T-3672

A GEOMETRIC PROGRAMMING-BASED ALGORITHM FOR PREPROCESSING NONLINEAR SIGNOMIAL OPTIMIZATION PROGRAMS

by

John B Kirk

T-3672

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Mineral Economics).

Golden, Colorado
Date 11/17/88

Signed: _____
John B Kirk

Approved: _____
Dr. R. E. D. Woolsey
Thesis Advisor

Golden, Colorado
Date 11/17/88

_____
Dr. John E. Tilton
Professor and Head
Mineral Economics Department

ABSTRACT

This thesis presents a systematic approach to the initial evaluation of nonlinear signomial optimization problems in an algorithm that relies on common sense, addition, subtraction, algebraic manipulation, and geometric programming. Using problems from the literature and based on computational experience it is shown that the algorithm will solve or bound the answer close enough that further analysis may be delayed or proven to be unnecessary. In other cases it will be shown that it will provide a properly structured problem that a computer program can solve rapidly. In some cases the algorithm will provide no information but may aid the user in selecting the most appropriate computer code to apply to the problem.

Working through the algorithm the user may be able to modify his particular formulation to take advantage of the structural convexity inherent in geometric programming. This can be accomplished either by releasing variables previously considered to be fixed or by fixing variables which are uncontrollable. Additionally, this process can provide the user with the understanding of the optimization process necessary to pose separable convex or concave programs and subprograms.

One of the central problems in convincing managers to use optimization techniques is the complexity of the algorithms and the indirect-

ness of the solution process. As shown in this thesis this algorithm provides a solution technique that allows the analyst to answer optimization questions directly. It follows that while this method solves some problems it cannot solve the vast majority but it will engage the manager in the solution process and demonstrate the need for other methods, thereby increasing their use.

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGEMENTS vii

Chapter 1 INTRODUCTION 1

1.1 Purpose 1

1.2 Description of the Algorithm 4

1.3 Geometric Programming 5

Chapter 2 CONCEPTS 7

2.1 Algorithms 7

2.2 Understanding Theory 10

2.3 Some Helpful Definitions 13

2.4 The Geometric Inequality 14

2.5 The Formulation of the Dual Problem 16

2.6 Optimality Conditions 20

2.7 Optimality Relationships in the Objective Function 22

2.8 Optimality Relationships in the Constraints 23

2.9 Classification of General Nonlinear Optimization Problems 24

2.10 Regression Analysis in Nonlinear Programming 26

Chapter 3 PREPROCESSING ALGORITHM 28
3.1 About the Algorithm 28
3.2 Make a Worksheet 29
3.3 The Algorithm 30
3.4 Analyzing the Results 53
Chapter 4 CONCLUSIONS 54
4.1 Summary of Accomplishments 54
4.2 Applications of Algorithm 55
4.3 Suggestions for Further Research 56
REFERENCES CITED 58
Appendix A: ALGORITHM WORKSHEET 60
Appendix B: MARKETING MIX PROBLEM 86
Appendix C: HELICAL SPRING DESIGN PROBLEM 118
Appendix D: AN EXAMPLE FOR MANIPULATING THE DELTA$_{GP}$ MATRIX 140

ACKNOWLEDGMENTS

Many students, faculty and staff members have contributed to my education at the Colorado School of Mines. Though unnamed here I am sincerely grateful for their encouragement and support.

Dr. R. E. D. Woolsey has been mentor and advisor during my pursuit of this degree. He has given freely of his time training, educating, and coaching me through a very diverse and at times difficult process. Dr. David L. Olson provide the means for me to complete the degree and provided me the opportunity to observe the university from a different and important perspective. Dr. Mogens Henriksen provided many opportunities for growth and contributed extensively to the integration of my engineering background and operations research studies. Dr. David Fletcher provided a great deal of his time for discussions of management, which have done much to shape my goals and ideals. Dr. Robert Drury who helped me develop an intellectual perspective for integrating the engineering, economics, and operations research components of my program. Warren Spaulding has helped me not only with the educational program by serving on my committee but he has been a mentor and friend during the transition from Army life to student and faculty member.

A special thanks to the Guild both past and present for their

comments, help in putting this thesis together, their encouragement and support, and development of the concepts included in this thesis. Particularly, Gys Wessels, Timucin Kilinc, Dick Hewitt, Cecilia Oatney, Jeff Wilkenson, Jim Thome, Scott Bailey, and Jean Goldberg. Special thanks to a real friend and the provider of the technology to produce this thesis, a man I will always be indebted to, Craig Graham.

Chapter 1
INTRODUCTION

1.1 Purpose

Procedures common to business include accounting standards and practices, manufacturing assembly line work rules, and a host of others down to and including coffee break rules for the typing pool. These procedures are established to facilitate the business operations and simplify the process of getting things done. While mathematics provides rules and neglects their order, an algorithm provides the procedures for invoking the rules and solving the problem.

The algorithm developed in this thesis provides a systematic approach to the initial evaluation of nonlinear signomial optimization problems based on geometric programming techniques. These relationships are used to identify the separable subprograms, exploit the information available in the constraints, identify unstated constraints, and analyze the formulation of the program. Managers and modelers with minimal mathematical programming experience will find this method useful in analyzing problems before they decide on a computer algorithm or engage an analyst. This approach uses addition, subtraction, algebraic manipulation, and common sense to interpret the geometric programming optimality relationships.

Every step of this algorithm is based on mathematical theory. In some cases the source is specific enough to reference, in others it is sufficiently common knowledge that no reference is made. The user is

referred to any standard linear algebra text for a more complete development of the theory involved in this algorithm.

Selected examples demonstrate that the problem's solution can be developed from this algorithm. In many cases the optimum of the objective function or the variables or both can be approximated such that further analysis is undesirable. This method lends itself to solution by inspection. It was developed because the author was fascinated by Dr. R. E. D. Woolsey's ability to solve seemingly complex problems without the benefit of either a calculator or computer. In other cases it provides a properly structured problem that a computer program may solve rapidly. Sometimes the algorithm will provide no information but will aid the user in selecting the most appropriate computer algorithm or code.

Working through the algorithm, modifications of the formulation to take advantage of the structural convexity inherent in geometric programming will become apparent. The decision maker can also change the complexity of the problem by either releasing fixed variables or by fixing independent variables at a "normal" operating value. This is especially helpful if the manager has little or no control over that particular independent variable. This will allow the decision maker to vary the program's degree of difficulty and potentially pose a well conditioned problem.

Traditional operations research practitioners have advocated the preprocessing of optimization programs to take advantage of the structure of the problem, the knowledge of the decision maker, and the solu-

tion techniques reasonably available. This algorithm provides the decision maker the necessary understanding of the optimization process to pose a reasonable set of separable subprograms. Using it a manager may sit down with his staff, refine the model, evaluate the constraint set, bound the solution, and thereby, developing an understanding of the problem before bringing in an analyst. In doing so, he will gain computational insight, develop a qualitative understanding of the interaction of the variables, and avoid development pitfalls in the optimization program.

It is explicitly granted that the mathematician with computational experience and a strong background in optimization will find this to be little more than a check sheet. It is unlikely, however, that the mathematician will have the qualitative background and experience to understand the problems this algorithm is designed for. The problem with nonlinear optimization techniques is that they are often not used because the solution techniques available are necessarily complex and there is a general lack of awareness among practicioners of the methods available and their potential uses. By providing a self-explanatory solution technique which shows the need for other methods, general use of these techniques should increase.

Many objective functions are developed by statistical or regression analysis. There is significant information available in the regression analysis which can be used to simplify the optimization process. For example, most independent variables in the regression data set can be

bounded by their mean and standard deviations. Regression usually trades off model sophistication for goodness of fit. Seldom does the regression trade off goodness of fit for ease in optimization. There are some good trade-offs available here but no general rule for applying them. The decision maker is the only one in a position to make this type of trade-off.

1.2 Description of the Algorithm

The initial steps of the algorithm reduce the problem to its simplest form by eliminating terms, constraint redundancies, and bound redundancies. The next steps bound the independent variables in the constraints, the context of the program, and the regression statistics. Intermediate steps include evaluating variable contributions in the objective functions and setting unconstrained variables at their bounds, separating the unconstrained subprograms, setting up the geometric programming dual function and its linear constraints and analyzing variable relationships in the dual formulation. Finally, the information gained is used to classify the program for the selection of the appropriate computer code to solve the residual problem.

The class of problems that this algorithm can preprocess includes the set of generalized polynomials called signomials. They differ from polynomials in the inclusion of real exponents and coefficients i.e., objective functions and constraints of the form:

$$\sum_{i=1}^{n} \sigma_i \, Coeff_i \, Var_i^{\,e_i}$$

Where σ_t is *1* or *-1*, and e_t is any real number.

Polynomials have both positive and negative coefficients and only positive integer exponents. Although initial steps in the algorithm call for the substitution of series expansions for trigonometric and logarithmic functions, these may not be desirable and may needlessly complicate the solution. Trigonometric or logarithmic function programs can be solved with other methods more efficiently.

1.3 Geometric Programming

Geometric programming, a simple solution method, requires little more than first courses in derivative calculus and vector algebra. It has excellent potential for business, economic, and engineering problem analysis because of its common sense appeal. Analyses using geometric programming closely parallel those used by an experienced decision maker and generally support his intuition. Geometric programming, however, may not be a particularly effective tool for solving nonlinear optimization problems when applied indiscriminately to a broad class of problems. This algorithm does not rely on the solution to the geometric dual function, rather it examines the optimality conditions to establish additional relationships between the variables.

A qualitative understanding of the interaction of the program variables and the objective function is often more important to the decision maker than the actual answer. If a manager can see how a variable affects the solution then decisions which reduce or increase that variable will be easier to evaluate. Likewise, if the variables are properly bounded

he can determine the feasibility of the current operating variable values. On the other hand, if he is making a profit and several variables are out of bounds then he can readily see that he needs a new model.

An example of an embedded problem involves the single variable bound. In the design of a steel reinforced concrete beam, where minimizing the cross section (which directly determines the weight and hence cost) of the beam is the principal optimization goal. An important variable to this optimization program is the yield strength of the steel. Yield strength as opposed to ulitmate strength, where failure of the steel member actually occurs, is the point where deformation occurs nonlinearly with additional stress. Design of reinforced concrete is governed by state and local building codes which usually reference the American Concrete Institute's design procedures. The yield strength of steel used in reinforced concrete design is bound by that design procedure and hence by law. Additionally, the yield strengths of steel reinforcing bar available from manufacturers are discrete and limited.

Likewise the discreteness of a variable may significantly reduce the difficulty in enumerating the solution possibilities. For example, suppose the design of a pipeline system consists of determining the number of pumping stations and the selection of pipe diameters. It is not possible to build half of a pumping station and pipe is only manufactured in specific diameters. Therefore, within a small range on each of these variables the possible combinations can be enumerated to find the optimum.

Chapter 2
CONCEPTS

2.1 Algorithms

An algorithm is a step-by-step problem-solving procedure. Traditionally, programmers and mathematicians write algorithms before beginning the detailed and problem of programming. It is also a method of communicating the solution process. Often managers brought up through the ranks will claim that they can make accurate decisions without mathematical analysis. These managers talk about having a feel for the problem. Casual observers may think that the data they use to solve these problems have no apparent bearing on the problem. This intuitive and oft times unconscious process most often involves the evaluation of what cannot be done. In other words, the manager is evaluating the problem against a set of unstated constraints. Optimization is at once both, a rigid mathematical process with firm rules and procedures, and a creative intuitive process calling for judgement and acceptance of risk.

Providing a deductive ordering of instructions allows the engineer, economist, or manager to explore his model for the unstated constraints, both intuitively, drawing on his knowledge of the process, and exhaustively, evaluating every reasonable combination of variables. Once these constraints are identified they can be included in the evaluation of the stated constraints. As an example, knowing that production must be maintained above a certain level (constraint), the model to minimize total cost can be evaluated in its proper context. Likewise,

minimizing cost to the point production is zero, if the real goal is to maximize profit (dual function), has no meaning. The user knows, regardless of payoff, his manufacturing plant's production is limited by a set of immutable resource and capacity constraints, at least in the short run. In many cases actions are taken (i.e., constraints are active) in the short term which have no bearing on the overall long run operational direction. The manager knows how and why that action was necessary and when the action is no longer valid.

Understanding the model and the optimization process allows the algorithm's user to bring his knowledge and intuitive understanding of the problem to bear. In most engineering, economic, and business problems the objective function variables are much more tightly bound than we as analysts would like to admit, making optimization less important than feasibility. An example of knowing the process is found in the marketing mix problem (Balachandran and Gensch 1974), where the "optimum" solution identifies last period's advertising and current in-store promotion variables as critical to maximizing sales in the current period. The stated constraints cause the mathematical algorithm to trade current advertising for current in-store promotion, in other word sacrificing future profits for current profits. Knowing that the short term is often sacrificed for the long run but that the long run is seldom sacrificed for the short run should cause the modeler to reevaluate his model. Once identified this dichotomy can be resolved by rewriting the objective function to optimize and balance these variables in both periods.

Sensitivity analysis of constrained optimization programs often means analyzing the constraints to find out how changing a constraint will affect the objective function. If the user can bound the variables accurately based on experience, common sense, and an intuitive understanding of the meaning of each constraint in the context of the program, then he is well on the way toward effective optimization. An example of how this can be done is found in the analysis of cost minimization for a manufacturing plant. The current contracted production represents a real lower bound and the sales forecast is a reasonable upper bound on total production. In this case, the lower bound is hard and the upper bound is soft. If our analysis shows that the variable is optimum at its upper bound then we will think of it in terms of what it could be; if it is optimum at its lower bound then we will fix it at the hard limit. If sensitivity analysis shows that capacity is constrained to the point that production is inefficient, then the manager may be able to change the problem statement by allocating additional resources to increase capacity.

The sensitivity analysis of the variables in a computer algorithm will not necessarily pinpoint this capacity variable. If it does the manager or the analyst may not recognize its importance without having done the type of analysis this algorithm requires. Therefore, working through the problem the manager should see these inter-relationships (sensitivities) among the operating variables. Even if he cannot make the variable "optimal" he can move the process in the right direction.

2.2 Understanding Theory

Since an understanding of nonlinear optimization theory, particularly geometric programming, is helpful, the reader is referred to Duffin, Peterson, and Zener (1967) and Beightler and Phillips (1976) for an exposure to the underlying geometric programming relationships. While the mathematical representations used in the first may be confusing at first once the reader evaluates one or two functions it is much easier to understand. With such an understanding most of the analysis will appear to be relatively easy. The key to the successful prosecution of the algorithm is a thorough understanding of the problem on which the model is based. Every practitioner should have a healthy skepticism for the data and the model. This skepticism will prevent unnecessarily detailed analysis and allow more time for refining the model.

A minimization problem is best solved by minimizing the variables and terms that make the goal (objective function) larger (i.e., the minimizing the positive parts of the program) and maximizing the variables and terms that make the goal smaller (i.e., maximizing the negative parts). Likewise a maximization problem maximizes positive variables and terms and minimizes negative variables and terms.

Positive variables appear with positive exponents in terms with positive coefficients or with negative exponents in terms with negative coefficients. Negative variables appear with negative exponents in terms with positive coefficients or positive exponents in terms with negative coefficients. The geometric programming sign ($sign_{GP}$) of a variable is a

convenient way of remembering whether the variable is positive or negative.

The original work in geometric programming was completed by Duffin, Peterson, and Zener (1967). Their text outlines the basic concepts of posynomial programming and their mathematical relationships to duality, convexity, Kuhn-Tucker conditions, and Lagrange multipliers. Since that time, extensions of geometric programming have included condensation techniques (Passy 1969), signomial programming (Duffin and Peterson 1973), and many others. Rijckaert and Martins (1976), Dembo (1978), and Ecker (1980) have compiled extensive reviews and bibliographies of the geometric programming literature and software.

Published work in Geometric Programming has declined in the past ten years and except for a book edited by Allen (Forthcoming), no major publications were found to be contemplated. Several researchers at the Colorado School of Mines have continued basic research in the area of condensation and signomial programming. Other researchers continue the development of concepts outlined by Peterson, Duffin, Zener, and others. Nothing published in the last ten years appears to affect the concepts used in this thesis.

Peterson (1976) presents a full review of the theoretical development of posynomial geometric programming for both the unconstrained and constrained cases. He includes a discussion of the optimality conditions, duality, and decomposition in a rigorous manner. Duffin and Peterson

(1973) present a method of transforming signomial programs into equivalent posynomial programs. Passy (1969) reports the development of a method of changing signomial constraints into posynomial constraints using condensation.

The concept developed in this thesis was introduced in a text edited by Beightler and Phillips (1976) which contained chapters contributed by R. E. D. Woolsey, and R. S. Dembo. Woolsey's contribution was entitled "Practical Aspects of Geometric Problem Solving." It included subsections entitled "Reducing Degrees of Difficulty by Variable Elimination" and "Reducing Degrees of Difficulty by Engineering Knowledge of the System and Common Sense." Dembo's contribution was a chapter entitled "Selected Applications of Geometric Programming" including a subsection "Solution to a "Marketing Mix" Problem." Research, lectures, and working papers by Woolsey and graduate students at the Colorado School of Mines in linear, integer, and geometric programming complete the basis of this work.

Each of these works focuses on the separability of the nonlinear programs found in engineering, economics, and business. Geometric programming is ideally suited for solving these types of problems by exploiting the algebraic structure of the problems. The algorithm proposed in this thesis takes this concept one step further by using geometric programming relationships to identify the separable subprograms.

2.3 Some Helpful Definitions

$Sign_{GP}$ - The $sign_{GP}$ of each term is the product of the sign on the coefficient and the sign on the variable's exponent.

Term - A product consisting of a constant coefficient and variables separated from other terms by a plus (+) or minus (-) sign. Mathematically we can describe a term as:

$$\sigma C \prod_{i=1}^{n} x_i^{e_i}.$$

Where $\sigma = +1$ or -1, C is the term's coefficient, and x_i are the variables in the program and e_i are the exponents of the program's variables in the term.

Polynomials - A summation of multi-variable terms with positive integer exponents, positive and negative coefficients, and real independent variables. In equation form this is:

$$\sum_{j=1}^{m} \sigma_j C_j \prod_{i=1}^{n} x_{ij}^{e_{ij}}.$$

Where $\sigma_j = +1$ or -1, C_j is the term's coefficient, and x_{ij} are the variables in the program and e_{ij} are the positive integer exponents of the program's variables in the term.

Posynomials - A generalized polynomial with real exponents, positive coefficients, and strictly positive independent variables. Mathematically it is similar to the polynomial:

$$\sum_{j=1}^{m} \sigma_j^2 C_j \prod_{i=1}^{n} x_{ij}^{e_{ij}}.$$

Where $\sigma_j = +1$ or -1, C_j is the term's coefficient, and x_{ij} are strictly positive variables in the program and e_{ij} are the real exponents of the program's variables in the term.

Signomials - A generalized polynomial with real exponents and real coefficients, and strictly positive independent variables.

$$\sum_{j=1}^{m} \sigma_j C_j \prod_{i=1}^{n} x_{ij}^{e_{ij}}.$$

Where $\sigma_j = +1$ or -1, C_j is the term's coefficient, and x_{ij} are strictly positive variables in the program and e_{ij} are the real exponents of the program's variables in the term.

Objective Function - The mathematical representation of the optimization's goal (e.g., a cost or production function).

Constraints - The signomial representation of the bounds on the solution due to limited resources.

Single Variable Bounds - A constraint that includes only one variable with either an upper bound, a lower bound, or both (*e.g.*, $2 \le x \le 4$ *or* $x \ge 2$ *or* $x \le 4$).

2.4 The Geometric Inequality

Duffin, Peterson, and Zener (1967) explained the geometric inequality elegantly by showing that

$$\frac{U_1 + U_2}{2} \quad \ge \quad U_1^{1/2} U_2^{1/2}$$

Understanding the proof of this relationship will make acceptance

of the optimality conditions or rules easier. Proofs of convexity and sophisticated mathematical proofs have been published for this relationship in abundance and will not be repeated here. Geometric programming depends on this distinction and hence the posynomial designation for generalized polynomials with positive coefficients and real exponents.

Signomial programs cannot be solved globally with generalized geometric programming techniques. This is also true for any nonlinear programming technique. Since the convexity of the function and the constraint region is difficult if not impossible to establish the convexity of signomial programs. This algorithm does not guarantee a solution vector. It does allow the user to reduce the region in which he or his solution technique will search for the solution.

The algebraic proof follows.

Let U_1 and U_2 be any non-negative numbers. Then:

$$(U_1 - U_2)^2 \geq 0,$$

or

$$U_1^2 - 2U_1U_2 + U_2^2 \geq 0.$$

Adding $4U_1U_2$ to both sides the inequality becomes:

$$U_1^2 + 2U_1U_2 + U_2^2 \geq 4U_1U_2,$$

or

$$(U_1 + U_2)^2 \geq 4U_1U_2.$$

Taking the square root of both sides:

$$U_1 + U_2 \geq 2U_1^{1/2}U_2^{1/2},$$

or

$$\frac{U_1+U_2}{2} \geq U_1^{1/2}U_2^{1/2}.$$

The geometric inequality becomes an equality if and only if U_1 and U_2 are equal. In order to be more general, let U_1 and U_2 represent terms with non-negative values. This can be shown to expand to the generalized form of the inequality which is:

$$\delta_1 U_1+\delta_2 U_2+\ldots+\delta_n U_n \geq U_1^{\delta_1}U_2^{\delta_2}\ldots U_n^{\delta_n}$$

where, $\delta_1+\delta_2+\ldots+\delta_n=1,$

and $\delta_i>0.$

This can be expanded by introducing the variable σ which will be defined as either +1 or -1. This allows the same development of the normality conditions for signomial problems. The signomial equivalent is:

$$\sigma_1\delta_1+\sigma_2\delta_2+\ldots+\sigma_n\delta_n = \sigma$$

and $\delta_i>0.$

Returning to the posynomial formulation. Let $u_i=\delta_i U_i$ and the inequality is in the standard form for geometric programming:

$$u_1+u_2+\ldots+u_n \geq \left(\frac{u_1}{\delta_1}\right)^{\delta_1}\left(\frac{u_2}{\delta_2}\right)^{\delta_2}\ldots\left(\frac{u_n}{\delta_n}\right)^{\delta_n}.$$

2.5 The Formulation of the Dual Problem

The formulation of the dual program is important to the development of the optimality equations for the objective function terms and the

constraint terms. These optimality conditions define additional relationships between the variables which will better define the programs. Duffin, Peterson, and Zener developed geometric programming based on duality theory and the requirement that the minimum of the sum

$$u_1 + u_2 + \ldots + u_n$$

is greater than the maximum of the product

$$\left(\frac{u_1}{\delta_1}\right)^{\delta_1}\left(\frac{u_2}{\delta_2}\right)^{\delta_2}\cdots\left(\frac{u_n}{\delta_n}\right)^{\delta_n}$$

which is true if and only if

$$\delta_1 + \delta_2 + \quad \ldots \quad + \delta_n = 1$$

and

$$\delta_i > 0$$

The inherent elegance of geometric programming is that it converts the problem from a (primal) nonlinear minimization program into a (dual) concave maximization program with linear constraints. If the solution to the dual is found in the square matrix of independent linear constraints, the program is said to have, zero degree of difficulty. The way to calculate the degree of difficulty is to count the number of terms in the objective function and the constraints and subtract the number of variables plus one from this count. This is equivalent to the number of columns in the dual constraint matrix minus the number of rows. The basis for this

primal-dual relationship is set forth in Farkas' Lemma of linear programming which states that a feasible solution to the primal is an infeasible solution to the dual except at optimality.

The solution to the unconstrained problem is not unimportant. In fact, it is a necessary piece of information for decision making when constraints can be eliminated by buying additional resources. In the algorithm the identification of separable unconstrained subprograms will make use of this information to bound the optimization program solution. However, the central theme of this algorithm is the identification of constraints and the narrowing of the constraint region until the solution can be found directly.

A constrained optimization program includes both a function to be optimized, called the objective function, and constraints. Geometric programming requires that all constraints be in the form of a signomial less than or equal to zero. This form will facilitate the development of the dual constrained function.

To develop the unconstrained dual function the concept of weights must be revisited. In the unconstrained dual the δs used in the geometric inequality were normalized. This was acceptable because the relationship was unbounded. The development of the dual constrained function requires that a dual be developed with weights that are not normalized for the constraints. Borrowing from Duffin and Peterson (1973):

A constraint of the form

$$1 \geq c_1 + c_2 + \ldots + c_n$$

can be converted by use of the geometric inequality to

$$1 \geq c_1 + c_2 + \ldots + c_n \geq \left(\frac{c_1}{\delta_1}\right)^{\delta_1}\left(\frac{c_2}{\delta_2}\right)^{\delta_2}\ldots\left(\frac{c_n}{\delta_n}\right)^{\delta_n}$$

where

$$\delta_1 + \delta_2 + \ \ldots \ + \delta_n = 1$$

Define weights ω_i such that,

$$\omega_1 + \omega_2 + \ \ldots \ + \omega_n = \lambda$$

Let, $\delta_i = \frac{\omega_i}{\lambda}$

and substituting, the equation becomes

$$1 \geq c_1 + c_2 + \ldots + c_n \geq \left(\frac{c_1}{\frac{\omega_1}{\lambda}}\right)^{\frac{\omega_1}{\lambda}}\left(\frac{c_2}{\frac{\omega_2}{\lambda}}\right)^{\frac{\omega_2}{\lambda}} \ \ldots \ \left(\frac{c_n}{\frac{\omega_n}{\lambda}}\right)^{\frac{\omega_n}{\lambda}}$$

or

$$1 \geq c_1 + c_2 + \ldots + c_n \geq \left(\frac{c_1}{\omega_1}\right)^{\frac{\omega_1}{\lambda}}\left(\frac{c_2}{\omega_2}\right)^{\frac{\omega_2}{\lambda}}\ldots\left(\frac{c_n}{\omega_n}\right)^{\frac{\omega_n}{\lambda}} \cdot \lambda^{\left(\frac{\omega_1}{\lambda}\right)+\left(\frac{\omega_2}{\lambda}\right)+\ldots+\left(\frac{\omega_n}{\lambda}\right)}.$$

Since,

$$\frac{\omega_1}{\lambda} + \frac{\omega_2}{\lambda} + \ldots + \frac{\omega_n}{\lambda} \ = \ 1,$$

the equation becomes

$$1 \geq c_1 + c_2 + \ldots + c_n \geq \left(\frac{c_1}{\omega_1}\right)^{\frac{\omega_1}{\lambda}}\left(\frac{c_2}{\omega_2}\right)^{\frac{\omega_2}{\lambda}}\ldots\left(\frac{c_n}{\omega_n}\right)^{\frac{\omega_n}{\lambda}} \cdot \lambda$$

Raising everything to the power of the sum of the weights,

$$1^{\lambda} \geq c_1 + c_2 + \ldots + c_n{}^{\lambda} \geq \left(\frac{c_1}{\omega_1}\right)^{\omega_1}\left(\frac{c_2}{\omega_2}\right)^{\omega_2}\ldots\left(\frac{c_n}{\omega_n}\right)^{\omega_n}\cdot\lambda^{\lambda}$$

Note: One to any power is still one.

Multiply the unconstrained dual function by the extreme right- and left-hand sides of the constraint:

$$(u_1 + u_2 + \ldots + u_n)\cdot 1 \geq \left(\frac{u_1}{\delta_1}\right)^{\delta_1}\left(\frac{u_2}{\delta_2}\right)^{\delta_2}\ldots\left(\frac{u_n}{\delta_n}\right)^{\delta_n}\left(\frac{c_1}{\omega_1}\right)^{\omega_1}\ldots\left(\frac{c_n}{\omega_n}\right)_n^{\omega}\cdot\lambda^{\lambda},$$

where

$\lambda = \omega_1 + \omega_2 + \quad \ldots \quad + \omega_n.$

Duffin, Peterson, and Zener (1967) showed that λ, the sum of the weights in the constraint, is the Lagrange multiplier for that constraint.

2.6 Optimality Condition

Developing the normality conditions requires that the program be stated in a standard form. The objective function must be a minimization function, and the constraints must be arranged so that all of the variable terms are on the left-hand side of an inequality less than or equal to 1. This may require a good deal of algebraic manipulation. The process for getting the program in this format is described in Step 1 of the preprocessing algorithm.

The normality condition is the first type of optimality condition which form the linear constraints on the dual function. This is the same condition referred to in the discussion of the geometric inequality.

$$\delta_1 + \delta_2 + \quad \ldots \quad + \delta_n \quad = \quad 1$$

It is easiest to think of each δ as the percentage weight an objective function term represents in the optimum solution of this program.

Orthogonality conditions are the second type of optimality condition forming the dual constraint functions. The orthogonality conditions are formed as follows. Number the terms in the objective function 1 through n where n is the number of terms in the objective function. Starting with n+1, number the terms in the constraints from the first term in the first constraint to the last term in the last constraint, omitting the right hand sides of the constraints. Then, for each variable, form the orthogonality condition by determining the coefficient on δ from the product of the sign on the term and the exponent of the variable in the term. For each variable the orthogonality condition could contain n+m terms, where n is the number of objective function terms and m is the number of terms in the constraints excluding the right hand sides.

Example:

$$\textit{Minimize}: \quad x_1^2 + x_2^2 - x_1 x_2$$

$$\textit{subject to} \quad x_1^{-3} + x_2^{-1} \quad \leq \quad 1$$

Normality $\delta_1 + \delta_2 - \delta_3 = \sigma$

Orthogonality on x_1 $\quad 2\delta_1 - \delta_3 - 3\delta_4 = 0$

Orthogonality on x_2 $\quad 2\delta_2 - \delta_3 - \delta_5 = 0$

Solving the geometric programming dual function involves the solu-

tion of these the system of equations produced by these optimality conditions. In problems which have zero degree of difficulty these equations will from a square linear matrix which is easily inverted. Further development of these procedures is beyond the scope of this thesis. An excellent review of the method for posynomials is presented by Peterson (1976) and for signomials by Duffin and Peterson (1973).

2.7 Optimality Relationships in the Objective Function

Optimality relationships for the objective function were developed in Woolsey and Swanson (1975) and are included in Woolsey's Four Rules for Geometric Programming used in his course in geometric programming. This relationship follows from the definition of the arithmetic mean and is derived as follows:

$$u_1 + u_2 + \ \ldots \ + u_n = TC^*$$

where $u_i = \delta_i U_i$ and TC^* is the weighted average of the U_i and the value of the objective function at optimality.

Therefore $u_i = \delta_i \cdot TC^*$ or $\frac{u_i}{TC^*} = \delta_i$.

Dividing the first equation by TC^*

$$\frac{u_1}{TC^*} + \frac{u_2}{TC^*} + \ \ldots \ + \frac{u_n}{TC^*} = 1,$$

and substituting $\frac{u_i}{TC^*} = \delta_i$,

$$\delta_1 + \delta_2 + \ldots + \delta_n = 1$$

which is the normality condition.

In general

$$TC^* = \frac{u_1}{\delta_1} = \frac{u_2}{\delta_2} = \quad \ldots \quad = \frac{u_n}{\delta_n}.$$

This is called Rule 3 by Woolsey.

2.8 Optimality Relationships in the Constraints

Optimality relationships for the constraints were also developed in Woolsey and Swanson (1975) and are included in Woolsey's Four Rules for Geometric Programming used in his course in geometric programming. These relationships follow from the definition of the weighted dual constraint function and is derived as follows:

$$1^{\lambda} \geq (u_1 + u_2 + \ldots + u_n)^{\lambda} \geq \left(\frac{u_1}{\omega_1}\right)_1^{\omega} \left(\frac{u_2}{\omega_2}\right)_2^{\omega} \ldots \left(\frac{u_n}{\omega_n}\right)_n^{\omega} \cdot \lambda^{\lambda}.$$

Removing the power and assuming the constraint is tight at optimality

$$1 = u_1 + u_2 + \ldots + u_n = \left(\frac{u_1}{\omega_1}\right)^{\frac{\omega_1}{\lambda}} \left(\frac{u_2}{\omega_2}\right)^{\frac{\omega_2}{\lambda}} \ldots \left(\frac{u_n}{\omega_n}\right)^{\frac{\omega_n}{\lambda}} \cdot \lambda.$$

Rewriting the relationship ignoring the center equality. The equation becomes

$$1 = \left(\frac{u_1}{\omega_1}\right)^{\omega_1} \left(\frac{u_2}{\omega_2}\right)^{\omega_2} \ldots \left(\frac{u_n}{\omega_n}\right)^{\omega_n} \cdot \lambda,$$

or

$$1 = \left(\frac{\lambda u_1}{\omega_1}\right)^{\omega_1} \left(\frac{\lambda u_2}{\omega_2}\right)^{\omega_2} \ldots \left(\frac{\lambda u_n}{\omega_n}\right)^{\omega_n}$$

The general solution to this equation is

$$\left(\frac{\lambda u_i}{\omega_i}\right) = 1,$$

or

$$\lambda u_i = \omega_i$$

and finally,

$$\omega_i = u_i\left(\sum_{i=j} \omega_i\right) \quad \forall j \in \textit{constraint } k.$$

Woolsey calls this Rule 4.

2.9 Classification of General Nonlinear Optimization Problems

The algorithm uses the basic system for characterization of optimization programs described by Gill, Murray, and Wright (1981) to classify residual programs. This system is based on a programming standard used in most linear and nonlinear programming codes.

Linear programming is a separate and distinct methodology and is detailed here because one of the algorithm's outputs can be residual linear programs. Linear programs are problems which have linear functions in the objective function and constraints (i.e., single variable terms with positive exponents equal to 1).

Example:

$$\textit{Maximize}: \quad x_1 + x_2 - x_3$$

$$\textit{subject to} \quad x_1 - x_2 = 5$$

$$x_3 - x_2 \geq 3$$

$$x_i \geq 0 \quad \forall i$$

Subclasses of linear programs:

LEP - Linear objective function and linear equality constraints

LIP - Linear objective function and linear inequality constraints

LCP - Linear objective function and both linear equality and inequality constraints

Nonlinear programs are problems with linear objective functions and nonlinear constraint functions, nonlinear objective functions and linear constraint functions, or nonlinear objective functions and nonlinear constraint functions.

Example:

$$\textit{Minimize Total Cost} \quad = \quad a_1q_1 + b_1q_1^{-1} + a_2q_2 + b_2q_2^{-1} + ... + a_nq_n + b_nq_n^{-1}$$

$$\textit{subject to} \quad k_1q_1^{-1} + k_2q_2^{-1} + ... + k_nq_n^{-1} \quad \leq \quad 1$$

Subclasses of nonlinear programs:

NLEP - Nonlinear objective function and linear equality constraints

NLIP - Nonlinear objective function and linear inequality constraints

NLCP - Nonlinear objective function and both linear equality and inequality constraints

NEP - Nonlinear objective function or linear objective function and with nonlinear equality constraints

NIP - Nonlinear objective function or linear objective function and nonlinear inequality constraints

NCP - Nonlinear objective function or linear objective function and both nonlinear equality and inequality constraints

2.10 Regression Analysis in Nonlinear Programming

One method of developing the objective function is a regression analysis of the independent variables important to optimization. Selecting the variables to be used in a regression function is an important first step in achieving reasonable and usable results. These variables must not only provide sufficient accuracy for the prediction, but also be manageable. This is critical to the user because it accomplishes nothing to optimize a function consisting of uncontrollable variables. For example, the sale of winter coats goes up in winters which have one or more blizzards since the number of blizzards in a year. Since the number of blizzards is outside the control of the coat manufacturer increasing sales cannot be accomplished by increasing the number of blizzards. Another trap some analysts fall into is that while some variables provide excellent statistical results they do not fit the generally accepted operational theory of the industry. A regression which has these types of variables will very difficult to use unless the analyst is prepared to defend the model in detail. Above all the model must be sensible.

Selecting the type of regression must also fit the generally accepted operational theory. In structural mechanics the modulus of elasticity of steel and aluminum is determined by linear regression while the modulus of elasticity for concrete, rock, and soils can be determined by nonlinear regression. Though in some regions of the stress strain relationships a nonlinear modulus might predict the behavior of steel or aluminum better only an academic researcher would be able to justify

using that relationship. An engineer in practice would be in violation of several building and safety codes if he attempted such a substitution.

In business selecting the regression type is usually less restricted. However, as we see in the example described in Appendix B some variables which are good regression estimators can cause some operational problems if used indiscriminately. In business, applications of nonlinear regression techniques, especially the logarithmic regression analysis which lead to the complex nonlinear programs for which this algorithm is designed, the user should take the time to insure that complexity of the model is justified by better results. Many engineering and economic applications generate logarithmic regressions which fit well with the strength of geometric programming techniques in dealing with real exponents.

Chapter 3

PREPROCESSING ALGORITHM

3.1 About the Algorithm

This algorithm is a step-by-step procedure for reducing large constrained nonlinear optimization programs. Its premise is that the solution to these programs is a boundary point and if there are insufficient constraints in the program, then sufficient constraints can be found either by analysis or by evaluation of the operational environment. To some extent the user will be concerned about signomial (negative coefficients) problems because of their inherent nonconvexity. No method developed to date will converge consistently to a global optimum in nonconvex problems (multiple local optimum). In using the algorithm the signomial problem will generally be unchanged; in some cases where the negative terms are separable it will generate a convex subproblem, if that happens then the program can be solved globally. Regardless of the form of the program, the algorithm allows the user to classify the problem and continue with other methods. Optimization seldom offers a closed form solution to all problems. Applying this algorithm successfully requires an understanding of the structure of the model, the importance of the variables, and acceptable outcomes. Evaluating mathematical formulations requires a qualitative understanding of the problem in its environmental context.

3.2 Make a Worksheet

Making a worksheet is important since there are many substitutions which will reoccur every time the problem is resolved. It also will assist in back tracking if a technique complicates the problem, in finding and correcting mistakes, and identifying spreadsheet calculations to automate future changes in dependent variables. When a value is assigned to a variable, make the substitution everywhere it appears and keep track of the changes in the objective function and the constraints by restating them. This will also identify errors quickly. An example worksheet is provided in the Appendix A.

The algorithm is written in anticipation of every type of problem the author could imagine. If the step does not seem to apply then, skip it; likewise, as the need for an interim or new step is found, add it to the worksheet. The opportunity to loop or repeat selected steps is given at selected points in the algorithm. The order of operations has a good deal of utility and to a large extent each operation leads to the next, but it is not immutable and can be changed to fit the application. Quit when the problem is solved or classified.

As the process becomes familiar take the opportunity to reformulate the model when a problem is encountered. Keep refining the model until it is representative and consistent with the process being modeled. This is where the creative intuitive problem solving will begin to assert itself.

3.3 The Algorithm

Step 1 Put the Program into Standard Format

List the objective function.

List the constraints in the following order with the variable terms on the left hand side of the equations and the constants on the right.

- Equality constraints.
- Inequality constraints.
  - Less than or equal to constraints.
  - Greater than or equal to constraints.

List the single variable bounds.

Explanation of Step 1

Putting the problem into a standard form is necessary not only for this algorithm but for most computer programs. List the objective function as minimize or maximize a quantity (e.g., sales, cost, or profits) equal to a generalized polynomial (signomial). This algorithm allows the evaluation of only one objective function at a time. Multiple objective functions require the use of dynamic programming concepts not included in this algorithm.

Listing the constraints in this order allows the user of the algorithm and later the computer program to check the most restrictive constraints first before checking the less restrictive constraints. Write the constraints with the variable terms on the left-hand side of the equations

and the constants on the right-hand side.

Finally, single variable bounds, when they are available, provide important information and reduce the difficulty of the problem by defining the region (vector space) where the solution (vector) must lie. Single variable bounds are not effective constraints in and of themselves and hence are singled out of the constraints during the algorithm's analysis. Most computer programs have utilities for using bounds and will generate additional bounds as part of the solution process.

Step 2 Simplify the Problem Statement

Complete algebraic operations and collect terms.

Make linear substitutions for variables with unique exponents and unique variable combinations.

Make substitutions for messy things in the program and define these substitutions with an equality constraint.

Sum all constant terms in the objective function. Store this value and delete this term from the objective function.

Explanation of Step 2

Complete all multiplication operations, including expansion of summations with integer exponents, to expand the objective function and constraints to their simplest algebraic form (e.g., $x(1 - x)$ is $x - x^2$ and $(1 - x)^2$ is $(1 - 2x + x^2)$ and solve for the roots of single variable bounds. Reverse the exponent sign of term denominators. Division may lead to a different type of problem called messy things. If $(1 - x)^{-2}$ appears in the

function then its expanded form is $(1 - 2x + x^2)^{-1}$ which cannot be simplified further. Define a new variable $y = 1 - 2x + x^2$ in the constraints. This will increase the program's degree of difficulty, but simplify the analysis.

Collect terms by identifying terms with identical variable exponents but not necessarily identical coefficients and adding them. Programmers will have to think of all variables as having an exponent in a term, some of which are zero, in order to make these 1 to 1 comparisons. (Reminder: any variable to a zero power is equal to 1.)

Make linear substitutions for variables with unique exponents and unique variable combinations. If a variable has the same real exponent every time it appears in the objective function and constraints but not in the single variable bounds, then a change of variable will simplify the problem. Experience has shown that some highly nonlinear programs will reduce to linear programs in this manner. For example, suppose x_2^3 appears in two program terms, once in the objective function and once in a constraint, and the variable x_2 appears nowhere else in the program with a nonzero exponent. Let $z_2 = x_2^3$. Do not make this substitution an equality constraint. After determining the final solution calculate $x_2 = \sqrt[3]{z_2}$. The cube root of z_2 has positive, negative, and complex roots, remembering the assumption that the variables are strictly positive choose the positive root.

Replace uniquely appearing variable combinations with linear variables. Suppose that x_2 appears multiplied by x_4; and the variable x_4 only appears when multiplied by x_2. Then make a change of variable $z_{24} =$

x_2x_4. Do not make this substitution an equality constraint. After determining the final solution calculate $x_2x_4 = z_{24}$. This does not uniquely define x_2 or x_4 but an arbitrary assignment based on the variable bounds or the physical process can be made without affecting the optimality of the solution.

Tying two or more variables together for optimization will cause some conceptual problems initially. This is not uncommon but is easy to demonstrate, that while the variables may be physically identifiable they are linked in the problem. A good example is the economic concept of substitution. Substitution occurs because, while the consumption of good A is preferred, the substitution of good B in sufficient quantities will satisfy the demand. Therefore, the variables for good A and good B may become linked in the problem because the demand for both goods is the same.

Sum all constant terms in the objective function and store the value for later use. Constants contribute to the final solution but cannot be optimized.

Step 3 Check Assumptions

Are all variables strictly non-negative (*i.e.*, *all* $x_i \geq 0$)?

Are there any strict inequality constraints?

Is the problem posed as a signomial optimization program?

Explanation of Step 3

Are all variables strictly non-negative? Before forcing this assump-

tion look at each variable carefully. Does it make sense in the engineering, economic, or business context of the problem? Are there bounds on the variables which support this assumption? If some of the variables are strictly non-positive (*i.e.*, $x_i \leq 0$), will a simple change of the coefficients associated with those variables solve the problem?. If it can have both positive and negative values, then do not make any substitution and quit. The algorithm will not solve this problem because of the underlying assumption of non-negativity.

Are there any strict inequality constraints? Strict inequalities imply that the function is undefined at some bound and therefore replacing it with an inequality without changing the limit may not give the desired result. If, however, the inequality was assigned without considering whether it could be equal to the limit, reexamine the constraint.

Is the problem posed as a signomial? If not, can it be reduced to a signomial form simply and without changing the nature of the problem? One way of doing this is to replace the non-signomial terms with series expansions (see any standard math tables for Taylor, binomial, logarithmic, and trigonometric series expansions).

Examples:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!}$$

$$\ln x = \frac{x-1}{x} + \frac{1}{2}\left(\frac{x-1}{x}\right)^2$$

There are some cases where these series expansion substitutions

are unnecessary and counter productive (*e.g., Minimize:* e^{x+y} *subject to* $x + 2y \geq 5$ is equivalent to *Minimize:* $x + y$*: subject to* $x + 2y \geq 5$). Make the equivalent program substitution when possible. Limit series expansion substitutions to the first two or three terms of the series. There are instances when more are appropriate but not the first iteration and probably not until the program has been run on a nonlinear optimization computer code. Be especially careful with trigonometric functions because of their periodicity. Make sure that the variable's solution region is well defined before substituting.

Step 4 Remove Redundant Constraints

Look for constraints that are simple multiples of each other.

Look for groups of constraints which added and subtracted from each other equal another constraint or its multiple.

Set up a modified coefficient matrix and do Gauss-Jordan elimination to verify independence for large problems.

Explanation of Step 4

The first two steps provide an easy method for identifying obviously redundant constraints in small problems. Large problems require a more structured approach such as the Gauss-Jordan method. There are several software packages (e.g., MATLAB™ and MATHCAD™) which have this routine. To use this you must create a modified coefficient matrix as if each variable combination were a different variable. (*i.e.,* $4x_1^2x_2 + 2x_1^3x_3^2 + 5x_1 + 3x_2^2 \leq 0$ would have the row [5,4,2,3] and the column

$[x_1, x_1^2 x_2, x_1^3 x_3^2, x_2^2]^T$. Most operations research and linear algebra texts describe this method. A good reference is Winston (1987, 30). Because of the nonlinearity of the set of equations this step may not identify all of the redundant constraints. The next step will identify additional redundancies indirectly.

Step 5 Check for Solutions in Constraints

How many variables are there? How many equality constraints? How many inequality constraints?

Is the number of independent equality constraints equal to the number of variables?

Is there a subset of independent equality constraints which have equal numbers of equations and variables?

Use either a linear or nonlinear simultaneous equation solver to get an unique solution to the problem. Then check inequality constraints and single variable bounds.

Is the number of independent equality and inequality constraints equal to the number of variables?

Is there a subset of independent equality and inequality constraints which have equal numbers of equations and variables?

Assume all inequalities constraints are equalities. Use either a linear or nonlinear simultaneous equation solver to get a "feasible problem" solution. Then check the single variable bounds. Save this solution for use as a starting point in the algorithm to solve the problem.

Explanation of Step 5

If the number of independent equality constraints is equal to the number of variables then there is an unique solution to the constraint

set, provided there are no hidden redundancies. Use a nonlinear simultaneous equation solver to get this unique solution which is "optimum." Substitute this solution into the remaining inequality constraints and single variable bounds. If the solution does not satisfy these constraints, then there is an inequality constraint that is tighter than one or more of the equality constraints and the formulation is incorrect.

If the number of independent equality constraints and inequality constraints equals the number of variables, then assume all inequalities are equalities. Use a nonlinear simultaneous equation solver to get a "feasible" solution to the problem. This solution will be useful later as a starting point, if the solution algorithm is iterative. Many of the solution methods implemented in computer programs are iterative and a good feasible starting point will greatly improve the speed of these methods.

If the solution from the nonlinear equation solver is nonsensical or the program fails to converge, then there is probably a redundant set of equations still in the program. Many of the nonlinear programming codes using derivative approaches will solve this problem even with the redundancies.

Step 6 Separate Variables

Identify and list unconstrained objective function variables.

Identify and list constrained variables not in the objective function which appear in constraints which do no contain either objective function variables nor variables that are in constraints with objective function terms.

Identify and list variables which appear in both objective function and constraint terms or in constraints containing variables that link to objective function variables in other constraints.

Look for subsets of the objective function terms containing variables which are either unconstrained or constrained by unique subsets of the constraints. Restate these separated programs as independent subprograms and return to Step 1 to continue the analysis on each of them separately.

Explanation of Step 6

Examples:

Unconstrained objective function variables:

$$\textit{Minimize}: \quad x_1^2 x_2^{-1} + x_3^{1.3} x_4^{-3}$$

$$\textit{subject to}: \quad x_1^{-2} + x_3^{-1} \leq 1$$

x_3 and x_4 are unconstrained and since they both have negative signs$_{GP}$ the optimal solution will allow them to approach infinity unless they are bounded.

Non-objective function variables not linked to objective function variables.

$$Minimize: \quad x_1^2 x_2^{-1} + x_3^{1.3} x_4^{-3}$$

$$subject \quad to \quad x_1^{-2} + x_3^{-1} x_5 \leq 1$$

$$\text{and } x_5^{-2} - x_6^4 \leq 1$$

$$\text{and } x_7^2 \leq 1$$

x_7 fits this criteria while x_5 and x_6 are linked to the objective function because x_5 appears in a constraint with x_1 and x_2 both of which are objective function variables. The optimal solution is insensitive to the value assigned to x_7. Therefore make any assignment which does not violate the constraint.

Separable subsets of objective function terms.

$$Minimize: \quad x_1^2 x_2^{-1} + x_3^{1.3} x_4^{-3}$$

$$subject \quad to: \quad x_1^{-2} + x_2^{-1} \leq 1$$

x_1 and x_2 are constrained by an equation which does not include either of the other variables and therefore can optimized independently of the other term without affecting the optimal solution. x_3 and x_4 form an unconstrained separable subset of the problem. A companion thesis by Wessels (Forthcoming) develops other separable subprograms and looks at these relationships in more detail using the geometric programming $sign_{GP}$ table.

Step 7 Look for this Problem Type

The objective function is a single term and the left-hand side of the constraints are single terms (constants on the right-hand side).

If so, then take the logarithm of the objective function and both sides of the constraints and single variable bounds to form a linear program.

Explanation of Step 7

Example:

$$\textit{Minimize}: \quad x_1^{-2}x_2^{3}x_3^{-2}$$

$$\textit{subject to} \quad x_1x_2^{-2} \le 1$$

and $x_3 \le 1$

can be reduced to

$$\textit{Minimize}: \quad -2\ln x_1 + 3\ln x_2 - 2\ln x_3$$

$$\textit{subject to} \quad \ln x_1 - 2\ln x_2 \le \ln 1$$

and $\ln x_3 \le \ln 1.$

Let z = ln x and only a linear program remains.

Step 8 Remove Redundant Bounds

Remove redundant bounds. Redundant bounds are those that are less restrictive than similar bounds on the same variable.

Compare bounds to constraints. Are there any bounds that would violate the constraints? If so, then the bound is redundant.

Check variable bounds for infeasibility or unboundedness.

Explanation of Step 8

Compare the single variable bounds and remove the redundant ones. Redundant bounds are less restrictive than similar bounds on the same variable (*e.g., $2 \le x \le 8$ and $5 \le x \le 6$;* satisfying the latter also satisfies the former). Check variable bounds for infeasibility or unboundedness (*e.g., $1 \le x \le 2$ and $3 \le x \le 4$,* taken together are infeasible, while *maximize: x, subject to* $x \ge 1$ is unbounded). Compare bounds to constraints and remove any bounds that would violate the constraints. In the next step these bounds will be replaced.

Step 9 Bound Variables in the Constraints

Lower bound variables in the equality and greater than or equal to constraints by assuming all other variables at their upper bounds.

Upper bound variables in the equality and less than or equal to constraints by assuming all other variables at their lower bounds.

List the tightest set of upper and lower bounds you can for each variable.

Look for unique solutions to the constraints in the bounds.

Note: Do not replace constraints with single variable bounds.

Explanation of Step 9

Examples:

Lower bounding

$$x_1, x_2 \geq 0.1$$

$$x_1 + x_2 \leq 0.35$$

Holding x_1 at its lower bound ($x_1 \geq 0.1$) *then* x_2 must be ≥ 0.15 and vice versa.

Upper bounding

$$x_3, x_4 \geq 0.1$$

$$x_3 + x_4 \leq 0.35$$

Holding x_3 at its lower bound then x_4 must be ≤ 0.25 and vice versa.

Bounding variables in the constraints is an excellent way to get a

feel for the operating ranges. These bounds will often span ranges which are infeasible so do not confuse them with the operating bounds given in the problem unless they are tighter bounds. The bounding methods described can be quite tedious if the problem is very large. Some nonlinear computer programs include a bounding routine, which cab be used separately to supplement the analysis.

Look for unique solutions to the constraints in the bounds (*i.e., If* $0 \le x_1 \le 0.2$, $0 \le x_2 \le 0.2$, *and* $x_1 + x_2 \ge 0.4$ *then* $x_1 = x_2 = 0.2$ is a unique solution).

Step 10 Row Zero Test

If all the coefficients of a constraint are positive and less than or equal to zero (*e.g.*, $x_1^2 + x_2^3 + x_3 + x_4^2 \le 0$), then set all variables in that constraint equal to zero (trivial result based on our initial assumption that all variables are strictly non-negative).

If all the coefficients of a constraint are negative and greater than or equal to zero (*e.g.*, $- x_1^2 - x_2^3 - x_3 - x_4^2 \ge 0$), then set all variables in that constraint equal to zero.

Step 11 Column $Sign_{GP}$ Test

Minimization problems:

If the variable always appears with a positive $sign_{GP}$ in the objective function and all less than or equal to constraints and negative $sign_{GP}$ in all greater than or equal to constraints fix it at its lower bound.

If the variable always appears in the objective function and all less than or equal to constraints with a negative $sign_{GP}$ and positive $sign_{GP}$ in all greater than or equal to constraints fix it at its upper bound.

Maximization problems:

If the variable always appears with a positive $sign_{GP}$ in the objective function and all greater than or equal to constraints and negative $sign_{GP}$ in all less than or equal to constraints fix it at its lower bound.

If the variable always appears in the objective function and all greater than or equal to constraints with a negative $sign_{GP}$ and positive $sign_{GP}$ in all less than or equal to constraints fix it at its upper bound.

Check constraints.

Explanation of Step 11

Minimization problems:

If the variable always appears in the objective function with a positive coefficient and a positive exponent or a negative coefficient and a negative exponent (i.e., positive $sign_{GP}$) and positive $sign_{GP}$'s in all less than or equal to constraints and negative $sign_{GP}$'s in all greater than or equal to constraints, then fix it at its lower bound in the single variable constraints developed in previous steps or given in the problem statement. In some cases it may be necessary to search for a lower bound in the process itself, but it sill be easier knowing that it is important.

If the variable always appears in the objective function with a negative coefficient and a positive exponent or a positive coefficient and a negative exponent (i.e., negative $sign_{GP}$) and negative $sign_{GP}$'s in all less than or equal to constraints and positive $sign_{GP}$'s in all greater than or equal to constraints, then fix it at its upper bound.

Maximization problems:

If the variable always appears in the objective function with a positive coefficient and a positive exponent or a negative coefficient and a negative exponent (i.e., positive $sign_{GP}$) and negative $sign_{GP}$'s in all less than or equal to constraints and positive $sign_{GP}$'s in all greater than or equal to constraints, then fix it at its upper bound.

If the variable always appears in the objective function with a negative coefficient and a positive exponent or a positive coefficient and a neg-

ative exponent (i.e., negative $sign_{GP}$) and positive $sign_{GP}$'s in all less than or equal to constraints and negative $sign_{GP}$'s in all greater than or equal to constraints, then fix it at its lower bound.

Check constraints:

If all of the constraints are still feasible continue. If not, isolate the variables assigned to the violated constraints in this step and return the program to it original state before these assignments were made. This should not occur and may indicate an error in the assignment process.

Step 12 Restate the Reduced Problem

Check variable assignments. Sum constant terms in the objective function and on the left-hand side of remaining constraints.

Restate the program in the format given in Step 1.

Is the problem still there?

Repeat previous steps as necessary.

Were any additional problems generated?

Explanation of Step 12

Carefully check each of the variable assignments made in the algorithm so far. Make sure that the variable values determined were substituted into the objective function and constraints. Sum constant terms in the objective function and sum constants on left-hand side of remaining constraints.

Restate the program in the format given in Step 1. This is important for exiting the algorithm or reentering it for further analysis.

Is the problem still there? If not, then calculate the objective function value, add in the stored constant terms, list the variable assignments and check the constraints.

Were any changes made during this iteration? If yes, then go back to Step 2. Unless there are no more changes to be made.

Were any additional problems generated? If yes, then go back to Step 1 and process the other problems.

Step 13 Normalize the Constraints

Normalize the constraints (but not the single variable bounds) by dividing through by the nonzero right hand side.

Step 14 Determine the Degree of Difficulty

How many terms in the problem? How many variables?

DD = Variables - Terms - 1

Step 15 Convert the Program

Convert maximization programs to minimization programs.

Convert the constraints to signomials less than or equal to 1.

Explanation of Step 15

Further processing using geometric programming requires a minimization program. A maximization objective function can be converted to minimization by multiplying it by -1 or inverting it. More difficult conversions are made by creating a new variable and equality constraint.

Save the statement of the maximization program for use later if a nonlinear optimization computer code is used. Geometric programming assumes all constraints to be tight. Equality constraints can, therefore, be written as less than or equal to constraints.

Step 16 Create the Delta$_{GP}$ Matrix

Write the normality constraint as the first row. If the program is a posynomial, then set the row equal to 1. If it is a signomial set the row equal to σ

Write the remaining rows by summing the deltas associated with each variable equal to zero (orthogonality constraints).

Explanation of Step 16

The delta$_{GP}$ matrix is based on the terms of the problem. Each term is numbered consecutively starting with the objective function and continuing through the constraints omitting the right hand sides of the constraints. (Neglect any remaining single variable bounds.)

The first row (normality constraint) of the delta matrix is the sum of the deltas associated with the terms of the objective function set equal to σ, a variable in the first row. (i.e., $\sigma\delta_1 + \sigma\delta_2 + \bullet\bullet\bullet + \sigma\delta_n = \sigma$; where n = the number of terms in the objective function and $\sigma = \pm 1$). The sign (σ) of the coefficient on each delta is the same as the sign$_{GP}$ of the term.)

Write the remaining rows by summing the deltas associated with each variable equal to zero (orthogonality conditions). The coefficient sign (σ) for the delta is the term's variable sign$_{GP}$ while the coefficient is

the exponent of the term's variable.

Look for subsets of simultaneous linearly independent equations which have the same number of equations as variables (δs). If such a subset exists then solve this subset of equations for the variables and substitute their values into the remaining problem and save them for use in the next step.

Step 17 Use the Optimality Relationships

Write the optimality relationships for the terms in the objective function using Rule 3.

Isolate combinations of terms containing one and two variables.

Write the optimality relationships for the constraints using Rule 4.

Substitute delta values from Step 16 and solve for program variables at optimality.

Explanation of Step 17

Rule 3:

$$TC^* = \frac{term_1}{\delta_1} = \frac{term_2}{\delta_2} = \quad \ldots \quad = \frac{term_n}{\delta_n}$$

Isolate combinations of terms which contain one or two variables. Solve for the single variable directly and substitute its value into the remaining terms of the program. In combinations containing two variables, solve for one variable and substitute the result into the remaining terms of the program. Repeat this step until no more substitutions can

be made.

Rule 4:

$$\omega_i = constraint \quad term_i \quad \left(\sum_{i=j} \omega_i\right) \quad \text{for all j in constraint k}$$

If the degree of difficulty is greater than zero, it is possible that not all constraints will be active in the solution. If a value for a constraint term delta was found in Step 16 then this relationship can be temporarily assumed to be active at optimality for isolating variable relationships. Do not commit these relationships to a solution until all possible variable solutions have been found.

Step 18 Manipulate the Optimality Relationships

Search both the $delta_{GP}$ matrix and optimality relationships for subsets of independent relationships and solve them for the δs and program variables.

Check the values determined in the above in the constraints and bounds for feasibility. Enumerate the remaining objective function values.

Step 19 Classify the Remaining Linear Program

If the residual program is linear classify it into one of the following subclasses.

LEP - Linear objective function and linear equality constraints.

LIP - Linear objective function and linear inequality constraints.

LCP - Linear objective function and both linear equality and inequality constraints.

Step 20 Classify the Remaining Nonlinear Program

If the residual program is nonlinear classify it into one of the following subclasses.

LEP - Nonlinear objective function and linear equality constraints.

NLIP - Nonlinear objective function and linear inequality constraints.

NLCP - Nonlinear objective function and both linear equality and inequality constraints.

NEP - Nonlinear objective function or linear objective function with nonlinear equality constraints.

NIP - Nonlinear objective function or linear objective function and nonlinear inequality constraints.

NCP - Nonlinear objective function or linear objective function and both nonlinear equality and inequality constraints.

3.4 Analyzing the Results

Is it a minimization or a maximization program? If the program was converted for geometric programming analysis, it may be worthwhile to return to Step 15 and pick up the residual program for use in an computer optimization code. The next step is to put it in a code and run it. A discussion of all the drawbacks of nonlinear optimization is not included in this thesis. The reader is referred to Gill, Murray, and Wright (1981) for a comprehensive examination of nonlinear optimization techniques. The user is expected to have a healthy skepticism for the any computer program results. Working through the algorithm should have forced you to evaluate the problem and should have shown the ranges of feasible answers.

Computational experience with engineering and economics problems has shown that most variables are bound by operational requirements which are usually not stated in the proposed formulation. This appears to be due to the mathematicians lack of practical experience and the inability of the user to articulate his real constraint set in mathematical terms. The appendices include illustrative examples of this.

Chapter 4

CONCLUSIONS

4.1 Summary of Accomplishments

This thesis represents three significant accomplishments in the area of nonlinear programming.

It details an algorithm for the analysis of economic, engineering, and business optimization problems which can be used by almost anyone with a basic understanding of mathematical programming. This simplistic lock step approach is intended to be the first step for the engineer, or businessman in developing his awareness of the methods available to solve his problem.

It represents the first steps in developing a rule-based system for analyzing nonlinear optimization programs. The basic decision logic is outlined in the 20 steps of the algorithm and sufficient mathematical detail has been provided for the programmer to begin the task of writing the system. For use in today's computer environment where software programs are notorious for being format-sensitive, much of the algorithm's emphasis is on getting the format right.

Finally, it delineates the boundary between mathematical analysis and systems analysis. At some point every solution has to be implemented by someone. Since there must be an man-machine interface even on an automated assembly line, it is important to know where it is and how it can be used to solve problems.

4.2 Applications of Algorithm

The clearest need for this algorithm is demonstrated by the examples in appendices B and C. Modelers need a method to analyze their models as the they are being developed and before the model is committed to large scale computer testing. The literature, however, is not replete with examples of the application of these techniques to real problems nor is it likely that it will be in the near future. Much of mathematical programming has been developed and published in theoretical journals in a language and notation which is difficult to read at best. The problem sets that nonlinear algorithms are tested against bear no resemblance to real problems, making it difficult for the novice to recognize and integrate.

Separable programming is of little interest to the pure mathematician because it represents only a small portion of mathematical programming. Other disciplines such as engineering, business, and economics are faced with solving problems which by their very nature are separable or the interaction of the variables is sufficiently small in the overall context of the problem that they can be ignored. This is a state of affairs which will end quickly, however, and separable models represent a necessary interim step for these disciplines, making the transition from linear and quadratic models to highly nonlinear interactive models.

The existence of a mathematical tool in no way guarantees its use. For examples one need look no further than the canned program library on the nearest main frame computer. While these are powerful tools for

solving a wide range of problems it is difficult to imagine a less friendly system. The nature of the computer age is such that major advances are being made so quickly that analyst have had to specialize in very small compartments. Getting practitioners and scientist to use the tools available will require that basic algorithms such as this be written to demonstrate the capabilities that they are missing and ease the transition to the more sophisticated programs.

4.3 Suggestions for Further Research

Four areas of research are needed in this area. The first is in the area of evaluating redundancy in nonlinear constraint sets. The methods available are principally derivative and difficult to implement. The identification of additional variable relationships via geometric programming provides additional linear information which may allow more general use of linear independence evaluation methods. The concept not developed in the literature but several simultaneous nonlinear equation solvers use the principles involved, Allen (1980) and Baker (1980).

Wessels (Forthcoming) in a companion thesis is developing the follow-on algorithm for bounding the variables remaining after the preprocessing is completed and using geometric programming to solve multiple degree of difficulty signomials reliably. Since these two efforts were made in parallel no feedback loop was established to take advantage of the interface. Once a variable has been bounded it is probable that the bound could be used in improving the constraint set or in identifying another constraint.

A third area of research is in the area of rule based-programming. This algorithm is particularly unsuited for standard programming applications using programming languages such as FORTRAN or C. This is due to the requirement that the program be able to recognize patterns and follow a decision tree. These requirements are basic and well suited for a rule based application in programming languages such as LISP and PROLOG.

The last area concerns the development of a feedback system for evaluating regression models to facilitate optimization. This may be no more than a set of rules which require the user or the program application to identify separable problems within the regression equation. This will allow the user to analyze the trade-off of prediction against optimization model complexity.

REFERENCES CITED

Allen, P. D., 1980, Unbounded Weights: A New Approach to the Solution of Nonlinear Optimization Problems with Applications to Mineral Economics. Ph.D. diss. Colorado School of Mines, Golden Colorado.

Baker, D. W., 1980, Application of the Geometric Inequality to the Solution of Systems of Nonlinear Equations. Ph.D. diss. Colorado School of Mines, Golden Colorado.

Balachandran, V. and Gensch, D. 1974. Solving the Marketing Mix Problem Using Geometric Programming. Management Science. 21: 160-171.

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

Dembo, R. S. 1978. Current State of the Art of Algorithms and computer Software for Geometric Programming. Journal of Optimization Theory and Applications. 26: 149-183.

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

Duffin, R. J. and Peterson, E. L. 1973. Geometric Programming with Signomials. Journal of Optimization Theory and Application. 11: 3-35.

Ecker, J. G. 1980. Geometric Programming: Methods, Computations, and Applications. SIAM Review. 22: 338-362.

Gill, P. E., Murray, W., and Wright, M. H. 1981. Practical Optimization. Orlando: Academic Press, Inc.

Mancini, L. J. and Piziali, R. L. 1976. Optimal Design of Helical Springs by Geometric Programming. Journal of Engineering Optimization. 2: 83-95.

Plenert, G. J., 1987, A Solution to Smith's General Case Bottleneck Problem Allowing for an Unlimited Number of Scheduled Jobs. Ph.D. diss. Colorado School of Mines, Golden Colorado.

Rijckaert, M. J. and Martens, X. M. 1978. Bibliographical Note on Geometric Programming. Journal of Optimization Theory and Application. 26: 325-337.

Wessels, G. Forthcoming. A Geometric Programming Algorithm for Solving a Class of Nonlinear Signomial Optimization Problems. Ph.D. diss. Colorado School of Mines, Golden Colorado.

Woolsey, R. E. D. and Swanson, H. S. 1969. Operations Research for Immediate Application: A Quick and Dirty Manual. New York: Harper and Row.

Woolsey, R. E. D. 1987. Class Notes - Mineral Economics Course 558 - Geometric Programming. Colorado School of Mines, Golden Colorado.

APPENDIX A

ALGORITHM WORKSHEET

Step 1 Put the Program into Standard Format

List the objective function.

List the constraints in the following order with the variable terms on the left hand side of the equations and the constants on the right.

- Equality constraints.
- Inequality constraints.
  - Less than or equal to constraints.
  - Greater than or equal to constraints.

List the single variable bounds.

Workspace

Objective Function

Equality Constraints

Less Than or Equal to Constraints

Greater Than or Equal to Constraints

Single Variable Bounds

Step 2 Simplify the Problem Statement

Complete algebraic operations and collect terms.

Make linear substitutions for variables with unique exponents and unique variable combinations.

Make substitutions for messy things in the program and define these substitutions with an equality constraint.

Sum all constant terms in the objective function. Store this value and delete this term from the objective function.

Workspace

Multiplication.

Addition and subtraction.

Division.

Substitutions for variables with unique exponents.

Substitutions for messy things.

New equality constraints for messy thing substitutions.

Sum constants to be added to the objective function after preprocessing.

Step 3 Check Assumptions

Are all variables strictly non-negative (*i.e., all* $x_i \geq 0$)?

Are there any strict inequality constraints?

Is the problem posed as a signomial optimization program?

Workspace

List the variables:	Estimate operating ranges:
List strict inequality constraints:	New constraints:
Nonsignomial terms:	Transformed terms:

Step 4 Remove Redundant Constraints

Look for constraints that are simple multiples of each other.

Look for groups of constraints which added and subtracted from each other equal another constraint or its multiple.

Set up a modified coefficient matrix and do Gauss-Jordan elimination to verify independence for large problems.

Workspace

List constraints.	List possible redundancy matches.
Write modified coefficient matrix.	Write modified variable column matrix.

Step 5 Check for Solutions in Constraints

How many variables are there? How many equality constraints? How many inequality constraints?

Is the number of independent equality constraints equal to the number of variables?

Is there a subset of independent equality constraints which have equal numbers of equations and variables?

Use either a linear or nonlinear simultaneous equation solver to get an unique solution to the problem. Then check inequality constraints and single variable bounds.

Is the number of independent equality and inequality constraints equal to the number of variables?

Is there a subset of independent equality and inequality constraints which have equal numbers of equations and variables?

Assume all inequalities constraints are equalities. Use either a linear or nonlinear simultaneous equation solver to get a "feasible problem" solution. Then check the single variable bounds. Save this solution for use as a starting point in the algorithm to solve the problem.

Workspace

The number of variables = _____(a)

The number of equality constraints = _____(b)

(a) - (b) = _____ (= 0 ? Then use nonlinear equation solver)

The solution is:

List variables and their values

The number of inequality constraints = _____(c)

(a) - (b) - (c) = _____(= 0 ? Then use nonlinear equation solver)

The feasible answer is:

List variables and their values

Coefficient matrices of independent subsets of equality constraints

The solution for these variables is:

List the variables and their values

Coefficient matrices of independent subsets of inequality and equality constraints

The feasible answer for these variables is:

List variables and their values

List the single variable bounds and substitute values from above.

Substitute and rewrite the program in standard form. See Step 1.

Step 6 Separate Variables

Identify and list unconstrained objective function variables.

Identify and list non-objective function variables appearing in constraints which contain only other non-objective function variables and do not appear in another constraint with an objective function variable.

Identify and list variables which appear in both objective function and constraint terms or in constraints containing variables that link to objective function variables in other constraints.

Look for subsets of the objective function terms containing variables which are either unconstrained or constrained by unique subsets of the constraints. Restate these separated programs as independent subprograms and return to Step 1 to continue the analysis on each of them separately.

Workspace

List unconstrained objective function variables.

List constrained variables not in the objective function which appear in constraints which do not contain either objective function variables nor variables that are in constraints with objective function terms.

List remaining variables.

List unconstrained subprograms.

List constrained subprograms.

Substitute and rewrite the remaining program in standard form.

Step 7 Look for this Problem Type

The objective function is a single term and the left-hand side of the constraints are single terms (constants on the right-hand side).

If so, then take the logarithm of the objective function and both sides of the constraints and single variable bounds to form a linear program.

Workspace

List the objective function.

List the logarithm of the objective function.

List the constraints.

List the logarithm of the constraints.

If this was a subprogram rewrite the remaining program in standard form.

Step 8 Remove Redundant Bounds

Remove redundant bounds. Redundant bounds are those that are less restrictive than similar bounds on the same variable.

Compare bounds to constraints. Are there any bounds that would violate the constraints? If so, then the bound is redundant.

Check variable bounds for infeasibility or unboundedness.

Workspace

List the single variable bounds for the first variable:

List the single variable bounds for the second variable:

Repeat as necessary.

Substitute the lower bound for all variables into the equality and greater than or equal to constraints they appear in.

Substitute the upper bound for all variables into the equality and less than or equal to constraints they appear in.

Step 9 Bound Variables in the Constraints

Lower bound variables in the equality and greater than or equal to constraints by assuming all other variables at their upper bounds.

Upper bound variables in the equality and less than or equal to constraints by assuming all other variables at their lower bounds.

List the tightest set of upper and lower bounds you can for each variable.

Look for unique solutions to the constraints in the bounds.

Note: Do not replace constraints with single variable bounds.

Workspace

Lower bounding computations.

Upper bounding computations.

List tightest bounds possible.

List unique variable answers.

Substitute and rewrite the program in standard form.

Step 10 Row Zero Test

If all the coefficients of a constraint are positive and less than or equal to zero (*e.g.*, $x_1^2 + x_2^3 + x_3 + x_4^2 \leq 0$), then set all variables in that constraint equal to zero (trivial result based on our initial assumption that all variables are strictly non-negative).

If all the coefficients of a constraint are negative and greater than or equal to zero (*e.g.*, $- x_1^2 - x_2^3 - x_3 - x_4^2 \geq 0$), then set all variables in that constraint equal to zero.

Workspace

List variables and their values.

Substitute and rewrite the program in standard form.

Step 11 Column $Sign_{GP}$ Test

Minimization problems:

If the variable always appears with a positive $sign_{GP}$ in the objective function and all less than or equal to constraints and negative $sign_{GP}$ in all greater than or equal to constraints fix, it at its lower bound.

If the variable always appears in the objective function and all less than or equal to constraints with a negative $sign_{GP}$ and positive $sign_{GP}$ in all greater than or equal to constraints, fix it at its upper bound.

Maximization problems:

If the variable always appears with a positive $sign_{GP}$ in the objective function and all greater than or equal to constraints and negative $sign_{GP}$ in all less than or equal to constraints, fix it at its lower bound.

If the variable always appears in the objective function and all greater than or equal to constraints with a negative $sign_{GP}$ and positive $sign_{GP}$ in all less than or equal to constraints, fix it at its upper bound.

Check constraints.

Workspace

Variable	Term	$Sign_{GP}$	Assignment

Substitute and rewrite the program in standard form.

Step 12 Restate the Reduced Problem

Check variable assignments. Sum constant terms in the objective function and on the left-hand side of remaining constraints.

Restate the program in the format given in Step 1.

Is the problem still there?

Repeat previous steps as necessary.

Were any additional problems generated?

Workspace

List variable assignments.

Objective Function.

Equality Constraints.

Less Than or Equal to Constraints.

Greater Than or Equal to Constraints.

Single Variable Bounds.

Sum constants to be added to the objective function after preprocessing.

Substitute and rewrite the program or the solution.

Step 13 Normalize the Constraints

Normalize the constraints (but not the single variable bounds) by dividing through by the nonzero right hand side.

Workspace

Divide constraints.

List normalized constraints.

Step 14 Determine the Degree of Difficulty

How many terms in the problem? How many variables?

DD = Variables - Terms - 1

Workspace

Step 15 Convert the Program

Convert maximization programs to minimization programs.

Convert the constraints to signomials less than or equal to 1.

Workspace

Objective function.

Write new objective function if required.

List new equality constraint for objective function conversion.

List old constraints.

List new constraints in proper form.

Step 16 Create the $Delta_{GP}$ Matrix

Write the normality constraint as the first row. If the program is a posynomial, then set the row equal to 1. If it is a signomial, set the row equal to σ

Write the remaining rows by summing the deltas associated with each variable equal to zero (orthogonality constraints).

Workspace

	δ_1	δ_2	δ_n	RHS
Normality				
Orthogonality Var 1 Var 2 Var 3				

List subsets of independent equations and solve for values of delta.

List values of delta.

Step 17 Use the Optimality Relationships

Write the optimality relationships for the terms in the objective function using Rule 3.

Isolate combinations of terms containing one and two variables.

Write the optimality relationships for the constraints using Rule 4.

Substitute delta values from Step 16 and solve for program variables at optimality.

Workspace

List Rule 3 relationships.

List relationship pairs for isolating variables.

List Rule 4 relationships.

List relationship combinations for isolating variables.

Step 18 Manipulate the Optimality Relationships

Search both the $delta_{GP}$ matrix and optimality relationships for subsets of independent relationships and solve them for the δs and program variables.

Check the values determined in Step 17 with the constraints and bounds for feasibility and bounds and enumerate the remaining objective function values.

Workspace

Step 19 Classify the Remaining Linear Program

If the residual program is linear, classify it into one of the following subclasses.

LEP - Linear objective function and linear equality constraints.

LIP - Linear objective function and linear inequality constraints.

LCP - Linear objective function and both linear equality and inequality constraints.

Step 20 Classify the Remaining Nonlinear Program

If the residual program is nonlinear, classify it into one of the following subclasses.

LEP - Nonlinear objective function and linear equality constraints.

NLIP - Nonlinear objective function and linear inequality constraints.

NLCP - Nonlinear objective function and both linear equality and inequality constraints.

NEP - Nonlinear objective function or linear objective function with nonlinear equality constraints.

NIP - Nonlinear objective function or linear objective function and nonlinear inequality constraints.

NCP - Nonlinear objective function or linear objective function and both nonlinear equality and inequality constraints.

APPENDIX B

MARKETING MIX PROBLEM

B.1 Description of the Problem

The following account is fictional but it serves as a vehicle for developing the intuitive framework for this problem.

The president of a notional brewing company has asked his vice president for marketing to justify his advertising budget in terms of its return on investment or contribution to sales. The vice president believes this to be the president's polite way of telling him he wants a budget cut. Since budgets are a measure of corporate standing among the vice presidents of the brewing company, he would like to avoid a budget cut if at all possible and he sincerely believes that marketing is the key to making the company more successful in the long run. Since the president of the company was formerly the vice president for marketing, he will not be swayed with the standard soft arguments used in the past. In desperation the VP turns to the business and operations analysis group.

The manager of the group see this as a golden opportunity for a bright young economics graduate of a prominent mid-western institution. He believes this to be a lead pipe cinch for this young analyst to make his name known and thereby increase the prestige of the group. After all, the literature is replete with articles on the "marketing mix" problem. In fact the article by Balachandran and Gensch (1974) is very similar to the

problem posed by the VP of marketing. So, our eager young analyst charges off to "solve the problem." As it turns out his problem is exactly the same as the one posed by Balachandran and Gensch, but that is another story.

The analyst immediately gathers all of the historical data available relating to sales volumes, promotional campaigns, and marketing expenditures and begins to feverishly design regression models to forecast sales. After many long nights he identifies the following variables as having relevance to the forecast and being managed by the company.

	Regression Variables for Forecasting Beer Sales (after Dembo 1976)
x_1	The amount of money to be spent on television advertising in the last period.
x_2	The amount of money to be spent on radio advertising in the last period.
x_3	The amount of money to be spent on television advertising in the current period.
x_4	The amount of money to be spent on radio advertising in the current period.
x_5	The amount of money spent on in store promotion in the current period.
x_6	The relative price of beer.
x_7	The relative price change from the current period.
x_8	Trade allowances in the last period.
x_{15}	Trade allowances in the current period.
x_9	The amount of money spent on the sales force in the last period.
x_{14}	The amount of money spent on the sales force in the current period.
x_{10}	The amount of money spent on distribution.
x_{11}	The relative packaging appeal.
x_{12}	The relative product quality.
x_{13}	The age composition of the beer.

The analyst posed this objective function based on his regression analysis.

$$\textit{Maximize} \quad \textit{Sales} = 16.212 + 3.937x_2^{0.91}x_{11}^{1.31} - 0.00021x_1^{-0.95}x_6^{-0.68}x_8^{-0.84}x_7^{-0.28} - 0.00305x_{13}^{-0.18}x_{12}^{1.76} - 0.0046x_5^{-0.3}x_{14}^{-1.1} - 0.0053x_6^{-0.76}x_{10}^{-1.12}$$

Where Sales is in millions of dollars.

The analyst further discovered that contracts were in place such that the amount of money spent on advertising had been and would continue to be fixed in the short term. These bounds were:

$x_1, x_2, x_3, x_4 \geq 0.1$

$x_1 + x_2 \leq 0.35$

$x_3 + x_4 \leq 0.35$

The vice president for marketing told our young analyst in their initial interview that the company was committed to a marketing strategy based on a thorough analysis of target audiences and marketing studies. This strategy had the effect of imposing three more constraints on the problem.

$x_3 + x_4 + x_5 \leq 0.6$

$x_9 + x_{15} \leq 2.0$

$x_6 + x_2 \leq 0.35$

Finally, the analyst looked at the means and standard deviations of each of the variables based on the data he had collected and assigned bounds for each of them. Since advertising variables were already bounded he did not provide bounds for them.

Promotion Bound	$0.2 \leq x_5 \leq 0.35$
Relative Price Bound	$0.1 \leq x_6 \leq 1.35$
Relative Price Change Bound	$0.6 \leq x_7 \leq 1.3$
Trade Allowance Bounds	$0.6 \leq x_8 \leq 1.3$
	$0.6 \leq x_{15} \leq 1.3$
Salesman Effort Bound	$0.8 \leq x_9 \leq 1.25$
	$0.6 \leq x_{14} \leq 1.3$
Distribution Bound	$0.3 \leq x_{10} \leq 0.8$
Packaging Bound	$0.6 \leq x_{11} \leq 1.8$
Quality Bound	$0.8 \leq x_{12} \leq 1.8$
Age Composition Bound	$0.8 \leq x_{13} \leq 1.3$

Since Balachandran & Gensch (1974) had solved their problem on a generalized geometric programming code, the analyst did the same thing and took the solution ($17 million in sales) to the operations research manager for presentation to management. The manager of operations research was a wily old fox and he saw immediately that the answer our young analyst was proposing was probably right, current sales were approaching $16 million, but that he had not answered the question. The vice president of marketing was only responsible for the radio and

television advertising budget, the in-store promotion budget, the sales force, and the packaging budget, so he sent him back to the drawing board.

In his search for the "right" answer the young analyst found the following algorithm which promised to maximize intuition and minimize computer usage.

STEP 1 Put the Program into Standard Format

List the objective function.

List the constraints in the following order with the variable terms on the left hand side of the equations and the constants on the right.

Equality constraints.

Inequality constraints.

Less than or equal to constraints.

Greater than or equal to constraints.

List the single variable bounds.

Workspace

Objective Function

$$Maximize\,Sales = 16.212 + 3.937x_2^{0.91}x_{11}^{1.31} - 0.00021x_1^{-0.95}x_6^{-0.68}x_8^{-0.84}x_7^{-0.28} -$$

$$0.00305x_{13}^{-0.18}x_{12}^{1.76} - 0.0046x_5^{-0.3}x_{14}^{-1.1} - 0.0053x_6^{-0.76}x_{10}^{-1.12}$$

where sales is in millions of dollars

Equality Constraints - NONE

Less Than or Equal to Constraints

Advertising Constraints	$x_1, x_2, x_3, x_4 \geq 0.1$
	$x_1 + x_2 \leq 0.35$
	$x_3 + x_4 \leq 0.35$
Strategy Constraints	$x_3 + x_4 + x_5 \leq 0.6$
	$x_9 + x_{15} \leq 2.0$
	$x_6 + x_2 \leq 0.35$

Greater Than or Equal to Constraints - NONE

Single Variable Bounds

Advertising Constraints	$0.1 \leq x_1 \leq$
	$0.1 \leq x_2 \leq$
	$0.1 \leq x_3 \leq$
	$0.1 \leq x_4 \leq$
Promotion Bound	$0.2 \leq x_5 \leq 0.35$
Relative Price Bound	$0.1 \leq x_6 \leq 1.35$
Relative Price Change Bounds	$0.6 \leq x_7 \leq 1.3$
Trade Allowance Bounds	$0.6 \leq x_8 \leq 1.3$
	$0.6 \leq x_{15} \leq 1.3$
Salesman Effort Bound	$0.8 \leq x_9 \leq 1.25$
	$0.6 \leq x_{14} \leq 1.3$
Distribution Bound	$0.3 \leq x_{10} \leq 0.8$

Packaging Bound	$0.6 \leq x_{11} \leq 1.8$
Quality Bound	$0.8 \leq x_{12} \leq 1.8$
Age Composition Bound	$0.8 \leq x_{13} \leq 1.3$

STEP 2 Simplify the Problem Statement

Complete algebraic operations and collect terms.

Make linear substitutions for variables with unique exponents and unique variable combinations.

Make substitutions for messy things in the program and define these substitutions with an equality constraint.

Sum all constant terms in the objective function. Store this value and delete this term from the objective function.

Workspace

Multiplication. - NONE

Addition and subtraction. - NONE

Division. - NONE

Substitutions for variables with unique exponents. - NONE

Substitutions for messy things. - NONE

New equality constraints for messy thing substitutions. - NONE

Sum constants to be added to the objective function after preprocessing. 16.212. He now knows how much advertising will improve sales in this period.

STEP 3 Check Assumptions

Are all variables strictly non-negative (*i.e., all* $x_i \geq 0$)?

Are there any strict inequality constraints?

Is the problem posed as a signomial optimization program?

Workspace

List the variables:	Estimate operating ranges:
$\mathbf{x_1}$	$0.1 \leq x_1 \leq$
$\mathbf{x_2}$	$0.1 \leq x_2 \leq$
$\mathbf{x_3}$	$0.1 \leq x_3 \leq$
$\mathbf{x_4}$	$0.1 \leq x_4 \leq$
$\mathbf{x_5}$	$0.2 \leq x_5 \leq 0.35$
$\mathbf{x_6}$	$0.1 \leq x_6 \leq 1.35$
$\mathbf{x_7}$	$0.6 \leq x_7 \leq 1.3$
$\mathbf{x_8}$	$0.6 \leq x_8 \leq 1.3$
$\mathbf{x_9}$	$0.8 \leq x_9 \leq 1.25$
$\mathbf{x_{10}}$	$0.3 \leq x_{10} \leq 0.8$
$\mathbf{x_{11}}$	$0.6 \leq x_{11} \leq 1.8$
$\mathbf{x_{12}}$	$0.8 \leq x_{12} \leq 1.8$
$\mathbf{x_{13}}$	$0.8 \leq x_{13} \leq 1.3$
$\mathbf{x_{14}}$	$0.6 \leq x_{14} \leq 1.3$
$\mathbf{x_{15}}$	$0.6 \leq x_{15} \leq 1.3$

List strict inequality constraints: NONE	New constraints: NONE
Nonsignomial terms: NONE	Transformed terms: NONE

STEP 4 Remove Redundant Constraints

Look for constraints that are simple multiples of each other.

Look for groups of constraints which added and subtracted from each other equal another constraint or its multiple.

Set up a modified coefficient matrix and do Gauss-Jordan elimination to verify independence for large problems.

Workspace

List constraints. NONE	List possible redundancy matches. NONE
Write modified coefficient matrix. NONE	Write modified variable column matrix. NONE

STEP 5 Check for Solutions in Constraints

How many variables are there? How many equality constraints? How many inequality constraints?

Is the number of independent equality constraints equal to the number of variables?

Is there a subset of independent equality constraints which have equal numbers of equations and variables?

Use either a linear or nonlinear simultaneous equation solver to get an unique solution to the problem. Then check inequality constraints and single variable bounds.

Is the number of independent equality and inequality constraints equal to the number of variables?

Is there a subset of independent equality and inequality constraints which have equal numbers of equations and variables?

Assume all inequalities constraints are equalities. Use either a linear or nonlinear simultaneous equation solver to get a "feasible problem" solution. Then check the single variable bounds. Save this solution for use as a starting point in the algorithm to solve the problem.

Workspace

The number of variables = ____15____(a)
The number of equality constraints = _____0____(b)

(a) - (b) = _____15_____ (= 0 ? Then use nonlinear equation solver)
The solution is:
List variables and their values

NONE

The number of inequality constraints = _____5_____(c)
(a) - (b) - (c) = _____10_____(= 0 ? Then use nonlinear equation solver)
The feasible answer is:
List variables and their values

NONE

Coefficient matrices of independent subsets of equality constraints

NONE

The solution for these variables is:
List the variables and their values

NONE

Coefficient matrices of independent subsets of inequality and equality constraints

NONE

The feasible answer for these variables is:
List variables and their values

NONE

List the single variable bounds and substitute values into the above.

NONE

Substitute and rewrite the program in standard form.

STEP 6 Separate Variables

Identify and list unconstrained objective function variables.

Identify and list non-objective function variables appearing in constraints which contain only other non-objective function variables and do not appear in another constraint with an objective function variable.

Identify and list variables which appear in both objective function and constraint terms or in constraints containing variables that link to objective function variables in other constraints.

Look for subsets of the objective function terms containing variables which are either unconstrained or constrained by unique subsets of the constraints. Restate these separated programs as independent subprograms and return to Step 1 to continue the analysis on each of them separately.

Workspace

List unconstrained objective function variables.

Variables that only appear in objective function terms and only single variable bounds. x_7, x_8, x_{10}, x_{11}, x_{12}, x_{13}, and x_{14}

List constrained variables not in the objective function which appear in constraints which do not contain either objective function variables nor variables that are in constraints with objective function terms.

Variables that appear only in constraints which do not contain

objective function variables, x_9, and x_{15}. These variables are not a part of the problem and may or may not be assigned depending on what the analyst wants to do with them.

List remaining variables.

Variables which appear in both objective function and constraint terms or in the constraints that include objective function variables. x_1, x_2, x_3, x_4, x_5, and x_6.

List unconstrained subprograms.

Maximize

$$-0.00305x_{13}^{-0.18}x_{12}^{1.76}$$

$$0.8 \le x_{12} \le 1.8$$

$$0.8 \le x_{13} \le 1.3$$

List constrained subprograms.

Maximize

$$+3.937x_2^{0.91}x_{11}^{1.31} - 0.00021x_1^{-0.95}x_6^{-0.68}x_8^{-0.84}x_7^{-0.28}$$

$$-0.0046x_5^{-0.3}x_{14}^{-1.1} - 0.0053x_6^{-0.76}x_{10}^{-1.12}$$

subject to

$$x_1 + x_2 \le 0.35$$

$$x_3 + x_4 \le 0.35$$

$$x_3 + x_4 + x_5 \le 0.6$$

$$x_6 + x_2 \le 0.35$$

$0.1 \leq x_1 \leq$

$0.1 \leq x_2 \leq$

$0.1 \leq x_3 \leq$

$0.1 \leq x_4 \leq$

$0.2 \leq x_5 \leq 0.35$

$0.1 \leq x_6 \leq 1.35$

$0.6 \leq x_7 \leq 1.3$

$0.6 \leq x_8 \leq 1.3$

$0.3 \leq x_{10} \leq 0.8$

$0.6 \leq x_{11} \leq 1.8$

$0.6 \leq x_{14} \leq 1.3$

Substitute and rewrite the remaining program in standard form.

STEP 7 Look for this Problem Type

The objective function is a single term and the left-hand side of the constraints are single terms (constants on the right-hand side).

If so, then take the logarithm of the objective function and both sides of the constraints and single variable bounds to form a linear program.

Workspace

List the objective function. - NO MATCH

List the logarithm of the objective function.

List the constraints.

List the logarithm of the constraints.

If this was a subprogram rewrite the remaining program in standard form.

STEP 8 Remove Redundant Bounds

Remove redundant bounds. Redundant bounds are those that are less restrictive than similar bounds on the same variable.

Compare bounds to constraints. Are there any bounds that would violate the constraints? If so, then the bound is redundant.

Check variable bounds for infeasibility or unboundedness.

Workspace

List the single variable bounds for the first variable:

List the single variable bounds for the second variable:

Repeat as necessary. - SEE ABOVE

Substitute the lower bound for all variables into the equality and greater than or equal to constraints they appear in.

Substituting the lower bounds on x_1, x_2, x_3, x_4 x_5, *and* x_6 *into the constraints we see that none are violated.*

$0.2 \leq 0.35$

$0.2 \leq 0.35$

$0.4 \leq 0.6$

$0.2 \leq 0.35$

Substitute the upper bound for all variables into the equality and less than or equal to constraints they appear in. - NONE

STEP 9 Bound Variables in the Constraints

Lower bound variables in the equality and greater than or equal to constraints by assuming all other variables at their upper bounds.

Upper bound variables in the equality and less than or equal to constraints by assuming all other variables at their lower bounds.

List the tightest set of upper and lower bounds you can for each variable.

Look for unique solutions to the constraints in the bounds.

Note: Do not replace constraints with single variable bounds.

Workspace

Lower bounding computations. - NONE

Upper bounding computations.

Since variables 1,2,3,4 are not upper bounded we can use the information in constraints to bound them.

$x_1 + x_2 \leq 0.35$

In this equation we see that if x_1 *is held at its lower bound, then* x_2 *must be* ≤ 0.25 *and vice versa.*

$x_3 + x_4 \leq 0.35$

Similarly, x_3, & $x_4 \leq 0.25$.

$x_3 + x_4 + x_5 \leq 0.6$

Since,

$0.20 \le x_3 + x_4 \le 0.35$

then $0.25 \le x_5 \le 0.4$

and a better bound can be derived from a combination of the two.

$0.25 \le x_5 \le 0.35$

List tightest bounds possible.

$0.1 \le x_1 \le 0.25$

$0.1 \le x_2 \le 0.25$

$0.1 \le x_3 \le 0.25$

$0.1 \le x_4 \le 0.25$

$0.25 \le x_5 \le 0.35$

$0.1 \le x_6 \le 0.25$

List unique variable answers. - NONE

STEP 10 Row Zero Test

If all the coefficients of a constraint are positive and less than or equal to zero (*e.g.*, $x_1^2 + x_2^3 + x_3 + x_4^2 \leq 0$), then set all variables in that constraint equal to zero (trivial result based on our initial assumption that all variables are strictly non-negative).

If all the coefficients of a constraint are negative and greater than or equal to zero (*e.g.*, $-x_1^2 - x_2^3 - x_3 - x_4^2 \geq 0$), then set all variables in that constraint equal to zero.

Workspace

List variables and their values. NO MATCH

Substitute and rewrite the program in standard form.

STEP 11 Column $Sign_{GP}$ Test

Minimization problems:

If the variable always appears with a positive $sign_{GP}$ in the objective function and all less than or equal to constraints and negative $sign_{GP}$ in all greater than or equal to constraints, fix it at its lower bound.

If the variable always appears in the objective function and all less than or equal to constraints with a negative $sign_{GP}$ and positive $sign_{GP}$ in all greater than or equal to constraints, fix it at its upper bound.

Maximization problems:

If the variable always appears with a positive $sign_{GP}$ in the objective function and all greater than or equal to constraints and negative $sign_{GP}$ in all less than or equal to constraints, fix it at its lower bound.

If the variable always appears in the objective function and all greater than or equal to constraints with a negative $sign_{GP}$ and positive $sign_{GP}$ in all less than or equal to constraints, fix it at its upper bound.

Check constraints.

Workspace

Variable	Term	$Sign_{GP}$	Assignment
7	2	+	1.3
8	2	+	1.3
10	5	+	0.8
11	1	+	1.8
12	3	-	0.8
13	3	+	1.3
14	4	+	1.3

$x_3 + x_4 \leq 0.35$

$x_3 + x_4 + x_5 \leq 0.6$

Since,

$0.20 \leq x_3 + x_4 \leq 0.35$

then $0.25 \leq x_5 \leq 0.4$

and a better bound can be derived from a combination of the two.

$0.25 \leq x_5 \leq 0.35$

x_2 and x_3 cannot be assigned directly for much the same reasons that x_9 and x_{15} were not assigned. We can, however, observe that x_5 has a positive $sign_{GP}$ and should be maximized.

Substitute and rewrite the program in standard form.

STEP 12 Restate the Reduced Problem

Check variable assignments. Sum constant terms in the objective function and on the left-hand side of remaining constraints.

Restate the program in the format given in Step 1.

Is the problem still there?

Repeat previous steps as necessary.

Were any additional problems generated?

Workspace

List variable assignments.

$x_3 =$

$x_4 =$

$x_5 = 0.35$

$x_7 = 1.3$

$x_8 = 1.3$

$x_9 =$

$x_{10} = 0.8$

$x_{11} = 1.8$

$0.8 = x_{12}$

$x_{13} = 1.3$

$x_{14} = 1.3$

$x_{15} =$

Objective Function

$$Maximize: \quad y = +8.5x_2^{0.91} - 0.00016x_1^{-0.95}x_6^{-0.68} - 0.0068x_6^{-0.76} - (.005)$$

Equality Constraints

NONE

Less Than or Equal to Constraints

$x_1 + x_2 \leq 0.35$

$x_6 + x_2 \leq 0.35$

Greater Than or Equal to Constraints

NONE

Single Variable Bounds

$0.1 \leq x_1 \leq 0.25$

$0.1 \leq x_2 \leq 0.25$

$0.1 \leq x_6 \leq 0.25$

Sum constants to be added to the objective function after preprocessing.

- 0.005

Substitute and rewrite the program or the solution.

STEP 13 Normalize the Constraints

Normalize the constraints (but not the single variable bounds) by dividing through by the nonzero right hand side.

Workspace

List normalized constraints.

$2.86x_1 + 2.86x_2 \leq 1$

$2.86x_6 + 2.86x_2 \leq 1$

STEP 14 Determine the Degree of Difficulty

How many terms in the problem? How many variables?

DD = Variables - Terms - 1

Workspace

7 TERMS - 3 VARIABLES - 1 = 3

STEP 15 Convert the Program

Convert maximization programs to minimization programs.

Convert the constraints to signomials less than or equal to 1.

Workspace

This is a good place to stop as you can see that the remaining constraints must be tight, and x_1 and x_6 can be solved explicitly for x_2 in the constraints, leaving:

$$Maximize\,Sales \quad = 16.212 + 8.5x_2^{0.91} - 0.00016(.35 - x_2)^{-1.63} - 0.0068(.35 - x_2)^{-0.76} - (.005)$$

At this point the hero of our story realizes that if he sets x_2 at its upper bound and by default sets x_1 and x_6 to their lower bounds, then he can compare it with x_2 set at its lower bound and x_1 and x_6 at their upper bounds.

$0.1 = x_1$	$x_1 = 0.25$
$x_2 = 0.25$	$0.1 = x_2$
$x_5 = 0.35$	$x_5 = 0.35$
$0.1 = x_6$	$x_6 = 0.25$
$x_7 = 1.3$	$x_7 = 1.3$
$x_8 = 1.3$	$x_8 = 1.3$
$x_{10} = 0.8$	$x_{10} = 0.8$
$x_{11} = 1.8$	$x_{11} = 1.8$
$0.8 = x_{12}$	$0.8 = x_{12}$
$x_{13} = 1.3$	$x_{13} = 1.3$
$x_{14} = 1.3$	$x_{14} = 1.3$
Maximum Sales = 18.57	Maximum Sales = 17.23

This last step was suggested by Dembo (1976).

Reminding ourselves of what the variables were

x_1	The amount of money to be spent on television advertising in the last period.
x_2	The amount of money to be spent on radio advertising in the last period.
x_3	The amount of money to be spent on television advertising in the current period.
x_4	The amount of money to be spent on radio advertising in the current period.
x_5	The amount of money spent on in store promotion in the current period.
x_6	The relative price of beer.
x_7	The relative price change from the current period.
x_8	Trade allowances in the last period.
x_{15}	Trade allowances in the current period.
x_9	The amount of money spent on the sales force in the last period.
x_{14}	The amount of money spent on the sales force in the current period.
x_{10}	The amount of money spent on distribution.
x_{11}	The relative packaging appeal.
x_{12}	The relative product quality.
x_{13}	The age composition of the beer.

Having been handed his head once, the not so eager analyst stops to consider what his optimization program is really telling him. Increasing the amount spent on radio advertising in the last period at the expense of television advertising in the last period and a decrease in the relative price will maximize sales. Management should understand that. He will tell them to lower the price and shift money from television to radio.

Further, he remembers that the in-store promotion was increased at the expense of current advertising. But this will affect sales in future periods. Management will not like that. He will have to go back and run the regression without current advertising. But that will not work because current advertising was not in the objective function to start with. Enter the crusty old operations research manager to point out that his constraint should have been

$$x_1 + x_2 + x_5 \leq 0.6$$

instead of

$$x_3 + x_4 + x_5 \leq 0.6$$

Before he sends the analyst back to the drawing table he asks him what else he has learned. The analyst responds that the last period's salesman effort and the current period's trade allowances are not important to the goal of maximizing sales and that management should spend as much as possible on current salesman effort, trade allowances, distribution, packaging, and age composition. He further states that quality should be minimized. After the explosion the analyst allows that

quality is probably not a good regression variable. He will also want to look at leaving out the lagged sales force variable and current trade allowance variable. All in all, the analyst has learned a lot about how his regression analysis should be structured from a very simple algorithm which forced him to look at the variable relationships

At this point the question to be answered is; "Why is this example important for demonstrating the utility of the algorithm?" It did not solve the problem and left four variables unassigned. In fact the purpose of the algorithm was not to solve the problem but to analyze or preprocess the problem before any solution was attempted. In fact it showed that some of the variables chosen for the forecast were inappropriate for management of the sales volume. The analyst can now make some intelligent changes in the regression model, based on this analysis, that he would not have been able to justify otherwise.

APPENDIX C

HELICAL SPRING DESIGN PROBLEM

C.1 Description of the Problem

Mancini and Piziali (1976) designed a model for a helical spring used in a cam-driven system as:

Minimize: $y = (f-m)c^{.86}d^{-2.86}$

Where y is the deflection of the spring, f is the force applied to the spring, m is the spring force, c is the outside diameter of the spring, and d is the inside diameter of the spring. This model was then solved using a generalized geometric programming algorithm with a computer. Subject to these constraints.

Yielding Constraint	$6.05x10^{-5}c^{.86}d^{-2.715} \le 1$
Surging Constraint	$3.55x10^{-2}c^{2}nd^{-1} \le 1$
Buckling Constraint	$mf^{-1} + .125nc^{2}d \le 1$
Spring Force Constraints	$.1m^{-1} \le 1 \quad \& \quad \frac{m}{20} \le 1$
Diameter Ratio Constraints	$.05 \le dc^{-1} \le .25$
Outside Diameter Constraint	$c + d <= 1.5$
Inside Diameter Constraint	$c - d \le 0.75$
Spring Pocket Length Constraint	$1.12nd + 1.6d \le 1$
Minimum Number of Coils Constraint	$3n^{-1} \le 1$

Step 1 Put the Program into Standard Format

List the objective function.

List the constraints in the following order with the variable terms on the left hand side of the equations and the constants on the right.

Equality constraints.

Inequality constraints.

Less than or equal to constraints.

Greater than or equal to constraints.

List the single variable bounds.

Workspace

Objective Function

Minimize: $y = (f-m)c^{.86}d^{-2.86}$

Equality Constraints

NONE

Less Than or Equal to Constraints

$6.05x10^{-5}c^{.86}d^{-2.715} \leq 1$

$3.55x10^{-2}c^{2}nd^{-1} \leq 1$

$mf^{-1} + .125nc^{2}d \leq 1$

$.1m^{-1} \leq 1$

$$\frac{m}{20} \leq 1$$

$$dc^{-1} \leq .25$$

$$c + d \leq 1.5$$

$$c - d \leq 0.75$$

$$1.12nd + 1.6d \leq 1$$

$$3n^{-1} \leq 1$$

Greater Than or Equal to Constraints

$$.05 \leq dc^{-1}$$

Single Variable Bounds

NONE

Step 2 Simplify the Problem Statement

Complete algebraic operations and collect terms.

Make linear substitutions for variables with unique exponents and unique variable combinations.

Make substitutions for messy things in the program and define these substitutions with an equality constraint.

Sum all constant terms in the objective function. Store this value and delete this term from the objective function.

Workspace

Multiplication

Minimize: $y = fc^{.86}d^{-2.86} - mc^{.86}d^{-2.86}$

Addition and subtraction

Division

Substitutions for variables with unique exponents.

Substitutions for messy things.

New equality constraints for messy thing substitutions.

Sum constants to be added to the objective function after preprocessing.

Substitute and rewrite the program in standard form.

Step 3 Check Assumptions

Are all variables strictly non-negative (*i.e., all* $x_i \geq 0$)?

Are there any strict inequality constraints?

Is the problem posed as a signomial optimization program?

Workspace

List the variables:	Estimate operating ranges:
f	This is a design variable.
m	$.1 \leq m \leq 20$
c	$c > 0$
d	$d > 0$
n	$n > 3$
List strict inequality constraints:	New constraints:
NONE	NONE
Nonsignomial terms:	Transformed terms:
NONE	

Substitute and rewrite the program in standard form.

Step 4 Remove Redundant Constraints

Look for constraints that are simple multiples of each other.

Look for groups of constraints which added and subtracted from each other equal another constraint or its multiple.

Set up a modified coefficient matrix and do Gauss-Jordan elimination to verify independence for large problems.

Workspace

List constraints.

$$6.05x10^{-5}c^{.86}d^{-2.715} \le 1$$

$$3.55x10^{-2}c^{2}nd^{-1} \le 1$$

$$mf^{-1} + .125nc^{2}d \le 1$$

$$.1m^{-1} \le 1$$

$$\frac{m}{20} \le 1$$

$$dc^{-1} \le .25$$

$$c + d \le 1.5$$

$$c - d \le 0.75$$

$$1.12nd + 1.6d \le 1$$

$$3n^{-1} \le 1$$

$$.05 \le dc^{-1}$$

List possible redundancy matches.

NONE

Write modified coefficient matrix.

Not Necessary for this problem

Write modified variable column matrix.

Substitute and rewrite the program in standard form.

Step 5 Check for Solutions in Constraints

How many variables are there? How many equality constraints? How many inequality constraints?

Is the number of independent equality constraints equal to the number of variables?

Is there a subset of independent equality constraints which have equal numbers of equations and variables?

Use either a linear or nonlinear simultaneous equation solver to get an unique solution to the problem. Then check inequality constraints and single variable bounds.

Is the number of independent equality and inequality constraints equal to the number of variables?

Is there a subset of independent equality and inequality constraints which have equal numbers of equations and variables?

Assume all inequalities constraints are equalities. Use either a linear or nonlinear simultaneous equation solver to get a "feasible problem" solution. Then check the single variable bounds. Save this solution for use as a starting point in the algorithm to solve the problem.

Workspace

The number of variables = ____5_____(a)

The number of equality constraints = _____0_____(b)

(a) - (b) = _____5_____ (= 0 ? Then use nonlinear equation solver)

The solution is:

List variables and their values

NONE

The number of inequality constraints = _____10_____(c)

(a) - (b) - (c) = _____-5_____(= 0 ? Then use nonlinear equation solver)

The feasible answer is:

List variables and their values

NONE

Coefficient matrices of independent subsets of equality constraints

NONE

The solution for these variables is:

List the variables and their values

NONE

Coefficient matrices of independent subsets of inequality and equality constraints

NONE

The feasible answer for these variables is:

List variables and their values

NONE

List the single variable bounds and substitute values from above.

NONE

Substitute and rewrite the program in standard form.

Step 6 Separate Variables

Identify and list unconstrained objective function variables.

Identify and list non-objective function variables appearing in constraints which contain only other non-objective function variables and do not appear in another constraint with an objective function variable.

Identify and list variables which appear in both objective function and constraint terms or in constraints containing variables that link to objective function variables in other constraints.

Look for subsets of the objective function terms containing variables which are either unconstrained or constrained by unique subsets of the constraints. Restate these separated programs as independent subprograms and return to Step 1 to continue the analysis on each of them separately.

Workspace

List unconstrained objective function variables.

NONE

List constrained variables not in the objective function which appear in constraints which do no contain either objective function variables nor variables that are in constraints with objective function terms.

NONE

List remaining variables.

f

m

c

d

n

List unconstrained subprograms.

NONE

List constrained subprograms.

Same as the original

Substitute and rewrite the remaining program in standard form.

Step 7 Look for this Problem Type

The objective function is a single term and the left-hand side of the constraints are single terms (constants on the right-hand side).

If so, then take the logarithm of the objective function and both sides of the constraints and single variable bounds to form a linear program.

Workspace

List the objective function. NO MATCH

List the logarithm of the objective function.

List the constraints.

List the logarithm of the constraints.

If this was a subprogram rewrite the remaining program in standard form.

Step 8 Remove Redundant Bounds

Remove redundant bounds. Redundant bounds are those that are less restrictive than similar bounds on the same variable.

Compare bounds to constraints. Are there any bounds that would violate the constraints? If so, then the bound is redundant.

Check variable bounds for infeasibility or unboundedness.

Workspace

List the single variable bounds for the first variable:

$3n^{-1} \leq 1$

Which can be rewritten as $n \geq 3$.

$.1m^{-1} \leq 1$

List the single variable bounds for the second variable:

Which can be rewritten as $m \geq .1$

$\frac{m}{20} \leq 1$

Which cam be rewritten as $m \leq 20$.

Substitute the lower bound for all variables into the equality and greater than or equal to constraints they appear in. Substitute the upper bound for all variables into the equality and less than or equal to constraints they appear in.

Step 9 Bound Variables in the Constraints

Lower bound variables in the equality and greater than or equal to constraints by assuming all other variables at their upper bounds.

Upper bound variables in the equality and less than or equal to constraints by assuming all other variables at their lower bounds.

List the tightest set of upper and lower bounds you can for each variable.

Look for unique solutions to the constraints in the bounds.

Note: Do not replace constraints with single variable bounds.

Workspace

Lower bounding computations.

Upper bounding computations.

List tightest bounds possible.

There are three constraints which involve only variables d and c.

$.05 \leq dc^{-1} \leq .25$

Which can be rewritten as $.05c \leq d \leq .25c$

$c + d \leq 1.5$

Which can be rewritten as $d \leq 1.5 - c$

$c - d \leq 0.75$

Which can be rewritten as $d \geq c - .75$

and $c - 0.75 \leq 1.5 - c$

or $2c \leq 2.25$

and $c \leq 1.125$

Reversing the procedure to isolate d

$.75 + d \leq c \leq 1.5 - d$

or $d \leq .375$

Using the upper bound on c we can get another upper bound on d

$d \leq (.25)(1.125) = .2812$

which is more restrictive. Leaving

$3 \leq n$

$.1 \leq m \leq 20$

$0 \leq c \leq 1.125$

$0 \leq d \leq .281$

List unique variable answers.

NONE

Step 10 Row Zero Test

If all the coefficients of a constraint are positive and less than or equal to zero (*e.g.*, $x_1^2 + x_2^3 + x_3 + x_4^2 \leq 0$), then set all variables in that constraint equal to zero (trivial result based on our initial assumption that all variables are strictly non-negative).

If all the coefficients of a constraint are negative and greater than or equal to zero (*e.g.*, $- x_1^2 - x_2^3 - x_3 - x_4^2 \geq 0$), then set all variables in that constraint equal to zero.

Workspace

List variables and their values. NO MATCH

Substitute and rewrite the program in standard form.

Step 11 Column $Sign_{GP}$ Test

Minimization problems:

If the variable always appears with a positive $sign_{GP}$ in the objective function and all less than or equal to constraints and negative $sign_{GP}$ in all greater than or equal to constraints, fix it at its lower bound.

If the variable always appears in the objective function and all less than or equal to constraints with a negative $sign_{GP}$ and positive $sign_{GP}$ in all greater than or equal to constraints, fix it at its upper bound.

Maximization problems:

If the variable always appears with a positive $sign_{GP}$ in the objective function and all greater than or equal to constraints and negative $sign_{GP}$ in all less than or equal to constraints, fix it at its lower bound.

If the variable always appears in the objective function and all greater than or equal to constraints with a negative $sign_{GP}$ and positive $sign_{GP}$ in all less than or equal to constraints, fix it at its upper bound.

Check constraints.

Workspace

m - appears with a negative $sign_{GP}$ in the objective function and a positive $sign_{GP}$ in all less than or equal to constraints.

f - appears with a positive $sign_{GP}$ in the objective function and a positive $sign_{GP}$ in all less than or equal to constraints.

c - appears in the objective function with both positive and negative $sign_{GP}$s and in the less than or equal to constraints with positive $sign_{GP}$s.

d - appears in the objective function with both positive and negative $sign_{GP}$s and in the less that or equal to constraints with both positive and negative $sign_{GP}$s.

n appears with a positive $sign_{GP}$ in all less than or equal to constraints and does not appear in the objective function so set it at its lower bound. n = 3.

Substitute and rewrite the program in standard form.

Step 12 Restate the Reduced Problem

Check variable assignments. Sum constant terms in the objective function and on the left-hand side of remaining constraints.

Restate the program in the format given in Step 1.

Is the problem still there?

Repeat previous steps as necessary.

Were any additional problems generated?

Workspace

List variable assignments.

$n = 3$

Objective Function

Minimize: $y = fc^{.86}d^{-2.86} - mc^{.86}d^{-2.86}$

Equality Constraints

NONE

Less Than or Equal to Constraints

Greater Than or Equal to Constraints

Single Variable Bounds

Sum constants to be added to the objective function after preprocessing.

Yielding Constraint	$6.05x10^{-5}c^{.86}d^{-2.715} \leq 1$
Surging Constraint	$10.65x10^{-2}c^{2}d^{-1} \leq 1$
Buckling Constraint	$mf^{-1} + .375c^{2}d \leq 1$
Spring Force Constraints	$.1 \leq m \leq 20$
Diameter Ratio Constraints	$.05 \leq dc^{-1} \leq .25$
Outside Diameter Constraint	$c + d <= 1.5$
Inside Diameter Constraint	$c - d \leq 0.75$
Spring Pocket Length Constraint	$d \leq 0.202$
	$c \leq 1.125$

Substitute and rewrite the program or the solution.

C.2 Analysis

The key to solving this problem is to recognize that f is not a variable. The force applied must be provided from design specifications. Since f is not a variable we can rewrite the objective function as:

$$Minimize: \quad \frac{y}{f} = c^{.86}d^{-2.86} - \frac{m}{f}c^{.86}d^{-2.86}$$

The only constraint containing m/f is

$$mf^{-1} + .375c^2d \le 1$$

or

$$mf^{-1} \le 1 - 0.375c^2d$$

If this constraint is tight at optimality then we can substitute it into the objective function for m/f.

$$Minimize: \quad \frac{y}{f} = c^{.86}d^{-2.86} - (1 - 0.375c^2d)c^{.86}d^{-2.86}$$

or

$$Minimize: \quad \frac{y}{f} = 0.375c^2d \cdot c^{.86}d^{-2.86}$$

or

$$Minimize: \quad \frac{y}{f} = 0.375c^{2.86}d^{-1.86}$$

From this you can see that we want to minimize c and maximize d. Earlier analysis upper bounded d at $d \le 0.202$.

Substituting this value into the other constraints

$.05 \leq dc^{-1} \leq .25$

$c \leq 4.04$ and $c \geq 0.808$

$c + d \leq 1.5$

$c \leq 1.298$

$c - d \leq 0.75$

$c \leq .952$

The tightest bounds on c are

$0.808 \leq c \leq .952$

Set $c = .808$ and $d = .202$.

Check Constraints

Yielding Constraint .003 ≤ 1 okay

Surging Constraint .344 ≤ 1 okay

m will be calculated when f is designed from the Buckling Constraint.

APPENDIX D

AN EXAMPLE FOR MANIPULATING THE DELTA$_{GP}$ MATRIX.

D.1 Smith's Economic Order Quantity with a Bottleneck Constraint.

Smith (1970) proposed the following economic order quantity model for parts that flow through a bottleneck machine. His solution involved estimates of the Lagrange multiplier, it gives good answers if you want to do the derivative calculus. The following approach uses geometric programming to generate the optimality conditions for the n variable case and requires only that you be able to add, subtract, multiply, divide, and take square roots. All of the optimization has been done and the solution follows from a very simple line search in one variable. This solution follows a solution technique by Plenert (1987) which used the optimality conditions to generate a quartic equation which he solved iteratively.

This problem is included here to demonstrate the utility of the delta$_{GP}$ matrix and the optimality conditions, as described in Steps 16, 17, and 18 of the alogorithm, to the analysis of multiple degree of difficulty problems.

$$\textit{Minimize Total Cost} \quad = \quad a_1q_1 + b_1q_1^{-1} + a_2q_2 + b_2q_2^{-1} + ... + a_nq_n + b_nq_n^{-1}$$

$$\textit{Subject to:} \quad k_1q_1^{-1} + k_2q_2^{-1} + ... + k_nq_n^{-1} \quad \leq \quad 1$$

Where:

VARIABLE	DESCRIPTION	UNITS
q_i	*Order Quantity*	*Each*
a_i	*Holding Cost*	*Dollars*
b_i	*Ordering Cost*	*Dollars*
k_i	$\frac{Annual Demand x Setup Time}{Total Setup Time Available}$	*Each*

Rewriting the problem in a more compact form.

$$Minimize: \quad \sum_{i=1}^{n} a_i q_i \quad + \quad \sum_{i=n+1}^{2n} b_i q_i^{-1}$$

$$Subject \quad to: \quad \sum_{i=1}^{n} k_i q_i^{-1} \leq 1$$

$$\forall \quad q_i \quad > \quad 0$$

Step 16 Create the Delta$_{GP}$ Matrix

Write the normality constraint as the first row. If the program is a posynomial, then set the row equal to 1. If it is a signomial, set the row equal to σ

Write the remaining rows by summing the deltas associated with each variable equal to zero (orthogonality constraints).

	δ_1	δ_2	δ_n	RHS
Normality				
Orthogonality Var 1 Var 2 Var 3				

The δ matrix for this problem is:

$$\sum_{i=1}^{2n} \delta_i = 1$$

$$\delta_j - \delta_{n+j} - \delta_{2n+j} = 0 \quad for \forall j$$

Step 17 Use the Optimality Relationships

Write the optimality relationships for the terms in the objective function using Rule 3.

Isolate combinations of terms containing one and two variables.

Write the optimality relationships for the constraints using Rule 4.

Substitute delta values from Step 16 and solve for program variables at optimality.

List Rule 3 relationships.

From the optimality relationships in the objective function (RULE 3):

$$\frac{a_i q_i}{\delta_i} = \frac{b_i q_i}{\delta_{n+i}}$$

List Rule 4 relationships.

From the optimality relationships in the constraints (RULE 4):

$$\delta_{2n+i} = k_i q_i^{-1} \left(\sum_{i=1}^{n} \delta_{2n+1} \right)$$

Step 18 Manipulate the Optimality Relationships

Search both the delta_{GP} matrix and optimality relationships for subsets of independent relationships and solve them for the δs and program variables.

Check the values determined in Step 17 with the constraints and bounds for feasibility and bounds and enumerate the remaining objective function values.

Manipulating the above equations.

From the delta_{GP} matrix

$$\delta_i - \delta_{n+i} = \delta_{2n+i},$$

and

$$\delta_{i+1} - \delta_{n+i+1} = \delta_{2n+i+1}.$$

Dividing the first equation by the second

$$\frac{\delta_{2n+i}}{\delta_{2n+i+1}} = \frac{\delta_i - \delta_{n+i}}{\delta_{i+1} - \delta_{n+i+1}}.$$

By dividing the optimality relationships from the constraints the left had side can be redefined as;

$$\frac{\delta_{2n+i}}{\delta_{2n+i+1}} = \frac{k_i q_i^{-1}}{k_{i+1} q_{i+1}^{-1}}.$$

These ratios can be written to compare any two δ_is. Substituting this definition into the previous equation,

$$\frac{k_i q_i^{-1}}{k_{i+1} q_{i+1}^{-1}} = \frac{\delta_i - \delta_{n+i}}{\delta_{i+1} - \delta_{n+i+1}},$$

Dividing through by δ_1,

$$\frac{k_i q_i^{-1}}{k_{i+1} q_{i+1}^{-1}} = \frac{1 - \delta_{n+i}/\delta_i}{\delta_{i+1}/\delta_i - \delta_{n+i+1}/\delta_i}.$$

Reordering the objective function optamility relationships:

$$\frac{\delta_{n+i}}{\delta_i} = \frac{b_i q_i^{-1}}{a_i q_i},$$

$$\frac{\delta_{n+i}}{\delta_i} = \frac{b_i}{a_i q_i^2}.$$

further,

$$\frac{\delta_{i+1}}{\delta_i} = \frac{a_{i+1} q_{i+1}}{a_i q_i},$$

and

$$\frac{\delta_{n+i+1}}{\delta_i} = \frac{b_{i+1} q_{i+1}^{-1}}{a_i q_i},$$

$$\frac{\delta_{n+i+1}}{\delta_i} = \frac{b_{i+1} q_{i+1}^{-1}}{a_i q_i q_{i+1}}.$$

Substituting

$$\frac{k_i q_i^{-1}}{k_{i+1} q_{i+1}^{-1}} = \left(\frac{1 - b_i/a_i q_i^2)}{a_{i+1} q_{i+1} - b_i/a_i q_i q_{i+1}}\right),$$

and simplifying:

$$\frac{k_i q_i^{-1}}{k_{i+1} q_{i+1}^{-1}} = \frac{(a_i q_i^2 - b_i)/a_i q_i^2}{(a_{i+1} q_{i+1}) - b_{i+1} q_{i+1}/a_i q_i}.$$

Reordering the equation it becomes,

$$\frac{k_i q_i^{-1}}{k_{i+1} q_{i+1}^{-1}} = \frac{\left(a_{iq_i}{}^2 - b_i\right)/a_i q_i^2}{(a_{i+1} q_{i+1}^2 - b_{i+1})/a_i q_i q_{i+1}},$$

dividing and reordering,

$$\frac{k_i q_{i+1}}{k_{i+1} q_i} = \frac{(a_i q_i^2 - b_i)}{(a_{i+1} q_{i+1}^2 - b_{i+1})}\left(\frac{q_{i+1}}{q_i}\right).$$

finally leaving,

$$\frac{k_i}{k_{i+1}} = \frac{\left(a_{iq_i}{}^2 - b_i\right)}{(a_{i+1} q_{i+1}^2 - b_{i+1})},$$

or

$$\frac{(a_i q_i^2 - b_i)}{k_i} = \frac{(a_{i+1} q_{i+1}^2 - b_{i+1})}{k_{i+1}}.$$

Similarly we can write,

$$\frac{(a_i q_i^2 - b_i)}{k_i} = \frac{(a_{i+2} q_{i+2}^2 - b_{i+2})}{k_{i+2}}$$

These equations can be reduced to n-1 simultaneous independent non-linear equations of the form:

$$\left(\frac{a_1}{k_1}\right) q_1^2 - \left(\frac{a_2}{k_2}\right) q_2^2 - \frac{b_1}{k_1} + \frac{b_2}{k_2} = 0$$

$$\left(\frac{a_1}{k_1}\right) q_1^2 - \left(\frac{a_3}{k_3}\right) q_3^2 - \frac{b_1}{k_1} + \frac{b_3}{k_3} = 0$$

$$\left(\frac{a_1}{k_1}\right)q_1^2-\left(\frac{a_4}{k_4}\right)q_4^2-\frac{b_1}{k_1}+\frac{b_4}{k_4}=0$$

and

$$\left(\frac{a_1}{k_1}\right)q_1^2-\left(\frac{a_n}{k_n}\right)q_n^2-\frac{b_1}{k_1}+\frac{b_n}{k_n}=0$$

Embedded in this matrix is a sub problem which will help with the solution to the overall problem.

D.2 Basic Optimality Equations:

$$\left(\frac{a_1}{k_1}\right)q_1^2-\frac{b_1}{k_1}+\frac{b_2}{k_2}=\left(\frac{a_2}{k_2}\right)q_2^2$$

$$\left(\frac{a_1}{k_1}\right)q_1^2-\frac{b_1}{k_1}+\frac{b_3}{k_3}=\left(\frac{a_3}{k_3}\right)q_3^2$$

$$\left(\frac{a_1}{k_1}\right)q_1^2-\frac{b_1}{k_1}+\frac{b_4}{k_4}=\left(\frac{a_4}{k_4}\right)q_4^2$$

and

$$\left(\frac{a_1}{k_1}\right)q_1^2-\frac{b_1}{k_1}+\frac{b_n}{k_n}=\left(\frac{a_n}{k_n}\right)q_n^2$$

Since q_i is defined as being strictly greater than zero and it makes no physical sense for it to be negative, then the left hand side of these equations must be greater than zero. Setting the left-hand sides equal to zero:

$$\left(\frac{a_1}{k_1}\right)q_1^2-\frac{b_1}{k_1}+\frac{b_2}{k_2}=0$$

$$\left(\frac{a_1}{k_1}\right)q_1^2-\frac{b_1}{k_1}+\frac{b_3}{k_3}=0$$

$$\left(\frac{a_1}{k_1}\right)q_1^2-\frac{b_1}{k_1}+\frac{b_4}{k_4}=0$$

and

$$\left(\frac{a_1}{k_1}\right)q_1^2-\frac{b_1}{k_1}+\frac{b_n}{k_n}=0$$

Solving for q_1^2:

D.3 Non-Negativity Equations

$$q_1^2=\left(\frac{k_1}{a_1}\right)\left(\frac{b_1}{k_1}-\frac{b_2}{k_2}\right)$$

$$q_1^2=\left(\frac{k_1}{a_1}\right)\left(\frac{b_1}{k_1}-\frac{b_3}{k_3}\right)$$

$$q_1^2=\left(\frac{k_1}{a_1}\right)\left(\frac{b_1}{k_1}-\frac{b_4}{k_4}\right)$$

and

$$q_1^2=\left(\frac{k_1}{a_1}\right)\left(\frac{b_1}{k_1}-\frac{b_n}{k_n}\right)$$

Notice that the we must use the maximum value of q_1 to satisfy all of the equations.

D.4 The Algorithm

Step 1 - Calculate the q_1 from non-negativity equations.
Step 2 - Take the maximum q_1. (Round to the nearest integer value).

Note - This value will necessarily violate the constraint because of the way the problem is formulated.

Step 3 - Calculate q_i for i = 2 to n by solving the basic optimality equations.
Step 4 - Use these values in the constraint equation to calculate the value of the left hand side of equation.
Step 5 - Calculate the value of the objective function concurrently to observe the unconstrained minimum.
Step 6 - If the value of the constraint equation's left hand side is greater than 1. Then increase q_1 by an integer amount and return to step 3 and resolve.
Step 7 - If the left hand side is less than 1 by an acceptable amount. Did you observe a steady decrease in the objective function value and then an increase? If yes, STOP! If no, return to step 3 and continue until you see an increase in the objective function value.

Note - If the ordering cost dominates the setup cost then the algorithm will run into the constraint before the Total Cost is at its minimum. This is the equivalent of solving the unconstrained problem.

Step 8 - If the amount is unacceptable go back to the last infeasible value of q_1 and decrease the step size.

Note - Your first guess will be infeasible (i.e., it will violate the constraint) and subsequent guesses should bring the left hand side of the equation closer to 1. Using non integer values makes no sense in the real problem so you will have to accept some slack in the constraint to get a realistic answer. This algorithm was tested on a Lotus 1-2-3 spreadsheet and it worked very nicely. It can also be managed on a hand calculator for small problems.

D.5 EXAMPLE PROBLEM USING A SPREADSHEET

HOLDING	ORDERING	CAPACITY
COST	COST	USAGE
a(i)	b(i)	k(i)
5.00	50000.00	4.00
20.00	72000.00	32.00
10.00	144000.00	120.00

NONNEGATIVITY CALCULATIONS

q(1)90.55

q(1)95.08

BASIC QUANTITY q(1)	RELATIVE QUANTITY q(i)	OBJECTIVE FUNCTION aq+b/q	OBJECTIVE FUNCTION Σ aq + b/q	VALUE OF CONSTRAINT k(i)/q(i)	TOTAL VALUE CONSTRAINT Σ k(i)/q(i)
1ST	ITERATION				
96.00		1000.83		0.04	
	45.08	2498.80		0.71	
	51.38	3316.41	6816.03	2.34	3.09
2ND	ITERATION				
98.00		1000.20		0.04	
	52.99	2418.54		0.60	
	91.98	2485.37	5904.12	1.30	1.95
3RD	ITERATION				
100.00		1000.00		0.04	
	60.00	2400.00		0.53	
	120.00	2400.00	5800.00	1.00	1.57
4TH	ITERATION				
102.00		1000.20		0.04	
	66.39	2412.31		0.48	
	143.04	2437.11	5849.61	0.84	1.36
5TH	ITERATION				
104.00		1000.77		0.04	
	72.33	2442.05		0.44	
	163.22	2514.43	5957.26	0.74	1.22
6TH	ITERATION				
106.00		1001.70		0.04	
	77.92	2482.45		0.41	
	181.49	2608.35	6092.50	0.66	1.11
7TH	ITERATION				
108.00		1002.96		0.04	
	83.23	2529.72		0.38	
	198.39	2709.77	6242.45	0.60	1.03

8TH	ITERATION				
110.00		1004.55		0.04	
	88.32	2581.59		0.36	
	214.24	2814.56	6400.70	0.56	0.96
9TH	ITERATION				
109.00		1003.72		0.04	
	85.80	2555.18		0.37	
	206.43	2761.90	6320.80	0.58	0.99