PROJECTION METHODS FOR LARGE SCALE STRUCTURED NONLINEAR PROGRAMMING PROBLEMS

by

PETER DOUGLAS SIMMS
B.Sc.(Ma.)(Hons.)(Adelaide)

A thesis submitted for the degree of
Doctor of Philosophy
in the University of Adelaide

January 1979

TABLE OF CONTENTS

SUMMARY

This thesis is principally concerned with solution techniques for large scale nonlinear programming problems having structured linear constraints and the feature that they probably could not be solved by second order methods.

A class of problems having a simple structure is first considered. Simple expressions for the gradient projection method's search direction and first order estimates of the Lagrange multipliers, in terms of operators, that have been found for this problem by Berry, are stated. A Lemma is given which allows these operators to be used in an efficient version of the gradient projection method, suitable for solving the structured problem with additional general linear constraints. Several other methods for the structured problem and the structured problem with additional linear constraints, that have been or could be used are also discussed.

A class of problems having two separate structures is then considered. A projection method which could be interpreted as a gradient projection or reduced-gradient type method is developed by using column transformations. The Lemma used for extending the single structured problem to one including additional linear constraints is again used for the double structured problem to produce an efficient solution technique.

The performance of the gradient projection method is evaluated. Several changes are made to the original method

and the incorporation of acceleration techniques is considered. The projection methods are then tested on several problems including a design problem for the Adelaide metropolitan telephone network.

SIGNED STATEMENT

This thesis contains no material which has been accepted for the award of any other degree or diploma in any University. To the best of my knowledge and belief, the thesis contains no material previously published or written by any other person, except where due reference is made in the text of the thesis.

(Peter D. Simms)

ACKNOWLEDGEMENTS

I would like to express my appreciation for the encouragement and help given by my supervisors, Dr. L.T.M. Berry and Dr. F.J.M. Salzborn during the course of this research. Many thanks are also extended to the typist, Mrs. A. McKay. I also wish to acknowledge the support of a Commonwealth Postgraduate Research Award.

CHAPTER 1

INTRODUCTION

The aim of this thesis is to examine techniques for solving large scale nonlinear programming problems having a particular structure with an emphasis on problems that cannot be solved by second order methods.

Most practical large scale linearly constrained non-linear programming problems have some particular feature allowing their solution by some method. The most common feature is when the matrices of constraint normals for such problems contain a large proportion of zero elements. These matrices will also often have a certain structure which can be used to devise special solution methods. In fact large scale problems with many linear constraints with neither of these features are likely to be difficult to solve simply because of the limitations imposed by computer storage. Problems can also have objective functions that are predominantly linear with nonlinearity in only relatively few variables. Second order solution techniques for such problems have been discussed by Gill and Murray [19, Chapter 4] and Murtagh and Saunders [31] . These problems will have the property that there will usually be a large number of constraints active at each iteration. In general the feasibility of the use of second order methods can be identified. It is in part related to the variable s, which is defined as the number of variables minus the total number of linearly independent active constraints.

It can be seen that if s is always small (that is, less than about 200) then second order methods can probably be used. For predominantly linear objective functions the value of s at any iteration will usually be small. However, when s can be expected to be large (at least 500, say) the only recourse is to use a robust variant of a first order method. Another special class of problems, discussed by Gill and Murray, occurs when both the matrix of constraint normals and the Hessian have special structure.

The present study was prompted by an approach by Berry [4] in which Rosen's gradient projection method was successfully adapted to solve a telephone network design problem which was a nonlinear program with 3423 variables, 1141 linear equality constraints and a highly nonlinear objective function (that is, a large proportion or all the variables occur nonlinearly). The success of this application of the gradient projection method can be partly attributed to the special structure of the linear constraints. The constraints have a structure which is known as *Generalized Upper Bounding* in linear programming. This structure was used by Berry to produce a modified version of the gradient projection method in which no matrix multiplications were necessary, thus resulting in large benefits in computer storage, computer time, numerical accuracy and ease of implementation. The important feature of this version of the gradient projection method is that the search direction and estimates of the Lagrange multipliers can be easily calculated by using simple explicit operators. This is important because it subsequently allows the solution of problems having additional general

linear constraints by a modification of the gradient projection method, with the amount of computer storage needed dependent only on the number of additional constraints.

Problems with Generalized Upper Bounding structure for which the objective function is predominantly linear, or having the property that s is generally small, can be solved by several second order methods. These methods and the reduced-gradient method can also be extended to solve problems having additional linear constraints.

The Generalized Upper Bounding structure is only one of the possible structures that a large scale problem could have. This class of problems can be generalized to a class of problems referred to here as *transportation-type* problems. This is because the immediate generalization of problems with Generalized Upper Bounding structure, have a structure similar to linear transportation problems. The transportation-type problems have the feature of having two separate structures. For such problems it cannot always be expected that operators of the same form as those obtained for the Generalized Upper Bounding structured problem can be obtained. Instead efficient operators can be found that are based on column transformation methods. An important feature of these operators is that no explicit matrix methods, such as calculating and storing inverses or factorizations are needed. Such a method clearly offers advantages in computer storage and increased numerical stability over any matrix method. It can also be seen that this class of operators can be interpreted as either gradient projection or reduced-gradient type operators. However, the interpretation of these operators as

gradient projection type operators allows an extension to problems having additional general linear constraints.

As the methods developed for structured problems with or without other general linear constraints are mainly based on the gradient projection method it is important to consider the failings of the method. The main problems with the method can be identified and certain remedies effected. To test the actual potential of methods based on gradient projection methods it is also important to test their performance with a real large scale problem. This can be done by considering the telephone network design problem of Berry and various modifications of the problem.

CHAPTER 2

A SINGLE STRUCTURED PROBLEM

2.1 THE PROBLEM

For notational convenience the *Generalized Upper Bounding problem* will be here defined as the nonlinear programming problem:

$$\min \quad f(\underset{\sim}{x}) \tag{2.1}$$

$$\text{s.t.} \quad \sum_{j=1}^{\phi(k)} x_j^k = b^k \,, \quad k = 1,\ldots,\bar{k} \tag{2.2}$$

$$\underset{\sim}{x} = [x_1^1,\ldots,x_{\phi(1)}^1,\ldots,x_1^{\bar{k}},\ldots,x_{\phi(\bar{k})}^{\bar{k}}] \geqslant \underset{\sim}{0} \tag{2.3}$$

$$b^k > 0, \quad k = 1,\ldots,\bar{k},$$

where f is continuously differentiable in the feasible region with gradient $\underset{\sim}{g}(\underset{\sim}{x}) = \underset{\sim}{\nabla} f(\underset{\sim}{x})$ and for later use $n = \sum_{k=1}^{\bar{k}} \phi(k)$.

The occurrence of constraints such as (2.2), together with other more general linear constraints, in a linear programming problem would enable the use of a specialized linear programming technique known as *Generalized Upper Bounding* [3]. The basis of this technique is centred on the structure of the constraints (2.2). For this reason the problem (2.1), (2.2) and (2.3) has been given the above name as it effectively implies the structure of the constraints.

Problems of the form (2.1), (2.2) and (2.3) can occur in particular practical applications. For example the program

can be interpreted as the mathematical formulation of a problem in the economic design of alternative routing telephone networks. The components x_j^k of the point $\underset{\sim}{x}$ will represent chain flows on a multiple origin-destination (OD) network. For each distinct OD pair k there exist $\phi(k)$ routes (chains) available to carry telephone traffic. The equality constraints ensure that the total traffic carried on these routes is a specified quantity b^k. A feasible set of chain flows on the network defines a chain flow pattern $\underset{\sim}{x}$, with a corresponding cost $f(\underset{\sim}{x})$. The problem is to determine a chain flow pattern $\underset{\sim}{x}^*$ with minimum cost $f(\underset{\sim}{x}^*)$.

Such *telephone network design problems* give rise to large programs with typically 5,000 - 10,000 variables. The non-linear function f, which represents the cost of junctions required to give specified OD performances, is a differentiable function.

Solutions to the problem (2.1), (2.2) and (2.3), where $f(\underset{\sim}{x})$ is linear can be given by inspection. If $f(\underset{\sim}{x}) = \underset{\sim}{c}^T\underset{\sim}{x}$ (say), then

$$x^{*k}_j = \begin{cases} b^k & \text{if } j = j(k) \\ 0 & \text{otherwise} \end{cases} \qquad (2.4)$$

where for $k = 1,\ldots,\bar{k}$ $j(k)$ satisfies

$$c^k_{j(k)} = \min_{1 \leq j \leq \phi(k)} \{c^k_j\} .$$

This is commonly known as the *all-or-nothing allocation principle.*

Just as this special structure provides an immediate solution for linear f, it can be expected that simpler versions

of standard nonlinear programming methods can be constructed for nonlinear f. Five solution techniques and the feasibility of their application to the Generalized Upper Bounding problem will be discussed. These will include a linear approximation technique, two first order methods and two second order methods. Their suitability will be evaluated in terms of computer storage and the amount of computation required per iteration.

2.2 THE CONDITIONAL GRADIENT METHOD

The *conditional gradient method* is a method used by Klessig [27] in an algorithm for solving nonlinear multi-commodity flow problems. The method stated by Klessig is a variation of the Frank and Wolfe method for quadratic programming [17].

The conditional gradient method can be described as follows: given a feasible point $\underset{\sim}{x}_0$, select $\alpha, \beta \in (0,1)$ and let Ω denote the feasible region. Then for $i = 1,2,\ldots$

(i) Find any solution $\tilde{\underset{\sim}{x}}$ to the linear program

$$\begin{aligned} \min \quad & \underset{\sim}{x}^T \underset{\sim}{g}(\underset{\sim}{x}_{i-1}) \\ \text{s.t.} \quad & \underset{\sim}{x} \in \Omega \end{aligned} \tag{2.5}$$

Set $\underset{\sim}{p} = \tilde{\underset{\sim}{x}} - \underset{\sim}{x}_{i-1}$.

(ii) If $\underset{\sim}{p}^T \underset{\sim}{g}(\underset{\sim}{x}_{i-1}) = 0$, then stop as $\underset{\sim}{x}_{i-1}$ is the optimal solution.

(iii) Compute the smallest integer k_i such that

$$f(\underset{\sim}{x}_{i-1} + \beta^{k_i} \underset{\sim}{p}) - f(\underset{\sim}{x}_{i-1}) - \alpha\beta^{k_i} \underset{\sim}{p}^T \underset{\sim}{g}_{i-1} \leqslant 0.$$

(iv) Set $\underset{\sim}{x}_i = \underset{\sim}{x}_{i-1} + \beta^{k_i} \underset{\sim}{p}$.

It should be noticed that the linear program in step (i) of the conditional gradient method may also be viewed as minimizing the linear approximation to f at the point $\underset{\sim}{x}_{i-1}$ for small values of λ, i.e.

$$\begin{aligned} \min \quad & f(\underset{\sim}{x}_{i-1}) + \lambda \, \underset{\sim}{g}(\underset{\sim}{x}_{i-1})^T (\underset{\sim}{x} - \underset{\sim}{x}_{i-1}) \\ \text{s.t.} \quad & \underset{\sim}{x} \in \Omega \quad . \end{aligned}$$

An all-or-nothing solution similar to (2.4) can be easily found to the linear program (2.5). Hence the question of computer storage is irrelevant for the conditional gradient method. Similarly the amount of computation per iteration is negligible except perhaps that required in the *line-search procedure* of step (iii). In a practical problem it seems to often be the case that a major part of the effort is involved in performing such line-searches. Hence the convergence of the method for large problems becomes an important consideration as this will determine the number of line searches needed and in turn the suitability of the approach. Thus it seems preferable to use a first or second order method, with the promise of better convergence, as long as the additional storage and computation then needed does not make the implementation of such a method impracticable.

2.3 THE GRADIENT PROJECTION METHOD

In a study by Berry [4] a mathematical program of the form (2.1), (2.2) and (2.3) was used for dimensioning the Adelaide metropolitan telephone network. This problem had 1141 constraints of the form (2.2), 3423 variables and a

highly nonlinear objective function. A method for solving this problem was devised by adapting Rosen's method [36] to take into account the special structure of the constraints.

An elementary description of Rosen's gradient projection method can be given as follows:

Given a feasible point $\underset{\sim}{x}_0$, then for $i = 1,2,\ldots$

(i) Calculate $\underset{\sim}{p} = - P_N \underset{\sim}{g}_{i-1}$ (2.6)

where

$$P_N = I - N^T[NN^T]^{-1}N,$$

N is the matrix whose rows are the normals of the currently active constraints and $\underset{\sim}{g}_{i-1} = \underset{\sim}{g}(\underset{\sim}{x}_{i-1})$.

(ii) If $\underset{\sim}{p} = \underset{\sim}{0}$, go to (iv).

(iii) Set $\underset{\sim}{x}_i = \underset{\sim}{x}_{i-1} + \alpha^* \underset{\sim}{p}$,

where

$$f(\underset{\sim}{x}_{i-1}+\alpha^*\underset{\sim}{p}) = \min_{\alpha\in(0,\alpha_{max}]} f(\underset{\sim}{x}_{i-1}+\alpha\underset{\sim}{p})$$

and

$$\alpha_{max} = \min \left\{ - \frac{x_j^k}{p_j^k} \;\middle|\; p_j^k < 0,\ j = 1,\ldots,\phi(k);\ k = 1,\ldots,\bar{k} \right\}.$$

Go to (i).

(iv) Calculate the first order estimates of the Lagrange multipliers

$$\underset{\sim}{\lambda} = N^+ \underset{\sim}{g}_{i-1} \qquad (2.7)$$

where

$$N^+ = [NN^T]^{-1}N.$$

Delete the constraint corresponding to the most negative Lagrange multiplier estimate and go to (i). If the estimates are all non-negative then the solution is optimal.

It can be seen that the direction $\underset{\sim}{p}$ is simply the solution of the program

$$\begin{aligned} \min \quad & \underset{\sim}{p}^T \underset{\sim}{g} \\ \text{s.t.} \quad & N\underset{\sim}{p} = \underset{\sim}{0} \\ & \|\underset{\sim}{p}\|_2^2 = 1 \end{aligned} \tag{2.8}$$

or, alternatively, it can be obtained by solving

$$\begin{aligned} \min \quad & \underset{\sim}{p}^T \underset{\sim}{g} \\ \text{s.t.} \quad & \underset{\sim}{p} = P_N \underset{\sim}{\bar{p}} \\ & \|\underset{\sim}{\bar{p}}\|_2^2 = 1. \end{aligned} \tag{2.9}$$

The solution of such a program is just the orthogonal projection of the gradient $\underset{\sim}{g}$ onto the intersection of active constraint hyperplanes. The problem of solving the first program is also equivalent to finding the minimum distance (in terms of the Euclidean norm) from some point to the intersection of given hyperplanes.

Because the matrix N has such a special structure for the Generalized Upper Bounding problem, Berry [5] was able to circumvent the necessity of using matrices to calculate P_N and N^+. Instead these matrices were described as operators. This can be done as follows: for an arbitrary vector $\underset{\sim}{w} \in R^n$,

$$[P_N \underset{\sim}{w}]_j^k = \delta_j^k \{w_j^k - (1/p(k)) \sum_{\ell=1}^{\phi(k)} \delta_\ell^k w_\ell^k\} \tag{2.10}$$

where

$$\delta_j^k = \begin{cases} 0 & \text{if } x_j^k = 0 \\ 1 & \text{otherwise} \end{cases}$$

and

$$p(k) = \sum_{\ell=1}^{\phi(k)} \delta_\ell^k \quad .$$

The elements of $N^+ \underset{\sim}{w}$ corresponding to $x_j^k = 0$ are numerically equal to

$$w_j^k - (1/p(k)) \sum_{\ell=1}^{\phi(k)} \delta_\ell^k w_\ell^k \tag{2.11}$$

and the elements of $N^+ \underset{\sim}{w}$ corresponding to $\sum_{j=1}^{\phi(k)} x_j^k = b^k$ are

$$(1/p(k)) \sum_{\ell=1}^{\phi(k)} \delta_\ell^k w_\ell^k . \tag{2.12}$$

Thus the gradient projection method is also a viable technique for solving the Generalized Upper Bounding problem as far as storage is concerned. This is because the search direction (2.6) and the multiplier estimates (2.7) can be easily calculated from (2.10), (2.11) and (2.12). This contrasts dramatically with a straightforward application of the gradient projection method which, for the telephone network design problem, needs the matrix $[NN^T]^{-1}$ which in this case will be of size $m \times m$, where $m \geqslant 1141$.

A later study by Bruyn [9] was conducted on the long-term planning of a telephone junction network. The mathematical programming problem in this case was broken into a series of smaller subproblems, each of the same size and type as those studied by Berry. The modified gradient projection of Berry was here further modified in that these subproblems

had non-differentiable objective functions and subgradients were used when the gradients did not exist.

2.4 THE REVISED GOLDFARB'S METHOD

The variable metric method of Davidon [11] was extended by Goldfarb [22] and combined with Rosen's gradient projection method in an effort to use second order information in solving linearly constrained nonlinear programs. This resulted in a conjugate-gradient type algorithm.

This was essentially accomplished by using the operator H_N in place of P_N in (2.6), where H_N is a symmetric, positive definite approximation to the operator

$$\hat{P}_N = G^{-1} - G^{-1}N^T[NG^{-1}N^T]^{-1}NG^{-1} \qquad (2.13)$$

where G is the Hessian of f at $\underset{\sim}{x}$. The reference to conjugate-gradients arises from the way in which the operator H_N is updated.

The direction $\underset{\sim}{p} = -\hat{P}_N \underset{\sim}{g}$ can be seen to be the solution of the program

$$\begin{aligned} \min \quad & f_0 + \underset{\sim}{a}^T\underset{\sim}{x} + \tfrac{1}{2}\underset{\sim}{x}^T G\underset{\sim}{x} \\ \text{s.t.} \quad & N\underset{\sim}{x} = \underset{\sim}{0} \end{aligned}$$

where $f(\underset{\sim}{x})$ is approximated by the form $f_0 + \underset{\sim}{a}^T\underset{\sim}{x} + \tfrac{1}{2}\underset{\sim}{x}^T G\underset{\sim}{x}$. In this case $\underset{\sim}{g}(\underset{\sim}{x}) = G\underset{\sim}{x} + \underset{\sim}{a}$. The use of the necessary and sufficient condition for a minima, $\underset{\sim}{g} = N^T\underset{\sim}{\alpha}$ for some $\underset{\sim}{\alpha}$, leads to $\underset{\sim}{p} = -\hat{P}_N\underset{\sim}{g}$. Alternatively $\underset{\sim}{p}$ can be seen to be a scalar multiple of the solution to the program

$$\begin{aligned} \min \quad & \underset{\sim}{p}^T \underset{\sim}{g} \\ \text{s.t.} \quad & N\underset{\sim}{p} = \underset{\sim}{0} \\ & \underset{\sim}{p}^T G \underset{\sim}{p} = 1. \end{aligned} \tag{2.14}$$

This is equivalent to finding the minimum distance, from a given point to the intersection of certain given hyperplanes, in the sense of the metric G. Alternatively $\underset{\sim}{p}$ can be obtained by solving

$$\begin{aligned} \min \quad & \underset{\sim}{p}^T \underset{\sim}{g} \\ \text{s.t.} \quad & \underset{\sim}{p} = \hat{P}_N L \, \underset{\sim}{\bar{p}} \\ & \| \underset{\sim}{\bar{p}} \|_2^2 = 1, \end{aligned} \tag{2.15}$$

where $LL^T = G^{-1}$ and it should be noted that $\hat{P}_N = P_{NL}$.

Buckley [10] suggested an alternative implementation of Goldfarb's method. In the revised version a nonsingular linear transformation E, to a new coordinate system $\underset{\sim}{y}$ is used, such that

$$\underset{\sim}{y} = E^{-1} \underset{\sim}{x}$$

$$E^{-1} = \begin{bmatrix} N \\ D \end{bmatrix},$$

D is any $s \times n$ matrix such that E has rank n and s is equal to n minus the total number of active constraints.

If $M = NE$, then $M = [I_{n-s} \; 0]$. In this case, any matrix

$$K_N = \begin{bmatrix} 0 & 0 \\ 0 & \tilde{K}_N \end{bmatrix},$$

where $\tilde{K}_N$ is a positive definite symmetric $s \times s$ matrix, meets the requirements of Goldfarb's algorithm. The matrix $\tilde{K}_N$ can be obtained as

$$\tilde{K}_N = DH_N D^T \quad .$$

The $s \times s$ matrix $\tilde{K}_N$ is now updated instead of the $n \times n$ matrix H_N .

The revised algorithm given by Buckley can be briefly described as follows:

Given a feasible point $\underset{\sim}{x}_0$, the matrix $E(n \times n)$ and the positive definite matrix $\tilde{K}_N$ $(s \times s)$, then for $i = 1,2,\ldots$

(i) Calculate

$$\underset{\sim}{p}_y = \begin{bmatrix} \underset{\sim}{0} \\ \hat{\underset{\sim}{p}}_y \end{bmatrix}$$

where

$$\hat{\underset{\sim}{p}}_y = - \tilde{K}_N \hat{\underset{\sim}{g}}_y , \quad \underset{\sim}{g}_y = \nabla_{\underset{\sim}{y}} f(\underset{\sim}{y}) = \begin{bmatrix} \bar{\underset{\sim}{g}}_y \\ \hat{\underset{\sim}{g}}_y \end{bmatrix}$$

and $\hat{\underset{\sim}{g}}_y \in R^s$.

(ii) If $\underset{\sim}{p}_y = \underset{\sim}{0}$, then go to (iv).

Set $\underset{\sim}{p} = E \underset{\sim}{p}_y$.

(iii) Set $\underset{\sim}{x}_i = \underset{\sim}{x}_{i-1} + \alpha^* \underset{\sim}{p}$

where

$$f(\underset{\sim}{x}_{i-1} + \alpha^* \underset{\sim}{p}) = \min_{\alpha \in (0, \alpha_{max}]} f(\underset{\sim}{x}_{i-1} + \alpha \underset{\sim}{p}) .$$

Revise $\tilde{K}_N$, E as necessary.

Go to (i).

(iv) Calculate the first order Lagrange multiplier estimates $\underset{\sim}{\lambda}_y = \bar{\underset{\sim}{g}}_y$.

Delete the constraint corresponding to the most negative estimate, revise $\tilde{K}_N$ and go to (i).

If all the estimates are non-negative the solution is optimal.

One defect of the original Goldfarb's method was that even if the constraints were well-conditioned, the manner in which the Hessian information was incorporated in the search direction is such that the search directions are contaminated and often prove to be infeasible. With the Revised Goldfarb's method it can be seen from the above algorithm that this contamination cannot occur.

In Buckley's method the matrix $E(n\times n)$ is stored and updated during the progress of the algorithm. For a large problem (e.g. $n > 3000$) storing and performing operations with an $n\times n$ matrix is clearly out of the question. However it can be noticed that for the Generalized Upper Bounding problem, E does not need to be stored. By the virtue of column permutations, the matrix N can be written as

$$N = \begin{bmatrix} I & N_{12} & N_{13} \\ 0 & 0 & I \end{bmatrix} .$$

In this case, if E is the matrix

$$E = \begin{bmatrix} I & -N_{13} & -N_{12} \\ 0 & 0 & I \\ 0 & I & 0 \end{bmatrix}$$

then

$$M = NE = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \end{bmatrix}$$

which is of the required form.

This means that only the matrix $\tilde{K}_N$ $(s\times s)$ is required. Hence this method could be applied to large scale Generalized Upper Bounding problems as long as s is no larger than about

200. For s significantly larger than 200, the method would become impracticable. This has been seen to be the case with the telephone network design problem, where it has been noticed that near the optimal solution the value of s is in the region of 500 to 600.

2.5 THE REDUCED-GRADIENT METHOD

Harris [24] has also constructed an efficient solution procedure based on the reduced-gradient method [41, 42, 12]. This modified reduced-gradient method was used in investigations of different concepts of optimality for telephone network design problems of the same nature as those studied by Berry.

The formulation of the constraints necessary for the application of the reduced-gradient method is

$$A\underset{\sim}{x} = \underset{\sim}{b} \tag{2.16}$$

$$\underset{\sim}{x} \geqslant \underset{\sim}{0} \; . \tag{2.17}$$

At the start of an iteration it is assumed that there is a feasible solution $\underset{\sim}{x}$ which is partitioned into *basic* and *nonbasic* variables (respectively) as follows:

$$\underset{\sim}{x} = \begin{bmatrix} \underset{\sim}{x}_B \\ \underset{\sim}{x}_N \end{bmatrix} \; .$$

The matrix $A = [A_B \; A_N]$ will be assumed to have been transformed, by the premultiplication of A_B^{-1}, to $[I \; \bar{A}_N]$ where $\bar{A}_N = A_B^{-1} A_N$. Hence the basic variables can be written in terms of the non-basic variables, i.e.

$$\underset{\sim}{x}_B = - \bar{A}_N \underset{\sim}{x}_N + \underset{\sim}{\bar{b}}$$

where $\underset{\sim}{\bar{b}} = A_B^{-1} \underset{\sim}{b}$.

Differentiation of f yields

$$df(\underset{\sim}{x}) = \frac{\partial f(x)}{\partial \underset{\sim}{x}_B}^T d\underset{\sim}{x}_B + \frac{\partial f(x)}{\partial \underset{\sim}{x}_N}^T d\underset{\sim}{x}_N$$

$$= \left[- \frac{\partial f(x)}{\partial \underset{\sim}{x}_B}^T \bar{A}_N + \frac{\partial f(x)}{\partial \underset{\sim}{x}_N}^T \right] d\underset{\sim}{x}_N \quad .$$

Hence if $\underset{\sim}{r}$ is a vector given by

$$\underset{\sim}{r}^T = \left[\underset{\sim}{0}^T \mid \frac{\partial f(x)}{\partial \underset{\sim}{x}_B}^T \bar{A}_N - \frac{\partial f(x)}{\partial \underset{\sim}{x}_N}^T \right]$$

it can be seen that $\underset{\sim}{r}$ is a descent direction. However $\underset{\sim}{r}$ is not necessarily a feasible direction. A feasible descent direction can be constructed as follows:

$$[\underset{\sim}{p}_N]_j = \begin{cases} 0 & \text{if } r_j < 0 \text{ and } x_j = 0 \\ r_j & \text{otherwise} \end{cases}$$

and

$$\underset{\sim}{p}_B = - \bar{A}_N \underset{\sim}{p}_N \quad .$$

The modified algorithm of Harris can be described as follows:

Given a feasible point $\underset{\sim}{x}_0$, then for $i = 1,2,\ldots$

(i) For $k = 1,\ldots,\bar{k}$ select $j(k)$ such that

$$\frac{\partial f}{\partial x^k_{j(k)}} = \min_{1 \leq \ell \leq \phi(k)} \left\{ \frac{\partial f}{\partial x^k_\ell} \right\} \quad .$$

The variable $x^k_{j(k)}$ will be a basic variable and the variables x^k_ℓ, $\ell \neq j(k)$ will be nonbasic variables.

(ii) Define the vector $\underset{\sim}{p}$ as follows:

For $\ell \neq j(k)$

$$p_\ell^k = \begin{cases} 0 \quad \text{if} \quad x_\ell^k = 0 \quad \text{and} \quad \dfrac{\partial f}{\partial x_{j(k)}^k} - \dfrac{\partial f}{\partial x_\ell^k} < 0 \\ \dfrac{\partial f}{\partial x_{j(k)}^k} - \dfrac{\partial f}{\partial x_\ell^k} \quad \text{otherwise} \end{cases} \tag{2.18}$$

and

$$p_{j(k)}^k = - \sum_{\ell \neq j(k)} p_\ell^k \ . \tag{2.19}$$

(iii) If $\underset{\sim}{p} = \underset{\sim}{0}$, stop, as solution is optimal.

(iv) Set $\underset{\sim}{x}_i = \underset{\sim}{x}_{i-1} + \alpha^* \underset{\sim}{p}$

where

$$f(\underset{\sim}{x}_{i-1} + \alpha^* \underset{\sim}{p}) = \min_{\alpha \in (0, \alpha_{max}]} f(\underset{\sim}{x}_{i-1} + \alpha \underset{\sim}{p}) .$$

This algorithm follows from the fact that for the Generalized Upper Bounding problem, N and $\underset{\sim}{p}^T$ can be written, after column permutations and deleting certain active non-negativity constraint normals, as

$$\begin{bmatrix} I & N_{12} & N_{13} \\ 0 & 0 & I \end{bmatrix} \tag{2.20}$$

and

$$[\ \underset{\sim}{p}_1^T \quad \underset{\sim}{p}_2^T \quad \underset{\sim}{p}_3^T]$$

respectively.

The algorithm then sets $\underset{\sim}{p}_3 = \underset{\sim}{0}$, $\underset{\sim}{p}_2$ from (2.18) and $\underset{\sim}{p}_1 = -N_{12}\underset{\sim}{p}_2$. This ensures that $\underset{\sim}{p}$ is both a feasible direction and a direction decrease.

Alternatively, if the matrix Z is defined as

$$Z = \begin{bmatrix} -N_{12} \\ I \\ 0 \end{bmatrix}$$

then the direction $\underset{\sim}{p}$ can be written as

$$\underset{\sim}{p} = -ZZ^T\underset{\sim}{g} \quad . \tag{2.21}$$

This can be compared with the direction obtained with the gradient projection method (with N as in (2.20), which can be rewritten as

$$\underset{\sim}{p} = -Z(Z^TZ)^{-1}Z^T\underset{\sim}{g} \quad . \tag{2.22}$$

It can also be noticed that the reduced-gradient search direction $\underset{\sim}{p}$ can be obtained by solving the program

$$\begin{aligned} \min \quad & \underset{\sim}{p}^T\underset{\sim}{g} \\ \text{s.t.} \quad & \underset{\sim}{p} = Z\bar{\underset{\sim}{p}} \\ & \|\bar{\underset{\sim}{p}}\|_2^2 = 1. \end{aligned} \tag{2.23}$$

The simple nature of expressions (2.18) and (2.19) means that the modified reduced-gradient method of Harris, like the modified gradient projection method of Berry has the advantage of negligible computer storage and a simple implementation. Also, the search directions produced by these two methods are similar in nature as comparison of (2.9) and (2.23) shows that the only difference is in the way the general solution to $N\underset{\sim}{p} = \underset{\sim}{0}$ is written. Thus there should be no general dominance in performance by either method.

2.6 THE MURTAGH-SAUNDERS METHOD

A second order technique for large-scale constrained problems, related to the reduced-gradient method has recently been proposed by Murtagh and Saunders [31]. This method is based on the reduced-gradient method and the related variable-reduction method of McCormick [29, 30] together with a quasi-Newton approach.

The mathematical program in this instance is reformulated as

$$\begin{aligned} \min \quad & f(\underset{\sim}{x}) \\ \text{s.t.} \quad & A\underset{\sim}{x} = \underset{\sim}{b} \\ & \underset{\sim}{\ell} \leqslant \underset{\sim}{x} \leqslant \underset{\sim}{u} \end{aligned}$$

where A is $(m \times n)$.

Gill and Murray [19] have considered a class of algorithms in which the search direction along the surface of active constraints is characterized by being in the column space of a matrix Z which is orthogonal to the matrix N, i.e. $NZ = 0$. Using this characterization of the active constraints, the quadratic program resulting from a second order Taylor expansion of f,

$$\begin{aligned} \min \quad & \tfrac{1}{2}\, \underset{\sim}{p}^T G \underset{\sim}{p} + \underset{\sim}{g}^T \underset{\sim}{p} \\ \text{s.t.} \quad & N\underset{\sim}{p} = \underset{\sim}{0} \end{aligned}$$

can be written as the unconstrained problem

$$\min_{\underset{\sim}{v}} \tfrac{1}{2}\, \underset{\sim}{v}^T Z^T G Z \underset{\sim}{v} + \underset{\sim}{g}^T Z \underset{\sim}{v} \quad . \tag{2.24}$$

If the matrix $Z^T G\, Z$ is positive definite, then the solution of the quadratic program is

$$\underset{\sim}{p}^* = Z\underset{\sim}{v}^*$$

where $\underset{\sim}{v}^*$ is the solution of

$$(Z^T G Z)\underset{\sim}{v}^* = - Z^T \underset{\sim}{g} \quad .$$

This can be contrasted with the revised Goldfarb's method's direction $\underset{\sim}{p} = - P_{\hat{N}}\, \underset{\sim}{g} = P_N\, \underset{\sim}{u}^*$, where $\underset{\sim}{u}^*$ is the solution of the unconstrained problem

$$\min_{\underset{\sim}{u}} \tfrac{1}{2}\underset{\sim}{u}^T P_N G P_N \underset{\sim}{u} + \underset{\sim}{g}^T P_N \underset{\sim}{u} \quad . \tag{2.25}$$

The direction $\underset{\sim}{p}^*$ can also be obtained by solving the program

$$\begin{aligned} \min \quad & \underset{\sim}{p}^T \underset{\sim}{g} \\ \text{s.t.} \quad & \underset{\sim}{p} = Z\, \bar{\underset{\sim}{p}} \\ & \bar{\underset{\sim}{p}}^T (Z^T G Z) \bar{\underset{\sim}{p}} = 1 \end{aligned} \tag{2.26}$$

or alternatively by solving

$$\begin{aligned} \min \quad & \underset{\sim}{p}^T \underset{\sim}{g} \\ \text{s.t.} \quad & \underset{\sim}{p} = ZL\bar{\underset{\sim}{p}} \\ & \| \bar{\underset{\sim}{p}} \|_2^2 = 1 \end{aligned} \tag{2.27}$$

where $LL^T = (Z^T G Z)^{-1}$.

For the Generalized Upper Bounding problem, it has been found that such a matrix Z for N in the form of (2.20) is given by

$$Z = \begin{bmatrix} - N_{12} \\ I \\ 0 \end{bmatrix} \quad .$$

The method proposed by Murtagh and Saunders uses the following quasi-Newton algorithm:

Given a feasible point $\underset{\sim}{x}_0$, for i = 1,2,...

(i) Compute the reduced-gradient $\underset{\sim}{g}_A = Z^T \underset{\sim}{g}_{i-1}$.

(ii) Form some approximation to the reduced Hessian

$$G_A \doteq Z^T G Z.$$

(iii) Obtain an approximate solution to the system

$$Z^T G Z \, \underset{\sim}{p}_A = - Z^T \underset{\sim}{g}_{i-1}$$

by solving

$$G_A \, \underset{\sim}{p}_A = - \underset{\sim}{g}_A \quad .$$

(iv) Compute the search direction $\underset{\sim}{p} = Z\underset{\sim}{p}_A$.

(v) Set $\underset{\sim}{x}_i = \underset{\sim}{x}_{i-1} + \alpha^* \underset{\sim}{p}$

where

$$f(\underset{\sim}{x}_{i-1} + \alpha^* \underset{\sim}{p}) = \min_{\alpha \in (0, \alpha_{max}]} f(\underset{\sim}{x}_{i-1} + \alpha \underset{\sim}{p}).$$

For a general matrix A, the matrix of active constraint normals can be written in the form

$$N = \begin{bmatrix} N_{11} & N_{12} & N_{13} \\ 0 & 0 & I \end{bmatrix}$$

where N_{11} is m×m and nonsingular, N_{12} is m×s and N_{13} is m×(n-m-s). The method of Murtagh and Saunders then uses a matrix

$$Z = \begin{bmatrix} -[N_{11}]^{-1} N_{12} \\ I \\ 0 \end{bmatrix} \quad .$$

The implementation suggested by the authors is to use an LU decomposition of N_{11} and the factorization $G_A = R^T R$, where R is upper triangular, for the approximation to $Z^T G Z$.

Thus for moderately sized problems, the method of Murtagh and Saunders can be used for the Generalized Upper Bounding problem. This is because N_{11} will be the identity matrix. As for the revised Goldfarb's method, if s is large (e.g.500) the method is impracticable.

2.7 A COMPARISON OF THE METHODS

Hence, for the Generalized Upper Bounding problem, the best method to use will depend mainly on the expected maximum value of s. If s can be expected to be small to moderate (i.e. less than about 200) then a second order method should be used if possible. Otherwise the only alternative is to use one of, the conditional gradient method, the gradient projection method or the reduced-gradient method.

It is interesting to compare the search directions produced for the five methods. An elementary comparison can be made by using the following problem:

$$\begin{aligned} \min \quad & f(x_1, x_2, x_3) \\ \text{s.t.} \quad & x_1 + x_2 + x_3 = b \\ & x_1, x_2, x_3 \geqslant 0. \end{aligned}$$

Let the current solution be $\bar{\underset{\sim}{x}} = [0, \bar{x}_2, \bar{x}_3]^T$ where $\bar{x}_2, \bar{x}_3 > 0$. Let the gradient at $\bar{\underset{\sim}{x}}$ be $\bar{\underset{\sim}{g}} = [\bar{g}_1, \bar{g}_2, \bar{g}_3]^T$. Also assume that $\bar{g}_1 = \min\{\bar{g}_1, \bar{g}_2, \bar{g}_3\}$.

(a) The conditional gradient method:

$$\underset{\sim}{p}_1^T = [b, -\bar{x}_2, -\bar{x}_3] .$$

Also $$\underset{\sim}{p}_1^T \underset{\sim}{\bar{g}} = \bar{x}_2(\bar{g}_1 - \bar{g}_2) + \bar{x}_3(\bar{g}_1 - \bar{g}_3) .$$

(b) The gradient projection method:

$$\underset{\sim}{p}_2^T = [0, \tfrac{1}{2}(\bar{g}_3 - \bar{g}_2), \tfrac{1}{2}(\bar{g}_2 - \bar{g}_3)] .$$

Also $$\underset{\sim}{p}_2^T \underset{\sim}{\bar{g}} = -\tfrac{1}{2}(\bar{g}_2 - \bar{g}_3)^2 .$$

(c) The reduced-gradient method:

$$\underset{\sim}{p}_3^T = [-(\bar{g}_1 - \bar{g}_2 + \bar{g}_1 - \bar{g}_3), (\bar{g}_1 - \bar{g}_2), (\bar{g}_1 - \bar{g}_3)] .$$

Also

$$\underset{\sim}{p}_2^T \underset{\sim}{\bar{g}} = -(\bar{g}_1 - \bar{g}_2)^2 - (\bar{g}_1 - \bar{g}_3)^2$$

(d) The revised Goldfarb's method:

$$N = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \qquad E = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & -1 & -1 \end{bmatrix} \qquad M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} ,$$

$$\underset{\sim}{p}_4^T = [0, \tilde{K}(\bar{g}_3 - \bar{g}_2), \tilde{K}(\bar{g}_2 - \bar{g}_3)] , \quad \tilde{K} > 0$$

$$= 2\tilde{K}\, \underset{\sim}{p}_2^T .$$

(e) The Murtagh-Saunders method:

$$Z = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}$$

$$\underset{\sim}{p}_5^T = [0, \tilde{G}(\bar{g}_3 - \bar{g}_2), \tilde{G}(\bar{g}_2 - \bar{g}_3)] , \quad \tilde{G} > 0$$

$$= 2\tilde{G}\, \underset{\sim}{p}_2^T .$$

For this particular example $s = 1$ and the directions in (b), (d) and (e) are equivalent. For $s \geqslant 2$ the directions produced will generally be different.

It can also be noted that with the gradient projection method, the Lagrange multiplier estimate for x_1 is

$$\bar{g}_1 - \tfrac{1}{2}(\bar{g}_2 + \bar{g}_3) < 0.$$

Hence constraint $x_1 = 0$ can be dropped from the active set. The gradient projection method then produces the steepest descent direction

$$\underset{\sim}{p}_2'^{T} = [-\tfrac{1}{3}(\bar{g}_1-\bar{g}_2+\bar{g}_1-\bar{g}_3),\ \tfrac{1}{3}(\bar{g}_1+\bar{g}_3-2\bar{g}_2),\ \tfrac{1}{3}(\bar{g}_1+\bar{g}_2-2\bar{g}_3)] .$$

It can then be seen that

$$\underset{\sim}{p}_2'^{T}\,\underset{\sim}{\bar{g}} = -\tfrac{1}{3}[(\bar{g}_1-\bar{g}_2)^2 + (\bar{g}_1-\bar{g}_3)^2 + (\bar{g}_2-\bar{g}_3)^2] .$$

CHAPTER 3

THE AUGMENTED GENERALIZED UPPER BOUNDING PROBLEM

3.1 THE PROBLEM

The mathematical program now under consideration is:

$$\min \quad f(\underset{\sim}{x}) \tag{3.1}$$

$$\text{s.t.} \quad \sum_{j=1}^{\phi(k)} x_j^k = b^k \quad , \quad k = 1,\ldots,\bar{k} \tag{3.2}$$

$$\sum_{k=1}^{\bar{k}} \sum_{j=1}^{\phi(k)} \gamma_{ij}^k \, x_j^k \leqslant c_i \, , \quad i = 1,\ldots,\bar{\ell} \tag{3.3}$$

$$\underset{\sim}{x} = [x_1^1,\ldots,x_{\phi(1)}^1,\ldots,x_1^{\bar{k}},\ldots,x_{\phi(\bar{k})}^{\bar{k}}] \geqslant \underset{\sim}{0} \tag{3.4}$$

$$b^k > 0 \, , \quad k = 1,\ldots,\bar{k}$$

with f having the same properties as in Section 2.1. It will also be assumed that $\bar{k} >> \bar{\ell}$ and that the number of active constraints of type (3.3) will be less than or equal to about 200.

It was shown in Section 2.1 that problems of the form (3.1), (3.2) and (3.4) can be interpreted as multicommodity minimum cost network flow problems. Thus the inclusion of constraints of the form (3.3) can be interpreted as including link capacity constraints on the problem.

The presence of the constraints (3.3) in the case when f is linear, makes the problem of solving the resulting linear program a nontrivial exercise. However the Generalized Upper Bounding technique [3] of linear programming does provide

a method taking maximum advantage of the structure of the constraints (3.2). It can then be similarly expected that specialized techniques can be developed when f is nonlinear.

3.2 POSSIBLE SOLUTION TECHNIQUES

In Chapter 2, five solution methods were discussed for the Generalized Upper Bounding problem. As problem (3.1) - (3.4) is a further extension of problem (2.1) - (2.3) in that additional constraints are included, it seems likely that suitable methods can be found by applying or modifying the techniques discussed in Chapter 2. To be viable for large scale problems, such techniques would need to be able to produce a reasonable feasible descent direction without excessive computation or storage. This also includes the necessity to ensure that such techniques will not break down through reasons of numerical instability.

The introduction of constraints (3.3) significantly changes the suitability of the conditional gradient method. The solution of the linear programming problem in step (i) of the algorithm of Section 2.2 is no longer a trivial exercise, particularly for moderately large values of $\bar{\ell}$. The linear program could be solved by using the Generalized Upper Bounding technique of linear programming, but if the linear program is large (e.g. $n = 3400$, $\bar{k} = 1100$, $\bar{\ell} = 200$), the effort needed to find the optimal solution becomes unreasonable particularly if many such linear programming subproblems need to be solved.

Klessig [27] used the conditional gradient method to solve a general class of minimum cost multicommodity flow problems. In this case the linear program generated at each iteration is a linear minimum cost multicommodity flow problem. Klessig also showed that when the network had several special features, substantial computational simplifications can result. In these cases, network techniques such as a shortest path algorithm can be used. Hence in certain isolated cases the conditional gradient method does offer the promise of a good solution technique. However for the general problem, the prospect of solving a different large linear program at each iteration is not very desirable.

In Sections 2.4, 2.5 and 2.6 the methods described were based on using one of the matrices Z or E. For the augmented Generalized Upper Bounding problem, the matrix N, after column permutations, can be written as

$$N = \begin{bmatrix} I & N_{12} & N_{13} & N_{14} \\ N_{21} & N_{22} & N_{23} & N_{24} \\ 0 & 0 & 0 & I \end{bmatrix}$$

where N_{22} is a nonsingular matrix.

Suitable matrices E and Z have been found and these can be given as follows:

(a)
$$E = \bar{E} \begin{bmatrix} I & 0 & -N_{13} & -N_{14} \\ 0 & I & -N_{23} & -N_{24} \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix}$$

where

$$\bar{E} = \begin{bmatrix} E_{11} & 0 \\ 0 & I \end{bmatrix} ,$$

$$E_{11} = \begin{bmatrix} I & -N_{12} \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & \hat{N}_{22}^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -N_{21} & I \end{bmatrix}$$

and

$$\hat{N}_{22} = N_{22} - N_{21} N_{12} .$$

(b)
$$Z = \begin{bmatrix} -E_{11} \begin{bmatrix} N_{13} \\ N_{23} \end{bmatrix} \\ I \\ 0 \end{bmatrix} .$$

The matrix $\hat{N}_{22}$ is similar to the matrix used in the Generalized Upper Bounding technique of linear programming. Hence $\hat{N}_{22}$ could be either maintained in product form or as an LU decomposition. The former is well described by Zoutendijk [43, Section 6.5] and efficient techniques for rank one modifications to an LU decomposition are described by Murtagh and Saunders [31] and Bartels and Golub [2].

In both the Murtagh-Saunders and revised Goldfarb methods, $s = n - (\bar{k} + \bar{\ell}' + n')$ where n' is the number of active type (3.4) constraints and $\bar{\ell}'$ is the number of active constraints of type (3.3). For the revised Goldfarb's method, $\hat{N}_{22}$ will be $\bar{\ell}' \times \bar{\ell}'$. For the reduced-gradient method and the Murtagh-Saunders method the matrix $\hat{N}_{22}$ will be $\bar{\ell} \times \bar{\ell}$. However, the matrix

$$\begin{bmatrix} I & N_{12} \\ N_{21} & N_{22} \end{bmatrix}$$

can be rewritten as

$$\left[\begin{array}{c|cc} I & \bar{N}_{12} & 0 \\ \hline \bar{N}_{21} & \bar{N}_{22} & 0 \\ 0 & 0 & I \end{array}\right]$$

where $\bar{N}_{22}$ is $\bar{\ell}' \times \bar{\ell}'$. Hence only the LU decomposition of an $\bar{\ell}' \times \bar{\ell}'$ matrix is needed. Thus both the Murtagh-Saunders and revised Goldfarb's method require the same amount of storage. However, for large scale problems (for example $s = 600$, $\bar{\ell}' = 200$, $n = 3300$, $\bar{k} = 1100$) neither method would be satisfactory. This leaves only the reduced-gradient method or some new version of the gradient projection method.

3.3 THE GRADIENT PROJECTION METHOD

The search direction $\underset{\sim}{p}$ used in the gradient projection method is determined by the operator P_N . The problem is to find the best way to calculate P_N . Obviously for large n (for example $n = 3000$), storing either P_N or N^+ would require prohibitive amounts of storage. In the original version of Rosen's method, the matrix $[NN^T]^{-1}$ is stored and updated whenever a row is added to or deleted from N_1. For a problem such as the telephone network design problem studied by Berry, NN^T will be $m \times m$, where $m \geqslant 1141$. In some cases m could be as large as about 3000. Any attempt to use the original method which stores the $m \times m$ matrix $[NN^T]^{-1}$, which will also in general be much denser that NN^T, is clearly unsatisfactory.

It was seen in Section 2.3 that for the Generalized Upper Bounding problem, P_N could be written as an explicit operator. For the augmented Generalized Upper Bounding

problem this can in general no longer be done. However it was found that the orthogonal projection matrix P_N does have the property that it can be computed in two stages. This result follows from the well known representation of the pseudo-inverse of a partitioned matrix [e.g. 7].

Lemma 3.1 Suppose $N = \begin{bmatrix} N_1 \\ N_2 \end{bmatrix}$ has full row rank. If P_{N_1} and P_N are the projection matrices corresponding to N_1 and N respectively and if

$$T = N_2 P_{N_1} N_2^T, \quad V = T^{-1} N_2 P_{N_1} \quad \text{and} \quad W = I - N_2^T V ,$$

then

(i) $$P_N = P_{N_1} W \tag{3.5}$$

(ii) $$N^+ = \begin{bmatrix} N_1^+ & V \end{bmatrix} . \tag{3.6}$$

Proof For a real matrix N, the following four matrix equations are used to define the full row rank pseudoinverse (generalized inverse) N^+:

(a) $N^T N^+ N^T = N^T$

(b) $N^+ N^T N^+ = N^+$

(c) $[N^T N^+]^T = N^T N^+$

(d) $[N^+ N^T]^T = N^+ N^T$.

Substitution of the right hand side of (3.6) into (a), (b), (c) and (d) shows that the four conditions are satisfied.
Then,

$$\begin{aligned} P_N &= I - N^T N^+ \\ &= I - N_1^T N_2^+ W - N_2^T V \\ &= W - N_1^T N_1^+ W \\ &= P_{N_1} W . \end{aligned}$$

The above Lemma expresses the relationship between P_N and P_{N_1} in terms of how P_N would be calculated if P_{N_1} were known. It is also interesting to notice that this relationship can be written as follows:

$$P_N = P_{N_1} - P_{N_1} N_2^T [N_2 P_{N_1} N_2^T]^{-1} N_2 P_{N_1} \tag{3.7}$$

$$= [I - (N_2 P_{N_1})^T [(N_2 P_{N_1})(N_2 P_{N_1})^T]^{-1} (N_2 P_{N_1})] P_{N_1}$$

which is the product of the orthogonal projection matrices for the mutually orthogonal matrices $N_2 P_{N_1}$ and N_1.

This means that if N_1 contains the normals of the, always active, constraints (3.2) and the active constraints from (3.4), then the operator expressions of (2.10), (2.11) and (2.12) can be used to full advantage. Hence with the rows of N_2 containing the active normals of the constraints (3.3), expressions for $P_N \underset{\sim}{w}$ and $N^+ \underset{\sim}{w}$ can be obtained, with the only requirement being that of maintaining a stable representation of the matrix $[N_2 P_{N_1} N_2^T]^{-1}$.

The dimension of the matrix $[N_2 P_{N_1} N_2^T]$ will be $\bar{\ell}' \times \bar{\ell}'$ which can be significantly smaller than NN^T. A method similar to that used in Rosen's algorithm, whereby $[N_2 P_{N_1} N_2^T]^{-1}$ is stored and updated when a row is added to or deleted from N could be used. When a row is added to or deleted from N_2 the operations required will be similar to those described in Rosen's method. However, when N_1 changes, and hence P_{N_1}, the task of updating $[N_2 P_{N_1} N_2^T]^{-1}$ is not very straightforward.

Instead of using Rosen's scheme of updating the inverse of a matrix it was decided to represent $N_2 P_{N_1} N_2^T$ by its

Cholesky decomposition LDL^T, where L is a unit diagonal lower triangular matrix and D is a diagonal matrix. This has an immediate advantage of needing almost half as much storage as that required for $[N_2 P_{N_1} N_2^T]^{-1}$. Another attractive feature of using a Cholesky decomposition is that the factorization LDL^T can be updated, economically and generally in a stable way, when either of the matrices N_1 and N_2 are changed.

3.3.1 A new row added to N_1

If the new N_1 matrix is

$$\bar{N}_1 = \begin{bmatrix} N_1 \\ \underset{\sim}{q}^T \end{bmatrix}$$

and $P_{\bar{N}_1}$ the corresponding projection matrix, then from (3.7), replacing N_2 by $\underset{\sim}{q}^T$,

$$P_{\bar{N}_1} = P_{N_1} - \frac{P_{N_1} \underset{\sim}{q}\, \underset{\sim}{q}^T P_{N_1}}{\underset{\sim}{q}^T P_{N_1} \underset{\sim}{q}}$$

so that

$$N_2 P_{\bar{N}_1} N_2^T = LDL^T + \sigma \underset{\sim}{z}\underset{\sim}{z}^T \quad , \qquad (3.8)$$

where

$$\underset{\sim}{z} = N_2 P_{N_1} \underset{\sim}{q} \quad \text{and} \quad \sigma = -1/\underset{\sim}{q}^T P_{N_1} \underset{\sim}{q} \ .$$

The new Cholesky factors $\bar{L}\bar{D}\bar{L}^T$ can then be calculated using some technique for the modification of the factorization after a rank one change. Such techniques will be discussed in Section 3.4.

3.3.2 A row deleted from N_1

When the last row of a matrix N is to be deleted, it has been established by Fletcher [13] that the pseudoinverse of the resulting matrix $\bar{N}$ is related to the pseudoinverse of N by

$$\begin{bmatrix} \bar{N}^+ \\ 0 \end{bmatrix} = N^+ - N^+ \frac{\underset{\sim}{w}\underset{\sim}{w}^T}{\underset{\sim}{w}^T\underset{\sim}{w}}$$

where $\underset{\sim}{w}^T$ is the last row of N^+.

It then follows that

$$[\bar{N}^T \quad 0]\begin{bmatrix} \bar{N}^+ \\ 0 \end{bmatrix} = N^T\begin{bmatrix} \bar{N}^+ \\ 0 \end{bmatrix}$$

$$= N^TN^+ - N^TN^+ \frac{\underset{\sim}{w}\underset{\sim}{w}^T}{\underset{\sim}{w}^T\underset{\sim}{w}}$$

and hence

$$P_{\bar{N}} = P_N + \frac{\underset{\sim}{w}\underset{\sim}{w}^T}{\underset{\sim}{w}^T\underset{\sim}{w}} \quad .$$

Thus, if the ℓth row in N_1 is deleted, it follows that

$$P_{\bar{N}_1} = P_{N_1} + \frac{\underset{\sim}{y}\underset{\sim}{y}^T}{\underset{\sim}{y}^T\underset{\sim}{y}}$$

where

$$\underset{\sim}{y} = N_1^{+T}\ \underset{\sim}{e}_\ell$$

and $\underset{\sim}{e}_\ell$ is the ℓth column of the identity matrix.

This means that

$$N_2 P_{\bar{N}_1} N_2^T = LDL^T + \sigma \underset{\sim}{z}\underset{\sim}{z}^T \qquad (3.9)$$

where

$$\underset{\sim}{z} = N_2\underset{\sim}{y} \quad \text{and} \quad \sigma = 1/\underset{\sim}{y}^T\underset{\sim}{y}.$$

Again the new Cholesky factors $\bar{L}\bar{D}\bar{L}^T$ of $N_2 P_{\bar{N}_1} N_2^T$ can be calculated using a suitable technique from those discussed in Section 3.4.

3.3.3 A new row added to N_2

The updating procedure associated with adding a new row to N_2 is precisely that of expanding the Cholesky factorization. Hence, the method can also be used to produce the initial Cholesky factorization or "re-invert" the factorization after a certain number of iterations.

If $N_2 P_{N_1} N_2^T$ has the Cholesky decomposition LDL^T and the new row to be added to N_2 is denoted by $\bar{\underset{\sim}{\gamma}}^T$, then

$$\begin{bmatrix} N_2 \\ \bar{\underset{\sim}{\gamma}}^T \end{bmatrix} P_{N_1} [N_2^T \quad \bar{\underset{\sim}{\gamma}}] = \begin{bmatrix} N_2 P_{N_1} N_2^T & N_2 P_{N_1} \bar{\underset{\sim}{\gamma}} \\ \bar{\underset{\sim}{\gamma}}^T P_{N_1} N_2^T & \bar{\underset{\sim}{\gamma}}^T P_{N_1} \bar{\underset{\sim}{\gamma}} \end{bmatrix}$$

will have the Cholesky decomposition $\bar{L}\bar{D}\bar{L}^T$, where

$$\bar{L} = \begin{bmatrix} L & 0 \\ \underset{\sim}{y}^T & 1 \end{bmatrix}$$

and

$$\bar{D} = \begin{bmatrix} D & 0 \\ 0 & \delta^2 \end{bmatrix}$$

and where $\underset{\sim}{y}$ is the solution of

$$LD\underset{\sim}{y} = N_2 P_{N_1} \bar{\underset{\sim}{\gamma}}$$

and

$$\delta^2 = \| P_{N_1} \bar{\underset{\sim}{\gamma}} \|_2^2 - \underset{\sim}{y}^T D \underset{\sim}{y} \; . \qquad (3.10)$$

This method is the simplest updating procedure and follows from the method described by Gill, Golub *et al* [18, p.532].

The simplicity of the method arises because only one transformation of a vector by the projection matrix P_{N_1} is used.

This updating method will only be satisfactory if the matrix $\bar{L}\bar{D}\bar{L}^T$ is well-conditioned. Problems with ill-conditioning will occur if the active constraints are almost linearly dependent. In this case the quantity δ^2 will theoretically be small. The method will in fact fail if the quantity defining δ^2 is either zero or negative. If the active constraints are actually linearly dependent, the new row does not need to be included in the matrix N_2 as the resulting search direction will automatically lie in the associated hyperplane. Thus, if the quantity δ^2 is small, zero or negative, the row could be omitted from N_2 with the hope that the method will move away from the current point to a point where the conditioning improves. However, such a scheme could lead to an infeasible direction, in which case the row not added to N_2 would be forcefully added if the algorithm were to continue.

It can be seen that δ^2 should be the quantity $\|P_N \bar{\underset{\sim}{\gamma}}\|_2^2$. Hence an alternative, although more computationally expensive, method for computing δ^2 is the following:

$$\begin{aligned} &\text{Solve} \quad L^T \underset{\sim}{w} = \underset{\sim}{y}. \\ &\text{Compute} \quad \underset{\sim}{v} = P_{N_1}(\bar{\underset{\sim}{\gamma}} - N_2^T \underset{\sim}{w}). \\ &\text{Set} \quad \delta^2 = \|\underset{\sim}{v}\|_2^2 . \end{aligned} \tag{3.11}$$

This method has the advantage that the quantity defining δ^2 in (3.11) can never be negative, but it will nevertheless be unsatisfactory if δ^2 is small. If the QR factorization

of $N_2 P_{N_1}$ were calculated by using the Gram-Schmidt method then the following factorization would be produced:

$$N_2 P_{N_1} Q_1 = LD^{\frac{1}{2}}$$

where

$$Q_1 Q_1^T = I.$$

The above updating procedure is then equivalent to performing the next step of the Gram-Schmidt method except that the matrix Q_1 is not available, but is instead represented by

$$Q_1 = P_{N_1} N_2^T (L^{-1})^T D^{-\frac{1}{2}} .$$

The Gram-Schmidt method however is well known to be numerically unreliable and as the above method has the added disadvantage of not having Q_1, then the method cannot be expected to be satisfactory if δ^2 is small.

It is a generally held opinion [e.g. 18, Section 5] that the best method of computing the orthogonal (or LQ) factorization of $N_2 P_{N_1}$, $N_2 P_{N_1} Q = [LD^{\frac{1}{2}} \; 0]$, is to store Q and use Given's matrices or Householder transformations for updating the factorization when either a row is added to or deleted from $N_2 P_{N_1}$. In this case, the projection matrix P_N and the pseudoinverse of $N_2 P_{N_1}$ are easily calculated as follows:

$$P_N = P_{N_1} Q_2 Q_2^T$$

and

$$[N_2 P_{N_1}]^+ = (L^{-1}) \; D^{-\frac{1}{2}} Q_1^T$$

where

$$Q = [Q_1 \; Q_2] .$$

If the LQ factorization were used, schemes similar to those described in Sections 3.3.1 and 3.3.2 together with the description of methods for a rank one update of the factorization in Gill and Murray [20, Section 4] could be used when N_1 changes. For changes in N_2 the methods described by Gill and Murray [20, Section 5] could be used. However the orthogonal matrix Q is an $n \times n$ matrix and thus for large scale problems, the use of such a method, although numerically stable, would be unreasonable in terms of computer storage.

At this stage, one alternative is to assume that in most practical applications the matrices LDL^T will be well-conditioned and to use either of the two above methods for updating L and D. However, research in the related area of least-squares solutions has made one other possibility available. This is the method of *iterative refinement* which has arisen from studies in refining least-squares solutions. The method can be easily extended to include the refinement of $\underset{\sim}{y}$ and δ^2 .

Given the matrices

$$R = D^{\frac{1}{2}}L^T, \quad \bar{R} = \begin{bmatrix} R & \underset{\sim}{p} \\ 0 & \delta \end{bmatrix}$$

and the vector

$$\underset{\sim}{p} = D^{\frac{1}{2}}\underset{\sim}{y}$$

such that

$$R^T R = N_2 P_{N_1} N_2^T ,$$

the problem is to find $\underset{\sim}{p}$ and δ such that

$$\bar{R}^T\bar{R} = \begin{bmatrix} N_2 \\ \bar{\underset{\sim}{\gamma}}^T \end{bmatrix} P_{N_1} [N_2^T \ \bar{\underset{\sim}{\gamma}}] \ .$$

This problem is then simply the problem of updating the QR factorization of $N_2 P_{N_1}$ where Q is orthogonal and R is upper triangular. The difficulty here, as before, is that Q is not available. This situation has been outlined by a discussion given by Gill, Golub *et al* [18, p.532], but no satisfactory algorithm is given. The problem has not yet been adequately treated in the literature, although there are two promising methods available.

The problem of updating R is intimately connected with the least-squares problem

$$\min_{\underset{\sim}{x}} \| P_{N_1} N_2^T \underset{\sim}{x} - P_{N_1} \bar{\underset{\sim}{\gamma}} \|_2$$

which has the solution $\underset{\sim}{x} = (N_2 P_{N_1})^+ \bar{\underset{\sim}{\gamma}}$. In order to obtain a sufficiently accurate solution $\underset{\sim}{x}$ for this least-squares problem it is often necessary to use the method of iterative refinement. A complete description of the least-squares problem, the use of the QR factorization and iterative refinement can be found in Stewart [39, Chapter 5] .

A method that does not use Q follows from a suggestion by Kahan in the paper by Golub and Wilkinson [23] in 1966. This can be stated as follows:

(i) Solve $\quad R^T \underset{\sim}{p}_0 = N_2 P_{N_1} \bar{\underset{\sim}{\gamma}}$

$$R \ \underset{\sim}{x}_0 = \underset{\sim}{p}_0 \quad .$$

(ii) For $i = 0,1,2,\ldots$

Compute $\underset{\sim}{r}_i = P_{N_1}(\bar{\underset{\sim}{\gamma}} - N_2^T \underset{\sim}{x}_i)$ (3.12)

$\underset{\sim}{g}_i = N_2 \underset{\sim}{r}_i$. (3.13)

Solve $R^T \underset{\sim}{\delta p}_i = \underset{\sim}{g}_i$

$R \underset{\sim}{\delta x}_i = \underset{\sim}{\delta p}_i$.

Compute $\underset{\sim}{p}_{i+1} = \underset{\sim}{p}_i + \underset{\sim}{\delta p}_i$

$\underset{\sim}{x}_{i+1} = \underset{\sim}{x}_i + \underset{\sim}{\delta x}_i$.

Exit if $\| \underset{\sim}{g}_i \|_2$ is sufficiently small.

(iii) Set $\delta = \| \underset{\sim}{r}_i \|_2$.

The quantity $\underset{\sim}{r}_i$ in (3.12) should be calculated using double precision and should be saved as a double precision vector. The quantity $\underset{\sim}{g}_i$ in (3.13) can then be calculated using double precision accumulation (single precision N_2 times double precision $\underset{\sim}{r}_i$).

Another method that does not use Q has been recently given by Björck [6] in his algorithm 2'. This is similar to the above method but slightly more complicated. Björck's algorithm can be described with a change of notation and a slight extension to obtain $\underset{\sim}{p}$ and δ^2 as follows:

(i) Solve $R^T \underset{\sim}{p}_0 = N_2 P_{N_1} \bar{\underset{\sim}{\gamma}}$

$R \underset{\sim}{x}_0 = \underset{\sim}{p}_0$.

Set $\underset{\sim}{r}_0 = P_{N_1}(\bar{\underset{\sim}{\gamma}} - N_2^T \underset{\sim}{x}_0)$.

(ii) For $i = 0,1,2,\ldots$

Compute $\underset{\sim}{f}_i = P_{N_1}(\bar{\underset{\sim}{\gamma}} - N_2^T \underset{\sim}{x}_i) - \underset{\sim}{r}_i$ (3.14)

$\underset{\sim}{g}_i = N_2 P_{N_1} \underset{\sim}{r}_i$. (3.15)

Solve $R^T \underset{\sim}{\delta p}_i = N_2 P_{N_1} \underset{\sim}{f}_i + \underset{\sim}{g}_i$

$R \underset{\sim}{\delta x}_i = \underset{\sim}{\delta p}_i$.

Compute $\underset{\sim}{\delta r}_i = \underset{\sim}{f}_i - P_{N_1} N_2^T \underset{\sim}{\delta x}_i$

$\underset{\sim}{p}_{i+1} = \underset{\sim}{p}_i + \underset{\sim}{\delta p}_i$

$\underset{\sim}{x}_{i+1} = \underset{\sim}{x}_i + \underset{\sim}{\delta x}_i$

$\underset{\sim}{r}_{i+1} = \underset{\sim}{r}_i + \underset{\sim}{\delta r}_i$.

Exit if $\| \underset{\sim}{g}_i \|_2$ is sufficiently small.

(iii) Set $\delta = \| \underset{\sim}{r}_i \|_2$.

The quantity $\underset{\sim}{f}_i$ measures the failure of the current fit $P_{N_1} N_2^T \underset{\sim}{x}_i$ and the residuals $\underset{\sim}{r}_i$ to go together and the quantity $\underset{\sim}{g}_i$ reflects the failure of the residuals $\underset{\sim}{r}_i$ to be orthogonal to the rows of $N_2 P_{N_1}$. For this reason it is essential that $\underset{\sim}{f}_i$ and $\underset{\sim}{g}_i$ be accurately computed. As a result the computations (3.14) and (3.15) should be done using double precision accumulation.

In both methods it could also be that $\| \underset{\sim}{g}_0 \|_2$ is negligible, so that the extra work involved will not be substantial. In fact it is common practice to use only single precision arithmetic for the first iteration. This will allow further economy in the likely event that $\| \underset{\sim}{g}_0 \|_2$ does prove to be small.

3.3.4 A row deleted from N_2

Consider the case in which the ℓth row of N_2 is to be deleted. For convenience let L and D be partitioned as follows:

$$L = \begin{bmatrix} L_1 & 0 & 0 \\ L_2 & 1 & 0 \\ L_3 & L_4 & L_5 \end{bmatrix}$$

and

$$D = \begin{bmatrix} D_1 & & \\ & d & \\ & & D_2 \end{bmatrix}$$

where L_1 and D_1 are $(\ell-1)\times(\ell-1)$, $L_2^T \in R^{\ell-1}$, L_3 is $(r-\ell)\times(\ell-1)$, $L_4 \in R^{r-\ell}$ and L_5 and D_5 are $(r-\ell)\times(r-\ell)$.

If $\bar{Q}$ is the orthogonal permutation matrix

$$\bar{Q} = \begin{bmatrix} I_{\ell-1} & 0 & 0 \\ 0 & 0 & I_{r-\ell} \\ 0 & 1 & 0 \end{bmatrix}$$

then

$$\bar{Q}L = \begin{bmatrix} L_1 & 0 & 0 \\ L_3 & L_4 & L_5 \\ L_2 & 1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} L_1 & 0 & 0 \\ L_3 & L_5 & L_4 \\ L_2 & 0 & 1 \end{bmatrix} \bar{Q}$$

so that

$$\bar{Q}LDL^T\bar{Q}^T = \begin{bmatrix} L_1 & 0 & 0 \\ L_3 & L_5 & L_4 \\ L_2 & 0 & 1 \end{bmatrix} \begin{bmatrix} D_1 & & \\ & D_2 & \\ & & d \end{bmatrix} \begin{bmatrix} L_1^T & L_3^T & L_2^T \\ 0 & L_5^T & 0 \\ 0 & L_4^T & 1 \end{bmatrix}$$

$$= \begin{bmatrix} L_1 & 0 & 0 \\ L_3 & I_{r-\ell} & 0 \\ L_2 & 0 & 1 \end{bmatrix} \begin{bmatrix} D_1 & 0 & 0 \\ 0 & \bar{L}_5\bar{D}_2\bar{L}_5^T & dL_4 \\ 0 & dL_4^T & d \end{bmatrix} \begin{bmatrix} L_1^T & L_3^T & L_2^T \\ 0 & I_{r-\ell} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where

$$\bar{L}_5\,\bar{D}_2\,\bar{L}_5^T = L_5\,D_2\,L_5^T + d\,L_4\,L_4^T\,. \qquad (3.16)$$

The modified factors can again be computed by using a suitable rank one updating method.

It then follows that the Cholesky decomposition of $\bar{N}_2 P_{N_1} \bar{N}_2^T$ is $\bar{L}\bar{D}\bar{L}^T$, where

$$\bar{L} = \begin{bmatrix} L_1 & 0 \\ L_3 & \bar{L}_5 \end{bmatrix}$$

and

$$\bar{D} = \begin{bmatrix} D_1 & 0 \\ 0 & \bar{D}_2 \end{bmatrix} .$$

3.4 RANK ONE MODIFICATION OF A CHOLESKY DECOMPOSITION

The success of using the method outlined in Section 3.3 depends heavily on the ability to be able to calculate the Cholesky decomposition of a symmetric positive definite matrix after a rank one change, in a numerically stable manner. The problem of achieving this has been extensively discussed in the literature [14, 18, 20, 21].

Let T be a symmetric positive definite matrix such that $T = LDL^T$. Let $\bar{T} = T + \sigma \underset{\sim}{z}\underset{\sim}{z}^T$ and $\bar{T} = \bar{L}\bar{D}\bar{L}^T$. The problem then is to find an efficient and numerically stable way to calculate $\bar{L}\bar{D}\bar{L}^T$ given LDL^T and the quantities σ and $\underset{\sim}{z}$. It can be noted that if $\sigma > 0$, as is the case in (3.9) and (3.16), then $\bar{T}$ must be positive definite. However, if $\sigma < 0$, as is the case in (3.8), then there is a positive scalar η such that for $\sigma < -\eta$, $\bar{T}$ is not positive definite. As this situation can only occur when a new row is added to N_1 a simple test could be used to ensure that such a situation does not occur. All this test would need

to do is to test for $P_N q = 0$, where q^T is the new row added to N_1, this is because the new matrix $N_2 P_{\bar{N}_1} N_2^T$ will be positive definite unless the new matrix $\bar{N}$ does not have full row rank. Hence it can be assumed that if $\sigma < 0$ then theoretically $\bar{T}$ will be positive definite.

Fletcher and Powell [14] considered possible updating procedures and performed error analyses on such methods. As a result they developed a composite procedure which used what they considered to be the best methods for overcoming problems that could occur for the respective cases $\sigma > 0$ and $\sigma < 0$. The *composite t-method* was the resulting recommended method. This can be described as follows:

Terminate if $\sigma = 0$: Let $t_1 = \sigma^{-1}$ and $z^{(1)} = z$.

Then

(i) if $\sigma > 0$ go to (v),

(ii) if v is not available, solve $Lv = z$,

(iii) compute $t_{i+1} = t_i + v_i^2/d_i$, $i = 1,\ldots,n$,

(iv) if any $t_{i+1} \geqslant 0$, recompute the t_i from

$$t_{n+1} = \epsilon/\sigma$$

$$t_i = t_{i+1} - v_i^2/d_i, \quad i = n,n-1,\ldots,1$$

where ϵ is the relative machine precision,

(v) repeat for $i = 1,\ldots,n$ the following sequence

$$v_i = z_i^{(i)},$$

if $\sigma > 0$ then $t_{i+1} = t_i + v_i^2/d_i$,

$$\alpha_i = t_{i+1}/t_i,$$

$$\bar{d}_i = d_i \alpha_i,$$

terminate if $i = n$,

$$\beta_i = (v_i/d_i)/t_{i+1},$$

$$z^{(i+1)} = z^{(i)} - v_i \ell_i,$$

if $\alpha_i > 4$ then $\gamma_i = t_i/t_{i+1}$ and (3.17)

$$\bar{\ell}_i = \gamma_i \ell_i + \beta_i z^{(i)} \text{ else } \bar{\ell}_i = \ell_i + \beta_i z^{(i+1)}.$$

The two main possible areas of trouble in modifying a Cholesky factorization have been recognised in the literature [14,20]. The first is the case when $\bar{T}$ is almost singular as can happen when $\sigma < 0$. This can lead to rounding errors and possible negative elements on the diagonals of $\bar{D}$. If this occurs the situation is recognised in step (iv) of the composite t-method. It can be easily verified that $\bar{d}_j$, $j = 1,\ldots,n$ will then be guaranteed to be positive irrespective of any rounding errors made in the computation. This also guarantees that $\bar{T}$ will be positive definite. The second possible area of trouble can occur when $\sigma > 0$. In this case, if the "simpler" method were used (i.e. remove the test "if $\alpha_i > 4$" and just use (3.17)) then there will be a potential error related to the fact that a bound on the error in the ith column of $\bar{L}$ involves the term $\bar{d}_i/d_i$ which could be arbitrarily large. Gill and Murray indicate [20,p.61] that the term $\bar{d}_i/d_i$ appears only in a bound on the error and analyses only indicate potential instability in the simpler method. The authors also point out that they had not encountered an example with matrices T and $\bar{T}$ having condition numbers less than 2^t for which this bound is achieved, where 2^{-t} is the relative machine precision of floating point arithmetic. This means that future analysis of the simpler method, which requires one less multiplication in the updating of an element of L, may show that it could be as stable as the composite t-method.

CHAPTER 4

THE DOUBLE STRUCTURED PROBLEM

The methods considered in Chapter 3 are based on the assumption that the constraints (3.3) have no particular structure. In the case where the additional constraints do have a given structure it seems desirable to use some other method for calculating feasible descent directions. This problem was approached by considering a class of problems which could be described as *transportation type problems.*

4.1 THE TRANSPORTATION TYPE PROBLEM

These problems can be formulated as follows:

$$\min \quad f(\underset{\sim}{x}) \tag{4.1}$$

$$\text{s.t.} \quad \sum_{j \in A_k} x_j^k = b^k \ , \quad k = 1,\ldots,\bar{k} \tag{4.2}$$

$$\sum_{k \in B_j} x_j^k \geqslant d_j \ , \quad j = 1,\ldots,\bar{j} \tag{4.3}$$

$$x_j^k \geqslant 0, \ j = 1,\ldots,\bar{j}, \ k = 1,\ldots,\bar{k}, \tag{4.4}$$

where for $k = 1,\ldots,\bar{k}$

$$A_k \subseteq \{1,\ldots,\bar{j}\}$$

and for $j = 1,\ldots,\bar{j}$

$$B_j \subseteq \{1,\ldots,\bar{k}\} \ .$$

The class of problems is given this name because if

$$A_k = \{1,\ldots,\bar{j}\} \quad \text{for} \quad k = 1,\ldots,\bar{k} \ ,$$

$$B_j = \{1,\ldots,\bar{k}\} \quad \text{for} \quad j = 1,\ldots,\bar{j} \ ,$$

$\bar{j} = \bar{k}$ and the inequalities of (4.3) are replaced by equalities, then the constraints are those of a transportation problem.

It can also be noted that if

$$A_k = \{1,\ldots,\phi(k)\} \quad \text{for} \quad k = 1,\ldots,\bar{k}$$

and

$$B_j = \phi \quad \text{for} \quad j = 1,\ldots,\bar{j}$$

where $\bar{j} = \max_{1 \leqslant k \leqslant \bar{k}} \phi(k)$, then the class of problems is that of Chapter 2.

If N is the matrix whose rows contain the normals of the active constraints from (4.2), (4.3) and (4.4) then N $(m \times n)$ can, after suitable column permutations, be written as

$$N = \begin{bmatrix} N_{11} & N_{12} \\ 0 & I \end{bmatrix} \tag{4.5}$$

where

$$N_{11} = \begin{bmatrix} A_1 & & \\ & \ddots & \\ & & A_{\bar{k}} \\ B_1 & \ldots & B_{\bar{k}} \end{bmatrix}$$

$$A_k = [\underbrace{1 \ \ldots \ 1}_{\bar{s}_k} \mid \underbrace{1 \ \ldots \ 1}_{\hat{s}_k} \mid \underbrace{0 \ \ldots \ 0}_{\bar{t}_k} \mid \underbrace{0 \ \ldots \ 0}_{\hat{t}_k}] \tag{4.6}$$

$$B_k = [\underbrace{\bar{I}_k}_{\bar{s}_k} \mid \underbrace{0}_{\hat{s}_k} \mid \underbrace{\hat{I}_k}_{\bar{t}_k} \mid \underbrace{0}_{\hat{t}_k}]\}q \tag{4.7}$$

and the matrix $[\bar{I}_k \mid \hat{I}_k]$, with full column rank, contains distinct columns of the identity matrix I_q. This can always be done as any column not corresponding to an active non-negativity constraint can contain at most two non-zero elements. As a result there are four possible forms such a column can have. These are represented in (4.6) and (4.7).

The transportation type problems will be approached with the idea of using column transformations. To do this it is convenient to define two types of column transformations. The first is called a *type A transformation.* This is an elementary column transformation which by post multiplication transforms one of the first $\bar{k}$ rows of N into a vector $e_{\sim \ell}^T$ which is some row of the identity matrix I_n. For example if the row of N in question is

$$[0\ 0\ 1\ 1\ 1\ 0\ 0\ 0]$$

then such a type A transformation matrix is

$$\left[\begin{array}{cc|ccc|ccc} 1 & & & & & & & \\ & 1 & & & & & & \\ \hline & & 1 & 0 & 0 & & & \\ & & 0 & 1 & 0 & & & \\ & & -1 & -1 & 1 & & & \\ \hline & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{array}\right].$$

The resulting row vector will be $e_{\sim s}^T$.

The other transformation is called a *type B transformation.* This is similar to a type A transformation except that it transforms one of the rows $\bar{k}+1,\ldots,\bar{k}+q$ of N. For example

if the row of N is

$$[1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0]$$

then such a type B transformation is

$$\left[\begin{array}{ccc|ccc|ccc} 1 & & & & & & & & \\ & 1 & & & & & & & \\ & & 1 & & & & & & \\ \hline -1 & 0 & 0 & 1 & & & -1 & 0 & 0 \\ & & & & 1 & & & & \\ & & & & & 1 & & & \\ \hline & & & & & & 1 & & \\ & & & & & & & 1 & \\ & & & & & & & & 1 \end{array}\right] .$$

The resulting row vector in this case is $e_{\sim 4}^{T}$. For this example the ***pivot column*** will be column 4.

The idea behind the approach is to use pivot columns that have only one non-zero element. The motivation is to transform N into a matrix M such that the rows of M contain distinct rows of the identity matrix I_n i.e. there exists a permutation matrix S such that

$$M = [I \quad 0]S . \tag{4.8}$$

Lemma 4.1 If N has full row rank, then it can be transformed by a sequence of type A and type B transformations to a matrix M of the form (4.8).

Proof (i) It can be seen from (4.6) and (4.7) that there can exist a k such that $\bar{t}_k \neq 0$. If this is the case then $\bar{t}_k$ type B transformations can be performed. After these transformations, the resulting N matrix can be changed by row and column permutations to a matrix with the same form as (4.5).

(ii) If $\bar{t}_k = 0$ for $k = 1,\ldots,\bar{k}$ then there must be some k such that $\hat{s}_k \neq 0$, otherwise by (4.6) and (4.7), N would not have full row rank. Hence a type A transformation can be performed, using for example, the column of N corresponding to the $\bar{s}_k+1$ th column of A_k. The resulting N matrix can then be changed via row and column permutations to a matrix with the same form as (4.5).

(iii) If $\hat{s}_k = 0$, $k = 1,\ldots,\bar{k}$ either $q = 0$, or there must be some k such that $\bar{t}_k \neq 0$, otherwise by (4.6) and (4.7), N will not have full row rank. Hence if $q \neq 0$, a type B transformation can be performed.

Thus by (i), (ii) and (iii) it can be seen that there is always a suitable candidate column for either a type A or type B transformation, except in the instance when $q = 0$ and $\hat{s}_k = 0$, $k = 1,\ldots,\bar{k}$ in which case the transformation is complete.

An algorithm can be constructed which follows from the proof of this Lemma. The following algorithm with $N = \{1,\ldots,n\}$ will generate index sets which can be used to calculate the vectors $E\underset{\sim}{w}$ and $E^T\underset{\sim}{w}$ for an arbitrary vector $\underset{\sim}{w}$, where E is the nonsingular matrix such that $NE = M$.

For reasons of notational convenience the term *alternative index* will be introduced. The term alternative index is used in the sense that, for example, $\underset{\sim}{x}^T = [x_1^1, x_2^1, x_1^2, \ldots]$ can also be written as $[x_1, x_2, x_3, \ldots]$. Also for notational convenience the sets $\bar{A}_k$ and $\bar{B}_j$ will be used. They are defined as follows:

If $j \in A_k$, then $s \in \bar{A}_k$, where s is the alternative index of x_j^k.

If $k \in B_j$, then $t \in \bar{B}_j$, where t is the alternative index of x_j^k.

Algorithm 4.1

Given the set N, let N_0 contain the alternative indices of the active type (4.4) constraints.

Set $\bar{N}_0 = N \setminus N_0$

$$K_0 = (\bigcup_{k=1}^{\bar{k}} \bar{A}_k) \cap \bar{N}_0$$

$$J_0 = (\bigcup_{j=1}^{\bar{j}} \bar{B}_j) \cap \bar{N}_0$$

$j^* = q$

$k^* = \bar{k}$.

(a) If $j^* = 0$ and $k^* = 0$ then stop.

Set $I_i = J_i \cap K_i$.

If $J_i \setminus I_i = \phi$ then go to (b).

Choose some $s_{i+1} \in J_i \setminus I_i$.

If $s_{i+1} \in \bar{B}_{j_{i+1}}$ then set

$$J_{i+1} = J_i \setminus (\bar{B}_{j_{i+1}} \cap \bar{N}_i),$$
$$\bar{N}_{i+1} = \bar{N}_i \setminus \{s_{i+1}\},$$
$$K_{i+1} = K_i,$$
$$j^* = j^* - 1.$$

Go to (a).

(b) If $K_i \setminus I_i = \phi$ then go to (c).

Choose some $t_{i+1} \in K_i \setminus I_i$.

If $t_{i+1} \in \bar{A}_{k_{i+1}}$, then set

$$K_{i+1} = K_i \setminus (\bar{A}_{k_{i+1}} \cap \bar{N}_i),$$
$$\bar{N}_{i+1} = \bar{N}_i \setminus \{t_{i+1}\},$$
$$J_{i+1} = J_i,$$
$$k^* = k^* - 1.$$

Go to (a).

(c) The set of active constraints is linearly dependent, thus by (4.6) and (4.7) a constraint of type (4.3) can be dropped.

Choose some $r \in I_i$.

Let $r \in \bar{B}_{\hat{j}}$.

Set $J_i = J_i \setminus (\bar{B}_{\hat{j}} \cap \bar{N}_i)$,

$j^* = j^* - 1$.

Go to (a).

The sets $\bar{N}_i$, K_i, J_i and I_i can be given the following interpretation,

(i) $\bar{N}_i$ contains the set of indices of possible pivot columns.

(ii) K_i contains the set of indices of pivot columns that could be used for type A transformations.

(iii) J_i contains the set of indices of pivot columns that could be used for type B transformations.

(iv) I_i contains the set of indices of pivot columns that could be used for both type A and B transformations.

In step (a) of the algorithm, the set $J_i \setminus I_i$, which is the set of indices of the pivot columns that could only be used for type B transformations, is examined. If this set is non-empty, then some element s_{i+1} is chosen. The indices corresponding to the transformed row are then deleted from J_i and s_{i+1} is removed from $\bar{N}_i$. Step (b) of the algorithm is similar. In step (c), one of the untransformed constraints of type (4.3) are deleted and the corresponding indices removed from J_i.

The sequence of transformations can then be summarized as follows:

ITERATION	1	2	3	4	5	. . .
TYPE A	k_1	-	-	-	k_5	
TYPE B	-	j_2	j_3	j_4	-	
COLUMN	t_1	s_2	s_3	s_4	t_5	

TABLE 1

This means that the transformation matrix E can be written as a product of elementary matrices $E = E_1E_2E_3E_4E_5 \ldots$. The numbers s_i and t_i indicate the pivot columns used. The numbers j_i and k_i are the indices of the associated sets $\bar{B}_j$ and $\bar{A}_k$ respectively, whose elements indicate which columns are involved in the transformation.

The class of problems can be simply extended to include related problems. For example,

(i) inequality constraints in (4.2)

(ii) equality or less than or equal to inequality constraints in (4.3)

(iii) x_j^k could be replaced by $a_j^k x_j^k$ where $a_j^k \neq 0$.

An example of such a problem is the *assortment problem* with nonlinear costs reported by Pentico [32]:

$$\min \quad \sum_{j=1}^{m} c_j(x_j^{m+1}) + \sum_{j=1}^{m} \sum_{k=1}^{m} c_j^k(x_j^k)$$

$$\text{s.t.} \quad \sum_{j=1}^{k} x_j^k = b^k \ , \ k = 1,\ldots,m$$

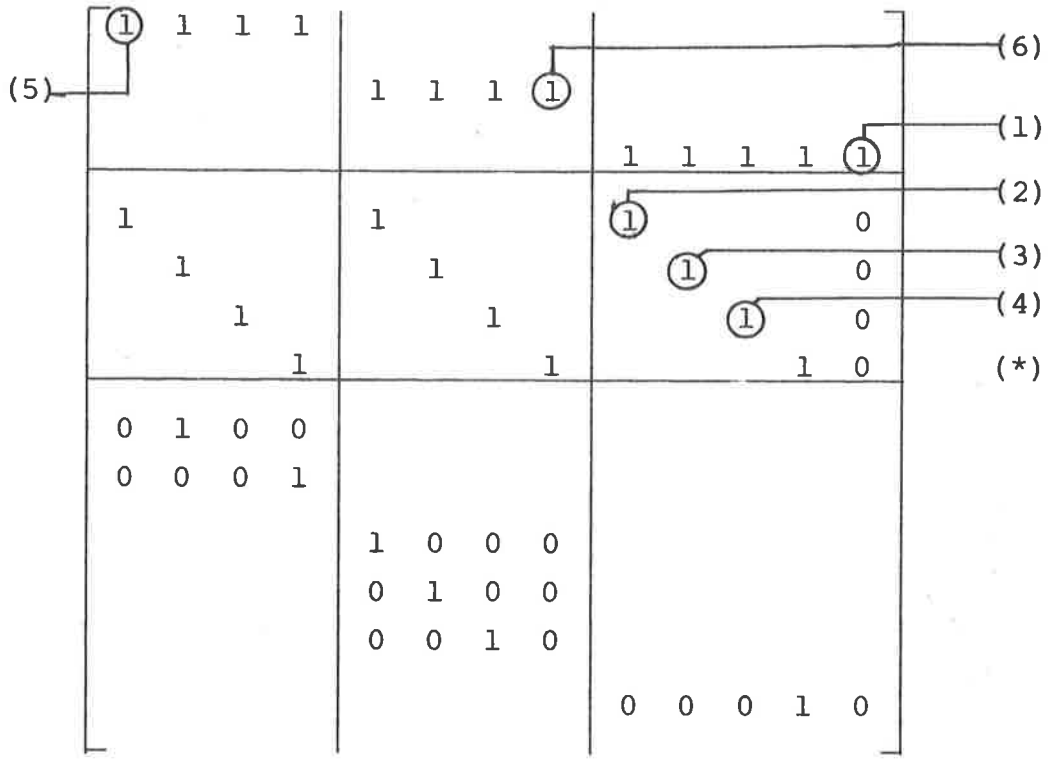
$$\sum_{k=j}^{m} x_j^k - x_j^{m+1} = 0, \ j = 1,\ldots,m$$

$$x_j^k \geq 0, \ \forall j,k.$$

4.2 A NUMERICAL EXAMPLE

Let the matrix N be as follows:

N =



(4.9)

$N = \{1,2,\ldots,13\}$, $N_0 = \{2,4,5,6,7,12\}$, $\bar{N}_0 = \{1,3,8,9,10,11,13\}$

$A_1 = \{1,2,3,4\}$, $A_2 = \{1,2,3,4\}$, $A_3 = \{1,2,3,4,5\}$

$\bar{A}_1 = \{1,2,3,4\}$, $\bar{A}_2 = \{5,6,7,8\}$, $\bar{A}_3 = \{9,10,11,12,13\}$

$B_1 = \{1,2,3\}$, $B_2 = \{1,2,3\}$, $B_3 = \{1,2,3\}$, $B_4 = \{1,2,3\}$

$\bar{B}_1 = \{1,5,9\}$, $\bar{B}_2 = \{2,6,10\}$, $\bar{B}_3 = \{3,7,11\}$, $\bar{B}_4 = \{4,8,12\}$

$K_0 = \{1,3,8,9,10,11,13\}$, $J_0 = \{1,3,8,9,10,11\}$

$j^* = 4$, $k^* = 3$.

(i) $I_0 = \{1,3,8,9,10,11\}$, $J_0 \setminus I_0 = \phi$, $K_0 \setminus I_0 = \{13\}$

$\therefore t_1 = 13 \in \bar{A}_3 \Rightarrow k_1 = 3$

$K_1 = \{1,3,8\}$, $\bar{N}_1 = \{1,3,8,9,10,11\}$, $J_1 = \{1,3,8,9,10,11\}$

$j^* = 4$, $k^* = 2$.

(ii) $I_1 = \{1,3,8\}$, $J_1 \setminus I_1 = \{9,10,11\}$.

Let $s_2 = 9 \in \bar{B}_1 \quad \therefore j_2 = 1$

$J_2 = \{3,8,10,11\}$, $\bar{N}_2 = \{1,3,8,10,11\}$, $K_2 = \{1,3,8\}$

$j^* = 3$, $k^* = 2$.

(iii) $I_2 = \{3,8\}$, $J_2 \setminus I_2 = \{10,11\}$

Let $s_3 = 10 \in \bar{B}_2 \quad \therefore j_3 = 2$

$J_3 = \{3,8,11\}$, $K_3 = \{1,3,8\}$, $\bar{N}_3 = \{1,3,8,11\}$

$j^* = 2$, $k^* = 2$.

(iv) $I_3 = \{3,8\}$, $J_3 \setminus I_3 = \{11\}$

$s_4 = 11 \in \bar{B}_3 \Rightarrow j_4 = 3$

$J_4 = \{8\}$, $K_4 = \{1,3,8\}$, $\bar{N}_4 = \{1,3,8\}$

$j^* = 1$, $k^* = 2$.

(v) $I_4 = \{8\}$, $J_4 \setminus I_4 = \phi$, $K_4 \setminus I_4 = \{1,3\}$

Let $t_5 = 1 \in A_1 \quad \therefore k_5 = 1$

$K_5 = \{8\}$, $\bar{N}_5 = \{3,8\}$, $J_5 = \{8\}$

$j^* = 1$, $k^* = 1$.

(vi) $I_5 = \{8\}$, $J_5 \setminus I_5 = \phi$, $K_5 \setminus I_5 = \phi$

$r = 8 \in \bar{B}_4 \Rightarrow \hat{j} = 4$

$J_5 = \phi$, $K_5 = \{8\}$

$j^* = 0$, $k^* = 1$.

(vii) $I_5 = \phi$, $J_5 \setminus I_5 = \phi$, $K_5 \setminus I_5 = \{8\}$

$\therefore t_6 = 8 \in \bar{A}_2 \Rightarrow k_6 = 2$

$K_6 = \phi$, $\bar{N}_6 = \{3\}$, $J_6 = \phi$

$j^* = 0$, $k^* = 0 \quad \therefore$ stop.

The sequence of pivots for this problem are those shown in (4.9) as (1),(2),...,(6). The row (*) was identified as being linearly dependent on the other rows and was subsequently deleted. Thus the matrix E, such that $\bar{N}E = M$, where $M = [e_{\ell_1}, \ldots, e_{\ell_m}]^T$, ℓ_i distinct, $\{\ell_1, \ldots, \ell_m\} = N \setminus \bar{N}_6$ and $\bar{N}$ is the matrix (4.9) with the seventh row deleted, has been essentially identified. The matrix E can be written as a product of elementary matrices that are either type A or type B transformations. This means that expressions such as $E\underset{\sim}{w}$, $E^T\underset{\sim}{w}$, $E^{-1}\underset{\sim}{w}$ and $(E^{-1})^T\underset{\sim}{w}$, for an arbitrary vector $\underset{\sim}{w}$, can be calculated with the only information needed being that held in a list of the form as given in Table 4.1. For the above example E_1 would be

$$E_1 = \left[\begin{array}{cccc|cccc} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ \hline & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & -1 & -1 & -1 & 1 \end{array}\right] .$$

4.3 A MODIFIED GRADIENT PROJECTION ALGORITHM

In Section 2.3, a simplified statement of Rosen's gradient projection algorithm was given. The algorithm uses the matrix N whose rows contain the normals of the currently active constraints, where the constraints can be assumed to be written as

$$A\underset{\sim}{x} \geqslant \underset{\sim}{b}$$

for some A and $\underset{\sim}{b}$.

However there is no obvious *a priori* reason why the feasible region should have been described by $A\underset{\sim}{x} \geqslant \underset{\sim}{b}$ and not by

$$AE\underset{\sim}{y} \geqslant \underset{\sim}{b}$$

where the variables $\underset{\sim}{y}$ are given by

$$\underset{\sim}{y} = E^{-1}\underset{\sim}{x} \quad .$$

In this new "y-space", the matrix whose rows contain the normals of the active constraints will be

$$M = NE \quad .$$

The corresponding orthogonal projection matrix P_M will be

$$P_M = I - M^T[MM^T]^{-1}M \quad .$$

This matrix could be used in the y-space to determine a direction of search

$$- P_{M\underset{\sim}{\nabla}_y} f(\underset{\sim}{y}) \quad .$$

Since $\underset{\sim}{y} = E^{-1}\underset{\sim}{x}$, it can be easily shown that

$$\underset{\sim}{\nabla}_y f(\underset{\sim}{y}) = E^T \underset{\sim}{\nabla} f(\underset{\sim}{x}) \quad .$$

Thus the direction

$$- EP_M E^T \underset{\sim}{\nabla} f(\underset{\sim}{x})$$

would be an equally as valid feasible descent direction in the x-space as the usual direction $- P_{N\underset{\sim}{\nabla}} f(\underset{\sim}{x})$. Hence, there exists a class of projection operators of the form

$$Q_M = EP_M E^T$$

where E is nonsingular and $M = NE$, that have the following properties:

(i) $Q_M N^T = 0$

(ii) $-Q_M \nabla f(\underset{\sim}{x})$ is a direction of decrease of f.

There are several special forms that E can have:

(a) $E = I$. In this case $Q_M = P_M = P_N$.

(b) E is orthogonal. This is a necessary and sufficient condition for $Q_M = P_N$.

(c) $E^T E = G$ where G is the Hessian (or positive definite approximation to the Hessian). Then

$$Q_M = G^{-1} - G^{-1} N^T [NG^{-1} N^T]^{-1} NG^{-1}$$

which is Goldfarb's variable metric projection operator.

(d) E is such that M has some special property. For example E could be such that M has Generalized Upper Bounding structure. In this case P_M can be calculated as in Section 2.3 and $Q_M = EP_M E^T$.

Case (d) only will be considered, as the motivation behind using the transformation E is to transform the original problem of calculating a search direction into a new problem with significantly reduced computational complexity.

For transportation type problems it has been shown in Section 4.1 that there exists an E such that $M(= NE)$ has the property $MM^T = I$. In this case, the operation of P_M on an arbitrary vector $\underset{\sim}{w}$ can be described by

$$[P_M \underset{\sim}{w}]_j = \begin{cases} w_j & \text{if } y_j > 0 \\ 0 & \text{otherwise} \end{cases} .$$

Alternatively,

$$[P_M \underset{\sim}{w}]_j = \begin{cases} w_j & \text{if } j \in \bar{N}_{i_t} \\ 0 & \text{otherwise} \end{cases}$$

where i_t is the number of iterations required to complete the application of Algorithm 4.1.

The main feature of using the operator Q_M for transportation type problems is that the matrix E, which is $n \times n$, does not have to be stored but can be fully described as a sequence of operators. The only storage needed is that for several lists. Thus there is no real limitation on the size of the program. For example, problems with both $\bar{j}$ and $\bar{k}$ of the order of 1000, could be tackled whereas such problems would otherwise be intractable.

Although the projection operator Q_M can be used in the x-space, there is no equivalent expression in the x-space for the Lagrange multiplier estimates. This is easily remedied by using the fact that in the y-space, the multiplier estimate corresponding to the active constraint $y_j = 0$ is just

$$[\nabla_{\underset{\sim}{y}} f(\underset{\sim}{y})]_j \quad .$$

This has an added advantage in that the multiplier estimates are independent, which in turn means that as many constraints can be simply dropped as required, without the usual computational expense involved when the estimates are interdependent.

Methods using column transformations have been described by Buckley [10] and are also related to the approach of Murtagh and Saunders [31] and the reduced-gradient method. The matrix Z used in these latter two methods (see Sections 2.5

and 2.6) can be easily constructed if M and E are known. This can be done as follows:

Let $M = [\underset{\sim}{e}_{\ell_1} \ \ldots \ \underset{\sim}{e}_{\ell_m}]^T$ and $\bar{M} = [\underset{\sim}{e}_{\ell_{m+1}} \ \ldots \ \underset{\sim}{e}_{\ell_n}]^T$ where $\ell_1, \ldots, \ell_m$ are distinct, $\ell_{m+1}, \ldots, \ell_n$ are distinct and $\{\ell_1, \ldots, \ell_n\} = \{1, \ldots, n\}$. Then if $Z = E\bar{M}^T$, then

$$\begin{aligned} NZ &= NE\bar{M}^T \\ &= M\bar{M}^T = 0 \quad . \end{aligned}$$

It can also be noted that for the reduced gradient operator ZZ^T,

$$\begin{aligned} ZZ^T &= E\bar{M}^T\bar{M}E^T \\ &= EP_M E^T \quad . \end{aligned}$$

Thus the operator Q_M is essentially the same as the reduced-gradient operator. In fact the reduced-gradient method could be interpreted as a "transformed steepest descent" gradient projection method. As for the Generalized Upper Bounding problem, if $s \leqslant 200$ (say), the revised Goldfarb's method or the Murtagh-Saunders method could be used.

CHAPTER 5

THE AUGMENTED TRANSPORTATION TYPE PROBLEM

Problems of the type discussed in Chapter 4 can also have additional linear inequality constraints. For such problems, Lemma 3.1 can be used to construct a version of the gradient projection method paralleling that of Chapter 3.

5.1 THE AUGMENTED PROBLEM

This class of problems can be stated as:

$$\min \quad f(\underset{\sim}{x}) \tag{5.1}$$

$$\sum_{j \in A_k} x_j^k = b^k, \quad k = 1,\ldots,\bar{k} \tag{5.2}$$

$$\sum_{k \in B_j} x_j^k \geq d_j, \quad j = 1,\ldots,\bar{j} \tag{5.3}$$

$$\sum_{k=1}^{\bar{k}} \sum_{j=1}^{\bar{j}} \gamma_{ij}^k x_j^k \geq c_i, \quad i = 1,\ldots,\bar{\ell} \tag{5.4}$$

$$x_j^k \geq 0, \quad j = 1,\ldots,\phi(k); \; k = 1,\ldots,\bar{k} \; . \tag{5.5}$$

Problems of this nature will probably not have the structure necessary to use an algorithm such as algorithm 4.1. However, this can be partly offset by using the active constraints of (5.2), (5.3) and (5.5), represented by the matrix N_1, and algorithm 4.1 to produce a matrix E.

From Lemma 3.1, the projection matrix in y-space is

$$\begin{aligned} P_M &= P_{M_1} - P_{M_1}(N_2 E)^T [(N_2 E) P_{M_1} (N_2 E)^T]^{-1} (N_2 E) P_{M_1} \\ &= P_{M_1} - P_{M_1} E^T N_2^T [N_2 Q_{M_1} N_2^T]^{-1} N_2 E P_{M_1} \end{aligned} \tag{5.6}$$

where $M_1 = N_1E$, $M = \begin{bmatrix} N_1 \\ N_2 \end{bmatrix} E$ and the rows of N_2 contain the normals of the active constraints of (5.4). Thus the operator Q_M can be written as

$$\begin{aligned} Q_M &= EP_M E^T \\ &= Q_{M_1} - Q_{M_1} N_2^T [N_2 Q_{M_1} N_2^T]^{-1} N_2 Q_{M_1} . \end{aligned} \tag{5.7}$$

This means that an expression for the search direction can be obtained by using (3.5) with P_N and P_{N_1} replaced by Q_M and Q_{M_1} respectively. Again the Lagrange multiplier estimates must be inspected in the y-space.

For each N_1, there will be different matrices E and M_1. This will necessitate the complete recalculation of Q_M after each change in N_1 unless a relationship can be established for successive transformations E and matrices M_1. This is quite straightforward when the matrix E is stored but raises several complications when E is represented implicitly as in Chapter 4.

A partial answer to the question of this relationship can be given by the following Lemma.

Lemma 5.1 If by some means it is known that there exists a nonsingular matrix E such that for N_1 $(m \times n)$,

$$M_1 = N_1E = [\underset{\sim}{e}_{\ell_1} \ \cdots \ \underset{\sim}{e}_{\ell_m}]^T$$

and

$$\underset{\sim}{e}_j^T E = \underset{\sim}{e}_j^T \quad \text{for all} \quad j \in \bar{N} = \{1,\ldots,n\} \backslash N$$

where

$$N = \{\ell_1, \ldots, \ell_m\} ,$$

then the application of Algorithm 4.1 with this N will produce a matrix E^* such that

$$N_1 E^* = [e_{u_1} \ \ldots \ e_{u_m}]^T$$

where

$$\{u_1, \ldots, u_m\} = N$$

and

$$e_j^T E^* = e_j^T, \ \forall j \in \bar{N} .$$

Proof Let $\hat{N}_1$ be the matrix whose rows contain the distinct vectors e_j^T, $j \in \bar{N}$.

Then $\begin{bmatrix} N_1 \\ \hat{N}_1 \end{bmatrix}$ is a nonsingular matrix.

The matrix $\begin{bmatrix} N_1 \\ \hat{N}_1 \end{bmatrix}$ can also be written (after row and column permutations in the form)

$$\left[\begin{array}{ccc} N_{11} & N_{12} & N_{13} \\ 0 & I & 0 \\ \hline 0 & 0 & I \end{array}\right] \begin{array}{l} \} \text{rows } j \text{ such that } j \in N \\ \} \text{rows } j \text{ such that } j \in \bar{N} \end{array}$$

This matrix is of the required form for the application of Algorithm 4.1. As Algorithm 4.1 guarantees that E can be calculated for a matrix N having full row rank the result follows.

5.2 UPDATES

As in Section 3.3, it was decided to use a Cholesky factorization of a symmetric positive definite matrix. Instead of factorizing $N_2 P_{N_1} N_2^T$, the Cholesky decomposition of $N_2 Q_{M_1} N_2^T$ is now maintained. It can be readily noticed that if a row

is added to or deleted from N_2 then the updating of LDL^T is exactly the same as in Sections 3.3.3 and 3.3.4 except that P_{N_1} is replaced by Q_{M_1}.

5.2.1 Adding a new row to N_1

Consider the case when a new constraint becomes active and has the effect of augmenting N_1 to $\begin{bmatrix} N_1 \\ \underset{\sim}{n}^T \end{bmatrix}$. Then

$$\begin{bmatrix} N_1 \\ \underset{\sim}{n}^T \end{bmatrix} E = \begin{bmatrix} M_1 \\ \underset{\sim}{n}^T E \end{bmatrix}$$

and

$$\underset{\sim}{e}_j^T E = \underset{\sim}{e}_j^T \quad \text{for} \quad j \in \bar{N} = \{1,\ldots,n\} \setminus N$$

where

$$N = \{j_1,\ldots,j_m\} \quad \text{and} \quad M_1 = [\underset{\sim}{e}_{j_1} \ \ldots \ \underset{\sim}{e}_{j_m}]^T .$$

Such a matrix E can be obtained by using Algorithm 4.1, as this algorithm only uses the pivot columns whose indices are in N.

If the row vector $\underset{\sim}{n}^T E$ is a linear combination of $\underset{\sim}{e}_{j_1}^T,\ldots,\underset{\sim}{e}_{j_m}^T$ (which can be easily checked) then the new constraint is linearly dependent on the transformed constraints and hence on the untransformed constraints. In this case the constraint does not have to be included in the active set. If this is not the case, then there exists some $j \in \bar{N}$ such that the jth element of $\underset{\sim}{n}^T E$ is nonzero, i.e. $\underset{\sim}{n}^T E \underset{\sim}{e}_j \neq 0$.

Without loss of generality, let such an element be j_{m+1}. Then the matrix $\begin{bmatrix} M_1 \\ \underset{\sim}{n}^T E \end{bmatrix}$ together with the row vectors associated

with the index set $\bar{N}\setminus\{j_{m+1}\}$ will be a linearly independent system of rank n. Hence it will also be true of the matrix $\begin{bmatrix} N \\ \underset{\sim}{n}^T \end{bmatrix}$ together with the row vectors associated with $\bar{N}\setminus\{j_{m+1}\}$.

Let F be the matrix

$$F = \begin{bmatrix} I_{j_{m+1}-1} & & \\ & \underset{\sim}{n}^T E & \\ & & I_{n-j_{m+1}} \end{bmatrix}^{-1}$$

$$= I + \tau \underset{\sim}{e}_{j_{m+1}} (\underset{\sim}{e}^T_{j_{m+1}} - \underset{\sim}{n}^T E)$$

where $\tau = 1/\underset{\sim}{n}^T E \underset{\sim}{e}_{j_{m+1}}$.

It then follows that

$$\begin{bmatrix} N_1 \\ \underset{\sim}{n}^T \end{bmatrix} EF = \begin{bmatrix} M_1 \\ \underset{\sim}{n}^T E \end{bmatrix} F = M_1^*$$

and

$$\underset{\sim}{e}_j^T EF = \underset{\sim}{e}_j^T \quad \text{for} \quad j \in \bar{N}\setminus\{j_{m+1}\}$$

where

$$M_1^* = [\underset{\sim}{e}_{j_1} \cdots \underset{\sim}{e}_{j_{m+1}}]^T .$$

Let

$$N^* = N \cup \{j_{m+1}\} = \{j_1, \ldots, j_{m+1}\} ,$$

$$\bar{N}^* = \bar{N} \setminus \{j_{m+1}\}$$

and

$$N_1^* = \begin{bmatrix} N_1 \\ \underset{\sim}{n}^T \end{bmatrix} .$$

Lemma 5.2 If E^* is the matrix constructed by Algorithm 4.1, given N^*, then $E^* = EFS$ where S is a permutation matrix.

Proof It is known that EF is a nonsingular matrix such that

$$M_1^* = N_1^* EF = [\underset{\sim}{e}_{j_1} \ \ldots \ \underset{\sim}{e}_{j_{m+1}}]^T$$

and

$$\underset{\sim}{e}_j^T EF = \underset{\sim}{e}_j^T \quad \text{for all} \quad j \in \bar{N}^* \quad .$$

By Lemma 5.1, Algorithm 4.1, with N^*, will produce a matrix E^* such that

$$N_1^* E^* = [\underset{\sim}{e}_{u_1} \ \ldots \ \underset{\sim}{e}_{u_{m+1}}]^T$$

where

$$\{u_1, \ldots, u_{m+1}\} = N^*$$

and

$$\underset{\sim}{e}_j^T E^* = \underset{\sim}{e}_j^T \ , \ \text{for all} \quad j \in \bar{N}^* \quad .$$

Now the system formed by N_1^* and the row vectors associated with $\bar{N}^*$ will be linearly independent and of rank n.

This means that the transformation matrix is unique. Hence $E^* = EFS$, where S is a permutation matrix.

Lemma 5.3 Let P_{M_1} be the projection matrix determined by N and $P_{\bar{M}_1}$ be the projection matrix determined by N^*. Then

$$Q_{\bar{M}_1} = E^* P_{\bar{M}_1} E^{*T} = Q_{M_1} + \sigma_1 \underset{\sim}{w}_1 \underset{\sim}{w}_1^T + \sigma_2 \underset{\sim}{w}_2 \underset{\sim}{w}_2^T$$

where

$$\sigma_1 = 1/\underset{\sim}{n}^T Q_{M_1} \underset{\sim}{n}$$

$$\sigma_2 = -\, 1/\underset{\sim}{n}^T Q_{M_1} \underset{\sim}{n}$$

$$\underset{\sim}{w}_1 = \frac{\underset{\sim}{n}^T Q_{M_1} \underset{\sim}{n}}{\underset{\sim}{n}^T E \underset{\sim}{e}_{j_{m+1}}} E \underset{\sim}{e}_{j_{m+1}} - Q_{M_1} \underset{\sim}{n}$$

and $$\underset{\sim}{w}_2 = Q_{M_1} \underset{\sim}{n} \quad .$$

Proof By Lemma 5.2,

$$\begin{aligned} Q_{\bar{M}_1} &= E^* P_{\bar{M}_1} E^{*T} \\ &= EFSP_{\bar{M}_1} S^T F^T E^T \\ &= EFP_{\bar{M}_1} F^T E^T \quad . \end{aligned}$$

As $j_{m+1} \in N^*$

$$P_{\bar{M}_1} \underset{\sim}{e}_{j_{m+1}} = \underset{\sim}{0} \quad .$$

Thus,

$$P_{\bar{M}_1} F^T = P_{\bar{M}_1} [I - \tau E^T \underset{\sim}{n} \underset{\sim}{e}^T_{j_{m+1}}] .$$

It can also be seen that

$$P_{\bar{M}_1} = P_{M_1} - \underset{\sim}{e}_{j_{m+1}} \underset{\sim}{e}^T_{j_{m+1}}$$

and

$$\underset{\sim}{e}^T_{j_{m+1}} [I - \tau E^T \underset{\sim}{n} \underset{\sim}{e}^T_{j_{m+1}}] = 0.$$

Hence

$$P_{\bar{M}_1} F^T = P_{M_1} [I - \tau E^T \underset{\sim}{n} \underset{\sim}{e}^T_{j_{m+1}}] \quad .$$

Therefore,

$$FP_{\bar{M}_1}F^T = P_{M_1} - \tau\, e_{j_{m+1}} n^T E P_{M_1} - \tau P_{M_1} E^T n\, e_{j_{m+1}}^T + \tau^2 (n^T Q_{M_1} n) e_{j_{m+1}} e_{j_{m+1}}^T .$$

By a process similar to completing the square,

$$FP_{\bar{M}_1}F^T - P_{M_1} = (1/n^T Q_{M_1} n)\hat{w}_1\hat{w}_1^T - (1/n^T Q_{M_1} n)\hat{w}_2\hat{w}_2^T$$

where

$$\hat{w}_1 = (n^T Q_{M_1} n).\tau.e_{j_{m+1}} - P_{M_1} E^T n$$

and

$$\hat{w}_2 = P_{M_1} E^T n .$$

Hence the result follows.

It is interesting to note that since

$$Ee_{j_{m+1}} = EP_{M_1} e_{j_{m+1}} = Q_{M_1} e_{j_{m+1}}$$

w_1 can also be written as

$$w_1 = Q_{M_1}[(n^T Q_{M_1} n / n^T Q_{M_1} e_{j_{m+1}})e_{j_{m+1}} - n] .$$

It can also be noted that the adding of a new row to N_1 could have been accomplished in a different manner. Firstly the projection operator

$$\hat{Q}_{M_1} = E^* P_{M_1} E^{*T}$$

could be calculated. Then the new row added to the matrix M_1 (i.e. $e_{j_{m+1}}^T$) could be used with Lemma 3.1 to calculate the new operator

$$\hat{Q}_{M_1} - \hat{Q}_{M_1} e_{j_{m+1}} [e_{j_{m+1}}^T \hat{Q}_{M_1} e_{j_{m+1}}]^{-1} e_{j_{m+1}}^T \hat{Q}_{M_1} .$$

Lemma 5.4

$$Q_{\bar{M}_1} = \hat{Q}_{M_1} - \hat{Q}_{M_1} e_{j_{m+1}} [e^T_{j_{m+1}} \hat{Q}_{M_1} e_{j_{m+1}}]^{-1} e^T_{j_{m+1}} \hat{Q}_{M_1} . \quad (5.8)$$

Proof

$$\begin{aligned} \hat{Q}_{M_1} &= E^* P_{M_1} E^{*T} \\ &= EFSP_{M_1} S^T F^T E^T \\ &= EFP_{M_1} F^T E^T . \end{aligned}$$

It can be noticed that since $j_{m+1} \in \bar{N}$,

$$e^T_{j_{m+1}} E = e^T_{j_{m+1}} .$$

It can also be seen that

$$F^T e_{j_{m+1}} = \tau . e_{j_{m+1}} .$$

Thus

$$\hat{Q}_{M_1} e_{j_{m+1}} = \tau . e_{j_{m+1}}$$

and hence

$$[e^T_{j_{m+1}} \hat{Q}_{M_1} e_{j_{m+1}}]^{-1} = 1/\tau^2 .$$

Hence the right hand side of (5.8) is

$$\begin{aligned} & E^* [P_{M_1} - e_{j_{m+1}} e^T_{j_{m+1}}] E^{*T} \\ = \; & E^* P_{\bar{M}_1} E^{*T} . \end{aligned}$$

However, the use of an approach whereby $\hat{Q}_{M_1}$ is calculated and then $Q_{\bar{M}_1}$ is not as efficient as the method given as it would be more computationally expensive.

If the Cholesky decomposition LDL^T of $N_2 Q_{M_1} N_2^T$ is available, then

$$\bar{L}\bar{D}\bar{L}^T = N_2 Q_{\bar{M}_1} N_2^T$$

$$= LDL^T + \sigma_1 \underset{\sim}{z}_1 \underset{\sim}{z}_1^T + \sigma_2 \underset{\sim}{z}_2 \underset{\sim}{z}_2^T \quad ,$$

where $\underset{\sim}{z}_1 = N_2 \underset{\sim}{w}_1$ and $\underset{\sim}{z}_2 = N_2 \underset{\sim}{w}_2$.

The Cholesky factors can then be updated in two stages by methods such as those described in Section 3.4. Some caution, however, is needed in doing this as $\sigma_1 > 0$ and $\sigma_2 < 0$. To ensure the positive definiteness of the factorization, it is necessary to calculate the Cholesky factors for $LDL^T + \sigma_1 \underset{\sim}{z}_1 \underset{\sim}{z}_1^T$ first. This is because the LDL^T factors of an indefinite matrix may not exist at all, and if they do it is possible for the elements of the lower triangular matrix L to be large enough to cause significant rounding error during the modification process.

It can also be noted that in the event that $\underset{\sim}{n} = \underset{\sim}{e}_{j_{m+1}}$, then

$$\bar{L}\bar{D}\bar{L}^T = LDL^T - \underset{\sim\sim}{z}\underset{\sim\sim}{z}^T \quad ,$$

where

$$\underset{\sim}{z} = N_2 E \underset{\sim}{e}_{j_{m+1}} \quad .$$

5.2.2 Deleting a row from N_1

Suppose the row to be deleted is the j_m'th row of N_1 and let it be denoted by $\underset{\sim}{n}^T$.

Now $\underset{\sim}{n}^T = \underset{\sim}{e}_{\ell_1}^T + \ldots + \underset{\sim}{e}_{\ell_r}^T$ for some $\ell_1, \ldots, \ell_r$. By the nature of construction of algorithm 4.1, one of the columns $\ell_1, \ldots, \ell_r$ of N_1 was used as a pivot in the transformation

producing $\underset{\sim}{e}_{j_m}^T$. Hence if M_1 and E had been produced by algorithm 4.1, then

$$j_m \in \{\ell_1, \ldots, \ell_r\} .$$

Hence $\underset{\sim}{n}^T \underset{\sim}{e}_{j_m} = 1$.

In retracing the steps of algorithm 4.1, at some stage $\underset{\sim}{n}^T$ is transformed to $\underset{\sim}{e}_{j_m}^T$ and j_m enters the index set N. The elementary matrix doing this is

$$\bar{E} = I + \frac{1}{\underset{\sim}{n}^T \underset{\sim}{e}_{j_m}} \underset{\sim}{e}_{j_m} (\underset{\sim}{e}_{j_m}^T - \underset{\sim}{n}^T)$$

$$= I + \underset{\sim}{e}_{j_m} (\underset{\sim}{e}_{j_m}^T - \underset{\sim}{n}^T) .$$

Thus it can be seen that

$$\underset{\sim}{e}_{j_m}^T \bar{E} = \underset{\sim}{e}_{j_m}^T + (\underset{\sim}{e}_{j_m}^T - \underset{\sim}{n}^T)$$

so that

$$\underset{\sim}{e}_{j_m}^T \bar{E} \underset{\sim}{e}_{j_m} = 1.$$

At this stage j_m enters N and hence no further operations are performed on column j_m for the remainder of the algorithm, i.e. the j_m'th element of the row vector $\underset{\sim}{e}_{j_m}^T \bar{E}$ would remain the same for the remainder of the algorithm. Hence

$$\underset{\sim}{e}_{j_m}^T E \underset{\sim}{e}_{j_m} = 1.$$

Now N_1^*, which is N_1 contracted after the row $\underset{\sim}{n}^T$ is removed, is such that

$$N_1^* E = [\underset{\sim}{e}_{j_1} \ \ldots \ \underset{\sim}{e}_{j_{m-1}}]^T .$$

Furthermore, $\underset{\sim}{e}_j^T E = \underset{\sim}{e}_j^T$ for $j \in \bar{N}$,

where $N = \{j_1, \ldots, j_m\}$ and as previously noted $(e_{\sim j_m}^T E) e_{\sim j_m} = 1$. Hence the system N_1^*, the rows formed by $\bar{N}$ and the row vector $e_{\sim j_m}^T$ is linearly independent.

Let

$$F = I + \frac{1}{e_{\sim j_m}^T E e_{\sim j_m}} e_{\sim j_m} (e_{\sim j_m}^T - e_{\sim j_m}^T E)$$

$$= I + e_{\sim j_m} (e_{\sim j_m}^T - e_{\sim j_m}^T E).$$

Then

$$N_1^* EF = [e_{\sim j_1} \ \ldots \ e_{\sim j_{m-1}}]^T$$

and

$$e_{\sim j}^T EF = e_{\sim j}^T \quad \text{for} \quad j \in \bar{N}^* = \bar{N} \cup \{j_m\}.$$

Let E^* be the matrix constructed by Algorithm 4.1 with $N^* = N \setminus \{j_m\}$. Then analogously to Lemma 5.2, $E^* = EFS$ for some permutation matrix S.

<u>Lemma 5.5</u> $Q_{\bar{M}_1} = E^* P_{\bar{M}_1} E^{*T} = Q_{M_1} + \sigma_1 w_1 w_1^T + \sigma_2 w_2 w_2^T$,

where

$$\sigma_1 = 1/\gamma, \ \sigma_2 = -1/\gamma$$

$$w_1 = \gamma E e_{\sim j_m} - Q_{M_1} e_{\sim j_m}$$

$$w_2 = Q_{M_1} e_{\sim j_m}$$

and

$$\gamma = 1 + e_{\sim j_m}^T Q_{M_1} e_{\sim j_m} = e_{\sim j_m}^T E P_{\bar{M}_1} E^T e_{\sim j_m}.$$

<u>Proof</u> As $e_{\sim j_m}^T E e_{\sim j_m} = 1$ and $P_{\bar{M}_1} = P_{M_1} + e_{\sim j_m} e_{\sim j_m}^T$

it follows that

$$\gamma = 1 + e_{\sim j_m}^T Q_{M_1} e_{\sim j_m}$$

$$= e_{\sim j_m}^T E P_{\bar{M}_1} E^T e_{\sim j_m}.$$

Now,

$$E^*P_{\bar{M}_1} E^{*T} = EFSP_{\bar{M}_1} S^T F^T E^T$$
$$= EFP_{\bar{M}_1} F^T E^T \quad .$$

As $j_m \in N$ it follows that $P_{M_1} \underset{\sim}{e}_{j_m} = \underset{\sim}{0}$.

Hence it can be seen that

$$P_{M_1} F^T = P_{M_1} - P_{M_1} E^T \underset{\sim}{e}_{j_m} \underset{\sim}{e}_{j_m}^T \quad .$$

It can also be seen that

$$\underset{\sim}{e}_{j_m} \underset{\sim}{e}_{j_m}^T F^T = \underset{\sim}{e}_{j_m} \underset{\sim}{e}_{j_m}^T \quad .$$

Hence

$$P_{\bar{M}_1} F^T = \underset{\sim}{e}_{j_m} \underset{\sim}{e}_{j_m}^T + P_{M_1} - P_{M_1} E^T \underset{\sim}{e}_{j_m} \underset{\sim}{e}_{j_m}^T \quad .$$

Thus,

$$FP_{\bar{M}_1} F^T = P_{M_1} - P_{M_1} E^T \underset{\sim}{e}_{j_m} \underset{\sim}{e}_{j_m}^T - \underset{\sim}{e}_{j_m} \underset{\sim}{e}_{j_m}^T EP_{M_1} + \gamma \underset{\sim}{e}_{j_m} \underset{\sim}{e}_{j_m}$$

$$= P_{M_1} + (1/\gamma)[\gamma \underset{\sim}{e}_{j_m} - P_{M_1} E^T \underset{\sim}{e}_{j_m}][\gamma \underset{\sim}{e}_{j_m} - P_{M_1} E^T \underset{\sim}{e}_{j_m}]^T$$

$$- (1/\gamma)[P_{M_1} E^T \underset{\sim}{e}_{j_m}][P_{M_1} E^T \underset{\sim}{e}_{j_m}]^T$$

by completing the square.

As before the Cholesky decomposition of $N_2 E^* P_{\bar{M}_1} E^{*T} N_2^T$ can be calculated from

$$\bar{L}\bar{D}\bar{L}^T = LDL^T + \sigma_1 \underset{\sim}{z}_1 \underset{\sim}{z}_1^T + \sigma_2 \underset{\sim}{z}_2 \underset{\sim}{z}_2^T$$

where $\underset{\sim}{z}_1 = N_2 \underset{\sim}{w}_1$ and $\underset{\sim}{z}_2 = N_2 \underset{\sim}{w}_2$.

A simplification can be noted when $\underset{\sim}{n} = \underset{\sim}{e}_{j_m}$, in which case

$$\bar{L}\bar{D}\bar{L}^T = LDL^T + \underset{\sim}{z}\underset{\sim}{z}^T$$

where

$$\underset{\sim}{z} = N_2 E \underset{\sim}{e}_{j_m} \quad .$$

CHAPTER 6

PERFORMANCE OF THE GRADIENT PROJECTION METHOD

The gradient projection method of Rosen is essentially no more than just the method of *steepest descent* in various subspaces. Thus an analysis of the prospective performance of the gradient projection method can be made by examining the weaknesses and possible remedies for these weaknesses with the steepest descent method.

6.1 THE METHOD OF STEEPEST DESCENT

The concept of steepest descent was first introduced by Cauchy in 1847 as a means for obtaining the simultaneous solution of equations. The main idea supporting steepest descent is that a minimum is sought by always moving in the direction in E^n yielding the greatest rate of decrease of $f(\underset{\sim}{x})$. It can be easily verified that the direction is just $-\underset{\sim}{g}(\underset{\sim}{x})$.

Unfortunately this very simple method has several serious shortcomings. These essentially result in slow convergence due to the deterioration of the method in a global sense, even though the individual iterations are locally optimal. Another feature of the method is that very short steps may be produced, accompanied by sharp changes in the gradient $\underset{\sim}{g}(\underset{\sim}{x})$. This "zigzagging" effect usually results in slow convergence and low computational efficiency despite the small amount of

computation per iteration. Furthermore, the steepest descent method depends on the scaling of the variables. For example, the program

$$\begin{aligned} \min \quad & 100x_1^2 + x_2^2 \\ \text{s.t.} \quad & x_1 \geqslant 0, x_2 \geqslant 0 \end{aligned}$$

is significantly harder to solve than the program

$$\begin{aligned} \min \quad & x_1^2 + x_2^2 \\ \text{s.t.} \quad & x_1 \geqslant 0, x_2 \geqslant 0 \end{aligned}$$

which is obtained by a change of variables. This is because the contours of the first objective function form a deep and narrow valley along the x_2 coordinate axis, while the contours of the second objective function are concentric circles.

It has also been shown by Kowalik and Osborne [28, Chapter 3] that the method is basically unstable with respect to small perturbations. Such perturbations may result from rounding errors or inaccuracy in determining optimal step lengths. It has been seen that in the extreme case these perturbations may cause the descent directions to be orthogonal to the directions calculated if no errors had been involved.

The rate of convergence of the steepest descent method when applied to a quadratic form $\hat{f}(\underset{\sim}{x})$ can be estimated [26,p.117] by

$$\hat{f}(\underset{\sim}{x}_{i+1}) \leqslant \hat{f}(\underset{\sim}{x}_i)\left(\frac{\mu_{max} - \mu_{min}}{\mu_{max} + \mu_{min}}\right)^2 \qquad (6.1)$$

where μ_{max} and μ_{min} are the largest and smallest eigenvalues of the quadratic form and both μ_{max} and μ_{min} are

positive. Akaike [1] has demonstrated that approximate equality holds in (6.1). The resulting implication is that the steepest descent method can have extremely poor convergence, particularly when $\mu_{max} >> \mu_{min}$. Such a situation could occur if the matrix in the quadratic form $\hat{f}(\underset{\sim}{x})$ is ill-conditioned, which is again related to scaling of variables. This has the general implication that the method of steepest descent is highly scale dependent. Expression (6.1) can also be used to estimate the final rate of convergence of the method when $\hat{f}$ is a general nonlinear function. Thus similar problems can be expected with the terminal convergence of a constrained nonlinear function when the gradient projection method is used.

6.2 A STEEPEST DESCENT TYPE GRADIENT PROJECTION METHOD

The direction of search used in the gradient projection method was stated in (2.8) to be the solution of the program

$$\begin{aligned} \min \quad & \underset{\sim}{p}^T \underset{\sim}{g} \\ \text{s.t.} \quad & N \underset{\sim}{p} = \underset{\sim}{0} \\ & \|\underset{\sim}{p}\|_2^2 = 1 . \end{aligned}$$

Zwart in 1970 [44] suggested using the *steepest descent* direction for a linearly constrained problem, which is the solution to the program

$$\begin{aligned} \min \quad & \underset{\sim}{p}^T \underset{\sim}{g} \\ \text{s.t.} \quad & N \underset{\sim}{p} \geqslant \underset{\sim}{0} \\ & \|\underset{\sim}{p}\|_2^2 = 1. \end{aligned}$$

This is the direction minimizing the rate of change of f (that is, $\underset{\sim}{p}^T\underset{\sim}{g}$) while not violating the currently active constraints.

In a theorem, Zwart characterized this steepest descent direction as the direction $\underset{\sim}{p} = - P_{\bar{N}}\, \underset{\sim}{g}$, where $\bar{N}$ contains some subset of the rows of N, and the following two conditions are satisfied:

(i) $\underset{\sim}{p}^T\underset{\sim}{n} \geqslant 0$ for $\underset{\sim}{n}^T$ a row of N but not of $\bar{N}$

(ii) $\bar{N}^+\underset{\sim}{g} \geqslant \underset{\sim}{0}$.

It is not generally obvious which subset of rows of N gives this matrix $\bar{N}$. However an iterative procedure can be constructed to obtain the direction $\underset{\sim}{p}$.

(i) Let $\underset{\sim}{\lambda} = N^+\underset{\sim}{g}$ and $\underset{\sim}{p} = - P_N\, \underset{\sim}{g}$.

Choose $\lambda_\ell = \min_i \lambda_i$.

If $\lambda_\ell < 0$, delete the ℓth row from N and go to (i).

(ii) Otherwise, stop and use the search direction $\underset{\sim}{p}$.

It can be seen from Section 3.2.2 that if the ℓth constraint is deleted from N, then

$$P_{\bar{N}} = P_N + \frac{\underset{\sim}{w}\underset{\sim}{w}^T}{\underset{\sim}{w}^T\underset{\sim}{w}}$$

where $\underset{\sim}{w} = (N^+)^T\underset{\sim}{e}_\ell$. It then follows that

$$\begin{aligned}\underset{\sim}{g}^T\,\bar{\underset{\sim}{p}} &= \underset{\sim}{g}^T\underset{\sim}{p} - \lambda_\ell^2/\underset{\sim}{w}^T\underset{\sim}{w} \\ &< \underset{\sim}{g}^T\underset{\sim}{p} \quad .\end{aligned}$$

Hence, every constraint deleted from N in this way should produce a better search direction in terms of giving a better

search direction in terms of giving a better rate of change of f. However the rate of change of f is not always a good criterion for choosing a search direction. Figure 1 provides an illustration of a simple example of this. Another factor against dropping a constraint unnecessarily is the phenomenon of *zigzagging* or *jamming*. This occurs when a constraint leaves and re-enters the active set several times in a relatively small number of iterations. An illustration of this is given in Figure 2.

An attempt to take advantage of the ability to obtain a better search direction by dropping constraints whilst simultaneously protecting against the occurrence of zigzagging was made with the following "ρ" strategy:

(i) Let $\underset{\sim}{\lambda} = N^{+}\underset{\sim}{g}$, $\underset{\sim}{p} = -P_N \underset{\sim}{g}$.

Choose $\lambda_\ell = \min_i \lambda_i$ and let $\underset{\sim}{n}_\ell^T$ be the normal of the corresponding constraint.

If $\lambda_\ell < -\rho \|\underset{\sim}{p}\|_\infty / \|\underset{\sim}{n}_\ell\|_\infty$, delete the ℓth row of N and go to (i).

(ii) Otherwise, stop and use the search direction $\underset{\sim}{p}$.

The parameter ρ is a scalar and can be set to appropriate values.

6.3 ACCELERATION OF THE GRADIENT PROJECTION METHOD

Forsythe and Motzkin [16] conjectured in 1951 that the search directions generated in the steepest descent method are ultimately asymptotic to just two directions. This means that the method essentially operates in a two-dimensional

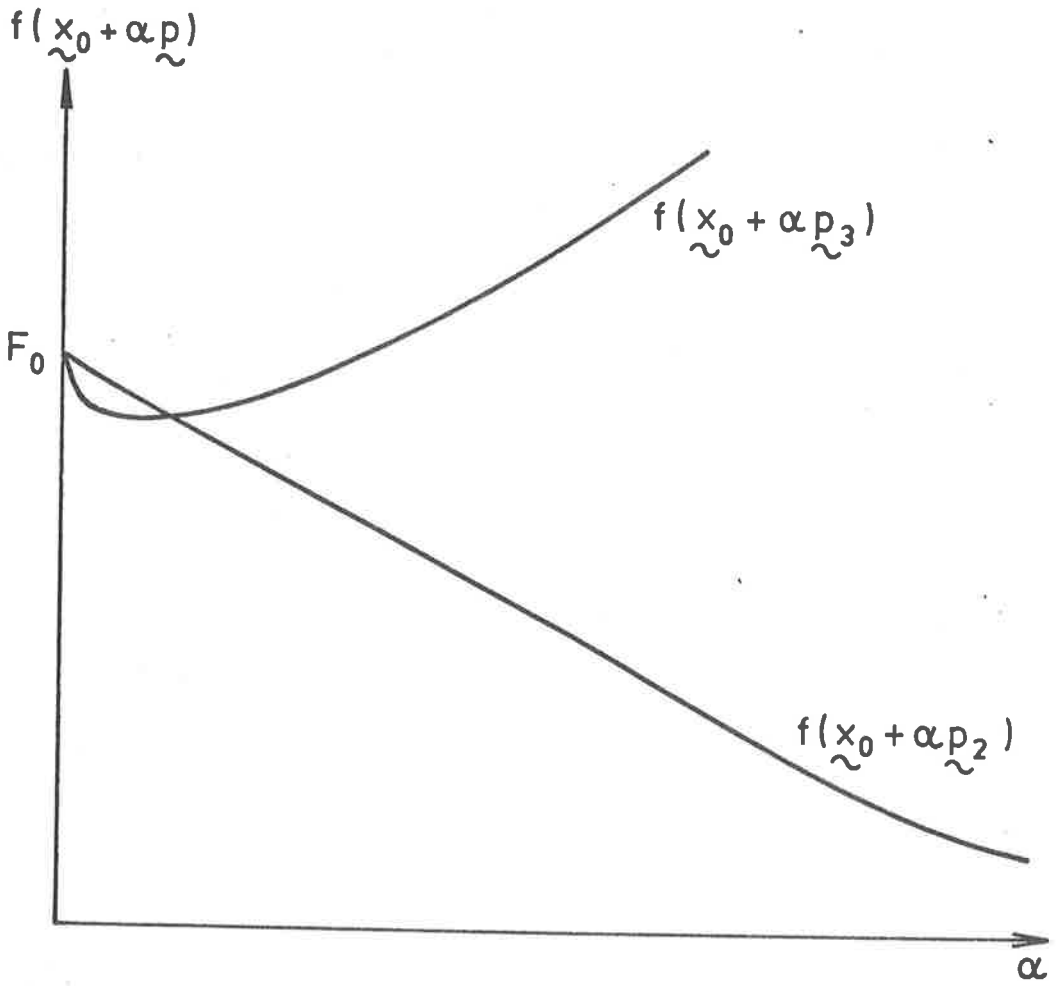


FIGURE 1 Graphs of $f(\underset{\sim}{x} + \alpha\underset{\sim}{p})$ for two different search directions $\underset{\sim}{p}_2$ and $\underset{\sim}{p}_3$.

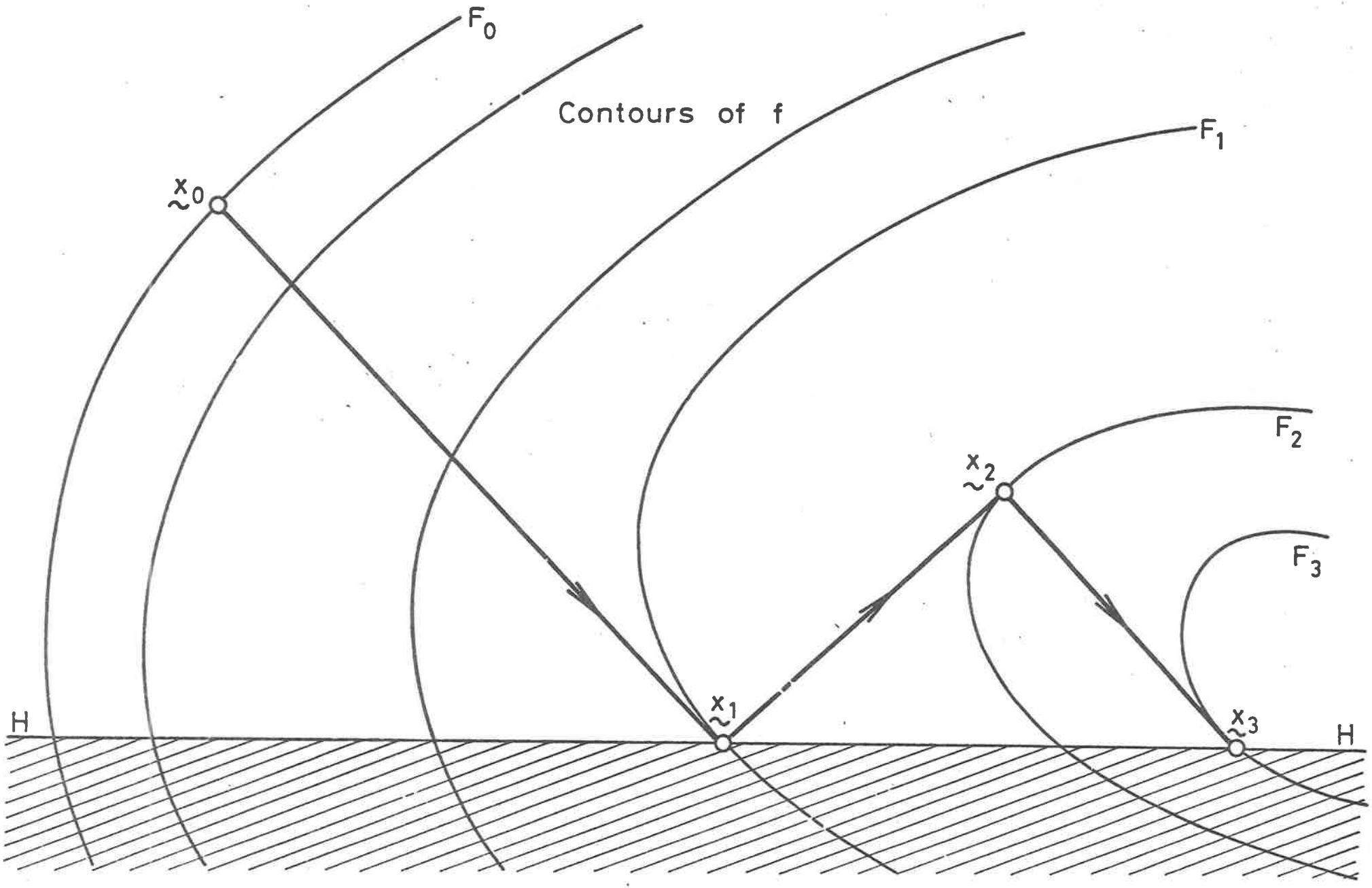


FIGURE 2 Zig zagging

subspace. This feature has been referred to as the "cage" by Stiefel [40] . The conjecture was later proved by Akaike [1].

Several schemes based on an idea of Forsythe and Motzkin have been proposed to accelerate the steepest descent method. They found that the solution of sets of linear equations by steepest descent could be accelerated if the direction

$$\underset{\sim}{p} = \underset{\sim}{x}_k - \underset{\sim}{x}_{k-2}, \quad k \geqslant 2 \qquad (6.2)$$

is used at certain times. The Forsythe-Motzkin conjecture stated that the steepest descent method converges to the minimum in such a way that the successive points $\underset{\sim}{x}_k$ eventually alternate between two lines. The directions given in (6.2) may coincide with one of these lines. Thus it seems reasonable to search along these directions. The use of the direction $\underset{\sim}{p}$ as in (6.2) will be here referred to as Partan-acceleration.

An example of the possible effectiveness of Partan-acceleration is given in Figures 3 and 4, which give two- and three-dimensional illustrations of the phenomenon of *hemstitching*. This corresponds to the way in which the steepest descent method usually descends a valley. Instead of moving straight down the valley or along the walls of the valley, the method tends to "bounce" from one wall of the valley to the other as it descends. The main point of the use of Partan-acceleration is that it enables straight and narrow valleys to be swiftly traversed. This is because points $\underset{\sim}{x}_4$ and $\underset{\sim}{x}_6$ (for example) define a line which parallels the valley. Hence, by searching along this parallel line

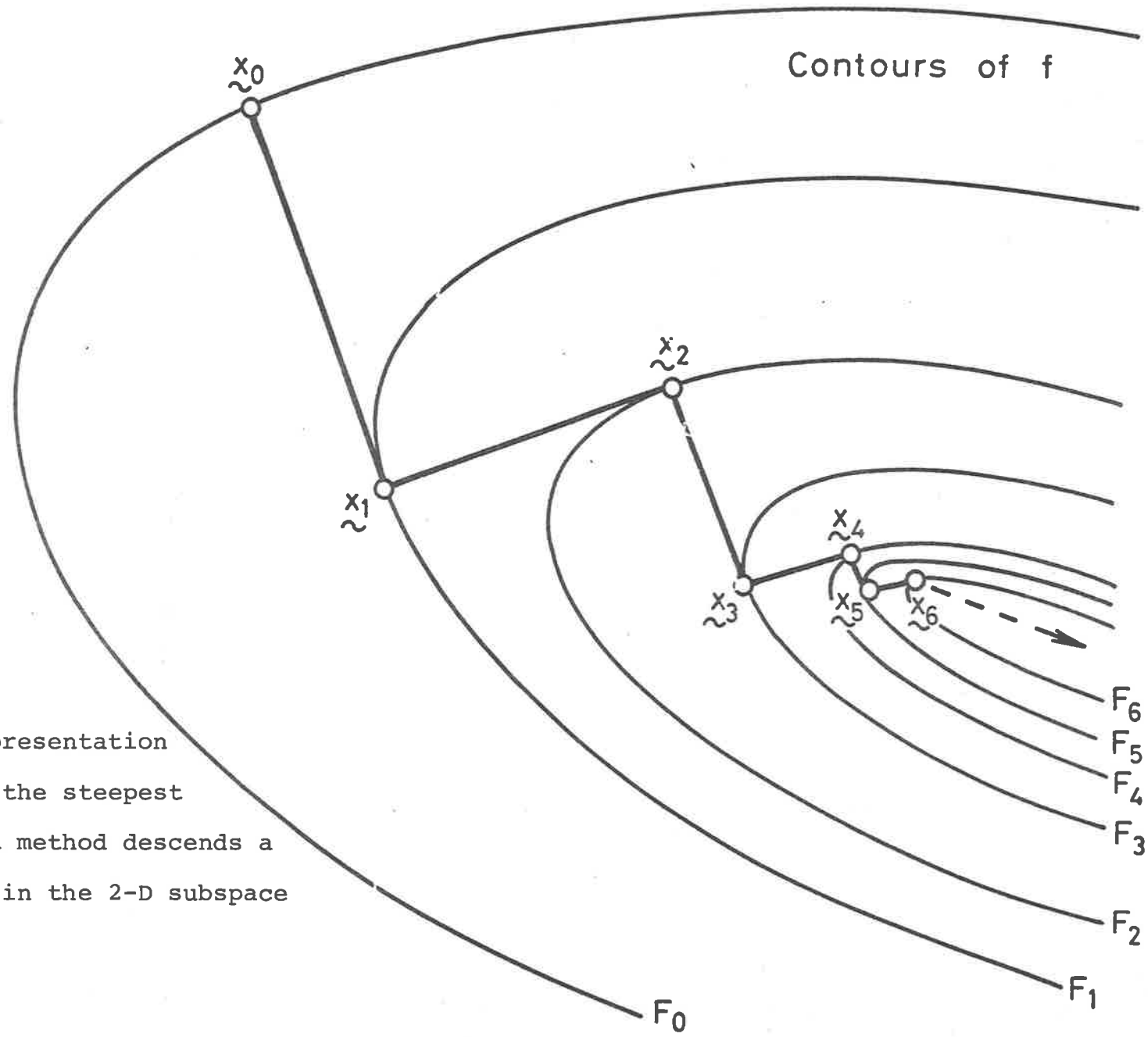


FIGURE 3 2-D representation of how the steepest descent method descends a valley in the 2-D subspace S.

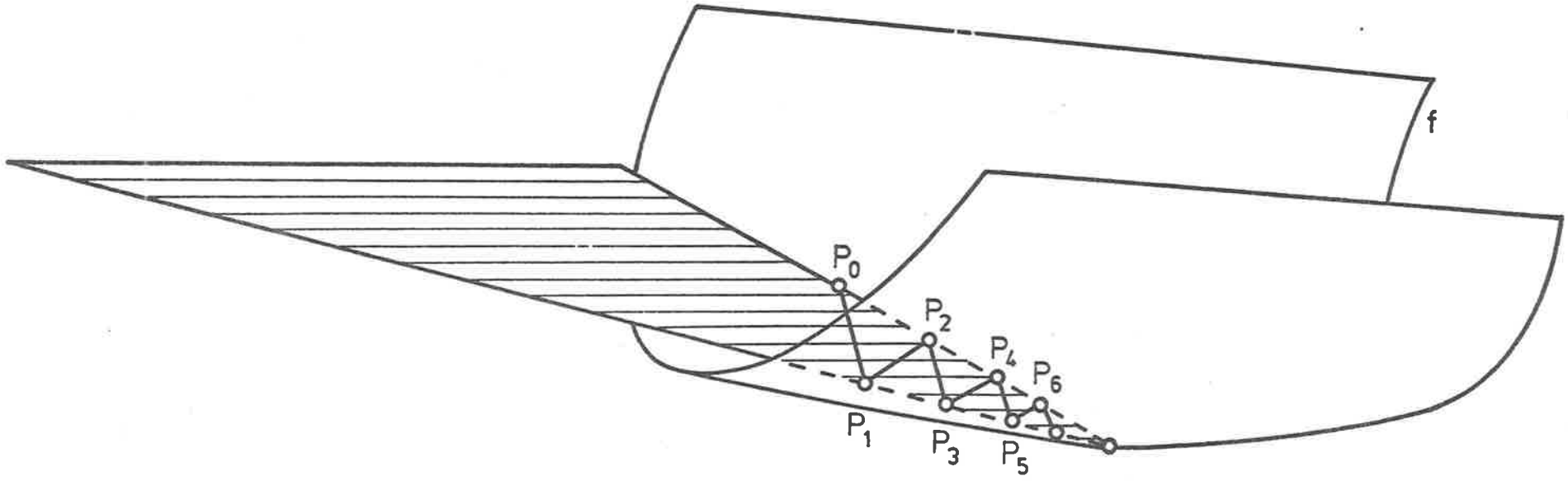


FIGURE 4 3-D representation of how the steepest descent method descends a valley in the 2-D subspace S.

the method effectively follows the valley. When the valleys are curved, Partan-acceleration is not as efficient but is certainly much better than just using steepest descent.

The name Partan (parallel tangents) is derived from a geometrical observation. For example, in Figure 3, the tangents at $\underset{\sim}{x}_2$ and $\underset{\sim}{x}_4$ are parallel and the line $\underset{\sim}{x}_2 - \underset{\sim}{x}_3$ is perpendicular to both of these tangents.

The method of continued-Partan is a generalization of the Forsythe-Motzkin idea proposed by Shah *et al* [37] in 1964. The method alternates between steepest descent directions and directions similar to those in (6.2). The method can be stated as follows:

$$\underset{\sim}{p}_0 = - \underset{\sim}{g}(\underset{\sim}{x}_0)$$

$$\underset{\sim}{p}_1 = - \underset{\sim}{g}_1$$

$$\underset{\sim}{p}_2 = - (\underset{\sim}{x}_2 - \underset{\sim}{x}_0)$$

$$\underset{\sim}{p}_k = \begin{cases} - \underset{\sim}{g}_k & k = 3,5,7,\ldots \\ - (\underset{\sim}{x}_k - \underset{\sim}{x}_{k-3}) & k = 4,6,8,\ldots \end{cases} .$$

Shah *et al* found that for most problems they considered, the performance of continued-Partan was better than other Partan methods they considered. They also proved that the optimum of $f(\underset{\sim}{x})$ is attained after at most $(2n-1)$ optimal line searches if $f(\underset{\sim}{x})$ is a quadratic with a well defined minimum.

Pierre [34] in 1969 showed that continued-Partan is a type of conjugate-gradient method. Sorenson [38] also showed that if a quadratic function is minimized then the vectors $\underset{\sim}{x}_1 - \underset{\sim}{x}_0, \underset{\sim}{x}_3 - \underset{\sim}{x}_1, \ldots, x_{2\ell-1} - x_{2\ell-3}$ are mutually conjugate,

but the vectors generated by continued-Partan are not, and the gradient vectors $\underset{\sim}{g}_0, \underset{\sim}{g}_1, \ldots, \underset{\sim}{g}_{2\ell-1}$ are orthogonal.

When the gradient projection method remains in the same subspace for several iterations, the problem is essentially just an unconstrained problem in a given subspace. It can therefore be expected that the convergence problems associated with the steepest descent method will also apply to the optimization in various subspaces. It can also be seen that the acceleration techniques of Partan-acceleration and continued-Partan can be used when the optimization procedure remains in a given subspace for several iterations. In this case $\underset{\sim}{g}(\underset{\sim}{x})$ is simply replaced by $P_N \, \underset{\sim}{g}(\underset{\sim}{x})$.

When $f(\underset{\sim}{x})$ is a quadratic function it was demonstrated by Sorenson that continued-Partan is considerably less efficient than the conjugate-gradient method. It was seen that continued-Partan goes through the same sequence of points as the conjugate-gradient method as well as through an equal number of other points. Thus the continued-Partan method requires twice as many iterations as are necessary for a quadratic function.

This observation suggests that a conjugate-gradient method could be used. The most suitable of the conjugate-gradient methods (in terms of computer storage) are methods such as that of Fletcher and Reeves [15]. This method was extended by Goldfarb [22] to be incorporated with the gradient projection method. Such methods can be described as follows:

(i) $\underset{\sim}{p}_0 = - \underset{\sim}{g}_0$.

(ii) For $k = 0,1,\ldots,n-1$ define

$$\underset{\sim}{x}_{k+1} = \underset{\sim}{x}_k + \alpha_k \, \underset{\sim}{p}_k$$

$$\underset{\sim}{p}_{k+1} = - P_N \, \underset{\sim}{g}_{k+1} + \beta_k \, \underset{\sim}{p}_k \quad .$$

There are various ways in which the scalar β_k can be defined. For example,

(a) Fletcher-Reeves : $\beta_k = \| P_N \, g_{k+1} \|_2^2 / \| P_N \, g_k \|_2^2$

(b) Polak-Ribiere [35] : $\beta_k = g_{k+1}^T P_N (g_{k+1} - g_k) / \| P_N \, g_k \|_2^2$

(c) Perry [33] : $\beta_k = g_{k+1}^T P_N (g_{k+1} - g_k - \alpha_k \, p_k) / p_k^T (g_{k+1} - g_k)$.

These acceleration devices, along with the possibility of multiple constraint deletion by using the parameter ρ , offer the promise of a much improved gradient projection method. However to substantiate this, such techniques need to be evaluated by actual computation, particularly with a large scale practical problem.

CHAPTER 7

NUMERICAL RESULTS

Computer programs were written to assess the viability of using methods based on the gradient projection method, the effectiveness of the acceleration techniques discussed in Chapter 6 and the importance of scaling when column transformation methods are used.

The first computer program that was written will be referred to as AGUBM as it is a program for solving augmented Generalized Upper Bounding problems. The method used is based on the results presented in Section 3.3. Also, another computer program, TTM, for transportation-type problems, was written. This program is based on the results of Chapter 4.

Both AGUBM and TTM are based on Rosen's gradient projection method with the following modifications:

(i) The parameter ρ, as described in Chapter 6 is used together with the provision for multiple constraint deletion.

(ii) The tolerance on the line-search accuracy is refined periodically as the infinite norm of the search direction tends towards zero. Partan-accelerated, continued-Partan, conjugate-gradient (Fletcher-Reeves) and "ordinary" versions of both AGUBM and TTM were written.

The evaluation of acceleration techniques for the gradient projection method was firstly made by using two small test problems and then with a class of problems based on

Berry's telephone network design problem. Numerical results in terms of the number of iterations and time taken will be given. However, the values of the time taken are an upper bound on the actual time taken as various pieces of information were printed out at each iteration and later used in the assessment of the performance of the algorithms.

7.1 TEST PROBLEM 1

The following small test problem was used to test the computer programs AGUBM and TTM and as a means for evaluating the effectiveness of the acceleration techniques.

$$\begin{aligned}\min \quad f(\underset{\sim}{x}) = & (x_1^1)^2 + (x_2^1)^2 + (x_3^1)^2 + (x_4^1)^2 \\ & -(x_1^2)^2 - (x_2^2)^2 - (x_3^2)^2 - (x_4^2)^2 \\ & +(x_1^3)^2 + (x_2^3)^2 + (x_3^3)^2 + (x_4^3)^2 + (x_5^3)^2 \\ & - 70x_1^1x_3^1x_5^3 + 60x_3^2x_4^2x_1^3 \\ & - 30x_2^1x_3^2x_5^3 - 570x_5^3\end{aligned}$$

subject to

$$\begin{aligned} & x_1^1 + x_2^1 + x_3^1 + x_4^1 = 8 \\ & x_1^2 + x_2^2 + x_3^2 + x_4^2 = 7 \\ & x_1^3 + x_2^3 + x_3^3 + x_4^3 + x_5^3 = 13 \\ & x_1^1 + x_1^2 + x_1^3 \geqslant 5 \\ & x_2^1 + x_2^2 + x_2^3 \geqslant 6 \\ & x_3^1 + x_3^2 + x_3^3 \geqslant 5 \\ & x_4^1 + x_4^2 + x_4^3 \geqslant 7 \\ & x_j^k \geqslant 0, \ \forall j,k \ . \end{aligned}$$

The starting point used was

[5,0,0,3,0,6,0,1,0,0,5,4,4]

with an objective function value of -2226.
The optimal solution was

[4,0,4,0,0,0,0,7,1,6,1,0,5]

with a corresponding objective function value of -8404. It can also be noted that at this point the set of active constraints is linearly dependent.

The main feature highlighted by this problem is illustrated by the iteration counts in Tables 2 and 3. With the ordinary TTM and $\rho = 0.5$ the program obtained the function value of -8392.01 at iteration 7. The solution procedure then continued slowly in the same subspace until iteration 1062, where three constraints were dropped. The remaining three iterations were then spent in refining the solution until the optimality conditions were satisfied. The application of the ordinary AGUBM produced similar results.

An explanation of this particularly slow convergence can be given as follows: the problem remained in the same subspace from iteration 7 to 1062, and in this subspace the movement of the successive points $\underset{\sim}{x}_i$ was down a narrow valley. Instead of moving directly downward in a direction parallel to the floor of the valley, the movement was such that successive points $\underset{\sim}{x}_i$ "bounced" from one wall of the valley to the other, while gradually creeping downwards.

Evidence for substantiating this explanation was provided by using the Partan-accelerated versions of AGUBM and TTM. It was noticed that the algorithm started to move as before until the stage where a Partan step was made. The resulting new point $\underset{\sim}{x}$ was then very close to the minimum and only

ρ	METHOD	ITERATIONS	TIME (CP SECS.)
0.5	ORDINARY	1009	15.168
	PARTAN	18	.346
	CONJUGATE	15	.328
0.6	ORDINARY	1009	15.056
	PARTAN	18	.341
	CONJUGATE	15	.302

TABLE 2 AGUBM: Results with acceleration devices and different values of ρ for test problem 1.

ρ	METHOD	ITERATIONS	TIME (CP SECS.)
0.5	ORDINARY	1065	15.202
	PARTAN	17	.318
	CONJUGATE	12	.264
0.6	ORDINARY	1070	15.121
	PARTAN	24	.409
	CONJUGATE	12	.250

TABLE 3 TTM: Results with acceleration devices and different values of ρ for test problem 1.

another three iterations were needed to refine $\underset{\sim}{x}$ to satisfy the optimality conditions.

From Tables 2 and 3 it can be seen that there is little difference between the performance of the computer programs AGUBM and TTM, except that TTM appears slightly better than AGUBM. This suggests that in this case, the change of scale produced by using a column transformation has no adverse effect on the convergence. It can also be noted that for both AGUBM and TTM, the two different values of ρ used resulted in similar performance.

7.2 TEST PROBLEM 2

The second test problem used in assessing the acceleration techniques for the gradient projection methods is the following *Weapon Assignment Problem:*

$$\min \quad f(\underset{\sim}{x}) = \sum_{j=1}^{20} u_j \left(\prod_{k=1}^{5} (a_j^k)^{x_j^k} - 1 \right)$$

subject to

$$\sum_{j=1}^{20} x_j^k = b^k \qquad k = 1,\ldots,5$$

$$\sum_{k=1}^{5} x_j^k \geqslant d_j \qquad j = 1,6,10,14,15,16,20$$

$$x_j^k \geqslant 0 \qquad j = 1,\ldots,20;\ k = 1,\ldots,5$$

where the constants a_j^k, b^k, d_j and u_j are given in Table 4.

This problem first appeared, with the additional requirement that x_j^k are integer, in the book by Bracken and McCormick [8] on nonlinear programming applications. It has also appeared in Himmelblau [25], problem 23 and in Murtagh and Saunders [31], problem 4. As in the latter, the continuous problem was considered. The optimal solution vector, for which $f(\underset{\sim}{x}) = -1735.569580$

and the initial starting point, with $f(\underset{\sim}{x}) = -649.94$, are given in Table 5.

This relatively small, but nontrivial problem can firstly be used as a means for comparison of AGUBM and TTM. From Tables 6 and 7 it can be seen that AGUBM is more successful. This can be directly related to the dependence upon scaling of the gradient projection method as can be seen by comparing the results for the ordinary versions of AGUBM and TTM. However, it can be seen that the adverse scaling produced by using TTM can be partly remedied by using the acceleration devices. This suggests that the deficiency of scale dependence inherent in the gradient projection method is not as pronounced when the acceleration devices are used. This can be expected since both Partan-acceleration and the use of conjugate-gradient directions introduce some second-order information into the search direction and second-order methods are generally scale independent.

The weapon assignment problem can also be used as a means of evaluating the acceleration devices used. The effect of parameter ρ is illustrated in Table 8. Here it can be seen that there is roughly an "optimal" value of ρ which minimizes the number of iterations required. For AGUBM it appears to be in the interval (.4,.6) and for TTM it appears to be in (.5,.7). The function of the ρ parameter is in determining which constraints should be dropped and which should remain active. This optimality is related to the following observations:

a_j^k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	b^k
1	1	.95	1	1	1	.85	.90	.85	.80	1	1	1	1	1	1	1	1	.95	1	1	200
2	.84	.83	.85	.84	.85	.81	.81	.82	.80	.86	1	.98	1	.88	.87	.88	.85	.84	.85	.85	100
3	.96	.95	.96	.96	.96	.90	.92	.91	.92	.95	.99	.98	.99	.98	.97	.98	.95	.92	.93	.92	300
4	1	1	1	1	1	1	1	1	1	.96	.91	.92	.91	.92	.98	.93	1	1	1	1	150
5	.92	.94	.92	.95	.95	.98	.98	1	1	.90	.95	.96	.91	.98	.99	.99	1	1	1	1	250
d_j	30	-	-	-	-	100	-	-	-	40	-	-	-	50	70	35	-	-	-	20	
u_j	60	50	50	75	40	60	35	30	25	150	30	45	125	200	200	130	100	100	100	150	

TABLE 4 Constants for test problem 2.

$\bar{x}_j^k$ / $\hat{x}_j^k$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	50	13.52				100	39.10	27.06	20.32	50			100							
2		1.36		23.49	20.90										100 26.21	24.24	3.80			
3						100									43.80		72.03	57.55	200 64.21	62.41
4											33.20	40.93		50 58.82		17.05				100
5	50.82	200 45.40	48.63							51.13			54.04			40				10

TABLE 5 Starting point $\bar{x}$ and optimal solution $\hat{x}$ (rounded to 2 d.p.) for test problem 2.

ρ	METHOD	ITERATIONS	TIME (CP SECS.)
0.4	ORDINARY	317	31.69
	PARTAN	113	10.62
	CONJUGATE	87	8.24
0.5	ORDINARY	381	37.88
	PARTAN	103	9.60
	CONJUGATE	87	8.28
0.6	ORDINARY	345	34.18
	PARTAN	146	13.49
	CONJUGATE	98	9.31

TABLE 6 AGUBM: Results with acceleration devices and different values of ρ for test problem 2.

ρ	METHOD	ITERATIONS	TIME (CP SECS.)
0.5	ORDINARY	1002	123.12
	PARTAN	252	27.08
	CONJUGATE	175	19.89
	C-PARTAN	277	30.16
0.6	ORDINARY	765	90.72
	PARTAN	187	19.29
	CONJUGATE	110	13.68
	C-PARTAN	202	24.21
0.7	ORDINARY	1100*	125.27
	PARTAN	281	29.44
	CONJUGATE	116	11.56
	C-PARTAN	311	33.47

TABLE 7 TTM: Results with acceleration devices and different values of ρ for test problem 2.

(* terminated at f = - 1735.50)

METHOD	ρ	ITERATIONS	TIME (CP SECS.)
PARTAN	0.5	252	27.08
	0.55	242	25.16
	0.575	231	28.04
	0.6	187	19.29
	0.625	196	19.74
	0.65	205	23.68
	0.7	281	29.44
CONJUGATE	0.5	175	19.89
	0.55	166	17.50
	0.575	142	14.75
	0.6	110	13.68
	0.625	109	11.28
	0.65	102	10.08
	0.7	116	11.56

TABLE 8 TTM: Comparison of different values of ρ for test problem 2.

(i) Dropping a constraint with a large negative Lagrange multiplier estimate should lead to a better search direction and hence lead to better convergence.

(ii) Dropping a constraint too soon will often lead to the phenomenon of zigzagging (or jamming). In this case, iterations are wasted in moving on and off the surface of some constraint and in some cases a significant reduction in the rate of convergence can occur.

The program TTM was in fact run with small values of ρ (for example, $\rho = 0.1$). The general effect noticed was that a large number of constraints were dropped. In the subsequent iterations, most of these constraints were returned, one at a time, to the active set.

The ρ value used in the programs AGUBM and TTM, is a fixed scalar. It seems plausible that if ρ were a variable, then the convergence could be further improved. The difficulty in determining what value ρ should have at any iteration is in fact the difficulty in knowing the best time to delete a constraint from the active set. This problem has still yet to be resolved in the literature.

The incorporation of the technique of Partan-acceleration can be seen, by reference to Tables 6 and 7, to give considerable improvement over the ordinary versions of AGUBM and TTM. The explanation for this arises from observations made from the computer output, that the problem seemed to have numerous narrow valleys. The ordinary methods generally took a considerable number of steps to pass down these valleys while the accelerated-Partan versions soon recognized the phenomenon and moved quickly downwards.

This *stalling* phenomenon occurring when narrow valleys are descended was in fact more pronounced in an earlier version of the AGUBM in which no ρ parameter test was used except for the original gradient projection method's test for dropping a constraint. With this particular computer program the problem often became "stuck" at values of f such as -1400 and -1500, which are a considerable distance from the minimum.

In an attempt to remedy this the program was modified by using the following constraint deletion strategy:

(i) Initially set positive tolerances ε_p and ε_λ. These will be changed to new values (that is, ε_p is reset to ε_p/m_1, ε_λ is reset to ε_λ/m_2 and for example $m_1 = m_2 = 10$) when $\|\underset{\sim}{p}\|_\infty < \varepsilon_p$.

(ii) If $\lambda_\ell < -\varepsilon_\lambda$, delete constraint number ℓ from active set. For $\varepsilon_p\ (= \varepsilon_\lambda)$ having initial values of 10^{-5}, 10^{-2} and 10^{-1} the values of f for which the program broke down, because of stalling, were -1400, -1435 and -1650 respectively. The best results were obtained for $\varepsilon_p = .5$ and .75 with $\varepsilon_p = .5$ giving slightly better convergence. However, the strategy involving the use of the parameter ρ proved to be clearly superior to this strategy.

As significant success was found in using the technique of Partan-acceleration it was decided to test the continued-Partan method. The results of such a trial are reported in Table 7. It was generally noticed that continued-Partan was not significantly better than Partan-acceleration. As a result it was decided, along with the consequent necessity of extra computation and computer storage, that its use was not warranted.

The final acceleration technique used was that of conjugate-gradient directions. As can be seen from Tables 6,7 and 8 it is a generally superior acceleration technique. The advantages are that second-order information can be easily incorporated into the search directions as soon as the optimization procedure remains in the same subspace for several iterations. This is in contrast to the Partan-acceleration method where it is necessary to wait for four or five iterations in the same subspace before a fairly reliable direction parallel to the valley walls can be obtained. It was noticed that if the accelerated-Partan direction was used too early then the search direction was not much better than that produced by the gradient projection method. This also means that the conjugate-gradient method has an advantage in early computations where optimizations in any one subspace seldom take more than three or four iterations.

Both Partan-acceleration and the use of conjugate-gradients produced superior convergence in the tail end of the optimizations. The ordinary methods were observed to be very slow near the optimal solution and the use of both Partan-acceleration and conjugate-gradients gave much improved convergence but with neither method being clearly dominant in terms of better search directions.

7.3 THE TELEPHONE NETWORK DESIGN PROBLEM

The class of problems considered by Berry were of the form:

$$\min \quad f(\underset{\sim}{x}) \tag{7.1}$$

$$\text{s.t.} \quad \sum_{j=1}^{\phi(k)} x_j^k = b^k \qquad k = 1,\ldots,\bar{k} \tag{7.2}$$

$$x_j^k \geq 0 \qquad j = 1,\ldots,\phi(k);\ k = 1,\ldots,\bar{k} \tag{7.3}$$

$$0 < b^k < t^k \qquad k = 1,\ldots,\bar{k} \quad . \tag{7.4}$$

The physical problem is one of assigning *junctions* (circuits) to links of an alternative routing telephone network in such a way that the total junction cost $f(\underset{\sim}{x})$ is minimized whilst ensuring that a specified fraction of the offered telephone traffic is carried between each pair of the origin and destination exchanges (OD pair). For each OD pair k there are a number of routes, $\phi(k)$, available to carry the OD traffic. The network is completely general in that the routing patterns and number of circuits can differ for each OD pair. The variable x_j^k (a chain flow) is the mean traffic (in erlangs) carried on the jth chain (route) between OD pair k. The total traffic offered to the kth OD pair is t^k, hence the specified OD traffic congestion is equal to $1-b^k/t^k$. A complete description of Berry's mathematical model relating the chain flow pattern $\underset{\sim}{x}$ to the number of junctions on each link as well as a description of $f(\underset{\sim}{x})$, $\nabla f(\underset{\sim}{x})$ and the constants b^k and t^k is given in [4].

There are two types of links which should be distinguished. The links carrying traffic directly between OD pairs are called *direct links* while the links connecting OD pairs via tandem exchanges carry traffic which overflows from the direct links and are called *overflow links*. In some cases the traffic offered to an OD pair is too small to justify the provision of

a direct link. When this occurs, the direct link is still permitted to exist but is considered to have zero junctions. It should also be noted that the chain having chain flow x_1^k is the kth direct link.

For the problem discussed by Berry [4], in which his mathematical model was applied to the Adelaide metropolitan telephone network, $\bar{k} = 1141$, $\phi(k) = 3$ for $k = 1,\ldots,\bar{k}$ and the number of overflow links was 212. This particular problem was first used as a basis for "tuning" the modified AGUBM and evaluating the usefulness of the acceleration devices. Once the "best" modified AGUBM had been decided upon the addition of overflow link capacity constraints was considered. Such additional constraints can be used in protecting weak or important parts of the network or can be used to force the flow on certain links to be zero if the number of junctions on that link is expected to be too small to warrant their inclusion. Alternatively such capacity constraints could be used in periodic updating of the telephone network in the cases where the number of junctions on a particular set of links should remain the same or not be decreased.

7.3.1 The Uncapacitated Problem

Three main starting points were used with the uncapacitated problem (7.1) - (7.4).

(i) Starting point 1.

$$(x_1^k, x_2^k, x_3^k) = \begin{cases} (.6,.25,.13)t^k & \text{if OD pair } k \text{ has a direct link} \\ (0,.6,.38)t^k & \text{otherwise.} \end{cases}$$

(ii) Starting point 2.

$$(x_1^k, x_2^k, x_3^k) = \begin{cases} (.98,0,0)t^k & \text{if OD pair } k \text{ has a direct link} \\ (0,0,.98)t^k & \text{otherwise.} \end{cases}$$

(iii) Starting point 3.

$$(x_1^k, x_2^k, x_3^k) = \begin{cases} (.98,0,0)t^k & \text{if OD pair } k \text{ has a direct link} \\ (0,.98,0)t^k & \text{otherwise.} \end{cases}$$

One of the first observations made was that an interior point such as starting point 1 is a bad choice of starting point. This is because a significant amount of time is spent in making small steps to the nearest non-active constraint hyperplane. A general characteristic of the gradient projection method is that at any iteration, usually no more than one constraint will become active. For this problem it has been seen that near the optimal solution, about 1600 (of a total of 3423) non-negativity constraints are active. This means that with such a starting point as starting point 1, at least 1600 iterations would need to be taken. This is clearly unsatisfactory as a function evaluation requires about 1 second and the calculation of the gradient vector takes about 3 seconds. Limited computational experiments were conducted with the strategy of placing a non-negativity constraint in the active set if $x_j^k < \varepsilon$, where $\varepsilon > 0$ is some tolerance. This means that if some variable x_j^k has a value ε_j^k, then after the gradient projection step, it will have the same value. Such strategies did give a reasonable acceleration of convergence but were still found to be unsatisfactory.

Starting points 2 and 3 however have the feature of having as many non-negativity constraints initially active as possible. The use of parameter ρ allows multiple constraint deletion during any iteration. This means that iterations should not be wasted as with starting point 1 nor with optimizing in subspaces that have good indications of not being the required subspace containing the optimal solution.

The objective function $f(\underset{\sim}{x})$ for the telephone network design problem has the feature of being flat near the optimal solution and relatively steep in the region of starting points 2 and 3. This means that in early iterations a line-search is not necessary as full steps are made to the nearest non-active constraint in the direction of the search. This provides significant savings in time as a line-search for such a function is very expensive. However, after a certain number of iterations this feature of the function begins to disappear occasionally. The one-dimensional search strategy then could be:

Make a full step if the function value decreases, otherwise use a line-search procedure.

Unfortunately, whilst saving time, such a strategy leads to the phenomenon of zigzagging. An example of this is given in Figure 5, where $\underset{\sim}{x}_0, \underset{\sim}{x}_1, \underset{\sim}{x}_2, \underset{\sim}{x}_3, \underset{\sim}{x}_4$ and $\underset{\sim}{x}_5$ are the points obtained by using such a strategy. The points $\underset{\sim}{x}'_1, \underset{\sim}{x}'_3$ and $\underset{\sim}{x}'_5$ are those that would have been obtained with line-searches from $\underset{\sim}{x}_0, \underset{\sim}{x}_2$ and $\underset{\sim}{x}_4$ respectively. It can also be noted that the optimal solution in this subspace, $\underset{\sim}{x}^*$, lies on the hyperplane $\hat{H}$ and if $\hat{H}$ had not been dropped at point $\underset{\sim}{x}_1$,

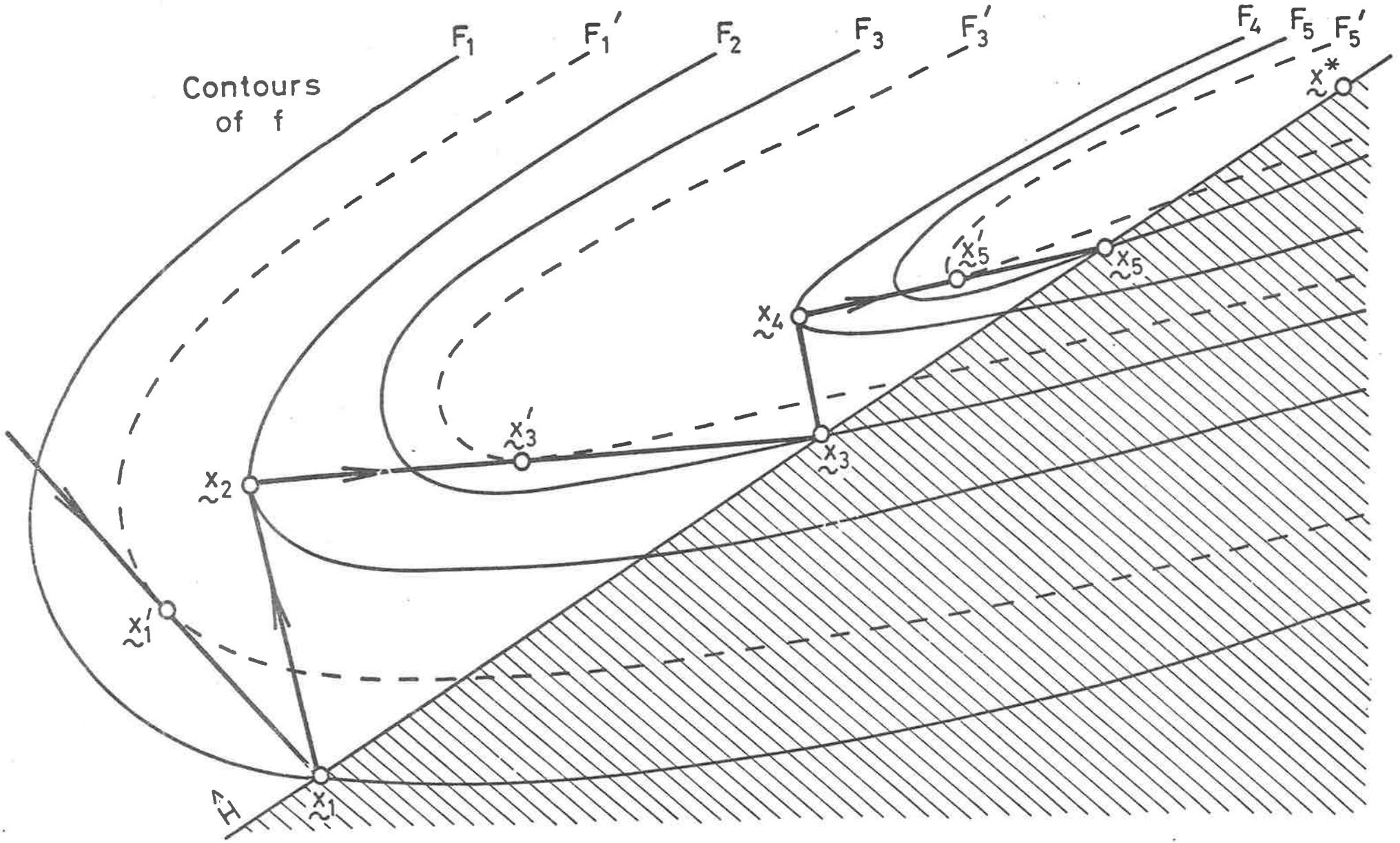


FIGURE 5 Zigzagging caused by non-optimal steplengths.

then the next point after $\underset{\sim}{x}_1$ would have been $\underset{\sim}{x}^*$ if a line-search had been used.

It was noticed that in early iterations with starting points 2 and 3, the values of $\|\underset{\sim}{p}\|_\infty$ were monotonically decreasing. After a certain number of iterations (200 for starting point 2 and 380 for starting point 3) the values of $\|\underset{\sim}{p}\|_\infty$ began to oscillate. At the same time it was found that taking a full step was not always giving a decrease in the objective function value and the phenomenon of zigzagging was frequently occurring. As a result the strategy of taking a full step (if possible) until the values of $\|\underset{\sim}{p}\|_\infty$ stop decreasing and then using some line-search procedure whose accuracy increases as $\|\underset{\sim}{p}\|_\infty$ decreases, was used. The strategy resulted in a significant improvement in convergence. It was also noticed that for starting points 2 and 3, that a full step could always be taken until the value of $\|\underset{\sim}{p}\|_\infty$ stopped decreasing.

Several values of ρ in the interval [.1,1.5] were used for starting points 2 and 3 but it was found that $\rho = 1.0$ was much better. The effect of dropping too many or too few constraints caused similar results of poor initial convergence.

The choice of starting point also affects the number of iterations required to obtain a good approximate optimal solution. This can be seen from Tables 9, 10, 11 and 12, where starting point 3 requires more iterations than starting point 2. However, both starting points lead to good approximate solutions in a reasonable number of iterations (especially since there are 3423 variables). Starting point 2 was run for a total of 1100 iterations giving an objective function

ITERATION	$f(\underset{\sim}{x})$	$\Delta f(\underset{\sim}{x})$	$\|\underset{\sim}{p}\|_\infty$
0	3,600,983.503		
1	3,583,700.740	17,282.763	17,398.178
2	3,455,797.715	127,909.025	8,558.985
3	3,315,614.129	140,177.586	4,303.889
4	3,294,873.589	20,740.540	2,261.122
5	3,282,504.404	12,369.185	2,145.563
6	3,276,623.163	5,881.241	2,077.340
7	3,241,693.348	34,929.815	2,044.882
8	3,239,718.797	2,174.551	1,855.787
9	3,236,931.908	2,786.889	1,845.499
10	3,236,624.315	307.593	1,830.982

TABLE 9 Iterates from the optimization of the telephone network design problem with starting point 2, $\rho = 1.0$ and conjugate-gradient acceleration.

ITERATION	$f(\underset{\sim}{x})$	$\Delta f(\underset{\sim}{x})$	$\|\underset{\sim}{p}\|_\infty$
0	3,676,484.987		
1	3,638,531.697	37,953.290	17,391.812
2	3,487,732.757	150,798.940	8,574.377
3	3,472,245.185	15,487.572	4,305.163
4	3,471,378.210	866.975	3,935,567
5	3,465,329.210	6,049.000	3,915.358
6	3,460,350.094	4,979.116	3,775.340
7	3,454,283.587	6,066.507	3,663.533
8	3,448,252.705	6,030.882	3,541.308
9	3,447,659.980	592.725	3,441.440
10	3,441,246.475	6,413.505	3,431.847

TABLE 10 Iterates from the optimization of the telephone network design problem with starting point 3, $\rho = 1.0$ and conjugate-gradient acceleration.

ITERATION	$f(\underset{\sim}{x})$	$\Delta f(\underset{\sim}{x})$	$\|\underset{\sim}{p}\|_\infty$
0	3,600,983.503		
20	3,070,336.934	530,646.569	1,192.560
40	2,950,363.670	119,973.264	851.708
60	2,893,671.478	56,692.192	725.286
80	2,842.602.786	51,068.692	398.448
100	2,813,717.046	28,885.740	267.218
120	2,798,884.656	14,832.390	262.337
140	2,780,652.634	18,232.022	183.941
160	2,769,652.772	10,999.862	143.077
180	2,760,212.007	9,440.765	112.053
200	2,749,507.969	10,704.038	91.012
220	2,744,133.006	5,374.963	
240	2,741,095.727	3,037.279	
260	2,738,738.227	2,357.500	
280	2,736,743.010	1,995.217	
300	2,734,733.622	2,009.388	
320	2,732,866.329	1,867.293	
340	2,730,716.928	2,149.401	
360	2,729,858.677	858.251	
380	2,728,672.636	1,186.041	
400	2,728,072.438	600.198	
420	2,727,335.761	736.677	
440	2,726,796.446	539.315	
460	2,726,207.916	588.530	
480	2,725,985.369	225.547	
500	2,725,432.496	549.863	
650	2,723,690.986	1,741.510	
800	2,722,612.551	1,078.435	

TABLE 11 Iterates from the optimization of the telephone network design problem with starting point 2, $\rho = 1.0$ and conjugate-gradient acceleration.

ITERATION	$f(\underset{\sim}{x})$	$\Delta f(\underset{\sim}{x})$	$\|\underset{\sim}{p}\|_\infty$
0	3,676,484.987		
20	3,298,404.121	378,080.866	2,175.840
40	3,222,734.979	75,669.142	1,785.963
60	3,132,744.236	89,990.743	1,354.837
80	3,057,081.304	75,662.932	1,123.290
100	2,987,091.083	69,990.221	928.230
120	2,936,381.844	50,709.239	796.098
140	2,903,160.177	33,221.667	725.058
160	2,874,873.505	28,286.672	476.436
180	2,852,474.176	22,399.329	379.435
200	2,832,155.461	20,318.715	304.180
220	2,821,201.176	10,954.285	270.954
240	2,804,262.872	16,938.304	249.920
260	2,790,923.717	13,339.155	237.899
280	2,780,342.011	10,581.706	169.843
300	2,770,551.182	9,790.829	133.328
320	2,764,059.150	6,492.032	111.732
340	2,754,393.606	9,665.544	91.919
360	2,750,838.660	3,554.946	86.072
380	2,745,512.190	5,326.470	68.482
400	2,741,138.865	4,373.325	
420	2,739,389.218	1,749.647	
440	2,737,193.975	2,195.243	
460	2,735,443.195	1,750.780	
480	2,733,789.642	1,653.553	
500	2,732,524.523	1,265.119	
650	2,727,543.639	4,980.884	
800	2,725,269.800	2,273.839	

TABLE 12 Iterates from the optimization of the telephone network design problem with starting point 3, $\rho = 1.0$ and conjugate-gradient acceleration.

value of 2,721,711.679 and a lower bound of 2,720,390.301 which is a difference of 1,321.378 or .05%. This lower bound estimating the objective function value was calculated by a method described by Faure and Huard [12, appendix].

7.3.2 The Inclusion of Capacity constraints

The link capacity constraints

$$\sum_{k=1}^{\bar{k}} \sum_{j=1}^{\phi(k)} \gamma_{ij}^{k} x_j^k \leq c_i \quad i = 1, \ldots, \bar{\ell} \qquad (7.5)$$

were also added to the mathematical program (7.1) - (7.4). Capacities were placed on the first 50 of the 212 overflow links. The values of c_i are given in Table 13. Three strarting points were used for this problem. The first two are starting points 2 and 3 of Section 7.3.1, which happen to be feasible for the problem (7.1) - (7.5). The third point used will be referred to as starting point 4 and was obtained by using the point obtained after 500 iterations with starting point 2 (that is $f = 2,725,432.496$) and a piecewise linear programming method that produces a feasible point given an infeasible point. The method used is described by Gill and Murray [19, Chapter 2] and can be described as follows:

$$\begin{aligned} \min \quad & \hat{f}(\underset{\sim}{x}) = - \sum_{j \in J_-(\underset{\sim}{x})} (\underset{\sim}{a}_j^T \underset{\sim}{x} - b_j) \\ \text{s.t.} \quad & \underset{\sim}{a}_j^T x \geq b_j \qquad j \notin J_-(\underset{\sim}{x}) \end{aligned}$$

where $J_-(\underset{\sim}{x})$ is the set of indices of constraints violated at $\underset{\sim}{x}$. This method quickly produced a feasible point in 45 iterations with 30 link capacity constraints initially violated. The objective function value of the resulting point (starting point 4) was 2,787,003.124.

.4393	61.1608	47.7482	35.9052	13.7169
52.4651	59.4231	54.0577	43.1306	57.2737
51.3952	40.7331	8.3869	97.4414	76.7687
98.8784	72.1752	8.4932	23.2275	53.6073
64.8261	59.1613	14.7295	56.1759	35.0659
90.2869	35.9268	65.7516	78.0830	66.8728
9.5002	32.0801	3.2526	14.4256	32.3960
32.0802	32.7527	25.3879	60.1992	9.3923
13.0866	16.9527	8.0288	13.8562	.4770
57.3250	24.5796	14.1182	29.6768	5.8982

TABLE 13 Values of c_i for $i = 1,\ldots,50$ (the values are ordered horizontally across the page).

As for the uncapacitated problem it can be seen from Tables 14 and 15 that starting point 2 produces better convergence. It can be seen from Tables 14, 15 and 16 that starting point 4 resulted in the best convergence. This is to be partly expected as starting point 4 is derived from a good approximation to the optimal solution for the uncapacitated problem. This suggests that if various sets of capacity constraints were to be added to the uncapacitated problem, a good approximation to the optimal solution for the uncapacitated problem could be stored and used to generate reasonable starting points for the capacitated problems.

With the three starting points considered, the number of active link capacity constraints at the respective points finally obtained were between 30 and 32. Other problems with slightly different c_i values were run and in one case the number of active link capacity constraints reached 45. In all these cases, no numerical instability was noticed, even though large numbers of non-negativity constraints entered or left the set of active constraints during the running of the computer program. As a result the merits of iterative refinement discussed in Section 3.3.3 have not been investigated in regard to large-scale nonlinear programming problems.

Another problem involving link capacity constraints considered was one in which the flows on overflow links 45, 66, 85, 105, 142, 167, 198, 205 and 211 were required to be zero. The reason for setting these flows to zero was that the expected number of junctions on these links were expected to be too small to warrant their inclusion. This was approached by setting c_i for these nine links to .001 to avoid potential

ITERATION	$f(\underset{\sim}{x})$	$\Delta f(\underset{\sim}{x})$	$\|\underset{\sim}{p}\|_\infty$
0	3,600,983.503		
20	3,070,336.934	530,646.569	1,192.560
40	2,950,363.670	119,973.264	851.708
60	2,895,292.015	55,071.655	733.663
80	2,852,116.368	33,175.647	424.891
100	2,822,194.205	29,922.163	297.302
120	2,805,861.727	16,332.478	242.474
140	2,783,459.162	22,402.565	191.618
160	2,771,156.165	12,302.997	148.554
180	2,762,554.410	8,601.755	120.645
200	2,753,166.149	9,388.261	97.334
220	2,745,174.071	7,992.078	
240	2,741,420.057	3,754.014	
260	2,739,047.799	2,372.258	
280	2,737,410.314	1,637.485	
300	2,736,418.313	992.001	
320	2,735,323.697	1,094.616	
340	2,734,178.442	1,145.255	
360	2,733,602.321	576.121	
380	2,733,215.084	387.237	
400	2,733,026.094	188.990	
420	2,732,775.782	250.312	
440	2,732,626.496	149.286	
460	2,732,420.072	206.424	
480	2,732,179.573	240.499	
500	2,731,985.106	194.467	
650	2,731,044.139	940.967	
800	2,730,737.887	306.252	

TABLE 14 Iterates from the optimization of the telephone network design problem with 50 link capacity constraints, starting point 2, $\rho = 1.0$ and conjugate-gradient acceleration.

ITERATION	$f(\underset{\sim}{x})$	$\Delta f(\underset{\sim}{x})$	$\|\underset{\sim}{p}\|_\infty$
0	3,676,484.987		
20	3,298,404.121	378,080.866	2,175.840
40	3,222,734.979	75,669.142	1,785.963
60	3,132,744.236	89,990.743	1,354.837
80	3,057,081.304	75,662.932	1,123.290
100	3,000,306.231	56,775.073	935.017
120	2,944,538.227	55,768.004	804.609
140	2,905,561.205	38,977.022	728.563
160	2,877,261.113	28,300.092	481.973
180	2,854,174.234	23,086.879	382.830
200	2,835,530.984	18,643.250	310.932
220	2,820,869.631	14,661.353	270.968
240	2,802,845.428	18,024.203	249.737
260	2,789,507.876	13,337.552	237.798
280	2,780,028.656	9,479.220	175.119
300	2,771,711.587	8,317.069	142.984
320	2,764,475.004	7,236.583	119.432
340	2,755,334.781	9,140.223	93.994
360	2,750,171.049	5,163.732	85.268
380	2,744,470.168	5,700.881	63.920
400	2,741,708.382	2,761.786	
420	2,739,371.993	2,336.389	
440	2,738,591.025	780.968	
460	2,737,168.650	1,422.375	
480	2,736,357.255	811.395	
500	2,735,672.163	685.092	
650	2,732,917.652	2,754.511	
800	2,731,933.818	983.834	

TABLE 15 Iterates from the optimization of the telephone network design problem with 50 link capacity constraints, starting point 3, $\rho = 1.0$ and conjugate-gradient acceleration.

ITERATION	$f(\underset{\sim}{x})$	$\Delta f(\underset{\sim}{x})$	$\|\underset{\sim}{p}\|_\infty$
0	2,787,003.124		
20	2,760,492.287	26,510.837	897.235
40	2,749,886.841	10,605.446	393.803
60	2,742,496.501	7,390.340	200.512
80	2,737,342.714	5,153.787	80.479
100	2,734,968.206	2,374.508	69.024
120	2,733,286.080	1,682.126	
140	2,732,672.415	613.665	
160	2,732,043.718	628.697	
180	2,731,827.983	215.735	
200	2,731,626.084	201.899	
220	2,731,521.330	104.754	
240	2,731,451.411	69.919	
260	2,731,289.745	161.666	
280	2,731,200.213	89.532	
300	2,731,117.846	82.367	
450	2,730,725.876	391.970	
600	2,730,583.451	142.425	
750	2,730,517.391	66.060	

TABLE 16 Iterates from the optimization of the telephone network design problem with 50 link capacity constraints, starting point 4, $\rho = 1.0$ and conjugate-gradient acceleration.

problems with linear dependence. In this case the starting point was derived from the point obtained after 650 iterations with the uncapacitated problem and starting point 2 (where f = 2,723,690.986). The feasible point method required 16 iterations to produce a feasible point (called starting point 5) with objective function value 2,725,161.777. Within 80 iterations (see Table 17) a point with a function value comparable to the original uncapacitated starting point was obtained with the added feature of zero flow and hence zero number of junctions on these nine overflow links.

A general feature noted with both the uncapacitated and capacitated problem was that the terminal convergence was very slow. It has been noticed that generally, the only significant changes in objective function value occurred in the iterations when a conjugate-gradient search direction was used. It has also been noticed that in early iterations the use of a conjugate-gradient direction usually resulted in a new constraint becoming active.

ITERATION	$f(\underset{\sim}{x})$	$\Delta f(\underset{\sim}{x})$
0	2,725,161.777	
20	2,724,056.302	1,105.475
40	2,723,954.256	102.046
60	2,723,728.951	225.305
80	2,723,589.582	139.369
100	2,723,404.261	185.321
200	2,722,826.869	577.392
300	2,722,499.243	327.626

TABLE 17 Iterates from the optimization of the telephone network design problem with 9 link capacity constraints, starting point 5, ρ = 1.0 and conjugate-gradient acceleration.

CHAPTER 8

DISCUSSION

In this thesis several techniques for solving large scale structured nonlinear programming problems have been discussed. These methods are mainly based on the gradient projection method. The techniques for the structured problems can also be extended for the solution of problems having additional general linear constraints. This is achieved by the use of Lemma 3.1 and the ability to update the operators P_{N_1} and Q_{M_1} when the matrix N_1 changes. Thus any similar operator can be used as long as suitable expressions for updating them after a change in N_1 can be obtained. For example Q_{M_1} can be interpreted as the reduced-gradient operator ZZ^T and (5.7) can be restated as

$$Q_M = ZZ^T - ZZ^T N_2^T [N_2 ZZ^T N_2^T]^{-1} N_2 ZZ^T$$

which is simply (3.7) with P_{N_1} replaced by ZZ^T .

It is anticipated that the use of column transformation methods can be relatively simply extended to other classes of structured problems. An example of such a problem is given as follows:

$$\min \quad f(\underset{\sim}{x})$$

$$\text{s.t.} \quad \sum_{j \in A_{ik}} x_{ij}^{k} \geq b_i^k$$

$$\sum_{k \in B_{ij}} x_{ij}^{k} \geq d_j^k$$

$$\sum_{i \in C_{jk}} x_{ij}^{k} \geq c_{jk}$$

$$x_{ij}^{k} \geq 0, \quad \forall i,j,k \ .$$

The projection methods considered have been seen to have the important feature that they use explicit projection operators or operators that can be described by several index sets. As a result the projection methods have favourable storage properties and an added advantage of increased numerical stability over other possible matrix methods.

The numerical results demonstrate that suitable variations of the gradient projection method can be successfully used to solve large scale structured problems and acceleration techniques such as the use of conjugate-gradients do produce a good acceleration of convergence. Research into other possible areas of improving the gradient projection method could lead to even better solution techniques for large scale problems that are too large to be solved by any second-order method.

BIBLIOGRAPHY

1. AKAIKE, H., *On a successive transformation of a probability distribution and its application to the analysis of the optimum gradient method*, Ann. Institute Statistical Mathematics, Tokyo 11,1 (1959).

2. BARTELS, R.H. and GOLUB, G.H., *The Simplex method of Linear programming using LU decomposition*, Communications of the Association for Computing Machinery 12, 5 (1969) 266-268.

3. BEALE, E.M.L., *Advanced algorithmic features for general mathematical programming systems*, in Abadie, J. ed. Integer and nonlinear programming (North Holland, 1970) 119 - 137.

4. BERRY, L.T.M., *A mathematical model for optimizing telephone networks*, (Ph.D. Thesis, University of Adelaide, December 1971).

5. BERRY, L.T.M., *On the solution of a structured nonlinear programme*, The Journal of the Australian Mathematical Society 19 (Series B) (1975) 242 - 248.

6. BJÖRCK, Å., *Comment on the iterative refinement of least-squares solutions*, Journal of the American Statistical Association 73, 361 (1978) 161 - 166.

7. BOULLION, T.L. and ODELL, P.L., Generalized inverse matrices (Wiley-Interscience, 1971).

8. BRACKEN, J. and McCORMICK, G.P., Selected applications of nonlinear programming (Wiley, New York, 1968).

9. BRUYN, S.J., *A mathematical model for the long-term planning of a telephone network*, (Ph. D. Thesis, University of Adelaide, October 1977).

10. BUCKLEY, A., *An alternative implementation of Goldfarb's minimization algorithm*, Mathematical Programming 8 (1975) 207 - 231.

11. DAVIDON, W.C., *Variable metric method for minimization*, ANL-5990, Atomic Energy Commission Research Development Report (1959).

12. FAURE, P. and HUARD, P., *Resolution de programmes mathematiques a fonction non lineaire par la methode du gradient reduit*, Revue Francaise de Recherche Operationnelle 9 (1965) 167 - 206.

13. FLETCHER, R., *A technique for orthogonalization*, Journal of Institute of Mathematics Applications 5 (1969) 162 - 166.

14. FLETCHER, R. and POWELL, M.J.D., *On the modification of* LDL^T *factorizations*, Mathematics of Computation 28, 128 (1974) 1067 - 1087.

15. FLETCHER, R. and REEVES, C.M., *Function minimization by conjugate gradients*, Computer Journal 7 (1964) 149 - 154.

16. FORSYTHE, G.E. and MOTZKIN, T.S., *Asymptotic properties of the optimum gradient method (abstract)*, American Mathematics Society Bulletin 57 (1951) 183.

17. FRANK, M. and WOLFE, P., *An algorithm for Quadratic programming*, Naval Research Logistics Quarterly 3 (1956) 95 - 110.

18. GILL, P.E., GOLUB, G.H., MURRAY, W. and SAUNDERS, M.A., *Methods for modifying matrix factorizations*, Mathematics of Computation 28 (1974) 505 - 535.

19. GILL, P.E. and MURRAY, W. eds. Numerical methods for constrained optimization (Academic Press, London, 1974).

20. GILL, P.E. and MURRAY, W., *Modification of matrix factorizations after a rank one change*, in Jacobs, D.,ed., The State of the art in numerical analysis (Academic Press, 1976) 55 - 83.

21. GILL, P.E. and MURRAY, W., *Numerically stable methods for quadratic programming*, Mathematical Programming 14, 3 (1978) 349 - 372.

22. GOLDFARB, D., *Extension of Davidon's variable metric method to maximization under linear inequality and equality constraints*, SIAM Journal on Applied Mathematics 17 (1969) 739 - 764.

23. GOLUB, G.H. and WILKINSON, J.H., *Note on the iterative refinement of least square solutions*, Numerische Mathematik 9 (1966) 139 - 148.

24. HARRIS, R.J., *Concepts of optimality in alternate routing networks*, (Ph.D. Thesis, University of Adelaide, January, 1974).

25. HIMMELBLAU, D.A., Applied nonlinear programming (McGraw-Hill, New York, 1972).

26. JACOBY, S.L.S., KOWALIK, J.S. and PIZZO, J.T., Iterative methods for nonlinear optimization problems (Prentice-Hall, 1972).

27. KLESSIG, R.W., *An algorithm for nonlinear multicommodity flow problems*, Networks 4 (1974) 343 - 355.

28. KOWALIK, J. and OSBORNE, M.R., *Descent methods* in Methods for unconstrained optimization problems (American Elsevier, New York, 1968).

29. McCORMICK, G.P., *The variable-reduction method for non-linear programming*, Management Science 17, 3 (1970) 146 - 160.

30. McCORMICK, G.P., *A second order method for the linearly constrained nonlinear programming problem*, in Rosen, J.B., Mangasarian, O.L. and Ritter, K., eds., Nonlinear programming (Academic, New York, 1970) 207 - 243.

31. MURTAGH, B.A. and SAUNDERS, M.A., *Large-scale linearly constrained optimization*, Mathematical Programming 14 (1978) 41 - 72.

32. PENTICO, D.W., *The assortment problem with nonlinear cost functions*, Operations Research 24, 6 (1976) 1129 - 1142.

33. PERRY, A., *An improved conjugate gradient algorithm*, Technical Note (March 1976). Dept. of Decision Sciences, Graduate School of Management. Northwestern University, Evanston, Illinois.

34. PIERRE, D.A., *Search techniques and nonlinear programming* in Optimization theory with Applications. (John Wiley and Sons, New York, 1969) Chapter 6.

35. POLAK, E., Computational methods in optimization: a unified approach (Academic, New York, 1971).

36. ROSEN, J.B., *The gradient projection method for nonlinear programming, Part I, Linear Constraints*, SIAM Journal on Applied Mathematics 8 (1960) 181 - 217.

37. SHAH, B.V., BUEHLER, R.J. and KEMPTHORNE, O., *Some algorithms for minimizing a function of several variables*, SIAM Journal 12 (1964) 74 - 92.

38. SORENSON, H.W., *Comparison of some conjugate gradient direction procedures for function minimization*, Journal of the Franklin Institute 288 (1969) 421 - 441.

39. STEWART, R.G., Introduction to Matrix Computations (Academic, 1974).

40. STIEFEL, E., *Uber Einige Methoden der Relaxationsrechnung*, Z. Angew Math. Physik 3 (1952).

41. WOLFE, P., *An extended Simplex method*, Notes of the American Mathematical Society 9, 4 (July, 1962).

42. WOLFE, P., *Methods of non-linear programming* in Graves, R.L. and Wolfe, P.,eds., Mathematical programming (McGraw-Hill, New York, 1963).

43. ZOUTENDIJK, G., Mathematical programming methods (North-Holland, 1976).

44. ZWART, P.B., *Nonlinear programming. The choice of direction by gradient projection*, Naval Research Logistics Quarterly 17 (1970)431 - 438.