

**A PROGRAM FOR MANIPULATING AND SOLVING NONLINEAR  
GENERALIZED POLYNOMIAL OPTIMIZATION PROBLEMS  
WITH GEOMETRIC PROGRAMMING AND  
SYMBOLIC CONSTRAINT REDUCTION**

By  
**David C. Ketchum**

ProQuest Number: 10794234

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



ProQuest 10794234

Published by ProQuest LLC (2018). Copyright of the Dissertation is held by the Author.

All rights reserved.

This work is protected against unauthorized copying under Title 17, United States Code  
Microform Edition © ProQuest LLC.


ProQuest LLC.  
789 East Eisenhower Parkway  
P.O. Box 1346  
Ann Arbor, MI 48106 – 1346

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 Master of Science (Mathematical and Computer Sciences).

Golden, Colorado

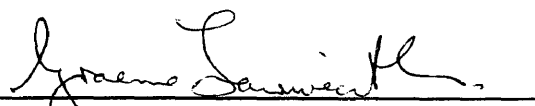
Date Oct 23, 1996

Signed:   
David C. Ketchum

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

Golden, Colorado

Date 10/29/96

  
Dr. Graeme Fairweather  
Professor and Head,  
Mathematical and Computer  
Sciences Department

## ABSTRACT

Non-linear programming problems consisting of generalized polynomials frequently occur in engineering environments. This thesis presents the capabilities of a program designed to aid the researcher in reducing the complexity of such problems through analysis, modification, and elimination of constraints and variables. The reduced program is then suitable for solution by any non-linear program solver. This program implements several Geometric Programming (GP) algorithms including a zero degree of difficulty solver and several unconstrained iterative condensation methods. The program is implemented in C++ using Object Oriented design and coding principles. The Algebraic and AlgSystem classes provide the infrastructure for manipulating generalized polynomials and systems of generalized polynomials. Computational experience is presented for problems chosen from the literature. The program, its documentation, and the User's Guide are available on a diskette. The program runs on Macintosh and Windows based personal computers. The program is suitable for use for developing new algorithms, research, and teaching of GP/DD

## TABLE OF CONTENTS

ABSTRACT . . . . .	iii
LIST OF FIGURES . . . . .	vi
ACKNOWLEDGMENTS . . . . .	vii
Chapter 1. INTRODUCTION . . . . .	1
Chapter 2. GEOMETRIC PROGRAMMING . . . . .	5
Form of Geometric Programming Problems . . . . .	5
Form of the Dual Problem . . . . .	7
Woolsey's Rules . . . . .	8
Degree of Difficulty . . . . .	9
Advanced Sign Table . . . . .	10
Chapter 3. GPSOLV DESCRIPTION . . . . .	12
General . . . . .	12
User Interface . . . . .	13
Algebraic Structure and Manipulations . . . . .	15
GP Manipulations . . . . .	19
Kirk's Algorithm . . . . .	20
Zero DD Solver . . . . .	26
POSCON, MULTICON, and SIGCON . . . . .	28
Systems of Equations . . . . .	29
Chapter 4. COMPUTATIONAL EXPERIENCE . . . . .	31
MX Missile Problem . . . . .	31
Separable Programs . . . . .	35
Ratliff's Degenerate Problem . . . . .	38

Chapter 5. FUTURE WORK . . . . .	41
Algebraic Manipulations . . . . .	41
User Interface . . . . .	42
Kirk's Algorithm . . . . .	43
Geometric Programming . . . . .	44
Additional Solution Methods and Features . . . . .	46
REFERENCES CITED . . . . .	49
APPENDIX . . . . .	52

## LIST OF FIGURES

Figure 1 - Typical Screen . . . . .	14
Figure 2 - Algebraic Expression Symbolic Tree . . . . .	16
Figure 3 - Example GPSOLV Output of Kirk's Algorithm . . . . .	20
Figure 4 - Flowchart of Implemented Steps of Kirk's Algorithm . . . . .	22

## ACKNOWLEDGMENTS

This work is based on the works of many prior C.S.M. Operations Research Guild members including W. Dolan, J. Kirk, R. Ratliff, J. Thome, G. Wessels, and, of course, R.E.D. Woolsey. I have not met some of these people, but I thank them for their contributions to this thesis and my education through the work they left behind. The program is based on the “Algebraic Manipulator” concocted by the author as a project for Dr. Jean Bell's Object Oriented Software Development class. While much of the code has changed and been reworked as new functionality was added, the contributions of the team that did the original project are significant. My thanks to Joel McIntyre, Joe Shoenbeck, and Tou Yang for their contributions in the design and implementation of the Algebraic, Tree, and Iterator classes. Dr. Ruth Maurer and Professor Bill Astle have made major contributions to my education while in the Guild. I thank them and wish them well in their new endeavors. My spouse, Karen, and my children supported me through this process despite my occasional ill humor and tendency to be over committed. Lastly, to the many members of the Guild with whom I have spent untold hours getting through graduate school, by and large, it was great fun.

## Chapter 1

### INTRODUCTION

In my opinion, the Geometric Programming (GP) class at CSM seems to be one part mathematics and one part magic. The ability of the technique to solve very complex appearing optimization problems with relative ease and rapidity is intriguing. Preprocessing is the word used to describe various activities designed to simplify solutions by looking for relationships, particularly in the constraints. Preprocessing involves identifying constraints that are dominated by other constraints and can be ignored, deciding which constraints are not tight from the bounds, identifying constraints that must be tight, looking for redundant constraints that are linear combinations of other constraints, and looking for simpler problems within larger problems. Dr. Woolsey's emphasis on 'looking at the problem' to gain insight and ease the solution of all types of problems was the main theme in the work of Kirk (1988) and Flewelling (1994). Kirk wrote down a step by step preprocessing algorithm for pen and paper solution of Geometric Program (GP) class problems. Flewelling used these preprocessing techniques on a suite of problems the Air Force Institute of Technology (AFIT) uses to test nonlinear computer codes. Using these techniques Flewelling was able to find solutions to a vast

majority of these problems by hand using both GP and non-GP based techniques.

Many times words to the effect of "I have done the work of defining the algorithm suitable for coding for the computer" come up in classes and in theses. The object of this thesis is a computer program which provides the infrastructure needed to implement GP algorithms, and which implements a small number of these algorithms. In particular, it provides an interactive environment for manipulating generalized polynomials. The goal was to provide help for the tedious algebraic manipulations needed to put the problem in GP form. Mathematica [1992], Theorist [1993], or Maple[1993] provide much richer environments for doing these kinds of symbolic manipulations and are far superior to GPSOLV in their capabilities. However, this program needed a method of storing and manipulating generalized polynomials so that the structures would be known and available to the algorithm writer. Symbolic programs like Mathematica provide only limited access to the algebraic structure of expressions stored in them. Secondly, the author desired some knowledge of the relatively new object oriented computer science technology. Algebraic expressions lent themselves well to object oriented design and implementation.

The program is general in its ability to manipulate equations symbolically, substitute for variables, and make deductions about the constraints based on Kirk's algorithm. This ability to preprocess

allows the problem variable space to be reduced. For instance, if a simple constraint is deduced to be tight, then one of the variables can be solved for in terms of the others and substituted everywhere else it appears in the problem. The problem space is thus reduced by one dimension. These reduced problems could then be fed to any non-linear solver. This program chooses to use only GP algorithms in an attempt to solve the problem. This program is a tool for teaching and doing research in the GP arena.

The program can solve zero degree-of-difficulty GP posynomial problems and signomial problems of the form described by Passy and Wilde (1971) and used in Beightler and Phillips (1976). The algorithms POSCON (Ratliff 1986), MULTICON (Ratliff 1986), and SIGCON (Thome 1988) are implemented to solve unconstrained problems of the appropriate form. The program analyzes the system of equations and if the system is in GP form (minimization with all constraints  $\leq 1$  and all expressed as generalized polynomials), will store the dual equation matrix, and exponent matrix with equation boundaries, a sign table matrix, and calculate the number of terms and degree of difficulty. These structures are described in detail in Chapter 3 pages 19-20. These data structures are a sufficient starting point for all GP algorithms implemented or envisioned. A researcher can make use of this program to derive data structures needed for a new algorithm. This allows the researcher to be free of worrying about the input and parsing of models and concentrate on

solving the problem at hand. Many of the steps of Kirk's preprocessing algorithms have been coded to demonstrate the power of the program to analyze a problem for structure and make use of this knowledge to reduce the problem.

## Chapter 2

### GEOMETRIC PROGRAMMING

#### Form of Geometric Programming Problems

Non-linear programming problems occur in engineering. Engineering designs are desired that minimize cost, or some other parameter, given a series of engineering constraints dictated by the problem, engineering standards, safety margins, or even statutes. Frequently, engineering problems are of the form of generalized polynomials, or can be transformed to that form. The function coefficients are often estimated by experimentation and regression. The regression method is generally used with linear functions or non-linear power functions. This thesis deals with minimization of non-linear problems which are formed by generalized polynomials. The general form of such problems is :

$$\begin{aligned} \text{Minimize : } g_0(\vec{x}) &= \sum_{i=1}^{I_0} \sigma_{0i} c_{0i} \left( \prod_{j=1}^J x_j^{a_{0ij}} \right) \\ \text{Subject to : } g_m(\vec{x}) &= \sum_{i=1}^{I_m} \sigma_{mi} c_{mi} \left( \prod_{j=1}^J x_j^{a_{mij}} \right) \leq 1; \quad m = 1, 2, \dots, M \\ \text{Where : } \sigma_{mi} &= \pm 1; \quad m = 0, 1, \dots, M \quad i = 1, 2, \dots, I_m \\ c_{mi} &> 0; \quad c_{mi} \in \mathfrak{R}; \quad m = 0, 1, \dots, M \quad i = 1, 2, \dots, I_m \\ x_j &> 0; \quad a_{mij} \in \mathfrak{R}; \quad j = 1, 2, \dots, J \quad m = 0, 1, \dots, M \quad i = 0, 1, \dots, I_m \end{aligned}$$

The number of variables is  $J$ . The  $I_m$  represent the number of terms in each equation. Notice that the “generalized” polynomials allow any real exponent represented by the  $a_{mij}$  above. The sign of the coefficients is represented by the  $\sigma_{mi}$ . A posynomial is a generalized polynomial where all constants are positive which in this case makes all  $\sigma_{mi}$  equal to one. A signomial does not restrict the  $\sigma_{mi}$  thus each coefficient can be positive or negative.

Geometric Programming is a technique for solving problems which are in generalized polynomial form. GP was first formalized in Duffin, Peterson, and Zener (1967). They were able to show that certain type of posynomial constrained problems could be transformed by their technique into a convex nonlinear problem. Convex problems are a well known class of problem which can be solved globally by many techniques. Reklaitis, Ravindran, and Ragsdell [1983] classify these techniques as Direction Generation methods (Feasible Directions, Reduced Gradient, Generalized Reduced Gradient chapter 9), and Quadratic Approximation methods (Constrained Variable Metric Method, Quadratic Programming Chapter 10). GP problems have the additional restrictions that all of the variables are positive and all constraints must be tight at optimality. There is an excellent review of the history of GP in Jackson (1994).

## Form of the Dual Problem

The primary advantage of GP is that the transformed problem, called the dual, always has linear constraints. An important contribution used in this thesis is Woolsey's "Four Rules of Geometric Programming" published in Woolsey and Swanson (1975), and more recently published in Woolsey (1992). Henceforth, these will be referred to as the "Rules" without further attribution. This thesis uses the Woolsey nomenclature and Woolsey's book should be familiar to readers of this thesis. The dual to the above problem is :

$$\begin{aligned} \text{Maximize } d_0(x) &= \sigma_0 \left[ \prod_{m=0}^M \prod_{i=1}^{I_m} \left( \frac{c_{mi} \delta_{m0}}{\delta_{mi}} \right)^{\sigma_{mi} \delta_{mi}} \right]^{\sigma_0} \\ &= \sigma_0 \left[ \prod_{i=1}^{I_0} \left( \frac{c_{0i}}{\delta_{0i}} \right)^{\delta_{0i}} \prod_{m=1}^M \delta_{m0}^{\delta_{m0}} \right]^{\sigma_0} \end{aligned} \quad (\text{Rule I})$$

where  $\sigma_0 = \pm 1$ , the sign of  $g_0$

$$\text{Subject To : } \sum_{i=1}^{I_0} \sigma_{0i} \delta_{0i} = \sigma_0 \quad (\text{Rule IIa})$$

$$\sum_{m=0}^M \sum_{i=1}^{I_m} \sigma_{mi} a_{mij} \delta_{mi} = 0; \quad j = 1, \dots, J \quad (\text{Rule IIb})$$

Where  $\delta_{mi} \geq 0$ ;  $\delta_{mi} \in \mathfrak{R}$ ;  $i = 1, \dots, I_m$  and  $m = 0, 1, \dots, M$   
all signum functions  $\sigma_{mi} = \pm 1$

$$\delta_{m0} \equiv \sum_{i=1}^{I_m} \sigma_{mi} \delta_{mi} \geq 0; \quad m = 1, 2, \dots, M$$

The rules require that the primal problem be in a canonical form of a minimization, subject to constraints that are less than or equal to one. Rules I and II provide a method to write down the dual problem. Woolsey only gives the simpler form of dealing with posynomials in his book. The more general form for signomials is given here.

Because there is no duality gap (Passy and Wilde,1971), the objective functions are equal at optimality. Let  $x^*$  and  $\delta^*$  represent the primal and dual variables at optimality. Then :

$$d_0(\delta^*) = g_0(\vec{x}^*)$$

For the dual variables to be allowed to equal zero it must be noted that by L'Hospital rule :

$$\lim_{\delta_{mi} \rightarrow 0} \left( \frac{c_{mi} \delta_{m0}}{\delta_{mi}} \right)^{\sigma_{mi} \delta_{mi}} = 1$$

### **Woolsey's Rules**

Rules III and IV establish relationship between the dual and primal variables at optimality. Rules III and IV can be written :

$$c_{0i} \prod_{j=1}^J x_j^{a_{0ij}} = \delta_{0i} \sigma_0 g_0(\vec{x}^*); \quad i = 1, 2, \dots, I_0 \quad (\text{Rule III})$$

$$c_{mi} \prod_{j=1}^J x_j^{a_{mij}} = \frac{\delta_{mi}}{\delta_{m0}}; \quad m = 1, 2, \dots, M \quad i = 1, 2, \dots, I_m \quad (\text{Rule IV})$$

$$\text{where } \delta_{m0} \equiv \sigma_m \sum_{j=1}^J \sigma_{mj} \delta_{mj} \geq 0; \quad m = 1, 2, \dots, M$$

$$\delta_{00} \equiv 1$$

$$I \equiv \sum_{m=0}^M I_m \quad (\text{Number of terms in System})$$

Note that Rule III says that each term of the objective function equals its dual variable times the optimal value of the objective function. The dual variables in the objective function give the proportion of contribution of each term. Similarly, Rule IV states that the dual variable corresponds to the contribution that term makes to satisfying the constraint. Both Rules III & IV are in the form of a term of the original problem equaling a simple function of the dual variables and the optimum value of the objective function. Since the optimal value of the objective function is computable by Rule I, the right hand sides of the Rule III and IV equations are easily determined. This form will be exploited later to help solve for the primal variables.

### Degree of Difficulty

An important concept in GP is the degree of difficulty (DD). The degree of difficulty of a GP problem is defined as the number of terms in the problem less the number of variables less one. This is

actually the number of columns minus the number of rows of the matrix formed by Rule II. When the degree of difficulty is zero, the matrix is square and the dual variables can be solved for directly and uniquely. A typical linear algebra method for solution to the zero DD problem involves matrix inversion. In the GP case, if the matrix is singular (has no inverse), then the problem is ill formed in some way.

### **Advanced Sign Table**

The concept of GP signs is used to create a sign table. The GP sign is the sign of the product of a term's constant with a variable's sign. Thus, in the term  $-3x^2y^{-3}$  the GP sign of the x in this term is minus one (-1) while the GP sign for y it is positive one (+1). It is the sense of the movement of the term as the variable is increased in magnitude with all other variables being held constant. This sign is used in several places in the code in particular in determining whether a new bound of a variable is an upper or lower bound.

Many researchers at the Colorado School of Mines over the years have defined algorithms, which were implemented in computer code, which solved particular classes of problems. Condensation is discussed in Beightler & Phillips (1976) as a technique to reduce problem difficulty. Ratliff (1986) used condensation to solve unconstrained posynomial problems in one or

more variables. Thome (1988) extended this work to a class of signomial one variable problems by allowing negative coefficients with positive powers. The one variable posynomial algorithm of Ratliff is generally referred to in this thesis by its subroutine name POSCON. Similarly, MULTICON is the routine for solution of multiple variable problems. SIGCON is the routine for one variable signomials of the form described in Thome. Dolan (1994) used condensation in much the same manner for a more general class of unconstrained signomial problems.

## **Chapter 3**

### **GPSOLV DESCRIPTION**

#### **General**

GPSOLV is the primary product of this thesis. Its design, implementation, and debugging consumed most of the time spent. GPSOLV is written in C++ and currently includes approximately 11,000 lines of code and comment. The description of the program can be divided into four sections - User interface, Algebraic manipulations, GP manipulations, and GP solution algorithms. This thesis will describe the results of using this program and an overview of its utility from the perspective of a potential user or researcher. Since the program is quite large and the data structure complex requiring a great deal of detail not appropriate for a thesis, the actual documentation of the program, the code of the program itself, the user's guide, and other technical details are included on the diskette which is an integral part of this thesis. This diskette will be in DOS format and contain the GPSOLV program and all software for use under Microsoft Visual C++ version 1.0 under Window 3.1. Most of the code was written by the author though some standard numeric routines like matrix inversion are from standard libraries (Shammas 1995) and some of the Algebraic and Tree classes were developed as

a project in MACS 567 in the fall of 1994. Some of the documentation from the final report of this project has been modified and included on the documentation disk.

### **User Interface**

The program was developed on a Macintosh Quadra 630 system using the Symantec C++ version 7.0. It was designed to be transportable to Microsoft Visual C++ with a minimum of conditional code or changes. For this reason the user interface is a simple command line in a terminal style scrolling window. This simple model was supported by both environments and, while not attractive, is quite functional. Commands are of the form :

```
command [arg1] [arg2] [arg3] ...
```

The commands are listed and described in detail in the User's Guide. The program's primary output is a list of the system of equations with ancillary output about the truth of the constraints, the value of the objective function at the current variable settings, the current settings of variables and the variable bounds, and the system's state as a GP problem. Commands are used on individual equations to get them in correct form for GP solution and then use the GP related commands to solve, explore, or simplify the problem. Algebraic expressions are in a format familiar to Basic and C

```

GP Algebra Assistant V1.00 8/14/96 . Please use HELP for
more information

obj : y = 452x1^-1x2^-1                Z = 11.18
c1 : 1.1e+04x1^-1.52x2 <= 1           1<=1      Tight
c2 : 0.0193x1x2^0.83 <= 1             1<=1      Tight

# Vars= 2    x1 = 174.278786    x2 = 0.231899
Lower Bnd   x1 > 1e-300        x2 > 1e-300
Upper Bnd   x1 < 1e+300        x2 < 1e+300
G.P. Form = O.K.    # Terms =3   D.D. =0

C2>

```

Figure 1 - Typical Screen

programmers. An example of the screen is shown below.

The problem from this screen in conventional mathematical form is :

$$\begin{aligned}
 \text{Minimize : } & y = 452x_1^{-1}x_2^{-1} \\
 \text{Subject to : } & 11000x_1^{-1.52}x_2 \leq 1 \\
 & 0.0193x_1x_2^{0.83} \leq 1
 \end{aligned}$$

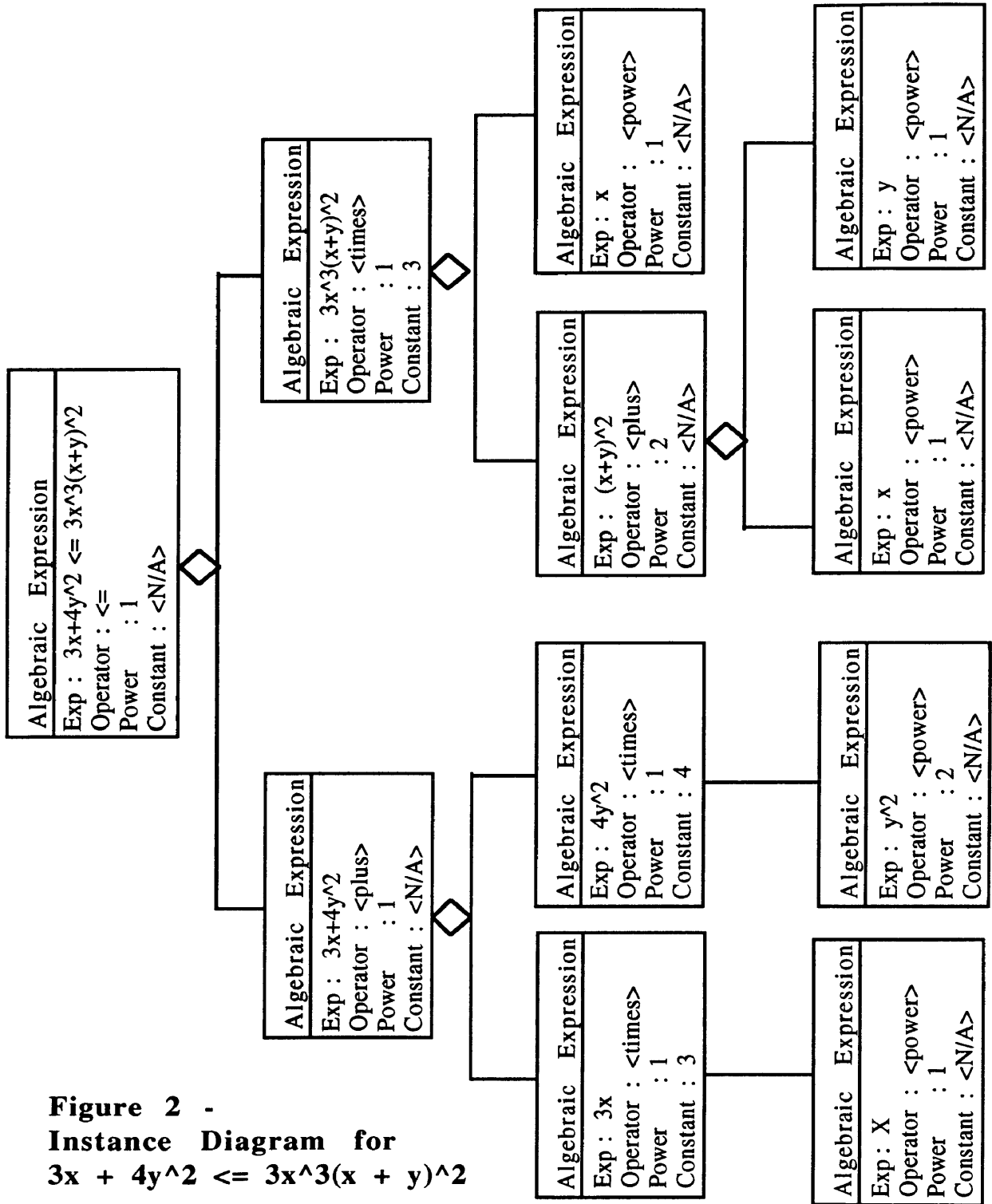
Note this model is at optimality. Both constraints evaluate to  $1 \leq 1$  and are tight. The value of the objective function is 11.18. The current settings of the two variables are  $x_1=174.239$  and  $x_2=0.2319$ . The bounds are at the default settings of  $10^{-300}$  and  $10^{300}$ . This problem is from Beightler and Phillips (1976) page 193. It is a posynomial problem with two variables and three terms. It is in appropriate canonical form for a GP problem as signified by the "G.P.

Form = O.K.” It is zero degrees of difficulty. The “C2>“ is the command prompt and indicates that constraint 2 is the focus of statement oriented commands.

### **Algebraic Structure and Manipulations**

The generalized polynomials are represented as a tree of Algebraics. As an example, the expression  $3x+4y^2 \leq 3x^3(x+y)^2$  is shown in schematic form in Figure 2. Tree is a C++ class and the Algebraic class uses the object oriented concept of inheritance to use the operations from the Tree class. These operations deal in adding leaves to the tree, cutting off branches, replacing branches, etc. An Algebraic takes five forms - a constant, an Algebraic to a power, a multiplier of Algebraics, an addition of Algebraics, or a binary relationship between Algebraics. The bottom most terms are always constants or a variable to a power. Each node of the tree can have several children. A simple term like  $2xy^2z$  has a multiply node indicating that its children are multiplied together. In this case the multiply node contains the constant 2 and has three children - a power node for x,  $y^2$ , and z. In this case, the power nodes do not have children as they are simple variables to powers.

The system is able to use these simple nodes to represent multinomials to powers, sums of terms which can be simple or very



**Figure 2 -**  
**Instance Diagram for**  
 $3x + 4y^2 \leq 3x^3(x + y)^2$

complex, and Algebraic relationships separated by any of the equality or inequality binary relations. The top of the tree is the binary operator less than or equal to ( $\leq$ ). It has two children representing the left hand and right hand sides of the inequality. The left hand side is a sum with two children which are multiplication nodes of constants and simple variables. The multiplication nodes only have one child since they have only one variable. The right hand side is more complex because there is parenthetical expression  $(x + y)$ . The right hand side node is a multiplication node with two children - a power node at power two over the sum of  $x+y$  and a simple variable power node for the  $x^3$ .

The organization into trees of Algebraics led to another object oriented structure called an iterator. These allow the tree to be explored in a systematic manner. One manner allows each node of a tree to be visited. Another type allows each of the children of a node to be visited, but not the grandchildren. These iterators are used for instance by the routines that simplify expressions. They visit each node and decide if a simplification is possible. For instance, on a addition node a routine might look at all of the children for constant nodes. If it finds more than one, it can sum up the constants and replace the multiple constant nodes with just one node. As another more complex example two iterators to a addition node could be used to look at each term. Iterators on the terms could then look at the

variables and their powers to see if the terms are conformable for addition. If two such nodes are found they are combined by adding their constants. As a last example there is a routine which looks at each node and decides if an equation is in simplified generalized polynomial form. That is, each side is a sum of simple variables multiplied together, a single term of simple variables multiplied together, or a constant.

Recursion is used within the Algebraic class to evaluate the value of the algebraic expressions and to create the text string that represents the expression to the user. Each node simply calls its children. A child returns a value if it is a constraint node or simple power node. In these nodes the current value of the variable is used to compute the value of the node if the function is to evaluate. If a node is too complex to evaluate, it just calls its children. Its children will eventually return a value after, possibly recursively, calling the evaluate function on multiple generations of descendant nodes. Similarly, the Ascii routine knows how to string together children nodes to form a string for the current nodes. All children are recursively called until the end nodes return simple strings.

Two commands make extensive use of these types of structures EXPAND and GATHER. EXPAND looks through an Algebraic searching for the most deeply nested parenthetical expression and then expands it. It can take multinomials to integral powers  $(x + y)^2$  or

distribute a multiplication over addition like  $2x(x+y)$ , or distribute two multinomials multiplied together  $(x+y)(2x+2y)$ . GATHER looks through Algebraics for conformable additions, terms which have canceled, variables which have zero power, etc. and takes appropriate action. For more information on the details of such routines see the documentation on the diskette in the Algebraic class documentation.

### **GP Manipulations**

When a system of equations is in GP form, several data structures are completed, converting the system into the typical matrix form that GP analysis programs need as their input. A simple GP problem is shown as Figure 3 with its corresponding arrays. This is the same problem displayed in screen form in Figure 1. The terms are numbered. This number becomes the index for a term in the other arrays. There is an array which contains indices to the last term of each equation or inequality. Using the index to equation array, an algorithm can tell in which equation a term resides. By convention the objective function is the first equation. The other structures are a sign table, a matrix of exponents, an array of constants for each term, and the matrix of Rule II equations. The GP routines use these structures or the Algebraic representation to manipulate the equations. The matrix or vector name within the Algebraic System class is given in parenthesis.

$$\text{Minimize : } y = 452x_1^{-1}x_2^{-1}$$

$$\text{Subject to : } 11000x_1^{-1.52}x_2 \leq 1$$

$$0.0193x_1x_2^{0.83} \leq 1$$

Constant(C)	Power(A)	Sign Table(Sign)	Term #(Nterm)
$\begin{bmatrix} 452 \\ 11000 \\ 0.0193 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 \\ -1.52 & 1 \\ 1 & 0.83 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 \\ -1 & 1 \\ 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$

Dual Variables(D)	(RHS)
$\begin{bmatrix} 1 & 0 & 0 \\ -1 & -1.52 & 1 \\ -1 & 1 & 0.83 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

Figure 3 - Example Arrays and Matrices

The number of terms is 3 and is the number of rows in the matrices. These arrays are used by the GP solution routines, the Kirk algorithm routine, and various utilities. The POSCON routine gets the constants of the terms(C), the powers of the variable array(A), and a number of terms as its input (NTERMS). It can then calculate the optimal power of the variable.

### Kirk's Algorithm

The GPKIRK routine does most of the steps of the algorithm by Kirk (1988). It differs from Kirk in that the constraints must already

be in the less than or equal to one form which satisfy Kirk's steps 1 through 3, 13, and 15 through 20 which basically simplify the problem and put it into the GP form. The remaining steps of the algorithm are commented on below. The step numbers correspond to Kirk's thesis. This series of steps is repeated if any changes are found. A flowchart of the step implemented in GPSOLV is included as Figure 4.

Step 4 - Remove Redundant Constraints - This step is looking for constraints which are linear combinations of other constraints and then reducing the constraint set by one of the constraints. The program does not implement this step, but it could be implemented in future work.

Step 5 - Check for Solutions in the Constraints - This step asks if the constraint set has  $N$  equations in  $N$  unknowns and, if so, solve using a non-linear equation solver. This program does not deal with equality constraints directly as they are not part of the GP model. A system that has constraints of the form Kirk describes is not an optimization problem since the constraints dictate the value of the variables and the objective function has no influence in the outcome.

Step 6 - Separate Variables - The program looks at all of the possible ways to divide the variables into two sets which contain all the variables and are mutually exclusive. The objective function and constraints are then examined to see if a subset of the objective

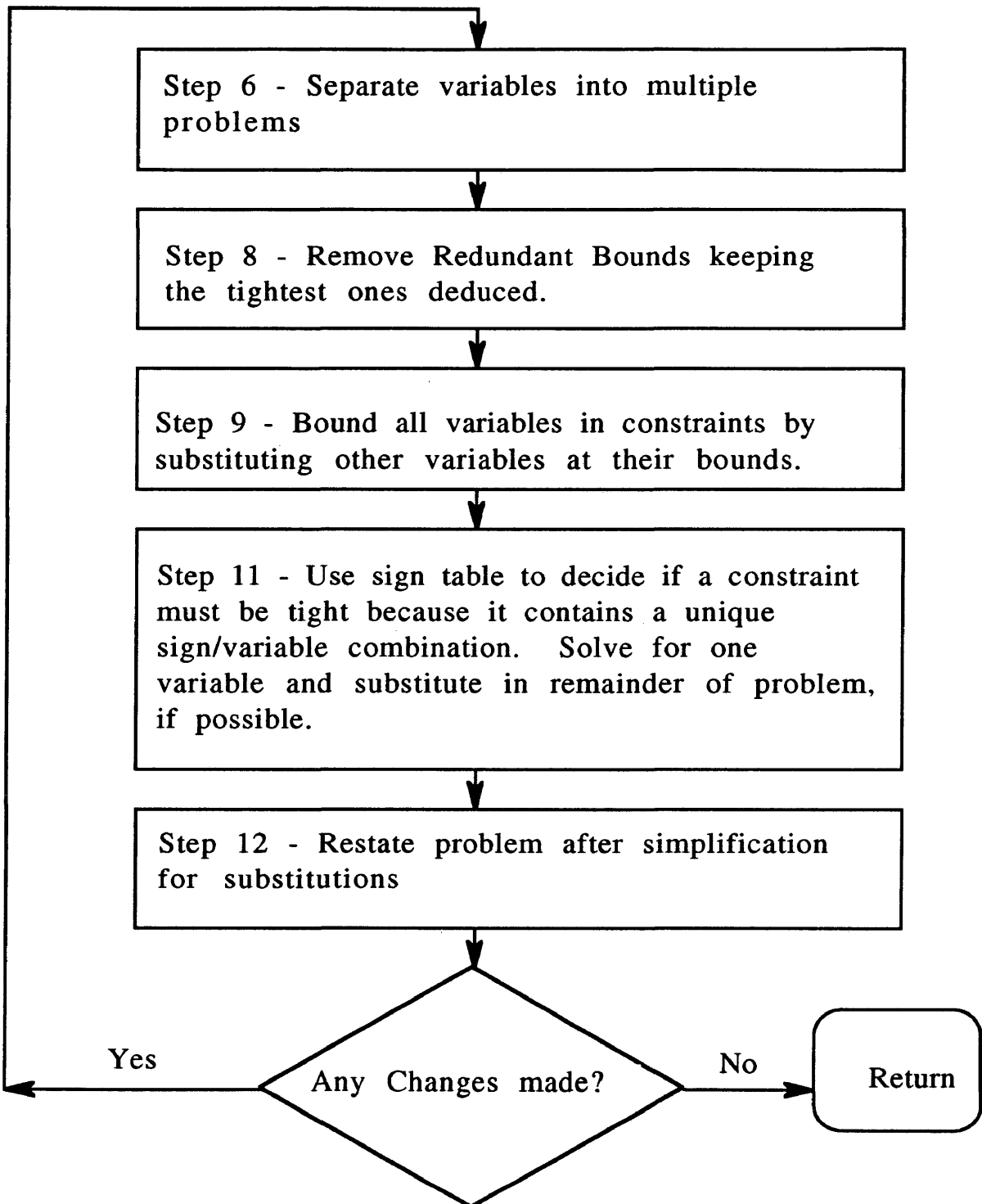


Figure 4 - Flowchart of Implemented Steps of Kirk's Algorithm

function terms and a subset of the constraints contain only the variable set under consideration. If such a set is found, the problem is separable. A new system of equations is formed using only the objective function variables and constraints from the set of variables under consideration. The remaining objective function terms and constraints are put in a second newly constructed system of equations. Each of these new systems of equations can be worked on individually. If a solution to one of the separable systems is found, it can be substituted into the original system thus reducing its complexity. Only one of the systems is the focus of the program at a time. However, any system created can be selected and become the focus of the program. Once a separation is found this function returns. There may be additional separations found by running the routine on the new subset systems.

Step 7 - Look for this Problem Type - If the problem is all single terms, a log transformation can be done which creates a new problem which is all linear. This problem can be solved by any linear programming code and the solution substituted and converted back to a solution to the original problem. This specialized recognition of such a problem is not implemented. However, it should be pointed out that any "special form" problem could be recognized by writing a routine to look for the form. The number of such special forms is open ended.

Step 8 - Remove Redundant Bounds - Look through all simple

bounds equations and eliminate bounds that are less restrictive. The program does perform this function as part of a more general routine that does this step and step 9.

Step 9 - Bound Variables in the Constraints - Bounds can be found by taking each constraint and for each variable in turn and substituting the value of all other variables at the bound which minimizes its contribution to each term. This creates as much room as possible for the variable under consideration in the constraint. If, after substitution, the equation is simple to solve, a new bound on the selected variable can be calculated. If it is tighter than the current bound on the variable, the bound can be made more restrictive. This routine uses the sign table to decide for each variable in each term the bound at which the variable should be set. If the sign table is positive, the variable is set at its lower bound. If the sign table is negative, the upper bound is used. Similarly, when a solution for the selected variable is found, the sign table entry for that variable determines if a possible new lower bound or upper bound has been found. Each variable in each constraint is chosen as the selected variable in succession. If any bounds changes have been made after all variables have had their turn being selected, the entire set is tried again until no changes are made.

Step 10 - Row Zero Test - This checks for constraints with a posynomial left hand side less than or equal to zero. Since this algorithm does not start unless all constraints are in less than or

equal to one form, this situation will never occur.

Step 11 - Column SignGP test - If a variable always appears with the same sign in the sign table, then a bound for it must exist. If such a variable is found, it is set to the appropriate bound and substituted for in the system of equations.

Step 12 - Restate the Reduced Problem - Since variables may have had solutions found for them, the resulting system may have simplifications that can be made. The equations are then simplified and the reduced equation set ready for further analysis.

The remaining steps in Kirk have to do with forming the GP dual problem and manipulating the relationships by hand to find solutions. Since the problem is already in GP form, these steps are already completed. Kirk designed his algorithm to work with much more general optimization problems.

Constraints are also marked as 'must have' when sign table analysis indicates it is the only occurrence of a variable with a sign needed to balance it in the remainder of the problem. If the sign table has a variable with the same sign in all terms, the variable is unbounded and will be driven to zero or infinity depending on its sign in the problem.

At each step the number of changes is tracked. If at completion of the steps some changes have been made, the algorithm is repeated. When no changes are made, the system is returned in its simplified form. Often the system is considerably reduced by these

steps. Kirk's algorithm is particularly effective when simple variable bounds are present. Fortunately, for many engineering problems such constraints are present. The MX missile problem is an example where Kirk's algorithm is very effective and is described in Chapter 4.

### **Zero DD solver**

GP problems which are entirely posynomials have been shown by Duffin, Peterson, and Zener (1967) to be transformable to a convex programming problem. Beightler and Phillips (1976) prove that stationary points in the primal domain have corresponding points in the dual and that at these points there is no duality gap. Since convex problems have only one minima and it is assured that the dual GP problem will have a corresponding stationary point, zero DD GP problems can be solved by inversion of the Rule II matrix. For the class of signomials in Beightler and Phillips (1976) the normal equation has a right hand sign 'signum' corresponding to the sign of the objective function at optimality. Whether the signum takes on a one or minus one value will result in a change of signs of all of the dual variables found by inversion. The zero DD solver inverts the Rule II matrix and checks for singularity. A singular matrix usually means the problem has constraints that are not independent. The program then checks for all negative dual variables. If all dual variables are found to be negative, then the value of the signum is

change to -1 and the dual variables negated. If the signs of the dual variables are mixed, then the problem is ill formed in some manner. Usually mixed sign dual variables result from unbounded variables.

For each term there is a Rule III or Rule IV relationship which involve the term and a expression involving the optimal value of the objective function and the dual variables. If each of these equations is manipulated to put all non-variables on one side and the variables on the other we have an expression that looks like :

$$\prod_{j=1}^J x_j^{a_{0j}} = \frac{\delta_{0i} \sigma_0 g_0(\vec{x}^*)}{c_{0i}} \quad (\text{Rule III})$$

$$\prod_{j=1}^J x_j^{a_{mj}} = \frac{\delta_{mi}}{c_{mi} \delta_{m0}} ; m = 1, 2, \dots, M \quad i = 1, 2, \dots, I_m \quad (\text{Rule IV})$$

Taking the logarithms of both sides we have :

$$\sum_{j=1}^J a_{0j} x_j = \log \left( \frac{\delta_{0i} \sigma_0 g_0(\vec{x}^*)}{c_{0i}} \right) \quad (\text{Rule III})$$

$$\sum_{j=1}^J a_{mj} x_j = \log \left( \frac{\delta_{mi}}{c_{mi} \delta_{m0}} \right) ; m = 1, 2, \dots, M \quad i = 1, 2, \dots, I_m \quad (\text{Rule IV})$$

This is I+1 equations in J unknowns. If the problem is zero DD, then the matrix is square. Thus, the equations can be solved for by inverting the log domain equations and solving for the logs of the variables and then transforming to the original problem domain

through exponentiation.

If a solution is found, all variables are compared against any bounds. If all bounds are satisfied, the variables form the optimal solution and are updated by the program. If a bound is violated, then the problem is not really zero DD since the bound was not included in the calculation. The practitioner must then decide how to proceed. In general, a single violated constraint is assumed tight and analysis continues on the thus reduced problem.

### **POSCON, MULTICON, and SIGCON**

Routines in C have been written by the author for POSCON, and MULTICON based on Ratliff (1986) and for SIGCON based on Thome (1988). The program will run these unconstrained algorithms on the objective function and ignore any constraints. The solution variables will not be substituted into the constraints automatically. The user can set the variables at values computed by this routine to see which constraints are violated. In some cases all constraints might be loose and the solution found. The routines to perform these calculations were not coded from the original author's code. They were developed and tested by the author based on routines written while taking the GP class at CSM. It should be noted that the name "MULTICON" was also used by Goddard (1987) for a routine that extends Ratliff's subroutine of the same name to handle certain

signomial multi-variable unconstrained problems. GPSOLV implements Ratliff's algorithm without Goddard's extensions.

The program allows the user to exclude constraints from consideration. This can be used to form subset problems that might be easier to solve. If solution to a problem with a subset of constraints satisfies the excluded constraints, then the solution is optimal. In this manner a number of subsets can be checked and the need for individual constraints explored.

### **Systems of Equations**

The program can maintain one or more systems of equations in a C++ class called an AlgSystem. An AlgSystem contains lists pointing to the equations or inequalities that make up the system, the number of variables and their settings and bounds, and matrices and arrays for the GP routines. The quick ability to create and copy systems of equations made coding of functions to deal with separable problems considerably simpler. Many of the operations that can be performed on a single equation such as EXPAND and GATHER can be done on an entire set of equations through similarly named functions in the AlgSystem class. This is an example of the object oriented software's concept of polymorphism.

Only one system of equations has the current focus. For instance when a separable problem is discovered, two new systems of equations are formed with each separable set of variables. The

user can then set the focus to either of the new systems or the original system. Normally, each separable system is explored in turn. If either system can be solved, its solution can be substituted into the original problem.

A copy of the current focus system of equations is made before each potentially modifying command. A list of such systems is kept and used to implement the UNDO command.

## Chapter IV

### COMPUTATIONAL EXPERIENCE

The program has the ability to read in a file as if its contents were typed by the user. Files were prepared to unit test the various routines and overall functioning of the program. The file names and a brief comment of the function tested is included in Appendix A along with reference solutions for comparison. The files contain the optimal solution as published in the reference in comment lines when available. Many of the files contain problems suitable for unit testing the various functions in particular for the zero DD solver and the condensation methods of POSCON and MULTICON. These will not be discussed here as the Appendix entries are sufficient.

Several problems obtained from the literature or constructed for the purpose are presented here to demonstrate the program's functionality. These will be presented with narrative and program output.

#### **MX Missile Problem**

The "MX Missile" problem is taken from Mishins and Sarin (1985) and was used in Woolsey (1992) as an example which could be analyzed using the sign table. Computer output will be put in the

Courier typeface and smaller print so that the full width of the screen can be viewed here. The problem in mathematical form is :

$$\text{Minimize : } T = 39,272,500A + 1,927,500B + 2300BD + 612.13BC$$

$$\text{Subject to : } 100A^{-1} + 2,295B^{-1} \leq 1$$

$$28 \times 10^{12} C^{-3} D^{-1} \leq 1$$

$$100A^{-1} \leq 1$$

$$100B^{-1} \leq 1$$

$$AB^{-1} \leq 1$$

Where A = Number of Missiles in the MX System.

$$\frac{D}{600} \leq 1$$

B = Number of Shelters in the MX System.

$$5000C^{-1} \leq 1$$

C = Uniform Distance between Shelters (Ft.).

$$\frac{C}{10000} \leq 1$$

D = Silo Hardness (psi).

The initial problem looks like this in GPSOLV (with sign table) :

```

GP Algebra Assistant V1.00 9/23/96 . Please use HELP for more information
obj : z = 3.927e+07a + 1.928e+06b + 2300bd + 612.1bc      Z = 4.12e+05
c1 : 100a^-1 + 2295b^-1 <= 1                            2.395e+05<=1
c2 : 2.8e+13c^-3d^-1 <= 1                               2.8e+21<=1
c3 : 100a^-1 <= 1                                       1e+04<=1
c4 : 100b^-1 <= 1                                       1e+04<=1
c5 : ab^-1 <= 1                                         1<=1
c6 : 0.001667d <= 1                                     1.667e-05<=1
c7 : 5000c^-1 <= 1                                     5e+05<=1
c8 : 0.0001c <= 1                                     1e-06<=1

```

	A	B	C	D
obj	+	+	+	+
c1	-	-		
c2			-	-
c3	-			
c4		-		
c5	+	-		
c6				+
c7				-
c8				+

The sign table gives the GP sign of each term for each variable. Each column is one variable with each line representing a term. The equations are interspersed such that they appear at the top of the series of terms representing that equation or inequality.

The “KIRK” command was issued with the screen output in as follows :

```

Check_Bounds on : z = 3.927e+07a + 1.928e+06b + 2300bd + 612.1bc
Check_Bounds on : 100a^-1 + 2295b^-1 <= 1
Check_Bounds on : 2.8e+13c^-3d^-1 <= 1
Check_Bounds on : 100a^-1 <= 1
    New Lower Bound= a = 100 > 1e-300 found for 100a^-1 <= 1
    New Lower Bound forces adjustment of variable to bound.
Check_Bounds on : 100b^-1 <= 1
    New Lower Bound= b = 100 > 1e-300 found for 100b^-1 <= 1
    New Lower Bound forces adjustment of variable to bound.
Check_Bounds on : ab^-1 <= 1
Check_Bounds on : 0.001667d <= 1
    New Upper Bound d = 600.000024<1e+300 found for 0.001667d <= 1
Check_Bounds on : 5000c^-1 <= 1
    New Lower Bound= c = 5000 > 1e-300 found for 5000c^-1 <= 1
    New Lower Bound forces adjustment of variable to bound.
Check_Bounds on : 0.0001c <= 1
    New Upper Bound c = 10000<1e+300 found for 0.0001c <= 1
Removing Bounds equations : c8 c7 c6 c4 c3
Delete variable after substitution : d = 2.8e+13c^-3

obj : z = 3.927e+07a + 1.928e+06b + 6.44e+16bc^-3 + 612.1bcZ = 1.656e+10
c1 ! 100a^-1 + 2295b^-1 <= 1                2<=1                False
c5 : ab^-1 <= 1                             0.04357<=1            True
# Vars= 3   a =      100   b =      2295   c =      5000
Lower Bnd  a >      100   b >      2295   c >      5000
Upper Bnd  a < 1e+300   b < 1e+300   c < 10000
Substitutions : d = 2.8e+13c^-3
G.P. Form = O.K.   # Terms =7   D.D. =3

Separable obj function : 6.44e+16c^-3 + 612.1c
Poscon Tec=3.440117e+06 at a-1=4214.934833  2 iterations
in 2 iterations

c is not within the bounds [5000,10000]
Expression is posynomial in single variable. Substitute at Bound c=5000
Delete variable after substitution : c = 5000

```

( 1 )

( 2 )

( 3 )

( 4 )

( 5 )

GP Algebra Assistant V1.00 9/23/96 . Please use HELP for more information

```

obj : z = 3.927e+07a + 5.503e+06b          Z = 1.656e+10
c1 ! 100a^-1 + 2295b^-1 <= 1             2<=1          False
c5 : ab^-1 <= 1                          0.04357<=1     True
# Vars= 2   a =      100   b =      2295
Lower Bnd  a >      100   b >      2295
Upper Bnd  a <  1e+300   b <  1e+300
Substitutions : d = 224 c = 5000
G.P. Form = O.K.   # Terms =5   D.D. =2

```

**( 6 )**

The numbers to the right in parenthesis will be referred to in the following paragraph.

The first thing Kirk's algorithm does is convert all of the single variable constraints (C3, C4, C6, C7, and C8) into simple bounds. That is variables  $A$  and  $B$  are lower bounded by 100,  $D$  is upper bounded by 600, and  $C$  is lower bounded by 5000 and upper bounded by 10000. It then deletes these constraints (1). The subroutine GPBOUND then runs and discovers that variable  $B$  has a new lower bound at 2295 from constraint 1. Similarly, it finds a new lower bound for  $D$  at 28 from constraint 2. Looking down the sign table column for variable  $D$  it can be seen that only constraint two (C2) has a negative sign. This implies that constraint two must be tight. At (2) the sign table analysis portion of Kirk's algorithm has determined that constraint 2 is tight and it has solved for  $D$  in terms of  $C$  and substituted everywhere in the problem. At (3) the reduced problem can be seen with the revised bounds. At (4) it is recognized that the problem can be separated into a problem with variable  $C$  and another with variables  $A$  and  $B$ . The new problem containing only

variable  $C$  has no constraints, so POSCON is used to solve for  $C$  in the objective function. At (5) the program observes that the value of  $C$  does not lie within bounds on  $C$ . Since  $C$  is a posynomial it chooses the bound with the lower value of 5000 and substitutes this value for  $C$ . After (5) the entire Kirk algorithm is run again and finds no changes that can be made. The revised system of equations is output at (6).

At this point Woolsey (1992) simply observed that the problem was of a previously solved form (Rasey and Allen 1982). Kirk's algorithm has reduced a 4 variable, 8 constraint, 8 DD problem to a 2 variable, 2 constraint, 2 DD problem of much simpler form. GPSOLV does not look for special forms of problems so this is all of the progress that could be made.

### Separable Programs

The following problem was constructed to test the ability of Kirk's algorithm subroutines to recognize and process separable problems.

$$\begin{aligned} \text{Minimize : } & z = 2a^2b^2 + c^2d^2 \\ \text{Subject to : } & 0.01a^{-1} + 0.01b^{-2} \leq 1 \\ & 0.1c^{-1} + 0.1d^{-2} \leq 1 \\ & e + d \leq 1 \end{aligned}$$

Clearly this system is separable into :

$$\begin{array}{ll} \text{Minimize : } z = 2a^2b^2 & \text{Minimize : } z = c^2d^2 \\ \text{Subject to : } 0.01a^{-1} + 0.01b^{-2} \leq 1 & \text{Subject to : } 0.1c^{-1} + 0.1d^{-2} \leq 1 \\ & e + d \leq 1 \end{array}$$

When GPSOLV separates a system into two parts it creates two completely new systems in which to store them. The program's 'select' command allows the user to choose which system is the current focus of all commands. The Kirk command was executed and the following output resulted.

Separation found. Vars in first set : a b

Separate system 1

```
obj : z = 2a^2b^2                Z = 2
c1 ! a^-1 + b^-2 <= 1           2<=1          False

# Vars= 2   a =          1   b =          1
Lower Bnd  a >          1   b >          1
Upper Bnd  a < 1e+300   b < 1e+300
G.P. Form = O.K.   # Terms =3   D.D. =0
```

Separate system 2

```
obj : z = c^2d^2                Z = 0.001235
c2 ! 0.1c^-1 + 0.1d^-2 <= 1    1.9<=1          False
c3 : e + d <= 1               0.3262<=1       True

# Vars= 3   c = 0.111111   d = 0.316228   e =          0.01
Lower Bnd  c > 0.111111   d > 0.316228   e > 1e-300
Upper Bnd  c < 1e+300     d <          1   e < 0.683772
G.P. Form = O.K.   # Terms =5   D.D. =1
```

Notice that the command also computed bounds for the variables based on the algorithms described in the MX missile

example. The variable  $e$  is not linked to the objective function or to the other constraint and hence can be set to whatever level minimizes its impact on the system. The first set of equations is zero DD and can be solved directly. The output for the solve command is :

```
Solve 0 d.d. function
Zero DD Solve # vars=2 # terms=3 DD=0
  1    0    0 =    1
  2   -1    0 =    0
  2    0   -2 =    0
Successful Gauss Jordan with sig0=1 The deltas are
  1    2    1
TEC= 13.5
  2    2 = 1.909543 Log of Rule III for term 0
 -1    0 = -0.405465 Log of Rule IV for term 1
  0   -2 = -1.098612 Log of Rule IV for term 2
Successful Gauss Jordan The X's are (*=Out of bounds values)
a=  1.5 b=1.732051
```

GP Algebra Assistant V1.00 9/23/96 . Please use HELP for more information

```
obj : z = 2a^2b^2                Z = 13.5
c1 : a^-1 + b^-2 <= 1           1<=1          True

# Vars= 2   a =      1.5   b =  1.732051
Lower Bnd  a >  1e-300   b >  1e-300
Upper Bnd  a <  1e+300   b <  1e+300
G.P. Form = O.K.   # Terms =3   D.D. =0

Number of systems : 3 This is System # 2
```

The second system is solved by substituting  $e=0$  and then trying the zero DD set ignoring constraint 2. This solution is :

```

Solve 0 d.d. function
Zero DD Solve # vars=2 # terms=3 DD=0
  1   0   0 = 1
  2  -1   0 = 0
  2   0  -2 = 0
Successful Gauss Jordan with sig0=1 The deltas are
  1   2   1
TEC= 0.00675
  2   2 = -4.998213 Log of Rule III for term 0
 -1   0 = 1.89712 Log of Rule IV for term 1
  0  -2 = 1.203973 Log of Rule IV for term 2
Successful Gauss Jordan The X's are (*=Out of bounds values)
c= 0.15 d=0.547723

GP Algebra Assistant V1.00 9/23/96 . Please use HELP for more information

```

```

obj : z = c^2d^2                                Z = 0.00675
c2 ! 0.1c^-1 + 0.1d^-2 <= 1                    1<=1          Tight

# Vars= 2   c = 0.15   d = 0.547723
Lower Bnd  c > 0.111111   d > 0.316228
Upper Bnd  c < 1e+300    d < 1
Substitutions : e = 0
G.P. Form = O.K.   # Terms =3   D.D. =0

Number of systems : 3 This is System # 3

```

Since the value for  $d$  at optimality does not violate constraint three, then this is the optimal solution to the problem. The program has the ability to substitute solutions for primal variables found in any system into one or more of the other systems. This is useful when one of the subsets is solved. It can be conveniently substituted into the original equations for further analysis.

### Raliff's Degenerate Problem

Ratliff (1986 pg. 44) reported a problem which, though a posynomial, was not solvable by MULTICON. The problem was constructed to have a exponent matrix which must have at least one

negative dual variable. The problem is :

$$\text{Minimize : } t = n^{-1.167} d^{-1} l^{-1.333} + p^{0.8} n^{-0.2} l^{-1} + ndl + lp^{-4.8} n^{-1.8} + n^{-1} p^{-1} l^{-1}$$

In fact this problem is zero DD and the GPSOLV output for its solution is :

```
Solve 0 d.d. function
Zero DD Solve # vars=4 # terms=5 DD=0
  1    1    1    1    1 =  1
 -1    0    1    0    0 =  0
-1.33333  -1    1    1    -1 =  0
-1.166667 -0.2    1 -1.8    -1 =  0
  0    0.8    0 -4.8    -1 =  0
Successful Gauss Jordan with sig0=1 The deltas are
0.545458 0.045442 0.545458 0.04545 -0.181809
1 negative deltas! problem may be unbounded

GP Algebra Assistant V1.00 9/23/96 . Please use HELP for more information

obj : t = n^-1.167d^-1l^-1.333 + p^0.8n^-0.2l^-1 + ndl      Z = 1.585e+11
      + lp^-4.8n^-1.8 + n^-1p^-1l^-1

# Vars= 4   d =      0.01   l =      0.01   n =      0.01   p =      0.01
Lower Bnd  d >  1e-300   l >  1e-300   n >  1e-300   p >  1e-300
Upper Bnd  d <  1e+300   l <  1e+300   n <  1e+300   p <  1e+300
G.P. Form = O.K.   # Terms =5   D.D. =0
```

The zero DD solver correctly diagnosed the meaning of this matrix of exponents. When GPSOLV's version of the Ratliff algorithm was run, it was discovered that several variables could be traded off against each other with some growing without bound and others approaching zero. The last few iterations of output were :

```
993 Tec=2.7507434e-07 var= 1.62934e-35 2.49407e+54 3.64488e-27 1.31895e+27
994 Tec=2.7086722e-07 var= 1.50848e-35 2.82133e+54 3.42703e-27 1.40281e+27
995 Tec=2.6672445e-07 var= 1.39658e-35 3.19153e+54 3.2222e-27 1.492e+27
996 Tec=2.6264504e-07 var= 1.29298e-35 3.6103e+54 3.02962e-27 1.58686e+27
997 Tec=2.5862802e-07 var= 1.19707e-35 4.08402e+54 2.84854e-27 1.68775e+27
```

```

998 Tec=2.5467244e-07 var= 1.10827e-35 4.6199e+54 2.67829e-27 1.79506e+27
999 Tec=2.5077736e-07 var= 1.02606e-35 5.2261e+54 2.51822e-27 1.90919e+27
1000 Tec=2.4694186e-07 var= 9.49946e-36 5.91184e+54 2.36771e-27 2.03058e+27
1001 Tec=2.4316501e-07 var= 8.79479e-36 6.68755e+54 2.22619e-27 2.15968e+27
Objective Function* = 2.43165e-07 d=8.794793e-36 l=6.687552e+54 n=2.226194e-27
p=2.159684e+27
in 11172 iterations of POSCON.

```

GP Algebra Assistant V1.00 9/23/96 . Please use HELP for more information

```

obj : t = n^-1.167d^-11^-1.333 + p^0.8n^-0.21^-1 + ndl      Z = 1.585e+11
      + lp^-4.8n^-1.8 + n^-1p^-11^-1

```

```

# Vars= 4   d =      0.01   l =      0.01   n =      0.01   p =      0.01
Lower Bnd  d >    1e-300   l >    1e-300   n >    1e-300   p >    1e-300
Upper Bnd  d <    1e+300   l <    1e+300   n <    1e+300   p <    1e+300
G.P. Form = O.K.   # Terms =5   D.D. =0

```

Clearly, variables  $d$  and  $n$  can be traded off against variables  $l$  and  $p$ . This trading off resulted in an objective function that can be made arbitrarily close to zero, but which does not have a finite stationary point. The GPSOLV version of MULTICON allows only 1000 iterations of the MULTICON loop and this was exceeded in this case.

## Chapter 5

### FUTURE WORK

During development of this thesis many 'features' that might be useful presented themselves especially when use as a teaching or research tool was considered. They have been broken roughly into areas of interest though many overlap.

#### **Algebraic Manipulations**

The program does only a small set of Algebraic manipulations well. This set was chosen with the idea that manipulations needed for solving GP type problems were all that were needed. The following features might be useful.

- 1) The ability to handle transcendental expressions.
- 2) The ability to substitute for complex Algebraics (i.e.  $x+y=2z+4w$ ). The current program cannot substitute for  $x+y$  everywhere. This is necessary to easily implement the engineer's approximation (Woolsey 1992) where variables are substituted for an expression and a new "must have" constraint added.
- 3) Allow multiplication or division by multinomials.

- 4) All of the simplifications or change of node type are now done in subroutine GATHER. While sufficient for the base code, this routine would need a more rigorous implementation to optimize efficiency and to handle any of the above enhancements.
- 5) Automatic conversion of problems to the “all constraints form” of Wessels (1989, pg. 19).

An alternate approach would be to find a symbolic manipulation program which would be used for all manipulations. Only when the problem was simplified to generalized polynomials would it be transferred to GPSOLV. At that point the current set of routines should be sufficient for internal algorithm needs. Hope on this horizon can be seen in the current development of object oriented distributed computing. In this way an object consisting of a system of equations from a symbolic manipulation program might be sent to GPSOLV invoking it to operate upon it. At this point the divisions between traditional programs has made the interchange of data in such a manner difficult and hard to automate without cooperation between the programs' authors.

### **User Interface**

For a student more interested in the computer science aspects of the program, the program could lend itself to a Graphical User

Interface similar to the one used in Mathematica (1992) and Theorist(1993). This would replace the command line interface with something more intuitive. The system of equations could also be kept in a separate window and in a more mathematical form. Better fonts, the use of superscripts for powers, etc. would increase the user's ability to look at the system of equations. This feature would make the program more platform dependent.

Implementation of commands based on mouse 'gestures' would make the interface more appealing. This enhancement would make the program much more platform dependent.

Additions of shortcuts such as "DIVIDE RHS" to cause division of the entire equation by the right hand side would save the user from typing the right hand side and allow greater accuracy in handling the constants.

Mathematica has an interface to communicate with coprocessing tasks called "Math Link". On initial inspection this interface appeared to be insufficient for the purposes of GPSOLV. However, a new version of Mathematica (release 3.0) is due out about the time of this thesis. It may be that the ability of this new revision would allow a more direct coupling between GPSOLV and Mathematica.

### **Kirk's Algorithm**

Step 4 of Kirk's algorithm (1988 pg. 35) calls for the

elimination of redundant constraints which are a linear combination of other constraints. This involves treating each term which has a unique combination of variables and/or exponents of variables as a 'signature' variable. The system of equations can then be viewed as a set of linear equations in the signature variables. If such a matrix has rows which are not linearly independent, then one or more constraints can be dropped. It is less clear how to determine which constraint is likely to be dropped or whether one or more constraints might be replaced by some linear combination of them.

Step 5 of Kirk's algorithm (1988 pg. 37) calls for solving equality constraints in the unlikely circumstance of equal numbers of variables and equations using a code designed for such problems.

Step 7 of Kirk's algorithm (1988 pg 41) looks for a problem which consists of all monomial generalized polynomials. This type of problem can be solved by logarithmic transformation and linear programming.

### **Geometric Programming**

Symbolic analysis might be used to look at the problem in its given form and decide how to convert it to appropriate GP form. For instance, deciding what term to move to the right hand side and then dividing by it so as to make the constraint a posynomial.

The program treats bounds as side constraints that are only checked when a solution to the remaining problem is found. A better

method of tracking and integrating bounds type constraints might be possible.

Implementation of the unconstrained signomial algorithm first proposed by Dolan (1994) is possible. This would complement the POSCON, MULTICON, and SIGCON routines currently implemented. Note that much of the Dolan FORTRAN code deals with input of the model. GPSOLV would have simplified this part of the work. Similarly the algorithm of Goddard (1987) has not been included for solving suitable signomials.

An automated method of selecting subsets for constraints that might be solved and checked against other constraints could be developed. Since some constraints might be "must have" and the need for the problem to be zero DD would reduce the number of combinations possible to a reasonable number.

Wessels (1986) proposed an algorithm which reduced any problem to a series of zero DD problems that could be iterative and used to solve a higher DD problem. The problem was in choosing the right terms to condense based on the advanced sign table and the columns of the dual variable matrix. With the data structures available within GPSOLV, a convenient platform for investigating the method exists.

Better analysis when substitution decisions become available to insure that the substitutions occur to integral power or otherwise to advantage. Substitution of multinomials for variables to non-integral

powers result in a modified problem which cannot be simplified to generalized polynomial form. Example : substituting  $x=y+z$  into  $x^{3/2}+y \leq 1$  results in  $(y+z)^{3/2}+y \leq 1$  which cannot be simplified to generalized polynomial form using only the simple manipulations of GPSOLV.

Implementation of the engineer's approximation (Woolsey, 1992) to handle cases like the above by defining a new variable equal to the contents of the parenthesis and adding a new 'must have' constraint.

### **Additional Solution Methods and Features**

With the addition of a differential operation the Hessian matrix could be formed and evaluation at any proposed stationary point. This would involve building the Lagrangian algebraic from the objective function and constraints and new Lagrangian variables. The Lagrangian would then be simplified to a generalized polynomial and the various partial derivatives taken and the resulting Algebraics stored as a matrix of points to the Algebraics for both the first and second order Karush-Kuhn-Tucker conditions. At a stationary point the Evaluate function could be called on each of these to find the value of the Hessian variable. Linear algebra could then be used to evaluate the definiteness of the resulting matrix. This

process can be performed manually in Mathematica and similar programs, but with considerable effort. Ratliff's method has been shown to converge to maximums and saddle points. Checking the nature of the stationary point would allow any algorithm to adapt and go on looking for minimums.

The bounds adjusting portion of Kirk's algorithm will occasionally discover variables that can be traded off against each other. This results in an infinite loop with each variable being adjusted by a small amount in each cycle. This cycle is broken by limiting the total number of changes that the bounds can take on during one invocation of the algorithm. A better algorithm for dealing with this case would be advantageous.

Problems that are reduced by GPSOLV could be exported in a format usable by other nonlinear optimization codes. Alternatively, additional nonlinear algorithms could be made subroutines to GPSOLV. This would allow comparison of run times for the reduced and full problems and to compare algorithms in the same program environment. Timing routines would need to be added to allow accurate comparison of algorithms.

While GPSOLV is designed to handle generalized polynomials, it can be used to manipulated systems of linear equations. In this manner it is possible to add a module that would solve linear programming or integer programming problems. These types of problems are generally solved by manipulating a matrix of coefficients

and a vector containing the right hand sides. Such data structures could easily be constructed from the algebraic representation of GPSOLV.

**REFERENCES CITED**

- Beightler, C.S. and D.T. Phillips, 1976, Applied Geometric Programming, New York: Wiley and Sons.
- Dolan, W. T. 1995. "A Geometric Programming Algorithm for Solving a Class of Unconstrained Signomials", Master's Thesis, Colorado School of Mines.
- Duffin, R.J., E.L. Peterson, C. Zener. 1988. Geometric Programming, New York: Wiley and Sons.
- Flewelling, R.T. 1994. "The Applicability of Geometric Programming Analysis to Aircraft Design Benchmark Problems". Master's Thesis. Colorado School of Mines.
- Goddard, N.W. 1987. "An Algorithm for the solution of a Class of Economic Models". Master's Thesis. Colorado School of Mines.
- Jackson, J.A. 1994. "A Mathematical Experiment in Dual Geometric Programming". Ph. D. thesis. Colorado School of Mines.
- Kirk, J.B. 1988. "A Geometric Programming Based Algorithm for Preprocessing Nonlinear Signomial Optimization Problems". Ph. D. Thesis, Colorado School of Mines.
- Passy, U., and D.J. Wilde. 1971. Generalized Polynomial Optimization. SIAM Journal of Applied Mathematics, vol. 15:1345-1356.
- Prescience Corporation. 1993. The Student Edition of Theorist. Boston, Mass. : PWS Publishing Company.

- Ratliff, R.M. 1986. "A Generalized Condensation Algorithm for the Solution of Unconstrained Balanced Posynomial Problems using Geometric Programming." Master's Thesis. Colorado School of Mines.
- Redfern, D. 1994. The Maple Handbook: Maple V release 3, New York: Springer - Verlag.
- Reklaitis, G.V., A. Ravindran, and K.M. Ragsdell. 1983. Engineering Optimization. New York: Wiley and Sons.
- Schweyer, H. 1955. Process Engineering Economics. New York: McGraw-Hill.
- Shammas, N.C. 1995. C/C++ Mathematical Algorithms for Scientists and Engineers. New York: McGraw-Hill, Inc.
- Sherwood, T.K. 1970. A Course in Process Design. M.I.T. Press, Massachusetts Institute of Technology, Cambridge, MA.
- Thome, J. 1988. "An Algorithm for Solving a Class of Nonlinear Unconstrained Signomial Economic Models Using the Greening Technique." Ph. D. thesis, Colorado School of Mines.
- Vankaya, B.B., V.A. Tischler, and S.M. Pitrof. 1993. "Benchmarking in Structural Optimization". An unpublished internal document of the Wright Labs; Wright-Patterson AFB, OH.
- Wessels, G.J. 1989. "A Geometric Programming Algorithm for Solving a Class of Nonlinear, Signomial Optimization Problems." Ph.D. thesis, Colorado School of Mines.
- Wilde, D.J. 1978. Globally Optimal Design. New York: Wiley and Sons.
- Wolfram, S. 1992. Mathematica Reference Guide. Reading, Mass.

Addison-Wesley.

Woolsey, R.E.D. 1992. Lecture notes for MAGN 558 at the Colorado School of Mines.

Woolsey, R.E.D. 1992. Engineering Design Optimization with Geometric Programming. Vol 1 in the Useful Engineering Series. Draft used in MAGN558 at the Colorado School of Mines.

Woolsey, R.E.D. and H.S. Swanson. 1975. Operations Research for Immediate Application: A Quick and Dirty Manual. New York: Harper and Row.

## APPENDIX

The following is a list of the test problems to be found later in this appendix. A brief description of the problem and its reference source is given. The remainder of the appendix consists of these test cases with the exception of `mx.alg` and `sep.alg` as these problems were covered in detail in the main body. The problems are described and the reference solution given along with the output of GPSOLV.

<b>File</b>	<b>Description</b>
<code>afit6.alg</code>	The AFIT problem #6 from Flewelling's thesis. This problem was found highly suitable for Kirk's algorithm as it has very strong bounds.
<code>bppos1.alg</code>	Beightler and Phillips, pg. 193, Posynomial zero DD with posynomial objective function and 2 posynomial constraints. Used to test zero DD solver.
<code>bpsig1.alg</code>	Beightler and Phillips, pg. 189, signomial zero DD with posynomial objective function & two signomial constraints. Used to test zero DD solver.
<code>bpsig2.alg</code>	Beightler and Phillips, pg. 205, example 5.3 Four variable signomial objective function, zero DD, 1 posynomial constraint. Used to test zero DD solver.

bpsig2b.alg	Modified Beightler and Phillips, pg. 205, ex 5.3 but unbounded in X4. Used to test zero DD solver for ill formed problems.
bpsig54.alg	Beightler and Phillips, pg 208, Example 5.4 negative one term objective function with one posynomial constraint. Used to test zero DD solver.
gp1.alg	Simple 3 variable zero DD problem with one constraint. Used to test zero DD solver.
gp2.alg	Simple 3 variable zero DD problem with one constraint. Used to test zero DD solver.
gp26.alg	Chemical plant unconstrained bounding from Woolsey (1992) pg. 2-6. Used to test Ratliff method.
mx.alg	MX missile problem from Woolsey (1992) page 2-21 to 2-26. Test case for effectiveness of Kirk's algorithm.
parse.alg	A series of complex expressions to test Parse function and storage form of Algebraics.
poscon.alg	Simple POSCON model (could be EOQ type model) source unknown. Used to test POSCON routine.
Redundant.alg	Excites modes in Kirk Redundant constraint routine reduces to zero DD.
rat1.alg	Ratliff (1986), pg. 36, (mining problem of Taylor)
rat2.alg	Ratliff (1986), pg. 37, Chemical plant problem of Beightler and Phillips

rat3.alg	Ratliff (1986), pg. 39, Batch size problem of Schweyer
rat4.alg	Ratliff (1986), pg. 42, (note Cost appears to have typographical error in Ratliff).
ratdegen.alg	Ratliff (1986), pg. 44, degenerative case (trade of variables to $t=0$ ).
sep.alg	Tests the separable algorithm. It ends up solvable in parts.

File `afit6.alg` Reference : The problem is taken from Flewelling (1994) who was using an unpublished document from Venkaya, Tischler, and Pitrof (1993) listing A.F.I.T.'s benchmarking suite. Flewelling (1994) referred to this as problem 6.

GP Algebra Assistant V1.00 9/23/96 . Please use HELP for more information

```

obj : z = 2.618x2^2x1x3^2
      + 11.73x2^2x1x3 -33.85x2^2x1 -1.508x1x6^2 -1.508x1x7^2 + 7.477x6^3 + 7.477x7^3
+ 0.7854x4x6^2
+ 0.7854x5x7^2
g1 : 27x1^-1x2^-1x3^-1 <= 1
g2 : 397.5x1^-1x2^-2x3^-2 <= 1
g3 : 1.93x2^-1x3^-1x4^3x6^-4 <= 1
g4 : 1.93x2^-1x3^-1x5^3x7^-4 <= 1
g5 : 45.87x4^2x2^-2x3^-2x6^-6 + 1397x6^-6 <= 1
g6 : 76.82x4^2x2^-2x3^-2x7^-6 + 2.18e+04x7^-6 <= 1
g7 : 0.025x2x3 <= 1
g8 : 5x1^-1x2 <= 1
g9 : 0.08333x1x2^-1 <= 1
g10 : 2.6x1^-1 <= 1
g11 : 0.2778x1 <= 1
g12 : 0.7x2^-1 <= 1
g13 : 1.25x2 <= 1
g14 : 17x3^-1 <= 1
g15 : 0.03571x3 <= 1
g16 : 7.3x4^-1 <= 1
g17 : 0.1205x4 <= 1
g18 : 7.3x5^-1 <= 1
g19 : 0.1205x5 <= 1
g20 : 2.9x6^-1 <= 1
g21 : 0.2564x6 <= 1
g22 : 5x7^-1 <= 1
g23 : 0.1818x7 <= 1
g24 : 1.5x4^-1x6 + 1.9x4^-1 <= 1
g25 : 1.1x5^-1x7 + 1.9x5^-1 <= 1
      Z = -2.022e-05
      2.7e+07<=1      False
      3.975e+12<=1    False
      1.93e+06<=1     False
      1.93e+06<=1     False
      4.601e+17<=1    False
      7.9e+17<=1      False
      2.5e-06<=1      True
      5<=1             False
      0.08333<=1      True
      260<=1           False
      0.002778<=1     True
      70<=1            False
      0.0125<=1       True
      1700<=1          False
      0.0003571<=1    True
      730<=1           False
      0.001205<=1     True
      730<=1           False
      0.001205<=1     True
      290<=1           False
      0.002564<=1     True
      500<=1           False
      0.001818<=1     True
      191.5<=1         False
      191.1<=1         False

```

G.P. Form = O.K. # Terms =38 D.D. =30

After running Kirk's algorithm this problem was reduced to :

GP Algebra Assistant V1.00 9/23/96 . Please use HELP for more information

```

obj : z = 2.618x2^2x1x3^2          Z = 2993
      + 11.73x2^2x1x3 -33.85x2^2x1 -1.508x1x6^2 -1.508x1x7^2 + 7.477x6^3 + 7.477x7^3
      + 0.7854x4x6^2 + 0.7854x5x7^2
g1 : 27x1^-1x2^-1x3^-1 <= 1      0.6483<=1      True
g2 : 397.5x1^-1x2^-2x3^-2 <= 1    0.802<=1      True
g3 : 1.93x2^-1x3^-1x4^3x6^-4 <= 1  0.5035<=1    True
g5 : 45.87x4^2x2^-2x3^-2x6^-6 + 1397x6^-6 <= 1  1.008<=1    False
g6 : 76.82x4^2x2^-2x3^-2x7^-6 + 2.18e+04x7^-6 <= 1  1.001<=1    False
g7 : 0.025x2x3 <= 1              0.2975<=1    True
g8 : 5x1^-1x2 <= 1              1<=1          True
g9 : 0.08333x1x2^-1 <= 1         0.4167<=1    True
g24 ! 1.5x4^-1x6 + 1.9x4^-1 <= 1  0.9478<=1    True
g25 ! 1.1x5^-1x7 + 1.9x5^-1 <= 1  1<=1          True

# Vars= 7 x1 =3.5 x2 = 0.7 x3 = 17 x4 = 7.3 x5 = 7.714333 x6 = 3.345758 x7 = 5.285757
Lower Bnd x1 >3.5 x2 > 0.7 x3 > 17 x4 > 7.3 x5 > 7.714333 x6 > 3.345758 x7 > 5.285757
Upper Bnd x1 <3.6 x2 < 0.72 x3 < 28 x4 < 8.3 x5 < 8.3 x6 < 3.9 x7 < 5.5
G.P. Form = O.K. # Terms =23 D.D. =15

```

Flewelling had  $z^*=2994.61$  with the  $x_5^*=7.7154$ ,  $x_6^*=3.35108$ ,  
and  $x_7^*=5.2867$  being the only differences. The program reduced  
the degree of difficulty from 30 to 15 and its current set of variables  
based on bounds is actually quite near the optimal solution.

File : bppos1.alg Reference : Part of Example 5.2, Beightler and Phillips (1976), pg. 193. This is a posynomial case.

$$\begin{aligned} \text{Minimize : } & y = 452x_1^{-1}x_2^{-1} \\ \text{Subject to : } & 11000x_1^{-1.52}x_2 \leq 1 \\ & 0.0193x_1x_2^{0.83} \leq 1 \end{aligned}$$

Reference Solution :  $y^*=11.184$   $x_1^*=174.28$   $x_2^*=0.232$

```
Solve 0 d.d. function
Zero DD Solve # vars=2 # terms=3 DD=0
  1   0   0 =  1
-1 -1.52  1 =  0
-1   1 0.83 =  0
Successful Gauss Jordan with sig0=1 The deltas are
  1 0.075168 1.114255
TEC= 11.183942
-1   -1 = -3.699203 Log of Rule III for term 0
-1.52  1 = -9.305651 Log of Rule IV for term 1
  1 0.83 = 3.94765 Log of Rule IV for term 2
Successful Gauss Jordan The X's are (*=Out of bounds values)
x1=174.278786 x2=0.231899
```

GP Algebra Assistant V1.00 9/23/96 . Please use HELP for more information

```
obj : y = 452x1^-1x2^-1                Z = 11.18
c1 : 1.1e+04x1^-1.52x2 <= 1            1<=1          Tight
c2 : 0.0193x1x2^0.83 <= 1             1<=1          Tight

# Vars= 2   x1 = 174.278786   x2 = 0.231899
Lower End   x1 > 1e-300      x2 > 1e-300
Upper End   x1 < 1e+300      x2 < 1e+300
G.P. Form = O.K.   # Terms =3   D.D. =0
```

File : bpsig1.alg Reference : Beightler and Phillips (1976), pg. 188.  
 This is a signomial in the constraints with a  
 posynomial objective function value. Signum then  
 must be plus one.

Minimize :  $y = x_1$

Subject to :  $5x_1^{-1}x_2 - x_1^{-1}x_3^2x_4^4 \leq 1$

$-2.5x_2x_3^{-2} + 1.5x_3^{-1}x_4 \leq 1$

Reference Solution :  $y^* = 0.796$   $x_1^* = 0.796$   $x_2^* = 0.239$   $x_3^* = 0.446$   
 $x_4^* = 1.19$

Solve 0 d.d. function

Zero DD Solve # vars=4 # terms=5 DD=0

1	0	0	0	0	=	1
1	-1	1	-0	0	=	0
0	1	-0	-1	0	=	0
0	0	-2	2	-1	=	0
0	0	-4	-0	1	=	0

Successful Gauss Jordan with sig0=1 The deltas are

1	1.5	0.5	1.5	2
---	-----	-----	-----	---

TEC= 0.795495

1	0	0	0	=	-0.228791	Log of Rule III for term 0
-1	1	0	0	=	-1.203973	Log of Rule IV for term 1
-1	0	2	4	=	-0.693147	Log of Rule IV for term 2
0	1	-2	0	=	0.182322	Log of Rule IV for term 3
0	0	-1	1	=	0.980829	Log of Rule IV for term 4

Successful Gauss Jordan The X's are (\*=Out of bounds values)

$x_1 = 0.795495$   $x_2 = 0.238649$   $x_3 = 0.445953$   $x_4 = 1.189207$

GP Algebra Assistant V1.00 9/23/96 . Please use HELP for more information

obj :  $y = x_1$

Z = 0.7955

c1 :  $5x_1^{-1}x_2 - 1x_1^{-1}x_3^2x_4^4 \leq 1$

1<=1 True

c2 :  $-2.5x_2x_3^{-2} + 1.5x_3^{-1}x_4 \leq 1$

1<=1 Tight

# Vars= 4  $x_1 = 0.795495$   $x_2 = 0.238649$   $x_3 = 0.445953$   $x_4 = 1.189207$

Lower Bnd  $x_1 > 1e-300$   $x_2 > 1e-300$   $x_3 > 1e-300$   $x_4 > 1e-300$

Upper Bnd  $x_1 < 1e+300$   $x_2 < 1e+300$   $x_3 < 1e+300$   $x_4 < 1e+300$

G.P. Form = O.K. # Terms =5 D.D. =0

File : bpsig2.alg Reference : Beightler and Phillips (1976), pg. 205  
 Example 5.3. This is a signomial objective function with a positive value (signum =1) and posynomial constraints.

$$\text{Minimize : } y = -2x_1x_2x_3^4x_4^{-1} + x_2^{-1}x_3^{-1} + 5x_1^{0.5}x_4$$

$$\text{Subject to : } x_4^{-1/2} + x_2^{1/3}x_3x_4^{-1/2} \leq 1$$

Reference Solution :  $y^*=38.322$   $x_1^*=0.4083$   $x_2^*=.00422$   $x_3^*=18.558$   
 $x_4^*=16$ .

Solve 0 d.d. function

Zero DD Solve # vars=4 # terms=5 DD=0

```
-1 1 1 0 0 = 1
-1 0 0.5 0 0 = 0
-1 -1 0 0 0.33333 = 0
-4 -1 0 0 1 = 0
1 0 1 -0.5 -0.5 = 0
```

Successful Gauss Jordan with sig0=1 The deltas are

0.666677 0.333323 1.333353 1.00003 3.00003

TEC= 38.303156

```
1 1 4 -1 = 2.546935 Log of Rule III for term 0
0 -1 -1 0 = 2.54689 Log of Rule III for term 1
0.5 0 0 1 = 2.323791 Log of Rule III for term 2
0 0 0 -0.5 = -1.386279 Log of Rule IV for term 3
0 0.33333 1 -0.5 = -0.287687 Log of Rule IV for term 4
```

Successful Gauss Jordan The X's are (\*=Out of bounds values)

x1=0.407573 x2=0.004219 x3=18.56552 x4=15.99952

GP Algebra Assistant V1.00 9/23/96 . Please use HELP for more information

```
obj : y = -2x1x2x3^4x4^-1 + x2^-1x3^-1 + 5x1^0.5x4          Z = 38.3
c1 : x4^-0.5 + x2^0.3333x3x4^-0.5 <= 1                    1<=1          Tight
```

```
# Vars= 4  x1 = 0.407573  x2 = 0.004219  x3 = 18.56552  x4 = 15.99952
Lower Bnd  x1 > 1e-300  x2 > 1e-300  x3 > 1e-300  x4 > 1e-300
Upper Bnd  x1 < 1e+300  x2 < 1e+300  x3 < 1e+300  x4 < 1e+300
G.P. Form = O.K.  # Terms =5  D.D. =0
```

File : bpsig2b.alg Reference : Beightler and Phillips (1976), pg. 205  
 exercise 5.3 modified to  $x_4$  unbounded. The dual  
 variables should be of mixed sign indicating a  
 problem that is ill formed.

$$\text{Minimize : } y = -2x_1x_2x_3^4x_4^{-1} + x_2^{-1}x_3^{-1} + 5x_1^{0.5}x_4$$

$$\text{Subject to : } x_4^{1/2} - x_2^{1/3}x_3 \leq 1$$

The point of this problem was to manufacture an unbounded case to  
 test the programs ability to recognize such cases.

Solve 0 d.d. function

Zero DD Solve # vars=4 # terms=5 DD=0

```

-1   1   1   0   0 = 1
-1   0  0.5  0  -0 = 0
-1  -1   0   0 -0.33333 = 0
-4  -1   0   0  -1 = 0
 1   0   1  0.5  -0 = 0

```

Successful Gauss Jordan with sig0=1 The deltas are

0.666677 0.333323 1.333353 -4.00006 -3.00003

2 negative deltas! problem may be unbounded

GP Algebra Assistant V1.00 9/23/96 . Please use HELP for more information

```

obj : y = -2x1x2x3^4x4^-1 + x2^-1x3^-1 + 5x1^0.5x4      Z = 1e+04
c1 : x4^0.5 -1x2^0.3333x3 <= 1                          0.09785<=1      True

```

```

# Vars= 4   x1 =      0.01   x2 =      0.01   x3 =      0.01   x4 =      0.01
Lower Bnd  x1 >  1e-300   x2 >  1e-300   x3 >  1e-300   x4 >  1e-300
Upper Bnd  x1 <  1e+300   x2 <  1e+300   x3 <  1e+300   x4 <  1e+300
G.P. Form = O.K.   # Terms =5   D.D. =0

```

File : bpsig54.alg Reference : Beightler and Phillips (1976), pg. 208, Example 5.4. This is a signomial objective function which is negative (signum =-1) with a posynomial constraint. All of the dual variables should be negative when first solved. The need to be reversed in sign for the signum=-1 case.

$$\text{Minimize : } y = -5x_1^2 + x_2^2x_3^4$$

$$\text{Subject to : } \frac{2}{3}x_2x_3^{-1} + \frac{5}{3}x_1^2x_2^{-1}x_3^{-1} \leq 1$$

Reference Solution :  $y^* = -0.795495$   $x_1^* = 0.4885$   $x_2^* = 0.446$   
 $x_3^* = 1.189$

Solve 0 d.d. function

Zero DD Solve # vars=3 # terms=4 DD=0

```
-1   1   0   0 = 1
-2   0   0   2 = 0
-0   2   1  -1 = 0
-0   4  -1  -1 = 0
```

Successful Gauss Jordan with sig0=1 The deltas are

```
-1.5  -0.5  -0.5  -1.5
```

All deltas negative. Switch signs and set sig0=-1.

TEC= -0.79554

```
2   0   0 = -1.432707 Log of Rule III for term 0
0   2   4 = -0.921882 Log of Rule III for term 1
0   1  -1 = -0.980729 Log of Rule IV for term 2
2  -1  -1 = -0.798504 Log of Rule IV for term 3
```

Successful Gauss Jordan The X's are (\*=Out of bounds values)

x1=0.48853 x2=0.445987 x3=1.189179

GP Algebra Assistant V1.00 9/23/96 . Please use HELP for more information

```
obj : y = -5x1^2 + x2^2x3^4          Z = -0.7955
c1 : 0.6666x2x3^-1 + 1.667x1^2x2^-1x3^-1 <= 1    1<=1    True
```

```
# Vars= 3   x1 = 0.48853   x2 = 0.445987   x3 = 1.189179
Lower Bnd  x1 > 1e-300    x2 > 1e-300    x3 > 1e-300
Upper Bnd  x1 < 1e+300    x2 < 1e+300    x3 < 1e+300
G.P. Form = O.K.   # Terms =4   D.D. =0
```

File : gp1.alg      Reference : None - Example solved by hand.

$$\text{Minimize : } z = x_1^2 + 2x_2^3x_3$$

$$\text{Subject to : } x_1^{-1}x_2^{-1} + x_1x_3^{-2} \leq 1$$

Hand solution :  $z^*=5.119136$   $x_1^*=1.6864$   $x_2^*=0.691807$   $x_3^*=3.43580$

Solve 0 d.d. function

Zero DD Solve # vars=3 # terms=4 DD=0

```

1   1   0   0 = 1
2   0  -1   1 = 0
0   3  -1   0 = 0
0   1   0  -2 = 0

```

Successful Gauss Jordan with sig0=1 The deltas are

0.555556 0.444444 1.333333 0.222222

TEC= 5.119136

```

2   0   0 = 1.045199 Log of Rule III for term 0
0   3   1 = 0.128908 Log of Rule III for term 1
-1  -1   0 = -0.154151 Log of Rule IV for term 2
1   0  -2 = -1.94591 Log of Rule IV for term 3

```

Successful Gauss Jordan The X's are (\*=Out of bounds values)

x1=1.686406 x2=0.691807 x3=3.435817

GP Algebra Assistant V1.00 9/23/96 . Please use HELP for more information

obj : z = x1^2 + 2x2^3x3

Z = 5.119

c1 : x1^-1x2^-1 + x1x3^-2 <= 1

1<=1

True

# Vars= 3    x1 = 1.686406    x2 = 0.691807    x3 = 3.435817

Lower Bnd    x1 > 1e-300    x2 > 1e-300    x3 > 1e-300

Upper Bnd    x1 < 1e+300    x2 < 1e+300    x3 < 1e+300

G.P. Form = O.K.    # Terms =4    D.D. =0

File : gp26.alg      Reference : Woolsey (1992) pg 2-6. Note reference solution is only a close bound. This is rat2.alg with the last coefficient changed to 90 from 9000.

$$\text{Minimize : } t = 1000p + 4 \cdot 10^9 p^{-1} r^{-1} + 250000r + 90pr$$

Reference Solution :    Upper Bound  $t=3,360,000$   $p=1000$   $r=4$   
                          Lower Bound  $t=3,000,000$

```
MULTICON (Ratliff's method)
1000  1 0
4e+09  -1 -1
250000  0 1
90  1 1
 1000.00  1.00  0.00
4000000000.00  -1.00  -1.00
250000.00  0.00  1.00
  90.00  1.00  1.00
nterms=4 sizeof=12 j=12
 1 Tec=3793966 var= 1915.65 2.22334
 2 Tec=3393935.5 var= 1224.39 3.01163
 3 Tec=3335887.9 var= 1022.23 3.38254
 4 Tec=3327231.8 var= 952.135 3.53761
 5 Tec=3325930.6 var= 926.091 3.5996
 6 Tec=3325734.4 var= 916.147 3.62395
 7 Tec=3325704.8 var= 912.308 3.63345
 8 Tec=3325700.3 var= 910.821 3.63715
 9 Tec=3325699.7 var= 910.243 3.63859
10 Tec=3325699.6 var= 910.019 3.63915
11 Tec=3325699.6 var= 909.932 3.63937
Objective Function* = 3.3257e+06 p=909.931854 r=3.639368
in 44 iterations of POSCON.
```

File : Parse.alg      Reference : File to test parsing of complex expressions and to test simplification of same.

$$t = x + x^2 + 2x^2$$

$$2x + x^{-3} \leq x$$

$$2y^2(x_1 + 2y^2)^2 \leq 2(x_1 + y^2)(x_1 + y^2)$$

$$2y^2(x_1^2 + 2y^2)^2 \leq (x^2)^2 + (2x_1 + y^2)^2$$

$$2((xy + ((x^2 + y^{1.5})^2)^3 + 4x^{0.5}y^{0.5})^{0.25})^{1.5} \leq 0.5987$$

$$3x_1y + x_1^2y^{-2}(2 + x_1^3 + 2y^{1.5} + 3x_1^3 + 3)x_1^3y^3 + 2x_1y \leq 1$$

GP Algebra Assistant V1.00 9/23/96 . Please use HELP for more information

```

obj : t = x + x^2 + 2x^2                      Z = 0.0103
c1 : 2x + x^-3 <= x                          1e+06<=0.01      False
c2 : 2y^2(x1^2 + y) <= 2(x1 + y^2)(x1 + y)    2.02e-06<=0.000 True
c3 : 2(x1 + 2y^2)^2 + 2(x^2)^2 <= (x^2)^2 + (2x1 + y^2)^2 0.0002081<=0.00 True
c4 : 2((xy + ((x^2 + y^1.5)^2)^3 + 4x^0.5y^0.5)^0.25)^1.5 <=1 0.5987<=1      True
c5 : 3x1y                                     0.0005<=1      True
      + x1^2y^-2(2 + x1^3 + 2y^1.5 + 3x1^3 + 3)x1^3y^3 + 2x1y <= 1

# Vars= 3   x =      0.01   x1 =      0.01   y =      0.01
Lower Bnd  x >    1e-300   x1 >    1e-300   y >    1e-300
Upper Bnd  x <    1e+300   x1 <    1e+300   y <    1e+300
G.P. Form = Not O.K.

```

c5> simp

GP Algebra Assistant V1.00 9/23/96 . Please use HELP for more information

```

obj : t = x + 3x^2                      Z = 0.0103
c1 : 2x + x^-3 <= x                      1e+06<=0.01      False
c2 : 2y^2x1^2 + 2y^3 <= 2x1^2 + 2x1y + 2y^2x1 + 2y^3 2.02e-06<=0.000 True
c3 : 2x^4 + 2x1^2 + 8x1y^2 + 8y^4 <= x^4 + 4x1^2      0.0002081<=0.00 True
      + 4x1y^2 + y^4
c4 : 2((xy + 4x^0.5y^0.5 + x^12 + 6x^10y^1.5          0.5987<=1      True
      + 15x^8y^3 + 20x^6y^4.5 + 15x^4y^6 + 6x^2y^7.5 + y^9)^0.25)^1.5 <= 1
c5 : 5x1y + 5x1^5y + 4x1^8y + 2x1^5y^2.5 <= 1        0.0005<=1      True

# Vars= 3   x =      0.01   x1 =      0.01   y =      0.01
Lower Bnd  x >    1e-300   x1 >    1e-300   y >    1e-300
Upper Bnd  x <    1e+300   x1 <    1e+300   y <    1e+300
G.P. Form = Not O.K.

```

File : poscon.alg Reference : Test of Poscon answer confirmed with Theorist.

$$\text{Minimize : } t = 1.2q^3 + 3.2q^{-2} + 102.3q$$

Theorist solution :  $t^* = 60.99$   $q^* = 0.39625$

POSCON function

1.2 3

3.2 -2

102.3 1

Variable\* = 0.396251 Objective Function\* = 60.991375 in 6 iterations.

File : rat1.alg Reference : Ratliff (1986), pg. 36. Taylor Mining  
Problem

$$\text{Minimize } c = 70.0035HL + 2333.33L^{-1} + 3333.33H^{-1} + 83333.33H^{-1}L^{-1}$$

Reference Solution :  $C^*=5758$   $H^*=7.357$   $L^*=5.150$

```

11 Tec=5779.997 var= 9.22539 4.19526
12 Tec=5773.3237 var= 8.88744 4.33838
13 Tec=5768.6768 var= 8.61529 4.46175
14 Tec=5765.4364 var= 8.39467 4.5676
15 Tec=5763.1744 var= 8.21484 4.65806
16 Tec=5761.5942 var= 8.06762 4.73511
17 Tec=5760.4899 var= 7.94665 4.80055
18 Tec=5759.7178 var= 7.84694 4.85599
19 Tec=5759.1778 var= 7.76457 4.90287
20 Tec=5758.8002 var= 7.69636 4.94244
21 Tec=5758.536 var= 7.6398 4.97578
22 Tec=5758.3512 var= 7.59282 5.00386
23 Tec=5758.2219 var= 7.55376 5.02746
24 Tec=5758.1315 var= 7.52125 5.0473
25 Tec=5758.0682 var= 7.49416 5.06395
26 Tec=5758.0239 var= 7.47159 5.07793
27 Tec=5757.9929 var= 7.45276 5.08965
28 Tec=5757.9713 var= 7.43704 5.09947
29 Tec=5757.9561 var= 7.42393 5.10771
30 Tec=5757.9455 var= 7.41297 5.1146
31 Tec=5757.938 var= 7.40382 5.12038
32 Tec=5757.9329 var= 7.39618 5.12522
33 Tec=5757.9292 var= 7.38979 5.12927
34 Tec=5757.9267 var= 7.38445 5.13266
35 Tec=5757.9249 var= 7.37999 5.1355
36 Tec=5757.9236 var= 7.37626 5.13787
37 Tec=5757.9228 var= 7.37314 5.13986
38 Tec=5757.9222 var= 7.37053 5.14152
39 Tec=5757.9217 var= 7.36835 5.14291
40 Tec=5757.9214 var= 7.36653 5.14408
41 Tec=5757.9212 var= 7.365 5.14505
42 Tec=5757.9211 var= 7.36373 5.14587
43 Tec=5757.921 var= 7.36266 5.14655
44 Tec=5757.9209 var= 7.36177 5.14712
45 Tec=5757.9209 var= 7.36102 5.1476
Objective Function* = 5757.920862 h=7.36102 l=5.147598
in 180 iterations of POSCON.

```

File : rat2.alg Reference : Ratliff (1986), pg. 37. Chemical Plant problem originally proposed by Theil (1972) appears in Beightler and Phillips (1976). This is GP26.alg with the last coefficient changed from 90 to 9000.

$$\text{Minimize : } t = 1000p + 4 \cdot 10^9 p^{-1} r^{-1} + 250000r + 9000pr$$

Reference Solution :  $y^* = 12,809,668$   $x_1^* = 401.565$   $x_2^* = 1.60557$  in 234 iterations.

```

57 Tec=12809719 var= 406.033 1.58846
58 Tec=12809712 var= 405.742 1.58957
59 Tec=12809707 var= 405.469 1.5906
60 Tec=12809702 var= 405.214 1.59157
61 Tec=12809698 var= 404.975 1.59248
62 Tec=12809694 var= 404.751 1.59333
63 Tec=12809691 var= 404.542 1.59413
64 Tec=12809688 var= 404.346 1.59488
65 Tec=12809686 var= 404.163 1.59558
66 Tec=12809684 var= 403.991 1.59624
67 Tec=12809682 var= 403.831 1.59685
68 Tec=12809680 var= 403.681 1.59742
69 Tec=12809679 var= 403.54 1.59796
70 Tec=12809677 var= 403.408 1.59847
71 Tec=12809676 var= 403.285 1.59894
72 Tec=12809675 var= 403.17 1.59938
73 Tec=12809674 var= 403.062 1.5998
74 Tec=12809674 var= 402.961 1.60019
75 Tec=12809673 var= 402.866 1.60055
76 Tec=12809672 var= 402.778 1.60089
77 Tec=12809672 var= 402.695 1.60121
78 Tec=12809671 var= 402.617 1.60151
79 Tec=12809671 var= 402.544 1.60179
80 Tec=12809671 var= 402.476 1.60205
81 Tec=12809670 var= 402.413 1.6023
82 Tec=12809670 var= 402.353 1.60253
83 Tec=12809670 var= 402.297 1.60274
84 Tec=12809670 var= 402.245 1.60294
85 Tec=12809670 var= 402.196 1.60313
86 Tec=12809669 var= 402.15 1.60331
87 Tec=12809669 var= 402.107 1.60347
88 Tec=12809669 var= 402.067 1.60363
Objective Function* = 1.280967e+07 x1=402.067134 x2=1.603628
in 352 iterations of POSCON.

```

File : rat3.alg Reference : Ratliff (1986), pg. 39. Batch Size problem  
of Schweyer (1955)

$$\text{Minimize } c = 10q^{1.2}p^{-1} + 600q^{-1} + 0.000001p$$

Reference Solution :  $c^*=0.90109$     $q^*=1770.1878$     $p^*=279687$

```
MULTICON (Ratliff's method)
MULTICON warning: Zero d.d. problem.
10  -1 1.2
600  0 -1
1e-06  1 0
   10.00  -1.00  1.20
   600.00  0.00  -1.00
     0.00  1.00  0.00
nterms=3 sizeof=12 j=12
 1 Tec=4.7699356 var= 3162.28 230.764
 2 Tec=1.1635224 var= 82775.3 1017.81
 3 Tec=0.9226779 var= 201651 1525.6
 4 Tec=0.90275182 var= 257078 1703.65
 5 Tec=0.90120654 var= 274680 1755.71
 6 Tec=0.90109017 var= 279687 1770.19
 7 Tec=0.90108149 var= 281068 1774.16
 8 Tec=0.90108084 var= 281446 1775.24
 9 Tec=0.90108079 var= 281549 1775.53
10 Tec=0.90108079 var= 281577 1775.62
Objective Function* = 0.901081 p=281577.194295 q=1775.615205
in 40 iterations of POSCON.
```

File : rat4.alg Reference : Ratliff (1986), pg. 41. Ammonia  
Refrigeration Problem proposed by Sherwood (1970)

$$\text{Minimize : } t = a + g + g^{2.8}n^{-1.8} + a^{-1} + a^{-1}g^{-0.8}n^{0.8} + g^{-1}$$

Reference Solution :  $t^*=5.2223$   $a^*=1.54264$   $g^*=0.82016$   $n^*=1.22642$

MULTICON (Ratliff's method)

```

1  1  0  0
1  0  1  0
1  0  2.8 -1.8
1  -1  0  0
1  -1 -0.8 0.8
1  0 -1  0
    1.00    1.00    0.00    0.00
    1.00    0.00    1.00    0.00
    1.00    0.00    2.80   -1.80
    1.00   -1.00    0.00    0.00
    1.00   -1.00   -0.80    0.80
    1.00    0.00   -1.00    0.00

```

nterms=6 sizeof=12 j=12

```

1 Tec=5.5382866 var= 1.41421 0.755376 1.0584
2 Tec=5.5269002 var= 1.51979 0.768919 1.11522
3 Tec=5.5241751 var= 1.53181 0.786633 1.15444
4 Tec=5.523026 var= 1.53596 0.798755 1.18037
5 Tec=5.5225551 var= 1.53842 0.806642 1.19728
6 Tec=5.522363 var= 1.53998 0.811725 1.20821
7 Tec=5.5222848 var= 1.54098 0.814988 1.21524
8 Tec=5.5222529 var= 1.54162 0.817078 1.21976
9 Tec=5.5222399 var= 1.54203 0.818415 1.22264
10 Tec=5.5222346 var= 1.54229 0.819269 1.22449
11 Tec=5.5222325 var= 1.54246 0.819814 1.22567
12 Tec=5.5222316 var= 1.54257 0.820162 1.22642
13 Tec=5.5222313 var= 1.54264 0.820384 1.2269
14 Tec=5.5222311 var= 1.54268 0.820526 1.22721
15 Tec=5.522231 var= 1.54271 0.820616 1.22741
16 Tec=5.522231 var= 1.54272 0.820674 1.22753
Objective Function* = 5.522231 a=1.542725 g=0.820674 n=1.227532
in 255 iterations of POSCON.

```

File : redundant Reference : None. Constructed to test the redundant constraint portion of the Kirk algorithm. Reduces to zero DD problem involving only the first constraint. Last two constraints give upper bounds on a and b. The 4th constraint must be satisfied at those bounds. Constraints 2 and 3 give lower bounds on a and b at 0.01. Remaining problem is zero DD.

$$\begin{aligned} \text{Minimize : } z &= 2a^2b^2 \\ \text{Subject to : } 0.01a^{-1} + 0.01b^{-2} &\leq 1 \\ 0.01a^{-1} &\leq 1 \\ 0.01b^{-1} &\leq 1 \\ a + b &\leq 1 \\ 2b &\leq 1 \\ 4a &\leq 1 \end{aligned}$$

GP Algebra Assistant V1.00 9/23/96 . Please use HELP for more information

```

obj : z = 2a^2b^2                Z = 2e-08
c1 : 0.01a^-1 + 0.01b^-2 <= 1    101<=1      False
c2 : 0.01a^-1 <= 1               1<=1        True
c3 : 0.01b^-1 <= 1               1<=1        True
c4 : a + b <= 1                   0.02<=1    True
c5 : 2b <= 1                      0.02<=1    True
c6 : 4a <= 1                      0.04<=1    True

# Vars= 2   a =      0.01   b =      0.01
Lower Bnd  a >  1e-300   b >  1e-300
Upper Bnd  a <  1e+300   b <  1e+300
G.P. Form = O.K.   # Terms =9   D.D. =6

```

c6> kirk

```

obj : z = 2a^2b^2                Z = 2.261e-06
c1 ! 0.01a^-1 + 0.01b^-2 <= 1    1.92<=1     False

# Vars= 2   a = 0.010417   b = 0.102062
Lower Bnd  a > 0.010417   b > 0.102062

```

