 if such a point is not immediately available. Take $\varepsilon > 0$ small but arbitrarily and $\varepsilon' > 0$ very small.

2. Suppose $x^0, x^1, \ldots, x^k$ have already been determined. Solve now the k-th direction-finding problem:

a. $(s, \sigma) \in S'(x^k, \varepsilon)$, or $(s, \sigma) \in S'(x)$, together with anti-zig-zagging requirements as used in AZ 2;

b. normalization N1, N2, N3 or N4;

c. $-g(x^k)^T s + \sigma \leqq 0$;

d. σ to be maximized.

3. If for the bounded maximum solution $\sigma_k' < \varepsilon$ holds, replace ε by $\frac{\varepsilon}{2}$, $S'(x^k, \varepsilon)$ by $S'(x^k, \frac{\varepsilon}{2})$ in AZ 1, omit the extra requirements in AZ 2, and solve the direction-finding problem again;

if, however, $\sigma_k' = 0$ and if $S'(x^k, \varepsilon) = S'(x^k)$ in AZ 1, or if there are no extra requirements in AZ 2, stop, since the problem has been solved (assuming R satisfies C 1; if not, we add small quantities Δb_i to the b_i for those i, $i \in I_C$, where $u_i > 0$ held in (7.3.11), see section 2.6);
if $\sigma_k' \geqq \varepsilon$ holds, perform 4.

4. Determine λ_k, using (7.4.1) and (7.4.4). If $\lambda_k = \infty$, stop; F(x) is unbounded on R by condition C 3. If $\lambda_k < \lambda_k'$ and AZ 2 is applied, omit all anti-zigzagging requirements in the next direction-finding problem.

5. Determine

$$x^{k+1} = x^k + \lambda_k s^k, \tag{7.5.8}$$

and

$$F(x^{k+1}).$$

6. If $F(x^{k+1}) - F(x^k) < \varepsilon'$ and the k-th direction-finding problem did not contain anti-zigzagging requirements, stop. As a check on the near-optimality we could now solve the linear programming problem given by the right-hand member of (2.6.7). This will show us what might be the maximum possible further increase of F(x). If this is sufficiently small we can be sure that x^k is near-optimal. In all other cases we shall repeat steps 2-5.

Procedure P 2:

1. Start with $x^0 \in R$. Let ε' be a predetermined small number.

2. Suppose $x^0, x^1, \ldots, x^k$ have already been determined. Solve the linear programming sub-problem.

$$\text{Max}\{x_0 \mid q_i(x^k)^T x + \Theta_i x_0 \leqq q_i(x^k)^T x^k + b_i - f_i(x^k),\ i \in I_C;$$

$$a_i^T x \leqq b_i,\ i \in I_L;\ 0 \leqq x_j \leqq c_j,\ j \in J;$$

$$-g(x^k)^T x + x_0 \leqq -g(x^k)^T x^k\}.$$

3. If $x_{0\,\max} = 0$, stop, x^k will be a maximum solution;
if $x_{0\,\max} > 0$, but $< \infty$, let $s^k = x_{\max} - x^k$;
if $x_{0\,\max} = \infty$, derive s^k from the extreme ray of the feasible region of the sub-problem which leads to infinity.

4. Determine λ_k, using (7.4.1) and (7.4.4). If $\lambda_k = \infty$, stop; F(x) is unbounded on R by condition C 3.

5. Determine

$$x^{k+1} = x^k + \lambda_k s^k, \tag{7.5.8}$$

and

$$F(x^{k+1}).$$

6. If $F(x^{k+1}) - F(x^k) < \varepsilon'$, stop. The linearized problem (2.6.7) may show us whether x^{k+1} is a near-optimal solution or not.
If $F(x^{k+1}) - F(x^k) \geqq \varepsilon'$, repeat steps 2-5.

It is realized that the linearized problem (2.6.7) may have an infinite or very large solution whereas x^{k+1} is nevertheless near-optimal. This is not very satisfactory. In chapter 11 we shall investigate how this check on near-optimally could be improved. Moreover some new procedures will be described in this chapter which are derived from P 1 and P 2 but converge much faster in most cases.

7.6 *Convergence of the Procedures*

We shall first prove three lemmas:

Lemma 1: If $x \in R$ then either x is a maximum of F(x) on R or there exists a scalar $\delta > 0$ and a vector $s \in S(x)$ such that $q_i(x)^T s \leqq \leqq -\Theta_i \delta$ for all $i \in I_C(x)$, and $g(x)^T s \geqq \delta$.

Proof: Suppose x is not a maximum of F(x) on R. From theorem 1, section 7.3, we then know that the direction-finding problem (7.3.7-10) will lead to an optimum solution with $\sigma_{max} > 0$. Hence we can take $\delta = \sigma_{max}$ and $s = s_{max}$.

Lemma 2: Let F(x) be an arbitrary function with continuous gradient vector g(x), defined on a closed convex region R.
Let $x^k \in R$, $k = 0, 1, 2, \ldots$ be a bounded sequence of points, let s^k, $k = 0, 1, 2, \ldots$ be a bounded sequence of vectors and λ_k a sequence of scalars such that $x^{k+1} = x^k + \lambda_k s^k$ holds for all k and that $g(x^k + \lambda s^k)^T s^k > 0$ holds for $0 \leqq \lambda < \lambda_k$. Suppose that for some $\delta > 0$ and sub-sequences $y^\ell = x^{k\ell}$, $t^\ell = s^{k\ell}$ and $\mu_\ell = \lambda_{k_\ell}$ we have that $g(y^\ell)^T t^\ell \geqq \delta$ for all ℓ, then $\Sigma \mu_\ell$ will be convergent.

Proof: Define $\bar{\mu}_\ell = \max \{\mu \leqq \mu_\ell \mid g(y^\ell + \mu t^\ell)^T t^\ell \geqq \frac{\delta}{2}\}$. We have that for each $p > 0$:

$$F(x^{k_p+1}) - F(x^0) = \sum_{h=0}^{k_p} \{F(x^{h+1}) - F(x^h)\} \quad \geqq \quad \sum_{\ell=0}^{p} \{F(x^{k\ell+1}) - F(y^\ell)\} \geqq$$

$$\geqq \sum_{\ell=0}^{p} \{F(y^\ell + \bar{\mu}_\ell t^\ell) - F(y^\ell)\} \geqq \frac{\delta}{2} \sum_{\ell=0}^{p} \bar{\mu}_\ell,$$

so that it follows from the boundedness of F(x) that $\Sigma \bar{\mu}_\ell$ converges. If $\bar{\mu}_\ell < \mu_\ell$ we shall have that $g(y^\ell + \bar{\mu}_\ell t^\ell)^T t^\ell = \frac{\delta}{2}$. If $\Sigma \mu_\ell$ diverges this will hold an infinite number of times. Let us only consider these values of ℓ ($\ell \in L_1$). The sequence y^ℓ is bounded, hence has a point of accumulation y'. It follows that, for some sub-sequence of the points y^ℓ, ($\ell \in L_2 \subset L_1$) we have that $\bar{\mu}_\ell < \mu_\ell$, $\lim y^\ell = y'$, so that scalars δ_1, δ_2 and ℓ_1 exist such that $\frac{\delta}{2} < \delta_1 < \delta_2 < \delta$ and, for $\ell \in L_2$ and $\geqq \ell_1$, $g(y')^T t^\ell \geqq \delta_2$ and $g(y' + \bar{\mu}_\ell t^\ell)^T t^\ell \leqq \delta_1$ hold. The sequence t^ℓ, $\ell \in L_2$ is also bounded, hence some sub-sequence of it converges to some vector t', say the sequence with $\ell \in L_3 \subset L_2$ ($t^\ell = t'$ a finite or infinite number of times is also possible). It follows that scalars δ_3, δ_4 and $\ell_2 \geqq \ell_1$ exist such that $\delta_1 < \delta_3 < \delta_4 < \delta_2$

and, for $\ell \in L_3$ and $\geqq \ell_2$, $g(y')^T t' \geqq \delta_4$ and $g(y' + \bar{\mu}_\ell t')^T t' \leqq \delta_3$. But this is impossible by virtue of the continuity of $g(x)$ and the convergence of $\Sigma \bar{\mu}_\ell$. Consequently $\Sigma \mu_\ell$ must be convergent which proves the theorem.

Corollary: If s^k, $k = 0, 1, 2, \ldots$ is a bounded sequence of usable vectors, if the sequence of points $x^k \in R$ is bounded and if, for some $\delta > 0$ and all k, $g(x^k)^T s^k \geqq \delta$ holds, then $\lambda_k = \lambda_k'' < \lambda_k'$ will only hold a finite number of times.
Proof: Since the conditions of lemma 2 are fulfilled $\Sigma \lambda_k$ will converge, so that $\lambda_k = \bar{\lambda}_k$ will always hold for k large enough ($\bar{\lambda}_k$ corresponds to the $\bar{\mu}$ of the proof of lemma 2). Hence $g(x^k + \lambda_k s^k)^T s^k \geqq \frac{\delta}{2} > 0$ if k large enough, so that $\lambda_k = \lambda_k'' < \lambda_k'$ cannot hold.

Lemma 3: Let $F(x)$ be an arbitrary function with continuous gradient vector $g(x)$; let $\{x^k \mid k \in K\}$ be a finite or bounded infinite set of points. Suppose a scalar $\delta > 0$ can be found so that for each $k \in K$ a bounded vector $s^k = s(x^k)$ exists such that $g(x^k)^T s^k \geqq \delta$. If, for $k \in K$, $\lambda_k = \max \{\lambda \mid F(x^k + \mu s^k) > F(x^k)$ for all $\mu < \lambda\}$, then $\inf_k \lambda_k > 0$.

Proof: From the definition it follows that λ_k will be either infinite or the first value of λ for which $F(x^k + \lambda s^k)$ equals $F(x^k)$. Let $\mu_k = \max \{\mu \mid g(x^k + \mu s^k)^T s^k \geqq \frac{\delta}{2}$ for all $\mu \leqq \mu_k\}$, then $F(x^k + \mu_k s^k) - F(x^k) \geqq \mu_k \frac{\delta}{2}$, so that $\lambda_k \geqq \mu_k$ holds. Suppose a sequence of points exists such that μ_k tends to zero. For this sequence we then have $g(x^k)^T s^k \geqq \delta$ and $g(x^k + \mu_k s^k)^T s^k = \frac{\delta}{2}$ for all k. In the same way as in the proof of lemma 2 it can be shown, however, that this is impossible. Hence $\inf_k \mu_k > 0$, so that $\inf_k \lambda_k > 0$ holds.

Corollary: If $f_i(x^k) \leqq b_i$, $q_i(x^k)^T s^k < -\Theta_i \delta$ for all k, s^k bounded, $x^{k+1} = x^k + \lambda_k s^k$ and $\Sigma \lambda_k$ convergent, then an index k_1 exists such that $f_i(x^k) < b_i$ for $k \geqq k_1$.
Proof: Take $F(x) = -f_i(x)$, $g(x) = -q_i(x)$ and $\delta = \Theta_i \delta$ and apply lemma 3.

We now prove:
Theorem 1: Procedure P 1 with one of the normalizations N 1 - N 5 for the directions, AZ 1 or AZ 2 and $\varepsilon' = 0$ will generate a sequence of points x^k, such that $\lim_{k \to \infty} F(x^k) = m = \max \{F(x) \mid x \in R\}$, provided R and $F(x)$ satisfy conditions C 1 and C 3, respectively.
Proof: If the total number of steps is finite and the final solution is finite the theorem is obviously true; if a stop is obtained for an infinite solution, ($\lambda_k = \infty$ for some k), then $m = \infty$ by condition C 3 and theorem 1, section 7.1. Suppose therefore that the number of steps is infinite. If the sequence of points x^k is unbounded, then condition C 3 will ensure that $\lim F(x^k) = \infty$, so that the theorem also holds in that case. Suppose therefore that the sequence x^k, hence also the sequence $F(x^k)$ is bounded. Let (s^k, σ^k) be the

bounded sequence of vectors proportional to the optimal solutions of the corresponding direction-finding problems (anti-zigzagging requirements, if any, included). If σ^k, hence $g(x^k)^T s^k \geqq \delta$ holds for some $\delta > 0$ and all k (a finite number excluded perhaps), then we can apply lemma 1: If λ_k is such that $x^{k+1} = x^k + \lambda_k s^k$, then $\Sigma\lambda_k$ will be convergent. Let $\bar{x} = \lim x^k$ and define, for $\bar{\varepsilon} > 0$ $R(\bar{x}, \bar{\varepsilon}) = \{x \in R \mid |x_j - \bar{x}_j| \leqq \bar{\varepsilon}$ for all j; $|y_i(x) - y_i(\bar{x})| \leqq \bar{\varepsilon}$ for all i$\}$ Take $\bar{\varepsilon}$ so small that for any $x \in R(\bar{x}, \bar{\varepsilon})$ we have that $H(x) \subset H(\bar{x})$, i.e. no $x \in R(\bar{x}, \bar{\varepsilon})$ will be in a constraining hypersurface in which $\bar{x}$ does not lie. The relation $\sigma^k \geqq \delta$ will entail that after a finite number of steps the ε of AZ 1 or AZ 2 will no longer be decreased. We take $k(\bar{\varepsilon})$ so large that, for $k \geqq k(\bar{\varepsilon})$, ε is no longer decreased, that $x^k \in R(\bar{x}, \bar{\varepsilon})$, and that $\lambda_k = \lambda_k'' < \lambda_k'$ does not hold any longer (possible by the corollary of lemma 2). When using AZ 1 we shall also take $\bar{\varepsilon} < \varepsilon$, the final ε of AZ 1 which is no longer decreased. We then have for $k \geqq k(\bar{\varepsilon})$ that $H(x^k, \varepsilon) \supset H(\bar{x})$, so that we can apply the corollary of lemma 3 which shows that $f_i(x^k) < b_i$ for all $i \in I_C(\bar{x}) \subset I_C(x^k, \varepsilon)$ if k is large enough. Hence we can not arrive in such a non-linear constraining hypersurface, nor can we return in a linear constraining hyperplane, owing to AZ 1, if we have left it, nor can we obtain a $\lambda_k < \lambda_k'$. We have thus arrived at an impossibility. With AZ 2 and $k \geqq k(\bar{\varepsilon})$ the argument is that, if k is large enough, we can not return in a constraining hypersurface once we have been in it twice, so that we shall also have no choice after a finite number of steps which leads to the same impossibility. Consequently σ^k will become arbitrarily small and at regular times we shall replace ε by $\frac{\varepsilon}{2}$. Let $y^\ell = x^{k_\ell}$ be the sub-sequence of the sequence x^k for which this holds and let $\bar{y}$ be a point of accumulation of the sequence y^ℓ. By taking an appropriate sub-sequence, if necessary, we can assume that the sequence y^ℓ converges to $\bar{y}$. Let $t^\ell = s^{k_\ell}$ and $\mu_\ell = \lambda_{k_\ell}$. If $F(\bar{y}) = m$, the theorem will hold. Hence suppose $F(\bar{y}) < m$ holds, so that by lemma 1 a scalar $\delta > 0$ and a feasible vector $\bar{s}$ in $\bar{y}$ exist such that

$q_i(\bar{y})^T \bar{s} \leqq -\theta_i \delta$ if $i \in I_C(\bar{y})$, $a_i^T \bar{s} \leqq 0$ if $i \in I_L(\bar{y})$,

$\bar{s}_j \geqq 0$ if $j \in J^-(\bar{y})$, $\bar{s}_j \leqq 0$ if $j \in J^+(\bar{y})$, $\bar{s}$ bounded and $g(\bar{y})^T \bar{s} \geqq \delta$.

It follows that an $\bar{\varepsilon} > 0$ can be found so that for all $x \in R(\bar{y}, \bar{\varepsilon})$:

1. $H(x) \subset H(\bar{y})$,
2. $q_i(x)^T \bar{s} \leqq -\theta_i \frac{\delta}{2}$ if $i \in I_C(\bar{y})$, and
3. $g(x)^T \bar{s} \geqq \frac{\delta}{2}$.

Let ℓ_1 be so large that $y^\ell \in R(\bar{y}, \bar{\varepsilon})$ for $\ell \geqq \ell_1$ and that $H(y^\ell, \varepsilon) = H(\bar{y})$ if AZ 1 is applied (this is possible since ε is gradually reduced). Then $g(y^\ell)^T t^\ell \geqq \frac{\delta}{2}$ if $\ell \geqq \ell_1$, so that by lemma 2 $\Sigma \mu_\ell$ will be convergent, hence $\ell_2 \geqq \ell_1$ exists so that $x^{k_{\ell+1}} \in R(\bar{y}, \bar{\varepsilon})$ holds for $\ell \geqq \ell_2$. Consequently $g(x^{k_{\ell+1}})^T s^{k_{\ell+1}} \geqq \frac{\delta}{2}$ from which we derive that $x^{k_{\ell+2}} \in R(\bar{y}, \bar{\varepsilon})$ for $\ell \geqq \ell_3 \geqq \ell_2$, etc. From the corollary of lemma 2 it is clear that the points $x^{k_{\ell+h}} \in R(\bar{y}, \bar{\varepsilon})$, $h = 1, 2, \ldots$ are always obtained by hitting a new constraining hypersurface from the set determining $\bar{y}$. But AZ 1 or AZ 2, $g(x^{k_{\ell+h}})^T s^{k_{\ell+h}} \geqq \frac{\delta}{2}$ and the

corollary of lemma 3 will cause such a hyperplane to be no longer available after a finite number of steps, so that we have arrived at a contradiction. The assumption $F(\bar{y}) < m$ was wrong, which proves the theorem.

Without anti-zigzagging precaution we could not have proved that $\sigma^k \geqq \delta$ for all k and F(x) bounded is impossible. Since N 5 is a normalization like the other ones we have also proved the convergence of P 2 with AZ 1 or AZ 2. In P 2, however, it is not necessary to have an anti-zigzagging precaution since we have:
Theorem 2: If R and F(x) satisfy conditions C 1 and C 3 respectively and if $\varepsilon' = 0$, procedure P 2 will generate a sequence of points $x^k \in R$ with $\lim F(x^k) = m$.
Proof: If the sequence x^k is finite the theorem is obviously true, if it is not finite and unbounded $\lim F(x^k) = \infty$ will hold by virtue of condition C 3. Hence suppose the sequence x^k is infinite and bounded. Let $\bar{y}$ be a point of accumulation of the sequence and $y^\ell = x^{k_\ell}$ be a sub-sequence converging to $\bar{y}$. Suppose $F(\bar{y}) < m$, so that (7.3.16) will have a solution $x_0 = \delta > 0$ in $\bar{y}$. Consequently $\bar{\varepsilon} > 0$ exists so that for all $y \in R(\bar{y}, \bar{\varepsilon})$ problem (7.3.16) will have a solution $x_0 \geqq \frac{\delta}{2}$ in y. Choose $\bar{\varepsilon} \leqq \frac{\Theta_i \delta}{4}$ so that, for all $y \in R(\bar{y}, \bar{\varepsilon})$, we have $b_i - f_i(y) \leqq \frac{\Theta_i \delta}{4}$ for all $i \in I_C(\bar{y})$. Choose $\ell(\bar{\varepsilon})$ so large that, for $\ell \geqq \ell(\bar{\varepsilon})$, $y^\ell \in R(\bar{y}, \bar{\varepsilon})$ holds. From (7.3.16) it then follows that $q_i(y^\ell)^T (x_{opt} - y^\ell) \leqq -\Theta_i \frac{\delta}{4}$ holds for $\ell \geqq \ell(\bar{\varepsilon})$ and $i \in I_C(\bar{y})$. From $x_0(y^\ell) \geqq \frac{\delta}{2}$ we can derive that the series $\Sigma \lambda_{k_\ell}$ is convergent since otherwise $F(x^k)$ would not be bounded. Hence λ_{k_ℓ} will tend to zero and $\lambda_{k_\ell} = \lambda^!_{k_\ell}$ will always hold for ℓ large enough. After a finite number of steps we can only hit non-linear constraining hypersurfaces of the set determining $\bar{y}$ but $q_i(y^\ell)^T (x_{opt} - y^\ell) \leqq -\Theta_i \frac{\delta}{4}$ if $i \in I_C(\bar{y})$ and the corollary of lemma 3 will make this impossible. Hence $F(\bar{y}) = m$ holds, which proves the theorem.

7.7 *Non-linear Programming without Convexity Assumptions*

Many mathematical programming problems are of the type in which a differentiable function F(x) with continuous partial derivatives has to be maximized in a closed connected region R, which is then given by a set of linear and non-linear inequalities. The question arises what will happen if a method of feasible directions is applied to such a "continuous" non-linear programming problem. This is an important question since in most practical problems we shall not know beforehand whether F(x) is concave or $f_i(x)$ convex. Hence let the problem be:

$$\text{Max } \{F(x) \mid f_i(x) \leqq b_i,\ i \in I_1;\ a_i^T x \leqq b_i,\ i \in I_2;\ 0 \leqq x \leqq c\}, \tag{7.7.1}$$

with $F(x)$ and $f_i(x)$ differentiable with continuous partial derivatives satisfying some regularity conditions like
C 1: for all $x \in R$ there is an $s \in S(x)$ such that $q_i(x)^T s < 0$ if $i \in I_1(x)$,
C 3: $R_\alpha = \{x \in R \mid F(x) = \alpha\}$ is bounded for each finite α.
The first problem is that there may now be more local maxima and that in the best case we shall only find one of these local maxima. There are no general methods at the moment which can deal with this difficulty. In some practical problems the danger of arriving at a local optimum can be reduced by using either the prior knowledge of the problem or many different starting points.

A second difficulty is that it is no longer true that any point x with the property that $g(x)^T s \leqq 0$ for all $s \in S(x)$ is a (local) maximum of $F(x)$ on R. Such a point, to be called a stationary point, can also be a local minimum or local saddle-point. The first possibility can only occur in the starting point x^0 if $g(x^0) = 0$ and is therefore of no practical importance. The second possibility is also rather unlikely to occur in practical problems but cannot be excluded. It is clear that, if x is such a local saddle-point, there will exist at least one $s \in S'(x)$ with $g(x)^T s = 0$ and $s \neq 0$ and it is perhaps possible to explore these directions.

Finally it should be remarked that the choice of λ can be either

$$\lambda = \text{Max}\,\{\lambda \leqq \lambda' \mid F(x + \lambda s) > F(x + \mu s) \text{ for all } \mu < \lambda\},$$

or

$$\lambda = \text{Max}\,\{\lambda \leqq \lambda' \mid g(x + \mu s)^T s \geqq 0 \text{ for all } \mu \leqq \lambda, \text{ and } > 0 \text{ for all } \mu,$$
$$0 \leqq \lambda_1 \leqq \mu < \lambda\},$$

where λ_1 is some scalar $< \lambda$.
The first choice will lead to the maximum of $F(x + \lambda s)$ under the condition $\lambda \leqq \lambda'$, the second choice will give the first (local) maximum of $F(x + \lambda s)$ under the condition $\lambda \leqq \lambda'$ and may be preferred if it is believed that x is already near to the maximum. It can easily be proved that any point of accumulation of procedure P 1 with AZ 1 or AZ 2 and one of the normalizations N 1 - N 5 or of procedure P 2 will be a stationary point. Actually the proofs of section 7.6 do not change at all since lemmas 1 - 3 hold for general functions $F(x)$ and $f_i(x)$.

It can thus be concluded that a method of feasible directions can be applied equally well to more general mathematical programming problems of the "continuous" type but that we shall in general not know whether the solution obtained is the global maximum or not. If the problem satisfies the regularity conditions and the requirements that $R'_\alpha = \{x \mid x \in R,\ F(x) \geqq \alpha\}$ is closed and connected for each α and that $g(x)^T s > 0$ if $g(x + \lambda s)^T s > 0$ for all λ, $0 < \lambda < \bar{\lambda}$, where $\bar{\lambda}$ is some positive number, then it is clear that a method of feasible directions will lead to a global maximum of $F(x)$ on R.

8. NORMALIZATIONS OF THE FEASIBLE DIRECTIONS

8.1 *Introduction*

In this chapter we shall study the different normalizations N 1 - N 5 of the feasible directions which have been mentioned in section 7.3. Normalization N 1 will require most attention. The computational features of N 3 do not differ very much from those of N 2. Normalization N 4 seems to be the simplest one, whereas N 5 leads to a different procedure. Section 8.5 will be devoted to a brief comparison of the various possible normalizations.

8.2 *Normalization N 1*

Let $x \in R$ and $Q = Q(x)$ be a matrix with rows $q_i(x)$, $i \in I_C(x)$; a_i, $i \in I_L(x)$; $-e_j$, $j \in J^-(x)$; e_j, $j \in J^+(x)$ and possibly having some more rows due to anti-zigzagging requirements. Then the direction-finding problem is of the type:

$$\text{Max } \{\sigma \mid Qs + \sigma\,\Theta \leqq 0,\ -g^T s + \sigma \leqq 0,\ s^T s \leqq 1\}, \qquad (8.2.1)$$

where Θ is a vector with components $\Theta_i \geqq 0$, and with $\Theta_i > 0$ if the corresponding row comes from a non-linear constraint, and $\Theta_i = 0$ if the corresponding row comes from a linear constraint.

We have:

Theorem 1: (8.2.1) has an optimum solution s_1, σ_1 proportional to the optimum solution s_2, σ_2 of the problem

$$\text{Max } \{\sigma \mid Qs + \sigma\,\Theta \leqq 0,\ -g^T s + \sigma \leqq 0,\ s^T s + \sigma^2 \leqq 1\}. \qquad (8.2.2)$$

Proof: It is clear that $\frac{s_1}{\sqrt{1+\sigma_1^2}}$, $\frac{\sigma_1}{\sqrt{1+\sigma_1^2}}$ is a feasible solution of (8.2.2) and that $\frac{s_2}{\sqrt{1-\sigma_2^2}}$, $\frac{\sigma_2}{\sqrt{1-\sigma_2^2}}$ is a feasible solution of (8.2.1). Hence, if $\sigma_1 = 0$, then $\sigma_2 = 0$ and conversely so that in that case $s_1 = 0$, $\sigma_1 = 0$ and $s_2 = 0$, $\sigma_2 = 0$ are proportional optimum solutions.

$\sigma_1 \geqq \frac{\sigma_2}{\sqrt{1-\sigma_2^2}}$ and $\sigma_2 \geqq \frac{\sigma_1}{\sqrt{1+\sigma_1^2}}$, hence $\sigma_1^2 \geqq \frac{\sigma_2^2}{1-\sigma_2^2}$ or $\frac{\sigma_1^2}{1+\sigma_1^2} \geqq \sigma_2^2 \geqq \frac{\sigma_1^2}{1+\sigma_1^2}$

It follows that $\sigma_2 = \frac{\sigma_1}{\sqrt{1+\sigma_1^2}}$. This proves the theorem.

This theorem shows that (8.2.1) as well as (7.3.13-15) with N 1 are of the type

$$\text{Max}\{p^Tx \mid Px \leqq 0,\ x^Tx \leqq 1\}, \tag{8.2.3}$$

where P is a matrix with m rows. This problem will be further studied. It is a convex programming problem itself with a linear objective function and one quadratic constraint. As long as $p^Tx > 0$ holds for a vector x satisfying $Px \leqq 0$ and $x^Tx \leqq 1$, we shall have $(x')^Tx' = 1$ for the vector x' which solves (8.2.3). This solution will be the vector of unit length from the cone $\{x \mid Px \leqq 0\}$ which makes the smallest angle with the vector p, hence it will be proportional to the projection of ponto this cone. This suggests

Theorem 2: If $p^Tx' > 0$ then the optimum solution x' is unique: Proof: Suppose x'' is another optimum solution. Let $x = \frac{1}{2}(x' + x'')$, then $p^Tx = p^Tx'$, $Px \leqq 0$ and $x^Tx < 1$ so that $\lambda > 1$ would exist such that λx is still feasible. But $\lambda\ p^Tx > p^Tx' = p^Tx''$, so that neither x' nor x'' would be optimal.

Theorem 3: If x' solves (8.2.3) and $p^Tx' > 0$, then for some $\lambda > 0$ $\lambda x'$ will solve the problem

$$\text{Min}\{x^Tx \mid Px \leqq 0,\ p^Tx = 1\}. \tag{8.2.4}$$

Proof: Since $Px' \leqq 0$ and $p^Tx' > 0$, (8.2.4) will have feasible solutions. Let λ be such that $\lambda\ p^Tx' = 1$. Suppose x'' solves (8.2.4), $(x'')^Tx'' = \alpha^2 \leqq \lambda^2 (x')^Tx' = \lambda^2$. If $\alpha < \lambda$, then $\frac{x''}{\alpha}$ would be feasible in (8.2.3) and $\frac{p^Tx''}{\alpha} > \frac{p^Tx''}{\lambda} = p^Tx'$ would hold, so that x' would not be optimal in (8.2.3).

From this theorem it follows that any standard method for solving quadratic programming problems could be applied to solve (8.2.3). But for the special problem (8.2.3) we recommend the method which will now be described:

Let us introduce:

$$y = -Px, \text{ hence } y \geqq 0. \tag{8.2.5}$$

Suppose x' solves (8.2.3). From (2.6.6) it then follows that a vector u' and a scalar β exist such that

$$p = P^Tu' + \beta x', \quad u' \geqq 0,\ \beta \geqq 0,\ (u')^Ty' = 0. \tag{8.2.6}$$

Multiplying (8.2.6) by the matrix -P and introducing

$$v' = -\beta Px' = \beta y' \geqq 0, \tag{8.2.7}$$

we obtain that u' and v' satisfy the following relations:

$$-PP^Tu + v = -P p, \tag{8.2.8}$$

$$u \geqq 0,\ v \geqq 0,\ u^Tv = 0. \tag{8.2.9}$$

On the other hand, if $\beta \not> 0$, then any solution u, v of (8.2.8), (8.2.9) will by virtue of (8.2.6) and (8.2.7) lead to vectors x and y which solve (8.2.3). If $\beta = 0$ it follows from (8.2.6) that $p^T x \leqq 0$ for any x satisfying $Px \leqq 0$, so that there is no feasible x with $p^T x > 0$.

Our method will consist of the following steps:

1. Form and solve (8.2.8), (8.2.9).
2. Calculate β x by means of (8.2.6) (normalizing is not necessary since we are not interested in normalized but in bounded directions. Therefore we only have to prevent that the absolute largest component of β x exceeds a given bound).

For solving (8.2.8) and (8.2.9) we shall start with the basic solution $v = -Pp$, $u = 0$ and choose a $v_i < 0$. If such a $v_i < 0$ does not exist we have a u and a v satisfying (8.2.8) and (8.2.9). But if $v_r < 0$, for instance, we replace v_r by u_r in the basis. This will entail a transformation of the matrix by means of transformation formulae equivalent to those used in the simplex or dual simplex method. In general if after ν iterations variables v_i^ν are in the basis and variables u_i^ν in the non-basis (v_i^ν or u_i^ν may be a v_i- or a u_i-variable) and if $v_i^\nu < 0$ for some values of i, then we choose one of the corresponding rows and replace v_i^ν in the basis by u_i^ν. This procedure will be justified by the following theorems:

Theorem 4: The diagonal elements of the transformed non-basis matrix are always non-positive.

Proof: The basis B of the ν-th iteration is a square submatrix of the matrix $(-PP^T, E)$. Now let $PP^T = \begin{pmatrix} R_1 & R_3 \\ R_3^T & R_2 \end{pmatrix}$ with R_1 and R_2 square and symmetric, R_3 reactangular of appropriate size and R_1 positive definite (R_1 and R_2 are always positive semi-definite by theorems 4 and 1 of section 2.3) and suppose $B = \begin{pmatrix} -R_1 & O \\ -R_3^T & E_2 \end{pmatrix}$, so that

$$B^{-1} = \begin{pmatrix} -R_1^{-1} & O \\ -R_3^T R_1^{-1} & E_2 \end{pmatrix} \quad \text{and } B^{-1}(-PP^T, E) =$$

$$= \begin{pmatrix} -R_1^{-1} & O \\ -R_3^T R_1^{-1} & E_2 \end{pmatrix} \cdot \begin{pmatrix} -R_1 & -R_3 & E_1 & O \\ -R_3^T & -R_2 & O & E_2 \end{pmatrix} =$$

$$= \begin{pmatrix} E_1 & R_1^{-1} R_3 & -R_1^{-1} & O \\ O & R_3^T R_1^{-1} R_3 - R_2 & -R_3^T R_1^{-1} & E_2 \end{pmatrix},$$

where E_1 and E_2 are unit matrices of appropriate size with $E = \begin{pmatrix} E_1 & O \\ O & E_2 \end{pmatrix}$.

The diagonal elements of the transformed non-basic matrix are the diagonal elements of $-R_1^{-1}$ and $(-R_3^T R_1^{-1} R_3 + R_2)$. These matrices, however, are negative semi-definite by theorems 3 and 6, section 2.3, so that they have non-positive diagonal elements ($-R_1^{-1}$ is of course negative definite).

Theorem 5: If the i-th diagonal element of the transformed non-basic matrix is zero, then the corresponding element in the right-hand member will be zero.

Proof: $-R_1^{-1}$ will always have negative diagonal elements since it is negative definite. Suppose $e_r^T (-R_3^T R_1^{-1} R_3 + R_2) e_r = 0$ for some r, i.e. suppose the r-th diagonal element of the bracketed matrix vanishes. Let the matrix P be partitioned row-wise in two matrices P_1 and P_2 so that $P_1 P_1^T = R_1$, $P_2 P_2^T = R_2$ and $P_1 P_2^T = R_3$. Hence $e_r^T(-P_2 P_1^T R_1^{-1} P_1 P_2^T + P_2 P_2^T) e_r = 0$ or $e_r^T P_2 (-P_1^T R_1^{-1} P_1 + E_2) P_2^T e_r = 0$. The bracketed matrix is symmetric and idempotent, hence positive semi-definite by theorem 5, section 2.3, so that, by theorem 2, section 2.3 we have that $(-P_1^T R_1^{-1} P_1 + E_2) P_2^T e_r = 0$, hence that $e_r^T (R_3^T R_1^{-1} P_1 p - P_2 p) = 0$. This is exactly the element of the right-hand member involved since

$$-B^{-1}P\,p = -\begin{pmatrix} -R_1^{-1} & O \\ -R_3^T R_1^{-1} & E_2 \end{pmatrix} \begin{pmatrix} P_1 p \\ P_2 p \end{pmatrix} = \begin{pmatrix} R_1^{-1} P_1 p \\ R_3^T R_1^{-1} P_1 p - P_2 p \end{pmatrix}$$

This proves the theorem.

From theorems 4 and 5 it follows that, if $v_i^\nu < 0$ holds, we can always replace v_i^ν by u_i^ν, and $u_i^{\nu+1} > 0$ will hold since the corresponding diagonal element is negative. Note that the requirement $u^T v = 0$ is always satisfied in this way. As soon as we obtain $v_i^\nu \geqq 0$ for all i, the optimum solution will therefore be obtained.

The last question is how the choice of the main row from the rows with $v_i^\nu < 0$ should be made, so that the method is finite. We shall indicate two possible choices based on the anti-degeneracy proposals of Charnes [7] and Dantzig, Orden and Wolfe [15] respectively, which will lead to finite methods but do not seem to be very efficient. After the proof of finiteness we shall then indicate two other possible choices which may perhaps not always lead to finite methods but seem to be more efficient in practical problems. Let $A = -PP^T$ have columns $a_{\cdot i}$, i = 1, ..., m, and let $a_{\cdot i}^\nu = \{B^\nu\}^{-1} a_{\cdot i}$ (this may be a unit column). Let $b_{\cdot i}^\nu = \{B^\nu\}^{-1} e_i$ (this may be a non-unit column) and let $p^\nu = \{B^\nu\}^{-1}(-Pp)$, hence $p^0 = -Pp$. We shall consider the following two ways of selecting a $v_i^\nu < 0$, if any:

1. Let $I_0^\nu = \{ i \mid p_i^\nu < 0 \}$. If I_0^ν is empty, the problem will be solved; if it consists of one element r, take the r-th row as main row; otherwise consider

$$I_1^\nu = \left\{ i_1 \in I_0^\nu \;\middle|\; \frac{b_{i_1 1}^\nu}{-p_{i_1}^\nu} = \min_{i \in I_0^\nu} \frac{b_{i1}^\nu}{-p_i^\nu} \right\}.$$

Suppose I_0^ν, I_1^ν, ..., I_{h-1}^ν have already been chosen. If I_{h-1}^ν consists of one element we take the corresponding row as main row; if it consists of more elements we define:

$$I_h^\nu = \{ i_h \in I_{h-1}^\nu \Big| \frac{b_{i_h h}^\nu}{-p_{i_h}^\nu} = \min_{i \in I_{h-1}^\nu} \frac{b_{i\,h}^\nu}{-p_i^\nu} \}. \qquad (8.2.10)$$

This procedure will be continued until a unique choice is obtained. This will happen for some $h \leqq m$ since no two rows of the inverse can be dependent.

2. The definition of the sets I_h^ν is:

$$I_0^\nu = \{ i \Big| p_i^\nu < 0 \};$$

$$I_h^\nu = \{ i_h \in I_{h-1}^\nu \Big| \frac{p_{i_h h}^\nu}{-p_{i_h}^\nu} = \min_{i \in I_{h-1}^\nu} \frac{p_{i\,h}^\nu}{-p_i^\nu} \}, \qquad (8.2.11)$$

where $p_{\cdot h}^\nu = a_{\cdot h}^\nu$ if $h \leqq m$, and $p_{\cdot h}^\nu = b_{\cdot h-m}^\nu$ if $h > m$.

Theorem 6. If in each iteration the variable $v_{r_{\nu+1}}^\nu$, to be replaced by $u_{r_{\nu+1}}^\nu$ in the basis, is determined in one of the described ways, then a finite number of replacements will suffice to solve (8.2.8) and (8.2.9).

Proof: Let us solve the quadratic programming problem

$$\mathrm{Max} \{ p^T P^T u - \tfrac{1}{2} u^T P P^T u \Big| u \geqq 0 \} \qquad (8.2.12)$$

by applying Wolfe's simplex method for quadratic programming [39], so that we must consider the problem

$$\mathrm{Max} \{ \lambda \Big| -PP^T u + \lambda\, Pp + v = 0,\ u \geqq 0,\ v \geqq 0,\ \lambda \leqq 1,\ u^T v = 0 \}. \qquad (8.2.13)$$

Apart from the relation $u^T v = 0$ this is a linear programming problem for which the initial basic feasible solution $v = 0$, $\bar{\lambda} = 1 - \lambda = 1$; $u = 0$, $\lambda = 0$ is immediately available and satisfies $u^T v = 0$. Wolfe uses the simplex method for solving (8.2.13) with an extra exclusion rule interdicting, for each i, u_i (v_i) to enter the basis if v_i (u_i) is already there. This exclusion rule will limit the choice of pivot columns. Nevertheless the procedure will lead to the optimum solution $\lambda = 1$ in a finite number of steps provided an anti-degeneracy precaution is taken. The corresponding vectors u and v will then solve (8.2.8) and (8.2.9). In this special problem the variable λ will enter the basis in the first iteration. The slack variable $\bar{\lambda}$ will stay in the basis until the last iteration. As soon at it leaves the basis λ will assume the value 1 and the problem will be solved. As long as $\bar{\lambda}$ is in the basis λ will stay there at zero level. Until the last iteration all variables in the basis except $\bar{\lambda}$, hence also the variable belonging to the main row, will have zero values. Therefore an anti-degeneracy procedure has to

be applied for which we choose the second one of section 3.2, due to Dantzig, Orden and Wolfe [15]. Let the pivot column be $\bar{p}_{\cdot s}^{\nu}$, i.e. one of the columns $\bar{p}_{\cdot j}^{\nu}$ of the transformed matrix $(-PP^T, E)$. For the selection of the pivot element we introduce sets $I_0^{\nu} \supset I_1^{\nu} \supset \ldots \supset I_h^{\nu}$ by

$$I_0^{\nu} = \{i \mid \bar{p}_{is}^{\nu} > 0\};\ I_h^{\nu} = \left\{ i_h \in I_{h-1}^{\nu} \;\middle|\; \frac{\bar{b}_{i_h h}^{\nu}}{\bar{p}_{i_h s}^{\nu}} = \min_{i \in I_{h-1}^{\nu}} \frac{\bar{b}_{ih}^{\nu}}{\bar{p}_{is}^{\nu}} \right\},$$

where $\bar{b}_{\cdot h}^{\nu} = \bar{p}_{\cdot m+h}^{\nu}$, $h = 1, 2, \ldots$. If I_0^{ν} is empty $\bar{\lambda}$ will leave the basis. If it is non-empty we shall obtain a set I_h^{ν} consisting of only one element for some $h \leqq m$, so that the choice of the pivot element is always unique. After the first iteration we shall always have bases consisting of the variables λ, $\bar{\lambda}$ and $m-1$ variables u_i or v_i, whereas there will be $m+1$ non-basic variables u_i or v_i. From this and the exclusion rule it follows that for exactly one r_ν, u_{r_ν} and v_{r_ν} will both be in the non-basis at the end of the ν-th iteration, $\nu = 1, 2, \ldots$. One of these variables will have left the basis in iteration ν, the other one will enter the basis in iteration $\nu+1$, since the exclusion rule will give us no other choice. Now suppose that we have performed ν iterations of our method for solving (8.2.8) and (8.2.9) using (8.2.10) for the choice of the sets determining the main rows. Let us consider minus the right-hand member p^{ν} as an extra column of the non-basic matrix and let us assign a variable λ to it and carry out an artificial iteration with $-p_{r_\nu}^{\nu}$ as pivot element, so that the variable u_{r_ν} or v_{r_ν} which just entered the basis will leave it again whereas λ will enter it. We then obtain a tableau with u_{r_ν} and v_{r_ν} both in the non-basis and λ at place r_ν in the basis. Our point is that this tableau is exactly the same as the tableau after the ν-th iteration of Wolfe's method apart from the missing row associated with $\bar{\lambda}$. If $\tilde{a}_{ij}^{\nu}$, $\tilde{b}_{ij}^{\nu}$ are the elements of the new tableau, then

$$\tilde{a}_{ij}^{\nu} = a_{ij}^{\nu} - a_{r_\nu j}^{\nu} \frac{p_i^{\nu}}{p_{r_\nu}^{\nu}},\ i \neq r_\nu;\ \tilde{a}_{r_\nu j}^{\nu} = -\frac{a_{r_\nu j}^{\nu}}{p_{r_\nu}^{\nu}};$$

$$\tilde{b}_{ij}^{\nu} = b_{ij}^{\nu} - b_{r_\nu j}^{\nu} \frac{p_i^{\nu}}{p_{r_\nu}^{\nu}},\ i \neq r_\nu;\ \tilde{b}_{r_\nu j}^{\nu} = -\frac{b_{r_\nu j}^{\nu}}{p_{r_\nu}^{\nu}}$$

and we must prove that $\bar{a}_{ij}^{\nu} = \tilde{a}_{ij}^{\nu}$ and $\bar{b}_{ij}^{\nu} = \tilde{b}_{ij}^{\nu}$. This will be the case if the same variables are in the basis in both tableaus, hence if the choice of the variables $v_{r_\nu}^{\nu-1}$ to be removed from the basis is the same in both methods. In the first iteration in Wolfe's method λ enters the basis and r_1 is determined by sets

$$I_0^0 = \{i \mid p_i^0 < 0\} \text{ and } I_h^0 = \left\{ i_h \in I_{h-1}^0 \;\middle|\; \frac{(e_h)_{i_h}}{-p_{i_h}^0} = \min_{i \in I_{h-1}^0} \frac{(e_h)_i}{-p_i^0} \right\}.$$

In our method this choice is exactly the same, so that v_{r_1} is the same in both methods and $\bar{a}_{ij}^{1} = \tilde{a}_{ij}^{1}$, $\bar{b}_{ij}^{1} = \tilde{b}_{ij}^{1}$ will hold. Now suppose that $\bar{a}_{ij}^{\nu} = \tilde{a}_{ij}^{\nu}$, $\bar{b}_{ij}^{\nu} = \tilde{b}_{ij}^{\nu}$, so that for the same value r_ν, u_{r_ν} and v_{r_ν} will be in the non-basis in both tableaus. In our method u_{r_ν} and v_{r_ν} will have interchanged in the ν-th iteration; in Wolfe's method $u_{r_\nu}(v_{r_\nu})$ will enter the basis in iteration $\nu+1$ if v_{r_ν} (u_{r_ν}) has left it in iteration ν. In Wolfe's method $r_{\nu+1}$ is then determined with the help of sets

$$I_0^{\nu} = \{\, i \mid \bar{p}_{is_{\nu+1}}^{\nu} > 0 \,\}$$

($\bar{p}_{is_{\nu+1}}^{\nu}$ is either $\bar{a}_{ir_\nu}^{\nu}$ or $\bar{b}_{ir_\nu}^{\nu}$, dependent on whether u_{r_ν} enters or v_{r_ν});

$$I_h^{\nu} = \left\{ i_h \in I_{h-1}^{\nu} \;\middle|\; \frac{\bar{b}_{i_h h}^{\nu}}{\bar{p}_{i_h s_{\nu+1}}^{\nu}} = \min_{i \in I_{h-1}^{\nu}} \frac{\bar{b}_{ih}^{\nu}}{\bar{p}_{is_{\nu+1}}^{\nu}} \right\}.$$

In our method these sets are

$$I_0^{\nu} = \{ i \mid p_i^{\nu} < 0 \};\quad I_h^{\nu} = \left\{ i_h \in I_{h-1}^{\nu} \;\middle|\; \frac{b_{i_h h}^{\nu}}{-p_{i_h}^{\nu}} = \min_{i \in I_{h-1}^{\nu}} \frac{b_{ih}^{\nu}}{-p_i^{\nu}} \right\}.$$

But $\bar{b}_{ij}^{\nu} = \tilde{b}_{ij}^{\nu} = b_{ij}^{\nu} - b_{r_\nu j}^{\nu} \dfrac{p_i^{\nu}}{p_{r_\nu}^{\nu}}$ $(i \neq r_\nu)$ and $\bar{p}_{is_{\nu+1}}^{\nu} = \tilde{p}_{ir_\nu}^{\nu} = -\dfrac{p_i^{\nu}}{p_{r_\nu}^{\nu}}$ ($\tilde{p}_{ir_\nu}^{\nu} = \tilde{a}_{ir_\nu}^{\nu}$ or $\tilde{b}_{ir_\nu}^{\nu}$),

so that

$$\min_i \frac{\bar{b}_{ih}^{\nu}}{\bar{p}_{is_{\nu+1}}^{\nu}} = \min_i \frac{b_{ih}^{\nu} - b_{r_\nu h}^{\nu} \dfrac{p_i^{\nu}}{p_{r_\nu}^{\nu}}}{-\dfrac{p_i^{\nu}}{p_{r_\nu}^{\nu}}} = p_{r_\nu}^{\nu} \min_i \frac{b_{ih}^{\nu}}{-p_i^{\nu}} + b_{r_\nu h}^{\nu}.$$

Since $p_{r_\nu}^{\nu} > 0$ always holds an $\bar{p}_{is_{\nu+1}}^{\nu} = -\dfrac{p_i^{\nu}}{p_{r_\nu}^{\nu}}$ it follows that the sets I_h^{ν} are the same in boths cases, so that the choice of $r_{\nu+1}$ will be the same and $\bar{a}_{ij}^{\nu+1} = \tilde{a}_{ij}^{\nu+1}$, $\bar{b}_{ij}^{\nu+1} = \tilde{b}_{ij}^{\nu+1}$ will hold for all $\nu \geqq 0$. Consequently in our method the same variable will be removed from the basis in each iteration as in Wolfe's method, so that the finiteness of our method follows from the finiteness of Wolfe's method. This proves theorem 6 in the case that anti-de-

generacy precaution (8.2.10) is applied. Anti-degeneracy precaution (8.2.11) is equivalent to Charnes' procedure for breaking ties applied to Wolfe's method for solving (8.2.12).

In some direction-finding problems we shall require $y_i = 0$ in (8.2.3) for some values of i (see chapters 9, 10 and 11). In that case $v_i = 0$ and u_i unrestricted will hold for these values of i in (8.2.8), (8.2.9). We can then start with the elimination of all these v_i from the basis in successive iterations after which these rows will never be chosen as pivot row.

We shall now consider some computational aspects of normalization N 1:

1. Although by theorem 6 we can choose the $v_{r_{\nu+1}}^{\nu}$ variables such that a finite method arises, the proposed choice does not seem to be the best one in practical cases. Other possibilities are:

a. $\min_i \{ p_i^{\nu} \mid p_i^{\nu} < 0 \}$;

b. $\min_i \{ \frac{p_i^{\nu}}{-p_{ii}^{\nu}} \mid p_i^{\nu} < 0 \}$,

where p_{ii}^{ν} is the diagonal element of the non-basis matrix at iteration ν.
Except perhaps in hypothetical cases, these choices seem to lead to fewer iterations whereas they are much simpler.

2. Suppose the components of u_i with $i \in I_1 \subset I$ are in the basis after a number of iterations and those with $i \in I_2$ are in the non-basis. It is no restriction to assume that I_1 consists of the first m_1 values of i ($m_1 < m$). From the proof of theorem 4 it follows that the transformed non-basis matrix always has the form:

$$\begin{pmatrix} -R_1^{-1} & R_1^{-1} R_3 \\ -R_3^T R_1^{-1} & R_3^T R_1^{-1} R_3 - R_2 \end{pmatrix}.$$

Hence

$$\begin{aligned} p_{ij}^{\nu} &= p_{ji}^{\nu} \text{ if } i \in I_1 \text{ and } j \in I_1 \text{ or if } i \in I_2 \text{ and } j \in I_2; \\ p_{ij}^{\nu} &= -p_{ji}^{\nu} \text{ if } i \in I_1 \text{ and } j \in I_2 \text{ or if } i \in I_2 \text{ and } j \in I_1. \end{aligned} \tag{8.2.14}$$

It is therefore not necessary to store the whole non-basic tableau but we can restrict ourselves to the part below the main diagonal. Assuming column storage and assuming that the main column $p_{\cdot s}^{\nu-1}$ (hence also the main row) is available, we then obtain the following computational scheme:

a. Calculate p^{ν}: $p_i^{\nu} = p_i^{\nu-1} - p_s^{\nu-1} \cdot \frac{p_{is}^{\nu-1}}{p_{ss}^{\nu-1}}$

b. If $p_i^{\nu} \geqq 0$ for all i, stop, since the problem has been solved; if $p_j^{\nu} < 0$ for some i and the next main row number is selected by rule a of point 1, find

$$p_r^{\nu} = \min_i \{ p_i^{\nu} \mid p_i^{\nu} < 0 \};$$

(hence $r_{\nu} = s_{\nu} = s$ and $r_{\nu+1} = s_{\nu+1} = r$);
if the next main row number is selected by rule b of point 1, calculate

$$p_{ss}^{\nu} = \frac{1}{p_{ss}^{\nu-1}}, \quad p_{ii}^{\nu} = p_{ii}^{\nu-1} - p_{si}^{\nu-1} \cdot \frac{p_{is}^{\nu-1}}{p_{ss}^{\nu-1}}, \quad i \neq s$$

(if i and s both belong to $I_1^{\nu-1}$ or $I_2^{\nu-1}$, then $p_{si}^{\nu-1} = p_{is}^{\nu-1}$; otherwise $p_{si}^{\nu-1} = -p_{is}^{\nu-1}$). Find r by $\frac{p_r^{\nu}}{-p_{rr}^{\nu}} = \min_i \{ \frac{p_i^{\nu}}{-p_{ii}^{\nu}} \mid p_i^{\nu} < 0 \}$.

c. Transform the part below the diagonal of all columns of the non-basis matrix. If $i < r$, select at the same time the r-th element; if $i = r$, select the remaining part of the next main column (always taking (8.2.14) into account).

Not only does this save storage but it will also lead to fewer multiplications since we only calculate one of the two elements p_{ij}^{ν} and p_{ji}^{ν}. It is better to store the diagonal elements themselves separately (this is certainly necessary if rule b of point 1 is used for the selection of the next main row).

3. In the general non-linear programming problem we shall have a sequence of direction-finding problems. The next one will differ from the previous one in the following respects:

a. $I \in I(x^k)$, $i \notin I(x^{k+1})$. In that case $v_i' > 0$ will hold for the final solution of the k-th problem and we can simply delete the i-th column and row.

b. New constraints have to be added to (8.2.3). This entails the addition of columns and rows to the matrix $-PP^T$ of (8.2.8). From this enlarged initial tableau and the final basis B of the previous problem we can immediately derive a good basis for the new problem: $\begin{pmatrix} B & O \\ Q & E \end{pmatrix}$, where Q is the part of the newly calculated rows of the initial tableau belonging to the variables in the final basis of the previous problem. Since the final basis B^{-1} of the previous problem is contained in the final problem we can by means of the formula

$$\begin{pmatrix} B & O \\ Q & E \end{pmatrix}^{-1} = \begin{pmatrix} B^{-1} & O \\ -QB^{-1} & E \end{pmatrix} \qquad (8.2.15)$$

find the enlarged inverse of the basis, the enlarged transformed tableau and the new right-hand member, so that we can start the

new direction-finding problem with this tableau as initial tableau.

c. Some of the rows of the matrix P change, e.g. $q_i(x^{k+1}) \neq q_i(x^k)$, $i \in I(x^k)$ and $i \in I(x^{k+1})$, or $g(x^{k+1}) \neq g(x^k)$, see section 7.5.

If the i-th row changes we can add a new row and column to the final tableau of the previous problem in the way described in point 3 b and must then distinguish three cases:

(i) in the final tableau of the previous problem v_i was in the basis. We can then simply delete the i-th column and row;

(ii) in the final tableau u_i is in the basis and $p'_{ii} < 0$. We can perform one iteration to replace u_i by v_i in the basis after which the i-th row and column can be deleted;

(iii) in the final tableau u_i is in the basis and $p'_{ii} = 0$. We cannot immediately remove u_i from the basis but must prevent u_i from becoming non-zero in the basis. As soon as this happens we remove u_i in the next iteration after which the i-th row and column can be deleted. It is clear, however, that we can hardly expect a computational advantage from such an approach compared with a complete restart.

The problem (8.2.8), (8.2.9) can also be solved by Hildreth's univariate method [26]. This is an infinite procedure for maximizing a concave quadratic function of n variables which are restricted to non-negative values. It is also a method of feasible directions. At the k-th step we take $s^k = \pm e^{\ell}$ where $k = \ell \pmod n$, the sign being determined such that $g(x^k)^T s^k \geqq 0$. If > 0 we determine λ_k (which may be zero) and x^{k+1}. There are no direction-finding problems, so that the procedure is very simple. Unfortunately it seems to converge rather slowly.

Instead we could apply the following infinite procedure, which is justified by the fact that if vectors x', y' and u' satisfy (8.2.6) and (8.2.7) with $\beta = 1$, x' will be proportional to the solution of (8.2.3).

The procedure is:

1. Start with $x^0 = p$ and $u^0 = 0$.
2. Take $i = 1$, $\nu = 1$.
3. Calculate $y_i^{\nu-1} = - p_{i.}^T x^{\nu-1}$ ($p_{i.}$ being the i-th row of the matrix P).
4. If $y_i^{\nu-1} = 0$ or if $y_i^{\nu-1} > 0$ and $u_i^{\nu-1} = 0$ go to 6;

   if $y_i^{\nu-1} > 0$ and $u_i^{\nu-1} > 0$ take $r_\nu = i$ and

$$\lambda_\nu = - \min \left(- \frac{p_{r_\nu .}^T x^{\nu-1}}{p_{r_\nu .}^T p_{r_\nu .}}, u_{r_\nu} \right);$$

   if $y_i^{\nu-1} < 0$, take $r_\nu = i$ and

$$\lambda_\nu = \frac{p_{r_\nu .}^T x^{\nu-1}}{p_{r_\nu .}^T p_{r_\nu .}}.$$

5. Calculate

$$x^\nu = x^{\nu-1} - \lambda_\nu p_{r_\nu .},$$

$$u^\nu = u^{\nu-1} + \lambda_\nu e_{r_\nu}.$$

6. Step up ν and i (modulo n) and repeat steps 3-6.

It can easily be verified that $x^T x$ will decrease monotonously. In fact the λ_ν of this procedure are equal to the λ_ν of Hildreth's procedure from which the convergence follows immediately. The advantage of this procedure is that it is not necessary to calculate first the matrix PP^T, which mostly has considerably more non-zero elements than the matrix P. On the other hand we always have to calculate the inner products $p_{i.}^T x$ which are immediately available in Hildreth's procedure.

The possibility of finding usable feasible directions by means of problems of type (8.2.3) and the equivalence between (8.2.3) and a quadratic programming problem has been discovered independently by Dennis (see [17], chapter 7).

8.3 *Normalization N 2*

Let $x \in R$, then the direction-finding problem becomes:

$$\begin{aligned} \text{Max}\{\sigma \mid\ & q_i(x)^T s + \Theta_i \sigma \leqq 0,\ i \in I_C(x);\ a_i^T s \leqq 0,\ i \in I_L(x); \\ & -g(x)^T s + \sigma \leqq 0;\ 0 \leqq s_j \leqq 1,\ j \in J^-(x); \\ & -1 \leqq s_j \leqq 0,\ j \in J^+(x);\ -1 \leqq s_j \leqq 1,\ j \in J_1(x)\}, \end{aligned} \tag{8.3.1}$$

where $J_1(x) = J - (J^-(x) + J^+(x))$.

This is a bounded-variables problem of type (6.2.6) if we introduce the lower bound 0 and the upper bound $\Sigma \mid g_j \mid$ for σ and replace σ by $\frac{\sigma}{\Sigma \mid g_j \mid}$. It can be solved by means of the dual bounded-variables technique as has been shown in section 6.2 but we prefer to dualize it, so that the absolute-value technique can be applied to it. The problem will then be of the form (6.3.2):

$$\begin{aligned} \text{Max}\{ & -\sum_{j \in J_1(x)} \mid v_j \mid - \sum_{j \in J^-(x)} \max(v_j, 0) + \sum_{j \in J^+(x)} \min(v_j, 0) \\ & \Big| \sum_{i \in I_C(x)} u_i q_i(x) + \sum_{i \in I_L(x)} u_i a_i - u_0 g(x) + v = 0; \\ & \sum_{i \in I_C(x)} u_i \Theta_i + u_0 = 1,\ u_0 \geqq 0,\ u_i \geqq 0\}. \end{aligned} \tag{8.3.2}$$

The procedure of section 6.3 can immediately be applied. In the last paragraph of section 6.3 we have already described how we can proceed from one direction-finding problem to the next one. The application of the revised product-form algorithm makes that the changes to be made are usually rather simple and do not require many calculations.

8.4 *Normalization N 3*

The direction-finding problem is now:

$$\begin{aligned}\text{Max } \{\sigma \,\big|\, & q_i(x)^T s + \Theta_i\, \sigma \leqq 0,\ i \in I_C(x);\ a_i^T s \leqq 0,\ i \in I_L(x);\\ & -g(x)^T s + \sigma \leqq 0;\\ & 0 \leqq s_j \leqq 1,\ j \in J^-(x) \text{ and } g_j(x) > 0;\\ & 0 \leqq s_j\ \ \ ,\ j \in J^-(x) \text{ and } g_j(x) \leqq 0;\\ & -1 \leqq s_j \leqq 0,\ j \in J^+(x) \text{ and } g_j(x) < 0;\\ & s_j \leqq 0,\ j \in J^+(x) \text{ and } g_j(x) \geqq 0;\\ & s_j \leqq 1,\ j \in J_1(x) \text{ and } g_j(x) > 0;\\ & -1 \leqq s_j\ \ \ ,\ j \in J_1(x) \text{ and } g_j(x) < 0\ \}. \end{aligned} \tag{8.4.1}$$

This is again a bounded-variables problem but many of the variables only have one bound, so that by means of a simple transformation a problem will be obtained with all variables required to be non-negative and only a few bounded, so that the direction-finding problems will be somewhat easier to solve than those of N 2. If we dualize (8.4.1) we shall obtain a problem having the same constraints as (8.3.2), extended, however, with requirements

$$v_j \geqq 0 \text{ if } g_j(x) \geqq 0 \text{ and } j \in J^+(x) + J_1(x)$$

and

$$v_j \leqq 0 \text{ if } g_j(x) \leqq 0 \text{ and } j \in J^-(x) + J_1(x).$$

The objective function will be

$$-\sum_{j \in J'} \max(v_j, 0) + \sum_{j \in J''} \min(v_j, 0),$$

where

$$J' = \{j \in J^-(x) + J_1(x) \,\big|\, g_j(x) > 0\}$$

and

$$J'' = \{j \in J^+(x) + J_1(x) \,\big|\, g_j(x) < 0\}.$$

For this dual problem a technique can be developed which is nearly the same as the absolute-value technique of section 6.3.

Remark: With normalization N 3 it may be possible that the sequence of usable feasible vectors is unbounded. Since boundedness is required in procedure P 1 (see section 7.6) we can consider λs instead of s if the absolute largest component of s exceeds a given limit. The scalars $\lambda < 1$ will then be determined such that this absolute largest component is exactly at the limit.

8.5 *Other Normalizations*

N 4: $\sigma \leqq 1$, or $g(x)^T s \leqq 1$ if there are only linear constraints. The direction-finding problems becomes:

$$\text{Max } \{\sigma \mid q_i(x)^T s + \Theta_i \sigma \leqq 0,\ i \in I_C(x);\ a_i^T s \leqq 0,\ i \in I_L(x);$$

$$-g(x)^T s + \sigma \leqq 0;\ s_j \geqq 0,\ j \in J^-(x);\ s_j \leqq 0,\ j \in J^+(x);\ \sigma \leqq 1\} \qquad (8.5.1)$$

This normalization is a very simple one since only one extra constraint has to be added to the tableau. The variables s_j with $j \in J_1(x) = J - (J^-(x) + J^+(x))$ will be free, hence may assume negative values. For the solution of (8.5.1) we propose the following special method (which is essentially the simplex method):

1. Make an initial tableau, consisting of a non-basis matrix with rows $(q_i(x)^T, \Theta_i)$, $(-g(x)^T, 1)$ and $(a_i^T, 0)$. The constraint $-g(x)^T s + \sigma \leqq 0$ will be considered as one of the constraints $q_i(x)^T s + \Theta_i \sigma \leqq 0$, so that $I_C(x)$ is extended with one element. The slack variables of these constraints will be denoted by t_i. Omit the right-hand member (which is completely zero), the constraint $\sigma \leqq 1$ and an initial d row. Introduce sets

$$t(I) = \{t_i \mid i \in I(x) = I_C(x) + I_L(x)\};\ s(J^+) = \{s_j \mid j \in J^+(x)\};$$

$$s(J^-) = \{s_j \mid j \in J^-(x)\}$$

and

$$s(J_1) = \{s_j \mid j \in J_1(x)\}.$$

2. Introduce the variable σ in the basis in the first iteration. The variable which leaves the basis will be determined in the way described in point 3 b.

3. Suppose a number of iterations has already been performed and a non-basic tableau with columns $a_{.h}^*$ has been obtained. The variable σ is still in the basis and the corresponding row of the non-basic matrix will consist of elements $a_{\sigma h}^*$. The non-basic variables will be denoted by s_h^*, the basic variables by t_i^*. The procedure is now:

a. Selection of the variable s_p^*, to be introduced in the basis:

(i) Determine $H^* = \{h \mid a_{\sigma h}^* \neq 0$ and $s_h^* \in s(J_1)$, or
$a_{\sigma h}^* < 0$ and $s_h^* \in s(J^-) + t(I)$, or
$a_{\sigma h}^* > 0$ and $s_h^* \in s(J^+)\}$.

(ii) If $H^* = 0$, stop, $s = 0$ is the optimum solution of (8. 5. 1).

(iii) If $H^* \neq 0$, determine $p \in H^*$ such that $|a_{\sigma p}^*| \geqq |a_{\sigma h}^*|$ for all $h \in H^*$.

b. Selection of the variable t_r^* which will leave the basis:

(i) Determine $I^* = \{i \mid -a_{ip}^* \operatorname{sgn} a_{\sigma p}^* > 0$ and $t_i^* \in t(I) + s(J^-)$ or $-a_{ip}^* \operatorname{sgn} a_{\sigma p}^* < 0$ and $t_i^* \in s(J^+)\}$.

(ii) If $I^* \neq 0$, choose $r \in I^*$ such that $|a_{rp}^*| \geqq |a_{ip}^*|$ for all $i \in I^*$ and perform the transformation. During the transformation the main column for the next iteration is selected.

(iii) If $I^* = 0$, stop and take as usable feasible direction:

$$s_p^* = -\lambda \operatorname{sgn} a_{\sigma p}^*;$$
$$s_j^* = 0, \quad j \neq p;$$
$$t_i^* = \lambda a_{ip}^* \operatorname{sgn} a_{\sigma p}^*;$$
$$\sigma = \lambda a_{\sigma p}^* \operatorname{sgn} a_{\sigma p}^*,$$

where λ could be determined such that $\sigma = 1$, i.e. $\lambda = \frac{1}{|a_{\sigma p}^*|}$. Since we need bounded vectors, however, we prefer to determine $\lambda \leqq 1$ such that the absolute largest component of the solution vector s does not exceed a given bound.

This procedure is justified by the following facts:

a. $d_h^* = a_{\sigma h}^*$ always holds since σ is the only variable contributing to the objective function and does not leave the basis after its arrival there in the first iteration (the set I^* of point 3 b never contains the row number belonging to the variable σ). It follows that the rules 3 a, (i) and (ii) are the normal rules for the selection of a main column in the case of some variables being unrestricted or being restricted by an upper bound instead of a lower bound.

b. As long as $I^* \neq 0$, all right-hand member figures will always be zero, so that r is chosen by the usual simplex criterion (the factor $-\operatorname{sgn} a_{\sigma p}^*$ takes the fact into account that we want to increase s_p^* if $a_{\sigma p}^* \leqq 0$ and to decrease it if $a_{\sigma p}^* > 0$ holds. Variables $t_i^* \in s(J^+)$ are not allowed to assume positive values; variables $t_i^* \in s(J_1)$ need not be considered since they are unrestricted. Ties are broken by taking the absolute largest of possible pivot elements. This is the usual criterion in computer codes, which does not guarantee finiteness in all cases but is more efficient in practice.

c. If $I^* = 0$ in some iteration we should obtain an infinite solution if there were no relation $\sigma \leqq 1$. With this relation we could now perform a final iteration with pivot element $a_{\sigma p}^*$ to eliminate σ from the basis by putting it at its upper bound in the non-basis but we prefer to omit this final iteration and to act as described in point 3 b, (iii) of the procedure. It follows that N 4 is equiva-

lent to a procedure without any normalization but with a stop at an infinite solution, where we then derive the usable feasible direction from the extreme ray leading to infinity.

The next direction-finding problem will be of the same form. Since we have not performed the last iteration, we shall be sure that a new constraint will also have a zero right-hand member if it is added to the final tableau of the previous problem.

The revised product-form algorithm can be applied to the solution of (8.5.1) but then it is hardly possible to use the final set of η-vectors for the next problem, the number of constraints not being constant. But instead we can apply the dual revised product-form algorithm to the dual problem (section 4.6, point c). Alternatively we could apply the explicit inverse algorithm to the primal problem since in this algorithm addition of constraints is possible.

N 5: This normalization also takes into account the constraints in which the trial solution x does not lie. It leads to a linear programming problem of type (7.3.16) which can be solved by means of the simplex method. It can of course also be solved by one of the methods of feasible directions already described, which will then be applied to a linear programming problem. Chapter 9 will be devoted to this application.

It is not difficult to find still other normalizations, leading to new convex programming procedures. Instead of N 3 we could for instance require $s_j \leqq g_j(x)$ if $g_j(x) > 0$ and $g_j(x) \leqq s_j$ if $g_j(x) < 0$.

8.6 *Comparison of the Different Normalizations*

Without computational experience it is very difficult to make a comparison of the different normalizations. Therefore we shall only briefly indicate what can be expected from a theoretical point of view:

1. Normalization N 1 will in general lead to fewer steps than N 2, N 3 of N 4, the reason being that it gives the feasible direction which makes the smallest possible angle with the gradient vector.

2. The amount of work to be expected in a step with normalization N 1 will be considerably greater than with any of the other normalizations N 2 - N 4. For, we always have to calculate the matrix PP^T which entails many multiplications but also leads to a matrix with more non-zero elements. It is less easy with N 1 to start with the final tableau of the previous direction-finding problem. If many rows in the matrix change, then the very fast re-inversion feature of the revised product-form algorithm makes it possible to restart with a re-inversion, so that a restart is relatively cheap. Since the revised product-form algorithm can hardly be applied in the case of N 1, a procedure with this normalization will have worse restart possibilities. This is especially of importance if some of the active constraints are non-linear.

In normalization N 2 - N 4 the simple constraints $s_j \geqq 0$ (if

$j \in J^-(x)$) or $s_j \leqq 0$ (if $j \in J^+(x)$) do not add to the size of the direction-finding problems but in N 1 they do. This is another big disadvantage of normalization N 1.

3. In the order N 2 - N 3 - N 4 we can on the average expect more steps but less calculating per step. What dominates can only be established computationally and will depend on the problem.

4. Normalization N 5 will lead to larger problems. In some trial examples the procedure P 2, based on it, appeared to converge slowly in the case of linear constraints and a non-linear objective function (optimum not in a vertex) but rather fast in the case of non-linear (quadratic) constraints and a linear objective function (optimum in a "vertex"). It can certainly be expected to lead to the most rapid procedure if the problem is only slightly non-linear, i.e. if most of the constraints are linear and the non-linear ones as well as the objective function consist for the greater part of linear terms, owing to the fact that the number of steps can be expected to be less, so that there will not be so many recalculations of the gradient vector and normals.

In chapter 11 we shall see how the convergence can be speeded up in the case of a non-linear objective function. When using N 5 it is hardly possible, if there are many non-linear constraints, to start a direction-finding problem with the final tableau of the previous problem but when applying the revised product-form algorithm we can of course start with a re-inversion.

9. THE LINEAR PROGRAMMING PROBLEM AND THE METHODS OF FEASIBLE DIRECTIONS

9.1 *Introduction*

In this chapter the methods of feasible directions will be applied to the linear programming problem. In section 9.2 the different procedures will be summarized and proofs of finiteness will be given, the computational aspects being dealt with in section 9.3. Section 9.4 will be devoted to a discussion of the procedures. We shall show that the simplex method itself and also the primal-dual method are methods of feasible directions. In section 9.5 we shall study an application of a method of feasible directions to the problem of large-scale linear programs.

9.2 *Survey of the Procedures*

Let the linear programming problem be

$$\text{Max } \{p^T x \mid A x \leqq b,\ 0 \leqq x \leqq c\}. \tag{9.2.1}$$

The rows of the matrix A will be denoted by a_i^T. If, for some i, $a_i^T x = b_i$ is required instead of $\leqq b_i$, then we can either eliminate some of the variables beforehand or we can consider each equation as two inequalities and, as soon as we have found an x^k satisfying $a_i^T x^k = b_i$ for this i, require $a_i^T s = 0$ instead of $\leqq 0$ in all direction-finding problems.

Procedure P 1 will now be as follows:

1. We start with $x^0 \in R$ (see section 7.2 if such a point is not available).

2. Let $x^0, x^1, \ldots, x^k$ have already been determined. Then solve the k-th direction-finding problem

$$\text{Max } \{p^T s \mid a_i^T s \leqq 0,\ i \in I(x^k);\ s_j \geqq 0,\ j \in J^-(x^k);$$
$$s_j \leqq 0,\ j \in J^+(x^k);\ N\ 1, 2, 3 \text{ or } 4\}. \tag{9.2.2}$$

3. If $p^T s^k = 0$, stop, x^k is an optimum solution;
if $p^T s^k > 0$, determine:

$$\lambda_k = \max \{\lambda \mid a_i^T (x^k + \lambda s^k) \leqq b_i,$$
$$0 \leqq x^k + \lambda s^k \leqq c\}. \tag{9.2.3}$$

4. If $\lambda_k = \infty$ stop, the problem has an infinite solution (this is only possible if some of the c_j are ∞);
if $\lambda_k < \infty$ determine

$$x^{k+1} = x^k + \lambda_k s^k, \tag{9.2.4}$$

and repeat steps 2-4.

Theorem 1: Procedure P 1 with normalization N 1 leads to a finite number of steps.
Proof: Suppose s^k is the solution of (9.2.2) with N 1. Let $a_i^T s^k = 0$ for $i \in I_1(x^k) \subset I(x^k)$, and < 0 if $i \in I_2(x^k) = I(x^k) - I_1(x^k)$. In the same way $s_j^k = 0$ if $j \in J_1^-(x^k) \subset J(x^k)$, and > 0 if $j \in J_2^-(x^k) = J^-(x^k) - J_1^-(x^k)$, and $s_j^k = 0$ if $j \in J_1^+(x^k)$, and < 0 if $j \in J_2^+(x^k) = J^+(x^k) - J_1^+(x^k)$. If the sets $I(x^k)$, $J^-(x^k)$ and $J^+(x^k)$ in (9.2.2) are replaced by $I_1(x^k)$, $J_1^-(x^k)$ and $J_1^+(x^k)$, then the solution s^k will remain the same. If $0 < \lambda < \lambda_k$ we have $I(x^k + \lambda s^k) = I_1(x^k)$, $J^-(x^k + \lambda s^k) = J_1^-(x^k)$ and $J^+(x^k + \lambda s^k) = J_1^+(x^k)$. Hence, for $\lambda < \lambda_k$, we have $s^k \in S(x^k + \lambda s^k)$ while, for any other normalized $s \in S(x^k + \lambda s^k)$ we have: $p^T s < p^T s^k$. The latter follows from the uniqueness theorem 3 of section 8.2. If $\lambda_k < \infty$, so that x^{k+1} is defined by (9.2.4), then $I(x^{k+1}) \supset I(x^k + \lambda s^k)$, $J^-(x^{k+1}) \supset J^-(x^k + \lambda s^k)$ and $J^+(x^{k+1}) \supset J^+(x^k + \lambda s^k)$ for all λ, $0 < \lambda < \lambda_k$. It follows that $S(x^{k+1}) \subset S(x^k + \lambda s^k)$ for all λ, $0 < \lambda < \lambda_k$ whereas $s^k \notin S(x^{k+1})$. Hence $s^{k+1} \in S(x^{k+1})$ satisfies $p^T s^{k+1} < p^T s^k$. No two direction-finding problems can thus be the same. The total number of sets $I(x)$, $J^-(x)$ and $J^+(x)$ to be selected from I and J respectively, hence the total number of possible direction-finding problems, is finite, from which the finiteness of the procedure follows.

With normalizations N 2, N 3 or N 4, it is possible that $p^T s^{k+1} = p^T s^k$, since no uniqueness theorem holds now. This will always be the case when N 4 is applied since $p^T s^k = 1$ will hold here for all k (bounding afterwards of the vectors s^k is not necessary in the linear programming case since there is no anti-zigzagging precaution AZ 1 or AZ 2). If N 2 or N 3 is applied, this difficulty can be overcome by adding ε^j to the components p_j, where ε is a very small positive number. If ε is small enough, we shall be sure that the vector p will never be dependent on less than n of the rows of the direction-finding problem, so that there will never be alternative solutions in such a problem. Consequently we shall always have $p^T s^{k+1} < p^T s^k$ from which the finiteness of the method follows as in the case of normalization N 1. The addition of ε^j leads to an order of preference for the variables s_j the rules of which can easily be determined, so that the actual addition of ε^j is not necessary. If N 4 is applied, this ε-procedure can not be used.

Theorem 2, section 9.4 shows us, however, that with N 4 a linear programming procedure is obtained which is completely equivalent to the simplex method itself, so that we can apply any anti-degeneracy precaution which works for the simplex method. It follows that we have:

Theorem 2: Procedure P 1 with one of the normalizations N 2, N 3 or N 4 and an anti-degeneracy precaution will lead to a finite number of steps.

The degeneracy in the case of P 1 with N 2, N 3 or N 4 and no precaution may lead to zigzagging between hyperplanes. This zigzagging can also be prevented by the anti-zigzagging precaution AZ 3 (see section 7.5).

Theorem 3: Procedure P 1 with N 2, N 3 or N 4 and AZ 3 leads to a finite number of steps.

Proof: AZ 3 will guarantee that after a finite number of steps we shall obtain $p^T s^k = 0$ and shall have to replace one or more of the requirements $n_h^T s = 0$ by $n_h^T s \leqq 0 (n_h = a_i, e_j \text{ or } -e_j)$. We have then arrived at a point x^k belonging to a certain set of hyperplanes and are sure that $p^T x = p^T x^k$ will hold for any x, belonging to the same set of constraining hyperplanes. Hence, if the calculation can be continued, we shall never return in the same set of hyperplanes. From this the finiteness easily follows.

Procedure P 2 will give the original linear programming problem, so that this procedure does not lead to a new method in the linear case.

9.3 *Computational Aspects*

1. In the linear case the change from a direction-finding problem to the next one will always be rather simple. We have to delete and to add some constraints. This can easily be done. If the direction-finding problems are solved by means of their dual problems to which the revised product-form algorithm is applied, then we only have some trivial changes in the original matrix of the direction-finding problem but no changes in the η-vectors obtained from the previous problems. Re-inversions can be performed at regular times to speed up future operations and to improve accuracy.

2. If we introduce $t_i^k = - a_i^T s^k$, then (9.2.3) is equivalent to

$$\lambda_k = \min (\lambda_{1k}, \lambda_{2k}, \lambda_{3k}), \tag{9.3.1}$$

with

$$\lambda_{1k} = \min \left\{ \frac{y_i^k}{-t_i^k} \;\middle|\; i \in I - I(x^k),\ t_i^k < 0 \right\}, \tag{9.3.2}$$

$$= \infty \text{ if } t_i^k \geqq 0 \text{ for all } i \in I;$$

$$\lambda_{2k} = \min \left\{ \frac{x_j^k}{-s_j^k} \,\middle|\, j \in J - J^-(x^k),\ s_j^k < 0 \right\}, \tag{9.3.3}$$

$$= \infty \text{ if } s_j^k \geqq 0 \text{ for all } j \in J;$$

$$\lambda_{3k} = \min \left\{ \frac{c_j - x_j^k}{s_j^k} \,\middle|\, j \in J - J^+(x^k),\ s_j^k > 0 \right\}, \tag{9.3.4}$$

$$= \infty \text{ if } s_j^k \leqq 0 \text{ for all } j \in J.$$

Hence we must calculate the variables t_i for all i belonging to rows which are not in the direction-finding problem. From this and the fact that rows must be added to or deleted from our direction-finding problems it follows that the original matrix must be stored row-wise. If, in a computer, the matrix is stored on magnetic tape and read for the calculation of λ_{1k}, then the row a_i^T can be kept in store if $t_i^k < 0$ and $\frac{y_i^k}{-t_i^k}$ is not greater than the smallest quotient obtained up to then. For this calculation y_i^k has to be available so that these quantities will also be calculated in each iteration:

$$y_i^{k+1} = y_i^k + \lambda_k t_i^k, \quad i = 1, \ldots, m. \tag{9.3.5}$$

Obviously $y_i^k = 0$ if $i \in I(x^k)$.

3. The methods of feasible directions described in this section essentially work with rows instead of columns. Since in many problems $n > m$, these rows may be relatively long which seems to be a rather unfavourable situation. By column partitioning, however, we can solve a sequence of restricted problems as will be shown in section 9.5. These restricted problems will have rows with fewer elements.

9.4 *Discussion of the Procedures*

Normalizations N 1 - N 4 all lead to different methods for solving the linear programming problem. These methods consist of a sequence of direction-finding problems connected by rather simple determinations of the step length. Two consecutive direction-finding problems have a great resemblance which can be used by starting the solution of such a problem with the optimal solution of the previous one. If N 2, 3 or 4 is applied, the number of rows in a direction-finding problem will always equal the number of active constraints with $h \in I(x)$ and will never be more, usually considerably less than the number of rows of the matrix A. If N 1 is applied the number of rows in the direction-finding problem will generally be not more than n.

We shall first prove the following theorems:

Theorem 1: The method of feasible directions with normalization N 3 applied to the dual of a linear programming problem is equivalent to the primal-dual method.
Proof: Let us apply the primal-dual method to a problem of type (3.4.1), then we obtain a sequence of restricted primal problems of type (3.4.4). Let us now apply the method of feasible directions with N 3 to the dual problem (3.4.2) in which the u_i variables are unrestricted. It follows from (8.4.1) that a sequence of direction-finding problems will be obtained of the type

$$\text{Max } \{-b^T s \mid -\Sigma a_{ij} s_i \leqq 0,\ j \in J(u);\ -s_i \leqq 1,\ i \in I\},$$

where

$$J(u) = \{j \mid v_j = 0\} \qquad \text{(see (3.4.3))}.$$

But the dual of this direction-finding problem is exactly (3.4.4), so that both methods lead to restricted problems of the same type. It can further easily be verified that $s_i' = d'(z_i) - 1$, where s' is the optimum solution of the direction-finding problem and $d'(z_i)$ is the final d value of variable z_i in (3.4.4). Comparison of the formulae (3.4.7) and (3.4.8), and the formulae (9.2.3), (9.3.2) and (9.2.4) shows that the determination of λ is also completely equivalent. This proves the theorem.

Theorem 2: The method of feasible directions with normalization N 4 and with the initial solution $x^0 = 0$, $y^0 = b \geqq 0$, leads to the same intermediate solutions x^k as the simplex method does.
Proof: If we start with $x^0 = 0$, $y^0 = b$, then all variables s_j will be restricted in the first direction-finding problem. Since an s_j variable in the basis will only leave it if x_j is at one of its bounds, we shall be sure that no unrestricted s_j will ever be in the non-basis. Hence, the rules 3a, (i), (ii) and (iii) of section 8.5. are exactly the same as in the simplex method for bounded variables (see section 6.2, the bounds are on the variables x_j, not on the s_j). The direction-finding problems are of the type

$$\text{Max } \{p^T s \mid a_i^T s \leqq 0,\ i \in I(x);\ s_j \geqq 0,\ j \in J^-(x);$$
$$s_j \leqq 0,\ j \in J^+(x);\ p^T s \leqq 1\}.$$

In the first one we have $J^+(x) = 0$, $J^-(x) = J$ and $I(x) = \{i \mid b_i = 0\}$. Since there are no unrestricted variables the rules 3b, (i) and (ii) of section 8.5. for the finding of pivot elements in the direction-finding problem will be the same as the rules of the simplex method until in some iteration a pivot element will be chosen in the simplex method which does not correspond to a zero b-figure. But in that case our direction-finding problem will be solved. The choice of $\lambda > 0$ in the simplex method is determined by the formulae (6.2.2), (6.2.3), (6.2.4) and (6.2.5). But since we have $s_i^* = -a_{ip}^*$ as solution of the direction-finding problem (the asterisk denotes the matrix elements at the moment that $I^* = 0$ held, p is

the main column number at that moment), the choice of our step length λ with the help of (9.3.1), (9.3.2), (9.3.3) and (9.3.4) is exactly the same. In the simplex method a pivot element will now be chosen in the row determining λ. (In case of ties we can take the absolute largest possible pivot element which is only a practical but not an anti-degeneracy device). In our method the rows which determine λ will be added to the direction-finding problem and in the first iteration of the new problem the choice of the pivot element will be in exactly the same row (provided the procedure for breaking ties is the same). Now suppose that we have already performed $\nu-1$ iterations and that there is still complete equivalence, so that in the final solution of the direction-finding problem s_j will be in the basis if x_j is in the basis in the simplex method. Variable t_i will be in the direction-finding problem if $y_i = 0$ and it will be in the basis if $y_i = 0$ is still in the basis. It is clear that the choice of pivot elements in the simplex method and the direction-finding problem will be the same as long as there is no increase in value in the simplex method. As soon as there is an increase the direction-finding problem will be solved. The choice of the step length λ will again be the same, as in the first iteration. In the simplex method one of the x_j variables may now be removed from the basis. In that case the new λ will be determined by the constraint $x_j = 0$ or $= c_j$, so that $s_j \geqq 0$ ($\leqq 0$) will be added to the direction-finding problem. In the next direction-finding problem s_j will then be removed from the basis in the first iteration. Hence, there will be complete equivalence after ν steps. This proves the theorem.

The equivalence will also hold if we start in an arbitrary vertex of the feasible region. There will then be a pre-inversion in the simplex method which is equivalent to the introduction of all unrestricted variables in the basis in our method. The direction-finding problems will always have a number of rows equal to the number of x_j variables in the basis of the simplex method plus the number of y_i variables which are in the basis at zero level.

Theorems 1 and 2 show that the simplex method itself and the primal-dual method are special cases of the more general method of feasible directions. The latter method uses the simplex change of basis for finding directions rather than the trial solutions themselves and is therefore capable of going inside the feasible region. This makes the extension to non-linear programming possible. We can state:

1. The primal-dual method is a large-step gradient method in the dual feasible region. Therefore the technique of section 7.2 can be applied if an initial feasible solution of the dual problem is not available.

2. The method of feasible directions with normalization N 3 and applied to non-linear programming problems can be considered as an extension of the primal-dual method to non-linear (convex) programming.

3. The method of feasible directions with normalization N 4 can be considered as an extension of the simplex method to non-linear (convex) programming.

4. Also in the case of linear programming the method of feasible directions with normalization N 4 is more general than the simplex method since it can be started in any feasible point, so that prior knowledge of the solution can be used.

5. The organizational difference between the simplex method and P 1 with N 4 is that the simplex method considers all constraints whereas in the method of feasible directions only those constraints are considered in the direction-finding problems which are active at that moment of the calculation. This makes for more work between steps in our method but reduces the size of the direction-finding problems.

6. Computationally it seems cheapest in the method of feasible directions to apply the revised product-form algorithm to the dual of the direction-finding problem (in the case of N 2 or N 3 we shall then apply the primal RPFA; in the case of N 4 the dual RPFA whereas in the case of N 1 we must apply the special technique of section 8.2; with normalization N 4 the explicit inverse algorithm, applied to the primal problem, will be the chief competitor).

7. A comparison between the various methods for the linear programming problem can hardly be made without computational experience but we shall have:

a. In the methods of feasible directions the number of steps will usually decrease in the order N 4 - N 3 - N 2 - N 1 but the amount of work per step will increase. Especially N 1 will be more expensive than N 2, 3 or 4 (see section 8.6).

b. The methods of feasible directions can make very good use of prior knowledge of the final solution and this will certainly lead to a computational advantage.

c. The simplex method will require the simplest computer code, even if the revised product-form algorithm is used as computational version.

d. All methods will give the final dual variables together with the primal solution, so that post-optimal studies can easily be made.

e. The methods of feasible directions can easily deal with bounded-variables problems as can the simplex method.

f. Parametric programming, post-optimal changes in the right-hand member, the objective function or the matrix elements can be accomplished as easily in the methods of feasible directions as in the simplex method, provided normalization N 2, 3 or 4 is used. This is a consequence of the fact that the final tableau of the last direction-finding problem is equivalent to a part of the last simplex tableau.

9.5 *Large-Scale Linear Programming Problems*

In this section we shall briefly outline how a method of feasible directions could be applied to the solution of a very large linear programming problem. If the problem becomes large because of

the fact that m is much greater than n, then the advantage of the methods of feasible directions is immediately clear. For, it may be expected that many of the rows will be inactive in that case, so that the direction-finding problems do not grow so large. In the more likely case of $n >> m$ a row-wise approach does not seem to be advantageous. We can partition the set J, however, in two subsets J_1 and J_2 and form the matrices A_1 and A_2 consisting of the columns $a_{.j}$ with $j \in J_1$ and $j \in J_2$, respectively. We then solve the restricted problem.

$$\text{Max } \{p_1^T x_1 \mid A_1 x_1 \leqq b,\ x_1 \geqq 0\}. \tag{9.5.1}$$

In other words we assume that $x_j = 0$ if $j \in J_2$ will hold in the final solution of (9.2.1). Instead we can also start with $x_j = x_j^0$ if $j \in J_2$ and replace b in (9.5.1) by $b - A_2 x_j^0$.

The solution of the restricted problem will be denoted by x'_{1j} and $x'_{2j} = x_{2j}^0$. Next we calculate the d values of the x_{2j}. This can easily be done with the help of the final data of the last direction-finding problem. If u'_i is the dual variable corresponding to the i-th constraint we simply have $d'(x_{2j}) = \Sigma\, u'_i\, a_{ij} - p_{2j}$. We now make a new partitioning J_3, J_4, e.g. in the following way:
if $j \in J_2$, $x'_{2j} = 0$ and $d'(x_{2j}) \leqq 0$, or if $x'_{2j} = c_j$ and $d'(x_{2j}) \geqq 0$, or if $0 < x'_{2j} < c_j$, assign j to J_3;
if $j \in J_2$, $x'_{2j} = 0$ and $d'(x_{2j}) > 0$, or if $x'_{2j} = c_j$ and $d'(x_{2j}) < 0$, assign j to J_4;
if $j \in J_1$ and $x'_{1j} = 0$ or $= c_j$ with $d'(x_{1j}) \neq 0$, assign j to J_4; the other $j \in J_1$ will belong to J_3.

In this way be obtain a new restricted problem

$$\text{Max } \{p_3^T x_3 \mid A_3 x_3 \leqq b - A_4 x'_4,\ x_3 \geqq 0\}. \tag{9.5.2}$$

Since x'_1, x'_2 is a feasible solution of (9.5.2) we shall obtain a solution x'' of (9.5.2), satisfying $p^T x'' \geqq p^T x'$ and with a non-degeneracy assumption $p^T x'' > p^T x'$. Each new restricted problem equals the primal problem under the additional assumption that $x_j = 0$ or $= c_j$ for some values of j.

Since the total value of the objective function will not decrease we can, relying on the existence of anti-degeneracy precautions, conclude that the same set of variables will never be at the same bounds from which the finiteness of the procedure follows. If the revised product-form algorithm is applied to the dual of the direction-finding problems, then we shall start each restricted problem of type (9.5.2) with a pre-inversion leading to the same variables in the basis as there were in the basis of the last direction-finding problem.

We shall apply this procedure to the following block-triangular system:

$$\text{Max } \{p_0^T x_0 + \sum_{h=1}^{r} p_h^T x_h \mid A_{h0} x_0 + A_{hh} x_h \leqq b_h;\ x_h \geqq 0,\ h = 1, \ldots, r;\ x_0 \geqq 0\}, \tag{9.5.3}$$

where, for each $h \geqq 0$, p_h and x_h are vectors with the same number n_h of components, b_h is a vector with m_h components and A_{h0} and A_{hh} are m_h by n_0, and m_h by n_h matrices, respectively. We now propose the following procedure which, except in the initial step, equals the procedure already described in this section.

1. Initial step:

a. Fix the values of the linking variables x_{0j} by using some estimate for them: $x_{0j} = x_{0j}^0$ (the better the estimate, the smaller the number of steps to be performed).

b. Solve r different small linear programming problems

$$\text{Max } \{p_h^T x_h \mid A_{hh} x_h \leqq b_h - A_{h0} x_0^0\},$$

leading to the final solution (y_h^*, x_h^*), the final matrix A_{hh}^* and the final inverse of the basis $\{B_{hh}^*\}^{-1}$.

c. Transform A_{h0}: $A_{h0}^* = \{B_{hh}^*\}^{-1} A_{h0}$.

We now have a new linear programming problem, still block-triangular:

$$\text{Max } \{(p_0^*)^T x_0 + \sum_{h=1}^{r} (p_h^*)^T x_h^* \mid A_{h0}^* x_0 + A_{hh}^* x_h^* \leqq b_h^*, \quad x_h^* \geqq 0, \; x_0 \geqq 0\}. \tag{9.5.4}$$

2. Apply the method of feasible directions to (9.5.4) starting with $x_0 = x_0^0$ and $x_h^* = 0$ and taking the initial partitioning $j \in J_1$ if j belongs to an x_0 variable, $j \in J_2$ otherwise, i.e. find the best solution x_0 assuming that $x_h^* = 0$ is correct.

3. Calculate dual variables and make a new partitioning J_3, J_4 in the way described, etc.

The initial step is only proposed since it will lead to a final tableau which is still block-triangular but is such that, if the initial estimate x_0^0 for x_0 is good, many variables x_{hj} may be expected to be rightly in the basis at the end of step 1, so that the direction-finding problems of steps 2 and 3 will contain relatively few constraints.

The same partitioning idea can be applied to other special structures.

10. QUADRATIC PROGRAMMING

10.1 *Introduction*

In this chapter we shall derive some finite quadratic programming procedures based on the idea of feasible directions and on the idea of conjugated directions. In section 10.2 the different procedures will be described and proofs of finiteness will be given. Section 10.3 will be devoted to a discussion of the quadratic programming methods. Here we shall state that Beale's quadratic programming method is also a method of feasible directions.

10.2 *Description of the Procedures*

Let the quadratic programming problem be given by

$$\text{Max } \{p^T x - \tfrac{1}{2} x^T C x \mid Ax \leqq b,\ 0 \leqq x \leqq c\}, \tag{10.2.1}$$

where p, c and x are n-component vectors, C is an n by n symmetric positive semi-definite matrix, A an m by n matrix, and b an m-component vector.

If some of the linear constraints are equations instead of inequalities we can either eliminate some of the variables or we can always require $a_i^T s = 0$ instead of $\leqq 0$ provided $x \in R$ holds. In the former case the number of variables will be reduced but a special, simple structure of the matrices A and C may get lost. It will thus depend on the problem which procedure is preferable.

Let $x \in R$. We always have:

$$g(x) = p - Cx. \tag{10.2.2}$$

Let $s \in S(x)$ and $g(x)^T s > 0$.

If

$$\lambda' = \max \{\lambda \mid x + \lambda s \in R\}, \tag{10.2.3}$$

and

$$\lambda'' = \max \{\lambda \mid g(x + \lambda s)^T s \geqq 0\},$$

i.e.

$$\lambda'' = \frac{g(x)^T s}{s^T C s} \quad \text{if} \quad s^T C s > 0,$$
$$= \infty \qquad \text{if} \quad s^T C s = 0, \tag{10.2.4}$$

then

$$\lambda = \min(\lambda', \lambda''). \tag{10.2.5}$$

We further have:

$$x^{k+1} = x^k + \lambda_k s^k; \tag{10.2.6}$$

$$g(x^{k+1}) = g(x^k) - \lambda_k C s^k; \tag{10.2.7}$$

$$\Delta F(x^k) = \lambda_k g(x^k)^T s^k - \tfrac{1}{2} \lambda_k^2 (s^k)^T C s^k, \tag{10.2.8}$$

$$= \tfrac{1}{2} \lambda_k \{g(x^k)^T s^k + g(x^{k+1})^T s^k\},$$

so that

$$\Delta F(x^k) = \tfrac{1}{2} \lambda_k \, g(x^k)^T s^k \quad \text{if} \quad \lambda_k = \lambda_k''. \tag{10.2.9}$$

We now propose the following extra requirements for the direction-finding problems:

1. AZ 3 (see section 7.5).
2. a If $\lambda_k = \lambda_k'' < \lambda_k'$ we shall add the requirement $s^T C s^k = 0$ to the next direction-finding problem and retain all constraints of this kind which were already present in the previous direction-finding problem.
   b If $\lambda_k = \lambda_k'$ all extra requirements of the form $s^T C s^k = 0$ will be omitted in the next direction-finding problem.

The relation $s^T C s^k = 0$ is a linear one (s^k is a known vector), so that it will not lead to difficulties in the direction-finding problem. It is equivalent to the relation $\{g(x^{k+1}) - g(x^k)\}^T s = 0$. It expresses the requirement that the new vector s shall be conjugated to s^k with respect to the matrix C.

Suppose $\lambda = \lambda_h'' < \lambda_h'$ holds for $h \geqq r$. Hence we always add a constraint of the form $s^T C s^h = 0$ to the direction-finding problem. By section 2.3, theorem 7, the vectors s^k will be linearly independent for $h \geqq r$, so that we shall get $g^T s_{opt} = 0$ in some direction-finding problem after a finite number of steps. Let this be at the k-th step. We have $x^k = x^r + \sum_{h=r}^{k-1} \lambda_h s^h$ and

Theorem 1: Suppose $\lambda_h = \lambda_h'' < \lambda_h'$ for $h \geqq r$, so that $(s^h)^T C \, s^\ell = 0$

for $h \geqq r$, $\ell \geqq r$ and $h \neq \ell$, where $s^h \in S(x^h)$. Let $x^{h+1} = x^h + \lambda_h s^h$ and suppose that x^k, $k > r$, is such that $g(x^k)^T s \leqq 0$ for all $s \in S(x^k)$, satisfying $s^T C\, s^h = 0$ for $h = r, r+1, \ldots, k-1$. Then $g(x^k)^T s \leqq 0$ for all $s \in S(x^k)$.

Proof: Suppose for some $s \in S(x^k)$, $g(x^k)^T s > 0$. Define the vector t by

$$s = \sum_{h=r}^{k-1} \mu_h s^h + t \quad \text{with } \mu_h = \frac{(s^h)^T C\, s}{(s^h)^T C\, s^h}.$$

Hence $(s^\ell)^T C t = 0$ for $\ell = r, r+1, \ldots, k-1$ and $g(x^k)^T s = g(x^k)^T t > 0$ (see the proof of theorem 8, section 2.3), so that t is conjugated to s^ℓ for $\ell = r, r+1, \ldots, k-1$, and $g(x^k)^T t > 0$. Moreover, if $i \in I(x^k)$, we have $i \in I(x^h)$ for all h, $r \leqq h \leqq k-1$ (otherwise $\lambda_h = \lambda_h'$ would hold for some h, $r \leqq h \leqq k-1$). It follows that, for all $i \in I(x^k)$, we have $a_i^T s^h = 0$, $r \leqq h \leqq k-1$ (if < 0 we should have left the i-th hyperplane and could not have returned in it, hence $i \in I(x^k)$ could not hold). The same holds for $J(x^k)$: if $j \in J^-(x^k) + J^+(x^k)$, then $s_j^h = 0$ holds for $r \leqq h \leqq k-1$. Hence $t \in S(x^k)$, so that t would be a usable feasible direction conjugated to all vectors s^h, $r \leqq h \leqq k-1$, contrary to the assumption that such a vector does not exist. This proves the theorem.

Before proving the finiteness we shall first summarize one step in the procedure:

1. Suppose $x^k \in R$ has already been determined and the final data of the last direction-finding problem are available. Suppose further that a set Z of indices i and j as well as an index $r \leqq k-1$ has already been determined (Z may be empty).

2. If $\lambda_{k-1} = \lambda_{k-1}'' < \lambda_{k-1}'$, add the constraint $(s^{k-1})^T C\, s = 0$ to the direction-finding problem, so that this becomes:

$$\begin{aligned} \text{Max } \{ g(x^r)^T s \;\big|\; & a_i^T s = 0,\ i \in Z;\ a_i^T s \leqq 0,\ i \in I(x^k) - Z; \\ & s_j = 0,\ j \in Z;\quad s_j \geqq 0,\ j \in J^-(x^k) - Z; \\ & s_j \leqq 0,\ j \in J^+(x^k) - Z;\ (s^\ell)^T C\, s = 0, \\ & \ell = r, \ldots\ldots, k-1;\ N\ 1,\ 2,\ 3,\ 4 \text{ or } 5 \}. \quad (10.2.10) \end{aligned}$$

If $\lambda_{k-1} = \lambda_{k-1}'$, add i to Z if $a_i^T x^{k-1} < b_i$, $a_i^T x^k = b_i$ and $a_i^T x^\nu = b_i$ for some $\nu < k-1$; add j to Z if $x_j^{k-1} > 0$, $x_j^k = 0$ and $x_j^\nu = 0$ for some $\nu < k-1$, or if $x_j^{k-1} < c_j$, $x_j^k = c_j$ and $x_j^\nu = c_j$ for some $\nu < k-1$; take $r = k$ and calculate $g(x^k)$. The direction-finding problem becomes:

$$\text{Max } \{g(x^k)^T s \mid a_i^T s = 0,\ i \in Z;\ a_i^T s \leqq 0,\ i \in I(x^k) - Z;$$
$$s_j = 0,\ j \in Z;\quad s_j \geqq 0,\ j \in J^-(x^k) - Z;$$
$$s_j \leqq 0,\ j \in J^+(x^k) - Z;\ N\ 1,\ 2,\ 3,\ 4 \text{ or } 5\}. \tag{10.2.11}$$

3. Solve the direction-finding problem, so that the solution s^k is obtained.

4. If $g(x^k)^T s^k > 0$, calculate $C s^k$, λ'_k and λ''_k by means of (10.2.3) and (10.2.4). Determine $\lambda_k = \min(\lambda'_k, \lambda''_k)$. If $\lambda_k = \infty$, stop (infinite solution).
If $g(x^k)^T s^k = 0$ and $Z = O$ (empty set), stop: the problem has been solved; if $g(x^k)^T s^k = 0$ and $Z \neq O$, replace Z by O and solve the direction-finding problem again.

5. Calculate x^{k+1} by means of (10.2.6) and $F(x^{k+1})$ from (10.2.8).

Theorem 2: The method of feasible directions with normalization N 1, 2, 3, 4 or 5, AZ 3 and the conjugation principle will solve the quadratic programming problem in a finite number of steps.
Proof: If $\lambda''_k = \infty$ for some k, (10.2.4) implies that $(s^k)^T C s^k = 0$, hence $C s^k = 0$ (theorem 2, section 2.3), so that $g(x^k + \lambda s^k) = g(x^k)$ for all λ and $F(x^k + \lambda s^k) = F(x^k) + \lambda g(x^k)^T s^k$, so that $F(x^k + \lambda s^k)$ is unbounded as a function of λ. It follows that, if $\lambda_k = \infty$, hence $\lambda''_k = \infty$ for some k, F(x) will be unbounded on R, so that the theorem holds. (Note that no condition C 3 is necessary here). Suppose therefore that $\lambda_k < \infty$ for all k. AZ 3 and the conjugation principle will lead to $g(x^k)^T s_{opt} = 0$ in some direction-finding problem after a finite number of steps. We shall then, by theorem 1, have obtained the maximum of F(x) under the additional condition that equality will hold for a certain set of constraining hyperplanes. If, by replacing the equality sign in some constraints $a_i^T s = 0$ or $s_j = 0$ by the inequality sign, a restart is possible, we shall be able to increase F(x) further, so that we shall never return in the same set of constraining hyperplanes. A second stop will be obtained after a number of steps that is again finite, etc. Finiteness then follows from the fact that the number of sets of constraining hyperplanes to be obtained from the set of all constraining hyperplanes is finite.

Since normalization N 5 differs from the other ones in some respects, we shall also summarize procedure P 2, based on N 5 (procedure P 1 will then be concerned with normalization N 1, N 2, N 3, or N 4):

1. Suppose $x^k \in R$ has already been determined and the final data of the last linear programming sub-problem are available. Suppose further that a set Z of indices i and j as well as an index $r \leqq k - 1$ have already been determined.

2. If $\lambda_{k-1} = \lambda''_{k-1} < \lambda'_{k-1}$, add the constraint $(s^{k-1})^T C(x - x^k) = 0$ to the tableau. The new linear programming sub-problem becomes:

$$\text{Max}\{g(x^r)^T x \mid a_i^T x = b_i \text{ if } i \in Z;\ a_i^T x \leqq b_i \text{ if } i \notin Z;$$

$$0 \leqq x_j \leqq c_j \text{ if } j \notin Z;\ x_j = x_j^k \text{ if } j \in Z;$$

$$(s^\ell)^T C\, x = (s^\ell)^T C\, x^k,\ \ell = r, \ldots, k-1\}. \quad (10.2.12)$$

If $\lambda_{k-1} = \lambda'_{k-1}$, add i to Z if $a_i^T x^{k-1} < b_i$, $a_i^T x^k = b_i$ and $a_i^T x^\nu = b_i$ for some $\nu < k-1$; add j to Z if $x_j^{k-1} > 0$, $x_j^k = 0$ and $x_j^\nu = 0$ for some $\nu < k-1$, or if $x_j^{k-1} < c_j$, $x_j^k = c_j$ and $x_j^\nu = c_j$ for some $\nu < k-1$; take $r = k$ and calculate $g(x^k)$. The linear programming sub-problem becomes:

$$\text{Max}\{g(x^k)^T x \mid a_i^T x = b_i \text{ if } i \in Z;\ a_i^T x \leqq b \text{ if } i \notin Z;$$

$$0 \leqq x_j \leqq c_j \text{ if } j \notin Z;\ x_j = x_j^k \text{ if } j \in Z\}. \quad (10.2.13)$$

3. Solve the linear programming sub-problem, so that the solution $(x^k)'$ is obtained and form $s^k = (x^k)' - x^k$ ($(x^k)'$ may be infinite in which case s^k is an extreme ray to be derived from the final tableau).

4. If $g(x^k)^T s^k > 0$, calculate $C\, s^k$, λ'_k and λ''_k ($\lambda'_k = 1$ if $(x^k)'$ finite, λ''_k follows from (10.2.4)). Determine $\lambda_k = \min(\lambda'_k, \lambda''_k)$. If $\lambda_k = \infty$, stop (infinite solution). If $g(x^k)^T s^k = 0$ and $Z = O$ (empty set), stop: the problem has been solved; if $g(x^k)^T s^k = 0$ and $Z \neq O$, replace Z by O and restart, if possible.

5. Calculate x^{k+1} by means of (10.2.6) and $F(x^{k+1})$ from (10.2.8).

Both procedures, P 1 and P 2, start with $x^0 \in R$ and $Z = O$ (see section 7.2 if such an x^0 is not available). By introducing an arbitrary $\lambda_{-1} = \lambda'_{-1}$ we can use the same scheme for the determination of s^0, i.e. the first direction-finding problem does not contain conjugation-principle constraints. If $\lambda_{k-1} = \lambda''_{k-1} < \lambda'_{k-1}$, a recalculation of the gradient vector is not necessary since for any s satisfying $(s^k)^T C\, s = 0$, we shall have $g(x^k)^T s = g(x^{k-1})^T s$. Therefore we can use $g(x^r)$ instead of $g(x^k)$ as objective vector in (10.2.10) and (10.2.12). As soon as $\lambda_{k-1} = \lambda'_{k-1}$ we need $g(x^k)$ itself as objective vector and since, if $\lambda_{k-2} = \lambda''_{k-2} < \lambda'_{k-2}$, $g(x^{k-1})$ will not be available it will not be possible to use (10.2.7) for it. There is another possibility, however. Suppose we have the final data of the (k-1)-st direction-finding problem of type (10.2.10) or (10.2.12), especially the inverse of the dual basis (which follows immediately from the η-vectors if the primal or dual revised product-form algorithm is applied to the dual problem of (10.2.10) or (10.2.12)). Let this dual basis be denoted by B. If $s^T C\, s^h = 0$ holds in this direction-finding problem (s^h being a known vector, s the vector looked for), the dual variable of this constraint will be unrestricted, so that it is in the dual basis, say at place j_h.

Since $g(x^{k-1}) = g(x^r) - \sum_{h=r}^{k-2} \lambda_h C s^h$ we shall have $B^{-1}g(x^{k-1}) =$

$= B^{-1}g(x^r) - \sum_{h=r}^{k-2} \lambda_h e_{j_h}$, so that we obtain $B^{-1}g(x^{k-1})$, i.e. the transformed $g(x^{k-1})$, by subtracting λ_h from $B^{-1} g(x^r)$ at place j_h for $h = r, \ldots, k-2$. Moreover we have

$B^{-1}g(x^k) = B^{-1}g(x^{k-1}) - \lambda'_{k-1} B^{-1}C s^{k-1}$.

Since $C s^{k-1}$ has already been calculated we only have to multiply it by the η-vectors.

10.3 *Discussion of the Quadratic Programming Methods*

Normalizations N 1 - N 5 lead to different methods for solving the quadratic programming problem. Apart from these 5 different methods, there are already available a number of well-known quadratic programming methods, e.g. those of Beale [3], [4], Frank and Wolfe [20], Markowitz [30], and Wolfe [39].

Theorem 1: Beale's quadratic programming method is also a method of feasible directions using the same conjugation principle. It is equivalent to our method with normalization N 4 but without AZ 3.

The proof will be omitted. It is not difficult but would entail a detailed explanation of Beale's method. It should be remarked that organizationally and computationally Beale's method is quite different. The matrix C is gradually transformed in it, whereas in our method we always refer to the original matrix C.

In our normalizations N 1 - N 4, the direction-finding problems are relatively small but between two problems of this kind the determination of the step length requires an additional operation C s. The direction-finding problems themselves require in general less work for their solution in the order N 1 - N 4 but the number of steps will mostly increase in that order. It may be expected that the decrease of this number when using normalization N 1 will not balance the extra work per step except perhaps in special cases. Procedure P 2 (normalization N 5) will usually lead to a further decrease of the number of steps but at the cost of larger direction-finding problems. If the quadratic function is nearly linear, e.g. if there are only a few quadratic terms, it may be expected that this procedure is fastest. The number of constraints of the linear programming sub-problems will gradually grow if more constraints have to be added of the form $(s^k)^T C s = 0$ but it will never exceed and usually be considerably less than m + n. The number of non-basis variables will always be n.

When using P 1 with normalization N 2, 3 or 4 the number of constraints will usually be less than n. It equals the number of

active hyperplanes plus the number of conjugation-principle constraints. If many of the constraints $x_j = 0$ or $x_j = c_j$ are active, this number will be considerably less than n (except in highly degenerate cases). In all our methods it will always be necessary to do the matrix-times-vector multiplication C s in each iteration. This is not necessary in the other methods but these methods will have to work with much larger linear programming sub-problems. For instance, in Frank and Wolfe's method as well as in Wolfe's method we shall have m + n constraints whereas the number of non-basic variables will be m + 2n. In Beale's method the number of constraints will also be m + n but the number of non-basic variables will be n. In all these methods the matrix-times-vector multiplication can be avoided but is done implicity by means of a gradual transformation of the matrix C. Therefore it can be expected that these methods are superior if the matrix C is full but that the organization of our methods of feasible directions is to be preferred if the objective function contains only a few quadratic terms, a situation which may be expected in most practical problems.

Wolfe's method will require the simplest computer code; only a few changes have to be made in an existing simplex linear programming code. Beale's, and Frank and Wolfe's methods do not have this organizational simplicity but this will be compensated in Beale's method by smaller linear programming sub-problems. The methods of feasible directions will require a still more complicated code but on the other hand smaller sub-problems whereas the possibility of starting at any feasible point may lead to a considerable computational advantage.

11. CONVEX PROGRAMMING

11.1 *Introduction*

In this chapter the general convex programming problem will be considered. It will be shown in section 11.2 how the convergence of the procedures described in section 7.5 can be improved. Section 11.3 will be devoted to the computational aspects of the procedures. In section 11.4 the procedures will be discussed and some attention will be paid to two other methods for maximizing a concave non-linear function subject to linear constraints, namely Frank and Wolfe's method and Rosen's gradient-projection method. They will be shown to be equivalent to methods of feasible directions.

11.2 *Convex Programming Procedures*

In section 7.5 we have given a number of convex programming procedures. The procedures were different in the way the normalization of the feasible directions was chosen and in the way zigzagging was prevented. In section 7.6 we have seen that all these procedures will lead to a sequence of trial solutions x^k with monotonously increasing values for the objective function $F(x)$ which moreover will converge to the maximum of $F(x)$ on the convex set R. It may be expected, however, that, if the objective function is highly non-linear, the convergence of the procedures will sometimes be rather slow. In the case of a quadratic objective function it appeared to be possible to obtain a finite method by using the principle of conjugated directions and by using another anti-zigzagging precaution (AZ 3). If the objective function is an arbitrary concave function, it may be expected that application of the same principles will improve the convergence.

In the quadratic case we added the relation $(s^k)^T C\ s = 0$ to the direction-finding problem if $\lambda_k = \lambda'_k < \lambda_k$ had to be chosen and dropped these extra constraints at a new choice $\lambda_\ell = \lambda'_\ell$ for some $\ell > k$. The relation $(s^k)^T C\ s = 0$ is equivalent to $\{g(x^{k+1}) - g(x^k)\}^T s = 0$ and this constraint can equally well be added in the case of a more general non-linear objective function. It may be expected that such an addition will prevent a great number of relatively small steps far from the optimum caused by rapid changes of the gradient vector.

Another way of efficiently preventing this zigzagging if normalization N 1, 2, 3 or 4 is applied is by an anti-zigzagging precaution and AZ 3 seems to be the most efficient procedure in this respect. But it can not easily be applied in the case of non-linear

constraints while moreover it does not seem to guarantee convergence to a maximum in all possible cases. Therefore we propose to use a combination of AZ 2 and AZ 3.

The two convex programming procedures finally proposed for the problem of maximizing a concave function in a convex region are the following:

Procedure P 3:

1. Find some $x^0 \in R$ (see section 7.2 if such a point is not available), take $\varepsilon_0 > 0$ arbitrarily and let $\varepsilon' > 0$ be chosen so small that it is reasonable to stop if the increase in value for the objective function is less than ε'. Take further $Z = O$ (empty set) and $\varepsilon'' > 0$.

2. Suppose points $x^0, x^1, \ldots, x^k$ ($x^h \in R$, $F(x^h) > F(x^{h-1})$), a number ε_k, and a set $Z \subset I_L(x^k) + J^-(x^k) + J^+(x^k)$ of indices i and j have already been determined. We now perform the following operations to find x^{k+1}:

a. Calculate $g^k = g(x^k)$.

b. If $\lambda_{k-1} = \lambda''_{k-1} < \lambda'_{k-1}$ ($\lambda_1 = \lambda'_1$ by definition), add the constraint $(g^k - g^{k-1})^T s = 0$ to the direction-finding problem, so that this becomes:

$$\text{Max}\{\sigma \mid (s, \sigma) \in S'(x^k, \varepsilon_k);\ n_h^T s = 0 \text{ if } h \in Z;$$
$$(g^{l+1} - g^l)^T s = 0 \text{ for } l = r, \ldots, k-1;$$
$$-(g^k)^T s + \sigma \leqq 0;\ \text{N } 1,\ 2,\ 3 \text{ or } 4\}, \qquad (11.2.1)$$

where $S'(x^k, \varepsilon_k)$ is defined by (7.5.7); $n_h = a_i$, e_j or $-e_j$, and r is such that $\lambda_{r-1} = \lambda'_{r-1}$ but $\lambda_h = \lambda''_h < \lambda'_h$ for $r \leqq h \leqq k-1$.

If $\lambda_{k-1} = \lambda'_{k-1}$, add h to Z if $n_h^T x^{k-1} < b_h$, $n_h^T x^k = b_h$ and $n_h^T x^\nu = b_h$ for some $\nu < k-1$ (if $n_h = -e_j$, then $b_h = 0$, if $n_h = e_j$, then $b_h = c_j$). The direction-finding problem becomes:

$$\text{Max}\{\sigma \mid (s, \sigma) \in S'(x^k, \varepsilon_k);\ n_h^T s = 0 \text{ if } h \in Z;$$
$$-(g^k)^T s + \sigma \leqq 0;\ \text{N } 1,\ 2,\ 3 \text{ or } 4\}. \qquad (11.2.2)$$

c. Solve the direction-finding problem, so that the bounded solution s^k, σ_k is obtained (bounding must be performed afterwards, if necessary).

d. If $\sigma_k < \varepsilon''$ and $Z \neq O$, replace Z by O and solve (11.2.1) or (11.2.2) again;
if $\sigma_k < \varepsilon''$, $Z = O$ and there are relations of the type $(g^{l+1} - g^l)^T s = 0$ in the direction-finding problem, drop these relations and solve the direction-finding problem again;
if $\sigma_k < \varepsilon''$, $Z = O$ and there are no relations of the type $(g^{l+1} - g^l)^T s = 0$, or if $\sigma_k > \varepsilon''$, perform step 2 e.

e. If $\sigma_k < \varepsilon_k$, replace $S'(x^k, \varepsilon_k)$ in (11.2.1) or (11.2.2) by $S'(x^k, \frac{\varepsilon_k}{2})$, write ε_k for $\frac{\varepsilon_k}{2}$ and solve (11.2.1) or (11.2.2) again, if possible, but

if $\sigma_k = 0$ and $S'(x^k, \varepsilon_k) = S'(x^k)$, stop since x^k will be a maximum solution;
if $\sigma_k \geqq \varepsilon_k$, take $\varepsilon_{k+1} = \varepsilon_k$ and perform step 2 f.

f. Determine λ_k by means of (7.4.1) and (7.4.4). If $\lambda_k = \infty$, stop (infinite solution).

g. Calculate $x^{k+1} = x^k + \lambda_k s^k$, $\Delta F(x^k)$ and $F(x^{k+1})$.

h. If $\Delta F(x^k) < \varepsilon'$, $Z = O$ and the direction-finding problem does not contain any relations $(g^{l+1} - g^l)^T s = 0$, stop;
if $\Delta F(x^k) < \varepsilon'$, $Z \neq O$ or the direction-finding problem contains relations of the type $(g^{l+1} - g^l)^T s = 0$, replace Z by O, omit the relations $(g^{l+1} - g^l)^T s = 0$ in the next direction-finding problem (i.e. act as if $\lambda_k = \lambda'_k$ holds) and repeat steps 2 a-h;
if $\Delta F(x^k) \geqq \varepsilon'$, repeat steps 2 a-h.

Procedure P 4:

1. Find some $x^0 \in R$ (section 7.2), choose $\varepsilon' > 0$ so small that it is reasonable to stop if the increase in value for the objective function is less than ε'.

2. Suppose $x^0, x^1, \ldots, x^k$ ($x^h \in R$, $F(x^h) > F(x^{h-1})$) have already been determined. We now perform the following operations to find x^{k+1}:

a. Calculate $g^k = g(x^k)$.

b. If $\lambda_{k-1} = \lambda''_{k-1} < \lambda'_{k-1}$ ($\lambda_{-1} = \lambda'_{-1}$ by definition), solve the linear programming sub-problem:

$$\begin{aligned} \text{Max}\{x_0 \mid\ & q_i(x^k)^T x + \Theta_i x_0 \leqq q_i(x^k)^T x^k + b_i - f_i(x^k),\ i \in I_C; \\ & a_i^T x \leqq b_i,\ i \in I_L;\ 0 \leqq x_j \leqq c_j,\ j \in J; \\ & (g^{l+1} - g^l)^T x = (g^{l+1} - g^l)^T x^k,\ l = r, \ldots, k-1; \\ & -g(x^k)^T x + x_0 \leqq -g(x^k)^T x^k\}, \end{aligned} \tag{11.2.3}$$

where r is such that $\lambda_{r-1} = \lambda'_{r-1}$ but $\lambda_h = \lambda''_h < \lambda'_h$ for $r \leqq h \leqq k-1$. If $\lambda_{k-1} = \lambda'_{k-1}$, solve the linear programming sub-problem:

$$\begin{aligned} \text{Max}\{x_0 \mid\ & q_i(x^k)^T x + \Theta_i x_0 \leqq q_i(x^k)^T x^k + b_i - f_i(x^k),\ i \in I_C; \\ & a_i^T x \leqq b_i,\ i \in I_L;\ 0 \leqq x_j \leqq c_j,\ j \in J; \\ & -g(x^k)^T x + x_0 \leqq -g(x^k)^T x^k\}. \end{aligned} \tag{11.2.4}$$

c. If the linear programming sub-problem has an infinite solution, derive s^k from the extreme ray which leads to infinity and perform step 2 d;
if the linear programming sub-problem has a finite solution $(x^k)'$, $(x_0^k)'$ and $(x_0^k)' = 0$:
stop if there were no relations of the type $(g^{l+1} - g^l)^T(x - x^k) = 0$ (x^k will be a maximum solution), but
omit all relations of this type if there were any in the linear programming sub-problem and solve the linear programming sub-problem again;
if $(x_0^k)' > 0$, take $s^k = (x^k)' - x^k$.

d. Determine λ_k by means of (7.4.1) and (7.4.4). If $\lambda_k = \infty$, stop (infinite solution).

e. Calculate $x^{k+1} = x^k + \lambda_k s^k$, $\Delta F(x^k)$ and $F(x^{k+1})$.

f. If $\Delta F(x^k) < \varepsilon'$ and the sub-problem did not contain relations of the type $(g^{l+1} - g^l)^T s = 0$, stop;
if $\Delta F(x^k) < \varepsilon'$ and the sub-problem contains relations of this type, repeat steps 2 a-f assuming $\lambda_k = \lambda'_k$ (so that the next sub-problem is of the type (11.2.4);
if $\Delta F(x^k) \geqq \varepsilon'$, repeat steps 2 a-f.

Theorem 1: If F(x) and R satisfy conditions C 3 and C 1 respectively, and if $\varepsilon' \neq 0$, then procedure P 3 will generate a sequence of points $x^k \in R$ with $\lim F(x^k) = m = \max \{F(x) \mid x \in R\}$.
Proof: Condition C 3 will ensure that, if $\lambda_k = \infty$ for some k, $F(x^k + \lambda s^k)$ will be unbounded as a function of λ. If the sequence of points x^k is infinite and either unbounded or $\sigma_k \geqq \delta$ for $\delta > 0$ and some infinite sub-sequence, then $\lim F(x^k) = \infty$, so that the theorem holds. Hence, if $m < \infty$, σ_k will tend to zero, so that $\sigma_k < \varepsilon''$ will hold for all $k >$ some number $k(\varepsilon'')$. Consequently, if $k > k(\varepsilon'')$ there will be no difference between procedures P 3 and P 1, so that the proof is equivalent to the proof of theorem 1, section 7.6.

Theorem 2: If F(x) and R satisfy conditions C 3 and C 1 respectively and if $\varepsilon' = 0$, then procedure P 4 will generate a sequence of points $x^k \in R$ with $\lim F(x^k) = m = \max \{F(x) \mid x \in R\}$.
Proof: Suppose, for $h \geqq r$, that $\lambda_{h-1} = \lambda''_{h-1} < \lambda'_{h-1}$ and that quantities u_h can be found such that $\sum_{h \geqq r} u_h s^h = 0$. It follows that

$\sum_{h \geqq r} u_h (g^{r+1} - g^r)^T s^h = 0$ or $u_r (g^{r+1} - g^r)^T s^r = 0$. Since

$(g^r)^T s^r > 0$ and $(g^{r+1})^T s^r = 0$ we have that $u_r = 0$. In the same way $u_{r+1} = 0$, $u_{r+2} = 0$ etc. Hence the vectors s^k are linearly independent, so that $\lambda_{h-1} = \lambda''_{h-1} < \lambda'_{h-1}$ cannot hold indefinitely. Therefore, the conjugation-principle constraints will be dropped at regular times since we obtain either $(x_0^k)' = 0$ or $\lambda_k = \lambda'_k$ for some k. If we only consider this sub-sequence, then we can follow the proof of theorem 2, section 7.6.

Theorem 3: If F(x) satisfying condition C 3 is bounded on R and if $\varepsilon' > 0$, then the procedures P 3 and P 4 lead to a finite number of steps (i.e. after a finite number of steps $\Delta F(x^k) < \varepsilon$ is obtained, while the direction-finding problem does not contain extra requirements due to AZ 3 - only in P 3 - or the conjugation principle).
Proof: If F(x) is bounded, $\Delta F(x^k) \geqq \varepsilon'$ cannot hold indefinitively. If $\Delta F(x_k) < \varepsilon'$ and the direction-finding problem contains extra requirements due to AZ 3 or the conjugation principle, then the next direction-finding problem will not contain such requirements. It follows that after a finite number of steps we shall be sure that $\Delta F(x^k) < \varepsilon'$ while the direction-finding problem does not contain extra requirements due to AZ 3 or the conjugation principle.

Hence the calculation stops after a finite number of steps. This proves the theorem.

The question arises how the near-optimality of the final solution x^k can be checked. This can be done by linearizing the problem in the point x^k, so that we obtain the linear programming problem:

$$\text{Max}\{g(x^k)^T x \mid q_i(x^k)^T x \leqq q_i(x^k)^T x^k + b_i - f_i(x^k),\ i \in I_C;$$
$$a_i^T x \leqq b_i,\ i \in I_L;\ 0 \leqq x \leqq c\}. \qquad (11.2.5)$$

Let the optimal solution of this problem be denoted by x'. By virtue of theorem 8, section 2.6 we know that

$$\text{Max}\{F(x) \mid x \in R\} - F(x^k) \leqq g(x^k)^T x' - g(x^k)^T x^k. \qquad (11.2.6)$$

Hence, if a point $\bar{x} \in R$ is considered to be near-optimal if $m - F(\bar{x}) \leqq \bar{\varepsilon}$ for some predetermined $\bar{\varepsilon} > 0$, then x^k will be near-optimal if $g(x^k)^T x' - g(x^k)^T x^k \leqq \bar{\varepsilon}$ holds. This condition is sufficient but not necessary. In fact, it is easy to construct examples in which (11.2.5) has an infinite solution for each k although it may be doubted whether this will ever happen in a practical problem ($c_j = \infty$ for some j must hold in that case). The following theorem holds, however:

Theorem 4: If the sequence x^k converges to a point $\bar{x}$ with $F(\bar{x}) = m$ and if s^k is a sequence of vectors of unit length directed for each k from x^k to the solution $(x^k)'$ of (11.2.5) (or, if $(x^k)'$ is infinite, directed along the extreme ray of (11.2.5) which indicates the infinite solution), then $g(x^k)^T s^k$ will tend to zero.

Proof: Since $\bar{x}$ is optimal we know that non-negative numbers $\bar{u}_i$, $\bar{v}_j^-$ and $\bar{v}_j^+$ can be found such that $g(\bar{x}) = \Sigma \bar{u}_i q_i(\bar{x}) + \Sigma \bar{u}_i a_i - \Sigma \bar{v}_j^- e_j + \Sigma \bar{v}_j^+ e_j$, $\bar{u}_i = 0$ if $i \notin I_C(\bar{x}) + I_L(\bar{x})$, $\bar{v}_j^- = 0$ if $\bar{x}_j > 0$, and $\bar{v}_j^+ = 0$ if $\bar{x}_j < c_j$. We can write for each k: $g(x^k) = \Sigma \bar{u}_i q_i(x^k) + \Sigma \bar{u}_i a_i - \Sigma \bar{v}_j^- e_j + \Sigma \bar{v}_j^+ e_j + h(x^k)$. From the continuity of $g(x^k)$ and $q_i(x^k)$ it follows that the vectors $h(x^k)$ are such that $\lim_{k \to \infty} h(x^k) = 0$, so that $h(x^k)^T s^k$ tends to zero. For each small positive number ε a $k(\varepsilon)$ can be found so that, for all $k \geqq k(\varepsilon)$ we have $n_h^T s^k \leqq \varepsilon$ for $h \in H(\bar{x})$. Hence $0 < g(x^k)^T s^k \leqq \varepsilon\{\Sigma \bar{u}_i + \Sigma \bar{v}_j^- + \Sigma \bar{v}_j^+\} + h(x^k)^T s^k$. This proves the theorem.

Hence, if $g(x^k)^T s^k \geqq \varepsilon_1$ holds for the final solution x^k of the convex programming procedure, where s^k is defined as in theorem 4 and ε_1 is some predetermined number, then we do better to continue the calculation after replacing ε' by, say $\frac{\varepsilon'}{2}$. If $g(x^k)^T s^k < \varepsilon_1$, there are two possibilities:

1. We know beforehand that, if λ and the vector $\bar{s}$ of unit length

are such that $\bar{x} = \bar{x}^k + \lambda \bar{s}$, then λ_1 is an upper limit for λ. In that case we can take $\varepsilon_1 = \frac{\bar{\varepsilon}}{\lambda_1}$ and are sure of the near-optimality if $g(x^k)^T s^k < \varepsilon_1$. If $c_j < \infty$ for all j, for instance, we can take $\lambda_1^2 = c^T c$.

2. We cannot find λ_1 beforehand (this can hardly be expected to occur in any practical problem). Instead of (11.2.5) we could now solve the problem:

$$\begin{aligned} \text{Max } \{ g(x^k)^T x \mid\ & q_i(x^k)^T x \leqq q_i(x^k)^T x^k + b_i - f_i(x^k),\ i \in I_C; \\ & a_i^T x \leqq b_i,\ i \in I_L;\ 0 \leqq x \leqq c; \\ & q_i(x^l)^T x \leqq q_i(x^l)^T x^l \text{ for } l = 1, 2, \ldots, k-1 \text{ and } i \in I_C(x^l); \\ & -g(x^l)^T x \leqq -g(x^l)^T x^l \text{ for } l = 1, 2, \ldots, k-1 \}, \end{aligned} \tag{11.2.7}$$

i.e. we retain all tangent planes to the hypersurfaces $f_i(x) = b_i$ in the points x^l and all tangent planes to the hypersurfaces $F(x) = F(x^l)$. Clearly $\bar{x}$ satisfies all constraints in (11.2.7). If (11.2.7) has an optimum solution $(x^k)''$, then:
if $g(x^k)^T (x^k)'' - g(x^k)^T x^k < \bar{\varepsilon}$, we can stop the calculation;
if $g(x^k)^T (x^k)'' - g(x^k)^T x^k \geqq \bar{\varepsilon}$, we shall continue the calculation.

Problem (11.2.7) is a linear programming problem with a great number of rows but with n non-basic variables. The dual revised product-form algorithm can therefore easily be applied to it.

11.3 *Computational Aspects*

Several remarks can be made with respect to the computational aspects of the two procedures described in the previous section.

1. If there are equations among the linear constraints of the convex programming problem, then we can either eliminate some of the variables beforehand or we can start with a point $x^0 \in R$ and require $a_i^T s = 0$ instead of $\leqq 0$ in all direction-finding problems. In the former case the number of variables will be reduced but on the other hand a special, simple structure of the functions $F(x)$, $f_i(x)$ and $a_i^T x$ may be destroyed or there may be more non-zero coefficients in these functions. Equations among the non-linear constraints can hardly be handled since in that case the feasible region R will no longer be convex (except in exceptional cases, where it will be convex but without satisfying regularity condition C 1).

2. When all constraints are linear, the introduction of the extra variable σ is not necessary. Instead of σ we then maximize $g(x^k)^T s$. The procedures become computationally much simpler since there are no recalculations of vectors $q_i(x)$, so that the transition from one direction-finding problem to the next one is easier, and since the determination of λ_k^l is much quicker.

3. If $\lambda_{k-1} = \lambda''_{k-1} < \lambda'_{k-1}$, i.e. if the relation $\{g(x^k) - g(x^{k-1})\}^T s = 0$ is added to the direction-finding problem, it is not necessary to change the relation $-g(x^{k-1})^T s + \sigma \leqq 0$ into $-g(x^k)^T s + \sigma \leqq 0$, so that the gradient vector need not be recalculated.

4. If in P 3 $b_i - \varepsilon_k \leqq f_i(x^k) < b_i$ holds for some i, so that the extra requirement $q_i(x^k)^T s + \Theta_i \sigma \leqq 0$ should be added, and if the previous direction-finding problem contained the constraint $q_i(x^l)^T s + \Theta_i \sigma \leqq 0$ for some $l \leqq k-1$ and the same i, then we need not calculate the normal $q_i(x^k)$ but can maintain the requirement $q_i(x^l)^T s + \Theta_i \sigma \leqq 0$. A recalculation is only necessary if $f_i(x^k) = b_i$ holds. This will reduce the number of rows to be recalculated and makes an application of AZ 1 less time-consuming.

5. Convergence of the procedures can be speeded up by an appropriate choice of the quantities Θ_i. These quantities can be changed during the calculation. Computational experience with a certain type of problems will undoubtedly show how they can most efficiently be changed.

6. The final data of a direction-finding problem can always be used as initial data for the next problem. If the revised product-form algorithm is applied to the dual of the direction-finding problem, either in its primal form (with N 2 or N 3) or in its dual form (with N 4 or N 5), then it is not difficult to use the final set of η-vectors of the previous problem as an initial set for the next problem. But, the greater the number of changes that have to be made in the direction-finding problem, the more advantageous it becomes to start the next problem with a pre-inversion. See further section 6.3 where the various possible changes are described in the last two paragraphs.

7. When applying procedure P 4 convergence may be speeded up by taking the following linear programming sub-problems:

$$\begin{aligned}\text{Max } \{x_0 \mid\ & q_i(x^k)^T x + \Theta_i x_0 \leqq q_i(x^k)^T x^k + b_i - f(x^k),\ i \in I_C;\\ & a_i^T x \leqq b_i,\ i \in I_L;\ 0 \leqq x \leqq c;\\ & -g(x^k)^T x + x_0 \leqq -g(x^k)^T x^k;\\ & q_i(x^l)^T x \leqq q_i(x^l)^T x^l \text{ for } l = 1, 2, \ldots, k-1 \text{ and } i \in I_C(x^l);\\ & -g(x^l)^T x \leqq -g(x^l)^T x^l \text{ for } l = 1, 2, \ldots, k-1\}. \qquad (11.3.1)\end{aligned}$$

The constraint set, defined by (11.3.1), contains the set $\{x \in R \mid F(x) \geqq F(x^k)\}$, so that the extra constraints cannot do any harm. Anti-zigzagging requirements of the type $\{g(x^{l+1}) - g(x^l)\}^T (x - x^k) = 0$ can also easily be added here. If a problem of type (11.3.1) has been solved, it is not difficult to obtain the solution of (11.2.7). We only have to replace all the Θ_i by 0 and can easily find the corresponding changes in the final data of (11.3.1). This makes the near-optimality check less expensive.

We finally remark that it is not necessary to take all constraints $f_i(x) \leqq b_i$ into account when making a sub-problem in P 4.

We could restrict ourselves to those for which $b_i - \varepsilon \leqq f_i(x^k) \leqq b_i$, ε being some small positive number.

11.4 *Discussion of the Procedures*

Depending on the normalization we have proposed five different methods for solving the convex programming problem. In this section we shall try to make a brief comparison. We must then distinguish between the case of only linear constraints and the general case of an arbitrary constraint set. In the latter case the determination of the step lengths λ_k will be a considerably more expensive operation since $\lambda_k^!$ cannot be obtained as easily as in the case of only linear constraints. It will therefore be advantageous to have a method which will generally lead to fewer steps even if the amount of work per step will be more. Therefore procedure P 3 with normalization N 1 may be preferred in some cases, especially in those non-linear programming problems in which the number of constraints is relatively small and in which these constraints are highly non-linear and are composed of many non-zero terms. But in most problems we can expect that the extra amount of work per step does not balance the reduction in the number of steps, owing to the rather bad restart possibilities of this normalization. If there are many constraints which contain a few terms, if the non-linear constraints are mainly composed of linear terms and if the objective function does not contain many non-linear terms, hence if the greater part of the problem is linear then procedure P 4, hence normalization N 5, may be the best. This procedure can moreover be modified as explained in section 11.3, point 7. If the constraints and the objective function are non-linear to a larger extent, then procedure P 3 with normalization N 2, N 3 or N 4 may have to be preferred. In this order the number of steps will usually increase but the amount of work per step will decrease.

If all constraints are linear but the objective function is non-linear, then we can compare our methods with some existing ones, for instance those developed by Frank and Wolfe [20], Rosen [34] and Dennis [17], chapter 7.

We shall have:

Theorem 1: Procedure P 2, applied to a problem with only linear constraints, is equivalent to Frank and Wolfe's non-linear programming method.

Proof: In Frank and Wolfe's method two sequences of feasible points are distinguished: a sequence z^k and a sequence x^k. The starting point is a vertex $x^0 = z^0 \in R$. At the k-th step the linear programming sub-problem $\mathrm{Max}\{g(x^k)^T x \mid x \in R\}$ is solved by applying the simplex method to it, so that the optimum solution x^{k+1} is obtained (or an extreme ray leading to infinity). Then $F\{z^k + \lambda(x^{k+1} - z^k)\}$ is maximized under the condition that $z^{k+1} = z^k + \lambda_k(x^{k+1} - z^k) \in R$, i.e. under the condition $\lambda \leqq 1$ if x^{k+1} is finite. It is clear that this procedure is exactly equivalent to our

procedure P 2, where the points z^k and x^k are called x^k and $(x^k)'$, respectively.

Our procedure P 2 will solve the linear programming sub-problems by taking the point z^k of Wolfe's method as starting point for the next problem and by then applying a method of feasible directions to the sub-problems (e.g. the one with normalization N 4, which is a direct extension of the simplex method). It can also be applied to problems with non-linear constraints; its convergence can considerably be improved by the conjugation principle, so that we obtain procedure P 4, and possibly by the extensions of section 11.3, point 7; it will moreover be finite in the case of a quadratic objective function as we have seen in chapter 10. Therefore it may be considered as an extension of Frank and Wolfe's method.

Rosen's gradient-projection method can also be considered as a method of feasible directions. In this method the projection s' of the gradient vector g onto the cone $\{s \mid a_i^T s = 0,\ i \in I(x);\ s_j = 0,\ j \in J^-(x) + J^+(x)\}$ is obtained. If $g^T s' > 0$, s' is taken as usable feasible direction. If $g^T s' = 0$, then one of the equations, chosen by considering the dual variables, is replaced by the weaker inequality which will then in the non-degenerate case lead to a feasible vector s_1' with $g^T s_1' > 0$. Step lengths are determined in the same way as in our procedures. Convergence of $F(x^k)$ to m can be proved if the sets $I(x)$, $J^-(x)$ and $J^+(x)$ which determine the cones are replaced by $I(x, \varepsilon)$, $J^-(x, \varepsilon)$ and $J^+(x, \varepsilon)$, where $\varepsilon > 0$ is gradually reduced as in our procedures. The computational work needed to find the vector s' is of the same nature as in our P 1 with normalization N 1. Hence, the discussions in section 8.6 about normalization N 1 also apply to the gradient-projection procedure. The only additional remark is that the vectors s' of the gradient-projection method are not the feasible directions making the smallest possible angle with the gradient vector, so that it can be expected that the number of steps in the gradient projection method is higher than in P 1 with N 1. This will especially be true if, owing to the requirements $a_i^T s = 0$ or $s_j = 0$ instead of $a_i^T s \leqq 0$ or $s_j \geqq 0$ ($\leqq 0$), we are forced to stay in a hyperplane and to make many small steps in it, whereas this hyperplane does not contain the optimum point.

Dennis' method is completely equivalent to our P 1 with N 1. However, Dennis does not mention an extension to problems involving non-linear constraints and does not give an efficient computational scheme for solving the quadratic direction-finding problems of type (8.2.1).

Still another method which uses a principle comparable to our P 1 with normalization N 1 is Frisch's multiplex method [21]. This method has primarily been developed for the linear programming problem. The problem of projecting the gradient vector onto the cone of feasible vectors is not fully discussed in it.

REFERENCES

1. K.J. Arrow, L. Hurwicz and H. Uzawa, (editors), Studies in Linear and Non-Linear Programming, Stanford University Press, Stanford, Cal., 1958.
2. E.M.L. Beale, An Alternative Method of Linear Programming, Proc. Cambr. Phil. Soc., 50 (1954) 513-523.
3. E.M.L. Beale, On Minimizing a Convex Function Subject to Linear Inequalities, J. Roy. Statist. Soc. (B), 17 (1955) 173-184.
4. E.M.L. Beale, On Quadratic Programming, Nav. Res. Log. Quart., 6 (1959) 227-243.
5. R. Bellman, Dynamic Programming, Princeton University Press, Princeton, N.J., 1957.
6. J.F. Benders, Partitioning in Mathematical Programming, Thesis, Utrecht, 1960 (in the press).
7. A. Charnes, Optimality and Degeneracy in Linear Programming, Econometrica, 20 (1952) 160-170.
8. A. Charnes and C.E. Lemke, Computational Theory of Linear Programming, Part I - The "Bounded Variables" Problem. Office of Naval Research, Research Memorandum No. 10, Carnegie Institute of Technology, 7 Jan. 1954.
9. A. Charnes and C.E. Lemke, Minimization of Non-Linear Separable Convex Functions, Nav. Res. Log. Quart., 1 (1954) 301-312.
10. E.W. Cheney and A.A. Goldstein, Newton's Method for Convex Programming and Tchebycheff Approximation, Numerische Mathematik, 1 (1959) 253-268.
11. G.B. Dantzig, Computational Algorithm of the Revised Simplex Method, The Rand Corporation, Santa Monica, Cal., RM-1266, 26 October 1953.
12. G.B. Dantzig, Upper Bounds, Secondary Constraints and Block Triangularity in Linear Programming, Econometrica, 23 (1955) 174-183.
13. G.B. Dantzig, L.R. Ford, Jr. and D.R. Fulkerson, A Primal-Dual Algorithm for Linear Programs, in: "Linear Inequalities and Related Systems", Annals of Mathematics Studies, No. 38, Princeton Univ. Press, Princeton, N.J., (1956), 171-181.
14. G.B. Dantzig and Wm. Orchard Hays, Alternate Algorithm for the Revised Simplex Method Using Product Form for the Inverse, The Rand Corporation, Santa Monica, Cal., RM-1268, 19 November 1953.

15. G.B.Dantzig, A.Orden and P.Wolfe, The Generalized Simplex Method for Minimizing a Linear Form under Linear Inequality Constraints, Pacific Journal of Mathematics, 5 (1955) 183-195.

16. G.B.Dantzig and P.Wolfe, Decomposition Principle for Linear Programs, J.Oper.Res.Soc.Am., 8 (1960) 101-111.

17. J.B.Dennis, Mathematical Programming and Electrical Networks, J.Wiley and Sons, New York, 1959.

18. J.Farkas, Theorie der einfachen Ungleichungen, J. für reine und angewandte Mathematik, 124 (1902) 1-27.

19. L.Ford Jr. and D.R.Fulkerson, A Primal-Dual Algorithm for the Capacitated Hitchcock Problem, Nav.Res.Log.Quart., 4 (1957) 47-54.

20. M.Frank and P.Wolfe, An Algorithm for Quadratic Programming, Nav.Res.Log.Quart., 3 (1956) 95-110.

21. R.A.K.Frisch, The Multiplex Method for Linear Programming, Memo.Univ.Inst. of Economics, Oslo, 17 October 1955.

22. S.I.Gass, Linear Programming, Methods and Applications, Mc Graw-Hill Book Company, New York, 1958.

23. S.I.Gass and T.Saaty, The Computational Algorithm for the Parametric Objective Function, Nav. Res. Log.. Quart., 2 (1955) 39-45.

24. R.A.Good, Systems of Linear Relations, Soc.Ind.Appl.Math. Review, 1 (1959) 1-31.

25. M.R.Hestenes and E.Stiefel, Methods of Conjugate Gradients for Solving Linear Systems, J. Res. Nat. Bur. Standards, 49 (1952) 409-436.

26. C.Hildreth, A Quadratic Programming Procedure, Nav.Res. Log.Quart., 4 (1957) 79-85.

27. W.S.Jewell, Optimal Flow through Networks, Interim Technical Report no. 8, June 1958, Massachusetts Institute of Technology.

28. H.W.Kuhn and A.W.Tucker, Non-Linear Programming, in: "Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability", Berkeley, Cal., 1950, 481-492.

29. C.E.Lemke, The Dual Method of Solving the Linear Programming Problem, Nav.Res.Log.Quart., 1 (1954) 36-47.

30. H.M.Markowitz, The Optimization of a Quadratic Function Subject to Linear Constraints, Nav.Res.Log.Quart., 3 (1956) 111-133.

31. H.M.Markowitz, The Elimination Form of the Inverse and Its Application to Linear Programming, Management Science, 3 (1957) 255-269.

32. Wm. Orchard-Hays, Evolution of Computer Codes for Linear Programming, The Rand Corporation, Santa Monica, Cal., P-810, 14 March 1956.

33. V. Riley and S. I. Gass, Linear Programming and Associated Techniques, The John Hopkins Press, Baltimore, Md., 1958.

34. J. B. Rosen, The Gradient Projection Method for Nonlinear Programming, Part I - Linear Constraints, J. Soc. Indust. and Appl. Maths., 8 (1960) 181-217.

35. A. W. Tucker, Dual Systems of Homogeneous Linear Relations, in: "Linear Inequalities and Related Systems", Annals of Mathematics Studies, No. 38, Princeton Univ. Press, Princeton, N. J., 1956, 3-18.

36. S. Vajda, Theory of Games and Linear Programming, J. Wiley and Sons, New York, 1956. Methuen and Co. Ltd., London, 1956.

37. H. M. Wagner, A Comparison of the Original and Revised Simplex Method, J. Oper. Res. Soc. Am., 5 (1957) 361-369.

38. H. M. Wagner, The Dual Simplex Algorithm for Bounded Variables, Nav. Res. Log. Quart., 5 (1958) 257-261.

39. P. Wolfe, The Simplex Method for Quadratic Programming, Econometrica, 27 (1959) 382-398.

40. G. Zoutendijk, Maximizing a Function in a Convex Region, J. R. Statist. Soc. (B), 21 (1959) 338-355.

SAMENVATTING

METHODEN VAN TOELAATBARE RICHTINGEN, EEN STUDIE IN LINEAIRE EN NIET-LINEAIRE PROGRAMMERING

Het probleem dat in het laatste deel van dit proefschrift (hoofdstuk 7-11) bestudeerd wordt betreft het maximaliseren van een concave, differentieerbare functie in een gesloten convex gebied. Voor dit zogenaamde convexe programmeringsprobleem worden een aantal oplossingsprocedures gegeven die Methoden van Toelaatbare Richtingen genoemd worden. Het zijn iteratieve procedures die de volgende eigenschappen bezitten:

1. Gestart wordt met een punt binnen het convexe gebied. Aangegeven wordt hoe een dergelijk punt verkregen kan worden indien het niet onmiddellijk beschikbaar is.
2. In iedere iteratie wordt een nieuw punt binnen het convexe gebied bepaald met een hogere waarde voor de te maximaliseren functie. Een iteratie bestaat uit twee gedeelten:

a. de bepaling van een bruikbare toelaatbare richting, d.i. een richting waarin men vanuit het reeds verkregen punt een stap kan maken zodanig dat het convexe gebied niet verlaten wordt en de waarde van de functie stijgt;
b. de bepaling van de lengte van de stap die in de gevonden richting gemaakt zal worden.

3. De verkregen rij van punten is zodanig dat de corresponderende functiewaarden convergeren naar het maximum binnen het convexe gebied (zo dit bestaat).

De methoden van toelaatbare richtingen zijn speciaal ontwikkeld om de oplossing van grote convexe programmeringsproblemen met behulp van elektronische rekenmachines mogelijk te maken. Veel aandacht is derhalve besteed aan de numerieke aspecten. De verschillende methoden verschillen onderling in de wijze waarop de bruikbare toelaatbare richtingen gevonden worden. Deze richtingsproblemen blijken in bijna alle methoden lineaire programmeringsproblemen te zijn. Daarom is in deel twee, hoofdstuk 3 - 6, van het proefschrift een uitgebreide studie gemaakt van de welbekende simplex methode. Van deze methode bestonden reeds drie numerieke versies waaraan een vierde wordt toegevoegd. De vier versies worden vergeleken aan de hand van een aantal veronderstellingen aangaande het gedrag van in de praktijk voorkomende grote lineaire programmeringsproblemen. Duidelijk blijkt dat de nieuwe algoritme tot de kleinste rekentijden leidt. Deze algoritme is derhalve gekozen ter oplossing van de richtingsproblemen in de methoden van toelaatbare richtingen. In hoofdstuk 6 wordt in het bijzonder bestudeerd tot welke soort problemen dit leidt.

Van alle methoden van toelaatbare richtingen wordt de convergentie bewezen. Zij blijken eindig te zijn in het geval van een lineaire of kwadratische functie welke gemaximaliseerd moet worden onder een aantal lineaire bijvoorwaarden. In het eerste geval hebben we te maken met een lineair programmeringsprobleem. In hoofdstuk 9 dat hieraan gewijd is wordt aangetoond dat de simplex methode en de z.g. "primal-dual" methode equivalent zijn met methoden van toelaatbare richtingen. Tevens wordt in dit hoofdstuk aangegeven hoe een methode van toelaatbare richtingen van dienst kan zijn bij het oplossen van zeer grote lineaire programmeringsproblemen.

In het geval van een kwadratische functie en lineaire bijvoorwaarden blijkt de methode van Beale beschouwd te kunnen worden als een methode van toelaatbare richtingen; in geval van een algemene concave, differentieerbare functie en lineaire bijvoorwaarden blijkt dit met de methode van Frank and Wolfe het geval te zijn.

De beide genoemde delen van het proefschrift worden voorafgegaan door een algemeen inleidend hoofdstuk en een hoofdstuk waarin de mathematische theorie der convexe programmering behandeld wordt.

Resumerend kan gesteld worden dat de belangrijkste onderwerpen welke in dit proefschrift ter sprake komen, zijn:

1. de ontwikkeling van een nieuwe numerieke versie van de simplex methode welke voor grote problemen tot veel kleinere rekentijden zal leiden;
2. de ontwikkeling van de methoden van toelaatbare richtingen voor het convexe programmeringsprobleem, welke methoden eindig zijn bij een lineaire of kwadratische waardefunctie en lineaire beperkingen;
3. het vaststellen van een equivalentie tussen sommige bestaande oplossingsmethoden en de methoden van toelaatbare richtingen.

STELLINGEN

1. De door D. van Dantzig ontwikkelde methode van de collectieve kenmerken kan ook gebruikt worden voor de berekening van waarschijnlijkheidsmatrices van "runs" in continue Markof processen.

   D. van Dantzig et G. Zoutendijk, Itérations markoviennes dans les ensembles abstraits, J. de Mathématique pure et appliquée, 38 (1959) 183-200.

2. De door Schlesinger ingevoerde produktintegralen kunnen tevens worden gedefinieerd voor van Dantzigs gegeneraliseerde matrices. Zij kunnen toegepast worden in de theorie der stochastische processen.

   L. Schlesinger, Neue Grundlagen für einen Infinitesimalkalkul der Matrizen, Math. Zeitschrift 33 (1931) 33-61.

3. Laat P(t) de matrix der overgangswaarschijnlijkheden zijn voor een stationair, separabel Markof proces.

   Als $L(\xi) = \int_0^\infty e^{-\xi t}\, p(t)dt$ en $M = \lim_{t \to 0} \frac{P(t)-E}{t}$ (E is een eenheidsmatrix), dan geldt:

   (1) $L(\xi) = (\xi E - M)^{-1}$ voor $|\xi| > ||M||$,

   (2) $L^{(n)}(\xi) = (-1)^n n! \{L(\xi)\}^{n+1}$ en

   (3) $L(\xi + \eta) = \frac{L(\xi)}{E + \eta L(\xi)}$ voor $|\eta| < |\xi|$.

4. Laat een netwerk gegeven zijn met n knooppunten waarbij c_{ij} het geleidingsvermogen aangeeft van de tak die de knooppunten P_i en P_j verbindt. Laat $[c_i]$ de som zijn van de geleidingsvermogens van alle takken die P_i met enig ander knooppunt verbinden. Beschouw de substitutieweerstand r_g van dit netwerk indien P_0 op potentiaal 1 en $P_1, P_2, \ldots\ldots, P_m$, $(m < n)$, op potentiaal 0 gehouden worden. Definieer een stochastische wandeling op dit netwerk met kansen $p_{ij} = 0$ als P_i en P_j niet verbonden en $p_{ij} = \frac{c_{ij}}{[c_i]}$ als P_i en P_j wel verbonden zijn. Beschouw de kans p_g dat een in P_0 startend springend punt in $\{P_1, \ldots, P_m\}$ komt zonder in P_0 te zijn teruggeweest. Nash-Williams heeft aangetoond dat voor eindige netwerken $r_g \cdot p_g = \frac{1}{[c_0]}$. Deze relatie geldt tevens voor oneindige, recurrente netwerken.

   C. St. J. A. Nash-Williams, Random Walk and Electric Currents in Networks, Proc. Cambr. Phil. Soc., 55 (1959) 181-194.

5. De substitutieweerstand van een niet-recurrent netwerk kan alleen berekend worden indien wordt aangegeven hoe het oneindige netwerk als limiet van een eindig netwerk is ontstaan.

6. De relatie van stelling 4 maakt het mogelijk veel "stochastische-wandeling"problemen door elektrische analoga op te lossen.

7. Als, voor $i \in I$, de functies $f_i(x)$ en $F(x)$ convex en differentieerbaar zijn en als $R = \{x \mid f_i(x) + y_i = b_i,\ i \in I;\ x \geqq 0,\ y \geqq 0\}$ begrensd is en voldoet aan de eis dat $\exists_{x,y}\ \forall_{i,j}\ x_j > 0,\ y_i > 0$, dan geldt voor alle $\alpha > 0$ dat $m(\alpha) < \infty$, waarbij $m(\alpha) =$
$= \operatorname{Min} \{F(x) - \alpha \sum_j \log x_j - \alpha \sum_i \log y_i \mid y_i = b_i - f_i(x)\}$, en voor de oplossing $x(\alpha)$, $y(\alpha)$ dat $\lim_{\alpha \downarrow 0} x(\alpha) = x'$, $\lim_{\alpha \downarrow 0} y(\alpha) = y'$, $\lim_{\alpha \downarrow 0} m(\alpha) =$
$= m' = F(x') = \max \{F(x) \mid x \in R\}$.

8. Als, voor $\alpha > 0$, de problemen $\operatorname{Max} \{\Sigma p_j x_j + \alpha \Sigma \log x_j + \alpha \Sigma \log y_i \mid Ax + y = b\}$ en $\operatorname{Min} \{\Sigma b_i u_i - \alpha \Sigma \log u_i - \alpha \Sigma \log v_j \mid A^T u - v = p\}$ eindige oplossing $x_j(\alpha)$, $y_i(\alpha)$, respectievelijk $u_i(\alpha)$, $v_j(\alpha)$ hebben, dan geldt:

(1) $\quad \frac{\alpha}{x_j(\alpha)} = v_j(\alpha),\ \frac{\alpha}{y_i(\alpha)} = u_i(\alpha)$;

(2) $\lim_{\alpha \downarrow 0} x_j(\alpha) = x_j^!$, $\lim_{\alpha \downarrow 0} y_i(\alpha) = y_i^!$, $\lim_{\alpha \downarrow 0} u_i(\alpha) = u_i^!$,
$\lim_{\alpha \downarrow 0} v_j(\alpha) = v_j^!$, de optimale oplossingen van de duale lineaire programmeringsproblemen $\operatorname{Max} \{p^T x \mid Ax + y = b;\ x \geqq 0;\ y \geqq 0\}$
en $\operatorname{Min} \{b^T u \mid A^T u - v = p;\ u \geqq 0;\ v \geqq 0\}$;

(3) $\quad \Sigma b_i u_i(\alpha) = \Sigma p_j x_j(\alpha) + (m + n)\,\alpha$.

Uit (1), (2) en (3) volgen door limietovergang de bekende dualiteitsrelaties van de lineaire programmering.

9. De door G.B.Dantzig voorgestelde procedure om lineaire ongelijkheden toe te voegen ter oplossing van een discreet programmeringsprobleem leidt in het algemeen niet tot een convergente methode.

G.B.Dantzig, Note on Solving Linear Programs in Integers, Naval Res. Log. Quart. 6 (1959) 75-76.

10. Bij het maken van efficiënte lineaire programmeringsroutines voor zeer grote problemen is het gewenst te beschikken over een elektronische rekenmachine die:
a. een grote woordlengte heeft;
b. een zeer groot snel geheugen bezit;
c. magnetische banden bezit, welke in beide richtingen gelezen kunnen worden;
d. zoveel inleeskanalen bezit dat de snelheid der banden geen beperkende factor is.

11. Het bindend voorschrijven van een algemene symbolische machinetaal werkt verstarrend op de ontwikkeling van nieuwe elektronische rekenmachines en moet derhalve ontraden worden.

12. Nagegaan moet worden in hoeverre het zin heeft in Nederland een aparte vereniging en een apart tijdschrift op te richten voor "Operational Research".

13. De verschoolsing van de universiteit heeft het gevaar dat creatieve intelligentie minder mogelijkheid heeft tot ontplooiing te komen en moet daarom niet aangemoedigd worden.

14. Bij het vaststellen der inkomstenbelasting moet niet alleen gelet worden op het totale belastbare inkomen doch tevens op de totale tijd dat voor dit inkomen gewerkt is.