SIGNOMIAL PROGRAMS WITH EQUALITY CONSTRAINTS: NUMERICAL SOLUTION AND APPLICATIONS

by

JAMES YAN

B.A.Sc., University of British Columbia, Vancouver, 1969
M.A.Sc., University of British Columbia, Vancouver, 1971

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

in the Department
of
Electrical Engineering

We accept this thesis as conforming to the
required standard

THE UNIVERSITY OF BRITISH COLUMBIA

December 1976

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the Head of my Department or by his representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Electrical Engineering

The University of British Columbia
2075 Wesbrook Place
Vancouver, Canada
V6T 1W5

Date Dec. 17, 1976

ABSTRACT

Many design problems from diverse engineering disciplines can be formulated as signomial programs with both equality and inequality constraints. However, existing computational methods of signomial programming are applicable to programs with inequality constraints only. In this thesis an algorithm is proposed and implemented to solve signomial programs with mixed inequality and equality constraints. The algorithm requires no manipulation of the constraints by the user and is compatible with the results of signomial programming.

The proposed algorithm is a synthesis shaped by three concepts: the method of multipliers viewed as a primal-dual method, partial dualization, and the retention of the structure of a signomial program. The strategy of the algorithm is to replace the original problem with a sequence of subproblems each subject to the original signomial inequality constraints only. The subproblem's objective function is the original cost augmented with the equality constraints and specified by a parameter vector $\underline{\lambda}$ and a penalty constant K. The algorithm then alternates between solving the subproblem and updating $\underline{\lambda}$ and K. The convergence of the algorithm is, under suitable assumptions, guaranteed by the convergence of the method of multipliers and the method used to solve the subproblem. Because the subproblem has the form of a regular signomial program, it can in principle be solved by the techniques of signomial programming.

A new numerical method implementing a variant of the Avriel-Williams algorithm for signomial programs is suggested for solving the subproblem. The method relies on monomial condensation, and the combined use of the reduced gradient method and a cutting plane scheme. Details

of the method's implementation are considered, and some computational experience has been acquired. The proposed method also has the advantageous flexibility of being able to handle non-signomial differentiable objective functions.

Four updating schemes for $\underline{\lambda}$ and K are formulated and evaluated in a series of numerical experiments. In terms of the rate of convergence, the most promising scheme tested is the use of the Hestenes-Powell rule for updating $\underline{\lambda}$ and the moderate monotonic increase of K after the completion of each subproblem. Convergence can also be considerably accelerated by properly scaling the equality constraints and performing only inexact minimization in the first few subproblems.

The applicability of the algorithms developed in this thesis is illustrated with the solution of three design examples drawn from structural design and chemical process engineering.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

ACKNOWLEDGEMENT

I would like to thank Dr. Erik V. Bohn for introducing me to geometric programming and for supervising my doctoral research.

My first encounter with the method of multipliers was due to Dr. Mordecai Avriel of Technion, Israel. I am also grateful to Dr. Avriel for his encouragement and for the many valuable discussions we had during his stay at U.B.C. in 1974-75.

I wish to acknowledge the use of Dr. Ron Dembo's code GGP made available to me by Dr. Avriel.

The timely assistance of my friend, Rose Cam, during the preparation of the first draft provided a much needed boost. She deserves a bouquet of roses.

Many thanks go to Misses Mary-Ellen Flanagan and Sannifer Louie for their excellent work in typing the final copy.

Finally, the graduate fellowships from UBC and a National Research Council of Canada Postgraduate Scholarship are gratefully acknowledged. Partial support of the project also came from NRC Grant No. A3134.

I. INTRODUCTION

1.1 Nonlinear Programming and Engineering Design

Design is the essential task of engineering. The design process usually begins with the recognition of a need. The process then proceeds with the analysis of the need, the synthesis of solution concepts likely to satisfy the need, the evaluation of the proposed solutions, and the selection of a final proposal. The selected concept is then further refined and made sufficiently specific for prototype evaluation and possible revision. Finally, the design plan is implemented.

In most stages of the design process opportunities for optimization exist. However, the disparate problems faced in each stage dictate the use of radically different optimization methods. One important step of the design process in which the efficacy of optimization has been amply demonstrated is the specification of a selected solution concept. At this point of the design process, the thorough analysis of the perceived need and the evaluation of the set of suggested solutions have usually produced a mathematical model. The model characterizes the structure of the accepted concept, identifies the decision variables and their functional relationships, and lists the constraints that have to be imposed because of physical, economic, or social considerations. The task the designer now faces is to choose the values of the design variables in order to impart to the selected solution concept a quantitative individuality. Most often, there is a gamut of acceptable choices, with each choice implying a certain level of performance by the model. If the designer's expectations from the model can be stated mathematically, as they often can, then each chosen set of satisfactory design variables can be judged relative to the mathematical norm of performance. This is

precisely a situation ripe for systematic optimization.

In many problems of engineering design, the mathematical model is described by a set of continuous functions of a Euclidean space. For such models, the search for the optimal set of design parameters is nothing else but a nonlinear program. For the realm of nonlinear programming is to solve the optimization problem

$$\text{minimize (min)} \quad f(\underline{x}) \tag{1.1}$$

$$\text{subject to (s.t.)} \quad g_j(\underline{x}) \leq 0, \quad j = 1, 2, \ldots, p \tag{1.2}$$

$$h_k(\underline{x}) = 0, \quad k = 1, 2, \ldots, q \tag{1.3}$$

where $\underline{x}$ is an m-dimensional vector, and f, g_j, and h_k are all continuous functions of the Euclidean space R^m. The designer therefore has at his disposal the powerful methods of nonlinear programming to seek the optimum design. He can abandon the traditional approach of obtaining only feasible solutions. With the aid of nonlinear programming he can achieve better performance with economical savings, improve modeling by sensitivity analysis, and even generate new solution concepts.

1.2 A Preview of the Thesis

A signomial is a nonlinear function defined by the difference of two positive sums of power functions. If a nonlinear program involves only signomials, it is known as a signomial program. In engineering design, many models from diverse engineering disciplines can be described by signomial programs. This observation is confirmed by the increasing number of published articles applying signomial programming to design. However, while engineering design models generally have both inequality and equality constraints, the theory of signomial programming has been developed in terms of inequality constraints only. Applying the

theory to solve design problems, therefore, often requires the transformation of equality constraints to inequalities. While such a transformation can be conveniently carried out in some cases, it is not clear how the transformation should be undertaken in general. The aim of this thesis is to develop a numerical method which the designer can use to solve signomial programs with mixed constraints without having to change one type to the other. The method is still rooted in the theory of signomial programming. Consequently, the method can make full use of the results of the theory.

In order to provide the background necessary for later discussions, the main results of signomial programming are summarized in Chapter II. Geometric programming is reviewed as a significant special case of signomial programming. Methods for solving both geometric programs and signomial programs are discussed.

In Chapter III, the central problem of signomial programs with inequality and equality constraints is formulated and solved by a primal-dual method. The development of the proposed algorithm is based on three ideas: the method of multipliers viewed as a primal-dual algorithm, partial dualization, and the retention of a signomial program's structure in the primal problem. The algorithm replaces the original problem with a sequence of inequality-constrained signomial programs dependent on a set of parameters defined as the vector $\underline{\lambda}$ and the scalar K. Between the solution of successive signomial programs, the parameters are updated iteratively according to some rules. The convergence of the algorithm is guaranteed by that of the method of multipliers and by the convergence of the algorithm used to solve the sequence of primal signomial programs.

The implementation of the algorithm proposed in Chapter III requires the selection of an algorithm for solving the primal signomial

programs and the specification of how the parameters $\underline{\lambda}$ and K should be updated. In Chapter IV, a new numerical method based on the Avriel-Williams algorithm for solving signomial programs is proposed. In this method the original nonconvex feasible set defined by signomial inequalities is approximated by a convex set obtained by monomial condensation. The nonconvex objective function is then minimized over the convex approximant by a combined reduced gradient and cutting plane algorithm. The minimization's solution in turn determines the next convex approximant. Considerations for implementing the new method are discussed in detail. Numerical experience with the method shows that it is comparable to, at times better than, another recent implementation of the Avriel-Williams algorithm. The proposed method, however, has the flexibility of handling a wider class of objective functions such as algebraic functionals of signomials. Finally, the proposed algorithm is refined to treat separately simple upper-bound constraints.

The work in Chapter V focuses on the performance of a set of updating rules for the parameters $\underline{\lambda}$ and K. A total of four combinations of updating rules for $\underline{\lambda}$ and K is tested in a series of numerical experiments. In the experiments, a sensitivity study is also performed to help understand the impact of the range of values of K. The results of the experiments serve as the basis for the choice of the suitable updating rules for solving the design problems of Chapter VI.

To illustrate how the algorithms discussed in the preceding chapters can be applied to engineering design, three selected design problems are presented in Chapter VI. The first is on obtaining the minimum-cost design of a hyperstatic pin-jointed structure. The second is concerned with achieving the optimum operating conditions of an

alkylation process. The last example involves the optimal trade-off between installation and operating costs of a small heat exchanger network. In each example, the problem is formulated with sufficient detail to allow appreciation of the physical basis of the objective function and the constraints. Numerical data are then substituted into the problem to cast it into a format compatible with this thesis' algorithms. The numerical solution is then obtained and interpreted.

Chapter VII concludes the thesis by giving a synopsis of the thesis and some suggestions for further research.

II. SIGNOMIAL PROGRAMMING

2.1 Introduction

The basic theory of signomial programming is reviewed in this chapter. The fundamental notion of a posynomial is first defined, and its relations to convexity and the geometric inequality are summarized. The formulation of a signomial program is then given, and the important properties of the formulation are examined. The stated problem is also compared to other equivalent formulations. If a signomial program has no negative terms, then a geometric program results. The theory of geometric programming, in particular, its duality properties, are discussed along the lines of Duffin, Peterson and Zener [1]. Finally, algorithms for solving geometric and signomial programs are considered.

2.2 Posynomials and Their Condensation

A posynomial $P_k(\underline{x})$ is a real-valued function defined as

$$P_k(\underline{x}) = \sum_{j=1}^{r_k} c_{jk} \prod_{i=1}^{m} x_i^{a_{ijk}} \tag{2.1}$$

where $\underline{x} = [x_1, x_2, \ldots, x_m]'$. The exponents a_{ijk} are arbitrary real numbers while the coefficients c_{jk} and the variables x_i are restricted to being positive. Each term of (2.1) is called a monomial. The form of a posynomial appears in many engineering design equations. Hence an optimization theory such as signomial programming that is based on posynomials is a powerful aid to the designer in his search for optimal designs.

In general, a posynomial is nonconvex in the space of $\underline{x}$. But if $t_i = \ln x_i$ and $\underline{t} = [t_1, t_2, \ldots, t_m]'$, then

$$x_i = \exp(t_i) \tag{2.2}$$

and

$$P_k(\underline{t}) = \sum_{j=1}^{r_k} c_{jk} \exp\left(\sum_{i=1}^{m} a_{ijk}\, t_i\right) \tag{2.3}$$

Since the exponential function is convex and all the coefficients c_{jk} are positive, it follows [1, p 54] that $P_k(\underline{t})$ is convex in the space of $\underline{t}$, which is just the Euclidean space R^m. Hence any posynomial can be transformed into a convex function. A consequence of this convexity property of posynomials is the bounding from below of any posynomial by a monomial. Let $\tilde{P}_k(\underline{t})$ be defined as

$$\tilde{P}_k(\underline{t}) = \ln P_k(\underline{t}) \tag{2.4}$$

It can be shown that $\tilde{P}_k(\underline{t})$ is a continuously differentiable convex function. Then for a given point $\hat{\underline{t}} > 0$,

$$\tilde{P}_k(\underline{t}) \geq \tilde{P}_k(\hat{\underline{t}}) + \sum_{i=1}^{m} (t_i - \hat{t}_i)\, \partial\tilde{P}_k(\hat{\underline{t}})/\partial t_i \tag{2.5}$$

The inequality (2.5) is simply the definition of a continuously differentiable convex function [2, p 28]. Exponentiating both sides of (2.5) gives

$$\exp(\tilde{P}_k(\underline{t})) \geq \exp(\tilde{P}_k(\hat{\underline{t}}) \cdot \prod_{i=1}^{m} \exp((t_i - \hat{t}_i)\, \partial\tilde{P}_k(\hat{\underline{t}})/\partial t_i \tag{2.6}$$

Substituting $t_i = \ln x_i$ and $\hat{t}_i = \ln \hat{x}_i$ simplifies (2.6) to

$$P_k(\underline{x}) \geq P_k(\hat{\underline{x}}) \cdot \prod_{i=1}^{m} (x_i/\hat{x}_i)^{b_i} \triangleq \hat{P}_k(\underline{x}, \hat{\underline{x}}) \tag{2.7}$$

where

$$b_i = \partial\tilde{P}_k(\underline{t})/\partial t_i = [(\partial P_k(\underline{x})/\partial x_i)(x_i/P_k(\underline{x}))]_{x = \hat{x}} \tag{2.8}$$

Observe that $\hat{P}_k(\underline{x}, \hat{\underline{x}})$ is a monomial. This technique of approximating a posynomial $P_k(\underline{x})$ at $\underline{x} = \hat{\underline{x}}$ with a monomial $\hat{P}_k(\underline{x}, \hat{\underline{x}})$ is called the condensation of $P_k(\underline{x})$ at $\underline{x} = \hat{\underline{x}}$. $\hat{P}_k(\underline{x}, \hat{\underline{x}})$ is known as the condensed posynomial of $P_k(\underline{x})$ at $\hat{\underline{x}}$.

The relationship between $P_k(\underline{x})$ and $\hat{P}_k(\underline{x}, \hat{\underline{x}})$ is given by Lemma 2.1.

Lemma 2.1

Let $\hat{P}_k(\underline{x}, \hat{\underline{x}})$ be the monomial condensed from the posynomial $P_k(\underline{x})$ at $\hat{\underline{x}}$. Then

(a) $P_k(\underline{x}) \geq \hat{P}_k(\underline{x}, \hat{\underline{x}})$ for all $\underline{x} > 0$

(b) $P_k(\hat{\underline{x}}) = \hat{P}_k(\hat{\underline{x}}, \hat{\underline{x}})$ (2.9)

(c) $[\partial P_k/\partial x_i]_{\underline{x} = \hat{\underline{x}}} = [\partial \hat{P}_k/\partial x_i]_{\underline{x} = \hat{\underline{x}}}$

Part (a) of Lemma 2.1 follows directly from the definition of the condensed posynomial. Part (b) can be verified by direct substitution. Part (c) can be shown by differentiating the right side of (2.7) and direct substitution.

Another approach to condensing posynomials is to use the geometric inequality based on the following lemma.

Lemma 2.2

If $u_j > 0$ and $\varepsilon_j \geq 0$ for $j = 1, 2, \ldots, N$, then

$$\left(\sum_{j=1}^{N} u_j\right)^{\lambda} \geq \prod_{j=1}^{N} \left(u_j/\varepsilon_j\right)^{\varepsilon_j} \lambda^{\lambda} \tag{2.10}$$

where

$$\lambda = \sum_{j=1}^{N} \varepsilon_j \tag{2.11}$$

and

$$(u_j/\varepsilon_j)^{\varepsilon_j} = 1 \quad \text{if } \varepsilon_j = 0 \tag{2.12}$$

Moreover the inequality becomes an equality if and only if

$$\varepsilon_j \sum_{i=1}^{N} u_i = u_j \sum_{i=1}^{N} \varepsilon_i, \quad j = 1, 2, \ldots, N \tag{2.13}$$

The proof of Lemma 2.2 may be found in [3].

Given a posynomial $P_k(\underline{x})$ defined by

$$P_k(\underline{x}) = \sum_{j=1}^{r_k} u_{jk}(\underline{x}), \quad u_{jk}(\underline{x}) = c_{jk} \prod_{i=1}^{m} x_j^{a_{ijk}} \tag{2.14}$$

Lemma 2.2 may be used to derive the condensed posynomial $\hat{P}_k(\underline{x}, \hat{\underline{x}})$ at the point $\hat{\underline{x}}$ by setting

$$\varepsilon_j = u_{jk}(\hat{\underline{x}})/P_k(\hat{\underline{x}}) \tag{2.15}$$

and

$$\lambda = 1.0 \tag{2.16}$$

The condensed posynomial $\hat{P}_k(\underline{x}, \hat{\underline{x}})$ becomes

$$\hat{P}_k(\underline{x}, \hat{\underline{x}}) = \theta_k(\hat{\underline{x}}) \prod_{i=1}^{m} x_i^{\alpha_{ik}(\hat{\underline{x}})} \tag{2.17}$$

where

$$\theta_k(\hat{\underline{x}}) = \prod_{j=1}^{r_k} (c_{jk}/\delta_j)^{\delta_j} \tag{2.18}$$

and

$$\alpha_{ik}(\hat{\underline{x}}) = \sum_{j=1}^{r_k} \varepsilon_j \, a_{ijk} \tag{2.19}$$

It can be easily verified that (2.17) satisfies Lemma 2.1.

2.3 Signomial Programming

A signomial $g_k(\underline{x})$ is a real-valued function defined as

$$g_k(\underline{x}) = P_k(\underline{x}) - Q_k(\underline{x}) \tag{2.20}$$

where $\underline{x} \in R_+^m \triangleq$ positive orthant of R^m. P_k and Q_k are posynomials of the form given in (2.1). A signomial program (SP) with inequality constraints only is the optimization problem defined as follows

$$\begin{aligned} (SP) \qquad & \min \; x_1 \\ & \text{s.t.} \quad g_k(\underline{x}) \leq 1, \quad k = 1, 2, \ldots, p \\ & \underline{x} \in R_+^m \end{aligned} \tag{2.21}$$

A signomial program with a signomial objective function $g_o(\underline{x})$ can be easily put into the format of (2.21) by adding the inequality $(g_o(\underline{x}) + C) \leq x_1$ to the constraint set. C is a constant added to insure the positivity of the left side for all feasible $\underline{x}$. Note that (2.21) also allows simple upper and lower bound constraints. Hence in this thesis, the formulation of (2.21) is considered as the standard problem statement of a signomial program. Different but equivalent formulations had been introduced by Passy and Wilde [4], Avriel and Williams [5], and Duffin and Peterson [6].

Because the difference of two convex functions is generally nonconvex, a signomial program is nonconvex. The solution of a signomial program cannot therefore guarantee a global minimum. Only local minima are assured under certain regularity conditions. Another consequence of the nonconvexity of signomial programs is the loss of a strong duality theory that is associated with convex programs. There is, however, a weak duality theory relating the local solutions of (SP) (the primal problem) to those of a linearly constrained dual problem. The precise definition of the dual problem depends on how the primal signomial program is formulated. The essence of the duality theory, however, remains the same, namely under some dual constraint qualifications a dual Kuhn-Tucker point can be derived from a known primal Kuhn-Tucker point, and vice versa. In either case, the primal and dual objective functions are equal when evaluated at the respective Kuhn-Tucker points. More details on the duality properties of signomial programs may be found in [4] - [6].

2.4 Geometric Programming

If no negative term is present in the signomial program of (2.21), then the problem becomes a geometric program involving posynomials only. The problem statement may, for later convenience, be rephrased as

$$\text{(GP)} \qquad \min P_o(\underline{x})$$

$$\text{s.t.} \quad P_k(\underline{x}) \leq 1.0, \quad k = 1, 2, \ldots, p \tag{2.22}$$

$$\underline{x} \in R_+^m$$

where

$$P_k(\underline{x}) = \sum_{j=M_k}^{N_k} u_j(\underline{x}), \quad u_j(\underline{x}) = c_j \prod_{i=1}^{m} x_i^{a_{ij}} \tag{2.23}$$

and

$$M_o = 1, \quad M_k = N_{k-1} + 1, \quad N_k = M_{k-1} + n_k \tag{2.24}$$

n_k is the number of terms in $P_k(\underline{x})$.

(2.22) is called the primal geometric program and x_i are the primal variables. If the logarithmic transformation $t_i = \ln x_i$ is used and (2.22) is expressed in terms of t_i, $i = 1, 2, \ldots, m$, then from the discussion of Section 2.2, the minimization problem reduces to a convex program since $P_k(\underline{t})$, $k = 0, 1, \ldots, p$, are convex functions of $\underline{t}$. By exploiting the monotone increasing property of the natural logarithmic function, the primal geometric program can be cast into the following transformed primal problem associated with the primal geometric program.

$$\text{(TGP)} \qquad \min \ln P_o(\underline{t})$$

$$\text{s.t.} \quad \ln P_k(\underline{t}) \leq 0, \quad k = 1, 2, \ldots, p \tag{2.25}$$

where

$$P_k(\underline{t}) = \sum_{j=M_k}^{N_k} c_j [\exp(\sum_{i=1}^{m} a_{ij} t_i)] \tag{2.26}$$

and M_k and N_k are given by (2.24).

Another important problem that can be associated with a given primal geometric program is the dual geometric program stated as follows:

$$\text{(DGP)} \qquad \max\ v(\underline{\delta}) = \prod_{j=1}^{N_p} (c_j/\delta_j)^{\delta_j} \prod_{i=1}^{p} \lambda_k^{\lambda_k} \tag{2.27}$$

$$\text{s.t.} \quad \lambda_k = \sum_{j=M_k}^{N_k} \delta_j, \quad k = 1, 2, \ldots, p \tag{2.28}$$

$$\sum_{j=1}^{N} {}^{p} a_{ij}\, \delta_j = 0, \quad i = 1, 2, \ldots, m \tag{2.29}$$

$$\sum_{j=1}^{N_o} \delta_j = 1 \tag{2.30}$$

and

$$\delta_j \geq 0, \quad j \tag{2.31}$$

In this program a_{ij}, C_j, M_k and N_k are defined in the same way as in the primal programs. (see (2.23) and (2.24).)

The function $v(\underline{\delta})$ is called the dual function and the variables δ_j are the dual variables. The relations (2.29), (2.30), and (2.31) are known as, respectively, the orthogonality condition, the normality condition, and the positivity condition. Note that each dual variable δ_j is linked to a monomial term in the primal problem (GP), while each λ_k is paired with a posynomial inequality constraint of (GP). It is shown in [1] that $\log v(\underline{\delta})$ is concave. Since max log $(v(\underline{\delta}))$ = max $v(\underline{\delta})$ over any feasible set of $\underline{\delta}$, with $\log v(\underline{\delta})$ replacing $v(\underline{\delta})$ as the objective function, (DGP) is a concave program with linear constraints. The significance of this transformation will be seen in the following discussion of the duality properties of geometric programs.

It has been shown that the solution of a geometric program (GP) is equivalent to the solution of a convex program (TGP) derived from (GP). Since convex programs have strong duality properties, the primal problem (GP) is expected to satisfy a strong duality theorem. This indeed is the case as shown by Theorem 2.1 due to Duffin, Peterson and Zener [1].

A program (primal or dual) is consistent if there exists a vector satisfying the program's constraints. (GP) is superconsistent if there exists a vector $\underline{\hat{x}}$ such that $P_k(\underline{\hat{x}}) < 1$, $k = 1, 2, \ldots, p$, and $\underline{\hat{x}} > 0$.

Theorem 2.1

Suppose that a primal geometric program (GP) is superconsistent and $P_o(\underline{x})$ is minimum at $\underline{x}^*$, a feasible vector of (GP). Let F_P and F_D be, respectively, the feasible sets of (GP) and its associated dual program (DGP). Then

(i) F_D is non-empty, and for all $\underline{x} \in F_P$ and $\underline{\delta} \in F_D$,

$$P_o(\underline{x}) \geq \min_{\underline{x} \in F_P} P_o(\underline{x}) \triangleq P_o(\underline{x}^*) = v(\underline{\delta}^*) \triangleq \max_{\underline{\delta} \in F_D} v(\underline{\delta}) \geq v(\underline{\delta}) \tag{2.32}$$

(ii) there exist non-negative Lagrange multipliers μ_k^* associated with $P_k(\underline{x}^*)$, $k = 1, 2, \ldots, p$, such that

$$\delta_j^* = \begin{cases} u_j(\underline{x}^*)/P_o(\underline{x}^*), & j = 1, 2, \ldots, N_o \qquad (2.33) \\ \mu_j^* u_j(\underline{x}^*)/P_k(\underline{x}^*), & j = M_k, \ldots, N_k;\ k = 1, 2, \ldots, p \qquad (2.34) \end{cases}$$

Furthermore,

$$\lambda_k^* = \mu_k^*/P_k(\underline{x}^*) \tag{2.35}$$

(iii) if $\underline{\delta}^*$ is the maximizer of (DGP), then $\underline{x}^*$ is the solution to any system of m linearly independant equations selected from the following equations:

$$\sum_{i=1}^{m} a_{ij} \log x_i = \log (\delta_j^* v(\underline{\delta}^*) c_j), \quad j = 1, 2, \ldots, N_o \tag{2.36}$$

$$\sum_{i=1}^{m} a_{ij} \log x_i = \log (\delta_j^*/c_j u_k^*), \quad j = M_k, \ldots, N_k, \quad k = 1, 2, \ldots, p \tag{2.37}$$

Theorem 2.1 has two important practical implications. From a computational point of view, it is indeed attractive that the solution to the primal program can be obtained via the solution of a concave program with linear constraints, and the solution of a system of linear algebraic equations. From an engineering design viewpoint, (2.32) is appealing since it allows the designer to use the inequality $P_o(\underline{x}) \geq v(\underline{\delta})$ for $\underline{x} \in F_p$ and $\underline{\delta} \in F_D$ to decide whether the current feasible design is acceptable, depending on the gap $P_o(\underline{x}) - v(\underline{\delta})$.

2.5 Solution of Geometric Programs

Theorem 2.1 offers two options for solving a geometric program - a direct primal solution or one via the dual problem (DGP). The main advantage of the dual method is the structure of (DGP) - the maximization of a concave function over a linear manifold. Algorithms tailored for this class of nonlinear programs may be used. (e.g. see references [7] - [10].) A prime consideration in the dual approach is the linear manifold's dimension usually called the degree of difficulty d. Since there are N_p dual variables and m + 1 dual linear equality constraints, $d = N_p - (m + 1)$ for the case when the $m \times N_p$ ($m < N_p$) exponent matrix $A = [a_{ij}]$ has full rank. If the rank of A is less than m, the primal variables can be redefined to insure that A has full rank [1, pp 82-83]. If $d = 0$, the dual optimal solution $\underline{\delta}^*$ is conveniently obtained as the unique solution of a system of linear algebraic equations. But if $d > 0$, iterative methods must be used.

Employing the dual approach has its difficulties. The degree of difficulty d can be quite large, although the primal space has low dimensionality. Hence the use of the dual method involves a trade off

between the case of linear constraints and the possible burden of high dimensionality. Another serious potential difficulty is due to the existence of slack primal constraints. From the necessary complementary slackness condition for optimality, the optimal Lagrange multiplier μ_k^* associated with the kth slack primal inequality constraint must be zero. Hence the kth set of equations defined by (2.37) cannot be used. It is possible that not enough equations are available to yield the optimal primal vector $\underline{x}^*$. To overcome the difficulty posed by slack primal constraints, Kochenberger [11] proposed the addition of slack variables to loose constraints. The drawback of this method is that the loose constraints have to be known a priori or else slack variables need to be added to each constraint. In the latter case, both the dimension of the primal problem and the degree of difficulty are increased.

A primal approach would solve the transformed primal geometric program (TGP) given in (2.25). The motivation is obvious. While (GP) is generally nonconvex, its equivalent problem (TGP) is a nice convex program that can be solved by any of the algorithms developed for convex programs. For example, Avriel et al [12] has applied a cutting plane algorithm to solve geometric programs. Beside exploiting convexity, the primal approach can handle loose constraints with no difficulty. Simple upper and lower bound constraints on $\underline{x}$ can also be easily taken into account, while the same type of constraints would mean increased degree of difficulty in a dual method.

2.6 Solution of Signomial Programs

Signomial programs are nonconvex programs and their solutions are usually at best local minima only. As in the case of geometric pro-

grams, the solution to a signomial program can be obtained by a primal or a dual method. The dual approach solves a dual program that still has only linear constraints, but the logarithm of its dual objective function is not concave [4] [6]. Furthermore, the dual solution no longer satisfies a strong duality theory. However, the potential difficulties of a dual approach, as discussed in Section 2.5, remain.

The primal approach to solving signomial programs is based on convex approximation. In lieu of solving the original nonconvex program, a sequence of convex programs is solved such that the corresponding sequence of solutions approaches a local minimum of the nonconvex program. Charnes and Cooper [13] first suggested such a computational strategy for signomial programs. Later several authors [5], [14], [15] adopted the same strategy but differed in the specifics of how to convexify the original nonconvex program. A practical algorithm that has been used for this thesis is that proposed by Avriel and Williams [5] and further developed by Avriel et al [12]. A computer code implementing the algorithm was written by Dembo [16].

The Avriel - Williams algorithm solves the following signomial program

$$\text{(SP)} \qquad \begin{aligned} &\min\ x_1 \\ &\text{s.t.} \quad \underline{x} \in X \end{aligned} \tag{2.38}$$

where $\underline{x} = [x_1, x_2, \ldots, x_m]'$,

$$X = \{\underline{x} : \underline{x} \in R^m;\ P_k(\underline{x}) - Q_k(\underline{x}) \leq 1,\ k = 1,2,\ldots,p;\ 0 < x^L \leq \underline{x} \leq \underline{x}^U\}$$

Note that (2.38) is almost identical in formulation to (2.21) except in the former the simple upper and lower bounds on $\underline{x}$ are explicitly stated. Consider the kth signomial inequality constraint in the feasible set X. The constraint can be rewritten as

$$P_k(\underline{x})/(1 + Q_k(\underline{x})) \leq 1 \tag{2.39}$$

(2.39) is clearly a ratio of two posynomials. Assume that a point $\underline{x}^{(i-1)} \in X$ is known. The denominator $(1 + Q_k(\underline{x}))$ can be condensed at $\underline{x}^{(i-1)}$ to yield a monomial $\hat{Q}_k(\underline{x}, \underline{x}^{(i-1)})$. If the denominator of each constraint in X is replaced by their respective condensed posynomial, the following problem, called $(GP^{(i)})$, results.

$$(GP^{(i)}) \qquad \min x_1$$

$$\text{s.t. } P_k(\underline{x})/\hat{Q}_k(\underline{x}, \underline{x}^{(i-1)}) \leq 1, \quad k = 1, 2, \ldots, p \tag{2.40}$$

$$0 < \underline{x}^L \leq \underline{x} \leq \underline{x}^U$$

The above problem has the following properties:

(a) It is a geometric program since a posynomial divided by a monomial remains a posynomial.

(b) Its feasible set is contained in the feasible set X of (2.38) because by (2.9)

$$P_k(\underline{x})/(1 + Q_k(\underline{x})) \leq P_k(\underline{x})/\hat{Q}_k(\underline{x}, \underline{x}^{(i-1)}) \leq 1 \tag{2.41}$$

for any $\underline{x}$ feasible for $(GP^{(i)})$

(c) By (2.41), its optimal solution is feasible, but not necessarily optimal, for (SP) (2.38).

The Avriel - Williams algorithm solves a sequence of problems of the form specified by (2.40). It is shown in [5] that the algorithm produces a sequence of points converging to a local minimum of the original problem (SP), provided that the signomial program has a regular feasible set. A feasible set X is <u>regular</u> if it is non-empty and compact, and the gradients of its tight constraints generate a pointed cone, (i.e., the origin $\notin$ convex hull of the gradients mentioned).

In summary, the algorithm may be stated in the following manner:

<u>Algorithm 2.1</u>

<u>Step 0</u>: Given $\underline{x}^{(0)} \in X$ and let $\underline{x}^{(i)}$ be the solution to $GP^{(i)}$. Set i = 1.

Step 1: Construct $GP^{(i)}$ according to (2.40).

Step 2: Solve $GP^{(i)}$ and obtain the point $\underline{x}^{(i)}$.

Step 3: Stop if the convergence criterion is satisfied. Otherwise, $i = i + 1$ and go to Step 1.

It is worth noting that this algorithm may use any primal or dual algorithm for solving the geometric programs. The Avriel-Williams algorithm is a primal method only with respect to the nonconvex program (SP).

III. SIGNOMIAL PROGRAMS WITH EQUALITY CONSTRAINTS

3.1 Motivation

Since its first introduction in 1961, geometric programming, and later its generalization signomial programming, have been formulated in terms of inequality constraints only. Such a formulation allows the development of a duality theory that is, in the case of geometric programming, elegant and computationally attractive. However, an examination of many engineering design problems shows that there is a strong need for extending the problem formulation to include explicitly equality constraints in the feasibility set. From the theoretical and computational viewpoints, there are also compelling reasons for an algorithm that can handle equality constraints and still retain the basic structure of the original signomial programming formulation.

The presence of equality constraints in engineering design can be attributed to numerous reasons. Some of these are: 1) the conservation laws of mass, energy, momentum, and charge, 2) the input-output relation of a transformation, such as a stage in a process, 3) the need to maintain equilibrium conditions, original topology, etc., 4) the relation of a variable in the cost function to other design parameters, 5) the need to satisfy boundary conditions. Some specific examples that are amenable to the signomial programming formulation and that illustrate well some of the mentioned reasons are chemical reactor design [17], steady-state process optimization [18], and optimal structure design [19].

A nonlinear program with nonlinear equality constraints is always nonconvex. In the case of geometric programming, the strong duality relations of Theorem 2.1 no longer apply. In fact, it can easily be shown

that a geometric program with equality constraints is equivalent to an inequality-constrained signomial program, just as are signomial programs with equality constraints. Theoretically, these equivalent signomial programs can be solved by algorithms for solving signomial programs. But this approach requires the replacement of each equality constraint $g_k(\underline{x})=1$ with the pair of inequalities $g_k(\underline{x})\leq 1$ and $g_k(\underline{x})\geq 1$. The method has two computational drawbacks: 1) the size of the problem is increased, 2) the primal feasible set has no interior and hence is not regular. Another possible scheme is to replace each equality constraint $g_k(\underline{x})=1$ with either $g_k(\underline{x})\leq 1$ or $g_k(\underline{x})\geq 1$. The first objection to this scheme is that if the number q of equality constraints is large, there are 2^q combinations to consider, and the trial-and-error computation can be prohibitive unless some heuristic search plan is followed. A more important objection to this scheme is that there is no guarantee that the original solution will be obtained. It is conceivable, for example, that each of the 2^q combinations has a slack constraint, yet the solution sought requires all the constraints to be tight since they are equality constraints.

In this chapter an algorithm is developed to handle signomial equality constraints without the replacement by equivalent inequality constraints. The algorithm incorporates the equality constraints into the objective function and solves a sequence of inequality-constrained signomial programs in lieu of the given problem. The original inequality constraints are retained as they are.

3.2 Previous Work

The published literature on signomial programming contains very limited work on the solution of signomial programs with equality

constraints. Blau and Wilde [17] reported an application of signomial programming involving equality constraints. They, however, used physical reasoning and some algebra to transform the constraints into inequalities. The same approach was adopted by Rijckaert [18]. In applying signomial programming to nonlinear assignment problems, Passy [20] used the standard technique of substituting each equality constraint with a pair of inequalities opposite in sense. Later, Blau and Wilde [21] considered signomial programs with equality constraints only. They proposed a Newton-Raphson method to solve for the stationary points of the primal's Lagrangian. It seems that the algorithm reported in this thesis and in [22] is the first effort to solve signomial programs with mixed inequality and equality constraints, in a manner that preserves the key properties of signomials without requiring any constraint transformation by the user.

3.3 Problem Statement

The problem that is of interest in this chapter may be stated formally as follows

$$\text{(SPE)} \qquad \begin{aligned} &\min\ x_1 \\ &\text{s.t.} \quad \underline{x} \in X \\ &h_k(\underline{x}) \triangleq g_{p+k}(\underline{x}) - 1 = 0, \quad k = 1, 2, \ldots, q \end{aligned} \tag{3.1}$$

where

$$X = \{\underline{x} : \underline{x} \in R^m;\ g_k(\underline{x}) \leq 1,\ k = 1, 2, \ldots, p;\ 0 < \underline{x}^L \leq \underline{x} \leq \underline{x}^U\}$$

All the functions $g_\cdot : R_+^m \rightarrow R$ are signomials as defined in (2.20). The difference between problem (SP) (2.38) and (SPE) (3.1) is solely due to the q signomial equality constraints. The next section details the development of an algorithm that solves (SPE) via a sequence of (SP)-type problems which can be conveniently solved by the various algorithms referred to in Section 2.6.

3.4 Proposed Algorithm

3.4.1 Background

Constraints in nonlinear programming are generally treated in four ways, each being the basis of a family of algorithms. The first is to transform the constrained problem into an unconstrained one by incorporating all the constraints into the objective function. While such an approach exploits the efficiency and ease of unconstrained optimization algorithms, its major disadvantage is that whatever special structure the constraints may have, is lost. The second strategy is the primal feasible method in which unconstrained optimization algorithms are modified to insure that the sequence of points generated are all feasible for the given problem. The primal feasible method works well with linear constraints, but with nonlinear constraints, it is plagued with the difficulties of getting an initial feasible point and remaining within the feasible set. Approximation leading to the use of linear programming or simplex-type operations is the third way to solve constrained nonlinear programs. This approach can be quite efficient, depending on how well the feasible set is approximated. The fourth approach is the Lagrangian or dual method which is motivated by the viewpoint that the Lagrange multipliers are the fundamental unknowns associated with a constrained optimization problem. Hence, the method does not solve directly the original problem but instead attacks an alternate problem, the dual problem, whose unknowns are the Lagrange multipliers of the original problem.

In almost all the algorithms of each family, both equality and inequality constraints are treated in the same manner. They are all included in the objective function of a new problem, or all are used to maintain feasibility, or all are approximated in some form, or all are dualized to

define the dual problem. The algorithm discussed in this chapter is an exception to this pattern of handling identically both equality and inequality constraints. The algorithm is designed to be a synthesis of the four basic solution strategies in order to preserve the useful properties of a signomial program.

3.4.2 Development of the Algorithm

The difficulty of Problem (SPE) defined in (3.1) is posed by the signomial equality constraints (3.1b). Without these constraints the problem simply reduces to a signomial program that can be conveniently solved by a variety of algorithms discussed in Section 2.6. It is clear then that the inequality constraints of (3.1) should remain untouched. The immediate question that has to be confronted is: How should the equality constraints be manipulated? The techniques discussed in Section 3.1 have been ruled out for the reasons mentioned there. An examination of the basic solution strategies in nonlinear programming suggests that the transformation method, otherwise known as the penalty function method, is perhaps the suitable approach.

Using an external penalty formulation [23], the problem (SPE) can be rewritten as

$$\begin{aligned} \min \quad & x_1 + r^{(i)} \sum_{k=1}^{q} |h_k(\underline{x})|^{\beta} \\ \text{s.t.} \quad & \underline{x} \in X \end{aligned} \tag{3.2}$$

In (3.2), $\beta \geq 1$ and the sequence $\{r^{(i)}\}$ is an unbounded increasing positive sequence. If β is an even integer (it is usually 2), then the objective function of (3.2) is a signomial since any even integral power of a signomial is itself a signomial. Thus in this case (3.2) is a signomial program as defined as problem (SP). Let $\underline{x}^{(i)}$ be the solution to the ith subproblem defined by

(3.2). Also let x_1^* be the minimum value of x_1 in (SPE). The convergence of the sequence $\{\underline{x}^{(i)}\}$ to a local minimum of (SPE) is obtained according to the following result due to Fiacco and McCormick [23]: If the local minima of problem (SPE) form a nonempty compact set A and $r^{(i)} \to \infty$ as $i \to \infty$, then there exists a compact set S such that $A \subset$ interior of S and for sufficiently large i, $\underline{x}^{(i)} \in$ interior of S. Moreover, $x_1 \to x_1^*$, and every limit point of any convergent subsequence of $\{\underline{x}^{(i)}\}$ is in A. While the convergence of the exterior penalty formulation is satisfactory, the transformation, indeed all penalty function methods, possess the unfavorable property of exhibiting ill-conditioned Hessians. It is a fundamental property [24] of penalty function methods that as $r^{(i)} \to \infty$, the Hessian of the transformed objective function is equal to the sum of the Hessian of the original problem's Lagrangian and a matrix of rank R, whose eigenvalues approach infinity. The number R gives the number of active constraints. To overcome the numerical difficulties caused by the ill-conditioning of the Hessian as $r^{(i)} \to \infty$, the requirement that $r^{(i)} \to \infty$ in order to achieve convergence must be relaxed. This goal is realized by the method of multipliers first independently proposed by Hestenes [25] and Powell [26]. Because the method is a modified penalty function method, it preserves the attractive property that the transformed problem is an inequality-constrained signomial program.

In its original formulation the method of multipliers solves the nonlinear program

$$\begin{aligned} &\min\ f_o(\underline{x}) \\ \text{s.t.}\quad &f_k(\underline{x}) = 0,\ k = 1, 2, \ldots, q \\ &\underline{x} \in R^m \end{aligned} \tag{3.3}$$

by solving a sequence of unconstrained subproblems each defined as

$$\min \quad \ell(\underline{x}, \underline{\lambda}^{(i)}, K^{(ii)}) = f_o(\underline{x}) + \sum_{k=1}^{q} \lambda_k^{(i)} f_k(\underline{x}) + K^{(i)} \sum_{k=1}^{q} f_k^{\,2}(\underline{x})$$

$$\text{s.t.} \quad \underline{x} \in R^m \qquad (3.4)$$

$$\underline{\lambda} \in R^q$$

It is understood that $\underline{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_q]'$. In (3.4), $\lambda_k^{(i)}$ is the ith estimate of the optimal Lagrange multipliers λ_k^*, $k = 1,2, \ldots,q$ and $K^{(i)}$ is a sufficiently large but finite positive constant. The general outline of the algorithm is as follows:

Algorithm 3.1

Step 1 Set $i = 1$. Assume $\underline{\lambda}^{(i)}$ and $K^{(i)}$ are specified.

Step 2 Solve (3.4). Let the solution be $\underline{x}^{(i)}$.

Step 3 If $(\underline{x}^{(i)}) = 0$, $\forall k$, stop. Otherwise, set $i = i + 1$.

Update $\underline{\lambda}^{(i)}$ and/or $K^{(i)}$, and go to step 2.

The updating rules for $\underline{\lambda}^{(i)}$ and $K^{(i)}$ will be discussed later. Conditions for the convergence of the algorithms will be considered in the next section. Now what is of interest is the adaptation of the method of multipliers to accomodate inequality constraints.

Besides being considered as a modified penalty method, the method of multipliers can also be viewed as a modified primal-dual method. This view was sketched by Luenberger [24] and more recently discussed in detail by Bertsekas [27]. Several other researchers (e.g. [28] - [32]) have adopted the same point of view to study the theoretical properties of the method of multipliers. Consider the nonlinear program given by (3.3). An equivalent problem, in the sense of having the same solution and the same minimum objective function, is the following problem

$$\min \quad f_o(\underline{x}) + K \sum_{k=1}^{q} f_k^{\,2}(\underline{x})$$

$$\text{s.t.} \quad f_k(\underline{x}) = 0, \quad k = 1, 2, \ldots, q \qquad (3.5)$$

$$\underline{x} \in R^m$$

for any $K > 0$. The Lagrangian of (3.5) is

$$\ell(\underline{x}, \underline{\lambda}, K) = f_o(\underline{x}) + \sum_{k=1}^{q} \lambda_k f_k(\underline{x}) + K \sum_{k=1}^{q} f_k^2(\underline{x}) \qquad (3.6)$$

Let $\underline{x}^*$ be a local minimum of the original problem (3.3). Associated with this solution is the optimal Lagrange multiplier $\underline{\lambda}^*$. Then for sufficiently large but finite K, say $K \geq K^*$ for some K^*, the Hessian of $\ell(\underline{x},\underline{\lambda},K)$ at $(\underline{x}^*,\underline{\lambda}^*)$ is positive definite. As a result, for every $K \geq K^*$, problem (3.5) has a locally convex structure, and local duality theory is applicable.

Let the function d_K: $R^q \rightarrow R$ be defined as

$$d_k(\underline{\lambda}) = \min_{\underline{x} \in R^m} \ell(\underline{x}, \underline{\lambda}, K) \qquad (3.6)$$

Then the method of multipliers can be cast as a primal-dual algorithm alternating between the primal problem (3.6) and the dual problem defined as

$$\max_{\underline{\lambda} \in R^q} d_k(\underline{\lambda}) \qquad (3.7)$$

Note that both problems are unconstrained. Now suppose that a set of inequality constraints $\theta_j(\underline{x}) \leq 0$, $j = 1,2, \ldots, p$, is appended to the feasible set of (3.5). These additional constraints can be easily handled by the concept of partial duality [24], [32]. The definition of the dual function $d_k(\underline{\lambda})$ need not include the Lagrange multipliers of all the primal constraints. If local convexity applies, dualization can be with respect to any subset of the primal constraints. Hence, the dual function $d_K(\underline{\lambda})$ can be redefined as

$$d_k(\underline{\lambda}) = \max_{\underline{x} \in \hat{X}} \ell(\underline{x}, \underline{\lambda}, K) \qquad (3.8)$$

where $\hat{X} = \{\underline{x} : \underline{x} \in R^m, \ \theta_j(\underline{x}) \leq 0, \ j = 1, 2, \ldots, p\}$

The dual problem is the same as that of (3.7). Since the method of multipliers is a primal-dual method, it follows that one way to apply the method of multipliers to nonlinear programs with inequality constraints is simply to

use partial duality and incorporate the inequality constraints into the definition of the dual function. As a result, the original problem is replaced with a sequence of less constrained subproblems. Clearly such an approach is advantageous only if the subproblem's constraints exhibit a special structure worthy of special consideration. This is precisely the case with signomial programs with equality constraints.

Applying the method of multipliers to the signomial program stated in (3.1) yields the following ith subproblem

$$(\pi_1^{(i)}) \qquad \min\ x_1 + \sum_{k=1}^{q} \lambda_k^{(i)} h_k(\underline{x}) + K^{(i)} \sum_{k=1}^{q} h_k^{\,2}(\underline{x}) \qquad (3.9)$$

$$\text{s.t.} \qquad \underline{x} \in X$$

where $X = \{\underline{x} : \underline{x} \in R^m;\ g_k(\underline{x}) \leq 1,\ k = 1, 2, \ldots, p;\ 0 < \underline{x}^L \leq \underline{x} \leq \underline{x}^U\}$

It can be seen that the minimization problem (3.9) is an inequality-constrained signomial program. If a sufficiently large constant is added to the objective function, (3.9) can be easily cast into the standard format of (2.21) for solution by the Avriel-Williams algorithm.

3.4.3 Statement of the Algorithm

The algorithm that has just been developed for solving signomial programs with equality constraints may now be formally stated.

Given the problem (SPE) posed in (3.1) a solution can be obtained iteratively by the following algorithm:

Algorithm 3.2

Step 1 Set i = 1. Assume that $\underline{\lambda}^{(i)}$ and $K^{(i)}$ are specified. Obtain a point $\underline{x}^{(o)}$ that is feasible for problem $(\pi_1^{(i)})$ as stated in (3.9).

Step 2 Solve $(\pi_1^{(i)})$ by the Avriel-Williams algorithm with $\underline{x}^{(i-1)}$ as the starting point. Let the solution be $\underline{x}^{(i)}$.

Step 3 If the equality constraints $h_k(\underline{x}^{(i)}) = 0$, k = 1, 2, ..., q, are satisfied, stop. Otherwise, i = i + 1. Update $\underline{\lambda}^{(i)}$ and/or $K^{(i)}$. Go to Step 2. The parameters $\underline{\lambda}$ and K may be updated in various ways, and the updating rules will be discussed later.

The operation of the proposed algorithm may be illustrated by considering the following simple signomial program with both equality and inequality constraints.

$$\begin{aligned} \min \quad & x_1^{\,2} \\ \text{s.t.} \quad & 3 - x_1 - x_2 \leq 1 \\ & x_2 = 1 \\ & 0 < x_1, \; x_2 \end{aligned} \tag{3.10}$$

Assuming that K is sufficiently large and fixed, the kth subproblem is

$$\begin{aligned} \min \quad & x_1^{\,2} + (\lambda^{(k)} - 2K)\, x_2 + K x_2^{\,2} \\ \text{s.t.} \quad & 3 - x_1 - x_2 \leq 1 \\ & 0 < x_1, \; x_2 \end{aligned} \tag{3.11}$$

From the Kuhn-Tucker conditions of the subproblem, the solution to (3.11) is

$$x_1^{(k)} = \frac{2K + \lambda^{(k)}}{2(K+1)}, \qquad x_2^{(k)} = \frac{2K + 3 - \lambda^{(k)}}{2(K+1)} \tag{3.12}$$

One effective updating rule for $\underline{\lambda}$ is the Hestenes-Powell formula [25], [26] given as $\underline{\lambda}^{(k+1)} = \underline{\lambda}^{(k)} + 2K\underline{h}(\underline{x}^{(k)})$. Using this formula yields

$$\begin{aligned} \lambda^{(k+1)} &= \left(\frac{1}{K+1}\right) \lambda^{(k)} + \frac{2K}{K+1} \\ &\triangleq a\, \lambda^{(k)} + b, \qquad a, b > 0 \end{aligned} \tag{3.13}$$

From the theory of difference equations, the closed-form solution to (3.13) is

$$\lambda^{(k)} = c\, a^k + b\left(\frac{1 - a^{k+1}}{1 - a}\right) \tag{3.14}$$

for some arbitrary constant c. Set $\lambda^{(o)} = 0$. Then c = -b and

$$\lambda^{(k)} = 2[1 - \frac{1}{(K+1)}k] \tag{3.15}$$

It follows that

$$x_1^{(k)} = 1 - 1/(K+1)^{k+1}, \qquad x_2^{(k)} = 1 + 1/(K+1)^{k+1} \tag{3.16}$$

As $k \to \infty$, convergence to the unique solution ($x_1 = 1$, $x_2 = 1$) occurs for all $K > 0$. Hence, K need not approach infinity. However, the rate of convergence increases rapidly with modest increase in K. This fact can be verified by substituting values of K into (3.15) - (3.16). If the objective function is just x_1, it can be shown that there is a one-step convergence independent of the initial value of $\underline{\lambda}$. The general convergence properties of Algorithm 3.2 are discussed in the next section.

3.5 Convergence Considerations

The proposed algorithm is a specialization of the method of multipliers to signomial programs with equality constraints. The convergence of the algorithm is therefore determined by the convergence of the method of multipliers. In this section, the theoretical results pertaining to the convergence properties of the method of multipliers are stated. The theorems quoted here are due to Polyak and Tret'yakov [34], although recently Bertsekas [35] independently proved similar results.

The problem of interest is

$$\begin{aligned} &\min \quad f_o(\underline{x}) \\ &\text{s.t.} \quad h_k(\underline{x}) = 0, \qquad k = 1, 2, \ldots, q \\ &\qquad \underline{x} \in X \subseteq R^m \end{aligned} \tag{3.17}$$

where f_o and h_k are functions mapping X to R. Note that in (3.17) $\underline{x}$ is restricted to X. The augmented Lagrangian $\ell(\underline{x}, \underline{\lambda}, K)$ can be constructed as

$$\ell(\underline{x}, \underline{\lambda}, K) = f_o(x) + \underline{\lambda}' \underline{h}(\underline{x}) + K||\underline{h}(\underline{x})||^2 \tag{3.18}$$

where $\underline{h}(\underline{x}) = [h_1(\underline{x}), \cdots, h_k(\underline{x})]'$ and $||\ .\ ||$ is the Euclidean norm. Then a primal-dual pair can be defined as follows

$$\text{(P)} \qquad \min \quad \ell(\underline{x}, \underline{\lambda}, K) \qquad (3.19)$$
$$\text{s.t.} \quad \underline{x} \in X$$

and

$$\text{(D)} \qquad \max \quad d_K(\underline{\lambda}) \qquad (3.20)$$
$$\text{s.t.} \quad \underline{\lambda} \in R^q$$

where the dual functional is defined as

$$d_K(\underline{\lambda}) = \min \quad \ell(\underline{x}, \underline{\lambda}, K) \qquad (3.21)$$
$$\text{s.t.} \quad \underline{x} \in X$$

Suppose that at $(\underline{x}^*, \underline{\lambda}^*)$, $\underline{x}^*$ is a local minimum of (3.17) and together with $\underline{\lambda}^*$ satisfies the second-order sufficient optimality conditions. Also assume that the Hessian matrices $\nabla^2 f(\underline{x})$ and $\nabla^2 h_k(\underline{x})$ are Lipschitz in the neighborhood of $\underline{x}^*$. Then the following two theorems are valid.

Theorem 3.1

For every $\underline{\lambda}$ in the bounded set $Y = \{\underline{\lambda} : ||\underline{\lambda} - \underline{\lambda}^*|| \leq \rho\}$, there exists a number $K^*(\rho)$ such that for $K \geq K^*(\rho)$, the following are true:

a) (P) has a unique local minimum $\tilde{x}$ in the neighborhood of $\underline{x}^*$;

b) $\ell(\underline{x}, \underline{\lambda}, K)$ is locally convex about $\underline{x}^*$;

c) For some scalar $c_o > 0$,

$$||\tilde{x} - x^*|| \leq c_o ||\underline{\lambda} - \underline{\lambda}^*||/K, \quad \forall\ K > K^* \qquad (3.22)$$

$$||\underline{\lambda} + K\underline{h}(\tilde{x}) - \underline{\lambda}^*|| \leq c_o ||\underline{\lambda} - \underline{\lambda}^*||/K, \quad \forall\ K > K^* \qquad (3.23)$$

Theorem 3.2

For every ρ, there exists a number $K^*(\rho)$ such that if $K \geq K^*$, the dual functional $d_K(\underline{\lambda})$ is twice differentiable and strictly concave with respect to $\underline{\lambda}$. Furthermore, the first and second derivatives of $d_K(\underline{\lambda})$ are given by

$$\nabla d_K(\underline{\lambda}) = \underline{h}(\tilde{\underline{x}}) \tag{3.24}$$

$$\nabla^2 d_K(\underline{\lambda}) = -J(\tilde{\underline{x}}) \, [\nabla^2 \ell(\tilde{\underline{x}}, \underline{\lambda}, K)] \, J(\tilde{\underline{x}}) \tag{3.25}$$

where J is the Jacobian of the equality constraints.

Theorem 3.1 proves not only the existence of local convexity at $\underline{x}^*$, but also the convergence of $\tilde{\underline{x}}$ to $\underline{x}^*$ as $\underline{\lambda} \to \underline{\lambda}^*$. Furthermore, the rate of convergence is estimated by (3.22) - (3.23). If K is finite, a linear rate is in effect. If $K \to \infty$, the rate is superlinear.

Theorem 3.2 gives the differentiability properties of the dual functional $d_K(\underline{\lambda})$. The theorem also bears relevance to the choice of the updating rule for $\underline{\lambda}$. This point is discussed further in Chapter V.

The convergence of the overall algorithm relies on reaching a local minimum of the primal problem (P). This is equivalent to the execution of Step 2 of the proposed algorithm in which the Avriel-Williams algorithm is used. That the latter algorithm yields a local minimum was proved by Avriel and Williams [5] using Zangwill's Convergence Theorem with some regularity condition imposed on the inequality constraints. It follows that subject to the suitable assumptions, the proposed algorithm converges.

3.6 Solution of the Subproblem

The proposed algorithm, as stated in Section 3.4.3, leaves unspecified two major steps: the solution of the subproblem (3.9) and the updating formulas for $\underline{\lambda}$ and K. In this section the discussion is focused on some ways of solving the subproblem. Considerations on how $\underline{\lambda}$ and K should be updated are deferred to Chapter V.

A significant property of the signomial program defined by (3.9) is that the program is likely to have a large number of terms. Specifically, if $g_{p+k}(\underline{x})$ has N_k terms, the subproblem (3.9) has $\sum_{k=1}^{q} N_k(N_k+1)/2$ terms more

than the original problem (3.1). A large number of primal terms means that the degree of difficulty is large, or equivalently that the dual problem's dimensionality is high. For example, a three-variable geometric program with a monomial objective function and two three-term equality constraints would have a signomial subproblem whose dual problem has a dimensionality of 16. In view of the potential difficulty posed by the dual problem's high dimensionality a primal approach has been selected to solve the subproblem (3.9).

Applying a primal algorithm to (3.9) would require either the expansion of the quadratic terms or the avoidance of such expansion by using some indirect method. The former course of action is tedious and entails cumbersome preparation work for data entry. It would indeed be desirable to just have to enter the terms as stated in the equality constraints. In the rest of this section, three techniques which satisfy this requirement and which have been numerically tested are discussed and compared.

The first technique, referred to as Method I, is the condensation scheme of Avriel and Gurovich [36] for algebraic programs (programs involving algebraic functions of signomials). Consider problem $(\pi_1^{(i)})$. Let $z_k \geq |g_{p+k}(\underline{x})| = |P_{p+k}(\underline{x}) - Q_{p+k}(\underline{x})|$.

Then the following problem is equivalent to (3.9)

$$\min\ x_1 + \sum_{k=1}^{q} (\lambda_k^{(i)} - 2K^{(i)})\, g_{p+k}(\underline{x}) + K^{(i)} \sum_{k=1}^{q} z_k^2$$

$$\text{s.t.} \quad \text{(a)} \quad |g_{p+k}(\underline{x})| \leq z_k$$

$$\text{(b)} \quad 0 < z_k, \qquad k = 1, 2, \ldots, q \tag{3.26}$$

$$\text{(c)} \quad \underline{x} \in X$$

Constraint (3.26a) (omitting the index subscript) is equivalent to the following pair of inequalities:

$$P(\underline{x})/G(\underline{x}, z) \triangleq P(\underline{x})/(Q(\underline{x}) + z) \leq 1 \ ,$$
$$Q(\underline{x})/H(\underline{x}, z) \triangleq Q(\underline{x})/(P(\underline{x}) + z) \leq 1 \tag{3.27}$$

Condensing G and H at $(\hat{\underline{x}}, \hat{z})$ gives

$$P(\underline{x})/G(\underline{x}, z) \leq P(\underline{x})/(G(\hat{\underline{x}}, \hat{z}) \prod_{j=1}^{m} (x_j/\hat{x}_j)^{\alpha_j} (z/\hat{z})^{\alpha_{m+1}}) \leq 1$$
$$Q(\underline{x})/H(\underline{x}, z) \leq Q(\underline{x})/(H(\hat{\underline{x}}, \hat{z}) \prod_{j=1}^{m} (x_j/\hat{x}_j)^{\beta_j} (z/\hat{z})^{\beta_{m+1}}) \leq 1 \tag{3.28}$$

where

$$\alpha_j = (1/G(\hat{\underline{x}}, \hat{z})) \left[x_j \, \partial G/\partial x_j\right]_{\underline{x} = \hat{\underline{x}}} \ ,$$
$$\beta_j = (1/H(\hat{\underline{x}}, \hat{z})) \left[x_j \, \partial H/\partial x_j\right]_{\underline{x} = \hat{\underline{x}}} \ , \tag{3.29}$$
$$j = 1, 2, \ldots, m$$
$$\alpha_{m+1} = \hat{z}/G(\hat{\underline{x}}, \hat{z}); \qquad \beta_{m+1} = \hat{z}/H(\hat{\underline{x}}, \hat{z})$$

Since α_{m+1} and β_{m+1} are both positive, the right inequalities of (3.28) can be written as the following single inequality:

$$\max \{E^{2/\alpha_{m+1}}, F^{2/\beta_{m+1}}\} \leq z^2 \tag{3.30}$$

where

$$E \triangleq \frac{\hat{z}^{\alpha_{m+1}} \; P(\underline{x})}{G(\hat{\underline{x}}, \hat{z}) \prod_{j=1}^{m} (x_j/\hat{x}_j)^{\alpha_j}} \tag{3.31}$$

$$F \triangleq \frac{\hat{z}^{\beta_{m+1}} \; Q(\underline{x})}{H(\hat{\underline{x}}, \hat{z}) \prod_{j=1}^{m} (x_j/x_j)^{\beta_j}} \tag{3.32}$$

From (3.28), satisfying (3.30) implies that (3.26a) is satisfied. Furthermore, (3.30) has to be tight at the minimum. Hence, (3.26) is solved by solving a sequence of problems each of the form

$$\min \; \omega_0$$

$$\text{s.t. (a)} \quad \omega_1^2 + \sum_{k=1}^{q} \max \{E_k^{2/\alpha_{k,m+1}}, \; F_k^{2/\beta_{k,m+1}}\} \leq \omega_0$$

$$\text{(b)} \quad x_1 + C^{(i)} + \sum_{k=1}^{q} (\lambda_k^{(i)} - 2K^{(i)}) \; g_{p+k}(\underline{x}) \leq K^{(i)} \; \omega_1^2 \tag{3.33}$$

$$\text{(c)} \quad \underline{x} \in X, \quad \omega_1 > 0, \quad \omega_0 > 0$$

In (3.33) ω_o and ω_1 are positive auxiliary variables satisfying (3.33a) and (3.33b). They are introduced to put the subproblem into the standard format and to eliminate the parameters $\underline{\lambda}^{(i)}$ and $K^{(i)}$ from constraint (3.33a). $C^{(i)}$ is a sufficiently positive constant to insure that the left side of (3.33b) is positive.

The minimization problem posed in (3.33) is a piecewise "pseudo" signomial program. One way to solve it is as follows: (i) Use the scheme proposed in [36] to condense (3.33a) to a monomial, (ii) Simultaneously condense the signomials in (3.33b) and (3.33c) to posynomials; (iii)Solve the resulting geometric program, (iv) Repeat steps (i) - (iii) until some convergence criterion is satisfied.

The advantage of this technique is that the size of the original problem remains the same. A major weakness is the numerical difficulty that can result from either α_{m+1} or β_{m+1} being very small. When this

condition occurs, exponent overflow or gross clipping errors destroy the convergence of the iterative solution.

The second method, named Method II, exploits the quadratic character of the penalty terms in the subproblem's objective function. The subproblem (3.9) is recast as follows:

$$(\pi_2) \qquad \min \omega_0$$

$$\text{s.t. (a)} \quad \omega_1^2 + \sum_{k=1}^{q} g_{p+k}^2(\underline{x}) \leq \omega_0$$

$$\text{(b)} \quad x_1 + C + \sum_{k=1}^{q} (\lambda_k - 2K)\, g_{p+k}(\underline{x}) \leq K\omega_1^2 \tag{3.34}$$

$$\text{(c)} \quad \underline{x} \in X, \quad \omega_1 > 0, \quad \omega_0 > 0$$

where C is large enough to make the left side of (3.34b) positive. Since a primal approach to solving problem (π_2) has been adopted, the positive and negative terms of all the signomial inequality constraints must be distinguished. Such a distinction among the terms in (3.34b) and (3.34c) is straightforward. In (3.34a) because of the quadratic exponents, the separation of positive and negative terms can also be easily carried out by replacing $g_{p+k}(\underline{x})$ with $P_{p+k}(\underline{x}) - Q_{p+k}(\underline{x})$ and expanding the squares. Then the constraint (3.12a) can be written as

$$\omega_1^2 + \sum_{k=1}^{q} [P_{p+k}^2(\underline{x}) + Q_{p+k}^2(\underline{x})] - 2\sum_{k=1}^{q} P_{p+k}(\underline{x})Q_{p+k}(\underline{x}) \leq \omega_0 \tag{3.35}$$

Being posynomials, $P_{p+k}(\underline{x})$ and $Q_{p+k}(\underline{x})$ are positive for $\underline{x} \in X$. Hence, ω_1^2 and the first summation constitute the positive terms while the rest are the negative terms.

The specific primal algorithm selected to solve (3.34) is the

Avriel-Williams algorithm explained in Section 2.6. The algorithm requires that each signomial inequality constraint of (3.34) be rewritten as a ratio of positive terms to one plus the negative terms, with the ratio bounded from above by unity. The solution to problem (π_2) (3.34) is then obtained by solving a sequence of geometric programs each of which is obtained by condensing the denominator of each ratio constraint at a feasible point of (π_2).

The condensation of the ratio version of (3.35) merits special attention. Written as a ratio of posynomials, (3.35) becomes

$$\frac{\omega_1^2 + \sum_{k=1}^{q} [P_{p+k}^2(\underline{x}) + Q_{p+k}^2(\underline{x})]}{\omega_0 + 2\sum_{k=1}^{q} P_{p+k}(\underline{x})\, Q_{p+k}(\underline{x})} \leq 1 \qquad (3.36)$$

If condensation is viewed as an application of the arithmetic-geometric inequality, then the multiplication in each squared or product term has to be carried out. But this is the precise procedure that is unwanted. However, if condensation is interpreted as exponentiation following the first-order Taylor approximation of the logarithm of a posynomial, then either the numerator or the denominator of (3.36) can be condensed directly in the manner of (2.7) - (2.8).

The geometric programs approximating the problem (π_2) have been solved by Dembo's cutting-plane method [12], [16]. The numerical experience acquired suggests that convergence is attained rather slowly. A closer analysis of the results shows that a good part of the computing time is spent on approximating the constraint (3.34a). This discovery leads to the adoption of the third method detailed in Chapter V following the discussion of an intimately related algorithm in the next chapter. Method III differs from Method II in the following ways:

1) The objective function in (3.9) is not incorporated into the constraint set, as it is done in (3.34)
2) Only the set X is convexified by monomial condensation.
3) The nonconvex objective function of (3.9), not its approximation, is directly minimized over the convex approximant of X.

The numerical results obtained with Method III are detailed in Chapter VI. As a whole, the results indicate that Method III is preferred.

3.7 Conclusion

In this chapter an algorithm has been synthesized to solve signomial programs with equality constraints. The development of the algorithm has been shaped by three main concepts: the method of multipliers viewed as a primal-dual algorithm, partial duality, and the retention of the structure of signomial programs for convexification. The algorithm replaces the original problem with a sequence of inequality-constrained signomial programs. However, if each subproblem is written out explicitly in the standard format of a signomial program, inconvenience in data preparation and computational difficulty due to large problem size are both very likely. To circumvent these problems, three indirect methods based on monomial condensation have been numerically explored, and one has been singled out as computationally promising. These methods require no or limited increase in variables and constraints, and the effort for data entry is the necessary minimum. Finally, the convergence of the proposed algorithm is related to that of the method of multipliers and the Avriel-Williams algorithm.

The proposed algorithm, as stated in Section 3.4.3, serves as a broad framework for specific methods. The conditions for the algorithm's convergence permit wide latitude of arbitrariness in casting the algorithm

into a more concrete form for implementation in a computer. One major arbitrary area, the algorithm for solving the subproblem (3.9), has been discussed and will be continued in Chapter V. The other important unspecified step in the proposed algorithm is the updating rules of $\underline{\lambda}$ and K. The discussion of these rules is deferred to Chapter V. In both parts, the selected procedures not only must satisfy the basic conditions for convergence, but also should exhibit experimental efficiency in solving problems. In Section 3.6, the computational performance of the three indirect methods has been assessed on the basis of numerical experience acquired in the process of selecting a suitable method. In the subsequent chapters, further computational considerations are discussed and more numerical experimental results are offered.

IV. A PROPOSED ALGORITHM FOR INEQUALITY-CONSTRAINED SIGNOMIAL PROGRAMS

4.1 Introduction

The Method II discussed in Section 3.6 for solving the subproblem (3.9) is effectively the application of the Avriel-Williams algorithm to transform a nonconvex signomial program to a sequence of convex geometric programs via posynomial condensation. Each of the convex geometric programs, in turn, is solved by Dembo's [12], [16] cutting plane algorithm. In this method, the signomial objective function is treated as a constraint through the use of an additional variable. The computational experience associated with Method II indicated that convergence is slow, that many cuts may be required, and that most of required cuts are to satisfy the constraint derived from the objective function. A suspected reason why the objective function turned constraint requires so many linearizations is the two-stage approximation of its many terms with a single monomial term. To overcome this probable block on achieving acceptable convergence, an algorithm is proposed in which only the nonconvex feasible set of an inequality-constrained signomial program is convexified and then outer linearized. The nonconvex objective function remains unchanged in the whole iterative process.

As in any approximation-based strategy, the approximating problem is useful only if it can be handled with ease. In the problem of interest, an effective algorithm for minimizing a nonconvex function subject to linear constraints must be selected. A promising candidate in this regard is the reduced gradient method first suggested by Wolfe [37]. Its generalized version [38] for nonlinear constraints was ranked as among the best of the techniques tested in Colville's study [39]. The reduced gradient method is a hybrid of

simplex-type algorithms and those that are gradient-based. Thus, if the solution point is an interior point, the method zeroes in towards the solution point with the characteristic rate of the steepest descent method. This feature is a potential aid to accelerate the solution of the approximating convex program either by reaching an interior solution without too many cuts, or by ending at a point at which a deep cut can be made.

The algorithm proposed in this chapter is, in principle, applicable to all inequality-constrained signomial programs. For this reason the problem solved by the algorithm is formulated in Section 4.2 as a general signomial program. In Section 4.3, the algorithm is developed and compared with related algorithms. Important computational considerations are elaborated in Section 4.4. Section 4.5 reports the computational experience with the algorithm. In the last two sections, the inclusion of simple upper-bound constraints and the extension to non-signomial objective functions are considered.

4.2 Problem Statement

The problem to be solved is

$$(T) \qquad \min\ g_o(\underline{x}) \quad \text{s.t. } \underline{x} \in F \qquad (4.1)$$

where $F = \{\underline{x}:\ \underline{x} \in R^m;\ 0 < \underline{x}^L \leq \underline{x};\ g_k(\underline{x}) \leq 1,\ k = 1, 2, \ldots, p\}$ and $g_o(\underline{x})$ are signomials. Note that F differs from X of (3.1) only in that there is no simple upper bound constraint on $\underline{x}$ in F. Consideration of such type of constraint is taken up in Section 4.6.

A primal approach is adopted to solve (4.1). Such an approach is favorable if the problem has a high degree of difficulty and/or slack primal

constraints. The algorithm to be discussed is particularly useful if the objective function $g_o(\underline{x})$ possesses so many terms that approximating it with a monomial is not preferred.

4.3 The Proposed Algorithm

4.3.1 Preview

Consider the problem (T) in (4.1). It is a nonconvex program whose nonconvex feasible set F can be convexified by posynomial condensation. In the Avriel-Williams algorithm, the objective function $g_o(\underline{x})$ is first transformed into a constraint and added to the set F before convexification. The whole problem is thus replaced with an approximating convex program. In the algorithm proposed in this section, the set F is not augmented with $g_o(\underline{x})$. Hence, only F is convexified while $g_o(\underline{x})$ remains unchanged. The original problem in this case is approximated by a subproblem, say (ST), involving the minimization of the original nonconvex cost over an approximating convex feasible set. The subproblem is then solved by a combined reduced gradient and cutting plane (CRGCP) algorithm to be detailed later in the chapter.

The proposed method has two major steps: the convexification of the set F and the solution of the resultant subproblem (ST) by a CRGCP algorithm. The discussion of how F can be convexified by posynomial condensation may be found in Section 2.6, and is not dealt with here. The reduced gradient method as proposed by Wolfe [37] is stated in the next section for completeness and for explaining subsequent algebraic manipulations and computational considerations.

4.3.2 The Reduced Gradient Method

Assume that the problem to be solved is

$$\begin{aligned} &\min \quad M(\underline{x}) \\ &\text{s.t.} \quad A\underline{x} = \underline{b},\ \underline{x} \geq 0 \end{aligned} \tag{4.2}$$

where A is an m x n matrix and m < n. Let $\underline{x}$ be partitioned into basic (dependent) variables $\underline{x}_B = (x_{B_1}, \ldots, x_{B_m})'$ and nonbasic (independent) variables $\underline{x}_N = (x_{N_1}, \ldots, x_{N_{n-m}})'$. Without any loss of generality, a nonsingular square m x m matrix B can be defined as that consisting of the columns of A associated with $\underline{x}_B$. Then

$$B\underline{x}_B + C\underline{x}_N = \underline{b} \tag{4.3}$$

Or

$$\underline{x}_B = -B^{-1}C\underline{x}_N + B^{-1}\underline{b}$$

where C are the columns of A after those belonging to B have been deleted. The reduced gradient, i.e., $dM(\underline{x})/d\underline{x}_N$, is

$$dM(\underline{x})/d\underline{x}_N = \nabla_{\underline{x}_N} M(\underline{x}) - C'(B')^{-1} \nabla_{\underline{x}_B} M(\underline{x}) \tag{4.4}$$

The strategy of the reduced gradient method is to decrease $M(\underline{x})$ by maneuvering only in the subspace of $\underline{x}_N$ subject to the simple constraint $\underline{x}_N \geq 0$. The equality constraint $A\underline{x} = \underline{b}$ is satisfied by (4.3) and the positivity of $\underline{x}_B$ is assured by suitable restriction on the step size in $\underline{x}_N$. Furthermore, whenever a basic variable vanishes, a simplex pivot operation is done to interchange the vanishing basic variable with a nonzero nonbasic variable. Algorithm 4.1 states formally the reduced gradient method for solving (4.2).

Algorithm 4.1 (Wolfe's Reduced Gradient Mehtod)

Step 1: Set k = 0; $\underline{x}^0$ satisfies $A\underline{x}^0 = \underline{b}$.

Step 2: Compute $\nabla M(\underline{x}^k)$ and

$$\underline{v}^k = \nabla_{\underline{x}_N} M(\underline{x}^k) + C^T\underline{u}^k$$

where $\underline{u}^k = -(B^T)^{-1} \nabla_{\underline{x}_B} M(\underline{x}^k)$

Step 3: Define the descent direction $\underline{s}^k$ as

i) $s^k_{Ni} = 0$ if $x^k_{Ni} = 0$ and $v^k_i > 0$

Else, $s_{Ni}^k = -v_i^k$

ii) $\underline{s}_B^k = -B^{-1}C\underline{s}_N^k$

Step 4: If $||s_N^k|| < \varepsilon$, $\varepsilon > 0$ and small, stop. Else go to Step 5.

Step 5: Find i) $\lambda_M^k = \min \{-x_i^k/s_i^k : s_i^k < 0, \quad i = 1, 2, \ldots, n\}$

ii) λ_*^k minimizing $M(\underline{x}^k + \lambda\underline{s}^k)$, $\lambda \in (0, \lambda_M^k]$

Step 6: Set $\lambda^k = \lambda_*^k$

$$\underline{x}^{k+1} = \underline{x}^k + \lambda^k\underline{s}^k$$

If $\underline{x}_B^{k+1} > 0$, set k = k+1 and go to Step 2.

Else go to Step 7.

Step 7: Perform a pivot operation on B^{-1} to interchange the vanishing basic variable with a nonvanishing nonbasic variable. Update the indices of the basic and nonbasic variables. Set k = k+1 and go to Step 2.

From the foregoing summary of the reduced gradient method, four important requirements have to be met if Algorithm 4.1 is to be applied. They are 1) the constraint structure of (4.1) has to be satisfied; 2) an initial feasible point has to be available; 3) an initial basis matrix and its inverse has to be determined; 4) the gradient of the objective function can be calculated. The next section gives the necessary algebraic manipulations for casting the outer-linearized version of the subproblem (ST) into the form of (4.1). Consideration of the other requirements is found in Section 4.4.

4.3.3 Algorithm Development

Consider the problem (4.1). Let $g_k(\underline{x}) = P_k(\underline{x}) - Q_k(\underline{x})$, k = 1, 2, ..., p, where P_k and Q_k are posynomials. The feasible set F can be rewritten as

$$F = \{\underline{x} : \underline{x} \in R^m;\ 0 < \underline{x}^L \leq \underline{x};\ P_k(\underline{x})/(1 + Q_k(\underline{x})) \leq 1,\ k = 1, 2, \ldots, p\} \quad (4.5)$$

Assuming that a point $\underline{\hat{x}} \in F$ is known, the set F can be approximated with a smaller set $\overline{F}$ by applying posynomial condensation (see Section 2.2) to the denominator of $Q_k(\underline{x})$ at $\underline{\hat{x}}$ to yield the monomial $\overline{Q}_k(x,\underline{\hat{x}})$. The set $\overline{F}$ is then defined as

$$\overline{F} = \{\underline{x} : \underline{x} \in R^m;\ 0 < \underline{x}^L \leq \underline{x};\ P_k(\underline{x})/\overline{Q}_k(\underline{x}, \underline{\hat{x}}) \leq 1,\ k = 1, 2, \ldots, p\} \quad (4.6)$$

The approximate problem (ST) is

$$(ST) \qquad \min g_o(\underline{x}) \qquad \text{s.t. } \underline{x} \in \overline{F} \qquad (4.7)$$

Both problems (T) and (ST) are nonconvex programs because F and $\overline{F}$ are generally nonconvex for $\underline{x} \geq \underline{x}^L > 0$, $\underline{x} \in R^m$. But if the logarithmic transformation $z_i = \ln x_i$ is used to replace $\underline{x}$ with $\underline{z}$, the problem equivalent to (T) and (ST), respectively, may be stated as follows:

$$(T)_Z \qquad \min g_o(\underline{z}) \qquad \text{s.t. } \underline{z} \in Z \qquad (4.8)$$

where

$$Z = \{\underline{z} : \underline{z} \in R^m;\ \underline{z}^L \leq \underline{z};\ P_k(\underline{z})/(1 + Q_k(\underline{z})) \leq 1,\ k = 1, 2, \ldots, p\} \quad (4.9)$$

and

$$(ST)_Z \qquad \min g_o(\underline{z}) \qquad \text{s.t. } \underline{z} \in \overline{Z} \qquad (4.10)$$

where

$$\overline{Z} = \{\underline{z} : \underline{z} \in R^m;\ \underline{z}^L \leq \underline{z},\ P_k(\underline{z})/\overline{Q}_k(\underline{z}, \underline{\hat{z}}) \leq 1,\ k = 1, 2, \ldots, p\} \quad (4.11)$$

Note that now problem $(ST)_Z$ is convex. Another formulation of $(ST)_Z$ equivalent to (4.10) - (4.11) is

$$(ST)_{ZL} \qquad \min g_o(\underline{z}) \qquad \text{s.t. } \underline{z} \in \overline{Z}_L \qquad (4.12)$$

where

$$\overline{Z}_L = \{\underline{z} : \underline{z} \in R^m;\ \underline{z}^L \leq \underline{z},\ G_k(\underline{z}, \underline{\hat{z}}) \leq 0,\ k = 1, 2, \ldots, p\} \quad (4.13)$$

and

$$G_k(\underline{z}, \underline{\hat{z}}) = \ln(P_k(\underline{z})/\overline{Q}_k(\underline{z}, \underline{\hat{z}})) \quad (4.14)$$

In (4.12) - (4.14), the following statements are true:

1. $G_k(\underline{z},\underline{\hat{z}})$ is a convex function $\forall\ \underline{z} \in R^m$, $\underline{z}^L \leq \underline{z}$

2. $\overline{Z}_L = \overline{Z}$

3. $\overline{Z}_L$ is a convex set.

In Problem $(ST)_{ZL}$, a nonconvex function is minimized over a convex set. To solve this problem, a combined reduced gradient and cutting plane algorithm is used. The algorithm's strategy requires 1) the outer linearization of the set $\overline{Z}_L$, 2) the casting of the problem formulation into that of (4.2), 3) the use of Algorithm 4.1, 4) if required, the improvement of the approximation of $\overline{Z}_L$ with cutting planes.

To outer linearize $\overline{Z}_L$, replace the convex function $G_k(\underline{z},\underline{\hat{z}})$ with its first-order Taylor approximation $G_k(\underline{a},\underline{\hat{z}}) + (\underline{z} - \underline{a})' \nabla G_k(\underline{a},\underline{\hat{z}})$ for some $\underline{a} \in R^m$ satisfying $\underline{a} \geq \underline{z}^L$. Define the resultant polyhedron $\hat{Z}_L \subset R^m$ as

$$\hat{Z}_L = \{\underline{z} : \underline{z} \in R^m;\ \underline{z}^L \leq \underline{z},\ G_k(\underline{a},\underline{\hat{z}}) + (\underline{z}-\underline{a})' \nabla G_k(\underline{a},\underline{\hat{z}}) \leq 0,\ k=1,\ldots,p\} \quad (4.15)$$

Because of the convexity of $G_k(\underline{z},\underline{\hat{z}})$, $\overline{Z}_L \subseteq \hat{Z}_L$. Next let $\underline{\zeta} = \underline{z} - \underline{z}^L$ and $\underline{\zeta}_s = [\zeta_{s1}, \ldots, \zeta_{sp}]$ be the vector of slack variables. Then define the hyperplane $H \subset R^{m+p}$ as

$$H = \{(\underline{\zeta}', \underline{\zeta}_s') : (\underline{\zeta}', \underline{\zeta}_s') \in R^{m+p};\ 0 \leq (\underline{\zeta}', \underline{\zeta}_s');\ [J : I](\underline{\zeta}', \underline{\zeta}_s') = \underline{b}\} \quad (4.16)$$

where J is the p x m Jacobian matrix $[\partial G_k(\underline{a},\underline{\hat{z}})/\partial z_i]$, I is the pth-order identity matrix, and $\underline{b}$ is the p x 1 vector $-[(\underline{z}^L - \underline{a})' \nabla G_k(\underline{a},\underline{\hat{z}}) + G_k(\underline{a},\underline{\hat{z}})]$. The problem suitable for solution by Algorithm 4.1 is

$$\text{(LST)} \qquad \begin{aligned} &\min\ g_o(\underline{\zeta}) \\ &\text{s.t. } (\underline{\zeta}, \underline{\zeta}_s) \in H \end{aligned} \qquad (4.17)$$

Let the solution to (LST) be $\underline{\zeta}^*$ (or correspondingly, some $\underline{z}^*$). If $\underline{z}^* \in \overline{Z}_L$, proceed to the next convex approximation of Z. Otherwise, add a cutting plane to $\hat{Z}_L$. The cutting plane is obtained by linearizing the most violated constraint of $\overline{Z}_L$ at $\underline{z}^*$.

4.3.4 The Algorithm

In this section the proposed algorithm for solving inequality-constrained signomial programs of the type (4.1) is formally stated as Algorithm 4.2.

Algorithm 4.2

Iteration 0: Set k = 1. Assume that a point $\hat{z}^{(k)} \in Z$ is known

Iteration k: Step 1: Condense at $\hat{z}^{(k)}$ to produce problem $(ST)_Z$

Step 2: Set i = 1. Set $a^{(k)} = \hat{z}^{(k)}$

Step 3: Linearize $\overline{Z}_L$ at $a^{(k)}$ and set up problem (LST).

Step 4: Solve (LST) by Algorithm 4.1. Let the solution (in the z - domain) be $\tilde{z}^{(i)}$.

Step 5: If $\tilde{z}^{(i)} \in Z$ go to Step 7. Otherwise determine the most violated constraint, linearize it at $\tilde{z}^{(i)}$, and use the linearization as the next cutting plane. Proceed to the next step.

Step 6: Determine a new basis and a new feasible point for problem (LST). Set i = i+1 and go to Step 4.

Step 7: If $\tilde{z}^{(i)}$ satisfies the termination criterion, stop. Else, set k = k+1, $\hat{z}^{(k)} = \tilde{z}^{(i)}$, and go to Step 1.

Step 6 has to be included because before Algorithm 4.1 can be applied a feasible point and a basis must be available. How Step 6 is carried out is among the practical computational considerations taken up in Section 4.4

4.3.5 The Algorithm in Perspective

Algorithm 4.2 has been developed to solve signomial programs characterized by a high degree of difficulty and by an objective function with many terms. The main basis of the algorithm is that a set defined by signomial inequalities

can be readily approximated by a convex set via posynomial condensation. In this regard, the algorithm is similar to the Avriel-Williams algorithm. There is, however, a major difference between the two. Unlike the Avriel-Williams algorithm, the approach taken here does not incorporate the objective function into the constraint set. It is hoped that by minimizing the original generally nonconvex objective function over the approximating convex set, convergence towards the original solution can be attained faster.

The decision to approximate only the feasible set and leave the objective function untouched means that an efficient method for minimizing a nonconvex function over a convex set must be employed. A promising method is the novel algorithm proposed in this chapter. The algorithm merges the ability of Kelly's [40] cutting plane method to handle convex constraints with the efficiency of the reduced gradient method to minimize a general nonlinear function subject to linear constraints.

To tackle the issue of feasibility, the algorithm exploits the convexity property by adopting the strategy of relaxing the constraints by outer linearization and progressively tightening the constraints with cutting planes. The reduction of the nonconvex cost over the set defined by the relaxed constraint (a convex polyhedron) is achieved by the reduced gradient method. Thus the algorithm translates into a sequence of programs each solved by the reduced gradient method. In contrast, Kelley's cutting plane method for convex programs requires the solution of a sequence of linear programs. The likely benefit of Algorithm 4.2 is that because maneuvers into the interior of the feasible set are allowed, an interior solution point can be reached sooner, or if the solution is a boundary point, feasibility can be achieved with deeper cuts.

Another interesting comparison that can be made here is with Abadie and Carpentier's [38] generalized reduced gradient (GRG) algorithm for handling

nonlinear constraints. Because GRG is designed for general nonlinear constraints, feasibility is maintained in each iteration by a restoration phase in which a system of nonlinear equations (the constraints) is solved by Newton's method. In the problem of interest in this chapter, because of the convexity of the feasible set, the need to maintain feasibility in each iteration is dispensed with and is replaced with a relaxation strategy [41]. In GRG, the inverse of the Jacobian of the constraints with respect to the basic variables has to be either exactly or approximately updated at each point or when the basic variables are changed. This step can become a heavy computational load. In the suggested combined approach, because the relaxed constraints are just linear equations, the basis' inverse is constant and is easily updated by simple pivot operations when the basis is changed.

4.4 Considerations for Implementation

Together, Sections 4.3.2 and 4.3.4 spell out clearly the major steps of the proposed technique for solving signomial programs. However, if the proposed method is to be implemented in a computer, some important practical questions need to be answered. How can the point $\underline{\hat{z}}^{(i)} \in Z$ be found? How can the initial basis and the initial feasible point be derived before a call to the reduced gradient method (Algorithm 4.1)? How can the basis' inverse be updated? How should the one-dimensional minimization required in Step 5 of Algorithm 4.1 be carried out? What are the termination criteria and tolerances? These are the questions to which this section is addressed.

4.4.1 Finding a feasible Point of a Signomial Program

A requisite for approximating the set F defined in (4.1) with a smaller disguised convex set $\overline{F}$ is a point $\underline{\hat{x}} \in F$. To find $\underline{\hat{x}}$, Dembo's [16] Phase I method is used. In this method, a sequence of programs each of the

form

(PHSI) $$\min \prod_{k=1}^{p} \omega_k$$

$$\text{s.t. } P_k(\underline{x})/(1 + Q_k(\underline{x})) \leq \omega_k \tag{4.18}$$

$$0 < \underline{x}^L \leq \underline{x}, \quad 1 \leq \omega_k, \; k = 1, \ldots, p$$

is solved. Each problem (PHSI) is solved by Avriel-William's algorithm in conjunction with Kelley's cutting plane method. (See [12], [16]). The sequence is terminated either if $w_k = 1$, $\forall$ k, or some $w_k \neq 1$ when the optimal solution to (PHSI) has been obtained.

4.4.2 Getting an Initial Basic Feasible Solution of Prolbem (LST) - No Cut Added

From (4.16), the problem (LST) is defined as

$$\min g_o(\underline{\zeta})$$

$$\text{s.t. (a) } [J: I_p] \left[\frac{\underline{\zeta}}{\underline{\zeta}_s}\right] = \underline{b} \tag{4.19}$$

$$\text{(b) } 0 \leq \underline{\zeta}, \; 0 \leq \underline{\zeta}_s$$

where $\underline{\zeta} = \underline{z} - \underline{z}^L$, $J = [\partial G_k(\underline{a},\hat{\underline{z}})/\partial z_i]$, $\underline{b} = -[(\underline{z}^L - \underline{a})' \nabla G_k(\underline{a},\hat{\underline{z}}) + G_k(\underline{a}, \hat{z})]$, $\underline{z} \in Z$, and $\underline{a} \in R^m$ satisfying $\underline{a} \geq \underline{z}^L$. A convenient initial feasible solution to (4.19) can be obtained by the following lemma.

Lemma 4.1

Assume that $\hat{\underline{z}} \in Z$ and let $\hat{\zeta} = \hat{\underline{z}} - \underline{z}^L$. If $\underline{a} = \hat{\underline{z}}$ and $\hat{\underline{\zeta}}_s = b - J\hat{\underline{\zeta}}$, then the point $(\hat{\underline{\zeta}}, \hat{\underline{\zeta}}_s)$ is a feasible solution of problem (LST).

Proof:

By the hypothesis and Lemma 2.1,

$$G_k(\hat{\underline{z}},\hat{\underline{z}}) = \ln (P_k(\hat{\underline{z}})/\overline{Q}_k(\hat{\underline{z}},\hat{\underline{z}})) = \ln (P_k(\hat{\underline{z}})/(1 + Q_k(\hat{\underline{z}}))), \forall k \tag{4.20}$$

But $\hat{\underline{z}} \in Z$. It follows that

$$\ln(P_k(\hat{\underline{z}})/(1 + Q_k(\hat{\underline{z}}))) \leq 0 \qquad \forall k \tag{4.21}$$

Or for some $\hat{\underline{\zeta}}_s \geq 0$,

$$G_k(\hat{\underline{z}}, \hat{\underline{z}}) + \hat{\underline{\zeta}}_s = 0 \qquad \forall k \tag{4.22}$$

Adding to both sides (4.22) the quantity $J\hat{\underline{\zeta}}$ leads to the desired conclusion. Q.E.D.

While the point $(\hat{\underline{\zeta}}, \hat{\underline{\zeta}}_s)$ is feasible for (4.18), its nonzero basic variables still have to be identified. A basis of (4.19a) has to be specified. Or equivalently, a set of m linearly independent columns of the composite $m \times (m+p)$ matrix [J:I] has to be known. A procedure to identify a set of such columns is explained in the next paragraph.

Let the index sets V and W be defined as

$$V = \{k: \hat{\zeta}_{sk} = 0\} = \{v_1, v_2, \ldots, v_{L_1}\} \tag{4.23}$$

$$W = \{k: \hat{\zeta}_{sk} > 0\} = \{\omega_1, \omega_2, \ldots, \omega_{L_2}\} \tag{4.24}$$

Clearly, $L_1 + L_2 = p$. Reorder the rows of (4.19a) such that the first L_1 rows are indexed by V and the last L_2 rows, by W. Let the reordered equation with $\zeta = \hat{\zeta}$ and $\zeta_s = \hat{\zeta}_s$ be written as

$$\left[\begin{array}{c|c|c} Q & I_{L_1} & 0 \\ \hline S & 0 & I_{L_2} \end{array}\right] \left[\begin{array}{c} \hat{\underline{\zeta}} \\ \hline \hat{\underline{\zeta}}_s \end{array}\right] = \underline{b} \tag{4.25}$$

It is assumed that the components of $\hat{\underline{\zeta}}_s$ have been appropriately recorded. Then a basic solution to (4.25) consists of L_1 variables selected from the nonzero elements of $\hat{\underline{\zeta}}$ and the last L_2 nonzero slack variables. This basic solution is feasible by Lemma 4.1. The L_1 variables from $\hat{\underline{\zeta}}$ must correspond to linearly independent columns selected from $[-\frac{Q}{S}-]$ or simply from Q. An algorithm such as Gaussian elimination used for the determination of a matrix's rank can be used to pick the desired L_1 columns. Let the L_1 linearly independent columns of $[-\frac{Q}{S}-]$ be $[-\frac{\tilde{Q}}{\tilde{S}}-]$ where Q is a $L_1 \times L_1$ matrix and S is a $L_2 \times L_1$ matrix. Then the initial basis B of (4.25) is

$$B = \left[\begin{array}{c|c} \tilde{Q} & 0 \\ \hline \tilde{S} & I_{L_2} \end{array}\right] \tag{4.26}$$

Since $\tilde{Q}$ is nonsingular by construction, the initial inverse of the basis is

$$B^{-1} = \begin{bmatrix} \tilde{Q}^{-1} & 0 \\ -\tilde{S}\,\tilde{Q}^{-1} & I_{L_2} \end{bmatrix} \tag{4.27}$$

Hence, in general, only a smaller matrix $\tilde{Q}$ need to be inverted to obtain the full basis inverse.

To summarize, the following steps are undertaken to obtain an initial basic feasible solution to the problem (LST) with no cut added:

1. Obtain a point $\hat{z} \in Z$ and set $a = \hat{z}$, $\hat{\zeta} = \underline{\hat{z}} - \underline{z}^L$.
2. Generate the matrix J and the vector $\underline{b}$. Set $\underline{\zeta}_s = J\underline{\hat{\zeta}} - \underline{b}$.
3. Reorder J, $\underline{b}$ and $\underline{\zeta}_s$ such that the first L_1 rows are indexed by V while the last L_2 rows are indexed by W.
4. Identify L_1 linearly independent columns in Q. Let these columns form the $L_1 \times L_1$ matrix $\tilde{Q}$.
5. Invert $\tilde{Q}$ and generate B^{-1} according to (4.27).

4.4.3 Getting an Initial Basic Feasible Solution of Problem (LST) - After the Addition of a Cut

At the end of one application of the reduced gradient algorithm (i.e. the completion of Step 4 of Algorithm 4.2), let the solution be $(\tilde{\underline{\zeta}}, \tilde{\underline{\zeta}}_s)$. Suppose that at this solution point, at least one of the constraints of the convex set $\overline{Z}_L$ is violated. Let the most violated constraint's index be r. Then with $\tilde{\underline{z}} = \tilde{\underline{\zeta}} + \underline{z}_L$, $G_r(\tilde{\underline{z}}, \underline{\hat{z}}) > 0$. The cut to be added to the polyhedron $\hat{Z}_L$ is the supporting plane of $G_r(\underline{z}, \underline{\hat{z}})$ at $\underline{z} = \tilde{\underline{z}}$. Expressed in terms of $\underline{\zeta}$, the equation of the new cut is

$$\underline{\beta}'\underline{\zeta} + \gamma = c \tag{4.28}$$

where $\underline{\beta} = \nabla G_r(\tilde{\underline{z}},\hat{\underline{z}})$, $c = -[(\underline{z}^L - \tilde{\underline{z}})' \nabla G_r(\tilde{\underline{z}},\hat{\underline{z}}) + G_r(\tilde{\underline{z}},\hat{\underline{z}})]$ and γ is the new slack variable associated with the new cut. Equation (4.25) becomes

$$\left[\begin{array}{c|c|c} Q & I_{L_1} & 0 \\ \hline S & 0 & I_{L_2} \\ \hline \underline{\beta}' & 0 & 0 \end{array}\right] \left[\begin{array}{c} \underline{\zeta} \\ \hline \underline{\zeta}_s \\ \hline \gamma \end{array}\right] = \left[\begin{array}{c} \underline{b} \\ \hline c \end{array}\right] \tag{4.29}$$

Let $\tilde{\gamma} = c - \underline{\beta}'\hat{\underline{\zeta}}$. Clearly, $(\tilde{\underline{\zeta}},\tilde{\underline{\zeta}}_s, \tilde{\gamma})$ is a basic solution to (4.29). However, it is not feasible since by the assumption that $G_r(\tilde{\underline{z}},\hat{\underline{z}}) > 0$, $\tilde{\gamma} < 0$. A method to drive γ into the feasible set defined by (4.29) and the positivity constraint on $(\tilde{\underline{\zeta}},\tilde{\underline{\zeta}}_s,\tilde{\gamma})$ is to solve the following auxiliary problem (AUX)

$$\text{(AUX)} \qquad \begin{array}{c} \min \ -\gamma \\ \text{s.t.} \ \ (4.29),\ \underline{\zeta} \geq 0,\ \underline{\zeta}_s \geq 0 \end{array} \tag{4.30}$$

As it is formally stated, the problem (AUX) is a linear program in a form amenable to the use of a composite algorithm [42] to obtain an initial basic feasible solution. But because the reduced gradient method allows nonzero nonbasic variables, the point $(\tilde{\underline{\zeta}},\tilde{\underline{\zeta}}_s,\tilde{\gamma})$ cannot be used as the initial basic feasible point for the LP solution of (AUX). Using a completely different starting point would not only require a Phase I operation but also would destroy the continuity of the original iterative process. A very convenient answer to this difficulty is to solve problem (AUX) by a slightly modified version of the reduced gradient method.

The only modifications are the objective function and the definition of the bound λ_M on the step size variable λ. The usual bound is defined as

$$\lambda_M = \min \{-x_i/s_i : s_i < 0, \forall\, i\} \tag{4.31}$$

For solving problem (AUX), (4.31) is changed to the following

$$\lambda_m = \min\ [\min\ \{-x_i/s_i : x_i > 0,\ s_i < 0, \forall i\},\ \{-\gamma/s_\gamma : \gamma < 0,\ s_\gamma > 0\}] \qquad (4.32)$$

where s_γ is the component of the direction vector $\underline{s}$ along the new dimension of γ. The criterion specified by (4.32) ensures no new infeasibility, and eliminates the infeasibility due to $\gamma < 0$ as soon as it is possible. If both λ and s_γ are negative, then the constraints are inconsistent. Once $\gamma \geq 0$, the bound given by (4.31) is used, and the original nonconvex objective function $g_o(\underline{\zeta})$ is in force. The initial basic feasible solution to the problem (AUX) is simply $(\tilde{\underline{\zeta}}, \tilde{\underline{\zeta}}_s, \tilde{\gamma})$. The starting basis' inverse is

$$\tilde{B}^{-1} = \left[\begin{array}{c|c} B^{-1} & 0 \\ \hline -\omega' B^{-1} & 1 \end{array}\right] \qquad (4.33)$$

where B^{-1} is the basis' inverse at the termination of Step 4 in Algorithm 4.2, and $\underline{w}'$ is the vector of the components of β' that are associated with the basic variables in $\tilde{\underline{\zeta}}$. Note that once $\gamma > 0$, $\tilde{B}^{-1}$ serves as the starting inverse for the next execution of Step 4 in Algorithm 4.2. Thus, from a computational viewpoint, using the modified reduced gradient method does not perturb too much the sequence of solution points obtained by Algorithm 4.2 . Furthermore, the additional programming effort required to implement the method is very small since the method is still within the algorithmic framework of the main reduced gradient method.

4.4.4 A Linear Search Method

Like other gradient-based algorithms, the reduced gradient method requires in each of its iterations the solution of the following one-dimensional problem:

$$\begin{aligned} f(\lambda^k) = \min\ f(\lambda) &= \min\ g_o(\underline{\zeta}^k + \lambda \underline{s}^k) \\ \text{s.t.} \quad \lambda &\in (0,\ \lambda_M] \end{aligned} \qquad (4.34)$$

The conceptual algorithm calls for an exact linear search. In practice, however, exact linear searches often consume an exorbitant part of the total computing time of a problem. Generally, approximate linear minimization is found adequate to achieve convergence at a reasonable rate. The inexact approach is adopted in this section.

Inexact linear search methods generally fall under the following categories: direct searches involving function values only, low-order curve fitting requiring both function and/or gradient values, and nonoptimal step methods. The first two types are effectively iterative schemes which produce a sequence converging to the true minimizer of $g(\lambda)$. The nonoptimal step method subscribes to a different viewpoint, namely, that linear search is a step imbedded in a larger algorithm. The convergence of the larger algorithm can be, and is, maintained as long as the cost function is decreased sufficiently in each linear search. Nonoptimal step methods are preferred because they are operationally well defined and are efficient. Examples of nonoptimal step methods are the step size rules of Goldstein [43] and Armijo [44]. Other strategies were outlined by Bard [45].

In the problem (LST) that is of interest, the signomial objective function $g_o(\underline{\zeta})$ is continuous and differentiable over H. Furthermore, the evaluation of the derivative at a trial point requires very little additional effort once the function has been evaluated. This observation suggests that a low-order polynomial fit, such as the cubic fit scheme derived by Davidon [46] is suitable. However, a curve fitting scheme by itself may still require numerous repetitions if the termination criterion is the usual relative change in $f(\lambda^k)$ or in λ^k. An appealing approach is to couple Armijo's test for sufficient descent to the cubic fit method. The strategy is to use Armijo's step size

rule except when it is clear that a relative minimum exists in the interval $(0, \lambda_m]$. This condition is confirmed when $df/d\lambda < 0$ at $\lambda = 0$ and $df/d\lambda > 0$ at λ_m. The proposed linear search method, stated as Algorithm 4.3, is thus a synthesis of the cubic fit method and Armijo's step size rule.

Algorithm 4.3

Step 0: Given $\alpha \in (0,1)$, $\beta \in (0,1)$, $\gamma \in (0,1)$, and $\lambda_o > 0$. Let λ_M be determined by Step 5 of Algorithm 4.1. Set $\lambda_m = \min [\lambda_o, \lambda_M]$ and $\lambda_\ell = 0$. Calculate $f(\lambda_\ell)$ and $f'(\lambda_\ell)$.

Step 1: Set $\hat{\lambda} = \lambda_m$, and evaluate $f(\hat{\lambda})$ and $f'(\hat{\lambda})$.

Step 2: If $f'(\hat{\lambda}) > 0$, go to Step 4. Else go to Step 3.

Step 3: Set $\tilde{\lambda} = \lambda_m$ if $f(\lambda_m) < f(\lambda_\ell)$ and go to Step 5. Else, $\lambda_m = \gamma\lambda_m$ and go to Step 1.

Step 4: Apply Algorithm 4.4 to fit a cubic through $(\lambda_\ell, f(\lambda_\ell))$ and $(\lambda_m, f(\lambda_m))$. Let the minimum estimate be $\tilde{\lambda}$.

Step 5: If $f(\tilde{\lambda}) - f(\lambda_\ell) \leq -\alpha|\tilde{\lambda} - \lambda_\ell|\ |f'(\lambda_\ell)|$, $\lambda_k = \tilde{\lambda}$ stop. Else continue.

Step 6: If $\tilde{\lambda} = \lambda_m$, go to Step 7. Else, set $\lambda_\ell = \tilde{\lambda}$ if $f'(\tilde{\lambda}) < 0$, or $\lambda_m = \tilde{\lambda}$ if $f'(\tilde{\lambda}) > 0$. Go to Step 4.

Step 7: $\lambda_m = \beta\lambda_m$. Go to Step 1.

Algorithm 4.4 (Cubic Fit)

Step 0: Given λ_ℓ, λ_m, $f(\lambda_\ell)$, $f(\lambda_m)$, $f'(\lambda_\ell)$ and $f'(\lambda_m)$. Assume $f'(\lambda_\ell) < 0$ and $f'(\lambda_m) \geq 0$.

Step 1: Calculate $u_1 = f'(\lambda_\ell) + f'(\lambda_m) - 3(f(\lambda_\ell) - f(\lambda_m)) / (\lambda_\ell - \lambda_m)$

$$u_2 = [u_1^2 - f'(\lambda_\ell)\, f'(\lambda_m)]^{1/2}$$

Step 2: $\tilde{\lambda} = \lambda_m - (\lambda_m - \lambda_\ell)\,(f'(\lambda_m) + u_2 - u_1) / (f'(\lambda_m) - f'(\lambda_\ell) + 2u_2)$

Algorithm 4.3 combines the useful features of a cubic fit and Armijo's step size rule. The rule assures convergence by stipulating sufficient descent before a trial point is accepted as the next solution point. But the ith trial point is not always obtained by the relation $\tilde{\lambda}^{(i)} = \beta^i \lambda_m^{(0)}$, as it is the case in Armijo's algorithm. When the situation warrants it, the ability of a cubic fit to rapidly pick a good interior estimate is exploited to produce a trial point.

4.4.5 Optimality Criteria, Feasibility Tolerance, and Updating the Basis' Inverse

The main problem (T) posed in (4.1) is a nonconvex program. Solving (T) numerically with an algorithm generally leads to at best a local minimum at which the Kuhn-Tucker optimality conditions must be satisfied. In practice, unless a Lagrangian-type algorithm is used, testing the first-order necessary optimality conditions is likely not possible since the optimal Lagrange multipliers may not be known. Even if the multipliers are known, checking the Kuhn-Tucker conditions at the end of each iteration may create too large a computational load. The whole question of knowing when optimality is reached becomes a trade-off between reliability and extravagance. For the sake of computational efficiency, simple but imperfect stopping rules for terminating the iterative process have to be adopted. The rule adopted in this chapter is: Terminate if the relative change in $g_o(\underline{\zeta}) \leq \varepsilon$ and the relative change in $\underline{\zeta} \leq \varepsilon$, where ε is a small preassigned positive number. If either g_o or $\underline{\zeta} \to 0$, replace relative change with absolute change. Depending on which quantity is of interest, the rule may be relaxed in two ways. If only g_o is important (e.g. only cost, not the design parameters, is sought), the test on the relative change in $\underline{\zeta}$ may be dropped. Analogously, if only the accuracy of $\underline{\zeta}$ is emphasized, the test on g_o may be avoided. An excellent operational check on the answer obtained

by the previous stopping rules is to perturb the solution and resolve the problem with the perturbed point as the starting point.

Another important issue that must be faced in implementing a conceptual algorithm is the definition of feasibility, or more generally, the definition of inequality and equality. Theoretically, whether a and b are scalars or vectors, the expression $a \leq b$, $a \geq b$, and $a = b$ are unambiguous. However, in the implemented version of an algorithm, because numbers are represented in finite lengths, interpreting inequalities and equalities in their strict mathematical sense could just spawn a myriad of nuisances such as unexpected early program termination, undesired negative quantities, division by zero and other difficulties. For this reason, in the implementation of this thesis' algorithms, inequalities and equalities are replaced with ε-inequalities and ε-equalities. This means that given scalars a and b, and a small $\varepsilon > 0$, $a \leq b + \varepsilon$ replaces $a \leq b$; $a \geq b - \varepsilon$, $a \geq b$; and $|a - b| \leq \varepsilon$ replaces $a = b$. If $\underline{a}$ and $\underline{b}$ are vectors, then $\underline{a} \leq \underline{b} + \varepsilon \Rightarrow a_i \leq b_i + \varepsilon$. Similarly, $\underline{a} \geq \underline{b} - \varepsilon \Rightarrow a_i \geq b_i - \varepsilon$, and $||\underline{a} - \underline{b}|| \leq \varepsilon \Rightarrow \underline{a} = \underline{b}$. Note that the Euclidean norm is used.

The final point considered in this section is the updating of the basis' inverse $B^{-1} = [b_{ij}]$. The procedure followed is very similar to that of the revised simplex method. Let r be the index of the leaving basic variable. The key steps are: (1) Determine a nonzero nonbasic variable, say, ζ_s. Let its associated column be $\underline{a}_s$. (2) Calculate $\underline{y}_s = B^{-1}\underline{a}_s$. If $y_{rs} = 0$, return to (1) to determine another variable. If $y_{rs} = 0$ for all s, the system is ill-defined. (3) Assume that $y_{rs} \neq 0$. Let $\hat{B}^{-1} = [\hat{b}_{ij}]$ be the new inverse. Then

$$\begin{aligned} \hat{b}_{ij} &= b_{ij} - (y_{is}b_{rj})/y_{rs}, \quad i \neq r \\ \hat{b}_{rj} &= b_{rj}/y_{rs}, \qquad\qquad i = r \end{aligned} \tag{4.35}$$

4.5 Computational Experience

4.5.1 Posynomial Representation

A posynomial $P(\underline{x})$ is usually written as

$$P(\underline{x}) = \sum_{i=1}^{n} c_i \prod_{j=1}^{m} x_j^{a_{ij}}, \; c_i > 0 \; \forall i, \; x_j > 0 \; \forall j \tag{4.36}$$

An alternative way of expressing the function P is to map the positive orthant $\underline{x} > 0$ (x-domain) to the real space R^m (z-domain) by the logarithmic transformation $z_j = \ln x_j$. As a function of $\underline{z}$, P can be written as

$$P(\underline{z}) = \sum_{i=1}^{n} c_i \exp\left(\sum_{j=1}^{m} a_{ij} z_j\right), \quad z_j \in R, \forall j \tag{4.37}$$

The right side of (4.37) is defined as the exponential form of $P(\underline{x})$. While the expression in (4.36) is generally nonconvex, the function $P(\underline{z})$ is always a convex function of $\underline{z}$. The hidden convexity of (4.36) over the whole real space is thus unveiled by the logarithmic transformation. Further discussion on convex-transformable nonconvex functions may be found in [47].

The exponential form has more than a formal significance. In this chapter's primal approach to solving signomial programs, the exponential form is favored for two important practical reasons: 1) The problem (LST) solved by the proposed combined reduced gradient-cutting plane algorithm is defined in the z-domain. 2) The CPU time required by the algebraic operations in the exponential form is much less than that of the ordinary form. The second reason can be easily explained by considering the computational effort required to compute (4.36) and (4.37). The first equation requires nm multiplications, nm exponentiations, and nm-1 additions , while the second requires nm+n multiplications, n exponentiations, and nm-1 additions. A comparison between the two shows

that the exponential form requires n multiplications and n(m-1) additions more, but n(m-1) exponentiations less than the ordinary form. Because an exponentiation usually requires much more time than an addition or a multiplication [1], using the exponential form in any repetitive calculation of posynomials should definitely be preferred. It is true that if the initial starting point and the final solution are given in the x-domain, there is the additional overhead of taking m logarithms and m exponentiations. This overhead, however, becomes insignificant when compared to the savings in the iterations. In an attempt to gain some idea on how much saving can be obtained, Dembo's [16] program was compiled in two versions and then used to solve a four-variable, two-constraint problem. The improvement in total CPU time of using (4.37) in lieu of (4.36) was about 26%.

4.5.2 Software Implementation

In this section the software implementation in FORTRAN IV of the proposed CRGCP algorithm is briefly described. The complete software package, hereafter referenced as SP, consists of a master program MAIN which oversees the operations of 16 subroutines. As a whole, the computer program SP has the following highlighting features:

1. It contains all the capabilities of Dembo's program GGP [16].
2. It is compatible with the I/O format of GGP except for minor changes in the control cards and in the order of reading the data of the objective function.

---

[1] For example, in the IBM 370/168, the CPU time required by an exponentiation operation is approximately 17 and 26 times more than that of multiplication and addition, respectively.

3. It uses the exponential form (4.37) for computing posynomials.
4. It offers a choice of whether GGP or CRGCP should be used. The choice is indicated by the value of a single control parameter.
5. If CRGCP is selected, a function OBJFCN declared in an EXTERNAL statement has to be supplied for the purpose of computing the function and gradient values of the objective function.

The last two features provide the flexibility of using CRGCP to solve signomial programs whose objective function is more complex than an explicitly stated signomial. This aspect will be thoroughly discussed in Section 4.7. It will also prove useful for solving the subproblem (3.9) of Algorithm 3.2 designed to solve signomial programs with equality constraints.

4.5.3 Computational Results

In order to acquire some experience with the performance of the CRGCP algorithm, the code SP was used to solve a set of signomial programs. The test problems together with the starting points are given below. Note that all the starting points are infeasible.

Problem A (A quadratically constrained quadratic program)

minimize $g_0(\underline{x}) = 3x_1^2 + 2x_2^2 + x_3^2 + x_4^2 + 7x_1 + 565 - 39x_2 - 17x_3 - x_4$

s. t. $g_1(\underline{x}) = (1/6)x_1^2 + (1/18)x_2^2 + (1/9)x_4^2 + (1/18)x_1 - (1/9)x_3^2 - (1/18)x_2 \leq 1$

$g_2(x) = x_1^2 + x_3^2 + 2.5x_4^2 + x_3 + 9.5 - 4x_2^2 - x_4 \leq 1$

$$g_3(\underline{x}) = x_2^2 + 3x_3^2 + x_4^2 + x_2 - x_2^2 - x_1 - x_4 - 2 \leq 1$$

$$0.01 \leq x_j \leq 10.0, \quad j = 1, 2, 3, 4$$

Starting Point: $\underline{x} = (2.0, 2.0, 2.0, 2.0)'$

Problem B

$$\text{minimize } g_0(\underline{x}) = 0.000392x_1^3 + 5.46x_3^3 + 0.09x_1^{-1}x_2x_3^{-1} + 6.0x_2x_3 + 2.5856x_2^{-1} - 0.125x_2^3 - 1.8x_1x_3$$

$$\text{s. t. } g_1(\underline{x}) = 0.03x_1^2 + 0.1x_2^2 - 0.15x_3^2 \leq 1$$

$$g_2(\underline{x}) = 5.0x_1x_2 + 3.6x_2^{0.5}x_3 - 0.04x_1^2x_3^{-1} \leq 1$$

$$0.01 \leq x_j \leq 10.0, \quad j = 1, 2, 3$$

Starting Point: $\underline{x} = (2.0, 2.0, 2.0)'$

Problem C (Heat Exchanger Design [48])

$$\text{minimize } g_0(\underline{x}) = c_1x_1 + c_2x_2 + c_3x_3$$

$$\text{s. t. } g_1(\underline{x}) = c_4x_1x_4x_6^{-1} + c_5x_6^{-1} + c_6x_1^{-1}x_6^{-1} \leq 1$$

$$g_2(\underline{x}) = c_7x_2^{-1}x_5x_7^{-1} + c_8x_4x_7^{-1} + c_9x_2^{-1}x_4x_7^{-1} \leq 1$$

$$g_3(\underline{x}) = c_{10}x_3^{-1}x_8^{-1} + c_{11}x_5x_8^{-1} + c_{12}x_3^{-1}x_5x_8^{-1} \leq 1$$

$$g_4(\underline{x}) = c_{13}x_4 + c_{14}x_6 \leq 1$$

$$g_5(\underline{x}) = c_{15}x_5 + c_{16}x_7 + c_{17}x_4 \leq 1$$

$$g_6(\underline{x}) = c_{18}x_8 + c_{19}x_5 \leq 1$$

$$100.0 \leq x_1 \leq 10000.0$$

$$1000.0 \leq x_j \leq 10000.0, \quad j = 2, 3$$

$$10.0 \leq x_j \leq 1000.0, \quad j = 4, \ldots, 8$$

Starting Point: $\underline{x}$ = (5000.0, 5000.0, 5000.0, 200.0, 350.0, 150.0, 225.0, 425.0)

The coefficients $c_1, \ldots, c_{19}$ are given in Table 4.1.

The statistics of the test problems are summarized in Table 4.2. In this table the degree of difficulty D is defined as the difference between the total number of terms and the number of variables plus one. The terms due to the artificial simple lower and upper bounds are neglected. If a dual approach were used to solve the problems, then the dimension of the respective dual problem would be D.

All the problems were solved on an IBM 360/168 computer using double-precision arithmetic. The sign tolerance was 10^{-12} while the feasibility tolerance was 10^{-6}. The linear search was inexact and was terminated either when the norm of the directional derivative along the search direction was less than 10^{-3} or when a specified number of searches N_s had been completed. The final termination criterion was relaxed to be that the relative change in $g_o(\underline{x})$ be less than 10^{-4}. The solutions obtained by SP are given in Tables 4.3 (a)-(b) for $N_s = 5$ and $N_s = 10$. Also quoted in the tables are the elapsed CPU time for obtaining each solution. The elapsed time is the time between the moment when a feasible point has been obtained and the time when the termination criterion is satisfied. For the purpose of comparison, the solutions obtained by Dembo's GGP code under the same feasibility and optimality tolerances are included in the last column of each table.

j	c_j	j	c_j
1	1.0	11	1.0
2	1.0	12	-2500.0
3	1.0	13	0.0025
4	833.33252	14	0.0025
5	100.0	15	0.0025
6	-83333.333	16	0.0025
7	1250.0	17	-0.0025
8	1.0	18	0.01
9	-1250.0	19	-0.01
10	1250000.0		

Table 4.1 Coefficients for Problem C

	A	B	C
No. of Variables	4	3	8
No. of Constraints	3	2	6
No. of Terms	30	13	19
Degree of Difficulty	25	9	10

Table 4.2 Statistics of the test problems for the CRGCP algorithm.

	Starting Point	SP $N_s = 5$	SP $N_s = 10$	GGP
x_1	2.0	2.05478	2.05510	2.04978
x_2	2.0	2.45474	2.45420	2.46178
x_3	2.0	0.48977	0.59120	0.57044
x_4	2.0	0.44928	0.44892	0.44517
Optimal Cost	-	498.44053	498.44000	498.45041
CPU Time	-	1.190 sec	0.933 sec	0.531 sec

(a)

	Starting Point	SP $N_s = 5$	SP $N_s = 10$	GGP
x_1	2.0	2.92012	2.92008	2.92042
x_2	2.0	2.72930	2.72931	2.72920
x_3	2.0	0.06929	0.06929	0.06930
Optimal Cost	-	0.40205	0.40205	0.40205
CPU Time	-	0.264 sec	0.109 sec	0.307 sec

(b)

Table 4.3 Starting point and solution of (a) Problem A, (b) Problem B

The numerical results of Table 4.3 indicate that while the code SP is not consistently faster than GGP, the rate of convergence is in the same order of magnitude as that of GGP. The solutions obtained by the two codes are reasonably close. On a percentage basis, the variation in the optimal cost is less than that in the norm of $\underline{x}$. The smaller difference in the optimal cost is perhaps due to the bias introduced by the choice of the termination criterion. Stopping the inexact linear search earlier by reducing N_s may hasten convergence. However, in certain problems the gain in time due to less linear searches may be offset by the need to solve more approximating geometric programs.

	Starting Point	SP $N_s = 5$	SP $N_s = 10$	GGP
x_1	5.D3	553.58153	569.28585	568.55211
x_2	5.D3	1367.87035	1369.15269	1372.33969
x_3	5.D3	5128.21662	5110.96764	5108.52350
x_4	200.	179.82943	181.17369	181.11155
x_5	350.	294.87134	295.56165	295.65912
x_6	150.	220.17088	218.82631	218.88862
x_7	225.	284.95809	285.61097	285.45120
x_8	425.	394.87134	395.56161	395.65911
Optimal Cost	-	7049.6685	7049.4062	7049.4107
CPU Time	-	1.025 sec	1.268 sec	1.501 sec

(c)

Table 4.3 (cont'd) Starting point and solution of (c) Problem C

4.6 Presence of Simple Upper Bounds

Simple upper bound constraints of the form $\underline{x} \leq \underline{x}^u$ frequently occur in engineering design problems. In the problem statement of (4.1), such type of constraint is not explicitly expressed since in principle $\underline{x} \leq \underline{x}^u$ can be included in the feasible set F by writing m monomial constraints of the form $x_j/x_j^u \leq 1$. Such a straightforward approach, however, is computationally very inefficient. In this section the problem formulation of (4.1) is extended to allow explicit simple upper bounds on $\underline{x}$. The special simple structure of these constraints is exploited to specify modifications in Algorithm 4.1 so that the upper bounds are indirectly taken into account without any increase in the number of variables or constraints.

The problem to be solved is

$$\begin{aligned} &\min && g_o(\underline{z}) \\ &\text{s.t.} && \underline{x} \in X \end{aligned} \tag{4.37}$$

where $X = \{\underline{x} : \underline{x} \in R^m;\ 0 < \underline{x}^L \leq \underline{x} \leq \underline{x}^u;\ g_k(\underline{x}) \leq 1,\ k = 1, 2, \ldots p\}$ and $g_k(\underline{x})$ are signomials. As in Section 4.3.3, if $x_j = \exp(z_j)$, (4.37) can be expressed in terms of $\underline{z}$ to yield the following problem

$$\begin{aligned} &\min && g_o(\underline{z}) \\ &\text{s.t.} && \underline{z} \in Z \end{aligned} \tag{4.38}$$

where $Z = \{\underline{z} : \underline{z} \in R^m;\ \underline{z}^L \leq z \leq \underline{z}^u;\ P_k(\underline{z})/(1+Q_k(\underline{z})) \leq 1,\ k = 1, 2, \ldots p\}$. It follows that (4.38) can be solved by Algorithm 4.2 of Section 4.3.4 except now the problem corresponding to problem (LST) of (4.17) is defined as

$$\begin{aligned} &\min && g_o(\underline{\zeta}) \\ &\text{s.t.} && [J \mid I] \begin{bmatrix} \underline{\zeta} \\ \underline{\zeta}_s \end{bmatrix} = \underline{b} \end{aligned} \tag{4.39}$$

$$0 \leq \underline{\zeta} \leq \underline{\zeta}^u,\ 0 \leq \underline{\zeta}_s$$

where J and $\underline{b}$ are as defined in (4.16), and $\zeta_j^u = \ln(x_j^u/x_j^L)$, $j = 1, 2, \ldots, m$. Thus, the presence of simple upper bounds on $\underline{x}$ leads to simple upper bounds on $\underline{\zeta}$, but not on the slack variables $\underline{\zeta}_s$. It follows that simple upper bounds in the original problem necessitate changes only in the reduced gradient method used to solve (4.39).

Consider the reduced gradient method summarized in Algorithm 4.1. For convenience of notation, define the upper bound on $\underline{\zeta}_s$ as a vector $\underline{\zeta}_s^u$ of arbitrarily large numbers. Then the required changes in Algorithm 4.1 are in the definition of the search direction $\underline{s}^k$, the determination of the maximum allowable step size λ_M^k, and the criterion for the replacement of a basic variable. The revised algorithm capable of solving (4.39) is like Algorithm 4.1 except in the following steps in the kth iteration:

Step 2 i) $s_{N_i} = 0$ if a) $\zeta_{N_i} = 0$ and $v_{N_i} > 0$ or b) $\zeta_{N_i} = \zeta_{N_i}^u$ and $v_{N_i} < 0$

Else, $s_{N_i} = -v_{N_i}$;

Step 5 $\lambda_M = \min\left[\min\{-\zeta_j/s_j \mid s_j < 0, \forall j\}, \min\{\dfrac{\zeta_j^u - \zeta_j}{s_j} \mid s_j > 0\}\right]$;

Step 6 ζ_{B_i} becomes nonbasic if $\zeta_{B_i} = 0$ or $\zeta_{B_i}^u$. Note that a basic variable has to be between the two bounds while a nonbasic variable can be nonzero.

For reference purposes, the reduced gradient method with the above rules is called Algorithm 4.1(a).

A numerical example solved by Algorithm 4.1(a) is Colville's [39] third test problem given below.

$$\min \quad g_o(\underline{x}) = c_1 x_3^2 + c_2 x_1 x_5 + c_3 x_1 + c_4$$

$$\text{s.t.} \quad g_1(\underline{x}) = c_5 x_3 x_5 + c_6 x_2 x_5 + c_7 x_1 x_4 \leq 1$$

$$g_2(\underline{x}) = c_8 x_2 x_5 + c_9 x_1 x_4 + c_{10} x_3 x_5 \leq 1$$

$$g_3(\underline{x}) = c_{11} x_2^{-1} x_5^{-1} + c_{12} x_1 x_5^{-1} + c_{13} x_2^{-1} x_3^2 x_5^{-1} \leq 1$$

$$g_4(\underline{x}) = c_{14} x_2 x_5 + c_{15} x_1 x_2 + c_{16} x_3^2 \leq 1$$

$$g_5(\underline{x}) = c_{17} x_3^{-1} x_5^{-1} + c_{18} x_1 x_5^{-1} + c_{19} x_4 x_5^{-1} \leq 1$$

$$g_6(\underline{x}) = c_{20} x_3 x_5 + c_{21} x_1 x_3 + c_{22} x_3 x_4 \leq 1$$

$$78 \leq x_1 \leq 102$$

$$33 \leq x_2 \leq 45$$

$$27 \leq x_j \leq 45 , \quad j = 3, 4, 5$$

The coefficients $c_1, \ldots, c_{22}$ are given in Table 4.4. All the lower and upper bounds on x_j, $j = 1, \ldots, 5$, are part of the problem formulation. Hence the degree of difficulty of the problem is 26. Two feasible starting points and their respective solutions are presented in Table 4.5. The solutions were obtained with a feasibility tolerance of 10^{-6} and an optimality tolerance of 10^{-4}. Note that the solution involves lower and upper bounds. The CPU elapsed times quoted in Table 4.5 are as defined in Section 4.5.3.

j	c_j	j	c_j
1	5.35785470	12	-0.4200261
2	0.83568910	13	-0.30585975
3	37.239239	14	0.00024186
4	-40792.141	15	0.00010159
5	0.00002584	16	0.00007379
6	-0.00006663	17	2275.132693
7	-0.00000734	18	-0.26680980
8	0.00853007	19	-0.40583930
9	0.00009395	20	0.00029955
10	-0.0087777	21	0.00007992
11	1330.32937	22	0.00012157

Table 4.4 Coefficients of example problem with simple upper bounds.

	1st Starting Pt.	Solution	2nd Starting Pt.	Solution
x_1	78.62	78.0	80.0	78.0
x_2	33.44	33.0	35.0	33.0
x_3	31.07	29.99307	35.0	29.99307
x_4	44.18	45.0	35.0	45.0
x_5	35.22	36.78135	35.0	36.78135
Optimal Cost	-	-30670.093	-	-30670.093
CPU Time	-	0.086 sec	-	0.094 sec

Table 4.5 Starting points and solutions of example problem with simple upper bounds.

4.7 Minimizing Algebraic Functionals of Signomials

4.7.1 Introduction

The algebraic operations are addition, subtraction, multiplication, division, and exponentiation. An algebraic functional of signomials is a real-valued functional generated by performing specified algebraic operations on a finite set of signomials. The addition, subtraction, multiplication, and the positive integral powers of signomials can in principle be formally expressed as signomials. Ratios and real powers of signomials generally cannot be reduced to the standard form of a signomial. The problem of interest in this section is the minimization of algebraic functionals of signomials subject to signomial inequality constraints. This type of problem is termed as a semi-algebraic program.

The reasons for considering this class of problems are:

1. The subproblem (3.9) required in Step 1 of Algorithm 3.2 for signomial programs with equality constraints belongs to this class of problems.
2. A wider range of applications can be modeled by signomial programs.
3. Considerable saving in the effort for data preparation is likely as the algebraic operations on the signomials need not be carried out.

In Section 4.7.1, the problem of interest is formulated. The same section also discusses how the combined reduced gradient - cutting plane algorithm can be used to solve semi-algebraic programs. A numerical example of semi-algebraic programs is also presented in Section 4.7.1. In Section 4.7.2 the important practical constrained location-allocation problem is shown as an illustrative example of semi-algebraic programs. Although some constrained location-allocation problems can be formulated as standard signomial programs, such a formulation usually has a

large number of terms, hence, a high degree of difficulty. The advantage of solving constrained location-allocation problems by the CRGCP algorithm (Algorithm 4.2(a)) is demonstrated via an example.

4.7.2 Problem Formulation and Solution

Let $s_1(\underline{x}), \ldots, s_L(\underline{x})$ be signomials and define $f(s_1(\underline{x}), \ldots, s_L(\underline{x}))$ as a real-valued functional generated by executing a set of specified algebraic operations on $s_1, \ldots, s_L$. Then a semi-algebraic program may be formally defined as

$$\begin{aligned} &\min\ f(s_1(\underline{x}), \ldots, s_L(\underline{x})) \\ &\text{s. t.} \quad \underline{x} \in X \end{aligned} \tag{4.40}$$

where $X = \{\underline{x} : \underline{x} \in R^m;\ 0 < \underline{x}^L \leq \underline{x} \leq \underline{x}^u;\ g_k(\underline{x}) \leq 1,\ k = 1, 2, \ldots, p\}$ and $g_k(\underline{x})$, $k = 1, 2, \ldots, p$, are signomials.

The only difference between (4.40) and (4.37) is the form of the objective function. But since Algorithm 4.2(a) does not really assume that the objective function has to be a signomial, it follows that the algorithm remains applicable if it is used to solve a semi-algebraic program. This is precisely the approach adopted in this section. In fact, the only necessary restriction of f as a function of $\underline{x}$ is that f be continuously differentiable over X.

To illustrate the problem just formulated and the proposed method for solving the problem, consider the following two-variable numerical example

$$\begin{aligned} &\min\ f(\underline{x}) = (s_1(x))(s_2(x))^{-1.5} - (s_3(x))^{2.0}(s_4(x))^{0.55} \\ &\text{s. t. (a)}\ x_1^2/2.25 + x_2^2/16 + 5 - 4x_1/1.5 - 0.5x_2 \leq 1 \\ &\qquad\text{(b)}\ x_1/5 + x_2/15 \leq 1 \end{aligned} \tag{4.41}$$

$$\text{(c)}\quad 0.947x_1^{0.7} + 2.0 - x_2 \leq 1$$

$$\text{(d)}\quad 0.1 \leq x_j \leq 100.0, \quad j = 1, 2$$

where s_1, s_2, s_3, and s_4 are signomials given as

$$\begin{aligned} s_1(x) &= x_1^{0.5} - x_2^{0.4} \\ s_2(\underline{x}) &= x_1^2 + x_2 \\ s_3(\underline{x}) &= x_1x_2 - x_1^{1.25} \\ s_4(\underline{x}) &= x_2^{-0.4} + x_1^{-1.0}\, x_2 \end{aligned} \tag{4.42}$$

Note that the objective function is a signomial-like functional of the signomials s_1, s_2, s_3 and s_4. The feasible set defined by (4.41) (a) - (d) is a nonconvex set. Example (4.41) was solved by Algorithm 4.2 on the IBM 370/168 computer. Beginning with the infeasible initial point (1.5, 1.5) the algorithm obtains via Dembo's Phase I procedure the feasible starting point (1.67477, 2.45071). From this feasible point, the algorithm obtains, after four approximations of the nonconvex feasible set, the final solution of (2.43246, 7.70263). At the solution, constraints (4.41) (a) and (b) are tight while constraint (4.41) (c) is loose. The overall CPU time required is 1.01 sec.

4.7.3 Application to Constrained Location-Allocation Problems

The optimal location-allocation problem may be generally stated as follows: Given the location and the requirements of a set of fixed known destinations, determine (a) the number of sources, (b) the location and the capacity of each source, (c) the source-to-source and source-to-destination shipment pattern such that the destinations' requirements are satisfied and that the total set-up, operation and ship-

ping costs are minimized. This problem arises in the planning of warehouses, distribution centers, communication switching centers, production and municipal services facilities.

Under the guise of such names as the Fermat, the Steiner, or the Generalized Weber problem, variants of the location-allocation problem have been considered by a host of researchers. A recent bibliography [49] lists over 200 papers in the decade before 1973. Most of the proposed methods, however, cannot handle constraints restricting the sources' coordinates. In real situations, some constraints need to be imposed on the location of the sources for a variety of reasons such as geographical barriers, zoning laws, or safety requirements. In recent years some theoretical [50] and algorithmic [51] - [53] results related to the constrained location problem have been reported. In this section, a restricted version of the general problem is formulated as a semi-algebraic program that can be solved by the approach discussed in this chapter. Algorithm 4.2 is demonstrated to be another tool available for solving certain types of constrained location-allocation problems.

The version of the constrained location-allocation problem to be solved in this section is posed as follows: Suppose that the coordinates $\underline{a}_j$ and the demand r_j of the jth sink, j = 1, 2, ... , m, are known. Let the ith source, i = 1, 2, ... , n have a location at $\underline{x}_i$ and a capacity c_i. If it is assumed that the shipping cost from one point to another is proportional to the distance between the points, then the mathematical formulation is

$$\min \quad C(\underline{x}) = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ji} d(\underline{x}_j, \underline{a}_j) + \sum_{j=1}^{n-1} \sum_{k=1}^{n} v_{jk} d(\underline{x}_j, x_k)$$

$$\text{s.t.} \quad \text{(a)} \quad \sum_{j=1}^{m} w_{ji} - \sum_{j=1}^{n-1} v_{ji} \leq c_i , \qquad i = 1,2,\ldots, n \tag{4.43}$$

$$\text{(b)} \quad \sum_{i=1}^{n} w_{ji} \leq r_j , \qquad j = 1,2,\ldots, m$$

$$\text{(c)} \quad g_k(\underline{x}_1,\ldots,\underline{x}_n) \leq 1.0, \qquad k = 1,2,\ldots, p$$

where d(x, y) is a distance function of x and y, and $g_k(\underline{x}_1, \ldots, \underline{x}_n)$, $k = 1, 2, \ldots, p$ are signomials expressed in terms of the sources' coordinates. w_{ji} is the amount shipped from source j to sink i while v_{jk} is the shipment from source j to source k. Note that source n is assumed not to supply anything to other sources in order to model a facility whose output does not serve as the raw material of other sources. In (4.43), if the variables w_{ji} and v_{jk} are all held fixed, the problem becomes a multi-source location problem. The common distance functions used are the Euclidean and the rectangular ("city block") distance functions. In the example of this section, the Euclidean distance is used.

In (4.43), if each d is replaced by the appropriate Euclidean distance expression, it becomes clear that the constrained location - allocation problem as formulated in (4.43) is a semi-algebraic program. It follows that Algorithm 4.2 can be used in a straightforward manner. This point is illustrated by the solution of the following single-source constrained location problem adapted from [51].

$$\min \sum_{i=1}^{24} w_i[(x_1 - a_1^i)^2 + (x_2 - a_2^i)^2]^{1/2} \tag{4.44}$$

$$\text{s. t.} \quad \text{(a)} \ (x_1 - 44.0)^2 + (x_2 - 36.2)^2 \leq 2.0$$

$$(b)\ x_2 + 1.25x_1 \leq 92.75$$

$$(c)\ 0.0 < x_1, x_2$$

The constraints in (4.44) require the source to be located within 2 units of (44, 36.2) and to be below the line $x_2 = -1.25x_1 + 92.75$. The data of the problem are given in Table 4.6. Originally provided by Kuhn and Kuenne [54], the data are those of the 24 cities in the Ukraine which rank among the 100 most populous cities in the U.S.S.R. The weight w_i of the ith city is taken as the proportion of the city's population of the total population of the 100 largest Russian cities. The coordinates (a_1^i, a_2^i) of the ith city are the city's north latitude and east longitude correct to the nearest full degree.

The results of solving (4.44) by Algorithm 4.2 are summarized in Table 4.7. In [51], Gurovich did not enforce the two constraints of (4.44) simultaneously. Hence it is not possible to make a direct comparison between her results obtained by algebraic programming (AP) [36], [51] and those reported here. However, the results in Table 4.7 still are very much superior to Gurovich's results obtained when only either of the constraints is enforced. For example, in [51] enforcing (4.44a) only requires 6 convex approximations and about 6 CPU secs, while enforcing (4.44 b) only requires eight convex approximations and about 13 CPU secs. (IBM 370/168). With the use of Algorithm 4.2, enforcing <u>both</u> constraints requires only four convex approximations and 0.874 CPU sec. A plausible explanation for the wide margin in performance is that in AP the objective function is incorporated into the constraint set. This not only increases the number of variables by one but also requires more approximations. It was precisely this difficulty that led to the development of the main algorithm proposed in this chapter.

i	a_1^i	a_2^i	w^i	i	a_1^i	a_2^i	w^i
1	54	28	0.012	13	47	38	0.007
2	50	24	0.010	14	48	37	0.016
3	47	28	0.005	15	48	38	0.008
4	46	31	0.014	16	48	38	0.007
5	47	32	0.005	17	47	39	0.005
6	47	33	0.004	18	45	39	0.007
7	45	34	0.004	19	47	40	0.014
8	45	34	0.004	20	48	40	0.004
9	46	34	0.001	21	49	39	0.004
10	48	35	0.004	22	41	37	0.021
11	48	35	0.015	23	50	31	0.026
12	48	36	0.010	24	52	31	0.004

Table 4.6 Locations and weights for major Ukraine cities.

Initial Point	(40.0, 20.0)
Optimal Point	(45.3268, 36.0105)
Optimal Cost	1.0896
No. of Convex Approx	4
Constraint Tolerance	10^{-5}
Optimality Tolerance	10^{-4}
Total CPU Time	0.874 sec

Table 4.7 Summary of the solution of the constrained location problem.

V. NUMERICAL SOLUTION OF SIGNOMIAL PROGRAMS WITH EQUALITY CONSTRAINTS

5.1 Introduction

In Chapter III an algorithm based on the multiplier method and the concept of partial duality has been developed to solve signomial programs with equality constraints. The algorithm is so structured that a sequence of inequality-constrained signomial programs is solved. This approach allows the exploitation of a key property of signomial programs, namely, their easy approximation by convex programs. As discussed in Section 3.5, the convergence of the proposed method is assured by the convergence of the method of multiplier provided that the assumptions for convergence hold and the following conditions are satisfied:

1) the penalty constant K is sufficiently large after a finite number of iterations,
2) a local minimizing point of the primal problem (π_1) is obtained, and
3) the unconstrained dual problem is maximized.

Except for the discussion on how (π_1) could be solved, no specific methods for achieving conditions (1) and (3) are mentioned in Chapter III. The first purpose of this chapter is to delineate clearly how each of the three conditions can be met. Secondly, the proposed alternatives are tested in a numerical experiment and their computational efficiency is assessed.

How the subproblem (π_1) is solved is important not only because the solution method determines the accuracy of the answer wanted but also because it consumes a major part of the total computing time. As summarized in Section 3.6, earlier computational experience revealed

the numerical difficulties of the two indirect methods, Method I and Method II, that had been considered for solving (π_1). In Section 5.2, the Algorithm 4.2 developed in the previous chapter is adopted to solve (π_1). The numerical results presented in Chapter IV suggest that Algorithm 4.2 can handle well signomial programs whose objective function may have many terms or may not even be pure signomials. This ability of the algorithm is confirmed in the numerical tests of this chapter. In Section 5.3, the discussion focuses on the important question of how to update the multiplier estimate vector $\underline{\lambda}$ and the penalty constant K, given the initial values of these parameters. Section 5.4 contains the description and the results of the numerical experiments conducted to test the proposals formulated in Section 5.3.

5.2 Solution of the Subproblem (π_1)

The subproblem (π_1) of Algorithm 3.2 is

$$(\pi_1) \qquad \min \ell(\underline{x}, \underline{\lambda}, K) = x_1 + \sum_{k=1}^{q} \lambda_k h_k(\underline{x}) + K \sum_{k=1}^{q} h^2{}_k(\underline{x}) \qquad (5.1)$$

$$\text{s.t.} \qquad \underline{x} \in X$$

$$\text{where} \qquad h_k(\underline{x}) = g_{p+k}(\underline{x}) - 1, \quad k = 1, 2, \ldots q,$$

$$\text{and} \quad X = \{\underline{x} : \underline{x} \in R^m;\ g_k(\underline{x}) \leq 1,\ k = 1, 2, \ldots p;\ 0 \leq \underline{x}^L \leq \underline{x} \leq \underline{x}^u\}$$

It is pointed out in Section 3.6 that the signomial $\ell(\underline{x}, \underline{\lambda}, K)$ is likely to have many terms. This observation suggests that any method which approximates $\ell(\underline{x}, \underline{\lambda}, K)$ grossly may fail to exhibit good convergence, if there is any at all. Algorithm 4.2 developed in the last chapter is designed to solve inequality-constrained signomial programs without approximating the objective function. Later in Section 4.7, the algorithm is extended to solve programs whose objective functions are algebraic

functionals of signomials. Thus, Algorithm 4.2 is just suitable for (π_1) since a quick examination of (5.1) shows that $\ell(\underline{x}, \underline{\lambda}, K)$ is a quadratic form of the vector $[h_1, h_2, \ldots h_q]'$. In the numerical tests reported in this chapter, the solution of the subproblem (π_1) is accomplished with Algorithm 4.2. This is the Method III alluded to at the end of Chapter III.

The application of Algorithm 4.2 requires the calculation of both the function $\ell(\underline{x}, \underline{\lambda}, K)$ and its gradient $\nabla_x \ell(\underline{x}, \underline{\lambda}, K) = [\frac{\partial \ell}{\partial x_1}, \ldots, \frac{\partial \ell}{\partial x_m}]'$. The additional effort to calculate $\nabla_x \ell$ is little indeed. Let $P_{p+k} = \sum_{j\in[k]} u_{jk}$, $Q_{p+k} = \sum_{j\in\{k\}} v_{jk}$, $u_{jk} = c_{jk} \prod_{i=1}^{m} x_i^{a_{ijk}}$ and $v_{jk} = d_{jk} \prod_{i=1}^{m} x_i^{b_{ijk}}$. $[k]$ and $\{k\}$ are appropriately defined index sets. Then for $j \geq 2$

$$\frac{\partial \ell}{\partial x_j} = \sum_{k=1}^{q} (\lambda_k - 2Kh_k) \frac{\partial h_k}{\partial x_j} \tag{5.2}$$

where

$$\frac{\partial h_k}{\partial x_j} = \sum_{j\in[k]} a_{ijk} u_{jk} - \sum_{j\in\{k\}} b_{ijk} v_{jk} \tag{5.3}$$

Since in the process of calculating h_k, u_{jk}, $j\in[k]$, and v_{jk}, $j\in\{k\}$ can be temporarily stored, calculating $\frac{\partial h_k}{\partial x_j}$ simply needs the additional arithmetic operations specified by (5.3).

5.3 Schemes for Updating $\underline{\lambda}$ and K

5.3.1 Introduction

Algorithm 3.2 developed to solve signomial programs with equality constraints is an adaptation of the method of multipliers as modified

by partial duality. For a sufficiently large penalty constant K, the algorithm can be viewed as a primal-dual method that solves the original problem (SPE) (3.1) by solving the unconstrained dual problem

$$(D) \qquad \max_{\underline{\lambda} \in R^q} d_K(\underline{\lambda}) \tag{5.4}$$

where the dual functional $d_K(\underline{\lambda})$ is defined as

$$(P) \qquad d_K(\underline{\lambda}) = \min_{\underline{x} \in X} \ell(\underline{x}, \underline{\lambda}, K) \tag{5.5}$$

The objective function $\ell(\underline{x}, \underline{\lambda}, K)$ and the feasible set X are as defined in (3.9). It is stated in Section 3.5 that subject to certain assumptions, the dual functional $d_K(\underline{\lambda})$ is twice continuously differentiable with

$$\nabla d_K(\underline{\lambda}) = \underline{h}(\tilde{\underline{x}}) \tag{5.6}$$

$$\nabla^2 d_K(\underline{\lambda}) = - \nabla \underline{h}(\tilde{\underline{x}}) \; [\nabla^2 \ell(\tilde{\underline{x}}, \underline{\lambda}, K)]^{-1} \; \nabla \underline{h}(\tilde{\underline{x}}) \tag{5.7}$$

where $\underline{h}(\underline{x})$ is the vector of signomial equality constraints, $\nabla \underline{h}$ is the Jacobian of $\underline{h}$ and $\nabla^2 \ell$ is the Hessian of $\ell(\underline{x}, \underline{\lambda}, K)$ with respect to $\underline{x}$. $\tilde{\underline{x}}$ is the solution of (5.5) given $\underline{\lambda}$ and K. It follows that the problem (D) can, in principle, be solved by any unconstrained gradient-based ascent algorithm. In such type of algorithm, the (k + 1)st iterate is obtained according to the following general form

$$\underline{\lambda}^{(k+1)} = \underline{\lambda}^{(k)} + \alpha^{(k)} M^{(k)} \; \underline{h} \; (\tilde{\underline{x}}(\underline{\lambda}^{(k)}, K)) \tag{5.8}$$

where $\alpha^{(k)}$ is the step size and $M^{(k)}$ is a q x q positive definite matrix. A special case of (5.8) is the fixed-step steepest ascent formula of Hestenes and Powell in which $\alpha^{(k)} = 2K$ and $M^{(k)} = I$, the identity matrix. Note that in (5.8), K remains constant in each iteration. Also the

dependence of $\tilde{\underline{x}}$ on $\underline{\lambda}$ and K is emphasized in the equation. In Section 5.3.2, assuming that $M^{(k)} = I$, two choices of the step size $\alpha^{(k)}$ are considered. In Section 5.3.3, alternative ways of specifying K to ensure, albeit probabilistically, the global convergence of the algorithm.

5.3.2 Choice of Step Size α

Assume that $M^{(k)} = I, \forall k$ and that $K^{(k)} = K \geq K^*$, for $k \geq \bar{k}$, where K^* is the minimum value of K to assure global convergence and $\bar{k}$ is some finite integer. Then (5.8) simplifies to the iteration equation of the method of steepest ascent given by

$$\underline{\lambda}^{(k+1)} = \underline{\lambda}^{(k)} + \alpha^{(k)} \, \underline{h} \, (\tilde{\underline{x}}(\underline{\lambda}^{(k)}, K)) \tag{5.9}$$

The remaining question is: How should the step size $\alpha^{(k)}$ be selected? Before answering this question, it must first be noted that while $d_K(\underline{\lambda})$ and $\nabla d_K(\underline{\lambda}) = \underline{h}(\tilde{\underline{x}}(\underline{\lambda}, K))$ can be calculated, the effort required to compute $d_K(\underline{\lambda})$ amounts to solving a minimization problem. The gradient $\nabla d_K(\underline{\lambda})$ is essentially a free byproduct of the minimization. For this reason, the step size $\alpha^{(k)}$ should be determined as simply as it is possible. Schemes requiring gradient information are acceptable, provided they do not require too many trial points since each point calls for a minimization problem.

Two step size choices are considered. The first is the simple but effective Hestenes-Powell rule stipulating that in general, $\alpha^{(k)} = 2K^{(k)}$, $K^{(k)} \geq K^{(k-1)}$, so that

$$\underline{\lambda}^{(k+L)} = \underline{\lambda}^{(k)} + 2K^{(k)} \, \underline{h} \, (\tilde{\underline{x}}^{(k)}) \tag{5.10}$$

Note that the penalty constant $K^{(k)}$ may change from iteration to iteration. If $K^{(k)} = K_o, \forall k$, then (5.10) becomes the fixed-step gradient

method. The local convergence rate of (5.10) was studied by Bertsekas [27] who proved that if $(\underline{x}^*, \underline{\lambda}^*)$ satisfies the second-order sufficient optimality conditions and $K^{(k)} = K^*$, $k \geq \bar{k}$, for some positive number K^* and integer $\bar{k}$, then $\underline{\lambda}^{(k)} \to \underline{\lambda}^*$ at a linear rate with a convergence ratio essentially inversely proportional to $K^{(k)}$ provided $\underline{\lambda}^{(o)}$ is sufficiently close to $\underline{\lambda}^*$.

If $K^{(k)} = K_o, \forall k$, the Hestenes-Powell updating formula (5.10) uses only the gradient $\nabla d_{K_o}(\underline{\lambda})$ at the present point $\underline{\lambda}^{(k)}$ to estimate the next point $\underline{\lambda}^{(k+1)}$. A possible acceleration scheme that requires a modest increase in computational effort is to use a low-order polynomial interpolation scheme such as fitting a cubic. However, K_o may have to be increased after some iterations in order to ensure convergence. Suppose that $K^{(k+1)} = K_1$ and $K^{(k)} = K_o$, $K_1 \geq K_o$. Then $\nabla d_{K_1}(\underline{\lambda})$ and $\nabla d_{K_o}(\underline{\lambda})$ are gradients of different dual functionals. Clearly, they cannot be used to interpolate either $d_{K_o}(\underline{\lambda})$ or $d_{K_1}(\underline{\lambda})$. A common functional is needed if interpolation is to be used and the penalty constant K can still be changed. A solution is to use $d_0(\underline{\lambda})$, i.e.,

$$d_0(\underline{\lambda}) = \min_{\underline{x} \in X} \ell(\underline{x}, \underline{\lambda}, 0) \tag{5.11}$$

(5.11) is just the minimization of the ordinary Lagrangian. Observe that

$$d_0(\underline{\lambda} + 2Kh(\underline{x}(\underline{\lambda},K))) = d_K(\underline{\lambda}) \tag{5.12}$$

Hence to approximate $d_0(\underline{\lambda})$ by passing a low-order polynomial fit through two points, $d_0(\underline{\lambda} + 2Kh(\underline{x}(\underline{\lambda},K)))$ has to be evaluated for any two pairs of $\underline{\lambda}$ and K. This is the rationale of the following algorithm whose local convergence was proved by Bertsekas [27]. However, no comparative

numerical evidence was provided.

Algorithm 5.1

Step 0: Set k = 1. Assume $\underline{\lambda}^{(2k-1)}$ and the sequence $\{K^{(j)}\}$ are specified.

Step 1: Obtain $\underline{x}^{(2k-1)}$ by minimizing $\ell(\underline{x}, \underline{\lambda}^{(2k-1)}, K^{(2k-1)})$ s.t. $\underline{x} \in X$.

Step 2: $\underline{\lambda}^{(2k)} = \underline{\lambda}^{(2k-1)} + 2K^{(2k-1)}\ \underline{h}(\underline{x}^{(2k-1)})$. It follows from (5.12) and Step 2 that $d_0(\underline{\lambda}^{(2k)})$ and $\nabla d_0(\underline{\lambda}^{(2k)})$ are known.

Step 3: Obtain $\underline{x}^{(2k)}$ by minimizing $\ell(\underline{x}, \underline{\lambda}^{(2k)}, K^{(2k)})$ s.t. $\underline{x} \in X$. Hence $d_0(\underline{\lambda}^{(2k)} + 2K^{(2k)}\ \underline{h}(\underline{x}^{(2k)}))$ and its gradient are known.

Step 4: Approximate $d_0(\underline{\lambda})$ by fitting a cubic through $(\underline{\lambda}^{(2k)}, d_0(\underline{\lambda}^{(2k)}))$ and $(\underline{\lambda}^{(2k)} + 2K^{(2k)}\ \underline{h}(\underline{x}^{(2k)}), d_0(\underline{\lambda}^{(2k)} + 2K^{(2k)}\ \underline{h}(\underline{x}^{(2k)})))$. Let the estimated step size be $\alpha^{(2k)}$.

Step 5: Set the final step choice $\tilde{\alpha}^{(2k)}$ according to the following rule:

$$\tilde{\alpha}^{(2k)} = \begin{cases} 4K^{(2k)} & \text{if } 4K^{(2k)} \leq \alpha^{(2k)} \\ \alpha^{(2k)} & \text{if } 2K^{(2k)} \leq \alpha^{(2k)} < 4K^{(2k)} \\ 2K^{(2k)} & \text{if } \alpha^{(2k)} \leq 2K^{(2k)} \end{cases}$$

Step 6: Set $\underline{\lambda}^{(2k+1)} = \underline{\lambda}^{(2k)} + \tilde{\alpha}^{(2k)}\ \underline{h}(\underline{x}^{(2k)})$, k = k + 1, and go to Step 1.

In Algorithm 5.1 , cubic fit is used every second iteration, i.e., after every even iteration. The Hestenes-Powell step size rule is used in every odd iteration. In Step 5, the estimated step size $\alpha^{(2k)}$ is bounded within $[2K^{(2k)}, 4K^{(2k)}]$ because the rate of convergence analysis by Bertsekas shows such an interval provides linear local convergence.

5.3.3 Updating the Penalty Constant K

As discussed in Section 3.6, the penalty constant K must be sufficiently large to assure convergence. Another view of this condition on K is that K must be greater than some lower bound K* in order that a local convex structure exists. The latter condition, in turn, implies that local duality theory can be used. Furthermore, as K increases, the dual convergence rate improves, provided the initial value of $\underline{\lambda}$ is close enough to the optimal Lagrange multiplier vector $\underline{\lambda}^*$. In practice, however, it is difficult, if not impossible, to determine K*. If K is too large, the primal problem may be difficult to solve because of ill-conditioning. This situation is particularly likely if the initial $\underline{\lambda}$ is not in the neighborhood of $\underline{\lambda}^*$. It seems that the only practical recourse is to start with a modest value of K, and let it increase in some way so that both the convergence and the fast rate of convergence promised by the method of multipliers can be attained with high probability. This approach seems to be not different from the ordinary penalty method. There is, however, a crucial difference - the consolation that K does not have to approach ∞ for convergence to occur.

Two rules for choosing the sequence $\{K^{(k)}\}$ are considered here. Their performance in numerical experiments is detailed in Section 5.4. The two rules are stated below.

Rule 1: $K^{(k+1)} = K^{(k)}$ if $\| \underline{h}(\underline{x}^{(k+1)}) \| < \theta \| \underline{h}(\underline{x}^{(k)}) \|$, $\theta \leq 1.0$.

Else, $K^{(k+1)} = \phi K^{(k)}$, $\phi > 1.0$.

Rule 2: $K^{(k+1)} = \phi^k K$, $\phi > 1.0$.

Rule 1 increases the penalty constant only when a prescribed rate of convergence is not met. Rule 2 divorces the increase in K from the current

condition of the problem. In both rules, the penalty constant is increased in a geometric fashion. The geometric increase, however, is a convenient arbitrary realization of the requirement that there should be an increase.

5.4 Numerical Experiments

5.4.1 Introduction

The schemes discussed in Section 5.3 for updating $\underline{\lambda}$ and K are combined and tested in a series of numerical experiments. The experiments are designed to compare the convergence and the rate of convergence of each of the four possible combinations of updating rules. The numerical study also looks at the sensitivity of each scheme's performance to the key varying parameter of the updating rule for K.

Section 5.4.2 explains the software implementation and other related details. The test functions used in the study are given in Section 5.4.3 while the experiment plan is described in Section 5.4.4. The results of the experiments are presented and discussed in Section 5.4.5.

5.4.2 Software Details

The software needed to perform the numerical study requires the computer code SP developed in Chapter IV and an additional subroutine incorporated into the code. The functions of this subroutine are to monitor the violations of the equality constraints and, if required, to update $\underline{\lambda}$ and/or K. At the entry to this subroutine, the subproblem (π_1) has been approximately solved. Since the objective function of (π_1) is the augmented Lagrangian $\ell(\underline{x}, \underline{\lambda}, K)$ which involves all the equality constraints, these constraints need not be calculated inside the subroutine itself. They are all implicitly calculated in the last evaluation of

$\ell(\underline{x}, \underline{\lambda}, K)$ prior to the call of the subroutine.

The violations of the equality constraints $h_k(\underline{x}) = P_{p+k}(\underline{x}) - Q_{p+k}(\underline{x}) - 1$, $k = 1, 2, \ldots, q$, can be measured by various distance functions. For the study of this chapter, the measure of the equality constraint violations is given by the Euclidean distance function $\delta(N)$ defined as

$$\delta(N) = [\sum_{k=1}^{q} h^2_k(\underline{x}^{(N)})]^{1/2} \tag{5.11}$$

where N is the dual iteration count and $\underline{x}^{(N)}$ is the solution of the Nth subproblem (π_1). The equality constraints are said to be satisfied if $\delta(N) \leq \varepsilon_{EQ}$, where ε_{EQ} is a prescribed small positive number. Another possible measure is the Chebyshev distance $\max_{k=1,\ldots,q} |h_k(\underline{x})|$ used by Powell [26]. However, the Euclidean distance yields an error region smaller than that of the Chebyshev distance.

Two other important tolerances are the tolerance for terminating the subproblem (π_1) and the feasibility tolerance of the inequality constraints. Both are defined as in Section 4.4.5. Computational experience acquired during the preparatory phase of the experiments revealed that if the threshold for terminating the subproblem is small (e.g. $\leq 10^{-4}$) and uniform for all (dual) iterations, an inordinate amount of computing time would be consumed, especially in the first couple of iterations, without improving much the overall performance. Faster convergence could be achieved with earlier updating of $\underline{\lambda}$ and/or K. Hence a heuristic rule to accelerate convergence through controlling the threshold for terminating a subproblem is included in the subroutine. The rule can be described as follows: Let the threshold for terminating a subproblem be ε_{CGP} and the iteration count be k. Step 0: Set $\varepsilon^{(o)}_{CGP} = 10^{-2}$.

Step 1: If $\delta(k) \leq 10^{-1}$ or $k \geq 2$, $\varepsilon_{CGP}^{(k)} = \varepsilon_{CGP}^{(o)}/10.0$. Step 2: If $\delta(k) \leq 10^{-(1+j)}$, $1 \leq j \leq 5$, $\varepsilon_{CGP}^{(k)} = \varepsilon_{CGP}^{(k)}/10^{j}$. The rule links the reduction of the threshold to the closeness of the approximate solution to the true solution.

5.4.3 Test Problems

Four signomial programs of varying sizes serve as the test problems in the numerical experiments. The statistics of the problems are summarized in Table 5.1. The details of each problem are described in this section. Note that the starting value of $\underline{\lambda}$ in all cases is the zero vector with the appropriate dimension.

Test Problem	No. of Variables	No. of Ineq. Const.	No. of Eq. Const.	Total No. of Terms*	Degree of Difficulty
A	2	1	3	5	12
B	4	1	1	9	4
C	3	2	2	20	16
D	4	1	2	30	25

* Excludes lower and upper bounds on variables.

Table 5.1 Summary of Characteristics of Test Problems.

The following are the four test problems:

Test Problem A

$$\min \quad g_o(\underline{x}) = x_1 x_2^{-2} - 0.5\, x_2^{-2}$$

$$\text{s.t.} \quad g_1(\underline{x}) = 0.5\, x_1^{-1.0} x_2 \leq 1.0$$

$$g_2(\underline{x}) = 2x_1^2 + 2x_2^2 + 5.0 - 6x_1 - 2x_2 = 1.0$$

$$g_3(\underline{x}) = (2/3)x_1 + (1/3)x_2 = 1.0$$

$$g_4(\underline{x}) = 4x_1^2 + x_2^2 + 10.0 - 12x_1 - 2x_2 = 1.0$$

$$0.01 \leq x_o \leq 4.0$$

$$0.01 \leq x_j \leq 10.0, \quad j = 1, 2$$

Solution: $x_o = 0.5,\ x_1 = 1.0,\ x_2 = 1.0$

$$\lambda_1 = 0.48793,\ \lambda_2 = 0.075643,\ \lambda_3 = 0.0189$$

Starting point: $x_o = 0.5,\ x_1 = 0.5,\ x_2 = 1.0$

Test Problem B

$$\min \quad g_o(\underline{x}) = 2.0 - x_1 x_2 x_3$$

$$\text{s.t.} \quad g_1(\underline{x}) = 0.25\, x_3 + 3.75\, x_2 x_3 + 0.375\, x_3 x_4 \leq 1.0$$

$$g_2(\underline{x}) = x_1 + 2x_2 + 2x_3 + 1.0 - x_4 = 1.0$$

$$0.01 \leq x_o \leq 2.0$$

$$0.01 \leq x_j \leq 1.0, \quad j = 1, 2, 3$$

$$0.01 \leq x_4 \leq 2.0$$

Solution: $x_o = 52/27,\ x_1 = 2/3,\ x_2 = 1/3,\ x_3 = 1/3,\ x_4 = 2.0$

$$\lambda_1 = 1/9$$

Starting point: $x_o = 2.0,\ x_1 = 0.5,\ x_2 = 0.5,\ x_3 = 0.25,\ x_4 = 1.5$

Test Problem C

$$\min \quad g_o(\underline{x}) = x_1^{0.6}x_2 + x_2x_3^{-0.5} + 15.98x_1 + 9.0824x_2^2 - 60.72625x_3$$

$$\text{s.t.} \quad g_1(\underline{x}) = x_1x_2^{-2} + 1.48 - x_2^{-2}x_3 \leq 1.0$$

$$g_2(\underline{x}) = x_1^{0.25}x_3 + x_2^2 + 6.75 - x_1^{0.5}x_3^{2.0} \leq 1.0$$

$$g_3(\underline{x}) = x_1^2 + 4.0x_2^2 + 2.0x_3^2 - 57.0 = 1.0$$

$$g_4(\underline{x}) = x_1x_2^{-1}x_3^{2.5} + x_2x_3 - x_2^2 - 15.55 = 1.0$$

$$1.0 \leq x_o \leq 1000.0$$

$$0.1 \leq x_j \leq 10.0, \quad j = 1, 2, 3$$

Solution: $x_o = 319.4$, $x_1 = 1.0$, $x_2 = 2.5$, $x_3 = 4.0$

$\lambda_1 = -3.0$, $\lambda_2 = 1.5$

Starting Point: $x_o = 558.0$, $x_1 = 2.0$, $x_2 = 2.0$, $x_3 = 8.0$

Test Problem D

$$\min \quad g_o(\underline{x}) = 3x_1^2 + 2x_2^2 + x_3^2 + x_4^2 + 7x_1 + 565 - 39x_2 - 17x_3 - x_4$$

$$\text{s.t.} \quad g_1(\underline{x}) = (1/6)x_1^2 + (1/18)x_2^2 + (1/9)x_4^2 + (1/18)x_1 - (1/9)x_3^2 - (1/18)x_2 \leq 1.0$$

$$g_2(\underline{x}) = x_1^2 + x_3^2 + 2.5x_4^2 + x_3 + 9.5 - 4x_2^2 - x_4 = 1.0$$

$$g_3(\underline{x}) = x_2^2 + 3x_3^2 + x_4^4 + x_2 - x_1^2 - x_1 - x_4 - 2.0 = 1.0$$

$$1.0 \leq x_o \leq 1{,}000.0$$

$$0.1 \leq x_j \leq 10.0, \quad j = 1, \ldots, 4$$

Solution: $x_o = 505.0$, $x_1 = 2.0$, $x_2 = 2.0$, $x_3 = 1.0$, $x_4 = 1.0$

$\lambda_1 = -1.0$, $\lambda_2 = 3.0$

Starting Point: $x_o = 700.0$, $x_1 = 1.0$, $x_2 = 1.0$, $x_3 = 3.0$, $x_4 = 2.0$

In Test Problem A, the inequality constraint $g_1(\underline{x}) \leq 1.0$ is loose at the solution. The same is true with the constraint $g_1(\underline{x}) \leq 1.0$ of Test Problem D. All the other inequality constraints of all four test problems are tight at the quoted solutions.

5.4.4 Experiment Plan

Four schemes are considered in the experiments. The first two, Schemes I and II, result from combining the Hestenes-Powell step size rule and Rules 1 and 2, respectively, for updating K. Schemes III and IV are the combination of the same two rules for updating K and the use of Algorithm 5.1 to obtain the step size. When Rule 1 is used to update K, the initial values $K^{(o)}$ tested are 0.1, 1.0, and 10.0. The reduction factor θ is 1.0 while the expansion factor ϕ is set to 2.0. When Rule 2 is used, the values of ϕ are 2.0, 4.0, and 8.0. The parameter K_o is set to 1.0.

Of the four schemes, Scheme I is the most similar to the algorithm proposed by Powell [26]. There are, however, significant differences. The first is that while only either $\underline{\lambda}$ or K, but not both, is updated in Powell's algorithm, both can be updated in the same iteration in Scheme I. The second difference is the definition of the norm of $\underline{h}(\underline{x})$ as explained in Section 5.4.2. Finally, Powell assumes that $M^{(k)}$ is a diagonal matrix not necessarily equal to I, the identity matrix. Because of this more general assumption, not one but a maximum of $q \geq 1$ numbers have to be updated.

To carry out the numerical experiments, two performance measures need to be defined. The first is the convergence measure while the second is for the rate of convergence. In this study, overall convergence is attained if $\delta(N) \leq \varepsilon_{EQ}$, provided the Nth subproblem satisfies

the convergence criterion of Section 4.4.5 as modified by the heuristic rule explained in Section 5.4.2. An alternative convergence measure is to stop if $\delta(N) < \varepsilon_{EQ}$ and $|d_{K(N)}(\underline{\lambda}^{(N)}) - d_{k(N-1)}(\underline{\lambda}^{(N-1)})|/|d_{K(N-1)}(\underline{\lambda}^{(N-1)})| \leq \varepsilon_L$. Both ε_{EQ} and ε_L are small positive numbers.

Choosing a performance measure to reflect the rate of convergence requires some care. A first candidate is the number of dual iterations (i.e. updatings of $\underline{\lambda}$) before convergence is attained. This measure can be viewed as the number of evaluations of the dual functional $d_K(\underline{\lambda})$ and its gradient. While this measure may indicate the dual rate of convergence, it is not a fair indicator of the computational effort required to achieve a given rate. In a primal unconstrained optimization problem, a function evaluation requires the same effort at any point of the iterative process. This is not true with the evaluation of the unconstrained dual functional. In the dual case, each evaluation is a minimization problem whose completion depends on many factors such as the proximity of the current test point to the solution, the accuracy required, and the tolerance levels. A second candidate is the number of linearly constrained nonlinear programs solved. This measure, while it is an improvement over the first, nevertheless is still unacceptable because each such problem requires different computational effort. The final choice adopted is the CPU time measured from the confirmation of a feasible solution to the satisfaction of the convergence criterion[1]. The time used for accepting problem data and checking/achieving feasibility is common to all schemes and hence is ignored. In the literature on numerical studies of optimization algorithms, the chief objection to the

[1] For the rest of this chapter, this time period is simply referred to as CPU time.

use of CPU time is that its significance varies with the computer code used and with the computer system. But since these two factors are constant in this study, the use of CPU time seems to be the most meaningful measure for rate of convergence.

Throughout the experiments the following tolerance levels are observed: $\varepsilon_{EQ} = 10^{-4}$; $\varepsilon_{CGP}^{(o)} = 10^{-2}$; $\varepsilon_{IN} = 10^{-6}$. (Recall that ε_{IN} is the feasibility tolerance for inequality constraints.) Nonconvergence is defined by the following condition:

CPU time $\geq$ 3 sec or number of dual iterations $\geq$ 20.

5.4.5 Results and Discussion

The performance results of solving all four test problems by each of the four schemes are presented in Tables 5.2(a)-(d). In each box of each table, the top number is the CPU time in seconds; the middle number, the number of linearly constrained nonlinear programs solved; and the bottom one gives the number of dual iterations needed to satisfy the convergence criterion. The latter two criteria are included for completeness. Their lack of consistent correlation with computational effort is confirmed by the numerical data.

Overall, the experiments verify the convergence of the algorithms proposed in Chapters III and IV, irrespective of which scheme is used to update $\underline{\lambda}$ and K. In all cases except those noted in Table 5.2, convergence is attained in the sense that the satisfaction of the convergence measure implies a final primal solution of reasonable accuracy. In this study, the accuracy is within at least 10^{-3} of the true solution. Even in those few cases that were terminated because of likely excessive computation, the approximate solution progresses steadily but too slowly towards the theoretical solution. Similarly, when Problem D is solved

Problem \ $K^{(0)}$	0.1	1.0	10.0
A	0.651 (i)	0.559 30 10	0.436 18 5
B	2.019 89 21	1.217 46 11	0.820 48 8
C	1.009 49 9	0.392 21 5	1.637 79 8
D	(ii)	1.434 33 15	2.418 68 12

(a) Scheme I

Problem \ ϕ	2.0	4.0	8.0
A	0.423 18 5	0.448 18 4	0.727 24 5
B	0.756 28 7	0.579 23 4	0.455 15 3
C	0.363 17 3	0.366 17 3	0.371 16 3
D	1.156 29 12	0.675 17 6	0.681 17 6

(b) Scheme II

(i) Terminated after 20 dual iterations.
(ii) Terminated after 3 seconds of CPU time.

Table 5.2 Performance results of Schemes I - IV

Problem \ $K^{(0)}$	0.1	1.0	10.0
A	1.081 (i)	0.621 30 10	0.446 18 5
B	(ii)	1.877 61 18	0.936 54 5
C	1.046 51 10	0.393 21 5	(ii)
D	(ii)	1.43 37 15	2.251 62 9

(c) Scheme III

Problem \ ϕ	2.0	4.0	8.0
A	0.541 20 5	1.06 30 6	0.722 24 5
B	0.961 26 5	0.781 24 4	0.543 15 3
C	0.350 17 3	0.354 17 3	0.361 16 3
D	0.605 16 * 6	0.494 13 * 5	0.415 13 * 5

(d) Scheme IV

(i) Terminated after 20 dual iterations.
(ii) Terminated after 3 seconds of CPU time.
(*) Approaching but did not reach primal solution when dual convergence was satisfied.

Table 5.2 Performance results of Schemes I - IV

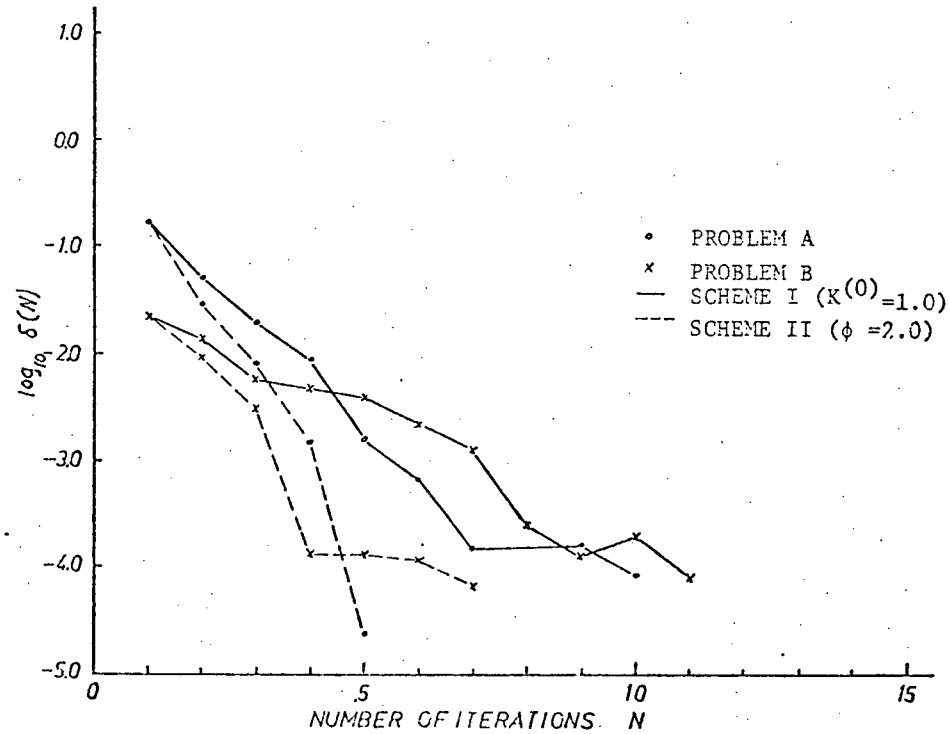
by Scheme IV, termination occurs because the equality constraints are satisfied, although the primal solution is still approaching but not sufficiently near the true solution. In this case, the dual rate is much faster than the primal rate of convergence. Some likely reasons for the discrepancy between the two rates are the inexactness in solving the subproblem and the poor performance of the primal algorithm in this particular problem. The global convergence of $\underline{\lambda}$ to a region near the optimal Lagrange multiplier $\underline{\lambda}^*$ is satisfactory in all cases. However, the local convergence to $\underline{\lambda}^*$ itself is sensitive to the tolerances, hence accuracy, of the primal computation as well as to the step size α.

While it is essential to confirm experimentally the convergence of the proposed algorithms, a major purpose of the experiments is to assess the effect of the updating rules of $\underline{\lambda}$ and K on the rate of convergence. To this end, the data of Table 5.2 reveal significant information. The foremost is that by far, the preferred scheme is Scheme II using the Hestenes-Powell choice for α and a monotonic geometric increase of K. From the convergence viewpoint, both step size choices are almost equally effective. However, the Hestenes-Powell updating formula is preferred for its simplicity, less demand for computation, and independence of the numerical accuracy of other quantities. Choosing the step size by cubic fit does not yield the benefit commensurate with the extra computational work. As for the updating rule of K, Rule 2 relying on increasing K according to a geometric progression is definitely superior to Rule 1 in which K is increased only if $\|\underline{h}(\underline{x})\|$ fails to decrease at a specified rate. The choice of Rule 2 is based on two related reasons. The first is that independent of how the step size is selected, Rule 2 generally needs far less CPU time. This observation can be checked by

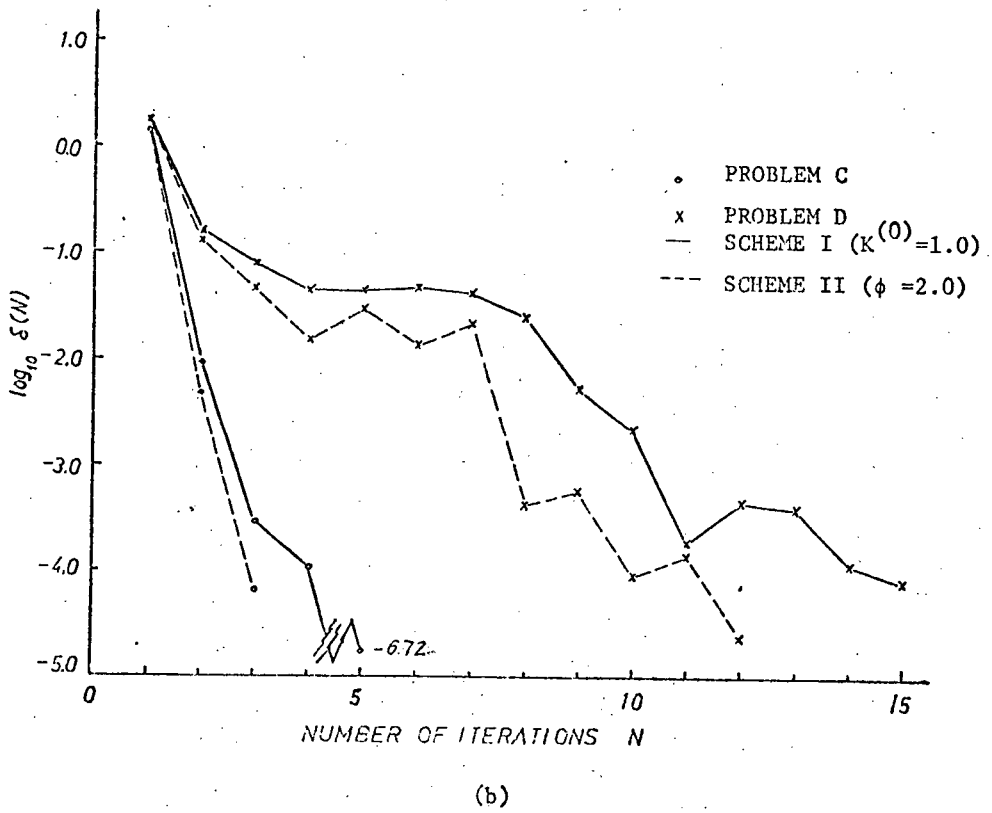
comparing Tables 5.2(a) and (b), and Tables 5.2(c) and (d). The second reason is that Rule 2 almost invariably yields a fast monotonic decrease in $\delta(N)$ (or $\|\underline{h}(\underline{x})\|$). On the other hand, using Rule 1 seems more likely to have nondecrease or increase in $\delta(N)$. When such a condition exists, it necessitates solving more subproblems.

Figure 5.1 gives a specific but typical example of the behavior of $\delta(N)$ as a function of the dual iteration count N. The figure, in which log $\delta(N)$ is plotted against N, is derived from the results of solving the test problems by Scheme I ($K^{(o)} = 1.0$, $\phi = 2.0$, $\theta = 1.0$) and Scheme II ($K_o = 1.0$, $\phi = 2.0$). The corresponding results from Schemes III and IV are not shown because updating $\underline{\lambda}$ by cubic fit is not as promising. Note that for the indicated parameters of Schemes I and II, each problem has the same starting subproblem. The step size choice is common to both. Also, whenever K has to be updated, it is increased by the same factor $\phi = 2.0$. Hence there is a fair basis for comparing the two updating rules for K. Let $\delta_I(N)$ and $\delta_{II}(N)$ be the function $\delta(N)$ obtained when Scheme I and Scheme II are used respectively. Several useful observations about $\delta_I(N)$ and $\delta_{II}(N)$ can be deduced from Fig. 5.1. The first is that in all cases, $\delta_{II}(N) \leq \delta_I(N)$. Furthermore, $\delta_{II}(N)$ tends to decrease faster than $\delta_I(N)$. The two points together suggest that not only does Scheme II lead to an earlier convergence when $\varepsilon_{EQ} = 10^{-4}$ (as it is the case in the experiments), the scheme also consistently finishes earlier than Scheme I for all values of ε_{EQ}. This means that whether an accurate and not too accurate solution is sought, Rule 2 should still be preferred.

In both Schemes I and III, the experimental data attest to the expected sensitivity of the rate of convergence to the value $K^{(o)}$. As $K^{(o)}$ is larger, the rate generally improves. The improvement is reflected in less dual iterations and a faster reduction of $\|\underline{h}(\underline{x})\|$. The same



(a)



(b)

Fig. 5.1 Typical plot of log $\delta(N)$ against the dual iteration count N.

effect is true for a larger ϕ in Schemes II and IV. In both instances, the CPU time usually decreases too. There are cases, however, in which the CPU time increases. This phenomenon is not caused by ill-conditioning in the primal problem. Instead it can be explained as follows. When $K^{(o)}$ (or ϕ) is selected to be too large, the dual convergence rate is higher and $\|\underline{h}(\underline{x})\|$ tends to decrease much faster. This means that $\|\underline{h}(\underline{x})\|$ becomes small early in the iterative process. Because of the heuristic rule controlling the tolerance ε_{CGP}, a small value of $\|\underline{h}(\underline{x})\|$ effectively implies accurate minimization of the primal subproblem. Hence more computational work is needed.

To avoid poor convergence or excessive effort to solve subproblems, prudent choices of the initial value $K^{(o)}$ (K_o in Schemes II and IV) and the factor ϕ must be made. It seems that for suitably scaled constraints, $K_o = 1.0$ and $\phi = 4.0$ are quite acceptable. The emphasis here is on the suitable scaling of the equality constraints. That these constraints should be properly scaled is implied by assuming $M^{(k)} = I$ in Section 5.3.2. Hence the scaling of the equality constraints and that of $\underline{\lambda}$ are equivalent. Since $K_o = 1.0$, it is advisable to scale $h_k(\underline{x})$, $\forall k$, such that initially, $|h_k(\underline{x})| \sim 1.0$, where "$\sim$" means "in the order of". Similarly, the original objective function should not be too much larger or smaller than $K^{(o)} \sum_k h_k^2(\underline{x})$.

In summary, the Hestenes-Powell step size rule and the updating of K by a geometric progression form an effective scheme to implement Step 3 of Algorithm 3.2 for solving signomial programs with equality constraints. To achieve the high convergence rate promised by this scheme, care must be taken to scale the equality constraints and the original objective function. Furthermore, approximate minimization of the

first few subproblems can save considerable computing time without diminishing the power of the algorithm.

VI. APPLICATIONS IN ENGINEERING DESIGN

6.1 Optimum Design of Statically Indeterminate Pin-Jointed Structure

6.1.1 Introduction

In the last decade the use of optimization techniques in structural design has been recognized as a more powerful approach than traditional direct design methods. The optimization approach can handle more design constraints and still produce the most economic solution within the limits set by the constraints. The applicability of the optimization approach spans the whole structural design process ranging from the choice of the topology of the structure to the selection of the member sizes of structures with a defined shape. This section is devoted to the optimum design of a class of structures with a fixed shape. As it will be made clear in the problem formulation, signomial equality constraints arise naturally in this class of structure design problems. Furthermore, both the objective function and the inequality constraints are signomials. It follows that the algorithms developed in this thesis can be conveniently employed to yield the optimum design.

The usefulness of the proposed algorithms extends beyond the group of structures considered in detail in this section. There are structures of other types (e.g. rigidly jointed structures) whose design requires the satisfaction of signomial equality constraints. Constraints of similar type also appear in different formulations of optimum structural design. It is, however, beyond the scope of this section to consider them.

6.1.2 Problem Formulation

Consider a statically indeterminate pin-jointed N-member structure with a specified geometry. It is desired to choose the most economical

cross-sectional area of each member such that some structural constraints are satisfied. In order to obtain the optimum design wanted, both the objective function and the constraints need to be established.

The objective function is the cost of the structure. Let ℓ_i and A_i be, respectively, the length and the area of the ith member. Then, if γ_i is the weight density of the ith member, the member's weight is $\gamma_i \ell_i A_i$. Suppose that the cost of the ith member is c_i/unit weight. The desired objective function becomes

$$C = \sum_{i=1}^{N} c_i \gamma_i \ell_i A_i \tag{6.1}$$

The structural constraints are of three types: stress constraints, deflection constraints, and compatibility constraint. The first two types are imposed for safety reasons and are in the form of inequalities. The compatibility constraints ensure that the topology of the structure remains the same during and after the application of external forces. These constraints are equalities. To derive all the constraints, it is first necessary to determine the member forces $\underline{P} = [P_1, P_2, \ldots, P_N]'$. Assume that there are m loading points with the external load vector $\underline{L}_b = [L_1, L_2, \ldots, L_m]'$ acting at these points. In a statically determinate structure, $\underline{P}$ is linearly related to and uniquely determined by $\underline{L}_b$. That is,

$$\underline{P} = B_b \underline{L}_b \tag{6.2}$$

where B_b is known as the load transformation matrix. However, in a statically indeterminate or hyperstatic structure, $\underline{L}_b$ is not sufficient to specify $\underline{P}$. The forces in the redundant members have to be included. In this case,

$$\underline{P} = B_b \underline{L}_b + B_r \underline{L}_r \tag{6.3}$$

where $\underline{L}_r = [R_1, R_2, \ldots, R_Q]'$, R_k being the force in the redundant member k, and B_r is the force transformation matrix relating $\underline{P}$ to $\underline{L}_r$.

The stress in a member is defined as the ratio of the member's force to its cross-sectional area. For hyperstatic structures, the stress constraints are therefore given by

$$A\underline{P} = AB_b\underline{L}_b + AB_r\underline{L}_r \leq \underline{\sigma} \tag{6.4}$$

where A is an NxN diagonal matrix whose ith diagonal entry is $1/A_i$. From (6.4), it can be seen that the stress constraint on member i is a signomial in A_i, R_1, R_2, $\ldots$, R_Q.

To derive the deflection constraints, the deflection vector $\underline{X}$ must first be related to the load vector $\underline{L}$. It can be shown [19] that in general for elastic structures,

$$\underline{X} = F\underline{L}$$

where F, known as the overall flexibility matrix, is given by B'fB. B is the force transformation matrix and f is an NxN diagonal matrix whose ith entry is ℓ_i/E_iA_i, E_i being the Youmg's modulus of elasticity of member i. In the case of hyperstatic structures, $\underline{X}$ can be partitioned into $\underline{X}_b$, the deflections due to external loads and $\underline{X}_r$, the deflections due to the redundant forces $\underline{L}_r$. Hence

$$\begin{bmatrix} \underline{X}_b \\ \underline{X}_r \end{bmatrix} = \begin{bmatrix} B_b'fB_b & B_b'fB_r \\ B_r'fB_b & B_r'fB_r \end{bmatrix} \begin{bmatrix} \underline{L}_b \\ \underline{L}_r \end{bmatrix} \tag{6.5}$$

The deflection constraints impose an upper bound on some or all of the components of $\underline{X}_b$. Mathematically, these constraints are

$$B_b'fB_b\underline{L}_b + B_b'fB_r\underline{L}_r \leq \underline{\delta} \tag{6.6}$$

Again, the ith inequality is a signomial inequality constraint in A_i, R_i, ..., R_Q.

In (6.4) and (6.6), the redundant forces $\underline{L}_r$ are unknown. For this reason, $\underline{X}_r$ must be required to be null. Otherwise, the structure would assume a different topology after the application of $\underline{L}_b$. Thus,

$$B_r' f B_b \underline{L}_b + B_r' f B_r \underline{L}_r = 0 \qquad (6.7)$$

This equality constraint is called the compatibility constraint. (6.7) is clearly a set of signomial equality constraints.

In summary, the design problem is to minimize the cost function given by (6.1) subject to the constraints (6.4), (6.6), and (6.7).

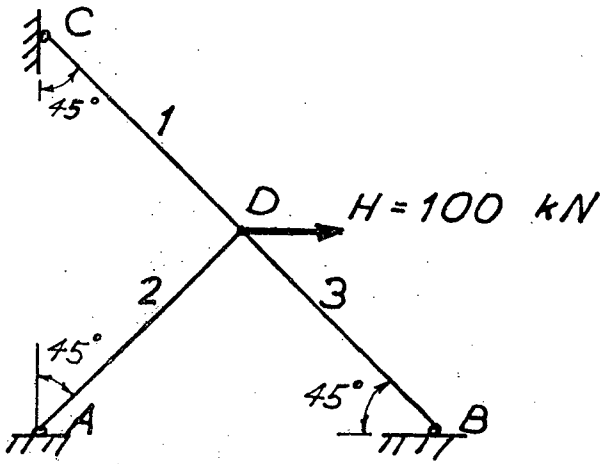
6.1.3 Example

Consider the pin-jointed structure shown in Fig. 6.1(a) and adapted from [19]. Suppose that the horizontal and vertical deflections at D are both limited to 4 mm, and that the numerical value of the stress in any member is not to exceed $10^6 kN/m^2$. All three members are one meter in length. Let A_i be the cross-sectional area of member i. It is required to choose the cross-sectional areas A_1, A_2, and A_3 such that the structure has minimum volume. Assume that Young's modulus of elasticity E is 207 kN/mm^2.

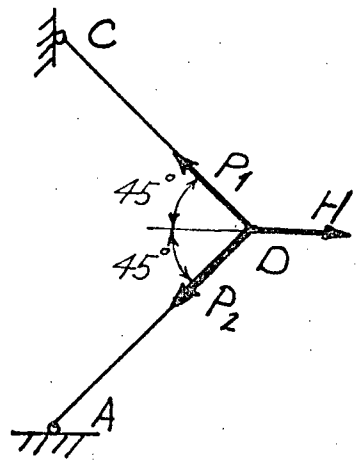
The first step is to derive the constraints (6.4), (6.6), and (6.7). To construct the matrix B_b, the redundant force is removed. In this example, member 3 may be designated as the redundant member. With member 3 removed, the structure becomes that shown in Fig. 6.1(b). Resolution of the forces at D gives

$$B_b = [\sqrt{2}/2 \quad \sqrt{2}/2 \quad 0]' \qquad (6.8)$$

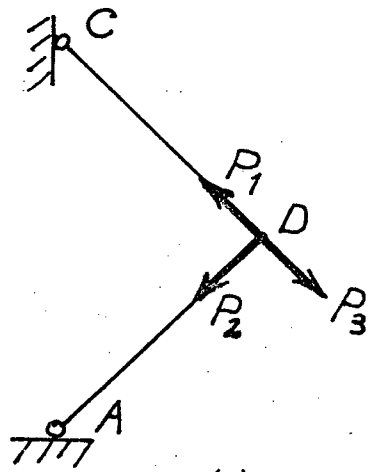
Next remove the external load H and subject joint D to force P_3 as shown



(a)



(b)



(c)

Fig. 6.1 The hyperstatic pin-jointed structure of the example.
(a) Frame and loading (b) Basic frame with external load (c) Basic frame subject to redundant force.

in Fig. 6.1(c). Again, resolving the forces at D yields

$$B_r = [1 \quad 0 \quad 1]' \tag{6.9}$$

The flexibility matrix f is

$$f = \begin{bmatrix} 1/EA_1 & 0 & 0 \\ 0 & 1/EA_2 & 0 \\ 0 & 0 & 1/EA_3 \end{bmatrix} \tag{6.10}$$

while the load vector $\underline{L}_b$ and the redundant force vector $\underline{L}_r$ are simply

$$\underline{L}_b = H, \qquad \underline{L}_r = P_3 \tag{6.11}$$

It follows from (6.4) that the stress constraints are

$$\begin{aligned} &\text{(a)} \quad (\sqrt{2H}/2\sigma)A_1^{-1} \quad (1/\sigma)A_1^{-1} \quad P_3 \leq 1 \\ &\text{(b)} \quad (\sqrt{2H}/2\sigma)A_2^{-1} \leq 1 \\ &\text{(c)} \quad (1/\sigma)A_3^{-1} P_3 \leq 1 \end{aligned} \tag{6.12}$$

where $\sigma = 10^6$ kN/m^2. From (6.6), (6.8)-(6.11), the deflection constraint may be written as

$$(H/2E\delta)A_1^{-1} + (H/2E\delta)A_2^{-1} + (\sqrt{2}/2E\delta)A_1^{-1} \quad P_3 \leq 1 \tag{6.13}$$

Young's modulus of elasticity E is assumed to be 207 kN/mm^2 and the deflection tolerance δ is set at 0.004 m. The compatibility constraint defined in (6.7) is given by

$$(\sqrt{2H}/2)A_1^{-1} + A_1^{-1} P_3 + A_3^{-1} P_3 = 0 \tag{6.14}$$

Finally, the areas A_1, A_2, and A_3 must be nonnegative.

The optimization problem is to minimize the volume $V = A_1 + A_2 + A_3$ subject to the inequality constraints (6.12)-(6.13), the equality constraint (6.14), and the nonnegativity constraint on the areas. To put the problem into a form compatible with the formulation of signomial programs defined in (3.1), several steps need to be undertaken. First,

from (6.14), it can be easily shown that P_3 is nonpositive. But signomial programs have only positive variables. Hence P_3 has to be replaced with $-P_3$. After this substitution, (6.12c) can immediately be dropped since the constraint is loose for all feasible values of A_3 and P_3. The final manipulation is to impose bounds on all the variables. Since signomials are defined over the positive orthant only, all the variables except A_2 need to be bounded from below by a small positive quantity. With the artificial upper bounds imposed and the problem data entered, the final problem is

$$
\begin{aligned}
\min \quad & A_1 + A_2 + A_3 \\
\text{s.t.} \quad & (7.0711 \times 10^{-4})A_1^{-1} - 10^{-6}A_1^{-1}A_4 \leq 1 \\
& (6.0385 \times 10^{-5})A_1^{-1} + (6.0385 \times 10^{-5})A_2^{-1} - \\
& (8.54 \times 10^{-7})A_1^{-1} P_3 \leq 1 \\
& 70.7107A_1^{-1} - A_1^{-1} P_3 - A_3^{-1} P_3 + 1 = 1 \qquad (6.15) \\
& 10^{-8} \leq A_1 \leq 1.0 \\
& 7.0711 \times 10^{-4} \leq A_2 \leq 1.0 \\
& 10^{-8} \leq A_3 \leq 1.0 \\
& 10^{-8} \leq P_3 \leq 1.0
\end{aligned}
$$

Note that the areas are in m^2 while P_3 is in kN.

The numerical example was solved with the computer code SP using Scheme II of Section 5.4.4 to update λ and K. The feasible primal starting point is at $A_1 = A_2 = A_2 = 0.001\ m^2$ and $P_3 = 0.001$ kN, while $\lambda^{(o)} = 0.0$. K_o and ϕ, the two parameters needed for updating K, are 1.0 and 4.0, respectively. Within 7 dual iterations (CPU time = 0.66 sec), the final answer obtained is $V = 0.00141412\ m^3$, $A_1 = 0.000707\ m^2$,

$A_2 = 0.00070711\ m^2$, $A_3 = 10^{-8}\ m^2$, $P_3 = 0.11159$ kN, and $\lambda = 0.6$. The tolerance used are $\varepsilon_{CGP}^{(o)} = 10^{-2}$, $\varepsilon_{IN} = 10^{-5}$, $\varepsilon_{EQ} = 10^{-4}$. Since $A_3 \simeq 0.0$, it can be deduced from the computed solution that the true solution is $A_1^* = 0.00070711$, $A_2^* = 0.00070711$, $A_3^* = 0.0$, and $P_3^* = 0.0$. The last relation is based on the fact that $P_3 = A_3/(A_1 + A_3)$. The discrepancy between P_3 and P_3^* is due to the finite error e in the equality constraint and the smallness of A_1. In fact, it can be shown that for negligible A_3, $P_3 \cong e/A_1$. In the example, $e = 7.8 \times 10^{-5}$.

From the design point of view, how should the optimal solution be interpreted? The most important result is that the proposed design concept should be changed. Instead of a three-member structure, a two-member structure is adequate (and optimal) to meet the design requirements. At the computed solution, the stress constraints are tight while the deflection constraint is loose. This result may prompt a re-examination of the allowable stress and deflection because usually the deflection constraint is more significant.

6.2 Optimization Examples from Chemical Process Engineering

6.2.1 Introduction

The history of applying signomial programming to steady-state chemical process design is almost as old as the optimization method's. Rijckaert [18] gives a good summary of this area's key application papers published in the period 1963-1972. A distinctive feature of the mathematical models used in process design is the frequent presence of signomials in both the cost (or profit) function and the design constraints. This affinity of the process models to signomials is the main reason why process engineering is such a fertile field for possible application of signomial programming. Another notable feature of the models is that

equality constraints are very common. These constraints arise because of mass and energy balance, as well as functional relationships derived empirically or from first principles. In all the related papers referenced by Rijckaert [18], the models containing both equality and inequality constraints are first transformed on the basis of "engineering intuition" into one with either only equality constraints or only inequality constraints. The design problems are then solved by methods accommodating one type of constraints only. In this section, two examples drawn from chemical process design demonstrate how the algorithms of Chapters III and IV can be used to solve the design problems based on the original models with mixed equality and inequality constraints. Because no constraint transformation is required, the physical arguments for equality or inequality constraints are not blurred by algebraic manipulations.

6.2.2 Alkylation Process Optimization

The example used in this section is typical of the models used in seeking the optimum operating conditions for a chemical process in order to maximize a profit function. This section's model of an alkylation process, a common process in the petroleum industry, was first described by Sauer, Colville, and Burwick [55] on the basis of the process relationships given by Payne [56]. The optimization problem was solved by Sauer et al. [55] via the solution of a sequence of linear programs. Later Bracken and McCormick [57] solved the problem using the penalty function method (SUMT). Recently, the same model, after it had been transformed into an inequality-constrained signomial program, was solved by Avriel et al. [12] using the computer code GGP developed by Dembo [16]. The mathematical formulation given below follows that of Bracken and McCormick.

Figure 6.2 shows a simplified process diagram of an alkylation process. The system's main units are the reactor and the fractionator. Olefin feed and isobutane make-up are pumped into the reactor. To catalyze the reaction, fresh acid is added and the spent acid is withdrawn. The hydrocarbon product from the reactor is fed into a fractionator, from the top of which isobutane is recycled back to the reactor. Alkylate product is withdrawn from the bottom of the fractionator. It is assumed that the olefin is pure butylene, that both the isobutane make-up and the isobutane recycle are pure butane, and that the fresh acid strength is 98% by weight.

The process variables and their upper and lower bounds are defined in Table 6.1. The bounds on the variables x_1, ..., x_5 are due to the limitations imposed by the capability of the plant and/or the economic situation under analysis. For example, only 2,000 barrels per

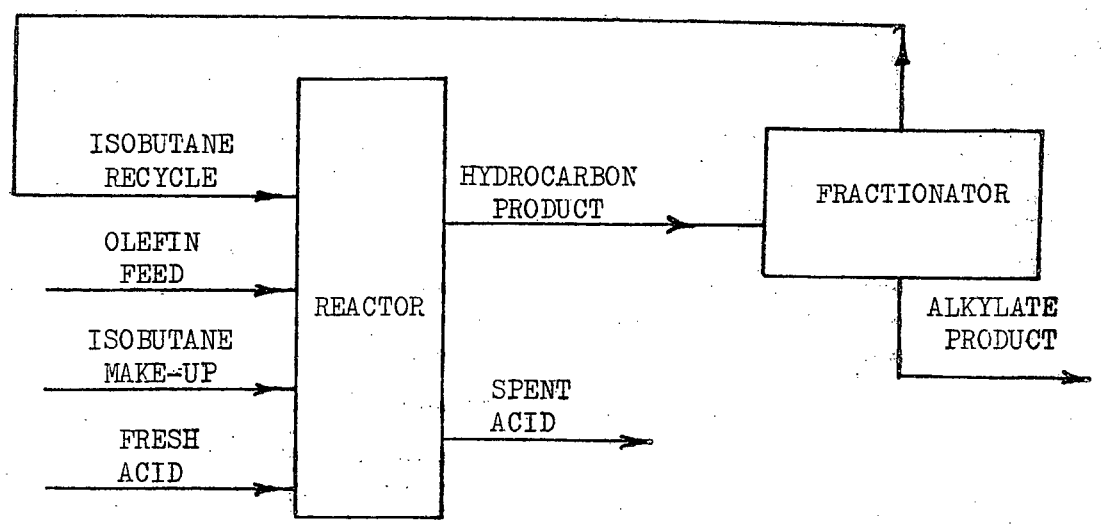


Fig. 6.2 Simplified alkylation process diagram.

	VARIABLE	LOWER BOUND	UPPER BOUND
x_1	olefin feed (barrels per day)	1.0	2000.0
x_2	isobutane recycle (barrels per day)	1.0	16000.0
x_3	acid dilution rate (1000 lb/day)	1.0	120.0
x_4	alkylate yield (barrels per day)	1.0	5000.0
x_5	isobutane make-up (barrels per day)	1.0	2000.0
x_6	acid strength (weight per cent)	85.0	93.0
x_7	motor octane number	90.0	95.0
x_8	external isobutane-to-olefin ratio	3.0	12.0
x_9	acid dilution factor	1.2	4.0
x_{10}	F-4 performance number	145.0	162.0

Table 6.1 The alkylation process problem's variables and their bounds.

	PROFIT AND COST PARAMETER	VALUE	
c_1	alkylate product value	\$0.063	per octane-barrel
c_2	olefin feed cost	\$5.04	per barrel
c_3	isobutane recycle cost	\$0.035	per barrel
c_4	acid addition cost	\$10.00	per thousand pounds
c_5	isobutane make-up cost	\$3.36	per barrel

Table 6.2 Definitions and values of the cost coefficients of the alkylation process problem.

day of olefin feed may be available. The other variables $x_6, \ldots, x_{10}$ have bounds directly related to the process.

The process constraints consist of two groups: statistical relations among some variables within certain operating ranges, and relationships due to material balances. The first group, described by linear or nonlinear regressions, leads to inequality constraints while the latter group is just a set of equality constraints. Each regression relationship is replaced with two inequality constraints, which specify the range for which the regression relationship is valid. Consider the regression equation $Y = f(z)$. This relationship would be expressed in the model as

$$d_\ell Y \leq f(z) \leq d_u Y \tag{6.16}$$

Or

$$f(z) \leq d_u Y$$

$$-f(z) \leq -d_\ell Y \tag{6.17}$$

The deviation parameters d_ℓ and d_u establish the percentage difference of the estimated value from the true value. In this model, the values of the d_ℓ and d_u for each regression relation are set to the same values as those given in [57].

Assuming that the reactor temperature is between 80°F and 90°F, and that the reactor acid by weight percent strength is 85-93%, nonlinear regression analysis relates the alkylate yield x_4 to the olefin feed x_1 and the external isobutane-to-olefin ration x_8 by the following pair of inequalities:

$$(99/100)x_4 \leq 1.12x_1 + 0.13167x_1x_8 - 0.00667x_1x_8^2 \leq (100/99)x_4 \tag{6.18}$$

Similarly, under the same reactor conditions, the motor octane number x_7 is related to x_8 and the acid strength by weight percent x_6 by

$$(99/100)x_7 \leq 86.35 + 1.098x_8 - 0.038x_8^2 + 0.325(x_6-89) \leq (100/99)x_7 \quad (6.19)$$

A linear regression relation expresses the acid dilution factor x_9 as a linear function of the F-4 performance number x_{10}, which in turn is linearly related to x_7 by regression analysis.

$$(9/10)x_9 \leq 35.82 - 0.222x_{10} \leq (10/9)x_9 \quad (6.20)$$

$$(9/100)x_{10} \leq -133 + 3x_7 \leq (100/99)x_{10} \quad (6.21)$$

Mass balance requires equality constraints for the isobutane make-up x_5, the acid dilution factor x_9, and the external isobutane-to-olefin ration x_8. Assuming that the volumetric shrinkage is 0.22 volume per volume alkylate yield, the isobutane make-up x_5 may be determined by a volumetric reactor balance given by

$$x_4 = x_1 + x_5 - 0.22x_4 \quad (6.22)$$

or

$$x_5 = 1.22x_4 - x_1 \quad (6.23)$$

The acid dilution factor x_9 may be derived from an equation expressing acid addition rate x_3 as a function of alkylate yield x_4, acid dilution factor x_9, and acid strength by weight percent x_6. Thus,

$$1{,}000x_3 = x_4x_9x_6/(98 - x_6) \quad (6.24)$$

or

$$x_9 = 98{,}000x_3x_4^{-1}x_6^{-1} - 1{,}000x_3x_4^{-1} \quad (6.25)$$

Finally, by definition,

$$x_8 = (x_2 + x_5)/x_1 \quad (6.26)$$

Combining (6.23) and (6.26), and rearranging yields

$$x_2 = x_1x_8 - 1.22x_4 + x_1 \quad (6.27)$$

The profit function is taken as the value of the output, the

alkylate yield, minus the feed and recycle costs. Other operating costs are assumed to be constant. The total daily profit to be maximized is

$$P = c_1x_4x_7 - c_2x_1 - c_3x_2 - c_4x_3 - c_5x_5 \tag{6.28}$$

where the cost coefficients $c_1, \ldots, c_5$ are as defined in Table 6.2. Also given in the same table are values of the coefficients used in the numerical solution.

To cast the design problem into the format compatible with the formulation of Chapter III, the following are done:

1) The maximization of P is replaced with the minimization of x_o subject to $x_o^{-1}(-P + c) \leq 1$, where c is a sufficiently large positive constant (e.g. 3,000) to insure that $x_o \leq 0$.
2) Each of the constraints (6.18)-(6.21) is rewritten as a pair of signomial inequality constraints.
3) Each of (6.23), (6.25), and (6.27) is stated as a signomial equality constraint. The resulting signomial program is

$$\begin{aligned}
&\min\ x_o \\
&\text{s.t. (a) } x^{-1}(5.04x_1 + 0.035x_2 + 10x_3 + 3.36x_5 + 3{,}000 - 0.063x_4x_7) \leq 1 \\
&\quad\text{(b) } 0.00667x_1x_8^2 + 0.99x_4 + 1 - 1.12x_1 - 0.13167x_1x_8 \leq 1 \\
&\quad\text{(c) } 1.12x_1 + 0.13167x_1x_8 + 1 - 0.00667x_1x_8^2 - 1.0101x_4 \leq 1 \\
&\quad\text{(d) } 0.038x_8^2 + 0.99x_7 - 0.325x_6 - 1.098x_8 - 56.425 \leq 1 \\
&\quad\text{(e) } 0.325x_6 + 1.098x_8 + 58.425 - 0.038x_8^2 - 1.0101x_7 \leq 1 \\
&\quad\text{(f) } 0.222x_{10} + 0.9x_9 - 34.82 \leq 1 \\
&\quad\text{(g) } 36.82 - 0.222x_{10} - 1.1111x_9 \leq 1 \\
&\quad\text{(h) } 134 + 0.99x_{10} - 3x_7 \leq 1 \\
&\quad\text{(i) } 3x_7 - 1.0101x_{10} - 132 \leq 1
\end{aligned} \tag{6.29}$$

(j) $1.22x_4x_5^{-1} - x_1x_5 = 1$

(k) $98{,}000x_3x_4^{-1}x_6^{-1}x_9^{-1} - 1.22x_2^{-1}x_4 = 1$

(l) $x_1x_2^{-1}x_8 - x_1x_2^{-1} - 1.22x_2^{-1}x_4 = 1$

(m) the bounds specified in Table 6.1.

The code SP was used to solve (6.29) with $\underline{\lambda}$ and K updated according to Scheme II. As in Section 6.1, $\underline{\lambda}^{(o)} = (0.0, 0.0, 0.0)'$, $K_o = 1.0$, and $\phi = 4.0$. The optimality and feasibility tolerances are also the same as those in Section 6.1. In the first few computation runs, it was discovered that the dual convergence was not satisfactory in the sense that $\delta(N)$ was either constant for several dual iterations or decreasing very slowly in spite of K being very large (e.g. 10^7). A modification that corrected the situation is to rewrite the equality constraints as

(j) $1.22x_4 - x_1 - x_5 + 1.0 = 1.0$

(k) $98000.0x_3x_6 - 1000.0x_3 - x_4x_9 + 1.0 = 1.0$

(l) $x_1x_8 - x_1 - 1.22x_4 - x_2 + 1.0 = 1.0$

The above equations, however, are poorly scaled. The final scaled equations used are

(j) $1.22x_4 - x_1 - x_5 + 1.0 = 1.0$

(k) $980.0x_3x_6^{-1} - 10.0x_3 - 0.01x_4x_9 + 1.0 = 1.0$

(l) $0.1x_1x_8 - 0.1x_1 - 0.122x_4 - 0.1x_2 + 1.0 = 1.0$

Table 6.3 gives the starting point and the optimal solution computed. Note that the initial point is feasible with respect to the inequality constraints. The solution obtained here is very close but not identical to that of Bracken and McCormick [57]. The slight discrepancy is perhaps due to the primal algorithm used. As can be

expected from the nature of the constraints (6.18)-(6.21), four of the inequality constraints of (6.29) are loose at the optimum.

	STARTING POINT	OPTIMAL SOLUTION
x_0	2200.0	1232.88
x_1	1745.0	1696.78
x_2	12000.0	15805.58
x_3	110.0	54.07
x_4	3048.0	3028.87
x_5	1974.0	1998.44
x_6	89.5	90.11
x_7	92.8	95.0
x_8	8.0	10.49
x_9	3.6	1.56
x_{10}	145.0	153.53

Table 6.3 The initial and optimal solution of the alkylation process problem.

Since the profit P is C - x_0, the maximum profit is deduced to be 1767.11 dollars/day. From Table 6.3, it can be seen that one of the inputs, the isobutane make-up x_5, is at its upper bound. This suggests that the profit may be increased further by allowing more isobutane make-up, the maximum profit is raised by about 5% to 1857.23 dollars/day. To accomplish this, the capacity of the fractionator has to be expanded to yield the isobutane recycle flow of 16593.27 barrels/day.

The computational experience with the alkylation problem offers some valuable insight into how an equality constraint of the form $y = f(\underline{x})$, where $f(\underline{x})$ is a signomial, should be converted to a standard signomial equality constraint $g(\underline{x}) = 1.0$. Such type of equation often arises in input-output relations and in functional relations defining one design variable in terms of other variables. The first way to transform the equation is to write $y^{-1}f(\underline{x}) = 1.0$. This format is usually well scaled provided y is in the proper range. However the factor $y^{-1.0}$ appears in every term of $f(\underline{x})$ and the augmented objective function is very nonlinear in y. The second way is to rewrite the equation as $f(\underline{x}) - y + 1.0 = 1.0$. The advantage of this format is that the augmented cost is quadratic in y. This format, however, is susceptible to poor scaling. If the latter form is to be used, a scale factor β must be introduced such that, for example, initially $\beta|f(\underline{x}) - y| \sim 1.0$. The final form becomes $\beta f(\underline{x}) - \beta y + 1.0 = 1.0$. As long as β is suitably selected, the second format seems to be preferable.

6.2.3 Design of a Heat Exchanger Network

The design of the heat transfer capability of a process varies very widely in complexity. The design task may range from the size specification of a simple heat exchange unit to the optimal structural synthesis of a heat exchanger network. The example in this section is concerned with the optimal choice of the heat transfer areas and the temperature variables in a small heat exchanger network with a known configuration.

Suppose that in an industrial chemical process, a flow denoted as Stream 2 has to be cooled from T_{21}°F to T_{22}°F. It is found that the required heat loss by Stream 2 cannot be completely absorbed by an

existing cooler C through which cooling water is pumped. There are, however, two streams in the process, Streams 1 and 3, which need to be heated and which can be conveniently routed to the vicinity of Stream 2. The two streams are therefore potential sinks for absorbing via heat exchangers part of the heat of Stream 2. Such a scheme, as shown diagrammatically in Fig. 6.3, not only makes it feasible to reduce the temperature of Stream 2 to T_{22}°F, but also may be cheaper than cooling by water alone, assuming the water flow is fast enough. The design goal is to satisfy system constraints and still achieve the optimal trade-off between the exchangers' installation costs and the operating cost of using cooling water. The exchangers are assumed to be the double-pipe type with countercurrent flow.

The design variables are the output temperatures $t_1, \ldots, t_5$ as indicated in Fig. 6.3, and the heat transfer areas A_1 and A_2 of exchangers X1 and X2, respectively. The annual cost is the sum of the exchanger's annual depreciation and the annual cost of cooling by water. The cost of installing a heat exchanger with a heat transfer area A is

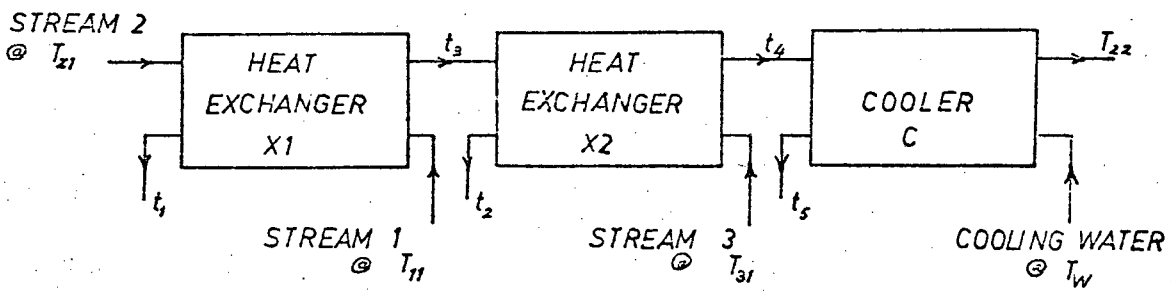


Fig. 6.3 A small heat exchanger network.

generally taken as hA^{γ} dollars [58] for some positive constants h and γ. If the cost is spread over its expected service lifetime of n years, the annual depreciation cost is cA^{γ} dollars/year where $c = h/n$. Because Streams 1 and 3 have different corrosion properties, X1 and X2 need to be constructed with different materials, and hence have different cost parameters c and γ. If the annual cost rate of cooling by water is c_3 dollars/°F, then it follows that the total annual cost $f(A_1, A_2, t_4)$ is

$$f(A_1, A_2, t_4) = c_1 A_1^{\gamma_1} + c_2 A_2^{\gamma_2} + c_3(t_4 - T_{22}) \qquad (6.30)$$

The constraints on the system of Fig. 6.3 consist of four groups:

1) Bounds on the minimum temperature difference (called the approach) between the hot and cold streams of an exchanger;
2) Bounds on the output temperature of the cold streams and on the areas of the heat exchangers;
3) Energy balance equalities relating the input and output temperatures of both streams;
4) Energy balance equalities relating the heat transferred and the heat transfer area of an exchanger.

Since the temperature profile of the hot and cold streams of an exchanger is not known, the approach T_A cannot be explicitly bounded. An approximate bound on T_A may be obtained by bounding the differences $T_{ho} - T_{ci}$ and $T_{hi} - T_{co}$. The subscripts o and i refer to the output temperature and input temperature respectively while the subscripts h and c refer to the hot and cold streams. Hence, from Fig. 6.3, the following inequalities must hold.

$$
\begin{aligned}
&\text{(a)} \quad T_{21} - t_1 \geq \tau_x, \\
&\text{(b)} \quad t_3 - T_{11} \geq \tau_x, \\
&\text{(c)} \quad t_3 - t_2 \geq \tau_x, \\
&\text{(d)} \quad t_4 - T_{31} \geq \tau_x, \\
&\text{(e)} \quad t_4 - t_5 \geq \tau_c
\end{aligned}
\tag{6.31}
$$

τ_x is the minimum allowable approach for the heat exchanger while τ_c is that for the cooler.

Because Streams 1 and 3 are required for other purposes, the temperatures t_1 and t_2 may have to be bounded. It is assumed that $t_1 \leq b_1$ and $a_2 \leq t_2 \leq b_2$. Furthermore, because of heat pollution control, the cooling water cannot be heated beyond T_w°F. Space limitation also demands that the heat exchangers' areas must be within 100 sq. ft.

Let W_i and C_{pi} be, respectively, the flow rate (lb/hr) and the heat capacity (BTU/lb-°F) of Stream i. Also, let the water's flow rate be W_w and its heat capacity be C_{pw}. Then energy balance requires

$$
\begin{aligned}
&\text{(a)} \quad W_2C_{p2}(T_{21} - t_3) = W_1C_{p1}(t_1 - T_{11}) \\
&\text{(b)} \quad W_2C_{p2}(t_3 - t_4) = W_3C_{p3}(t_2 - T_{31}) \\
&\text{(c)} \quad W_2C_{p2}(t_4 - T_{22}) = W_wC_w(t_5 - T_w)
\end{aligned}
\tag{6.32}
$$

Furthermore, the heat transfer rate Q of a heat exchanger can also be expressed as

$$Q = UA\ \Delta T_{log} \tag{6.33}$$

where U is the overall coefficient of heat transfer (BTU/hr-ft^2-°F), A is the area available for heat transfer, and ΔT_{log} is the logarithmic mean temperature difference. In terms of the input and output temperatures of both hot and cold streams, the logarithmic mean temperature

difference is defined as

$$\Delta T_{log} = \frac{(T_{hi} - T_{co}) - (T_{ho} - T_{ci})}{\ln[(T_{hi} - T_{co})/(T_{ho} - T_{ci})]} \tag{6.34}$$

where T_{hi} and T_{ci} are the input temperatures of the hot and cold streams, respectively, and T_{ho} and T_{co} are the output temperatures. Frequently ΔT_{log} is replaced with the arithmetic mean $[(T_{hi} - T_{co}) - (T_{ho} - T_{ci})]/2.0$. This approximation has less than 1% error if $(T_{hi} - T_{co})/(T_{ho} - T_{ci})$ is ≥ 0.7 and less than 10% if $(T_{hi} - T_{co})/(T_{ho} - T_{ci})$ is ≥ 0.33 [59]. For the purpose of this example, the arithmetic mean is used to yield

$$\begin{aligned} &\text{(a)} \quad W_1 C_{p1}(t_1 - T_{11}) = 0.5U_1A_1[(T_{21} - t_1) + (t_3 - T_{11})] \\ &\text{(b)} \quad W_3 C_{p3}(t_2 - T_{31}) = 0.5U_2A_2[(t_3 - t_2) + (t_4 - T_{31})] \end{aligned} \tag{6.35}$$

In summary, the design problem calls for the minimization of (6.30) subject to the inequality constraints (6.31), the bounds on t_1, t_2, t_5, A_1 and A_2, and the equality constraints (6.32) and (6.35). Table 6.4 gives the required data used to solve the numerical example presented in this section. To simplify and put the optimization problem into the proper format, the following are done: 1) The constant term $-c_3T_{22}$ is dropped from the objective function. 2) Constraint (6.31c) is ignored since $T_{22} - \tau_c = 260°F > b_w = 180°F \geq t_5$. 3) Constraints (6.32a) and (6.32c) are used to eliminate t_3 and t_4. The resulting signomial program is

$$\min \quad 350A_1^{0.6} + 275A_2^{0.8} + 142.5t_5$$

$$\text{s.t.} \quad \text{(a)} \quad (9.634669 \times 10^{-4})t_1 + (1.67124 \times 10^{-3})t_2 \leq 1.0$$

$$\text{(b)} \quad 0.72251t_2 + 0.95t_5 + 0.5765t_1 - 633.21639 = 1.0$$

$$\text{(c)} \quad t_1 + 0.010255A_1t_1 - 5.58343A_1 - 239.0 = 1.0$$

Stream 1		
Input temperature	T_{11}	240 °F
Flow rate	W_1	23,060 lb/hr.
Heat capacity	C_{p1}	0.5 BTU/lb °F
Upper bound of output temp.	b_1	550 °F
Stream 2		
Input temperature	T_{21}	480 °F
Output temperature	T_{22}	280 °F
Flow rate	W_2	25,000 lb/hr.
Heat capacity	C_{p2}	0.8 BTU/lb °F
Stream 3		
Input temperature	T_{31}	278 °F
Flow rate	W_3	20,643 lb/hr.
Heat capacity	C_{p3}	0.7 BTU/lb °F
Range of output temp.	$a_2 \leq t_2 \leq b_2$	300 °F $\leq t_2 \leq$ 350 °F
Cooling Water		
Input temperature	T_w	100 °F
Flow rate	W_w	19,000 lb/hr.
Heat capacity	C_{pw}	1 BTU/lb °F
Upper bound of output temp.	b_w	180 °F
Annual cooling cost	c_3	150$/°F yr.
Heat Exchanger X1		
Installation cost	$c_1 A^{\gamma_1}$	$350A^{0.6}$$/yr.
Overall heat transfer coefficient	U_1	150 BTU/hr. ft.^{2} °F
Heat Exchanger X2		
Installation cost	$c_2 A^{\gamma_2}$	$275A^{0.8}$$/yr.
Overall heat transfer coefficient	U_2	150 BTU/hr. ft.^{2} °F
Minimum approach for exchangers	τ_x	20 °F
Minimum approach for cooler	τ_c	20 °F

Table 6.4 Data for the heat exchanger network design problem.

$$\text{(d)}\quad t_2 + 0.0051903A_2t_2 + 0.0029922A_2t_1 - 2.72676A_2 - 0.0049308A_2t_5 - 277.0 = 1.0$$

$$\text{(e)}\quad 0 \leq t_1 \leq 500.0$$

$$\text{(f)}\quad 300.0 \leq t_2 \leq 350.0$$

$$\text{(g)}\quad 119.0 \leq t_5 \leq 180.0$$

$$\text{(h)}\quad 0 \leq A_i \leq 100.0, \quad i = 1, 2 \qquad (6.36)$$

As in the previous two design problems, the optimal solution is obtained by the code SP using Scheme II to update $\underline{\lambda}$ and K. The parameters of Scheme II are as before. Table 6.5 gives both the starting point and the optimal solution (6.36). The optimal values of t_3 and t_4 are obtained from t_1 and t_5 according to the following linear relations derived from (6.32a), (6.32c) and the data of Table 6.4.

$$t_3 = -0.5765t_1 + 618.0$$
$$t_4 = 0.95t_5 + 185.0 \qquad (6.37)$$

Thus at the optimum $t_3 = 391.13°F$ and $t_5 = 355.9°F$. It can be easily verified that the error due to replacing the logarithmic mean with the arithmetic mean is less than 5% in exchanger X1 and less than 1% in exchanger X2. Thus the approximation by the arithmetic mean is reasonable. The energy balance equations (6.32) and (6.35) are all satisfied within a relative discrepancy of only 0.02%.

From the optimal solution, it can be seen that t_5, the cooling water's output temperature, is almost at the maximum allowable value. This confirms that even at the optimum, the cooler by itself cannot absorb the heat load. The solution also reveals the interesting result that it is more economical to have A_1 at it upper bound than to have a higher output temperature t_1 in exchanger X1.

	STARTING POINT	OPTIMAL SOLUTION
Cost	313226.83	24787.63
t_1	300.0	394.15
t_2	325.0	326.76
t_5	150.0	179.9
A_1	50.0	100.0
A_2	50.0	66.04

Table 6.5 Starting point and optimal solution of the heat exchanger network problem.

VII. CONCLUSION

7.1 Synopsis

A wide range of engineering design problems can be modeled as signomial programs with both equality and inequality constraints. The theory of signomial programming, however, has been developed in terms of inequality constraints only. In this thesis, an algorithm is proposed and implemented to solve signomial programs with mixed inequality and equality constraints. The algorithm does not require its user to transform the equality constraints to inequalities. The equality constraints are automatically incorporated into the original objective function to reduce the original problem into a sequence of less constrained signomial programs with inequality constraints only. Because it is within the framework of signomial programming, the proposed algorithm is computationally practical. Because no problem manipulation is needed prior to data entry, the physical significance of the original problem formulation is preserved.

In Chapter II, a concise review of signomial programming is given. Signomial programs are defined and shown to be nonconvex. Because of nonconvexity, the solution of a signomial program can at best be guaranteed as a local minimum. Another consequence of nonconvexity is the existence of only a weak duality theorem relating the Kuhn-Tucker points of a primal signomial program to the Kuhn-Tucker points of a linearly constrained dual problem. At these Kuhn-Tucker points, the primal and dual objective functions are equal. An important special class of signomial programs is that of geometric programs. It is shown that geometric programs can be changed via the logarithmic transformation into convex

programs. As a result, geometric programming possesses an elegant strong duality theorem which identifies the global minimum of a geometric program as the global maximum of a concave dual function restricted to a linear manifold. The duality theorem thus offers two options, the primal and the dual approach, for computing the solution of a geometric program. Neither option can be categorically stated as being superior in all cases. However, given an adopted method to solve geometric programs, the numerical solution of a signomial program can be obtained by successively approximating the nonconvex signomial program with convex geometric programs, each of which can be conveniently solved. This is the strategy of the promising Avriel-Williams algorithm used in a slightly different form in the later chapters of the thesis. In the Avriel-Williams algorithm, the approximation is accomplished by condensing the signomials' negative terms into monomials.

The main problem of signomial programs with equality and inequality constraints is formulated and solved in Chapter III. The proposed algorithm, Algorithm 3.2, is the synthesis of three basic guiding concepts: the multiplier method viewed as a primal-dual method, partial dualization, and preservation of compatibility with signomial programming. In lieu of confronting the original problem, the algorithm solves a sequence of inequality-constrained signomial programs each specified by the Lagrange multiplier estimate vector $\underline{\lambda}$ and the penalty constant K. Alternating with the solution of each signomial program of the sequence is the updating of $\underline{\lambda}$ and K according to some rules. The convergence of Algorithm 3.2 is assured by that of the method of multipliers and by the convergence of the Avriel-Williams algorithm used to solve the sequence of signomial programs. It is observed that expressing each problem of the sequence in the stan-

dard format of a signomial program would likely entail inconvenience in data preparation and computational difficulty due to large problem size. To circumvent these problems, three indirect methods of applying the Avriel-Williams algorithm to the subproblem (3.9) have been numerically explored. All three are based on monomial condensation, require no or small increase in variables and constraints, and need only the minimum effort for data entry. Of the three methods tested, one is selected as promising and discussed in detail in Chapter IV. The consideration of the other major step of Algorithm 3.2, the updating of $\underline{\lambda}$ and K, is deferred to Chapter V.

Chapter IV presents a new numerical method to implement a proposed variant of the Avriel-Williams algorithm for solving signomial programs. In this method the original nonconvex feasible set defined by signomial inequalities is approximated by a convex set obtained by monomial condensation. The nonconvex objective function is then minimized over the convex approximant by a combined reduced gradient and cutting plane algorithm. The minimization's solution in turn determines the next convex approximant. In contrast to this method is the original version of the Avriel-Williams algorithm in which the signomial objective function is first incorporated into the feasible set prior to the convex approximation. The proposed method seems to match better the need of signomial programs characterized by a high degree of difficulty with most of the terms being in the objective function. Unlike other methods reported in the literature, the method here readily admits extension to problems with a nonsignomial objective function but signomial inequality constraints. A practical example of such a problem is the constrained location problem solved in Section 4.7.3.

The software implementation of the combined reduced gradient and cutting plane method requires detailed considerations of several aspects

of the algorithm. Section 4.4 considers the questions relating to how to obtain an initial basic feasible solution, how to generate an initial basis of the linear constraints, how to augment the basis' inverse and get a basic feasible solution after adding a cut, how to perform the linear search, and how to update the basis. In the same section, the optimality criteria and the feasibility tolerances are defined. Computational experience with the software implemented shows that the proposed numerical method compares favorably with another well tested recent implementation. However, the method suggested in this thesis has the added advantageous flexibility of handling general nonconvex objective functions such as the algebraic functionals of signomials demonstrated in Section 4.7.

The formulation and testing of the updating rules for $\underline{\lambda}$ and K are reported in Chapter V. The two rules for updating $\underline{\lambda}$ are the Hestenes-Powell rule and one based on fitting a cubic to the ordinary Lagrangian. The parameter K is increased by a factor $\phi > 1$ either only when the norm of the equality constraints does not decrease sufficiently, or in every dual iteration independent of the behavior of the equality constraints. Altogether four schemes are tested in a series of numerical experiments. In terms of rate of convergence, the most promising scheme is that using the Hestenes-Powell rule for $\underline{\lambda}$ and increasing K according to the second criterion. The impact of the range of values of K on the rate of convergence is also studied. It appears that best results are obtained when K is initially small and then is moderately increased, provided the equality constraints are initially scaled proportionately. Furthermore approximate minimization in the first few iterations can reduce considerably computing time without any serious loss of convergence and the required accuracy.

In Chapter VI, the algorithms discussed in this thesis are applied to three engineering design problems. The first design problem involves choosing the cross-sectional areas of a hyperstatic pin-jointed structure such that the total volume is minimized subject to stress, deflection and topology constraints. The second example required the specification of the most profitable operating condition of an alkylation process commonly found in the petroleum industry. The last problem aims at achieving the optimal trade-off between installation and operating costs of a small heat exchanger network that is typical of those used in process engineering. Each design problem is first formulated from its engineering point of view before the numerical solution is obtained and interpreted.

6.2 Contributions of the Thesis

The research reported in this thesis is the first known effort to develop an algorithm for solving signomial programs with mixed inequality and equality constraints without having to transform one type of constraints to the other. Furthermore, the algorithm is compatible with the existing methods of signomial programming. The specific original contributions of this thesis may be listed as follows:

1. the proposal of an algorithm that can automatically handle both inequality and equality signomial constraints within the framework of signomial programming and without any need for user manipulation of the constraints;
2. the development of a new numerical method to realize a variant of the Avriel-Williams algorithm for signomial programs whose objective function may have many terms or is an implicit signomial;
3. the extension of the numerical method of Item 2 to include non-signomial objective functions such as algebraic functionals of

signomials, and the application of the method to constrained location-allocation problems;

4. the software implementation of the aforementioned algorithms, and the acquisition of computational experience with the implementation;
5. the numerical study of the updating schemes for the parameters $\underline{\lambda}$ and K yielding useful guidelines for later applications of the algorithms;
6. the formulation and numerical solution of selected realistic design problems cast as signomial programs with mixed types of constraints.

6.3 Suggestions for Further Research

Further research based on the results of this thesis may be pursued along several directions. The first is to investigate alternative more efficient ways of solving the primal subproblem (3.9). The solution of the subproblem consumes a good part of the total computing time. Hence the ability to solve the subproblem faster is definitely desirable. Faster convergence may be achieved in various ways. Within the present framework of the combined reduced gradient and cutting plane (CRGCP) algorithm, effort may be focused on improving the inexact linear search, the procedure of obtaining simultaneously a basis and its inverse in the presence of tight constraints, and the method of obtaining a feasible and better point after the addition of a cut. A more involved study is to consider incorporating a quasi-Newton feature into the CRGCP algorithm so that superlinear convergence may replace the present algorithm's linear convergence.

Another item worthy of further research is to consider using the CRGCP algorithm for solving the class of problems characterized by a nonconvex objective function and convex constraints. This study would include devising procedures for obtaining an initial feasible (possibly interior) point and comparing different cutting plane schemes so as to achieve feasibility earlier.

More investigation is also needed on the specification and the updating of $\underline{\lambda}$ and K. For example, the initial values of $\underline{\lambda}$ and K may be more judiciously selected on the basis of primal feasibility or optimality. K may be allowed to be different with each of the q equality constraints, and then the q penalty constants may be automatically adjusted in an adaptive fashion to maintain proper scaling in the dual space.

The algorithms proposed in this thesis may also be used to solve wider classes of problems to extend the applicability of signomial programming. One such class is that of "pseudo" signomial programs a few terms of which have transcendental functions. Clearly such problems are not signomial programs. However, through signomial approximation of the transcendental functions and the use of new variables and equality constraints, the programs can be closely approximated by true signomial programs with equality constraints. Hence they can be solved by the algorithms of this thesis. The approximation schemes need to be explored, and it would be very convenient if the approximation and the required new variables and equality constraints could be internally generated by the software package.

REFERENCES

1. R.J. Duffin, E.L. Peterson, C. Zener, Geometric Programming, New York: Wiley, 1967.

2. W.I. Zangwill, Nonlinear Programming: A Unified Approach, Englewood Cliffs, N.J.: Prentice-Hall, 1969.

3. R.J. Duffin, "Linearizing geometric programs", SIAM Review, 12, 211-227, April, 1970.

4. U. Passy and D.J. Wilde, "Generalized polynomial optimization", SIAM J. Appl. Math., 15, 1344-1356, September, 1967.

5. M. Avriel and A.C. Williams, "Complementary geometric programming", SIAM J. Appl. Math., 19, 125-141, July, 1970.

6. R.J. Duffin and E.L. Peterson, "Geometric programming with signomials", J. Optim. Th. Applic., 11, 3-35, 1973.

7. G.W. Westley, "A geometric programming algorithm", Report No. ORNL-4650, Oak Ridge National Laboratory, Oak Ridge, Tennessee, July, 1971

8. T. Jefferson, "Geometric programming with an application to transportation planning", Ph.D. Dissertation, Dept. of Operations Research, Northwestern Univ., Evanston, Illinois, August, 1972.

9. P.A. Beck and J.G. Ecker, "Some computational experience with a modified convex simplex algorithm for geometric programming", Armament Development and Test Center, Elgin A.F.B., Florida, ADTC-72-20.

10. G.A. Kochenberger, R.E.D. Woolsey, and B.A. McCarl, "On the solution of geometric programs via separable programming", Operational Research Quarterly, 24, 285-294, June, 1973.

11. G.A. Kochenberger, "Geometric programming - extension to deal with degrees of difficulty and loose constraints", Ph.D. Thesis, School of Bus. Admin., Univ. of Colorado, 1969.

12. M. Avriel, R. Dembo and U. Passy, "Solution of generalized geometric programs", Int'l J. for Numerical Methods in Eng., 9, 149-168, January, 1975.

13. A. Charnes and W.W. Cooper, "A convex approximant method for nonconvex extensions of geometric programming", Proc. Nat'l Academy of Sciences, 56, 1361-1364, 1966.

14. R.J. Duffin and E.L. Peterson, "Reversed geometric programs treated by harmonic means", Indiana Univ. Math. J., 22, 531-550, December, 1972.

15. U. Passy, "Condensing generalized polynomials", J. Optim. Th. Applic., 9, 221-237, 1972.

16. R.S. Dembo, "Solution of complementary geometric programming problems", M.Sc. Thesis, Dept. of Chem. Eng., Technion, Haifa, Israel, 1972.

17. G.E. Blau and D.J. Wilde, "Generalized polynomial programming", Canad. J. Chem. Eng., 47, 317-326, June 1969.

18. M.J. Rijckaert, "Engineering applications of geometric programming", Optimization and Design, ed. by M. Avriel et al., Englewood Cliffs, N.J.: Prentice-Hall, 196-220, 1973.

19. K.I. Majid, Optimum Design of Structures, London-Newner-Butterworths, 1974.

20. U. Passy, "Nonlinear assignment problems treated by geometric programming", Operations Research, 19, 1675-1689, 1971.

21. G. Blau and D.J. Wilde, "A Lagrangian algorithm for equality constrained generalized polynomial optimization", A.I.Ch.E.Journal, 17, 235-240, January, 1971.

22. J. Yan and M. Avriel, "Signomial programs with equality constraints", Proc. 1975 Int'l Conf. Cybernetics and Society, 450-452, September, 1975.

23. A.V. Fiacco and G.P. McCormick, Nonlinear Programming: Sequential Unconstrained Minimization Techniques, New York, Wiley, 1968.

24. D.G. Luenberger, Introduction to Linear and Nonlinear Programming, Reading, Mass.: Addison-Wesley, 1973.

25. M.R. Hestenes, "Multiplier and gradient methods", J. Optim. Th. Applic., 4, 303-320, 1969.

26. M.J.D. Powell, "A method for nonlinear constraints in minimization problems", in Optimization, ed. by R. Fletcher, New York: Academic Press, 283-297, 1969.

27. D.P. Bertsekas, "Combined primal-dual and penalty methods for constrained minimization", SIAM J. Control, 13, 521-544, 1975.

28. R.T. Rockafellar, "A dual approach to solving nonlinear programming problems by unconstrained optimization", Math. Programming, 5, 354-373, 1973.

29. R.T. Rockafellar, "Augmented Lagrange multiplier functions and duality in nonconvex programming", SIAM J. Control, 12, 268-285, 1974.

30. O.L. Mangasarian, "Unconstrained Lagrangians in nonlinear programming", SIAM J. Control, 13, 772-791, July 1975.

31. H. Nakayama, H. Sayama, and Y. Sawaragi, "A generalized Lagrangian function and multiplier method", J. Optim. Th. Applic., 17, 211-227, 1975.

32. H. Nakayama, H. Sayama, and Y. Sawaragi, "Multiplier method and optimal control problems with terminal state constraints", Int. J. Sys. Sci., 6, 465-477, 1975.

33. A.M. Geoffrion, "Duality in nonlinear programming: a simplified applications-oriented development", SIAM Review, 13, No. 1, 1-37, 1971.

34. V.T. Polyak and N.V. Tret'yakov, "The method of penalty estimates for conditional extremum problems", USSR Computational Math and Math Phys. 13, 42-58, 1973.

35. D.P. Bertsekas, "On penalty and multiplier methods for constrained minimization", SIAM J. Control and Optim., 14, 216-235, 1976.

36. M. Avriel and V. Gurovich, "A condensation algorithm for a class of algebraic programs", submitted to Operations Research.

37. P. Wolfe, "Methods of nonlinear programming", in Recent Advances in Mathematical Programming, ed. by R.L. Granes and P. Wolfe, New York: McGraw-Hill, 1963.

38. J. Abadie and J. Carpentier, "Generalization of the Wolfe reduced gradient method to the case of nonlinear constraints", in Optimization, ed. by R. Fletcher, New York: Academic Press, 1969.

39. A.R. Colville,"A comparative study on nonlinear programming codes", Report No. 320-2949, IBM Scientific Center, 1968.

40. J.E. Kelley, "The cutting plane method for solving convex programs", SIAM J. Appl. Math., 8 703-712, 1960.

41. A.M. Geoffrion, "Elements of large-scale mathematical programming. Part I: concepts", Man. Sci., 14, 652-675, 1970.

42. M. Simonnard, Linear Programming Transl. by W.S. Jewell, Englewood Cliffs, N.J. : Prentice-Hall, 1966.

43. A.A. Goldstein, Constructive Real Analysis, New York: Harper, 1967.

44. L. Armijo, "Minimization of functions having continuous partial derivatives", Pacific J. Math., 16, 1-3, 1966.

45. Y. Bard, "Comparison of gradient methods for the solution of nonlinear parameter estimation problems", SIAM J. Numer. Anal., 7, 157-186, 1970.

46. W.C. Davidon, "Variable metric method for minimization", AEC Research and Development Report ANL-5990, Nov. 1959.

47. M. Avriel, Nonlinear Programming: Analysis and Methods, Prentice-Hall, in press.

48. M. Avriel and A.C. Williams, "An extension of geometric programming with applications in engineering optimization", J. of Eng. Math, 5, 187-194, July 1971.

49. R.L. Francis and J.M. Goldstein, "Location theory: a selective bibliography", Op. Res., 22, 400-410, 1974.

50. A.P. Hurter, Jr., M.K. Schaefer, and R.E. Wendell, "Solutions of constrained location problems", Man. Sci., 22, 51-56, 1975.

51. V. Gurovich, "Extensions of signomial programming: numerical solution", M.Sc. Thesis, Technion, Haifa, Israel, 1974.

52. M.K. Schaefer and A.P. Hurter, Jr., "An algorithm for the solution of a location problem with metric constraints", Naval Research Logistic Quart., 21, 625-636, 1974.

53. R.F. Love and J.G. Morris, "Solving constrained multi-facility location problems involving &p distances using convex programming", Op. Res., 23, 581-587, 1975.

54. H.W. Kuhn and R.E. Kuenne, "An efficient algorithm for the numerical solution of the generalized Weber problem in spatial economics, J. of Regional Science, 4, 21-33, 1962.

55. R.N. Sauer, A.R. Colville, Jr., and C.W. Burwick, "Computer points the way to more profits", Hydrocarbon Process. Petrol. Refiner., 43, 84-92, 1964.

56. R.E. Payne, "Alkylation - what you should know about this process", Petrol. Refiner., 37, 316-329, 1958.

57. J. Bracken and G.P. McCormick, Selected Applications in Nonlinear Programming, New York: Wiley, 1968.

58. J.R. Backhurst and J.H. Harker, Process Plant Design, London, U.K.: Heinemann Education Books, 1973.

59. R.A. Greenkorn and D.P. Kessler, Transfer Operations, New York: McGraw-Hill, 1974.