1-1-1984

A geometric programming algorithm for augmented-zero degree op difficulty problems.

Patricia Douglas-Jarvis

A GEOMETRIC PROGRAMMING ALGORITHM FOR AUGMENTED-ZERO DEGREE OF DIFFICULTY PROBLEMS

by

Patricia Douglas-Jarvis

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Industrial Engineering

Lehigh University

1984

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

Dec. 17, 1984
date

Professor in Charge

Chairperson of Department

CONTENTS

1. Abstract

Posynomial geometric programming is a methodology that optimizes nonlinear programs comprised of positive polynomial (posynomial) terms. This geometric program can be augmented in a way such that it will possess a degree of difficulty equal to zero. This thesis proposes a solution procedure that utilizes this augmentation. First, the dual program for the augmented program is solved for the primal variables and the slack variables. Next, an augmented Lagrangian function is introduced containing additional variables and a form of the Lagrange multipliers. This Lagrangian function is first solved for the Lagrange multipliers using an augmented Lagrangian approach. Lastly, the added variables are found using a projected gradient method with an active set strategy. The procedure iterates through the three phases until convergence of the primal objective occurs.

2. Introduction

Geometric programming is a technique for solving certain nonconvex nonlinear programs with linear or nonlinear constraints. A geometric program has the form:

$$\begin{aligned} &\text{minimize} \quad F(\underline{t}) \\ &\quad \underline{t} \in R \\ &\text{subject to} \quad g_i(\underline{t}) \leq 1 \quad \text{for} \quad i = 1,2,\ldots,m \\ &\qquad\qquad \underline{t} > 0 \end{aligned}$$

where $\underline{t}$ is a vector. F and g_i are polynomials in which each term is a product of the variables and each variable in a term is raised to a real numbered exponent.

The technique emerged in 1961 when Clarence Zener was solving problems in engineering design where a sum of component costs are to be minimized[1]. These types of problems were difficult to solve using conventional nonlinear programming methods. For example, the use of the Newton-Raphson method would be very difficult and may not converge. Even if it did converge, the proper answer may not be found.

Richard Duffin, at the same time, was developing a duality theory approach for solving nonlinear programming problems. When Duffin heard of Zener's work, he supported Zener's intuitive theory with a mathematical formulation[2]. This new method of nonlinear optimization used arithmetic and geometric means and was called geometric programming. Duffin and Zener worked with Elmor Peterson, one of Duffin's students, to improve the method to allow for inequality constraints. This method was restricted to functions with positive coefficients, due to the fact that the method was based on Cauchy's inequality. In 1967, Duffin, Peterson, and Zener produced a text book on geometric programming[3].

At about the same time, Passy and Wilde[4] developed a method without the restriction of positive coefficients. Instead of using Cauchy's inequality, Lagrange multiplier methods[5] and the Kuhn-Tucker conditions[4] were a basis for this method.

Much of the application of geometric programming is in the area of engineering design. With geometric programming, more realistic design methods can be

used. Some applications are in design of welded beam structures, gas transmission compressor design, marketing, chemical engineering processes, and the optimization of nuclear systems. These and many other applications are found in [6]. According to Beightler and Phillips[6], the use of geometric programming in engineering design has been slow to evolve. Commonly used optimization techniques are not easily applied to nonlinearly constrained problems. The strength of geometric programming is that it transforms the problem with nonlinear constraints into one with linear constraints. This transformation supplies a more realistic model than the questionable linearizations used in the past. When certain properties are present in this transformed problem, the problem can be solved through a set of well-defined linear equations[7]. Methods for solving problems when these conditions are not present is the subject of much recent research.

Today, most geometric programming techniques utilize the structure of the primal program, the dual program, or a transformation of the primal program.

These techniques are based on conventional nonlinear programming techniques.

3. Literature Review[8]

The geometric programming problem is:

$$\begin{aligned} \text{minimize} \quad & g_0(\underline{t}) \\ \text{subject to} \quad & g_k(\underline{t}) \leq \text{ for } k=1,2,\ldots,p \\ & t_i > 0 \text{ for } i=1,2,\ldots,m \end{aligned}$$

where:

$$g_k(\underline{t}) = \sum_{i \in J\{k\}} c_i \prod_{j=1}^{m} t_j^{a_{ij}} \quad \text{for } k=0,1,2,\ldots,p \quad (1)$$

$$c_i > 0 \quad \text{for} \quad i=1,2,\ldots,n$$

$J\{k\}$ is an index set $\{m_k, m_{k+1}, \ldots, n_k\}$ for $k=0,1,\ldots,p$, $m_0=1$, $m_1=n_0+1,\ldots,m_p=n_p+1$, $n_p=n$

The exponents a_{ij} are arbitrary real numbers and the c_{ij} are strictly positive real numbers. When the coefficients are positive, the functions are called posynomials. This paper will deal strictly with posynomial functions.

In a geometric programming problem with n terms and m variables, the degree of difficulty is defined as n-m-1. Geometric programming problems possessing a degree of difficulty of zero are easier to solve than those having degree of difficulty greater than zero. It is advantageous therefore, to convert a geometric program into a program that will have degree of difficulty equal to zero if possible. When the degree of difficulty is equal to zero, the solution to the associated dual program is uniquely found. The dual constraint set is a system of n linear equations with n unknowns.

A comparison of geometric programming algorithms was done by Sarma et al. in [8]. Most geometric programming techniques have been based on conventional nonlinear techniques used to exploit the structures of the primal program, the dual program, or the transformed primal program. Sarma et al. attempt to compare these solution alternatives. Five different nonlinear programming algorithms were chosen for the comparison.

3.1 Primal Approaches

The posynomial functions in a geometric program are continuously differentiable. Analytic formulas for the first and second derivatives are easily formed. Therefore, differential optimization techniques using second order derivatives can be used. The SUMT[10] code, using a second order optimization technique was included in the study by Sarma et al.

Furthermore, by changing variable t as follows,

$$t_i = \exp(z_i), \qquad \text{for } i=1,2,\ldots,m$$

the posynomial functions become convex. This convexity remains after logarithmic transformations so that

$$\log g_k(\mathbf{z})$$

are also convex[11]. Therefore, convex programming algorithms can be applied to the convexified primal problem. One approach, the GGP code[12], uses this technique, then linearizes the convexified problem. This program uses a form of Kelley's Cutting Plane

Algorithm[13] whereby the linear problems are solved using the dual simplex method.

3.2 Dual Approaches

The logarithm of the dual function is a concave function and is continuously differentiable. The logarithm of the dual function and its associated linear constraints can be maximized using several algorithms[14]. General techniques for linearly constrained problems can be used with slack variables to which a penalty factor has been applied[14][15][16]. A concave simplex algorithm to which penalized slack variables are added is called CS and is described in [17]. A modification of the concave simplex algorithm called MCS can also be used and is described in [18].

3.3 Transformed Primal Approaches

To transform the primal problem, the following transformation of variables is used:

$$t_i = \exp(z_i), \qquad \text{for } i=1,2,\ldots,m$$

along with the following definition

$$\underline{w} = A^{T} \underline{z} + \log \underline{c}$$

forms the transformed primal auxiliary problem[3][19][20],

$$\text{minimize } f_o(\underline{w}) = \sum_{i=1}^{T_o} \exp(w_i) \qquad (T \text{ is terms})$$

$$\text{subject to } f_k(\underline{w}) = \sum_{i=S_k}^{Tk} \exp(w_i) \leq 1 \qquad \text{for } k=1,2,\ldots,p$$

$$L(\underline{w} - \log \underline{c}) = 0$$

where the rows of matrix L are a set of linearly independent vectors spanning the null space of the exponent matrix A[8]. If the matrix A has full rank, this transformed problem is equivalent to the primal program.

This method utilizes the underlying convexity of the primal program. The program called DAP in the Sarma et al. study uses a reduced gradient algorithm with an active set strategy[21] to solve the

transformed program.

3.4 Comparison of Approaches

The problems used in the Sarma et al. study ranged in size from three variables and one constraint to twenty-four variables and forty-two constraints. All of the objective functions were nonlinear and non-quadratic. The exact dimensions of the problems can be found in [8], along with initial solutions and stopping criteria.

First, some disadvantages of using a dual approach should be pointed out. The dimensionality of a dual program will often be larger then that of a primal program, therefore requiring more work. Also, the rank of the log-linear equations may be less than m. This requires subsidiary maximizations to return to the primal solution[3][18]. The relationship between primal and dual variables is very sensitive to the values of the dual variables[22][23][18]. This means the dual variables have to be calculated very accurately, increasing computation time.

Some advantages of the transformed primal approach should also be pointed out. The transformed

primal problem is a separable convex programming problem. Each variable w_i occurs in only one of the nonlinear functions and the nonlinear functions are continuously differentiable. The number of linear equality constraints will be equal to n-m. The slackness in any inequality constraint does not cause difficulties numerically or analytically. The last advantage is that the single term constraints reduce to

$$w_i \leq 0.$$

However, the dimensionality of the transformed primal program is greater than in the primal program, but not to the degree of that of the dual program.

In comparing the dual approaches, the MCS code was more effective than the CS code with slack variables. This is especially true with large problems with more than four constraints. Sarma et al. claim that this is due to the increase in dimensionality and numerical difficulties brought on by the addition of slack variables and the small dual variables that result when a constraint is active.

Looking at the primal approaches, the GGP code was more effective than the SUMT code. In fact, the GGP code seemed more effective than all of the other code studied. Sarma et al. conclude that it is important to exploit the underlying convexity of the primal problem as well as the dual. Also, GGP does not require computation of feasible starting points. Constraint activity or looseness are handled implicitly in GGP without any complicated active set strategy. Another advantage of the GGP algorithm is that the convexity of the primal can be utilized without an increase in dimensionality.

In comparing the dual approach to the transformed primal approach, superiority depends on the constraints. When the multi-term constraints are tight, the dual approach is better. If more than one constraint is loose, the transformed primal approach is better. One advantage of the transformed primal approach over the dual approach is that the variables associated with the single term constraints do not appear in the objective function. In the dual approach, the single term constraints are in the objective function.

In general, an algorithm which does not utilize the underlying convexity of the primal program is not effective. There seems to be no advantage to using a dual approach over a transformed primal approach. The lower dimensionality of the convexified primal approach is likely to be more effective that the transformed primal approach.

4. McNamara's Solution to the Geometric Programming Problem

John R. McNamara[9] developed a solution procedure where an augmented geometric program possessing degree of difficulty equal to zero is found. This augmented program depends on the original program satisfying certain conditions. The iterative solution procedure as presented by McNamara does not guarantee convergence. This paper proposes a solution to McNamara's augmented problem that is more likely to converge. Also, a computer program to solve the geometric program is given.

McNamara used the following problem as an example of a geometric program.

$$\text{minimize} \quad g_0(\underline{t}) = 10t_1^{1.6} + 6t_1t_2 + 4t_2^{2.2}$$

$$\text{subject to} \quad g_1(\underline{t}) = 0.2t_1^{-2}t_2^{-1.5} + 0.4t_2^{1.1} \leq 1$$

$$g_2(\underline{t}) = 0.3t_1t_2^{-0.8} \leq 1$$

$$t_1 > 0$$

$$t_2 > 0$$

This primal program has an associated dual program:

$$\text{maximize} \quad v(\underline{\delta}) = \prod_{i=1}^{n} \left(\frac{c_i}{\delta_i}\right)^{\delta_i} \prod_{k=1}^{p} \lambda_k(\underline{\delta})^{\lambda_k(\underline{\delta})} \qquad (2)$$

$$\text{subject to} \quad \sum_{i=1}^{n} a_{ij}\delta_i = 0 \quad \text{for} \quad j = 1,2,\ldots,m$$

$$\sum_{i \in J\{0\}} \delta_i = 1$$

$$\delta_i > 0 \quad \text{for} \quad i = 1,2,\ldots,n$$

where

$$\lambda_k(\underline{\delta}) = \sum_{i \in J\{k\}} \delta_i \quad \text{for} \quad k = 1,2,\ldots,p$$

($\underline{\delta}$ is the vector of dual variables)

Theory relating the primal program to the dual program is found in [3]. When δ maximizes the dual program, the minimizing point for the primal problem satisfies the following equations:

$$c_i\, t_1^{a_{i1}}\, t_2^{a_{i2}} \cdots t_m^{a_{im}} = \begin{cases} \delta_i' v(\delta') , & i \in J\{0\} \\ \delta_i'/\lambda_k(\delta') , & i \in J\{k\} \end{cases}$$

By taking logarithms, the vector t can be found. The dual program for the previous example is:

$$\text{maximize } v(\delta) = (10/\delta_1)^{\delta_1} (6/\delta_2)^{\delta_2} (4/\delta_3)^{\delta_3} (0.2/\delta_4)^{\delta_4} (0.4/\delta_5)^{\delta_5} (0.3/\delta_6)^{\delta_6} (\delta_4 + \delta_5)^{\delta_4+\delta_5} (\delta_6)^{\delta_6}$$

$$\begin{aligned} \text{subject to } & 1.6\delta_1 + \delta_2 - 2\delta_4 + \delta_6 = 0 \\ & \delta_2 + 2.2\delta_3 - 1.5\delta_4 + 1.8\delta_5 - 0.8\delta_6 = 0 \\ & \delta_1 + \delta_2 + \delta_3 = 1 \\ & \delta_i \geq 0 \quad \text{for} \quad i = 1,2,\ldots,6 \end{aligned}$$

The dual optimal solution δ^* is related to the primal optimal solution t^* by the following:

$$v(\delta^*) = g_0(t^*)$$

The primal and dual solutions are related by the log-linear equations which are defined for j where $\delta_j^* > 0$.

$$\sum_{i=1}^{m} a_{iT} \log t_i^* = \log [\delta_T^*/c_T v(\delta^*)] \quad , \quad 1 \leq T \leq T_0$$

$$= \log (\delta_T^*/c_T \lambda_p^*) \quad , \quad S_p \leq T \leq T_p$$

The dual program is of larger dimension than the primal program.

5. McNamara's Augmented Problem

Suppose the primal problem with degree of difficulty greater than zero satisfies the following conditions:

1. Objective function has m terms and the rank of the matrix of exponents equals m.

2. There is at least one tight constraint at optimality.

Sometimes, (as in the example) the first condition is

not met. Often it is possible to transform the problem so that it meets condition 1. This is discussed in [3]. Condition 1 insures that the augmented problem to be proposed will be of degree of difficulty equal to zero. The example problem can be so transformed:

$$\begin{aligned}
\text{minimize} \quad & g_0(\underline{t}) = 10t_1^{1.6} + 4t_2^{2.2} + t_3 \\
\text{subject to} \quad & g_1(\underline{t}) = 0.2t_1^{-2}t_2^{-1.5} + 0.4t_2^{1.1} \leq 1 \\
& g_2(\underline{t}) = 0.3t_1t_2^{-0.8} \leq 1 \\
& g_3(\underline{t}) = 6t_1t_2t_3^{-1} \leq 1 \\
& t_1 > 0 \\
& t_2 > 0 \\
& t_3 > 0
\end{aligned}$$

This was done by adding a variable $t_3 = 6t_1t_2$ and a constraint associated with the new variable.

An augmented problem is constructed by:

1. Multiplying each term by a slack variable t_i, $i=m+1,\ldots,n$,

2. Adding a constraint that is the product of these added variables ($g_{p+1}(\underline{t}_s)$), where $\underline{t}_s$ is a vector of the added slack variables.

By augmenting, the primal problem becomes:

$$\text{minimize} \quad g_0(\underline{t}) = \sum_{i \in J\{0\}} c_i \prod_{j=1}^{m} t_j^{a_{ij}}$$

$$\text{subject to} \quad g_k(\underline{t},\underline{t}_s) = \sum_{i \in J\{k\}} c_i \prod_{j=1}^{m} t_j^{a_{ij}} \, t_i \leq 1$$

$$\text{for } k=1,2,\ldots,p \qquad (3)$$

$$t_i > 0 \quad \text{for } i = 1,2,\ldots,m$$

$$g_{p+1}(\underline{t}_s) = \prod_{i=m+1}^{n} t_i^{b_i} \leq 1 \qquad (4)$$

$$\text{where} \quad t_i > 0 \quad \text{for} \quad i = m+1,\ldots,n$$

$$b_i \leq -1$$

There are now n+1 terms (m=n+1) and n variables, so the degree of difficulty is zero. The augmented form of the transformed example is:

$$\text{minimize} \quad g_0(\underline{t}) = 10t_1^{1.6} + 4t_2^{2.2} + t_3$$

$$g_1(\underline{t},\underline{t}_s) = 0.2t_1^{-2} t_2^{-1.5} t_4 + 0.4t_2^{1.1} t_5 \leq 1$$

$$g_2(\underline{t},\underline{t}_s) = 0.3t_1 t_2^{-0.8} t_6 \leq 1$$

$$g_3(\underline{t},\underline{t}_s) = 6t_1 t_2 t_3^{-1} t_7 \leq 1$$

$$g_4(\underline{t}_s) = t_4^{b_4} t_5^{b_5} t_6^{b_6} t_7^{b_7} \leq 1$$

$$t_i > 0 \quad \text{for} \quad i = 1,2,\ldots,7$$

McNamara has shown that if $\underline{t}^*$, $\underline{t}_s^*$ solve the augmented problem and $t_i \geq 1$, $i=m+1,\ldots,n$, then $\underline{t}^*$ solves the primal geometric programming problem.

5.1 A Proposed Solution Procedure

Recall the constraint set for the dual program associated with the primal problem:

$$\sum_{i=1}^{n} a_{ij}\delta_i = 0 \quad \text{for} \quad j = 1,2,\ldots,m$$

After performing matrix operations on the exponent matrix of the augmented problem so that an identity matrix is in the first m rows and columns, a closed form solution for $\underline{\delta}$ can be found. The solution for $\underline{\delta}$ is:

$$\delta_{n+1} = \frac{1}{\sum_{j=1}^{m} \sum_{i=m+1}^{n} \gamma_{ji} b_i} \tag{5}$$

$$\delta_j = \delta_{n+1} \sum_{i=m+1} \gamma_{ji} b_i \quad \text{for} \quad j = 1,2,\ldots,m \tag{6}$$

$$\delta_i = -\delta_{n+1} b_i \quad \text{for} \quad i = m+1,\ldots,n \tag{7}$$

where γ_{ji} is an element of the manipulated exponent matrix.

The vector b is chosen so that δ is strictly positive, which will supply a canonical solution to the primal problem. δ will be strictly positive if b satisfies:

$$\begin{aligned} &\sum_{i=m+1}^{n} \gamma_{ji} b_i > 0 \quad \text{for} \quad j = 1,2,\ldots,m \\ &\text{and} \quad b_i \leq -1 \quad \text{for} \quad i = m+1,\ldots,n \end{aligned} \tag{8}$$

McNamara presents a solution procedure to estimate b and then solves for t, t_s. His procedure does not guarantee convergence nor does it guarantee that the condition

$$\sum_{i=m+1}^{n} y_{ji} b_i > 0 \qquad (9)$$

for j=1,2,...,m is satisfied.

The proposed procedure forms a Lagrangian function associated with the augmented problem. This Lagrangian has a term added to insure that the slack variables are greater than 1.0. The Lagrangian is maximized as a function of $\underline{b}$ and the Lagrange multiplier vector μ. This is done in two parts, first for μ, then for $\underline{b}$. The constraints are linear and an augmented Lagrangian method and a gradient projection method is used to solve the Lagrangian problem. Using the following constraints from the augmented problem,

$$g_k(\underline{t},\underline{t}_s) \leq 1 \quad \text{for } k=1,2,\ldots,p$$
$$g_{p+1}(\underline{t}_s) \leq 1$$
$$t_i > 0 \quad \text{for } i=1,2,\ldots,p$$
$$t_i > 1 \quad \text{for } i=m+1,\ldots,n$$

The Lagrangian is formed as follows:

$$L = g_0(t) + \sum_{k=1}^{p} \mu_k(g_k(t,t_s)-1) + \mu_{p+1}(g_{p+1}(t_s)-1) + \sum_{k=p+2}^{p+n-m+1} \mu_k(-t_{m-p+k-1}+1) \quad (10)$$

This Lagrangian will be maximized subject to:

$$b_i \leq -1 \quad \text{for} \quad i = m+1,\ldots,n \quad (11)$$

$$\sum_{i=m+1}^{n} y_{ji} b_i > 0 \quad \text{for} \quad j = 1,2,\ldots,m \quad (12)$$

$$\mu_k \geq 0 \quad \text{for} \quad k = 1,\ldots,p+n-m+1 \quad (13)$$

6. The Proposed Algorithm

The geometric program is solved using the dual program for the augmented problem and the Lagrangian function in equation (10) with constraints (11)-(13).

6.1 Solving the Dual for the Augmented Problem

An initial feasible vector $\underline{b}$ is chosen. Typically $\underline{b} = -1.0$ will work. An initial vector $\underline{\mu}$ is set using the following equation:

$$\mu_k = \lambda_k(\underline{\delta})\ g_0(\underline{t}) \quad \text{for } k = 1,2,\ldots,p+1$$

If condition (9) is not met for some j, and $y_{j,q} < 0$, the vector $\underline{b}$ can be modified by increasing the absolute value of b_q. The vector $\underline{\delta}$ is found using equations (5), (6), and (7). Then $\underline{\delta}$ is used in equation (2) to calculate the value of the dual objective function. Logarithms are taken in equation (3) to yield a linear system of equations with zero degree of difficulty. The vectors $\underline{t}$ and $\underline{t}_s$ are solved for uniquely using Householder transformations. Next, the Lagrangian objective function is solved for $\underline{b}$ and $\underline{\mu}$.

6.2 Solving the Lagrangian Objective Function

The vectors $\underline{t}$ and $\underline{t}_s$ found from the augmented dual are inserted into the Lagrangian function and an augmented Lagrangian method and the projected gradient method are used to solve for $\underline{\mu}$ and $\underline{b}$

respectively. Initial values for μ and b are set.

6.2.1 The Augmented Lagrangian Method for Determining μ Augmented Lagrangian methods or multiplier methods are viewed as a combination of penalty functions and local duality methods[25]. Let a non-linear program have the following form:

$$\text{maximize } f(\mathbf{x})$$
$$\text{subject to } h_k(\mathbf{x}) \leq 0 \text{ for } k=1,2,\ldots,p$$

This program can be written in equality constraints by adding a variable z_k^2 to every term of the constraint:

$$\text{maximize } f(\mathbf{x})$$
$$\text{subject to } h_k(\mathbf{x}) + z_k^2 = 0 \text{ for } k=1,2,\ldots,p$$

For simplicity, define $v_k=z_k^2$. The dual function $\Phi(\mu)$ can be written as:

$$\Phi(\mu) = \min_{\underline{v}\geq 0,\mathbf{x}} \left\{ f(\mathbf{x}) + \mu^T (h(\mathbf{x}) + \underline{v}) + \frac{1}{2} c \left| h(\mathbf{x}) + \underline{v} \right|^2 \right\} \qquad (14)$$

where c is a constant.

Since x only occurs in two of the above terms, the minimization with respect to v_k can be done with the following expression:

$$P_k = \mu_k \left(h_k(x) + \frac{1}{2}c \left(h_k(x) + v_k\right)^2\right) \quad \text{for } k=1,2, \ldots ,p$$

Minimizing with respect to x can be done analytically as follows:

$$v_k = \max(0,-h_k(x) - \mu_k/c) \quad \text{for } k=1,2,\ldots,p$$

Assuming strict complimentary slackness(i.e., if $h_k(x)=0$, then $\mu_k>0$). New values for the Lagrange multipliers can be found using the following:

$$\mu_{k,q+1} = \mu_{k,q} + c\, h[x(\mu_k,v_k,c)] \quad \text{for } k = 1,2, \ldots ,p$$

where q is the iteration number

This is used to estimate a new μ, then this new μ is used in the projected gradient phase to solve for b.

6.2.2 Projected Gradient Method for Determining $\underline{b}$

The projected gradient procedure is based on the method developed by J. B. Rosen[22]. A closed form solution for the partial derivatives of the Lagrangian objective function with respect to $\underline{b}$ is as follows:

$$\frac{\delta L}{\delta b_i} = \mu_{p+1}\, g_{p+1}(\underline{t}_s)\, \log\, t_i \qquad \text{for } i=m,\ldots,n \qquad (15)$$

Initially, $\underline{b}$ is feasible and all constraints are satisfied. A normalized direction vector is found by the following:

$$D_i = \frac{\dfrac{\delta L}{\delta x_i}}{\sqrt{\sum_{j=1}^{n-m+1} \left(\dfrac{\delta L}{\delta x_j}\right)^2}}$$

Let $\underline{x}$ consist of the vectors $\underline{b}$ and $\underline{\mu}$. The following steps are taken:

1. Test for convergence. If the procedure has converged, return to the augmented dual and solve again for $\underline{t}$ and $\underline{t}_s$.

2. Decide whether to keep the current working set or to delete a constraint.

   a. If the last $\alpha_F < \alpha$ then go to step 3.

   b. Calculate the Lagrange multipliers for the current working set. Check for the most negative Lagrange multiplier. Call this constraint index s.

   c. Apply Zoutendijks Rule[23]. If constraint s was previously deleted and the current point is not a stationary point, leave it in the working set and check the next most negative Lagrange multiplier. Call this constraint index s. Go to the beginning of step 2c. If the constraints with negative Lagrange multipliers are exhausted, keep the current working set.

3. Compute the direction vector $\underline{D}$.

4. Compute the step length (calculations described below):

   a. Compute a non-negative α, the maximum feasible step length along D.

   b. Determine α_F using the Armijo Rule[25] with $\beta = 0.5$, $\sigma = 0.01$, and $s = 1.0$.

   c. If $\alpha_F > \alpha$ then use α as the step size.

   d. Call the step size S.

5. If $\alpha_F \geq \alpha$, then add a constraint to the working set. Use the first constraint in w that is satisfied. Go to step 7.

6. If $\alpha_F < \alpha$, maintain the same working set in the next iteration.

7. Calculate the new point x as follows:

$$x_i = x_i + SD_i \qquad \text{for} \quad i = 1,2,\ldots,n+1.$$

6.2.3 Step Size Let W be the set of indices of the constraints in the working sets[24]. Let w be the indices of constraints not in the working set. The matrix A will represent the negative of the

coefficients in the constraint set (11), (12), and (13), where r is the negative constant value of the right hand side. If $a_w^T D \geq 0$, a positive move along d will not violate the constraint, and will impose no restriction on the step length. If $a_w^T D < 0$, there will be a critical step length h_w where the constraint becomes binding. The value of h_w is determined as follows:

$$h_w = \left(\frac{r_w - (a_w)^T x}{(a_w)^T D} \;\middle|\; w \notin W \text{ and } (a_i)^T D < 0 \right)$$

Define α as follows:

$$\alpha = \begin{cases} \min\{h_w\} & \text{if } (a_w)^T D < 0 \text{ for some } w \notin W \\ +\infty & \text{if } (a_w)^T D \geq 0 \text{ for all } w \notin W \end{cases}$$

The value α is the maximum non-negative feasible step length that can be taken along D and is taken as an upper bound on the final step length α_F.

The Armijo Rule is used to find the step length α_F. The step size α_F is set to $\beta^{m_k} s$ where m_k is the smallest non-negative integer for which the following

is true:

$$f(x) - f(x + \beta^m s D) \leq \alpha \beta^m s \nabla f(x)^T D$$

6.2.4 Direction Vector Calculation The direction vector D is calculated as follows:

$$D_i = \frac{\dfrac{\delta L}{\delta x_{qi}} + \sum_{k=1}^{r} \lambda_k \dfrac{\delta g_k}{\delta x_{qi}}}{\left| \sum_{j=1}^{n} \left(\dfrac{\delta L}{\delta x_{qi}} + \sum_{k=1}^{r} \left(\lambda_k \dfrac{\delta g_k}{\delta x_{qi}} \right) \right) \right|^{2\ 1/2}} \qquad (16)$$

for $i=1,2,\ldots,n$

r = number of constraints in the non-working set

q = iteration number for x_q

λ_k for $k=1,2,\ldots,r$ is determined as follows:

$$\sum_{i=1}^{n} \sum_{j=1}^{r} \left(\lambda_j \frac{\delta g_j}{\delta x_{qi}} \frac{\delta g_k}{\delta x_{qi}} \right) = - \sum_{i=1}^{n} \frac{\delta g_k}{\delta x_{qi}} \frac{\delta L}{\delta x_{qi}} \qquad (17)$$

This system of equations provides r equations with r

unknowns, $\lambda_1, \lambda_2, \ldots \lambda_r$ These equations are solved for a unique value of λ. The method uses the non-working set of constraints to find the direction vector. If

$$\frac{\delta L}{\delta x_{qj}} + \sum_{k=1}^{r} \left(\lambda_k \frac{\delta g_k}{\delta x_{qj}} \right)$$

is less than some limit for all $j=1,2,\ldots n$ then convergence is assumed and the procedure stops. Otherwise the process continues. When the projected gradient procedure terminates, b is used in the augmented dual program to find new values, which, in turn generates new values for t and t_s. The procedure continues to cycle through these steps until the value of the primal objective function has converged within 0.0001 to a minimum.

7. Computer Program

A computer program called GEOM was written to perform the entire procedure previously described. It is written in PL/1 and was tested on an IBM 370 system. The program consists of seven segments which will be described individually. The program has been reduced to data flow diagrams[26]. A data flow diagram for GEOM is in Figure 1. The code is

0. GEOM Program

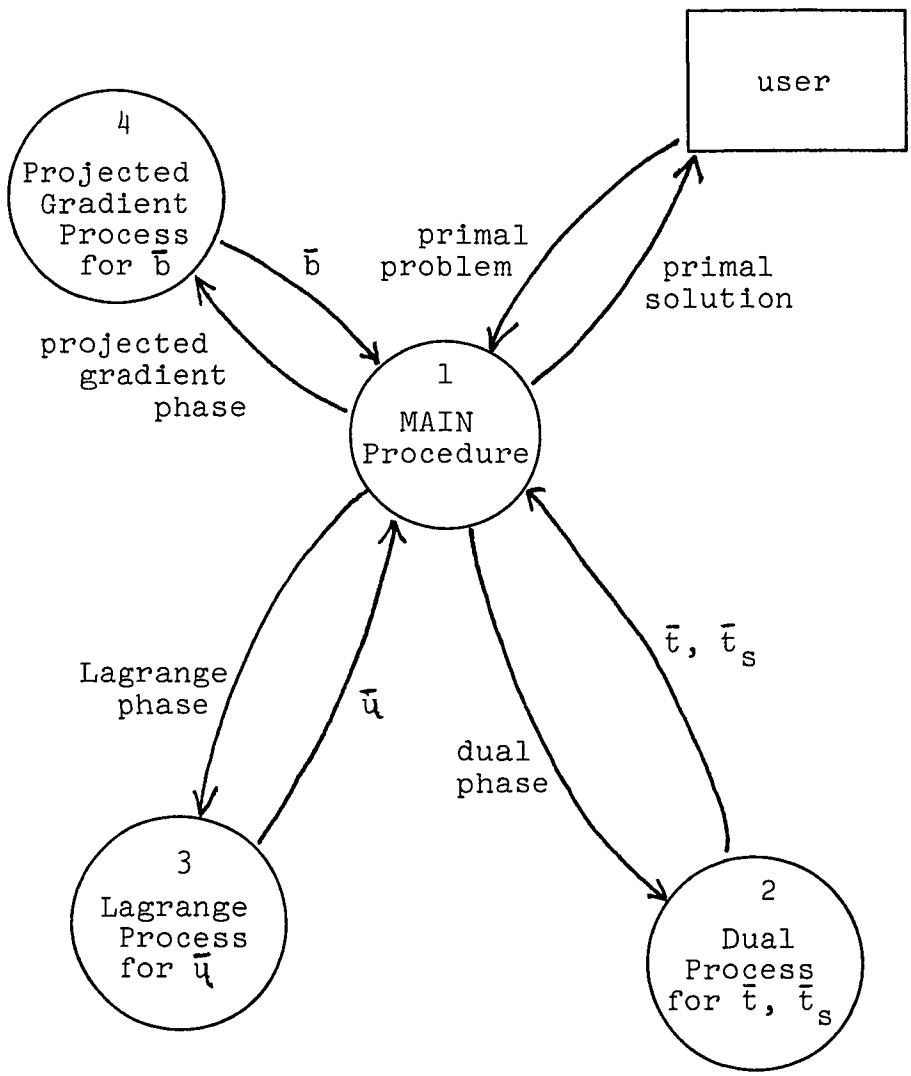

Figure 1

available from the Department of Industrial Engineering, Lehigh University, Bethlehem, Pa.

7.1 MAIN Program

The main program is interactive. All arrays describing the primal geometric program are initialized here, by prompting the user with questions and reading the users responses into appropriate arrays. The user is asked for the number of constraints, the number of variables, and the number of terms in each constraint and in the objective function. The user is then asked to provide the coefficients for each term in the objective function and in the constraints. Next, the user provides the exponents for each variable in each term of the geometric program. Based on these answers, a large character string is built. This represents the primal problem. After all questions are answered, the primal problem is printed to the user.

The augmented program is formed in MAIN. The matrix of exponents of the augmented problem is transformed and matrix operations performed to get an

identity matrix in the first m rows and columns. This matrix is used to get a closed form solution for the dual variables (see equations (5),(6),and (7)).

Feasible starting vectors b and μ are chosen. The vector b is set to -1.0 and the vector μ is set as described previously.

The MAIN program controls the three phases of the procedure, the dual program phase, the augmented Lagrangian phase, and the gradient projection phase. Data flow diagrams for this process are in Figures 2-5. This is done by setting a variable called SWITCH. When SWITCH = 1, the dual program phase is in effect. When SWITCH = 2, the augmented Lagrangian phase then the projected gradient phase is in effect. The first phase is solving for t and t_s using the McNamara's augmented dual program. The dual vector δ is determined using the matrix Y and the vector b. If condition (9) is not met, b_j associated with Y_{ij} is increased in absolute value and δ is determined again. This will only occur in the first iteration since the projected gradient phase insures that condition (9) is met. Then the value for the dual function $v(\delta)$ is calculated for the system of

equations set up by taking logarithms in (3). Given this system of m equations and m unknowns, subroutine MLSQ is called. Subroutine MLSQ solves the system of equations and returns the values for $\underline{t}$ and $\underline{t}_s$. These values are inserted in the adjusted Lagrangian objective function.

Next, the values for μ are found using the augmented Lagrangian approach. These values are then inserted in the adjusted Lagrangian objective function.

Now, the Lagrangian objective function is ready to be maximized with respect to $\underline{b}$ using the projected gradient method subroutine PGRAD. The vector which maximizes the Lagrangian function is returned to the MAIN procedure and saved for use in the augmented dual program. The MAIN procedure then changes the SWITCH variable and proceeds to the augmented dual program phase once again. This process continues until convergence is met on the primal objective function. The resulting vector $\underline{t}$ is displayed to the user.

1. MAIN Procedure

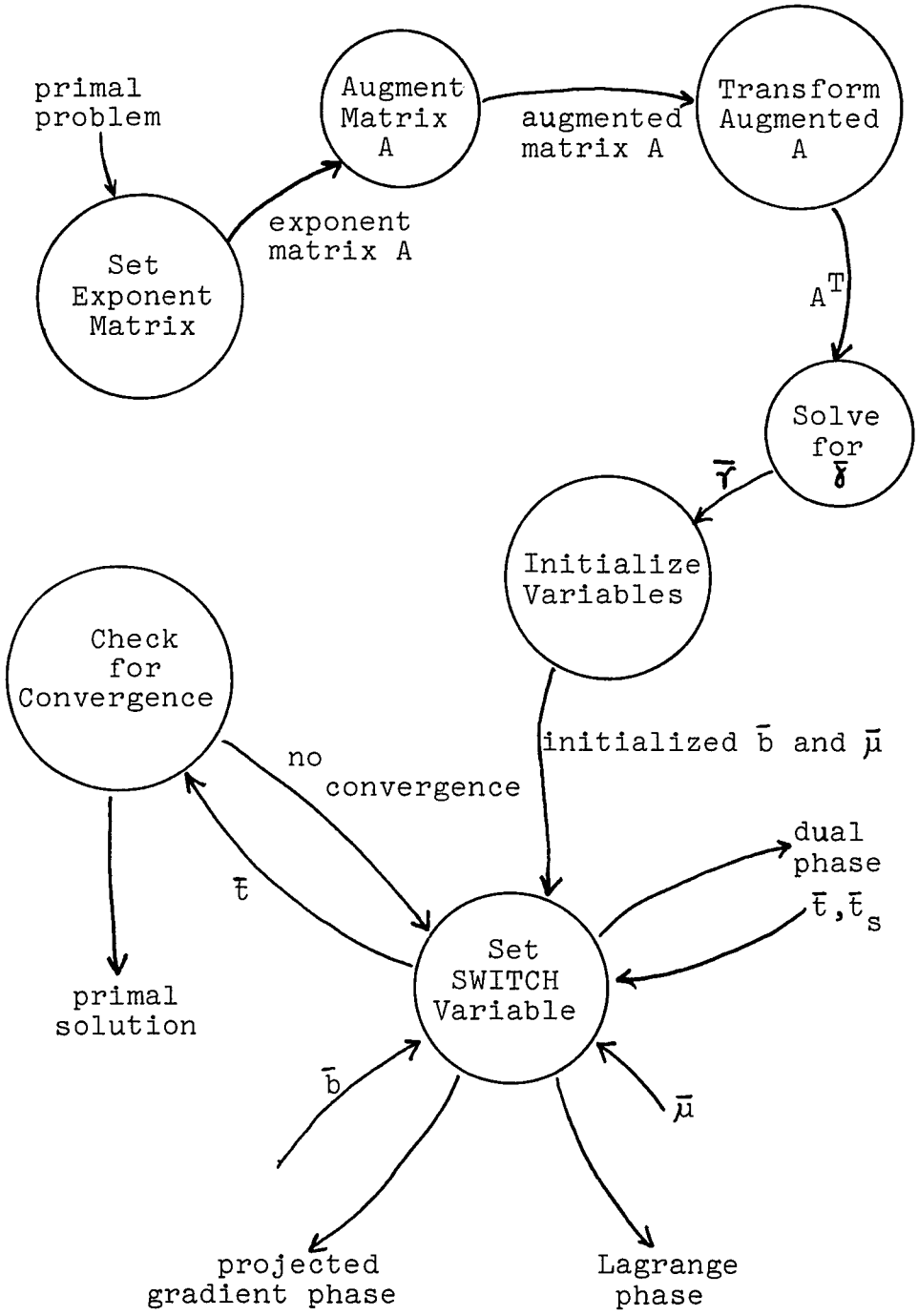

Figure 2

2. Dual Process for $\bar{t}$, $\bar{t}_s$

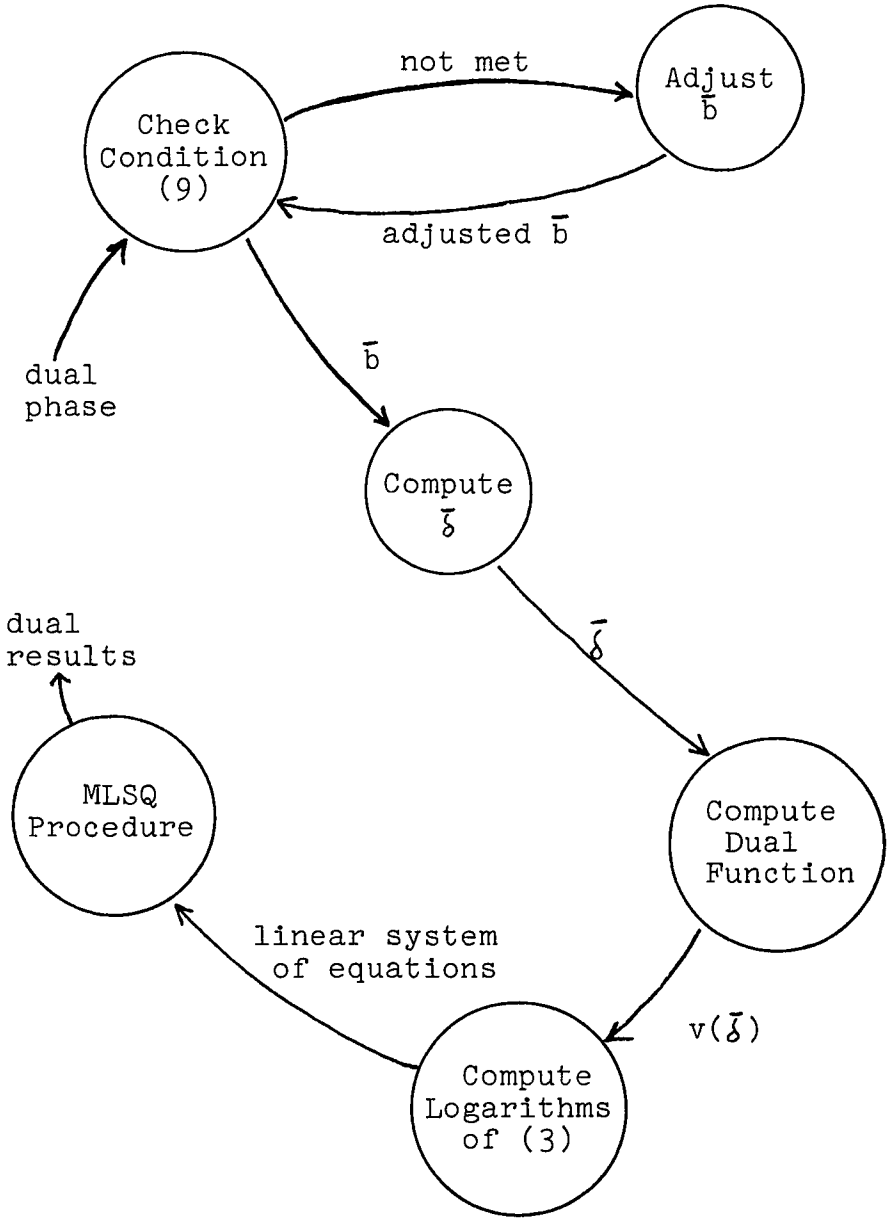

Figure 3

3. Lagrange Process for $\bar{u}$

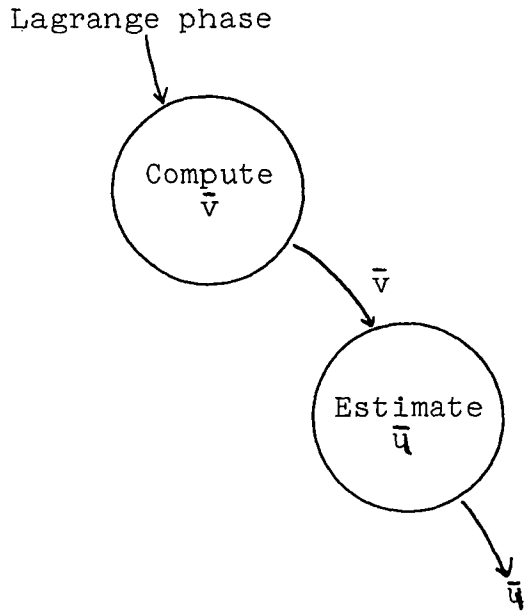

Figure 4

4. Projected Gradient Process for $\bar{b}$

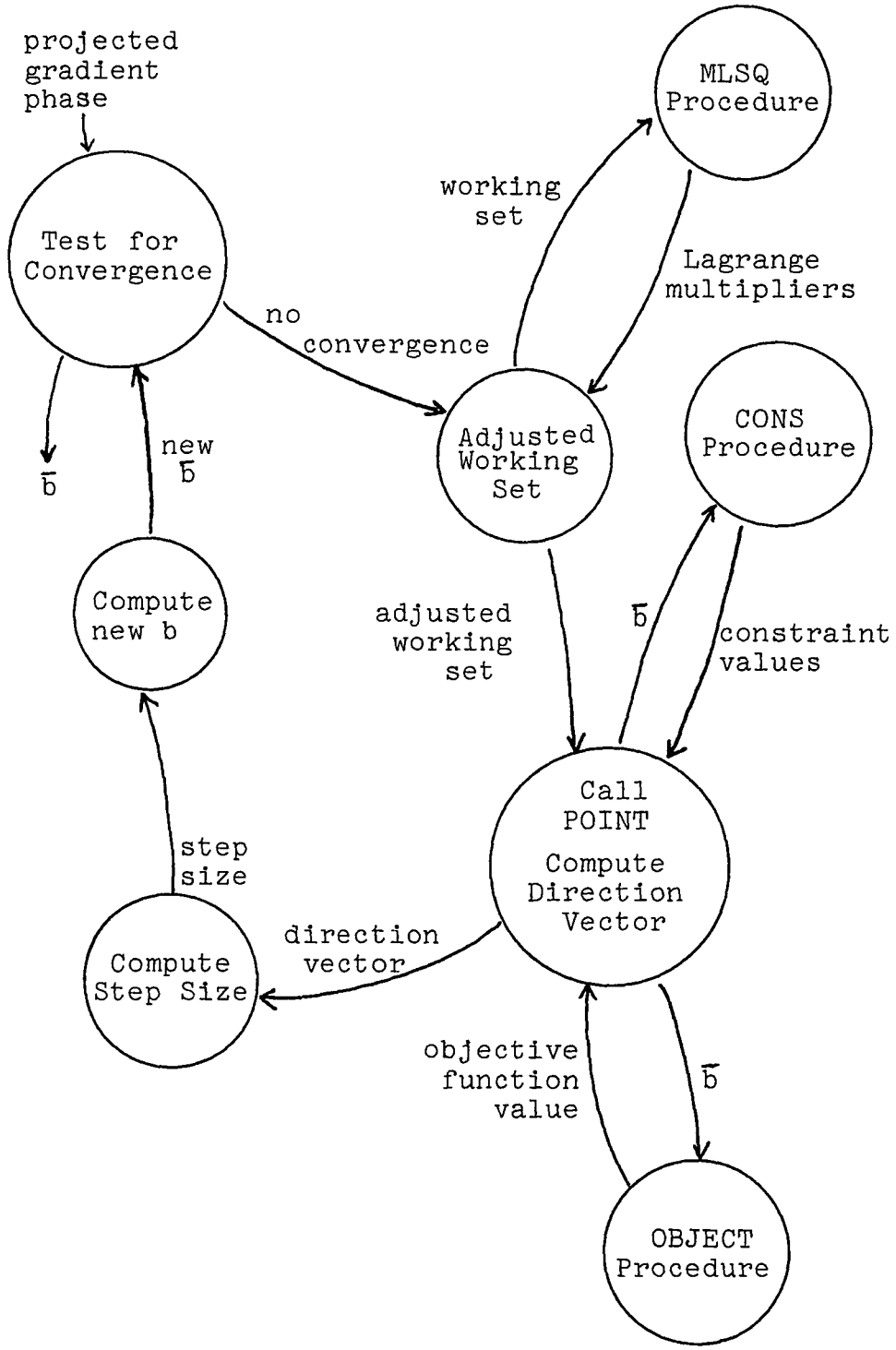

Figure 5

7.2 OBJECT Subroutine

OBJECT forms the Lagrangian function which serves as an objective function in the projected gradient subroutine PGRAD. Calls to the subroutine originate in the projected gradient subroutine PGRAD. The Lagrangian objective function is maximized with respect to vector $\underline{h}$ (See equation (14)) with vectors $\underline{t}$, $\underline{t}_s$ and $\underline{\mu}$ treated as constants. This routine is called from subroutine POINT each time there is a new estimate of the vector $\underline{h}$. The value of equation (10) is returned to subroutine PGRAD.

7.3 POINT Subroutine

The direction vector components are computed in subroutine PGRAD. When subroutine POINT is called, from PGRAD it first calculates the partial derivatives at $\underline{h}$. This is done using equation (15). Direction vectors are calculated using all constraints not in the working set. If all constraints are in the working set, the direction vector is assigned the values of the partial derivatives. When there are constraints in the non-working set the Lagrange multipliers are calculated

as in equation (17). This is done by setting up a system of equations using all constraints not in the working set, and then calling subroutine MLSQ to solve the equations for λ. The direction vectors are found as in equation (16). Values for the partial derivatives and direction vector components are returned. POINT is called each time a new vector h is calculated. The convergence criterion for the projected gradient procedure is also specified in this subroutine. This criterion is:

$$\sum_{n=m+1}^{n} \left(\frac{\delta L}{\delta b_{qi}} + \sum_{k=1}^{r} \left(\lambda_k \frac{\delta g_k}{\delta b_{qi}} \right) \right)^2 \leq 0.0001$$

for i = m+1,2,...,n

r = number of constraints in the non-working set

q = iteration number for x_q

7.4 CONS Subroutine

Subroutine CONS specifies the constraints for the Lagrangian objective function. These are the

constraints defined by equations (11)-(13).

7.5 PGRAD Subroutine

Subroutine PGRAD is a modification of a FORTRAN program called PROJG written at Arizona State University [22][24]. The program was written based on the gradient projection method developed by J. B. Rosen in 1960 [24]. The program is limited to linear constraints. The PROJG program uses subroutines similar in purpose to OBJECT, CONS, and POINT. In PROJG, the user codes the objective function, constraints and partial derivatives into the program code. In PGRAD, these are derived and calculated by the program.

PGRAD was written as a PL/1 version of PROJG with some changes. The subroutines OBJECT, CONS, and POINT serve the same purpose as in PROJG but do not require the user to do any coding. Subroutines OBJECT and CONS form the Lagrangian objective function and the constraints for PGRAD automatically. These subroutines in PGRAD are more complicated and do more calculations than in PROJG. Subroutine POINT differs in that the user does not have to enter the

partial derivatives of the Lagrangian objective function. This would be a very long and difficult task. Instead subroutine POINT calculates a closed form solution for the partial derivatives. A major change in the PROJG program has to do with active constraints. The PROJG program allows a maximum of two violated constraints at a time. Subroutine POINT therefore never handled more than two constraints when calculating direction vectors. This becomes inefficient when there are many constraints. Changes were made so that subroutine POINT incorporates an active set strategy and works with the entire non-working set. This is done by setting up the system of equations in (17) and calculating the Lagrange multipliers using subroutine MLSQ. The values of the Lagrange multipliers are used when deleting a constraint from the working set. Also, the PGRAD algorithm uses the Armijo step size rule described previously. This differs from the rather simple step size rules used in PROJG.

PGRAD is used as a subroutine and is frequently called from the MAIN procedure. The convergence criterion calculated in POINT is checked in PGRAD.

Once the convergence criterion is met, control is passed back to the MAIN procedure.

7.6 MLSQ Subroutine

Subroutine MLSQ is a subroutine found in [27] used for solving a system of linear equations. It is used in two different instances. It is called from the MAIN procedure when solving for t. It is also used from subroutine POINT in calculating Lagrange multipliers when more than one constraint is in the non-working set.

Subroutine MLSQ calculates X satisfying AX = B where A is the coefficient matrix and B is the vector of right hand sides. X is calculated using Householder transformations. A is reduced to upper triangular form using Householder transformations successively and the same transformations are performed on B. The solution X is then obtained using backsubstitution. A discussion of this method is found in [21].

7.7 LMULT Subroutine

Subroutine LMULT is called after the dual phase of the program. The new values for $g_k(\underline{x},\underline{x}_s)$ and $g_k(\underline{x}_s)$ are found first by calling subroutine OBJECT. Based on these values, α is calculated (equation (14)) and then used to calculate updated values for μ. The constant c used in equation (14) is set to 1.0.

8. Results from Running the Program

Since the program runs interactively, there are no input or output files necessary. Standard input and standard output are used (usually the terminal), and therefore need to be allocated. An example run of the program is found in Appendix I.

Five example programs were tested and the proposed procedure was compared to McNamara's method and to a conventional dual approach using a projected gradient method. The results are summarized in the following tables.

Table 1

McNamara's Method

	1	2	3	4	5
Number of Variables	3	4	5	8	10
Number of Terms	7	6	7	12	20
Number of Constraints	3	4	2	3	10
Number of Iterations	131	322	208	1000+	1200+
CPU time	2.26	1.23	5.38	>100	>240

Table 2

Dual Method

	1	2	3	4	5
Number of Variables	3	4	5	8	10
Number of Terms	7	6	7	12	20
Number of Constraints	3	4	2	3	10
CPU time	2.28	7.88	>100	114.31	>240

Table 3

Proposed Method

	1	2	3	4	5
Number of Variables	3	4	5	8	10
Number of Terms	7	6	7	12	20
Number of Constraints	3	4	2	3	10
Number of Iterations	351	15	123	396	600+
CPU time	5.29	15.11	10.30	72.41	>240

It is difficult to draw conclusions from these results. The complexity of the terms in the objective function and constraints plays a role in how quickly the program will converge as well as the distance the initial vector b is from the solution for b. This is clearly seen in problems 1 and 2. Although problem 1 has less variables, terms, and constraints than problem 2, it required more CPU time in all three runs. This may be due to problem 2 having terms involving more variables than in problem 1 and the solution for b is far from the initial value for b. The number of iterations used in solving problem 1 and problem 2 are very different for McNamara's method and the proposed method. The proposed method converged in only 15 iterations, while McNamara's method took 322 iterations. Although the number of iterations for the proposed method is less than for McNamara's method, the CPU time is much greater. This increased time may come from the projected gradient phase, which may have been long to converge. The dual method also took a long time to converge for problem 2 compared to McNamara's method.

The proposed method did converge for program 4, where McNamara's method had not converged as of 100 seconds of CPU time. It was expected that the proposed method would yield better results than McNamara's method as the programs get larger. However, convergence was slow for all methods when testing program 5.

Shortcomings of testing the computer program were found mostly having to do with finding examples. It is difficult to find posynomial programs with large dimension in the literature. Most programs in the literature are signomial. In trying to create programs, difficulties are found in satisfying condition (9) and in creating relationships between variables so that no variable is unconstrained. Another shortcoming in using McNamara's method is when the process does not converge, the method for updating b changes.

Future work includes testing the proposed method with a larger set of programs. Also direct comparisons should be made with other geometric programming algorithms by testing the same set of programs on the same computer. The projected

gradient phase may be improved by incorporating second order differential information in the calculation of the direction vector.

REFERENCES

1. Zener, C., "A Mathematical Aid in Optimizing Engineering Designs", Proc. National Academy Science, Vol. 47, No. 4 (April, 1961), pp. 537-539.

2. Duffin, R. J., "Cost Minimization Problems Treated by Geometric Means", Operations Res., Vol. 10, No.5 (Sept., 1962), pp. 668-675.

3. Duffin, R. J., Peterson, E. L., and Zener, C., Geometric Programming. New York: Wiley, 1967.

4. Passy, U., and Wilde, D. J., "Generalized Polynomial Optimization", SIAM Applied Math., Vol. 15, No. 5 (Sept. 1967), pp. 1344-1356.

5. Wilde, D. J., "A Unified Approach to Multivariable Optimization Theory", Industrial Engr. Chem., Vol. 57 (August, 1965), pp. 18-31.

6. Beightler, C. S., and Phillips, D. T., Applied Geometric Programming. New York: Wiley, 1976.

7. Beightler, C. S., and Phillips, D. T., Foundations of Optimization. New Jersey: Prentice-Hall, 1979.

8. Sarma, P. V. L. N., Martens, X. M., Reklaitis, G. V.,and Rijckaert, M. J., "A Comparison of Computational Strategies for Geometric Programs", Journal of Opt. Theory Applications, Vol. 26, No. 2 (Oct. 1978), pp. 185-203.

9. McNamara, J. R., "A Solution Procedure for Geometric Programming", Operations Research, Vol. 24, No. 1 (Jan. - Feb., 1976).

10. McCormick, G. P., Mylande, W. C., and Fiacco, A. V., Computer Program Implementing the Sequential Unconstrained Minimization Technique for Nonlinear Programming, Research Analysis Corporation, Mclean, Virginia, 1967.

11. Zangwill, W. I., Nonlinear Programming: A Unified Approach. New Jersey: Prentice-Hall, 1969.

12. Dembo, R.S., "GGP: A Computer Program for the Solution of Generalized Geometric Programming Problems", Users Manual, McMaster University, Hamilton, Ontario, Canada (1975).

13. Kelley, J. E., "The Cutting Plane Method for Solving Convex Programs", SIAM, Journal on Applied Mathematics, Vol. 8 (1960), pp. 703-712.

14. Kochenberger, G. A., "Geometric Programming, Extension to Deal with Degrees of Difficulty and Loose Constraints", University of Colorado, PhD Thesis (1969).

15. Duffin, R. J., and Peterson E. L. "Geometric Programs Treated with Slack Variables", Carnegie-Mellon University, Report No. 70-45 (1970).

16. Rijckaert, M. J., and Martens, X. M., "Numerical Aspects of the Use of Slack Variables in Geometric Programming", Katholieke Universiteit te Leuven, Leuven, Belguim, Report No. CE-RM-7501 (1975).

17. Van Dessel, J. P., Personal Comm. in [8], 1975.

18. Beck, P. A., and Ecker, J. G., "A Modified Concave Simplex Algorithm for Geometric Programming", Journal of Optimization Theory and Applications, Vol. 15 (1975), pp. 184-202.

19. Gochet, W., and Smeers, Y., "On the Use of Linear Programs to Solve Prototype Geometric Programs", Center for Operations Research, Heverlee, Belgium, Discussion Paper No. 7229 (1972).

20. Reklaitis, G. V., and Wilde, D. J., "Geometric Programming via a Primal Auxiliary Program", AIIE Transactions, Vol. 6 (1974), pp. 308-317.

21. Golub, G., "Numerical Methods for Solving Linear Least Squares Problems", Numerische Mathematik, Vol. 7 (1965).

22. Kuester, J. L., and Mize, J. H., Optimization Techniques with FORTRAN. McGraw: New York, 1973.

23. Zoutendijk, G., Methods of Feasible Directions. Elsevier: Amsterdam; 1960.

24. Gill, P. E., Murray, W., Wright, M., Practical Optimization. Academic press: New York; 1981.

25. Bertsekas, B. P., Constrained Optimization and Lagrange Methods. Academic Press: New York; 1982.

26. DeMarco, T., Structured Analysis and System Specification. Yourdon: New York; 1979.

27. IBM Corporation, "System 360 Scientific Subroutine Package (PL/1)", 1968.

28. Gill, P. E., Murray, W., Numerical Methods for Constrained Optimization. Academic Press: New York; 1974.

29. Beightler, C. S., Phillips, D. T., Wilde, D. J., Foundations of Optimization. Prentice-Hall: New Jersey; 1979.

30. Lindgren, B. W., Vector Calculus. MacMillan: New York; 1964.

Appendix 1

```
ENTER NUMBER OF TERMS IN OBJECTIVE FUNCTION: 3

ENTER NUMBER OF FORCED CONSTRAINTS: 3

ENTER NUMBER OF VARIABLES: 3

ENTER NUMBER OF TERMS IN CONSTRAINT  1:2

ENTER NUMBER OF TERMS IN CONSTRAINT  2:1

ENTER NUMBER OF TERMS IN CONSTRAINT  3:1

ENTER COEFFICIENTS FOR TERMS IN OBJECTIVE FUNCTION
:
10.0

:
4.0

:
1.0

ENTER COEFFICIENTS FOR TERMS IN CONSTRAINT NUMBER  1
:
0.2

:
0.4

ENTER COEFFICIENTS FOR TERMS IN CONSTRAINT NUMBER  2
:
0.3

ENTER COEFFICIENTS FOR TERMS IN CONSTRAINT NUMBER  3
:
6.0

ENTER MATRIX OF EXPONENTS
ENTER EXPONENTS FOR OBJECTIVE FUNCTION
TERM NUMBER  1
VARIABLE  1:1.6

VARIABLE  2:0.0

VARIABLE  3:0.0

TERM NUMBER  2
VARIABLE  1:0.0
```

```
VARIABLE  2:2.2

VARIABLE  3:0.0

TERM NUMBER  3
VARIABLE  1:0.0

VARIABLE  2:0.0

VARIABLE  3:1.0

ENTER EXPONENTS FOR CONSTRAINT NUMBER                1
TERM NUMBER  1
VARIABLE  1:-2.0

VARIABLE  2:-1.5

VARIABLE  3:0.0

TERM NUMBER  2
VARIABLE  1:0.0

VARIABLE  2:1.1

VARIABLE  3:0.0

ENTER EXPONENTS FOR CONSTRAINT NUMBER                2
TERM NUMBER  1
VARIABLE  1:1.0

VARIABLE  2:-0.8

VARIABLE  3:0.0

ENTER EXPONENTS FOR CONSTRAINT NUMBER                3
TERM NUMBER  1
VARIABLE  1:1.0

VARIABLE  2:1.0

VARIABLE  3:-1.0

THIS IS YOUR SYSTEM OF EQUATIONS

MIN 10.00*T(1)**1.60+4.00*T(2)**2.20+1.00*T(3)**1.00

SUBJECT TO
0.20*T(1)**-2.00*T(2)**-1.50+0.40*T(2)**1.10+<=1
0.30*T(1)**1.00*T(2)**-0.80+<=1
6.00*T(1)**1.00*T(2)**1.00*T(3)**-1.00+<=1
```

```
      T( 1)=   0.4325      T( 2)=   0.6625      T( 3)=   2.0925
 VALUE FOR PRIMAL OBJECTIVE FUNCTION IS    6.3252
      T( 1)=   0.5347      T( 2)=   0.7388      T( 3)=   1.7565
 VALUE FOR PRIMAL OBJECTIVE FUNCTION IS    7.4837
      T( 1)=   0.5708      T( 2)=   0.7479      T( 3)=   1.7866
 VALUE FOR PRIMAL OBJECTIVE FUNCTION IS    7.9755
      T( 1)=   0.5911      T( 2)=   0.7466      T( 3)=   1.8837
1VALUE FOR PRIMAL OBJECTIVE FUNCTION IS    8.2987
```

SKIPPING INTERMEDIATE ITERATIONS

```
      T( 1)=   0.6913      T( 2)=   0.6811      T( 3)=   2.7959
 VALUE FOR PRIMAL OBJECTIVE FUNCTION IS   10.0541
      T( 1)=   0.6914      T( 2)=   0.6811      T( 3)=   2.7960
 VALUE FOR PRIMAL OBJECTIVE FUNCTION IS   10.0542
      T( 1)=   0.6914      T( 2)=   0.6810      T( 3)=   2.7960
1VALUE FOR PRIMAL OBJECTIVE FUNCTION IS   10.0543
      T( 1)=   0.6914      T( 2)=   0.6810      T( 3)=   2.7961
 VALUE FOR PRIMAL OBJECTIVE FUNCTION IS   10.0544
      T( 1)=   0.6914      T( 2)=   0.6810      T( 3)=   2.7961
 VALUE FOR PRIMAL OBJECTIVE FUNCTION IS   10.0545
      T( 1)=   0.6914      T( 2)=   0.6810      T( 3)=   2.7962
 VALUE FOR PRIMAL OBJECTIVE FUNCTION IS   10.0546
      T( 1)=   0.6914      T( 2)=   0.6810      T( 3)=   2.7962
 VALUE FOR PRIMAL OBJECTIVE FUNCTION IS   10.0547
      T( 1)=   0.6914      T( 2)=   0.6810      T( 3)=   2.7963
 VALUE FOR PRIMAL OBJECTIVE FUNCTION IS   10.0548
```

Appendix II

Patricia Douglas-Jarvis

Born: April 18, 1957 in Dover, New Jersey

Education: Bachelor of Science in Mathematics and Biology with honors from the Mathematics Department from Washington College, Maryland, 1979.

Experience: AT&T Bell Laboratories, 1979 to present.

Patricia Douglas-Jarvis is a Member of Technical Staff at AT&T Bell Laboratories in Whippany, New Jersey. From 1979 to 1981 she was involved with software development for the Quality Assurance Center. This involved computerized systems for auditing quality related data from factories nationwide. Most of her work in this was in the area of graphics. She also developed software for data collection for various field tracking studies. From 1981 to present, she has been in the area of device reliability and reliability analysis. This includes some software development as well as involvement in reliability experiments.